Tesis Doctoral

Distribución de la Computación y Evaluación de la QoS de aplicaciones IoT en el Cloud-to-Thing continuum

Sergio Laso Mangas

Programa de Doctorado en Tecnologías Informáticas (TIN)

Con la conformidad de los directores
Dr. José Javier Berrocal Olmeda y *Dr. Juan Manuel Murillo Rodríguez*

Esta tesis cuenta con la autorización del director/a y coodirector/a de la misma y de la Comisión Académica del programa. Dichas autorizaciones constan en el Servicio de la Escuela Internacional de Doctorado de la Universidad de Extremadura

2023

A mi padre, que no ha podido estar acompañándome durante los últimos años pero que gracias a sus enseñanzas, principios y valores he llegado hasta aquí.

A mi madre, estando siempre al pie del cañón.

A mis hermanas, que siempre están en las buenas y en las malas.

A mi primera sobrina Victoria por traer la alegría a la casa.

Acknowledgments

First of all, I wanted to express my gratitude to my directors Javier Berrocal and Juan Manuel Murillo, for the trust they have placed in me and for giving me the opportunity to develop my research career. It has been a few years since 2018 when we started our journey together. Your advice and help have been indispensable to make the path much more straightforward.

To my colleagues throughout these years in the SPILab: Dani, Sheila, Sara Arroyo, Quique, Borja, Juanlu, Manuel, Javi, Rome, Jaime, Luis, Sara Chimento, Isa, and Joserra. The team, the people, and the work environment we have had is what I take the most from all this. Of course also all the advice and help you have given me and above all, having put up with me all these years.

To the members of the Quercus research group, Jose Manuel García, Silvia Pérez, Juan Hernández, Jaime Galán, and Pedro J. Clemente for helping me in one way or another. I would also like to thank Pablo Fernández and Antonio Ruiz Cortés from the University of Seville and Schahram Dustdar from the TU Wien for welcoming me with open arms for the research stays. In addition, I would also like to thank all my friends in Vienna and Sevilla for making me feel so well far away from home. For the following paragraphs, I switch to my native language.

Quiero agradecer con el corazón a todas esas personas especiales que tengo la gran suerte de tener a mi lado: a mi amiga Victoria, un pilar indispensable para mí, por todo el cariño, el apoyo, y todos los valores que me enseña. A mi grupo de amigas y amigos *Autumn* (lo siento pero me da pereza poneros a todos), mi segunda familia, siembre recibiendo un gran apoyo por parte de todos ell@s.

Por último y más importante, a toda mi familia. De nuevo agradecer a mi madre, mis hermanas y mi sobrina, a mis tías y tíos, a mis primas y primos, a mi abuelo, a los que ya no están, y a todos los que siguen estando. Agradeceros por todo mi trabajo fuera de aquí que os habéis echado a vuestras espaldas, y que me habéis acompañado día a día durante este largo y duro camino. Gran parte de este trabajo es gracias también a vuestro esfuerzo.

Resumen

La gran popularidad y acogida que están teniendo los dispositivos inteligentes ha fomentado el desarrollo de aplicaciones centradas en la Internet de las Cosas (IoT). La mayoría de estas aplicaciones requieren detectar casi constantemente cambios en el entorno, procesar los datos recogidos y realizar algunas acciones para adaptar el entorno a las necesidades y preferencias de los usuarios, gestión eficiente de los sistemas, monitorización de la salud de las personas, etc. Por lo que necesitan unos requisitos de calidad estrictos para que sean aceptados por ellos.

Actualmente, el desarrollo de estas aplicaciones está principalmente determinado por el paradigma Cloud Computing. Toda la información se envía y centraliza en servidores cloud para almacenarse y procesarse. No obstante, debido al aumento de la cantidad de estos dispositivos conectados a Internet que se esperan durante los próximos años junto con los estrictos requisitos de las aplicaciones IoT, se estima que puede conllevar a una saturación de la infraestructura debido al volumen de información que tienen que intercambiar. Esto supondrá una reducción afectando de la calidad de servicio (QoS), tiempos de respuesta, etc.

Para intentar mitigar esta situación, distintos investigadores están trabajando en la definición de paradigmas como, Fog, Edge y Mist Computing. Estos paradigmas permiten explotar las capacidades de diferentes dispositivos y nodos conectados a Internet a lo largo de toda la infraestructura para almacenar, procesar y proporcionar servicios. Esta distribución de la computación a lo largo de la infraestructura se le ha denominado Cloud-to-Thing continuum con el objetivo de reducir los tiempos de respuesta, mejorar la QoS y generar menos tráfico de datos global en la red. Sin embargo, el desarrollo de aplicaciones basado en estos paradigmas presenta dificultades. Mientras que plataformas y empresas proporcionan algunos estándares y tecnologías para facilitar el desarrollo de aplicaciones Cloud, existen pocos mecanismos y tecnologías similares para el desarrollo de aplicaciones con estos paradigmas

Por lo tanto, esta tesis se enfoca en investigar y contribuir al desarrollo de aplicaciones utilizando estos paradigmas arquitectónicos novedosos como

son Fog, Edge y Mist. Los objetivos principales son aportar nuevos procesos, técnicas y herramientas que ayuden a facilitar la implementación, evaluación y despliegue de aplicaciones IoT basadas en estos paradigmas, abarcando actividades importantes de las metodologías de desarrollo software. Con esto se ayudaría a que las empresas puedan introducir estos nuevos paradigmas arquitectónicos en el desarrollo de aplicaciones IoT de forma viable.

Abstract

The great popularity and acceptance of smart devices have encouraged the development of applications focused on the Internet of Things (IoT). Most of these applications require almost constant detection of changes in the environment, processing the collected data and performing some actions to adapt the environment to users' needs and preferences, efficient management of systems, monitoring people's health, etc. So they need strict quality requirements to be accepted by them.

Currently, the development of these applications is mainly determined by the Cloud Computing paradigm. All information is sent and centralized in cloud servers to be stored and processed. However, due to the increase in the number of these Internet-connected devices expected over the next few years along with the strict requirements of IoT applications, it is estimated that this may lead to a saturation of the infrastructure due to the volume of information they have to exchange. This will result in a reduction affecting the quality of service (QoS), response times, etc.

To try to mitigate this situation, different researchers are working on the definition of paradigms such as Fog, Edge, and Mist Computing. These paradigms make it possible to exploit the capabilities of different devices and nodes connected to the Internet throughout the infrastructure to store, process, and provide services. This distribution of computing across the infrastructure has been called the Cloud-to-Thing continuum with the objective of reducing response times, improving QoS, and generating less overall data traffic in the network. However, application development based on these paradigms presents difficulties. While platforms and companies provide some standards and technologies to facilitate the development of Cloud applications, there are few similar mechanisms and technologies for the development of applications with these paradigms.

Therefore, this thesis focuses on investigating and contributing to application development using these novel architectural paradigms such as Fog, Edge, and Mist. The main objectives are to provide new processes, techniques, and tools that help to facilitate the implementation, evaluation, and

deployment of IoT applications based on these paradigms, covering important activities of software development methodologies. This would help companies to introduce these new architectural paradigms in the development of IoT applications in a viable way.

Contents

List of Figures

List of Tables

Listings

"Science, my lad, is made up of mistakes, but they are mistakes which it is useful to make, because they lead little by little to the truth"

Jules Verne (1828-1905)

Chapter 1

Introduction

In this chapter, we present the context of the thesis, where we explain the motivations, problems and goals detected. We also present all the contributions that have been made in this thesis, such as contributions, publications, etc. We first present the research context in which this thesis is developed in Section 1.1. The problems detected in the PhD research are set out in Section 1.2. In Section 1.3, the hypotheses derived from the problems and the goals are exposed. Section 1.4 details the research methodology followed throughout the PhD. In Section 1.5, we summarise our contributions, publications, etc. Finally, Section 1.6 outlines the structure of the rest of the thesis.

Contents

1.1 Research Context

The capacities of end devices have increased enormously. Their computational and storage capacity have multiplied with the aim of being able to obtain more information from the environment and carrying out increasingly complex operations. This increase in the capacities of these devices has favoured the development of paradigms such as the Internet of Things (IoT), where end devices are more integrated into the Internet [1]. There are currently a large number of these devices in our society (mobile devices, IoT devices such as sensors, actuators, etc.). Moreover, the number of these devices is expected to increase considerably in the coming years [2]. In 2022, the Internet of Things market is expected to grow by 18% to 14.4 billion active connections. In 2025, as supply constraints ease and growth accelerates further, there are expected to be approximately 27 billion connected IoT devices. However, the actual 2021 data as well as the current 2025 forecast for IoT devices is lower than previous estimates, due to various issues such as prolonged supply disruptions due to the pandemic or current inflation [3].

All these connected devices generate a large flow of information, as they require almost constant sensing of the environment, processing the collected data, and performing some actions to adapt the environment to the needs and preferences of the users, efficient management of the systems, monitoring of people's health, etc. Most of these devices generate and consume information through different services deployed in the Cloud [4], thanks to the growing importance of the Cloud Computing paradigm [5, 6], which has changed the way of producing and consuming information. IoT applications are usually based on this paradigm, in which these devices act as simple clients obtaining information through their sensors, sending it to the used cloud, and obtaining from the same environment the actions to be executed or processed information. These environments centralize the most demanding functionalities, such as storage or computation of managed data, providing better scalability, fault tolerance, and greater control oAll these connected devices generate a large flow of information, as they require almost constant sensing of the environment, processing the collected data, and performing some actions to adapt the environment to the needs and preferences of the users, efficient management of the systems, monitoring of people's health, etc. Most of these devices generate and consume information through different services deployed in the Cloud [4], thanks to the growing importance of the Cloud Computing paradigm [5, 6], which has changed the way of producing and consuming information. IoT applications are usually based on this paradigm, in which these devices act as simple clients obtaining information through their sensors, sending it to the used cloud, and obtaining from the same environment the actions to be

executed or processed information. These environments centralize the most demanding functionalities, such as storage or computation of managed data, providing better scalability, fault tolerance, and greater control of operating costs.f operating costs.

Nevertheless, the number of these devices connected to the Internet is expected to increase in the coming years. Therefore, the increase of connected devices together with the strict requirements of IoT applications can lead to network saturation due to the volume of information they have to publish and consume [7]. The impact of network saturation would negatively affect the quality of service (QoS), increasing latency and response times. It will also lead to an increase in operational costs due to the increased volume of data they have to store, etc., making it even more difficult to achieve the QoS of IoT applications [8] to be truly useful and accepted by end users.

To meet IoT application requirements, several lines of research have proposed new architectural paradigms such as Fog Computing, Edge Computing, Mist Computing [9], etc. These are some of the existing paradigms on which this thesis focuses, although others are very similar such as Mobile Cloud Computing [10], Mobile Edge Computing[[11], etc. There are even authors who mix them up or consider them the same, it is one of the things in this area that has yet to be standardized. All of these proposals are within the Cloud-to-thing continuum [12], which are infrastructures that extend beyond centralized data centers, from the cloud to end devices. The goal of these paradigms is to distribute computing and store services in different layers so that developers can exploit the capabilities of nodes closer to the end user (IoT devices, smartphones, fog/edge nodes, etc.), in order to reduce the response time, the network overload, increase the privacy of managed data, etc. In short, to facilitate the achievement of QoS attributes that enable the provision of IoT applications with acceptable quality for all users. Figure 1.1 shows a possible distributed architecture formed by the above-mentioned paradigms where we can see the devices that would act in each layer.

However, there are several drawbacks to developing IoT applications based on these paradigms. The implementation of these IoT applications requires developers skilled in different areas (hardware, communication, operating systems, etc.). Whilst cloud platforms and other companies provide some standards and technologies abstracting from the low-level details to implement, there are few similar mechanisms and technologies for computational infrastructures closer to the end user (smartphones, IoT devices, wearables, etc.) [13]. The use of precarious and ad hoc mechanisms implemented directly by each developer to develop this type of application may affect the development time, maintainability, and reproducibility of the software. Therefore, companies may be reluctant to develop them due to these issues.

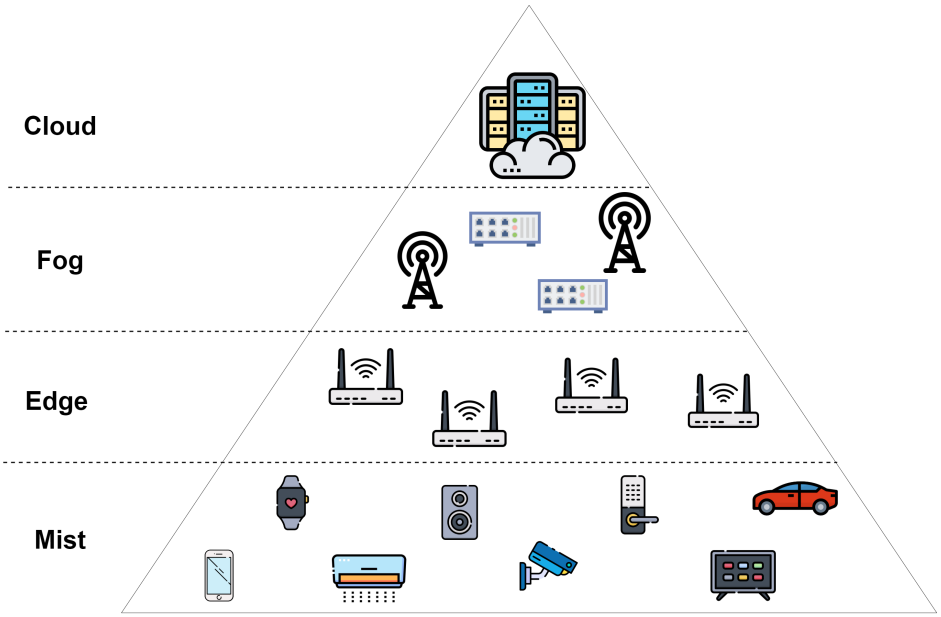


Figure 1.1: Cloud-Fog-Edge-Mist architecture

Another issue that companies may be reluctant to develop is QoS. The evaluation of QoS in applications is crucial for the success/failure of the application. An application that does not meet the expected QoS is more likely to be rejected by users, with the financial and image impact that this may cause to a company. The evaluation of IoT applications using these paradigms is not as trivial as in applications with more traditional architectures. There are more components fog/edge nodes, IoT devices, etc. that make it a more heterogeneous architecture and therefore make the evaluation much more complex [14]. Therefore, new techniques and tools are also needed to evaluate the QoS of applications using these paradigms.

Finally, another aspect to highlight is the selection of an architecture for deployment. When an application is deployed with a particular architecture and topology (following Cloud-Fog-Edge-Mist paradigms) it can provide the goal QoS at a given time and under specific contextual conditions. However, this may not be sufficient in changing environments. All IoT applications are exposed to different changing conditions, e.g. communication quality (intermittent communication due to area, bandwidth, etc.), variation over time in the number of users, the volume of information, or the freshness of the data. Sometimes this can lead to situations where a given deployment architecture complies with the established QoS, but fails to do so when any of these parameters change. Consequently, techniques that allow dynamic deployment of IoT applications throughout the Cloud-to-thing continuum are also needed to provide optimal QoS.

Therefore, the research in this thesis has focused on investigating and contributing to the development of these emerging architectural paradigms such as Fog, Edge, and Mist along the Cloud-to-thing continuum. The main goals are to provide new processes, techniques, and tools to help facilitate the implementation, evaluation, and deployment of IoT applications based on these paradigms, contributing to supporting the activities of development methodologies, such as DevOps [15]. Development methodologies, e.g. DevOps, promote rapid and continuous feedback from operations to development to detect problems before customers are affected. Thus, integrating the contributions into these development methodologies would help companies to introduce these new architectural paradigms into the development of IoT applications in a viable way.

1.2 Problem statement

The previous section sets the context of the thesis research. In this section the problems detected in the initial stage of the research are detailed. In addition, the possible causes and consequences are detailed. This analysis has been useful to clearly focus the whole research and development of the thesis.

- **Problem 1:** Development of applications applying emerging paradigms. Several researchers have proposed new emerging paradigms, such as Fog, Edge or Mist Computing, but they are rarely used by industry.
  - **Causes:**
    1. Lack of tools and/or services to facilitate the development of applications using these emerging paradigms.
    2. Extra effort and cost to develop applications with these emerging paradigms compared to conventional ones due to lack of resources.
  - **Consequences:**
    1. The capabilities of devices and nodes close to the end user are not exploited. The goal of these paradigms is to use devices and nodes close to the end users in order to reduce response times, network overhead, etc.
    2. High cost for companies. If there are no tools/services to facilitate development, developers will have to implement their solutions ad hoc, which could affect development time, maintainability, etc., and therefore higher costs for companies.

- **Problem 2:** QoS evaluation of applications applying emerging paradigms. The success or failure of applications depends to a large extent on the QoS. The QoS evaluation of these paradigms is a complex process.
  - **Causes:**
    1. Lack of tools and/or services that facilitate the evaluation of the QoS that can be provided using these paradigms.
    2. Extra effort and cost to evaluate applications that use these paradigms compared to conventional paradigms like Cloud computing.
  - **Consequences:**
    1. High cost for companies. If there are no tools/services to facilitate QoS evaluation, developers will have to evaluate QoS with precarious mechanisms, which could affect the evaluation time and thus lead to higher costs for companies.
    2. If an application offers a good service with poor quality, it will probably be rejected. Many of these companies are start-ups, so the failure of an application can bankrupt them.
- **Problem 3:** Dynamic deployment. An application is normally designed and deployed with a static topology, and therefore does not have the opportunity to change to another topology that provides better performance at certain times.
  - **Causes:**
    1. Lack of techniques to support dynamic deployments in Cloud-Fog-Edge-Mist environments.
    2. Duplication of effort and cost for companies to design and develop multi-style applications.
  - **Consequences:**
    1. Failure to take advantage of the benefits that different styles can bring can hinder the fulfillment of defined QoS.

1.3 Thesis Goal

In order to state the main goal of the thesis, we have first described different hypotheses taking into account the context of the research and the state of the problem. Subsequently, the main goal and its specific goals are presented.

1.3.1 Hypothesis

1. If tools and processes are provided to support the development of these new emerging architectural styles, then they will save cost and effort for companies and developers and can be used as a viable alternative to more conventional ones.

2. If techniques are provided to evaluate the QoS of IoT applications with these new emerging architectural styles before they are deployed in production, then companies will be able to provide higher quality applications to users, increasing user satisfaction and becoming more competitive.

3. If mechanisms and processes are provided that allow the dynamic deployment of an application to adapt its topology according to different parameters of the context and resources, it will allow companies to provide adequate QoS.

1.3.2 Main Goal

The hypotheses described above allow the following main goal to be set.

Main Goal: Contribute new processes, techniques, and tools to facilitate the implementation, evaluation, and deployment of IoT applications based on these new emerging architectural styles by leveraging the capabilities of end devices (Mist) and near nodes (Edge and Fog), covering important activities of the software development methodologies.

1.3.3 Specific Goals

The main goal is a general goal that involves a number of different research focuses. In order to determine and focus on the different points to be investigated in a clearer and more concise way, we have described specific goals. Each of them is described in the following:

- **SG1.** Providing tools and processes that speed up the implementation of emerging architectural styles such as Edge, Fog, and Mist Computing.

- **SG2.** Providing tools and processes that facilitate the evaluation of the QoS of IoT applications that apply architectural styles making use of paradigms such as Edge, Fog, and Mist Computing.

- **SG3.** Provide mechanisms so that an application can dynamically change the architecture topology to adapt to the context needs or resources required at the time.

1.4 Research Methodology

This thesis aims to address a general goal made up of three specific sub goals. To meet these sub goals, it is very important to carry out a plan with the different activities that must be carried out for the proper development of the work. The methodology followed in this work is detailed below.

For the planning and development of this project, it has been decided to adopt the Design Science methodology [16]. Design Science is a methodology that offers specific guidelines for research projects. It is especially used in the disciplines of Engineering and Computer Science, although it is applicable to other areas [17].

The main objective of this methodology is to obtain knowledge of a certain problem and its domain and the contribution of innovative products that minimize the problem [18]. This increase in knowledge and the creation of innovative products is achieved through the creation of artifacts. An artifact is an object made by people with the intention of being used to solve a practical problem. Artifacts can be sketches, pieces of software, hardware, etc. [19]. During the thesis, several artifacts/frameworks [20, 21] are presented that cover part of the goals proposed in the thesis

Design Science is based on approaching a problem, together with its causes and consequences, as detailed for this thesis in Section 1.2, and trying to solve them. Fig. 1.2 shows the different activities to be carried out in this research methodology.

Figure 1.2: Iterative structure of the Design Science methodology

- *Problem Explain:* The goal of this activity is to investigate and analyze the initial problem identified as input. The problem must be formulated in a precise and justified manner, highlighting the causes leading to the problem that must be analyzed. In our case, we have identified three

main problems through a literature review for the deployment of IoT applications along the Cloud-to-thing continuum.

- *Define Requirements:* Sketch the problem solution together with the artifact requirements, which can be business, architectural, functional and non-functional requirements that will be addressed by the artifacts. To sketch out our solutions we looked at different standards and existing technologies to try to adapt them to our solution and reduce the learning curve for the end developers using them.

- *Design and Develop Artifacts:* Creation of the artifact that addresses the problem and fulfills the defined objectives. In this phase the different artifacts and solutions have been developed for the context of this thesis, covering each of the problems or specific goals.

- *Demonstrate Artifacts:* Demonstration of the viability of the developed artifacts in proofs of concept, case studies, etc. to help justify that the solution delivers the expected results. All our contributions have linked case studies to justify the solution such as case studies related to Covid, smart cities, rural environments, etc. to better show the problems and solutions.

- *Evaluate Artifacts:* The evaluation of the artifact is the detailed valuation of the artifact and is based on the results of the demonstration. The evaluation has to demonstrate that the artefact meets the requirements and solves the problem. In our work, we have tried to deploy them in scenarios with characteristics as close to reality as possible in order to obtain the most realistic results possible.

The above activities are not followed strictly sequentially. Normally, and more in research, these tasks are iterative. The arrows between activities only show the expected inputs and outputs of each activity. Therefore, some authors establish a three-cycle view of this methodology [22]. This view is shown in Fig. 1.3.

- *Relevance Cycle*, initiates research by analyzing the environment and the problems to be addressed, defining the requirements of the artifacts and the acceptance criteria for the evaluation of the research results. It also identifies whether new iterations are necessary to solve the problem.

- *Design Cycle*, focuses on the construction and evaluation of the developed artifact. For this purpose, the requirements and criteria identified in the Relevance cycle are followed, and the methods and techniques identified in the Rigor cycle are applied.

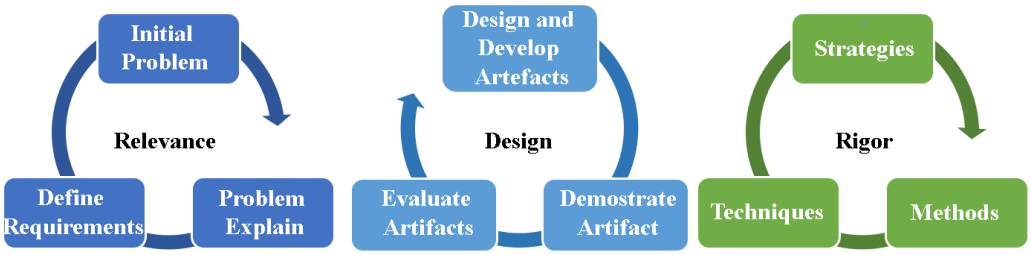


Figure 1.3: Design Science methodology proposed model

- *Rigor Cycle*, provides knowledge to the project to ensure that it is innovative, ensuring that the research produces contributions to the general knowledge base.

This methodology not only produces artifacts but also creates knowledge of general interest so it presents three important requirements: first, the use of rigorous research methods for the literature review and to define the problems and goals; second, the results must be correctly grounded and original, in this thesis through different case studies; third, the results must be communicated. All the results obtained from this thesis have been submitted and published in different journals, conferences, etc. for the dissemination of knowledge.

1.5 Contributions

1.5.1 Summary of Contributions

Along the PhD, we have made different contributions that try to cover the proposed goal *Contribute new processes, techniques, and tools to facilitate the implementation, evaluation, and deployment of IoT applications based on these new emerging architectural styles by leveraging the capabilities of end devices (Mist) and near nodes (Edge and Fog), covering important activities of the software development methodologies"*. The contributions cover different activities of the software development methodologies (implementation, evaluation, and deployment). In our work, we have focused on following the DevOps methodology and its Life Cycle [15]. Therefore, in the following sections, we classify our contributions by each activity covered within the DevOps methodology, **Code**, **Test**, and **Operate**, thus addressing the main goal of this thesis.

CODE

In this stage, the application code is developed using appropriate programming languages and IDEs. The contributions presented in this stage aim to provide tools, frameworks, standards, etc. that enable the development of IoT applications along the entire Cloud-to-thing continuum (Cloud, Fog, Edge, and Mist). These contributions are focused on covering SG1.

Human Microservices

- *Problem statement:* Over the last decade, the mobile application market has grown steadily thanks to the massive use of smartphones [23] and the emergence of cloud computing to offload computational tasks and improve the quality of experience. With the more recent deployment of IoT devices, this cloud-based architectural design and corresponding communication flow have been maintained [1]. However, the increasing amount of information being exchanged, the stringent requirements of many IoT applications, and the need for these applications to adapt their real-time behaviour to the user's context make these architectural assumptions challenging. Recently, paradigms such as Mist, or Edge computing [9, 11] have been proposed to exploit the computational and storage capabilities of today's smartphones and IoT devices to overload some tasks on them, reducing the overhead on both the cloud and the network. However, few mechanisms and technologies exist to develop applications following these paradigms. Developers use their mechanisms and ad hoc implementations to develop such applications which can affect the development time, maintainability, and reproducibility of the software.

- *Contribution:* In this work we present Human Microservices as a framework that facilitates the implementation of APIs on end devices following the Mist Computing paradigm to provide personal and up-to-date information that can be consumed by other entities. The framework facilitates the implementation of computing units on devices closer to end users, improving system response time by reducing the strain on cloud and network infrastructure. The proposed framework builds on existing standards to improve software quality (maintainability and reproducibility) and shorten the learning curve. The main results of this contribution have been presented at the *International Workshop on Gerontechnology (2019)*[24] and *2020 IEEE International Conference on Pervasive Computing and Communications (PERCOM'20)* [25, 26], and published in the journal *Software: Practice and Experience (2021)* [13].

SOLID in Smartphones

- *Problem statement:* Privatization and centralization of information are common practices in applications and companies, that intend to provide personalized services to their users based on their preferences and habits [27]. As a consequence, these policies result in potential privacy issues, inconsistencies, duplication, and insecurity problems [28]. This privatization trend leads sites to form information silos so that data storage is done independently and autonomously, with a large pool of duplicate data between services [29]. Therefore, access to other applications is disabled and users do not enjoy much control over their data. There are proposals that attempt to offer alternative models of storing and providing data from Mist devices to be computed using the Cloud continuum. In this way, users store their information and have control over it. The Social Linked Data (SOLID) [30] initiative becomes the most relevant among these solutions. SOLID proposes the decentralization of information to separate applications and personal data. However, existing implementations exploit SOLID, they do not draw a global mechanism for data storage.

- *Contribution:* In this contribution, we propose the implementation of the SOLID approach in smartphones. This combination takes advantage of the ubiquitous and contextual characteristics of these devices closer to end users to provide a new model for storing personal data and having more control over their data avoiding duplication in different services and applications. Smartphones are suitable devices for storing data, as most of the information involved in application processes is obtained from them. Thus, external applications request access that the user can authorize or deny. As a result, the user can know which applications consume information and when, while businesses benefit from a consistent source of data. The main results of these contribution has been accepted for publication at the *International Conference on Web Engineering (2020)*[31], and published in the journal *Mobile Information Systems (2021)* [32].

Distributed API Composer

- *Problem statement:* There are artifacts such as the one presented above (APIGEND) that allow the implementation of APIs in different end devices following the Mist Computing paradigm [13]. These devices become service providers and can be consumed directly by other devices connected to the Internet as if they were deployed in the cloud through

an API. The main problem with this distributed deployment of APIs is how to collect data from each device. For example, applications that involve a large number of individuals collectively collect and extract data from specific areas of interest [33]. Each user is independent of the others, but they have the option to share the collected data to collaborate toward a common goal. However, collecting all this data requires accessing all these devices one by one, and once collected, it is post-processed to finally arrive at the goal result. The data management process should require as little effort as possible for developers. In order to try to solve this problem, a system would be needed that, in a coordinated way, obtains the data from the APIs of the devices and aggregates them to obtain the final result. Currently, to develop such a system, developers have to use all their skills and knowledge to design and implement their ad hoc solutions. This leads, firstly, to an increase in development time and cost and, secondly, to a reduction in maintainability. Therefore, mechanisms and tools are needed to facilitate the development of such a system.

- *Contribution:* Our proposal, called Distributed API Composer (DAC), performs the whole process of coordinating the invocations, obtaining the data, and aggregating it from the distributed APIs in end devices. DAC aggregation is not only based on retrieving and exposing the raw data but also on performing some processing to expose it as adapted as possible to the final result and to improve compliance with the QoS requirements. To facilitate the aggregation of the data, different operations have been defined to process the data and can be reused between the different endpoints. In addition, it also allows defining conditions in the communication with the final APIs in order to offer an acceptable QoS. For this purpose, we have designed a conceptual model to capture all its concepts and characteristics. This model has helped us to develop an extension of the OpenAPI [34] language to make it easier for developers to implement it for the intermediate Edge and Fog layers, as well as in the Cloud. We also extended the APIGEND tool to support the DAC and facilitate its implementation. The main results of this contribution have been presented at the *Jornadas de Ciencia e Ingeniería de Servicios (JCIS), SISTEDES (2021)* [35].

TEST

Once the code build is ready, at this stage it is first deployed in the test environment to perform various types of tests such as user acceptance testing, security testing, integration testing, performance testing, end-to-end testing,

etc. This ensures that potential bugs in the developed software are eliminated and a reliable product is delivered to production. The contribution presented to cover this stage aims to provide a framework for QoS evaluation in applications deployed in the Cloud-to-thing continuum, as their evaluation is more complex due to the different components distributed throughout the application.

Perses

- *Problem statement:* The growth of the app market has generated a multitude and variety of apps. There are different dimensions, such as advertising, originality, and virality, that determine the success of an app. Success also depends on end user satisfaction by offering a good Quality of Experience (QoE) [36]. QoE also depends on the QoS offered by the application as a whole and the infrastructure supporting it, especially in the case of distributed applications where computing is not primarily performed in a single location (cloud environment), but on a set of distributed devices. These problems can be avoided by assessing the QoS of the entire system before deployment. To assess the required quality, developers can evaluate each side (back end and front end) independently, but E2E testing is also performed. E2E testing evaluates the correct integration and the correct functioning of the main functionalities, replicating the behaviour of the [37] users. End-to-end evaluation in such architectures is not as trivial as in more traditional architectures. QoS evaluation with E2E testing of distributed applications is complex to evaluate. With these new paradigms, it is necessary to deploy a considerable set of heterogeneous devices close to the real scenarios to properly evaluate QoS: latency and execution time can vary greatly depending on the devices involved. Therefore, new techniques are needed to assess the QoS of distributed applications by simulating a scenario close to reality.

- *Contribution:* We developed our framework called **Perses**. Perses allows to performance of QoS evaluations in distributed applications simulating virtual scenarios with heterogeneous devices. It can also be integrated into a software development process, such as DevOps, which allows automating the tasks of creating and deploying the virtual scenario, launching E2E tests, collecting results, etc., which reduces the effort required to perform these tests and it can be widely adopted by companies, saving the effort and cost required to perform these tests. The main results of this contribution has been accepted for publication at the *2021 IEEE International Conference on Pervasive Computing and*

Communications (PERCOM'21) [38, 39], the *Jornadas de Ciencia e Ingeniería de Servicios (JCIS), SISTEDES (2021)* [40] and published in the journal *Pervasive and Mobile Computing (2022)* [14].

OPERATE

At this stage, the release is ready to be used by customers. At this stage, the operations team is responsible for implementing the configuration and provisioning the infrastructure to deploy it. The inputs presented here are aimed at implementing the applications so that the deployment of your cloud-continuum infrastructure can be adapted dynamically to the needs of the application context (e.g. communication quality, transmission and data quality requirements, etc.).

OPPNets

- *Problem statement:* Rural areas are filled with business activities such as livestock farming and agriculture. There are also other important activities such as healthcare. The demographics in rural areas are commonly characterized by a significant percentage of people over 65 years old. However, they are often isolated by network operators because they are not interested in deploying their solutions due to the low benefits they acquire. Today, technological solutions for deploying applications across the Cloud-to-thing continuum for industry and healthcare have become the main disruptive solutions for improving people's production and quality of life [41]. However, in rural areas where there is no access to the Internet, these solutions cannot be easily deployed. Specific needs and technological isolation motivate research into alternative technologies that explore mechanisms for information transmission and communication systems for the proper deployment of cloud-continuum infrastructures in these rural environments.

- *Contribution:* In this contribution, we propose a solution based on an opportunistic network algorithm that enables the deployment of Cloud-to-thing continuum communication technology solutions in isolated areas where connectivity is poor. We present a system for both healthcare for the elderly and industrial activities in rural areas. This communication system is based on DTN (Delay-tolerant networking) offering one solution for monitoring elderly people without an Internet connection and another solution for transmitting the performance and production data of local companies. The proposed approach exploits a routing algorithm

based on cooperation between nodes by dynamically adapting the communication according to the type of data and the receiver's interests. The main results of this contribution have been published in the journal *Wireless Communications and Mobile Computing (2021)* [42].

Elastic Data Analytics

- *Problem statement:* The massive deployment of Internet-connected devices has led to an increase in the collection of data that are then used by companies to improve their decision-making processes. This growing trend demands more and more cloud and communications infrastructure. The limited resources, the need of sharing them, and the fact that many consumers are interested in the same data call for efficient management of the available resources. In order to reduce the network overhead and improve the QoS, IoT applications already attempt to take advantage of the Cloud-to-thing continuum to deploy services closer to information providers and consumers. However, such applications are still isolated systems that do not favor the sharing of data or analytics streaming.
- *Contribution:* We address these problems by using Elastic Data Analytics, whose behaviour can be dynamically modified according to the quality requirements defined by each IoT system (freshness and accuracy) and the available resources of the cloud-to-thing infrastructure. These analytics have two parameters (freshness and accuracy) that can be modified depending on the quality required by the analytics. This impacts on the infrastructure that has to adapt to the needs of the set of analytics and the available resources. To define and deploy these analyses, we have provided a framework with an orchestrator that manages the elasticity produced by the variation in the quality requirements of the analytics. This orchestrator evaluates the state of the infrastructure, the other ongoing analytics, and the QoS required by each analytic. As a result, it reconfigures all analytics to maximize the quality provided by each analytics without exhausting the resources of the cloud-to-thing infrastructure. The main results of this contribution have been published in the journal *IEEE Internet Computing* [43].

1.5.2 Publications

This section presents a complete list of the publications generated by our research work. The publications are listed in chronological order. In addition, where possible, information about the quality indicator or other details are

given. For journal articles, it indicates the Journal Citation Reports (JCR) rating of the journal and, for conference articles, the conference class, and conference rating according to the GII-GRIN-SCIE (GGS) conference classification [44].

2019

During the first year of the PhD, we analyzed different works and state of art to deploy services on devices close to the users, i.e. the Mist layer with a focus on **SG1** (*"Providing tools and processes that speed up the implementation of emerging architectural styles such as Edge, Fog and Mist Computing"*). With this began the development of APIGEND[20], supporting the development of APIs on smartphones. The initial analysis together with the study of the art and the development of APIGEND produced its first results with the publication of the first articles [45, 24, 46, 47].

- **JCIS'19 [45].** S. Laso, D. Flores-Martin, Jose García-Alonso, J. Berrocal, J. M. Murillo, "Estimación y Generación Temprana de Aplicaciones para la IoT," Jornadas Cienc. e Ing. Serv. (JCIS), SISTEDES, 2019.
- **IWoG'19 [24].** S. Laso, D. Flores-Martin, J. L. Herrera, C. Canal, J. M. Murillo, and J. Berrocal, "Providing Support to IoT Devices Deployed in Disconnected Rural Environment," in International Workshop on Gerontechnology, 2019, pp. 140–150.
- **IWoG'19 [48].** D. Flores-Martin, S. Laso, J. Berrocal, C. Canal, and J. M. Murillo, "Allowing IoT devices collaboration to help elderly in their daily lives," in International Workshop on Gerontechnology, 2019, pp. 111–122.
- **IWoG'19 [46].** E. Moguel, J. García-Alonso, and S. Laso, "Yourpantry: food monitoring through pantry analysis using the smartphone and making use machine learning and deep learning techniques," Commun. Comput. Inf. Sci., vol. 1185, pp. 3–10, 2019.
- **MMTC'19 [47].** S. Laso, D. Flores, J. Garcia-Alonso, J. M. Murillo, and J. Berrocal, "Deploying APIs: Edge vs Cloud Environments," MMTC Commun., 14 - 5, pp. 10 - 15. IEEE, 05/09/2019.

2020

In this second year, we continued to focus on **SG1**. Following the first positive results on the deployment of APIs on end devices in 2019, the APIGEND

tool was extended to support more device types such as microcontrollers and smartwatches. All this resulted in a couple of publications in *PERCOM* [39, 26]. Other adaptations of the use of the mobile device as a personal computing end device, such as depression detection and tracking of foods in people's diets, led to further publications in a *International Workshop on Gerontechnology* and an *IEEE Access* journal [49, 50]. The adaptation of the SOLID model for deployment on smartphones was also researched and developed [31] by participating in the *International Conference on Web Engineering*.

Other adaptations of the use of the mobile device as a personal computing end device, such as depression detection and tracking of foods in people's diets, led to further publications

- **PERCOM'20 [39]** S. Laso, M. Linaje, J. Garcia-Alonso, J. M. Murillo, and J. Berrocal, "Deployment of APIs on Android Mobile Devices and Microcontrollers," in 2020 IEEE International Conference on Pervasive Computing and Communications (PerCom Workshops), 2020, pp. 1–3.

- **PERCOM'20 [26].** S. Laso, M. Linaje, J. Garcia-Alonso, J. M. Murillo, and J. Berrocal, "Artifact Abstract: Deployment of APIs on Android Mobile Devices and Microcontrollers," in 2020 IEEE International Conference on Pervasive Computing and Communications (PerCom), 2020, pp. 1–2.

  Quality indicator: GGS class (rating) 1 (A+).

- **ICWE'20 [31].** M. Jesús-Azabal, J. Berrocal, S. Laso, J. M. Murillo, and J. Garcia-Alonso, "SOLID and PeaaS: Your Phone as a Store for Personal Data," in International Conference on Web Engineering, 2020, pp. 5–10.

  Quality indicator: GGS class (rating) 3 (B-).

- **IWoG'20 [49]** D. Flores-Martin, S. Laso, J. Berrocal, and J. M. Murillo, "Detecting and Monitoring Depression Symptoms According to People's Behaviour Through Mobile Devices," in International Workshop on Gerontechnology, 2020, pp. 3–10.

- **ACCESS'20 [50].** B. Sainz-De-Abajo, J. M. García-Alonso, J. J. Berrocal-Olmeda, S. Laso-Mangas, and I. De La Torre-Diez, "FoodScan: Food Monitoring App by Scanning the Groceries Receipts," IEEE Access, vol. 8, pp. 227915–227924, 2020.

  Quality indicator: JCR Q2.

2021

In the third year of the PhD, the first and most important publication was focused on **SG1**, published in *Software: Practice and Experience* journal with the **Human Microservices** [13]. Related, there was also an important publication in the *Mobile Information System journal* with the deployment of the **SOLID** model in smartphones [32]. The **Distributed API Composer (DAC)** was also developed, supporting the upper layers Edge and Fog, the result of which was presented at the SISTEDES national congress [35]. However, during this year, research is also focused on the **SG2** goals (*"Providing tools and processes that facilitate the evaluation of the QoS of IoT applications that apply architectural styles making use of paradigms such as Edge, Fog and Mist Computing"*) by developing our **Perses** framework for the evaluation of distributed applications resulting in the publications [40, 39, 38]. And **SG3** (*"Provide mechanisms so that an application can dynamically change the architecture topology to adapt to the context needs or resources required at the time"*) with the development of our proposal **OPPNets** [42], making an important publication in *Wireless Communications and Mobile Computing journal*.

- **SPE'21 [13].** S. Laso, J. Berrocal, J. García-Alonso, C. Canal, and J. Manuel Murillo, "Human microservices: A framework for turning humans into service providers," Softw. - Pract. Exp., 2021, doi: 10.1002/spe.2976. *Quality indicator: JCR Q2.*

- **WCM'21 [42].** M. Jesús-Azabal, J. L. Herrera, S. Laso, and J. Galán-Jiménez, "OPPNets and Rural Areas: An Opportunistic Solution for Remote Communications," Wirel. Commun. Mob. Comput., vol. 2021, 2021.

  Quality indicator: JCR Q3.

- **MIS'21 [32].** M. Jesús-Azabal, E. Moguel, S. Laso, J. M. Murillo, J. Galán-Jiménez, and J. García-Alonso, "Pushed SOLID: Deploying SOLID in Smartphones," Mob. Inf. Syst., vol. 2021, 2021.

  Quality indicator: JCR Q4.

- **JCIS'21 [40]** S. Laso, J. Berrocal, P. Fernández, A. Ruiz-Cortés, and J. M. Murillo, "Perses: Un framework para evaluar la Calidad de Servicio en aplicaciones móviles distribuidas," Jornadas Cienc. e Ing. Serv. (JCIS), SISTEDES, 2021.

- **JCIS'21 [35].** S. Laso, D. Bandera, J. Berrocal, J. Garcia-Alonso, J. M. Murillo, and C. Canal, "Una Propuesta para la Composición de APIs Distribuidas," Jornadas Cienc. e Ing. Serv. (JCIS), SISTEDES, 2021.

- **PERCOM'21 [39].** S. Laso, J. Berrocal, P. Fernández, A. Ruiz-Cortés, and J. M. Murillo, "Virtual Environment for Evaluating the QoS of Distributed Mobile Applications," in 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), 2021, pp. 440–441.

- **PERCOM'21 [38].** S. Laso, J. Berrocal, P. Fernández, A. Ruiz-Cortés, and J. M. Murillo, "Artifact: Virtual Environment for Evaluating the QoS of Distributed Mobile Applications," in 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), 2021, pp. 440–441.

- **NGDS'21 [51].** S. Laso, J. Berrocal, J. Garcia-Alonso, C. Canal, and J. M. Murillo, "Service Oriented Computing for Humans as Service Providers," in Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing of the Future, Springer, Cham, 2021, pp. 111–122.

2022

During the last year of PhD, an important publication of our Perses framework [14] in *Pervasive and Mobile Computing* journal was achieved. Related to **SG3**, the research and development of Elastic Data Analytics were initiated, obtaining an important publication in the *IEEE Internet Computing* journal [43]. Additional publications on distributed IoT systems beneficial for people's healthcare and occupational prevention [52, 53, 54] were also carried out.

- **IWoG'22 [54].** D. Flores-Martin, S. Laso, J. Berrocal, and J. M. Murillo, "Contigo: Monitoring People's Activity App for Anomalies Detection," in International Workshop on Gerontechnology, 2022, pp. 3–14.

- **PMC'22 [14].** S. Laso, J. Berrocal, P. Fernández, A. Ruiz-Cortés, and J. M. Murillo, "Perses: A framework for the continuous evaluation of the QoS of distributed mobile applications," Pervasive Mob. Comput., vol. 84, p. 101627, 2022. *Quality indicator: JCR Q2.*

- **JCIS'22 [52].** D. Flores-Martin, S. Laso, J. Berrocal, and J. M. Murillo, "Detección de Comportamientos Anómalos Basados en la Utilización del Smartphone," Jornadas Cienc. e Ing. Serv. (JCIS), SISTEDES, 2022.

- **JCIS'22 [53].** S. Laso, D. Flores-Martin, J.P. Cortes-Perez, M. Soriano Barroso, A. Cortes-Perez, J. Berrocal, and J. M. Murillo, "Sistema IoT para la Prevención de Riesgos Laborales en una EDAR," Jornadas Cienc. e Ing. Serv. (JCIS), SISTEDES, 2022. [**Best demo award**].

- **PMC'22 [14].** S. Laso, J. Berrocal, P. Fernández, A. Ruiz-Cortés, and J. M. Murillo, "Perses: A framework for the continuous evaluation of the QoS of distributed mobile applications," Pervasive Mob. Comput., vol. 84, p. 101627, 2022. *Quality indicator: JCR Q2.*

- **COMPUTING'22 [43]** S. Laso, J. Berrocal, P. Fernández, J.M. García, J. Garcia-Alonso, J. M. Murillo, A. Ruiz-Cortés, and S. Dustdar, "Elastic Data Analytics for the Cloud-to-Things Continuum", IEEE Internet Computing. IEEE Internet Computing, 26(6):42–49, 2022. *Quality indicator: JCR Q2.*

Summary of Publications

Figure 1.4 shows a summary of our publications throughout the PhD, grouped by year. For the most important publications, we also show the relationship with the specific goals (SG) of the thesis (specific goal 1 = SG1, specific goal 2 = SG2, specific goal 3 = SG3). The research stays carried out are also shown grouped by year. On the other hand, Table 1.1 shows a list of the publications that appear in the figure related to the specific goals of the thesis. In the case of journal articles, the Journal Citation Reports (JCR) rating of the journal is indicated and, in the case of conference articles, the type of conference and the conference rating according to the GII-GRIN-SCIE (GGS) conference classification [44]. It also indicates to which specific goal they are contributing.

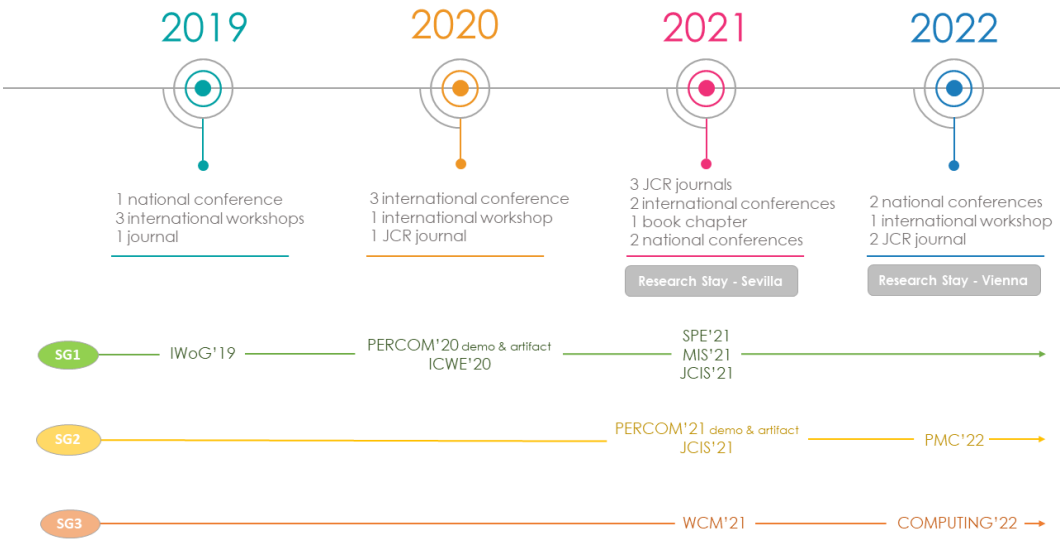


Figure 1.4: PhD timeline with publications grouped by year and specific goals.

Tittle	Conference / Workshop / Journal	Publication Date	Quality indicator	Goal
Providing Support to IoT Devices Deployed in Disconnected Rural Environment	International Workshop on Gerontechnology	November 2019		OE1
Deployment of APIs on android mobile devices and microcontrollers	International Conference on Pervasive Computing and Communications	March 2020	GGS class 1 (A+)	OE1
SOLID and PeaaS: Your Phone as a Store for Personal Data	International Conference on Web Engineering	June 2020	GGS class 3 (B-)	OE1
OPPNets and rural areas: an opportunistic solution for remote communications	Wireless Communications and Mobile Computing	January 2021	JCR Q3	OE3
Virtual environment for evaluating the QoS of distributed mobile applications	International Conference on Pervasive Computing and Communications (Workshop)	March 2021		OE2
Human microservices: A framework for turning humans into service providers	Software: Practice and Experience	April 20211	JCR Q2	OE1
Pushed SOLID: Deploying SOLID in Smartphones	Mobile Information Systems	August 2021	JCR Q4	OE1
Perses: Un framework para evaluar la Calidad de Servicio en aplicaciones móviles distribuidas	Jornadas de Ciencia e Ingeniería de Servicios	September 2021		OE2
Una Propuesta para la Composición de APIs Distribuidas	Jornadas de Ciencia e Ingeniería de Servicios	September 2021		OE1
Perses: A framework for the continuous evaluation of the QoS of distributed mobile applications	Pervasive and Mobile Computing	June 2022	JCR Q2	OE2
Elastic Data Analytics for the Cloud-to-Things Continuum	IEEE Internet Computing	December 2022	JCR Q2	OE3

Table 1.1: Summary of publications.

1.5.3 Developed tools

Our research has led to the creation of several tools that support the research approaches and goals presented in the thesis. The main tools developed are APIGEND[20, 55] and Perses[21]. Both tools are open source and available for use.

- **APIGEND.** API Generator for End Devices (APIGEND) is one of the main results of this thesis. APIGEND is an artifact that extends the OpenAPI Generator [56] to create and deploy an API on Android-based end devices, smartwatches, ESP32 microcontrollers, etc, simplifying the development of these APIs. One of the main problems of implementing APIs in Android devices is their consumption. Normally, due to security restrictions of the operating systems or the mobility of these devices (which can lead to an intermittent connection to the Internet

and changes in the IP address), the consumption of the services deployed cannot be based on synchronous communication. Therefore, an asynchronous communication scheme has also been supported in the developed extension which, in particular, includes different communication protocols such as Firebase Cloud Messaging (FCM)[57] and MQTT[58]. One of the advantages of this framework is that we have added all the necessary libraries to deploy an API and to support different communication schemes for automatic addition and inclusion in the generated skeleton.In this way, these APIs can be consumed by third parties as if they were deployed on a cloud environment. Another extension of this artifact was the inclusion of the Distributed API Composer (DAC) to facilitate its implementation across the Cloud-Fog-Edge layers. This element, as mentioned above, performed the entire process of coordinating the invocation, retrieval, and aggregation of data from the distributed APIs in the end devices or a set of DACs in a lower layer.

- **Perses.** Perses is another main result of this thesis. Perses is a framework that allows developers to easily deploy a virtual environment with multiple heterogeneous virtual devices to evaluate the QoS of a distributed mobile application simulating a real deployment environment. To use it, one only has to define a *configuration file* where the desired QoS and the characteristics of the environment where it is to be tested are indicated (number and type of devices, network configuration, etc.). Perses is fully integrated into a DevOps methodology, being able to automate all the different steps that should be executed during the evaluation process, from the deployment of the environment to the collection and checking of the results obtained to determine whether or not the application achieves the desired quality. This is an important feature for development companies since they can easily integrate Perses in their continuous integration pipeline.

1.5.4 Research Stays

During the research for this thesis, I have carried out two research stays. The first one at a national scale at the University of Seville, in particular in the Smart Computer Systems Research and Engineering (SCORE) Lab under the supervision of Prof. Dr. Pablo Fernández Montes and Prof. Dr. Antonio Ruiz Cortés. The second one was at an international scale at the Technische Universität Wien in Vienna (Austria), in particular in the Distributed System Group (DSG) under the supervision of Prof. Dr. Schahram Dustdar.

The first of them was from October to December of 2021 (two months). We worked closely with Prof. Dr. Pablo Fernández Montes. During the

research stay, we worked on two tasks in parallel. The first task was to extend and complete the Perses framework. The extension consisted of adding a new feature to simulate the network of virtual mobile devices (WIFI/LTE) with Kathara [59]. We write a research article on this framework which was published in the Pervasive and Mobile Computing journal (one of the relevant publications of this thesis) [14]. The second task was to explore a new line of research focusing on *elastic data analytics*. This line emerged with the visit of Prof. Dr. Schahram Dustdar to Seville. We developed a proof-of-concept scenario for the launch of elastic analytics. Also, we wrote a research article (*Elastic Data Analytics for the Cloud-to-Things Continuum*) presenting the proposed idea, the developed scenario, and the results which was published in IEEE Internet Computing [43].

The second of the research stays was from May to August (three months). We worked closely with Prof. Dr. Schahram Dustdar, an eminence in the area of new distributed systems such as Edge-Computing, Fog-Computing, etc. During the research stay, we continue to work on elastic data analytics to evaluate performance in the context of IoT applications. We started work on the development of a framework is being developed to generate scenarios with different conditions (number of devices involved, data granularity, etc.) to evaluate different parameters (QoS, costs, etc.) between an infrastructure with elastic data analytics and a traditional cloud and determine at which moments of an IoT application it is more beneficial to use one or the other depending on the context conditions. Currently, we continue to collaborate to finalize this work.

1.5.5 Grants

The goal of this section is to acknowledge all those who have supported us financially and allowed us to develop this thesis.

- **Industrial PhD.** Funded by the Spanish Ministry of Science and Innovation and Asociación Estatal de Investigación *MCIN/AEI/10.13039/501100011033* and European Union "Next GenerationEU/PRTR" through the *DIN2020-011586* grant.
- **Contexto Situacional: Una arquitectura de gestión de la información personal para una mejor integración persona-tecnología.** Funded by the Spanish Ministry of Science, Innovation and Universities, Ministry of Economy and Competitiveness - European Regional Development Fund (ERDF) through the *RTI2018-094591-B-I00 (MINECO/FEDER, UE)* project.

- **CoorDi: Coordinación automática de Dispositivos Inteligentes para su adaptación al contexto de los usuarios en ciudades inteligentes.** Funded by the Government of Extremadura and the European Union through the *TE-0046-18* grant.

- **Contexto-Situacional: Arquitectura tecnológica para automatizar la conexión de las personas a los dispositivos inteligentes**. Funded by the Department of Economy, Science and Digital Agenda of the Government of Extremadura through the *IB18030* project.

- **4IE+ Project.** Funded by Interreg V-A España-Portugal (POCTEP) 2014-2020 program through the *0499-4IE-PLUS-4-E* project.

- **Research Group support.** Funded by the Government of Extremadura, Ministry of Economy and Infrastructure, and European Regional Development Fund (ERDF) through the grants *GR18112* and *GR21133*.

1.6 Structure of the Thesis

This thesis is structured into four chapters and an appendix, each of which is composed of sections. The five chapters are as follows:

- **Chapter 1: Introduction.** This chapter includes a contextualization of the thesis for the readers including, goals, methodologies followed, and contributions.

- **Chapter 2: Contributions.** In this chapter, the main contributions of this thesis are explained.

- **Chapter 3: List Of Publications.** This chapter includes a description of the main publications of this thesis and their full texts. Supporting publications are also described.

- **Chapter 4: Conclusions.** This chapter is the final part of the thesis, it presents the conclusions, discusses the results obtained, and the future lines of research are also presented. Finally, there is also a section on personal reflections.

- **Appendix: Supporting Publications.** The Appendix contains the full text of the publications supporting this dissertation.

"We must reach far beyond our own lifespans. We must think not as individuals but as a species"

Dr. Brand (Michael Caine), Interstellar

Chapter 2

Results

This chapter presents the different contributions to research that have been achieved during the PhD. If we remember the main goal, it says *"Contribute new processes, techniques, and tools to facilitate the implementation, evaluation, and deployment of IoT applications based on these new emerging architectural styles by leveraging the capabilities of end devices (Mist) and near nodes (Edge and Fog), covering important activities of the software development methodologies"*. During this chapter different contributions to the development of IoT applications in the Clout-to-thing-continuum are shown that attempt to cover implementation, evaluation, and deployment activities in software development methodologies.

Today, when a company wants to develop an application, they are usually guided by software development methodologies such as DevOps[60]. Before, the teams worked in isolation, so the design, implementation, testing, and deployments were performed in isolation, all this added to the need for constant human intervention throughout the deployment processes and the possible errors arising from them, affecting the delivery time of the projects. With DevOps, all these times are considerably reduced, in addition to the automation of repetitive tasks, simplifying the processes. As a result, higher quality and more reliable end products are obtained. And as a consequence of all the above, we also reduce maintenance time in production [61].

DevOps Life Cycle is a series of development stages that guide everyone in the most efficient way possible for product development. Figure 2.1 shows the DevOps Life Cycle [15]. Each of them is explained below.

1. **Plan:** In this stage, teams identify the business requirement and collect end user feedback. They create a project roadmap to maximize the business value and deliver the desired product during this stage. Typical

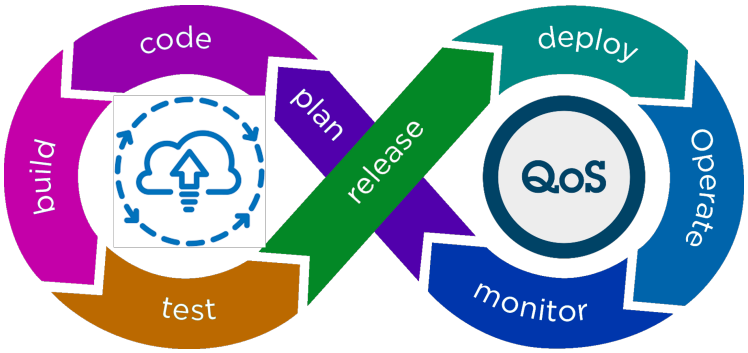


Figure 2.1: DevOps Life Cycle

activities include project management, scheduling, release plans, policies/requirements, etc.

2. **Code:** Application code is developed using appropriate programming languages and IDEs. Code is usually maintained using version control systems.

3. **Build:** In this stage, integrates various software modules to build executable files for product features or fully developed products.

4. **Test:** Once the build is ready, it is deployed to the test environment first to perform several types of testing user acceptance test, security test, integration testing, performance testing, etc. It ensures that potential errors are eliminated in developed software and that a reliable product is delivered to production.

5. **Release:** The build is ready to deploy on the production environment at this phase. Once the build passes all tests, the operations team schedules the releases or deploys multiple releases to production, depending on the organizational needs.

6. **Deploy:** It is a process to promote software components from one environment to the next. If it is successfully deployed, the features or product is ready for release.

7. **Operate:** The release is live now to use by customers. The operations team at this stage takes care of infrastructure configuring and provisioning.

8. **Monitor:** In this stage, the DevOps pipeline is monitored based on data collected from customer behavior, application performance, etc. Monitoring the entire environment helps teams find the bottlenecks impacting the development and operations teams' productivity

All these components of the DevOps Life Cycle are necessary to take full advantage of the DevOps methodology. The contributions presented cover the **Code**, **Test**, and **Operate** stages, supporting IoT applications across the Cloud-to-thing continuum.

In this chapter, the contributions will be grouped by the different activities in the DevOps Life Cycle that have been covered (Code, Test, and Operate). In Section 2.1 we explain our contributions in the area of implementation, in which we present processes and tools/frameworks to facilitate the development of these in the different layers (Mist, Edge, and Fog) of the Cloud-to-thing continuum. In Section 2.2 we present our framework called Perses to allow developers to easily deploy a virtual scenario with multiple heterogeneous virtual mobile devices to evaluate the QoS of a distributed mobile application. Finally, in Section 2.3 we present our proposal on OPPNets, a solution based on an opportunistic network algorithm based on cooperation between nodes by dynamically adapting the communication according to the type of data and interests of the receiver in rural areas with low connectivity and our approach of Elastic Data Analytics, whose behavior can be dynamically modified according to the quality requirements defined by each IoT system and the available resources of the Cloud-to-thing infrastructure.

Contents

2.1 Code

In this stage, the main purpose is to code the software. The whole development process is divided into smaller development cycles. This process makes it easier for developers to speed up the overall software development process.

The main problem we have encountered at this stage is that there are no tools, standards, frameworks, etc., that allow easy development for developers. While Cloud Computing architectures provide technologies that abstract from low-level details for development, deployment, etc., for computational infrastructures closer to the end user, such technologies do not exist, instead they have to design and implement their own solutions, leading to longer development times and low maintainability among other drawbacks.

The following sections present three contributions focused on facilitating the implementation of applications for the final Mist layer with **Human Microservices** to facilitate the implementation of applications to provide services from users' end devices and **SOLID on Smartphones** as a secure method of storing personal information, having control over access to data. For the Edge, Fog, and Cloud layers, we present our proposal called **Distributed API Composer**, which performs the whole process of coordinating the invocations, obtaining the data, and aggregating it from the distributed APIs in the end devices.

2.1.1 Human Microservices

With the massive deployment of IoT devices [23], mobile devices are also taking on the role of the control center for all of them. Through mobile devices, one can control and get information from almost any IoT devices owned by the user (from temperature sensors to smart wristbands) by just using the applications offered by the manufacturers.

Most of these applications are based on a pure client-server architecture in which mobile devices act as simple clients (or as the gateway of IoT devices). They get the information (if required, from the IoT devices) and post it to the cloud by invoking different microservices. These applications are usually designed using the Service Oriented Computing (SOC) paradigm [62], allowing developers to divide their functionalities into small and maintainable modules focused on specific features. Thus, these modules or microservices can be developed, deployed, modified, and re-deployed without compromising other functional aspects of the application, improving the quality, maintainability, and scalability of the application.

Mist Computing alleviates the stress of these environments by distributing computation tasks onto end devices (IoT devices, smartphones, etc.). Thus, since the distance between the consumer and the provider of the information is reduced, the response time and the stress on the cloud and network infrastructure are usually also reduced. However, these paradigms are in their

infancy, and the technological support for developers is very limited. For instance, smartphones have closed and security restricted operating systems so that microservices cannot be directly deployed and provided from them. In addition, these devices are mobile, so they are constantly changing from one network to another, leading to intermittent connections and changes of IP address. Therefore, traditional tools and standards cannot be used to deploy microservices on these devices.

However, developing applications based on such paradigms requires developers skilled in different areas (hardware, communication, operating systems, etc.). Whilst cloud providers and other companies provide some standards and technologies abstracting from the low-level details to develop, deploy, mash up, and consume APIs, there are few similar mechanisms and technologies for computational infrastructures closer to the end user (smartphones, IoT devices, wearables, etc.). Using precarious and ad hoc mechanisms directly implemented by each developer to develop such applications can affect the development time, maintainability, and reproducibility of the software.

For Mist computing, there are currently some solutions [63, 64, 65] for the development and deployment of IoT applications on these devices. Two examples are storing and sharing ECGs in healthcare scenarios [63], and replicating the sensed data among different IoT devices [65]. Nevertheless, again they are adapted and personalized to a specific scenario – there are no standard mechanisms and technologies for developing, deploying, and consuming services as there are for cloud environments. Therefore, developers have to use all their skills and knowledge to design and implement ad hoc infrastructures to distribute IoT applications. This firstly increases development time and cost, and secondly reduces maintainability.

We propose Human Microservices as a way to provide a framework in which the application of the Mist computing paradigm can be developed more quickly and easily, allowing computing services to be easily deployed on devices closer to the end user. The framework involved is capable of generating these microservices using the same technology that developers are currently using for cloud environments, thus shortening their learning curve, and promoting and favouring the use of SOC architectures and tools.

Development process

In order to perform a uniform deployment of APIs on mobile devices, developers need to be able to rely on a well defined process. The different steps proposed are shown in Figure 2.2. As can be seen, these are the activities that are usually carried out in designing and implementing APIs for cloud environments.

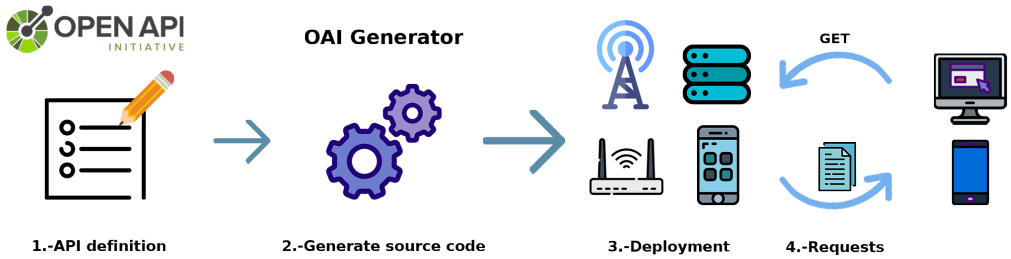


Figure 2.2: Deployment process to generate mobile APIs.

1. **API definition.** First, the API to be deployed must be designed. In this case, this has to be carried out using the standard OpenAPI Specification (OAS) which is widely used in the industry for the design and implementation of APIs that are to be deployed in cloud environments.

   The definition follows the same notation and logic as if it corresponded to deployment on a cloud environment (i.e., the specification defines the different microservices that will be available for consumption).

2. **Generation of source code.** In this step, the skeleton of the API is generated. Currently, there are different tools allowing the scaffolding of OAS-defined APIs for a client-server architecture. Examples are OAS-Generator [66], Guardrail [67], and Swagger Codegen [68]. These tools do not support the generation of APIs for end devices. Therefore, an extension of the OpenAPI Generator has been developed to create and deploy APIs on Android mobile devices. This tool is denoted APIGEND (API Generator for End Devices). APIGEND is a Web application developed with Spring [1] that uses Mustache [2] templates for code generation. At present, we are hosting it on Heroku [20], and it is available to be used by any developer.

3. **Deployment.** Before being able to deploy the API, the developer must finish the implementation by adding the business logic –i.e., the behaviour of the different endpoints defined (obtaining information from the sensors or the stored virtual profile)– and the configuration of the communications protocol. Then, the API can be deployed following the process defined by the manufacturer.

4. **Service invocation.** Finally, with the API deployed, any device with the necessary permissions will be able to consume the defined microservices, even chaining calls to the microservices to obtain progressively

[1] https://spring.io
[2] https://mustache.github.io

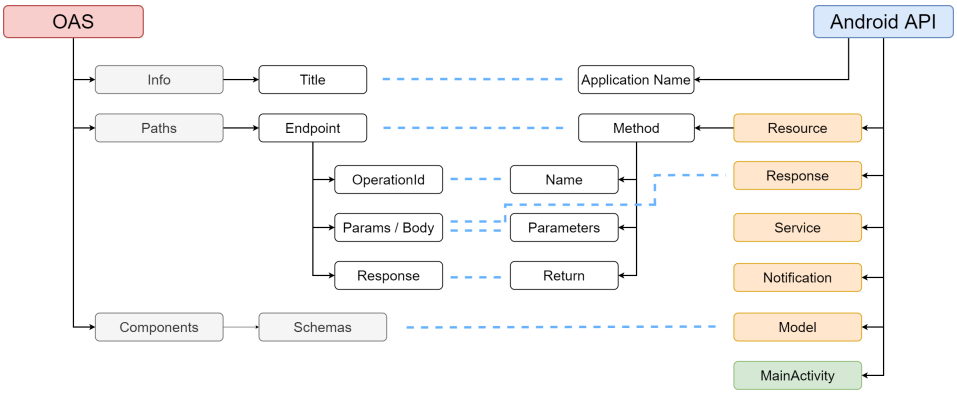


Figure 2.3: A synthesis of the conversions between OAS and the Android API's main structure.

more refined information from a user, an essential aspect in order to have IoT devices and information systems capable of reacting in demanding times to the user's needs. The endpoints provided can be consumed by other servers or devices, using a peer-to-peer schema.

Scaffolding of mobile device APIs

Once the API has been designed using OpenAPI, its skeleton has to be generated. As mentioned above, we extended OpenAPI Generator to support the generation of code for Android mobile devices. The process of generation and deployment is similar for all these devices, and to avoid being redundant, in this paper our focus will be on the whole process for mobile devices with Android OS.

This tool is denoted APIGEND (API Generator for End Devices). APIGEND is a Web application developed with Spring [3] that uses Mustache [4] templates for code generation. At present, we are hosting it on Heroku [5], and it is available to be used by any developer.

Figure 2.3 shows a diagram synthesizing the different conversions made between the OpenAPI specification and the Human Microservices. It details the different elements of the specification and the set of directories and classes generated for the API. The blue dashed lines represent the relationships connecting them.

[3]https://spring.io
[4]https://mustache.github.io
[5]https://openapi-generator-spilab.herokuapp.com/

Human Microservices Validation

We evaluate the Human Microservices with the development and deployment of a real case study with a resource consumption analysis. In addition, we evaluate the proposed framework through a survey of different developers who were trained in its use. A survey was conducted with 79 participants to evaluate the difficulty of implementing and deploying Human Microservices with an example and the results were very favorable.

The case study corresponds to a real environment in which the application was developed. The application, called Contigo [6], was developed during the initial stage of the COVID-19 pandemic for the institution SEPAD [7] (Extremadura Service for the Promotion of Autonomy and Attention to Dependence) to monitor elderly people during this pandemic. Contigo is especially aimed at elderly people who live alone, and therefore for whom it is found more difficult to know what their health status is. The application has deployed different microservices on the users' devices to check their activity, if this did not meet the requirements defined by the healthcare staff, they could access their location or personal data through their microservices.

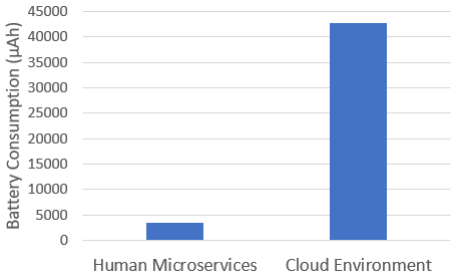
For its evaluation, we compared this application with deployment on users' smartphones and another one deployed in the cloud measuring different parameters; battery consumption, network usage, and costs. Figure 2.4 shows the results obtained from these measurements, including battery consumption, data traffic consumption, and operating costs generated, comparing the deployment of microservices using Human Microservices and a cloud environment.

Consuming Human Microservices that provide highly personalized and up-to-date information entails the consumption of many resources. As shown in Figure 2.4, the deployment on the users' smartphones reduces resource consumption by up to twelve times and does not generate any operational costs compared to their deployment in a cloud environment.

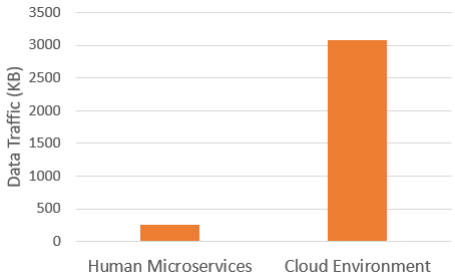
The training course was given at three universities (University of Malaga, University of Granada, and University of Valladolid) and at the International Conference on Web Engineering to students of different master's degrees, developers, and researchers. In total, there were 79 participants. The first part consisted of a survey inquiring into the problems of current architectural designs and the existing tools for deploying APIs on end devices. The second part focused on a training course explaining the concepts of mobile computing architectures and the use of the Human Microservices framework. And in the

[6] https://contigo.spilab.es/contigo/

[7] https://saludextremadura.ses.es/sepad/inicio



(a) Battery consumption.



(b) Data consumption.

(c) Operating Costs.

Figure 2.4: Human Microservices vs Cloud Environment resource consumption for the defined case study.

third part, we conducted another survey to inquire about the usefulness of the tools presented. The results were very favorable in both surveys and with very interesting results.

To know more about Human Microservices and its validation through surveys, we refer the reader to Section 3.3 [13].

2.1.2 SOLID in Smartphones

Privatization and centralization of information are common practices for applications and companies. Data provide a significant competitive advantage which enables a deeper understanding of the market and users. Services can be improved and adapted to the changing demands of the audience which can be analyzed and studied to infer these conclusions [69]. Besides, the targeted advertisement makes data collection lucrative [29]. However, this privatization trend drives sites to form information silos, so that data storage is performed independently and autonomously. Hence, access to other applications is disabled, and users do not enjoy much control over their data. As a consequence, these policies result in potential privacy problems, inconsistencies, duplicity, and insecurity issues.

There are proposals that attempt to offer alternative models of storing and delivering data from Mist devices to be computed using the Cloud continuum. In this way, users store their information and have control over it. Some of the most popular are WebBox [70] or Diaspora [8]; however, Social Linked Data (SOLID) initiative [30] becomes the most relevant among these solutions. SOLID proposes the decentralization of the information to separate applications and personal data [71]. This way, users store their information in autonomous entities called Personal Online DataStores (PODS). Therefore, applications would adopt a model where they do not store data but request it from PODS. Thus, users are able to control access and authorize or deny petitions. Adopting this model provides users with an enriched experience based on accuracy, privacy, and control, giving responses to information duplication and inconsistencies. This way, PODS becomes a useful element to enhance the experience for users and to improve companies' services.

This work proposes an architecture that enables the implementation of the SOLID initiative for storage in smartphones. The solution mixes these two lines into a project which situates the user at the center of information management. Thus, smartphones become the store of their data, empowering the devices to serve as providers of information to the external applications which require it. This proposal aims to provide a response to the increasing interest in data governance and presents a solution for inconsistencies and duplicity of information on personal profiles.

The proposed architecture relies on a set of components that collaborate to provide the information to external requests as shown in Figure 2.5. This scheme explains the complete process to communicate a petition from an external application (ϵ) to the SOLID PODS (P_{solid}) of the user, executed on its smartphone. This way, three main entities shape the architecture: the SOLID PODS (P_{solid}) and Pusher app (P_{app}) in the smartphone (S) and API Gateway (G) in the server. Additionally, Firebase (F) is used to communicate petitions between the server and the smartphone and the external application (ϵ) is the one that begins the process. Next, the elements of the architecture are detailed and explained.

We evaluate this work considering two points, an experimental evaluation of user's opinion and a system evaluation about the performance and technical viability of the proposal has been performed. For this evaluation, the implementation of Pushed SOLID counts with PODS which stores a profile with personal information, historical music played, and main interests

The SOLID PODS is the entity in charge of storing personal information. This element is executed locally in the user's smartphone. It has been

[8] https://diasporafoundation.org/

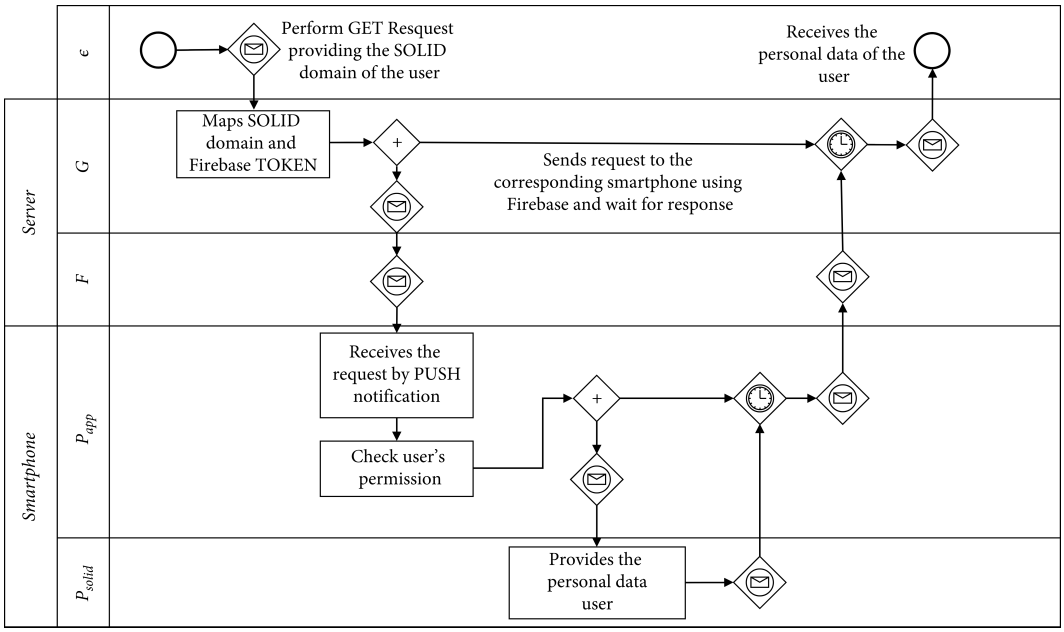


Figure 2.5: Pushed SOLID architecture.

designed to be deployed on a server; however, in the current proposal, they have been adapted to be deployed on smartphones. Thus, the PODS stores data locally in the device, independently from any external entity. In the current proof of concept of the solution, users can interact with the platform using the default web interface in the phone. Considering the main goal of this implementation is to analyze the viability of the proposal.

Once the SOLID PODS is configured, it is ready to provide information. However, the data store is locally deployed, so it can not be accessed externally, and only the own phone visualizes the platform. Nevertheless, we want P_{solid} to provide the requested information to external apps (ϵ), so that it is required an additional mechanism to perform this. Hence, we turn to the Pusher application (P_{app}) to manage external requests. This application provides communication between the API Gateway (G) and the SOLID Server (P_{solid}) which runs on the smartphone. The API Gateway is required to effectively resolve the petitions and to map the SOLID domain requests with the address of the device. Nevertheless, the adoption of this intermediate layer does not work against the decentralized philosophy since the API Gateway does not store further data than the tuples related to addressing the petitions. In order to perform this communication between the API Gateway and the smartphone, Firebase (F) [57] is used to connect the petitions from the device.

Once Firebase communicates the petitions received in the API Gateway, this component communicates the request with the smartphone executing a

callback method of the Pusher app. Therefore, the method executed in the Pusher app asks for the required information to the SOLID PODS, which selects the corresponding data from the profile and returns the values. As a result, the Pusher app provides the requested data to the API Gateway. Considering the callback method identifies the requester entity and SOLID PODS counts with access control; security and validation tasks can be easily integrated into the process. As a result, the technical details to replicate the implementation process can be found in the project repository [9].

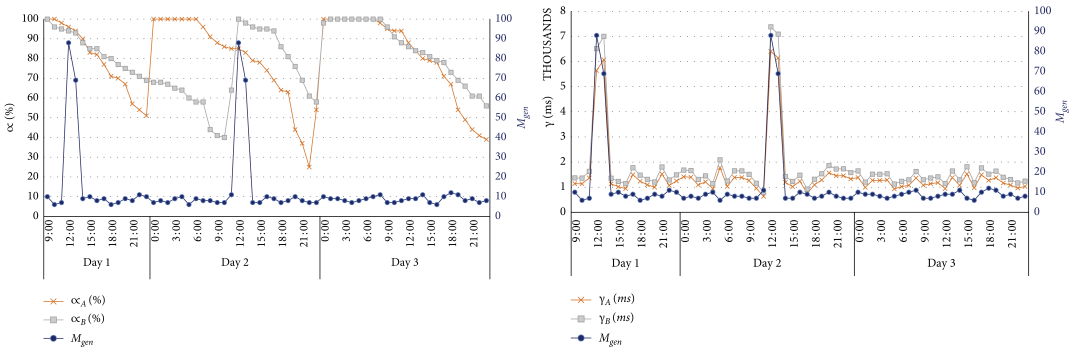
Pushed SOLID Validation

The evaluation performed in the experiment followed two dimensions: the perception and usability of the solution and the technical performance of the implementation.

The first step was to select a group of sixteen users, who are not related to the project, and to introduce them to the purpose of Pushed SOLID. The background of the participants varied, with only six technicians. This way, we try to avoid bias. A first survey about the perception of the idea was done, addressing privacy politics, data centralization, and opinions, from a neutral point of view. This way, the questions raised clearly the issue, approaching current tendencies and the concept that users have understood about the proposal. The answers were based on ranges: each question (Q) raised a topic and testers had to provide a score (S_p) from $0 \leq S_x \leq 5$, according to the relevance they give to that specific topic. Additionally, a comments box gathered opinions and suggestions. Taking into account the favourable results obtained in the usability tests, the proposal meets the usability requirements and provides a disruptive solution for managing information.

Regarding the evaluation of the performance of implementation the provided implementation of the Pushed SOLID was evaluated in a laboratory context. A three-day run was carried out continuously with the launching of random requests. Firstly, concerning battery consumption, it was not directly affected by the requests (even with the two programmed peaks) as shown in Figure 2.6a. Therefore, we could deduce that the power consumption derived from the operation of the architecture on smartphones is very low. Secondly, regarding response time, the programmed peaks of requests had a direct impact on the time required to solve the petitions as shown in Figure 2.6b. As a result, the time required to retrieve information from the smartphone can be optimized. However, considering the main goal of the assessment is analyzing the viability of the proposal, the delays can be accepted.

[9] https://bitbucket.org/spilab/solidsituational.context



(a) Battery (α) evolution and messages generated (M_{gen}) of smartphones.

(b) Response time (γ) evolution and messages generated (M_{gen}) in smartphones.

Figure 2.6: Pushed SOLID performance results.

For full details on our SOLID on smartphones approach, use case, validation results, etc., we refer the reader to Section A.8 and [31].

2.1.3 Distributed API Composer (DAC)

The increasing computational capabilities of both networks and end devices have led to a high distribution of IoT applications and their corresponding APIs. Paradigms such as Fog, Edge or Mist [9] allow developers to replicate different parts of these APIs in nodes closer to the end user, or even in their own devices, taking advantage of the computational capabilities of these devices to minimize response time and improve the Quality of Service[72]. This distribution of the computational load allows IoT applications to be deployed with a rigorous Quality of Service, being especially necessary in environments with strict QoS requirements, such as Industry 4.0. or the healthcare sector [73].

To consume these distributed APIs, it is necessary to have a single point of access to them, as it is not reasonable to access each device individually. This task places a significant overhead on the development of distributed systems. Managing this process is not trivial and considerably increases development time and effort when implementing the system.

An example of such systems are crowdsensing applications. These applications involve a large number of individuals collectively collecting and extracting data from specific areas of interest. Each individual is independent of the others, but they have the option to share the collected data to collaborate toward a common goal. In this case, APIs are replicated and distributed as close as possible to the source of the data with the goal of obtaining a

large amount of information while minimizing the computational overhead of the process. However, collecting all this data requires accessing all these devices one by one, and once collected, post-processing is necessary to ensure the integrity, and consistency and reduce redundancy of this data. This data management process should require as little effort as possible for developers.

Therefore, we present a proposal called DAC (Distributed API Composer), whose main goal is to coordinate the invocation to retrieve and aggregate information from a final API deployed in a multitude of devices. For this purpose, we have designed a conceptual model (Figure 2.8) to capture all its concepts and features. This model has helped us to develop an extension of the OpenAPI language with the goal of making its implementation and deployment easier for developers.

To retrieve the information, the DAC performs a parallel invocation to the final deployed APIs, these respond with the requested information and the DAC collects and aggregates all the information to expose it as the final result. The DAC aggregation is not only based on retrieving and exposing the raw information but also on performing some processing to expose it as adapted as possible to the final result and improve compliance with the Quality of Service requirements. To facilitate the aggregation of the data, different operations have been defined to process the data and can be reused between the different endpoints. In addition, it also allows defining conditions in the communication with the final APIs to offer an acceptable Quality of Service.

DAC acts like a regular REST API, with its endpoints and parameters and the same way of invocation. The interface of a DAC for an API is the same as that of the final API so that its invocation and use are completely transparent to any user. It can be deployed both in the cloud and in Fog and Edge nodes closer to the APIs to offer a better Quality of Service or reduce information traffic.

Figure 2.7 shows a general outline of an example case study, deploying the APIs on users' mobiles and the DAC for use in the shopping center. For example, entities with permission (mall staff, security, or IoT devices), can query the average temperature desired by users through the DAC endpoint, which coordinately invokes the APIs to obtain the temperature of each user and aggregates the information (the arithmetic average of temperatures) to return the average temperature as a result to control the climate control system.

The different concepts and characteristics necessary to define a Distributed API Composer are shown below. As well as the relationships between the different concepts. Figure 2.8 shows the conceptual model designed to show these concepts and relationships.

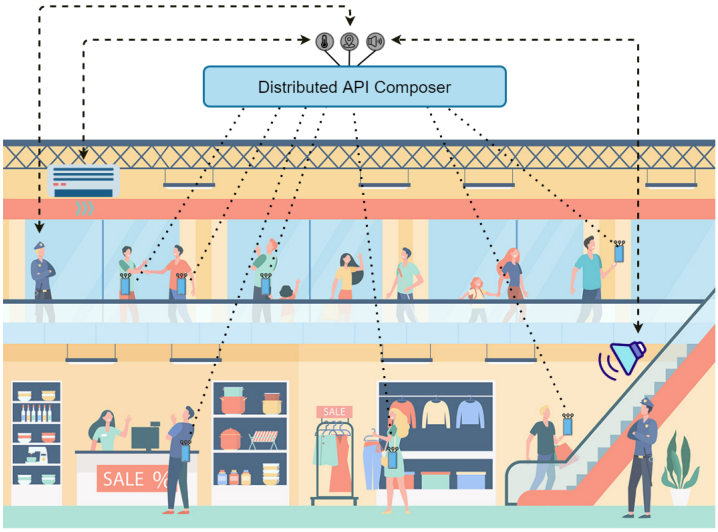


Figure 2.7: Case study: Deploying DACs and APIs in a shopping centre

API: is the minimum unit that can be deployed and is composed of one or more microservices.

- Endpoint: are the different microservices that can be consumed. These endpoints are the same for both an API and a DAC.
- Data Object: are the data types used for endpoint input and output. These data can be used as operands for the DAC.
  - DAC: defines the aggregation rules for each microservice when a supporting DAC is to be generated.
    - Operand: details the operands to which the aggregation rules have to be applied for a specific endpoint.
    - Operator: defines the different rules that will be used to aggregate the information when invoking the same endpoint of the distributed APIs. Currently, DAC supports different types of operators:
      - *Logical Operator:* are basic logical operators that can be applied to the operands. For example, AND, OR, etc.
      - *Math Operator:* these are basic mathematical, such as average, sum, etc.
      - *Domain Operator:* these are operators that perform some kind of processing depending on the domain of the API. Therefore, this element allow developers to extend and implement their own operators to cover domain-specif aggregation functionalities.

* Quality: allows defining the conditions under which the responses obtained from the APIs should be enough to obtain the required quality of the data. Two different conditions can be defined:
  - *Temporal:* allows defining a maximum response time (in milliseconds) to terminate the communication. If the number of connected devices is not known, it is important to define this condition to terminate data collection and proceed with the aggregation.
  - *Accuracy:* allows developers to define the maximum number of responses to be received from the APIs. There may be endpoints where it is not necessary to collect information from all the end devices with the API deployed, but only a representative sample, thus reducing response time.

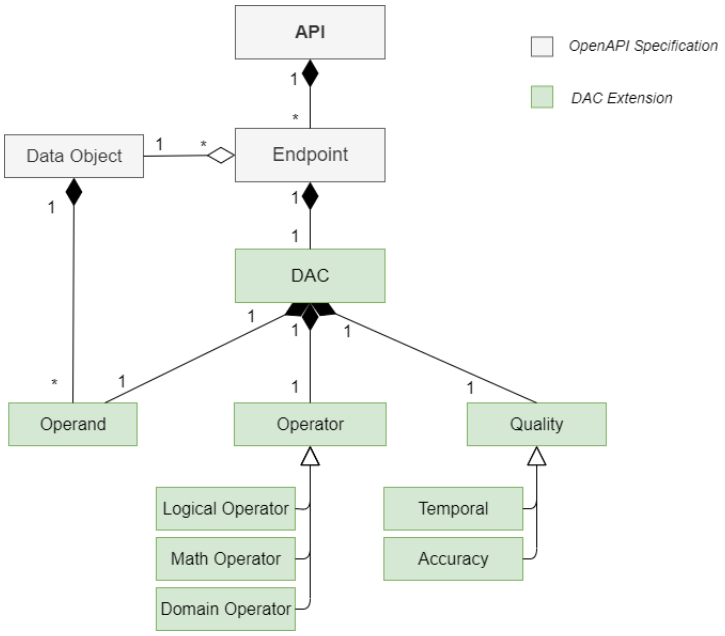


Figure 2.8: DAC - Conceptual Model

OpenAPI is a widely used industry standard that allows the design and documentation of APIs (sets of microservices) for both cloud and end devices [13] that can be used to generate source code, saving development time. Currently, with the existing specification, we do not have the necessary capabilities to define DAC properties. Nevertheless, it is an extensible language, allowing one to describe additional functionalities that are not covered by the original specification. Therefore, in this subsection, we show the creation of

an extension to support the concepts defined in the previously detailed conceptual model. This also allows developers to use the same specification to generate both an API and the DAC.

To describe new properties in the OpenAPI specification, fields preceded by *"x-"* are inserted, this is the nomenclature defined by the standard to add extensions without impacting existing validation and code generation tools. The extension has been defined following the conceptual model. Thus, to define a DAC, it must be declared using the tag *"x-dac"* at each endpoint. Within the label are specified the different properties defined in the model (operand, operator, and quality) that are necessary to define the aggregation rules.

Listing 2.1 shows part of the specification definition where the temperature microservice of the case study is defined. The DAC receives from the APIs an object called User, in the DAC it is indicated that in the property *User.temperature* the mathematical operator of the average is applied. As quality parameters, it is expected to receive information from 200 devices without exceeding a maximum of 1500 milliseconds. The specification for the case study can be found at GitHub [10].

```
{   paths:
      /temperature:
        get:
          summary: Gets the environment temperature
          operationId: getTemperature
          tags:
          - User
          x-dac:
            operand: User.temperature
            operator:
              type: math
              name: AVG
            conditions:
              precision: 200
              temporal: 1500
          responses:
        ...    }
```

Listing 2.1: OpenAPI extension for DAC definition.

As was the case with APIGEND [20] which allows the generation of source code for both the cloud side and the client side in different languages and end devices, through an OpenAPI specification. In this work, an extension to the APIGEND generator has been developed to support the generation of DAC code through the OpenAPI specification.

The next step is the writing and preparation of an article to be submitted to a journal. To learn more about the DAC where and its development

[10] https://github.com/slasom/JCIS2021/blob/main/shopping-center.yaml

process, we refer the reader to Appendix A.9 and [35].

2.2 Test

This phase, the developed software undergoes continuous testing to detect errors and evaluate the Quality of Service (QoS). The evaluation of QoS is one of the most crucial aspects playing an important role in satisfying users' needs [74]. For continuous evaluation, automation testing tools are used such as Apache JMeter[11], JUnit[12], Selenium[13], etc. These tools allow multiple code bases to be thoroughly tested in parallel to ensure that there are no bugs in functionality. End-to-end (E2E) testing is also common, where full functionality testing is performed, simulating end user behaviour. In this phase, Docker containers[14] can be used to simulate the test environment among others.

However, QoS evaluation testing of distributed applications is complex to evaluate. With these new paradigms, it is necessary to deploy a considerable set of heterogeneous devices close to the real scenarios to properly evaluate QoS - latency and execution time can vary greatly depending on the devices involved. This is especially relevant in today's mobile device ecosystem, due to market fragmentation [14].

Therefore, new techniques are needed to evaluate the QoS of distributed applications deployed across the Cloud-to-thing continuum by simulating a close-to-reality scenario. This will allow developers to launch E2E tests and measure the QoS obtained.

In the next section, we present our framework called **Perses**. Perses allows to evaluate the QoS in distributed applications simulating virtual scenarios with heterogeneous devices. It can also be integrated into a software development process, such as DevOps, so that it can be widely adopted by companies, saving the effort and cost required to perform these tests.

2.2.1 Perses

The success or failure of mobile applications largely depends on the quality of experience (QoE)[75] they offer, i.e., the satisfaction of a user with the provided

[11] https://jmeter.apache.org/
[12] https://junit.org/
[13] https://www.selenium.dev/
[14] https://www.docker.com/

functionality. The QoE is usually considered a broad term evaluating both the QoS [72] – quantitative measures of the performance of a service— and the user experience (UX) – efficiency of the interaction between the end user and the service.

To ensure their success, mobile applications usually follow a Cloud architecture. All computing demanding components (back end) are offloaded to cloud environments. Only the user interface and some basic components (front end) are deployed on mobile devices. This architectural style allows developers to reduce the resource consumption on these devices; thus, allowing their deployment on almost any device with a minimum set of characteristics – limiting the problems caused by the great heterogeneity of devices in this market.

To evaluate the required quality, developers can assess each side (back end and front end) independently, but also E2E testing is also performed. E2E tests asses the correct integration of the two sides and the correct functioning of the main functionalities, replicating the behavior of the users [37].

However, the QoS of distributed mobile applications is complex to evaluate. E2E evaluation in such architectures is not as trivial as in more traditional architectures. In a client-server architecture, it is only necessary to have the back end available and execute the end-to-end tests from a device. With these new paradigms, it is necessary to deploy a considerable set of heterogeneous devices close to real scenarios to properly evaluate the QoS –the latency and the execution time can vary greatly depending on the devices involved. This is especially relevant in today's ecosystem of mobile devices, due to the market fragmentation. This makes it even more necessary to know the expected quality in advance.

A potential solution for evaluating the QoS in these distributed applications is to use a real device farm that allows deploying the application on a large set of devices, and then, together with the cloud node, launch end-to-end tests, simulating the behavior of any user and the QoS obtained.

Currently, some commercial tools allow the evaluation of applications on different devices with a real device farm. AWS Device Farm [76] or Azure App Center Test [77] are platforms that provide a farm of real mobile devices. However, they do not allow the deployment of several mobile devices and the launch of E2E tests triggering different interactions among them, which is required to properly evaluate these distributed applications. Therefore, new techniques and tools are needed to evaluate the QoS of distributed mobile applications with E2E testing, enabling large-scale simultaneous evaluation of several highly interacting mobile devices and considering the heterogeneity of today's ecosystem.

Perses is a framework that allows developers to easily deploy a virtual scenario with multiple heterogeneous virtual mobile devices to evaluate the QoS of a distributed application. These applications are characterized by distributed computing, in which some or all of the main computation that significantly affects QoS is performed on the end devices (executing activities such as data processing, running an AI model, etc.). In addition, these applications seek to enhance user privacy by storing data on end devices locally. Given an initial file with the configuration of the virtual scenario (infrastructure, network, devices, tests, etc.), Perses deploys it in a cloud instance.

Perses is fully scalable, allowing the evaluation of virtual scenarios formed by a large number of virtualized heterogeneous devices. The heterogeneity of virtual mobile devices is characterized by the possibility of deploying them with different hardware, operating system, and configurations, being able to simulate close-to-real scenarios where there is a multitude of devices with different hardware and software characteristics.

Once deployed, Perses launches the tests, collects the logs from the devices, and analyses the results. To support the evaluation of different dimensions of the QoE, Perses allows the execution of QoS tests and UX tests. First, it allows the configuration and definition of E2E tests evaluating different devices and the interactions among them, which are essential for evaluating QoS in distributed mobile applications. The UX tests are focused on evaluating the user interface with Espresso. Thus, Perses covers the most important dimensions evaluated by developers.

Finally, to save the time required for the manual execution of Perses, it is fully integrated into the DevOps methodology, automating the different steps during the evaluation process, and allowing software companies to easily integrate Perses into their continuous integration pipeline.

Figure 2.9 shows the architecture of Perses, formed by different modules focused on specific functionalities that complement each other. The most important modules are described below.

Setup: Perses requires a set of credentials and a configuration file as input. This file contains the characteristics of the virtual scenario to be deployed, the tests to be launched and the desired QoS attributes to be evaluated during the test execution. This module checks both the credentials to connect to the cloud infrastructure provider and the configuration file with the different parameters defined in the virtual scenario.

Deployment: The deployment module creates and deploys the entire virtual scenario taking the configuration file as input. To create and deploy the scenario, an abstraction layer is defined that encapsulates Terraform [78]

– a framework with a high-level language that is used to define the deployment infrastructure of an application for cloud providers. Terraform has been extended so that in addition to deploying the cloud infrastructure, it also orchestrates the virtual scenario by installing the necessary resources, creating virtual mobile devices, and deploying distributed mobile applications on those devices. To host and deploy the virtual scenario, Amazon Web Services (AWS) is used as the cloud infrastructure provider. For the creation and deployment of virtual mobile devices, Docker [15] is used, where mobile devices are deployed in containers [79]. The management of these devices and the installation and deployment of the distributed application is performed through an Android Debug Bridge (ADB) [16].

Due to the network infrastructure provided by AWS, data is transmitted directly over the wired network. This is a significant difference to real mobile-cloud communication via WIFI or LTE. Kathara [59], a framework that enables network emulation in Docker containers, has therefore been integrated. This allows us to emulate the network infrastructure of the virtual devices with the cloud environment to replicate real mobile-cloud communications.

Tests Execution: this module executes the tests defined in the configuration file. Perses allows two different types of tests to be run. *E2E tests*, these tests evaluate the QoS attributes of the application by executing the core functionalities that trigger the interactions among different devices. For this, Perses integrates APIPecker [80]– a simple API performance tester where different attributes (concurrent users, iterations, and delay) can be defined to customize the tests, stress the devices, and obtain more information about the QoS. In addition, *user interface tests*, Perses allows developers to run Espresso tests to evaluate the UX and specific traditional functionalities.

Logs Manager: after launching the tests, this module collects the results obtained by aggregating the system logs of each of the virtual devices. After this, it analyses the results and determines whether the desired QoS defined in the configuration file is achieved.

CI Manager: this module integrates with DevOps. This module autonomously manages the execution of the different Perses actions and modules. This makes it possible to automate the entire process of creating and deploying virtual scenarios and all the management related to the launch and analysis of tests. This automation is carried out through a workflow defined with GitHub Actions [81].

Perses needs a configuration file (*.yaml* extension) [17] where the char-

[15]https://www.docker.com/
[16]https://developer.android.com/studio/command-line/adb
[17]https://github.com/slasom/COVID-Heatmaps/

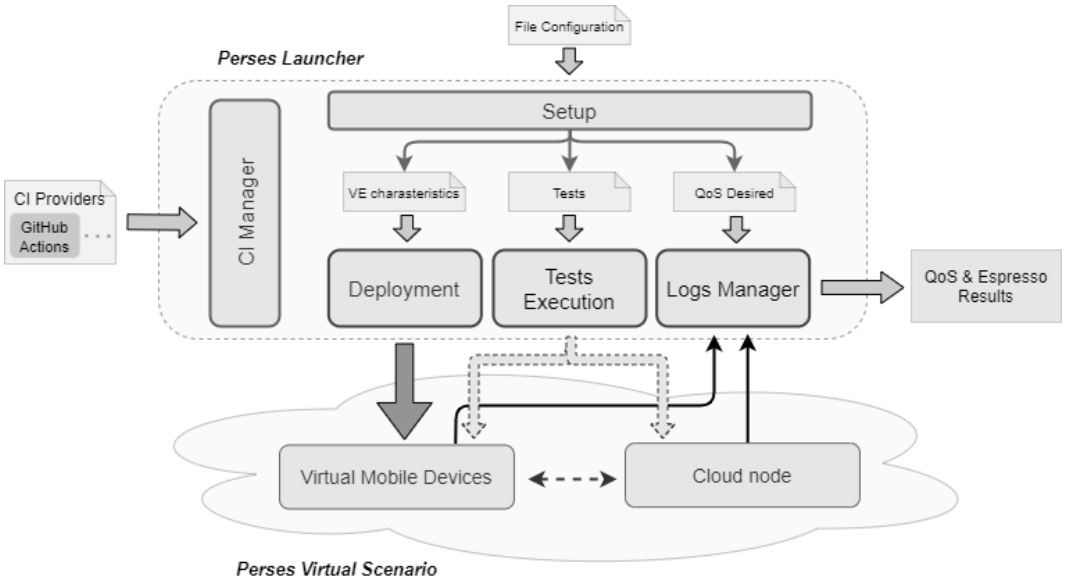


Figure 2.9: General diagram of Perses.

acteristics of the virtual scenario, the tests, and the desired QoS are defined. Figure 2.10 shows a conceptual model with all the parameters in the configuration file. The explanation of each of the parameters and the user guide is detailed on Github [18].

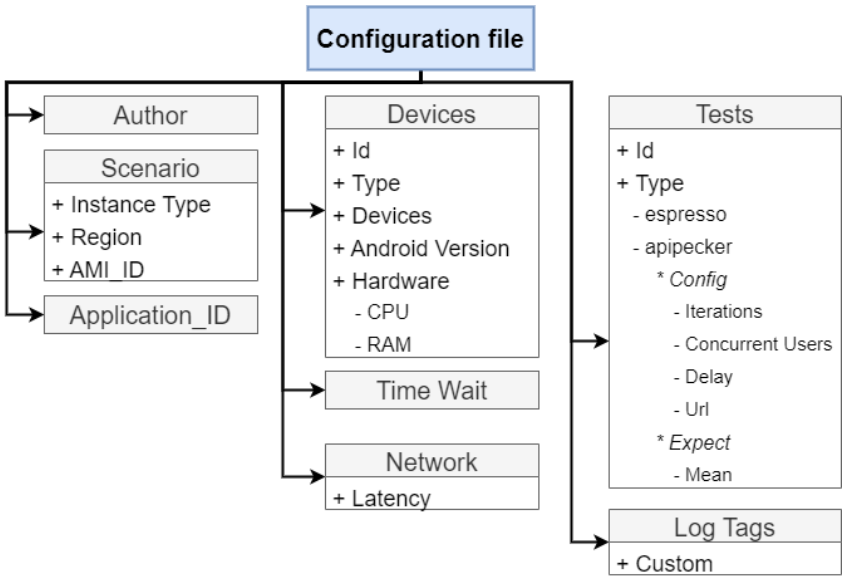


Figure 2.10: Conceptual Model. Configuration File.

[18]https://github.com/perses-org/perses

One of the features of Perses is the full integration into the DevOps methodology. For this purpose, it is currently integrated with GitHub Actions, which makes it easy to automate software workflows applying CI/CD. To perform this integration, it is necessary to define a file with the different steps that the workflow must follow. This workflow has been developed to run any application, i.e., it is not necessary to define a workflow file for each application. Listing 2.2.1 shows an extract of the workflow, and the complete file is available on GitHub[19].

```
- name: "Build Android project"
uses: vgaidarji/android-github-actions-build@v1.0.1
with:
args: "./gradlew assembleDebug assembleAndroidTest"...
- name: "Perses Setup"
run: |
cd.perses_runner
node index -a setup -g../.perses-full.yml -c../.perses-credentials.yml
---
```

Listing 2.2: Perses workflow

Perses evaluation

We evaluate Perses with the development and deployment of a Covid-19 case study. The experiment was carried out with seven physical mobile devices for the real scenario and seven virtual mobile devices for the virtual scenario. To make the test fair, each of the seven devices in each scenario was loaded with a specific set of locations so that the same volume of data was processed and transferred in both scenarios in the different tests.

Figure 2.11 shows an overview of how this *Covid Heatmaps* app works. First (step 1), when a COVID-19-positive user is registered, the mobile application processes its heatmap with the stored traces and sends them to the cloud node. Second, the cloud node delimits the user's movement areas according to the heatmap and sends them to the other mobile devices (step 2). These devices check with their location history to determine whether they have been in the received areas. The devices that have been in these areas process and send their heatmap to the cloud node (step 3). The cloud node compares

[19]https://github.com/perses-org/gha/blob/master/workflow/perses-workflow.yml

the heatmaps of the users with one of the COVID-19 positive, and the users who are considered close contacts are notified (step 4).

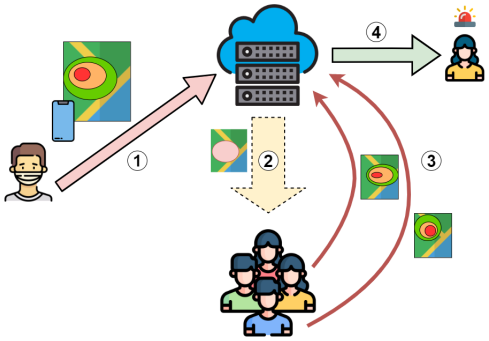


Figure 2.11: Covid-Heatmaps app

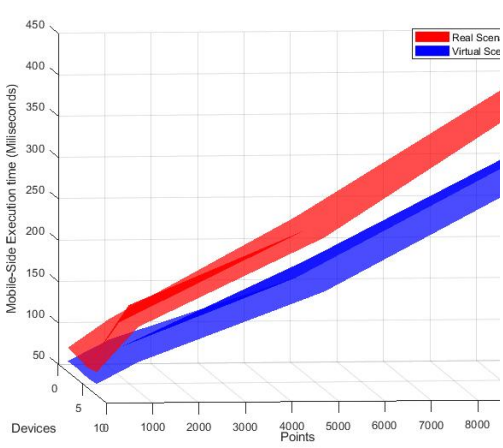
Three QoS parameters are measured that allow us to compare Perses with a real scenario. **Mobile-Side execution time** is the time it takes for mobile devices to process their heatmaps. **Transfer time** measures the time it takes to send the data from the cloud side to the mobile side and vice versa. **Aggregation time** captures the time the cloud spends processing and comparing the heatmaps to detect possible close contacts.

Mobile-Side Execution time: Figure 2.12a shows the computing time on real and virtual mobile devices for the different numbers of devices involved and location points exchanged. It can be seen that they follow the same trend in both scenarios. There is a small gap between both scenarios; this is due to the existing hardware inequality, and the processors of real mobile devices are less powerful due to their size, architecture, etc.

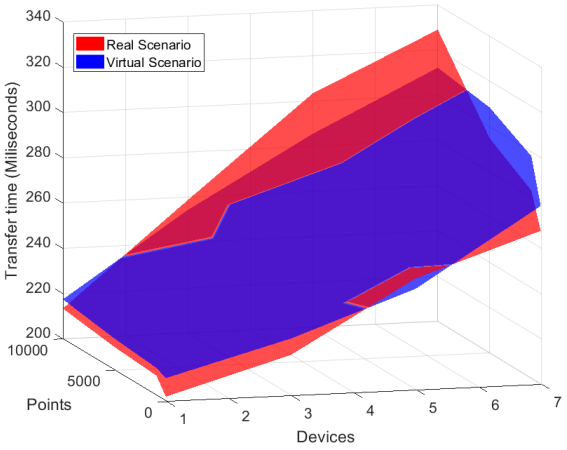
Transfer time: Figure 2.12b shows the results obtained on the data transfer time. As with Mobile-Side Execution time, they follow the same trend with the constant difference between the two scenarios. In this case, the difference is minimal, the integration of Kathara with Perses allows emulation of the communication of real mobile devices.

Aggregation time: Figure 2.12c shows the aggregation time in the cloud node of the heatmaps for the different numbers of devices involved and location points sent. In this case, it can be seen that the planes obtained from both scenarios are practically the same. The cloud node is the same for both scenarios, and the number of devices involved and the volume of data are the same.

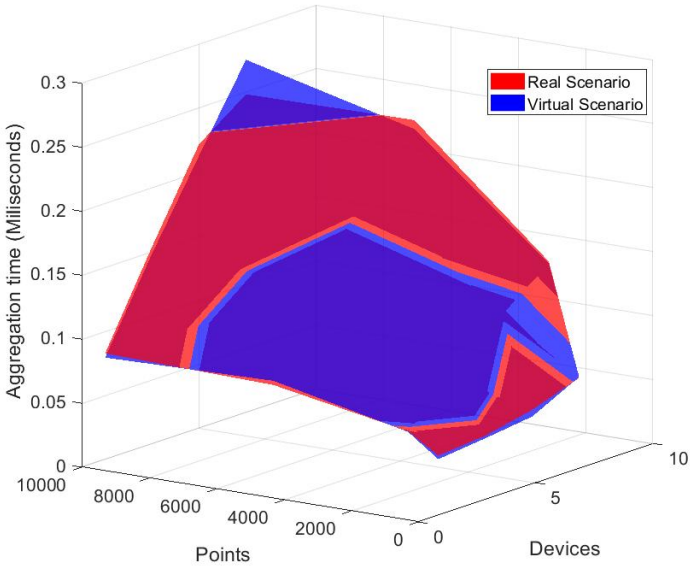
To learn more about Perses, the details of the case study evaluation, and its improvements for the future, we refer the reader to Section 3.4 and [14].



(a) Execution time.



(b) Transfer time.



(c) Aggregation time.

Figure 2.12: Perses evaluation results.

2.3 Operate

In this phase, it involves automating the release of the application and all updates to keep cycles short and give developers more time to focus on product development and accelerate time to market. It allows developers to detect problems quickly and create better versions of software products. In this phase, the operations team is responsible for the configuration and provisioning of the resources needed for the application. The contributions presented here are oriented to the development of applications so that the deployment of their cloud continuity infrastructure can be dynamically adapted to the needs of the application context (e.g. communication quality requirements, data transmission, and quality, etc.).

In this Section, we present two proposals for automating deployment. On the one hand, our proposal on **OPPNets**, focused on a network solution. This solution is based on an opportunistic network algorithm that enables the deployment of communication technology solutions for IoT applications (healthcare, livestock, agriculture, etc.) of the cloud-to-thing infrastructure deployed in rural areas where the communication infrastructure is poor. On the other hand, we present **Elastic Data Analytics**, which behavior can be dynamically modified according to the quality requirements defined by each IoT system and the available resources of the cloud-to-thing infrastructure. For defining and deploying these analytics, we also provide a framework with an orchestrator managing the elasticity. This orchestrator evaluates the state of the infrastructure, the other analytics in progress, and the QoS required by each analytic.

In the following sections, we present two proposals to dynamically adapt the infrastructure deployment depending on the context. On the one hand, our proposal on **OPPNets**, focused on a network solution. This solution is based on an opportunistic network algorithm that allows the deployment of communication technology solutions for IoT applications (health, livestock, agriculture, etc.) deployed in rural areas where the communication infrastructure is poor and can be adapted to different requirements of the nodes. On the other hand, we present **Elastic Data Analytics**, whose behavior can be dynamically modified depending on the quality requirements defined by each IoT system and the available resources of the Cloud-to-thing infrastructure. To define and deploy these analytics, we also provide a framework with an orchestrator that manages elasticity. This orchestrator evaluates the state of the infrastructure, the other ongoing analytics, and the QoS required by each analytic.

2.3.1 OPPNets: An opportunistic solution for remote communications

Communication technologies have experienced an exponential expansion in the last few years. The constant expansion of technology has promoted the improvement of infrastructures and an increase in the number of connected locations. Nevertheless, these encouraging statistics are not the same in all regions. These regions, where Internet infrastructures are minimal, are usually rural and isolated areas where network operators are not interested in deploying their solutions due to the limited benefits they acquire. These circumstances exacerbate the problem. Moreover, along with the technology gap, rural areas face additional challenges such as healthcare. Given this context, rural areas lacking communication infrastructures face difficult challenges. Specific needs and technological isolation motivate the diffusion of alternative technologies that explore mechanisms for information transmission and system communication.

Rural areas are full of entrepreneurial activities such as livestock and agriculture. There are also other important activities such as healthcare by a significant percentage of people over 65 years old. They are often isolated by network operators because they are not interested in deploying their solutions due to the low benefits they acquire. Today, technological solutions for deploying applications in the Cloud-to-thing continuum for industry and healthcare have become the main disruptive solutions to improve production and people's quality of life [41]. However, in rural areas where there is no access to the Internet, these solutions cannot be easily deployed. Specific needs and technological isolation motivate research into alternative technologies that explore information transmission mechanisms and communication systems for the proper deployment of infrastructures in these rural environments.

We propose a solution based on an opportunistic network algorithm that enables the deployment of communication technology solutions for Cloud-to-thing continuum infrastructures in isolated areas where connectivity is limited. Thus, two applications are proposed: a presence detection platform for elderly people living alone and an analytical performance measurement system for livestock. The algorithm is evaluated by considering several simulations under multiple conditions, and comparing the results of delivery probability, latency, and overhead with other well-known opportunistic routing algorithms. The approach exploits the use of SACAR OCVN [82], a routing algorithm based on the adaptation of requirements between different nodes. Specifically, this algorithm provides the mechanism to dynamically adapt communication depending on the type of data and the receiver's interests.

The network behaviour follows the scheme provided in Figure 2.13,

which identifies several components: i) sender nodes, ii) intermediate nodes and iii) gateways.

- **Sender nodes** refer to the homes of the elderly people and the smart livestock. These elements generate and transmit new information toward gateway nodes that are connected to the Internet. Thus, the messages are sent into the network in specified intervals of time. This way, the nodes send the messages to reachable intermediate nodes, which serve as data mules.
- **Intermediate nodes** are in charge of forwarding messages to the gateway. Therefore, cars and pedestrians receive, carry and deliver the information as they move through the streets and roads. These nodes decide the information they prefer to obtain and broadcast: presence info or livestock performance data. Additionally, there is another fundamental element in the scenario: throwboxes. These devices are placed on the main points of the road path and can store messages from any other intermediate node. Thus, throwboxes work as a "meeting point" for data and distribute it to other interested reachable nodes.
- **Gateways** are the destination elements of the messages generated by the sender nodes. They are connected to the Internet and are in charge of processing and transmitting information to the Cloud. As a result, the collected data can be externally processed, enabling the detection of anomalous patterns in the elderly activity and recognizing a possibly dangerous situation. In the same way, the performance information about livestock can be processed when it is finally delivered.

The opportunistic behaviour of the network is mainly based on the routing algorithm we propose to perform distributed communication among the aforementioned elements. In this paper, the idea behind the SACAR OCVN algorithm defined in [82] is applied to the rural scenario in order to rely on cooperating nodes to successfully deliver information from elders and livestock. The behaviour of the nodes in the scenario is based on the preferences of the nodes.

SACAR OCVN [82] automates the nodes' encounter process. Thus, the nodes in the scenario have a virtual profile that identifies them as sender nodes, intermediate nodes, or gateways. The different roles available on the virtual profile are Goals and Skills: the first one defines the information preferences, while the Skills indicate the actions the node is able to perform. This model, once it is adapted to the scenario, is used to distinguish between sender

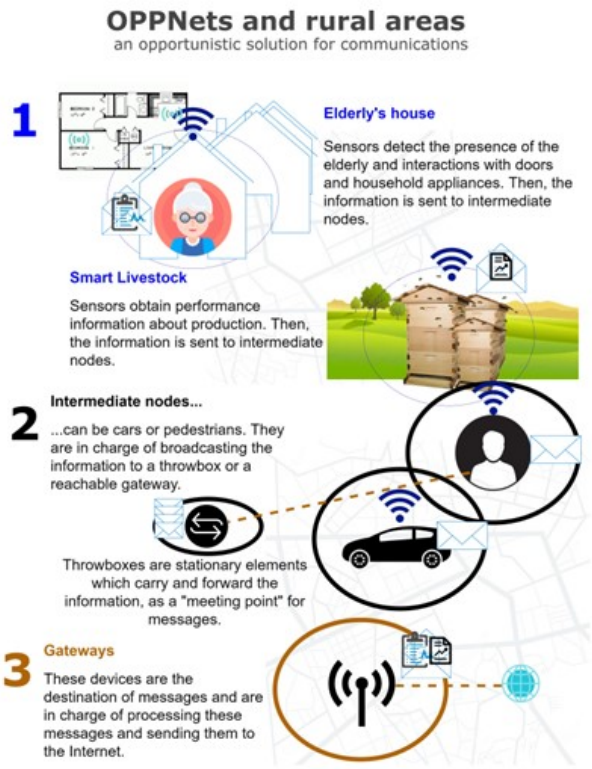


Figure 2.13: Working scheme of OPPNets in rural area.

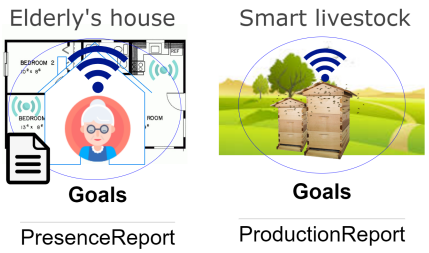


Figure 2.14: Virtual profile of sender nodes.

elements (represented in Figure 2.14), carrying nodes (depicted in Figure 2.15) and end gateways (Figure 2.16).

The communication takes place when two nodes encounter each other. This way, by using low-energy technologies, the information is exchanged within the corresponding range. Thus, the communication process follows three main steps: 1) devices encounter each other, 2) virtual profiles are gathered and 3) information is exchanged. All these steps are shown in Figure 2.17. The proposed scenario raises an opportunistic context where communication provides an improvement of the day-to-day routines of local inhabitants. The

Figure 2.15: Virtual profile of intermediate nodes.

Figure 2.16: Virtual profile of gateway nodes.

idea has been executed as a simulation that has tested the algorithm under different situations, trying to define the most appropriate context for interactions.

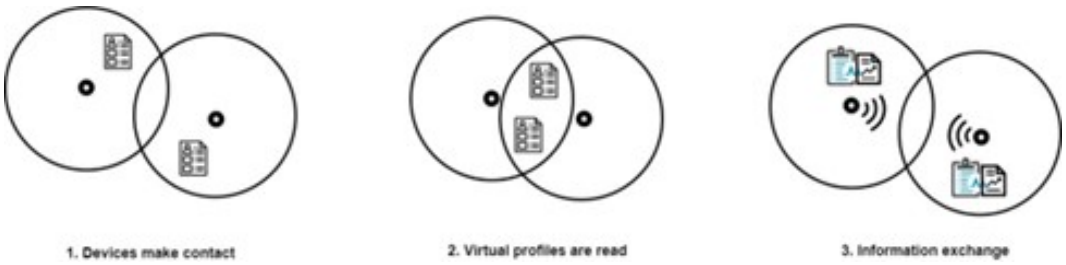


Figure 2.17: Detailed communication process.

SACAR CPVO Evaluation

We evaluated SACAR CPVO with positive results in simulated scenarios. Simulations have been carried out using The ONE simulator [83]. This tool provides a development environment for DTNs, allowing the specification of a detailed scenarios, movements, and protocols. Thus, the performance of the routing algorithm can be obtained through execution reports. In our approach, the simulated scenario recreates a rural area with low connectivity, which re-

flects a village community and the surrounding roads. Figure 2.18 represents the simulation map.

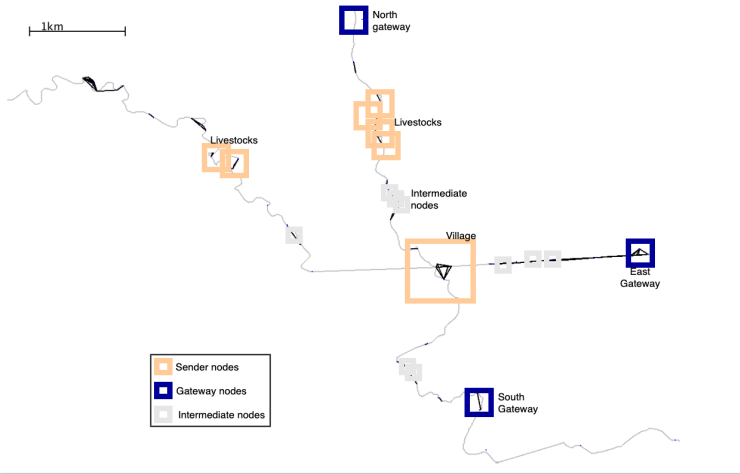


Figure 2.18: Rural scenario simulated in The ONE.

We obtained positive results. The delivery probability ($d_{\text{prob}} = 0.945$) was very high, guaranteeing a high rate of message broadcasting. This successful result means that the algorithm provides a good communication mechanism for the exposed scenario. As a consequence, the data on the presence of the elderly's homes is fluently transmitted through the network. Thus, the system allows the remote detection of possible dangerous situations, derivated from the inactivity of the elderly. Besides, the local industries can transmit performance data from exploitation.

Once SACAR OCVN best results are obtained, they are compared under the same context conditions as the rest of the DTN algorithms included in The ONE. DirectDeliveryRouter [84] works based on straight delivery between the sender and the receiver node. Thus, intermediate elements are ignored. EpidemicRouter [85], on the other hand, belongs to the flood algorithm family. Thus, the behaviour is based on duplicating the messages with every encountered node. MaxPropRouter [86] uses the previous node encounters to define the most appropriate path to a destination. ProphetRouter [87] works based on a probabilistic scheme while SprayAndWaitRouter [88] replies to messages by creating copies that can be specified by the user.

In the comparison process clearly showed the performance of SACAR OCVN compared with the opportunistic routing algorithms included in The ONE. The high delivery probability value guarantees the applicability of the solution in the described scenario, becoming a powerful tool in the implementation of the project.

To learn more about our proposal and to visualize all the results obtained in the performance evaluation and comparison with other algorithms, we refer the reader to Section A.6 and [42].

2.3.2 Elastic Data Analytics

Over the last few years, there have been a massive deployment of Internet of Things (IoT) devices as we have discussed above. This deployment has been especially driven by IoT applications that facilitate everyday tasks for every individual and organization. These tasks cover a wide range of use cases and applications, from managing an individual's personal health to understanding the flow of people and movement patterns in a city. This massive deployment of IoT devices has also led to the creation of networks of smart devices. This growing use of IoT devices is putting stress on current infrastructures in different dimensions.

On the one hand, the amount of devices deployed. As IoT applications become more complex they require combining devices from different domains for offering more useful functionalities. So far, new IoT applications and functionalities require the deployment of new devices. However, in this growing trend, the deployment of new IoT devices for sensing similar data is becoming unfeasible and unsustainable. For instance, a city that monitors the movement patterns of its citizens may also want to combine the location information with the heart rate in order to know what kind of activity citizens tend to do in each area of the city and, thus, provide more useful services and better plan their investments. For that purpose, instead of asking citizens to wear new heart rate monitors, it would be more feasible to ask them to share the data obtained by the monitors they already wear for other apps. Thus, in the coming years, we will see how IoT applications will share large networks of devices already deployed.

On the other hand, the infrastructure for data circulation and processing. Data, especially those coming from IoT devices, has become a strategic asset because of their availability to continuously monitor the environment without human intervention. IoT device networks are enablers of a new data economy in which more and more data is being exchanged not only within but also amongst companies [89]. This fact boosts the massive deployment of IoT device networks. However, it also entails an increase in the circulation of information and the need for its processing and, consequently, a management model to support optimal network usage. Such requirements will be even harder when different information systems require data from the same devices, for different needs and with different qualities. For example, a system

to monitor an individual's heart rate during her activities may require a very high information freshness; however, for a municipality to plan its services, such information does not have to be very fresh, but it has to be obtained from as many devices as possible.

In order to reduce the network overhead and improve the quality of service (QoS), IoT applications already attempt to take advantage of the cloud-to-thing continuum to deploy services closer to information providers and consumers. However, such applications are still isolated systems that do not favor the sharing of data or analytics streaming. Thus, a sustainable data economy in this market [90] poses many challenges, among which we can count the following:

- First, the shared use of IoT devices so that data they are sensing can be used by different applications minimizing the drain on resources.
- Second, the optimization of the use of the cloud-to-thing continuum infrastructure by enabling optimized deployment so IoT applications interested in the same data streaming and analytics can be deployed as close to the data as possible and together in the same nodes of this continuum.
- Third, elimination of duplicate streams by enabling applications interested in the same data or analytical streams to use the same datum.

We address these challenges by using elastic IoT data analytics, which behavior can be dynamically modified according to the quality requirements defined by each IoT system and the available resources of the cloud-to-thing infrastructure.

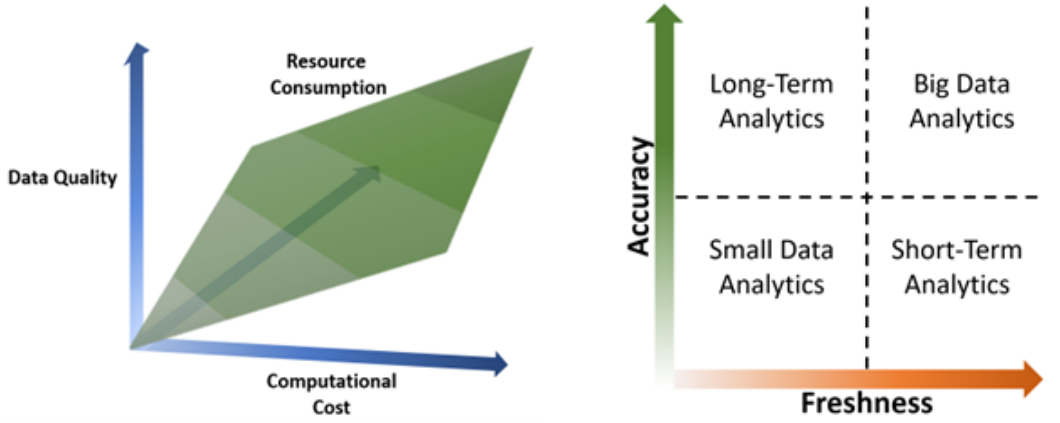
The execution of these analytics impacts several highly related dimensions, as Figure 2.19 shows, that must be controlled in a sustainable computing and information environment: data quality, resource consumption, and cost.

One of the most important aspects for data consumers is the quality of the information. This quality must be appropriate for the specific functionalities they want to provide. While there are different properties to measure data quality, for information coming from IoT devices two properties are particularly relevant [91]:

- Accuracy, which is the level to which data represents the real-world scenario. In this sense, in a network of IoT devices, the higher the number of devices involved in the data analytics the better they will represent the real world.

- Freshness, or the timeliness of the data. For some applications, the data has no value if it is not available at the right moment. Freshness indicates the frequency at which IoT devices have to provide information for the analytics to provide value to the consumer.

Combining both dimensions, as Figure 2.19 shows, different types of analytics can be executed. When the accuracy and the freshness are low, small data analytics are executed in order to have a limited quantity of highly granular data that usually provides valuable information for the system. As the freshness and the accuracy increase, bigger data analytics are executed focusing on processing large volumes of information for business decisions. Instead, if only the accuracy increases, long-term data analytics are executed for predictive decision-making processes. In addition, if mainly the freshness is important for achieving acceptable data quality, short-term data analytics are usually executed for developing reactive processes. For instance, for our case study, the healthcare system would require short-term analytics while the city council would require long-term analytics.



(a) Dimensions impacting the analytics

(b) Analytics depending on the data quality

Figure 2.19: Elastic Data Analytics.

Elastic Data Analytics validation

To achieve elastic data analytics, we propose a framework orchestrating the analytics required by different data consumers and leveraging the cloud-to-thing continuum to distribute them depending on the available resources and

the required quality. The source code of the framework [20] for the presented case study and a video[21] showing the achieved elasticity are publicly available.

Figure 2.20 shows an example of the infrastructure that can be used for our smart city case study. It shows the different layers of the cloud-to-thing continuum. Please, note that we limited the number of layers to improve readability. Data consumers (healthcare system and city council, for our case study) can request different analytics through the entry point to the infrastructure, the cloud. The deepest layer is composed of the end devices sensing and sending information to the cloud. This information goes through the fog and edge nodes on its way to the cloud where it is provided to data consumers.

Data analytics can be requested with different parameters. First, the application-specific parameters (for instance, the area to monitor in the smart city) and, second, the data quality parameters (accuracy and freshness). For the current implementation, the accuracy and the freshness can take different values from a range (Low, Medium, and High). The accuracy relates to the ratio of end devices involved and the freshness to the frequency at which the information is obtained.

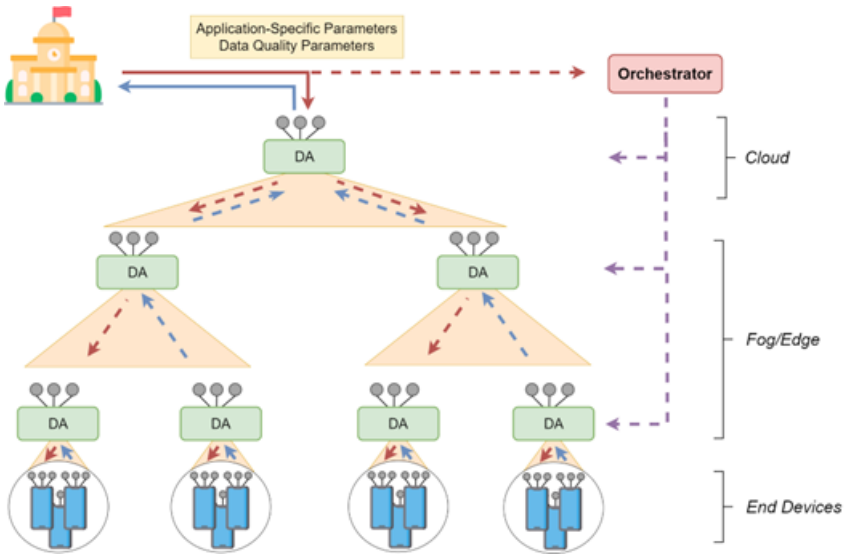


Figure 2.20: Architecture of the elastic data analytic framework.

Figure 2.20 also defines the most important components of the proposed framework. First, Cloud, Fog, and Edge nodes are composed of a Data Aggregator component (DA). This component processes the deployed analytics. To that end, it requests the required information to the lower nodes depending

[20] https://doi.org/10.5281/zenodo.5793214
[21] https://doi.org/10.5281/zenodo.5793189

on the data quality (mainly the freshness) required by the most demanding analytics, caches the obtained data to be reused by the other analytics, and processes it. In addition, the obtained results are also cached and always available, waiting for the higher levels to request them.

In addition, an elasticity orchestrator is proposed to balance the load of the whole infrastructure, modify the behavior of the elastic analytics depending on the previously defined dimensions, and provide the best QoS to all parties. When a new analytic is required, it is analyzed by the orchestrator to evaluate the desired quality (accuracy and freshness), the analytics already deployed in the architecture, and the workload of the different nodes. If there are enough resources, it directly deploys it providing the highest possible quality. If there are not enough resources, it checks if the analytics already deployed can be reconfigured to make room for the new one. This reconfiguration can be provided by redeploying existing analytics on other nodes to have a more efficient distribution or, if possible due to the quality defined by consumers, reducing the accuracy and freshness of some analytics to free up some resources.

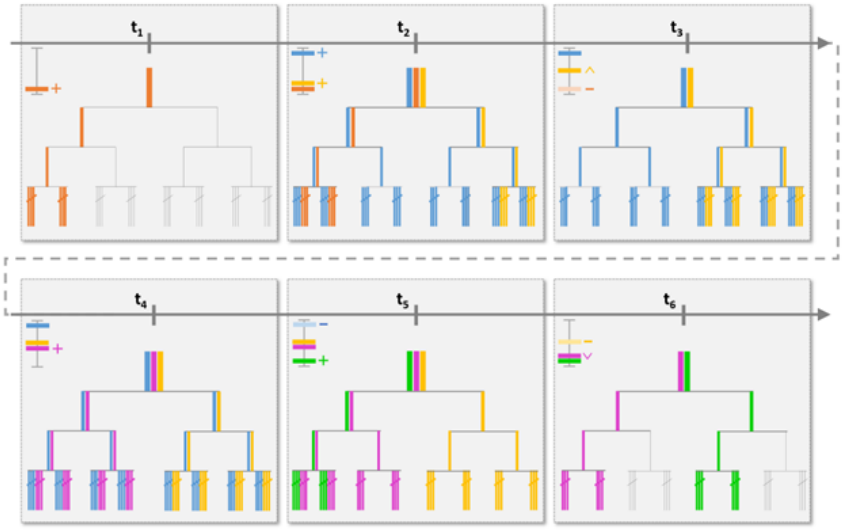


Figure 2.21: Execution of elastic data analysis.

Figure 2.21 shows a working example of the provided elasticity. The example consists of a timeline with 6 instants (t). At each instant, an action is triggered that has an impact on the infrastructure. At the top left of each instant, a small legend appears with a vertical bar. This legend indicates for each analytic (each colored horizontal bar) if it is inserted/removed from the system (+/-) and, depending on its height in the vertical bar, the accuracy that it will provide. Please, note that we only represent the accuracy dimension to improve readability. At *t1*, the orange analytic is requested, requiring

low accuracy. The orchestrator, therefore, assigns it to only a part of the infrastructure. At *t2*, two new analytics (yellow and blue) are inserted, with low and high accuracy respectively; the orchestrator deploys the blue analytic throughout the whole infrastructure while the yellow one is deployed only in the less loaded part. At *t3* the orange analytics ends and, because there are enough resources, the orchestrator upgrades the yellow one to medium accuracy. At *t4* the purple analytic is inserted and the orchestrator again assigns it the lowest loaded part of the infrastructure. At *t5*, the blue analytic ends, and a new one with a low accuracy is deployed. Finally, at *t6*, the yellow analytics ends, and the rest are reorganized to better distribute the resource consumption.

To learn more about our approach of Elastic Data Analytics, we refer the reader to Section 3.5 and [43].

"Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less"

Marie Curie (1867-1934)

Chapter 3

List Of Publications

This chapter presents the publications related to this thesis made during the PhD. On the one hand, we show the relevant publications that make up this compendium thesis. These are full articles published in high-level journals indexed in JCR. The full text of these publications can be found in Sections 3.4, 3.3, and 3.5. On the other hand, we also show the supporting publications of this thesis, i.e. full papers, demonstrations, and artifacts published in international/national conferences, international workshops, and other journals that have given rise to the relevant publications or have served as support for this thesis as a whole. The full texts of the supporting publications can be found in Appendix A.

Contents

3.1 Relevant Publications

This section shows the most relevant publications resulting from research during the PhD. Table 3.1 shows the list of relevant publications produced during the PhD. As can be seen, all of them are articles published in JCR-indexed journals.

Title	Journal	Publication date	Quality indicator	Section
Perses: A framework for the continuous evaluation of the QoS of distributed mobile applications	Pervasive and Mobile Computing	June 2022	JCR Q2	3.4
Human microservices: A framework for turning humans into service providers	Software: Practice and Experience	April 2021	JCR Q2	3.3
Elastic Data Analytics for the Cloud-to-Things Continuum	IEEE Internet Computing	December 2022	JCR Q2	3.5

Table 3.1: Relevant Publications

3.2 Supporting Publications

In addition to the most relevant publications presented above, our research has also led to further publications which, although not published in relevant journals or conferences, have been equally important for the progress of the research of this PhD, being works that have subsequently led to the relevant publications or have supported the whole research.

Table 3.2 lists the supporting publications of this thesis. As can be seen, we have published three papers in two other JCR-indexed journals, four tool and artifact demonstration papers (one of them indexed in the GGS conference ranking), and one full paper at an international conference (also indexed in the GGS conference ranking). Also two full-length papers in international workshops. Finally, we have also published three papers at national conferences, including two demonstrations (where in one of them we won the award for the best demonstration of the conference) and one full paper.

Title	Conference / Workshop / Journal	Publication date	Publication type	Quality indicator	Appendix
Providing Support to IoT Devices Deployed in Disconnected Rural Environment	International Workshop on Gerontechnology	November 2019	Full paper		A.1
Yourpantry: food monitoring through pantry analysis using the smartphone and making use machine learning and deep learning techniques	International Workshop on Gerontechnology	November 2019	Full paper		A.2
Deployment of APIs on android mobile devices and microcontrollers	International Conference on Pervasive Computing and Communications	March 2020	Demo and artifact paper	GGS class 1 (A+)	A.3
SOLID and PeaaS: Your Phone as a Store for Personal Data	International Conference on Web Engineering	June 2020	Full paper	GGS class 3 (B-)	A.4
FoodScan: Food monitoring app by scanning the groceries receipts	IEEE Access	December 2020	Full paper	JCR Q2	A.5
OPPNets and Rural Areas: An Opportunistic Solution for Remote Communications	Wireless Communications and Mobile Computing	January 2021	Full paper	JCR Q3	A.6
Virtual environment for evaluating the QoS of distributed mobile applications	International Conference on Pervasive Computing and Communications	March 2021	Demo and artifact paper		A.7
Pushed SOLID: Deploy SOLID in Smartphones	Mobile Information Systems	July 2021	Full paper	JCR Q4	A.8
Perses: Un framework para evaluar la Calidad de Servicio en aplicaciones móviles distribuidas	Jornadas de Ciencia e Ingeniería de Servicios	September 2021	Demo paper		??
Una Propuesta para la Composición de APIs Distribuidas	Jornadas de Ciencia e Ingeniería de Servicios	September 2021	Full paper		A.9
Sistema IoT para la Prevención de Riesgos Laborales en una EDAR	Jornadas de Ciencia e Ingeniería de Servicios	December 2022	Demo paper		A.10

Table 3.2: Supporting Publications

3.3 Human microservices: A framework for turning humans into service providers

Authors: *Sergio Laso, Javier Berrocal, José García-Alonso, Carlos Canal and Juan Manuel Murillo.*

Publication type: *Journal paper.*

Journal: *Software: Practice and Experience.*

DOI: *10.1002/spe.2976.*

Quality indicator: *JCR Q2.*

Received: 29 July 2020 | Revised: 3 March 2021 | Accepted: 17 March 2021

DOI: 10.1002/spe.2976

SPECIAL ISSUE PAPER

WILEY

Human microservices: A framework for turning humans into service providers

Sergio Laso[1] | **Javier Berrocal**[1] | **José García-Alonso**[1] | **Carlos Canal**[2] | **Juan Manuel Murillo**[1]

[1]Department of Computer and Telematic Systems Engineering, University of Extremadura, Caceres, Spain

[2]Department of Computer Science, University of Malaga, Malaga, Spain

Correspondence
Javier Berrocal, Department of Computer and Telematic Systems Engineering, University of Extremadura, Avda. de la Universidad s.n., 10003 Caceres, Spain.
Email: jberolm@unex.es

Funding information
Interreg V-A España-Portugal (POCTEP), Grant/Award Numbers: 4IE+, 0499-4IE-PLUS-4-E; Department of Economy, Science and Digital Agenda of the Government of Extremadura, Grant/Award Numbers: GR18112, IB18030; European Regional Development Fund; FEDER / Junta de Andalucia, Grant/Award Number: UMA18-FEDERJA-180; MCI / AEI / FEDER/ UE, Grant/Award Numbers: PGC2018-094905-B-I00, RTI2018-094591-B-I00; RCIS research network, Grant/Award Number: RED2018-102654-T

Summary

During the last decade, the mobile application market has grown steadily thanks to the massive use of smartphones and the emergence of cloud computing for offloading computation tasks and improving the quality of experience. With the more recent deployment of Internet of Things (IoT) devices, this cloud-based architectural design and the corresponding communication flow has been maintained. Nevertheless, the increasing amount of information exchanged, the stringent requirements of many IoT applications, and the need for these applications to adapt their behavior in real time to the user's context set these architectural assumptions a challenge. Paradigms such as mobile, mist, and edge computing have recently been proposed to exploit the computational and storage capabilities of current smartphones and IoT devices in order to onload some tasks onto them, reducing the overhead on both the cloud and the network. Currently, the application of these paradigms requires much attention from skilled developers to create ad hoc systems, as there lack standards and tools facilitating their use. This communication introduces Human Microservices as a framework facilitating the deployment of APIs on companion devices in order to provide personal and updated information that can be consumed by other entities. The framework improves the integration of humans in the IoT loop and facilitates the deployment of computation units in devices closer to end users, enhancing system response time by reducing the stress on cloud and network infrastructure. The proposed framework is based on existing standards in order to improve software quality and shorten the learning curve.

KEYWORDS

end devices, human in the loop, microservices, mist computing, mobile computing

1 | INTRODUCTION

The mobile phone market has grown to levels unsuspected a few years ago.[1] The combination of this growth together with the increasing importance of the cloud computing paradigm has led to a change in how information is produced and consumed through smartphones. Currently, most mobile applications store and access the information that their users request by invoking services deployed in cloud environments.

This information flow has increased during the last few years with the spread of the Internet of Things (IoT).[2] With the aim of adapting the physical environment to the users' needs and preferences, most IoT devices consume information produced by other devices, usually nearby, while this information is stored and provided by microservices[3] deployed far away, in the Cloud.

The client–server architecture is the design behind all these systems.[4] In this, the information generated by end devices is sent to be stored and processed in large servers to be later consumed by the same or other devices.[5] Indeed, the client–server architecture has several advantages. First, it integrates devices that do not have a lot of computing resources,[6] and second, complex applications can be built by dividing them into small pieces (i.e., microservices) that can be deployed, scaled, and maintained with relative ease and independence. In addition, there are standards, such as OpenAPI,[7] for designing and documenting APIs (sets of microservices) that can later be used to generate source code, saving development time.

Nevertheless, this architecture is not suitable for every application. IoT devices and mobile applications are starting to require more specific, personalized, and up-to-date information to adapt their behavior to the current context of each user. Paradigms such as Human-in-the-Loop or User-in-the-Loop[8] promote services or applications centered on people and awareness[9] of their context.

Using a pure client–server architecture for applications that require intense peer-to-peer or local communication generates additional stress due to the mandatory passage of data through cloud servers and the network infrastructure.[10,11] Paradigms such as mobile, mist, or edge computing,[12-14] among others, have been proposed to exploit the capabilities of nodes closer to end users to store data and execute different computing tasks with the aim of improving the quality of experience (e.g., the end-to-end performance at the service level from the user's perspective[15]) of these applications. Mobile and mist computing focus on exploiting the capabilities of end devices (i.e., smartphones, IoT devices, etc.). Edge computing focuses on the nodes at the edge of the network close to the end devices. These paradigms are especially useful for storing local, personalized, and fresh information (such as the context, or the virtual profile, of the users) that is consumed by nearby devices.[6]

However, developing applications based on such paradigms require developers skilled in different areas (hardware, communication, operating systems, etc.). While cloud providers and other companies provide some standards and technologies abstracting from the low-level details to develop, deploy, mashup, and consume APIs, there are few similar mechanisms and technologies for computational infrastructures closer to the end user (smartphones, IoT devices, wearables, etc.). Using precarious and ad hoc mechanisms directly implemented by each developer to develop such applications can affect the development time, maintainability, and reproducibility of the software.

In this paper, we introduce the concept of Human Microservices, and present a framework based on standards for their development. Human Microservices turn humans, more particularly, their smartphones and companion devices, into service providers. They provide accurate and up-to-date information about people (their owners) by means of the built-in sensors or the virtual profiles built and stored in their companion devices. These microservices can be directly consumed by other Internet-connected devices as if they were deployed in the cloud via APIs but without having to store all that information in a remote environment, thus reducing stress on the cloud and network infrastructure, and improving the quality of experience. It also allows for greater control over the data, thus improving the privacy of users.

We claim that a uniform deployment of APIs on end and companion devices is essential for the integration of humans in the IoT loop. We also present tools developed to make the application of the underlying concepts of Human Microservices easier for developers. This framework is built upon existing specifications and tools for cloud environments, so that the learning curve is shortened. In addition, the reuse of standards makes the applications generated easier to maintain and upgrade.

The rest of this communication is structured as follows. Section 2 motivates the need for human microservices using an example scenario. Section 3 explains in detail how to develop and deploy APIs on mobile devices using the framework developed. Section 4 details two validations of the framework, on the one hand, implementing the example scenario, and, on the other, surveying a group of developers, and students trained in the framework. Section 5discusses several of the benefits of Human Microservices. Section 6 gives a comparison with some relevant related work. Finally, the conclusions and lines for future work are presented in Section 7.

2 | MOTIVATION

Smartphones have become the main device for the consumption of information such as multimedia, social networks, or news.[1] In addition, with the massive deployment of IoT devices, mobile devices are also taking on the role of the control

centre for all of them. Through mobile devices, one can control and get information from almost any IoT devices owned by the user (from temperature sensors to smart wristbands) by just using the applications offered by the manufacturers.

Most of these applications are based on a pure client–server architecture in which mobile devices act as simple clients (or as the gateway of IoT devices). They get the information (if required, from the IoT devices) and post it to the cloud by invoking different microservices. These applications are usually designed using the service-oriented computing (SOC) paradigm,[16] allowing developers to divide their functionalities into small and maintainable modules focused on specific features. Thus, these modules or microservices can be developed, deployed, modified, and redeployed without compromising other functional aspects of the application, improving the quality, maintainability, and scalability of the application.

During the last few years, the number of IoT and mobile devices has increased considerably,[1,17] leading to exhaustion of the computing capabilities of cloud environments. In addition, their own mutual interactions have also been increasing. These devices require information about the user's context and preferences in order to coordinate themselves and adapt their behaviour in real time. This collaboration requires the exchange of large volumes of data and the processing of different tasks which again leads to increased stress on the cloud servers and lower quality of experience.

Mobile and mist computing alleviate the stress of these environments by distributing computation tasks onto end devices (IoT devices, smartphones, etc.). Thus, since the distance between the consumer and the provider of information is reduced, the response time and the stress on the cloud and network infrastructure is usually also reduced. However, these paradigms are in their infancy, and the technological support for developers is very limited. For instance, smartphones have closed and security restricted operating systems, so that microservices cannot be directly deployed and provided from them. In addition, these devices are mobile, so that they are constantly changing from one network to another, leading to intermittent connections and changes of IP address. Therefore, traditional tools and standards cannot be used to deploy microservices on these devices.

For mist computing, there are currently some solutions[18-20] for the development and deployment of IoT applications on these devices. Two examples are storing and sharing ECGs in health care scenarios,[18] and replicating the sensed data among different IoT devices.[20] Nevertheless, again they are adapted and personalized to a specific scenario—there are no standard mechanisms and technologies for developing, deploying, and consuming services as there are for cloud environments. Therefore, developers have to use all their skills and knowledge to design and implement ad hoc infrastructures to distribute IoT applications. This firstly increases development time and cost, and secondly reduces maintainability.

2.1 | Human Microservices for smartcities

In order to better illustrate the problem being addressed, let us describe a couple of scenarios focused on smartcities.

Many cities are currently deploying IoT devices and mobile applications to gather information about their residents and the environment, with the goal of providing better services to those residents. For instance, Figure 1 shows an

FIGURE 1 Emergency alert system for smartcities

emergency alert system that warns citizens if there is an alert situation in the city (health alert, earthquake, or fire). Such a system, similar to Facebook Safety 1, uses citizens' information (such as location, vital signs, or health records) to identify people who are at greater health risk to create more efficient evacuation plans. A more innovative functionality also allows other citizens to quickly identify the location and health status of their neighbors in order to help them in the event of a disaster.

A different sample scenario would be a tourist information system in which municipalities obtain information on people's location to identify which places in the city are most frequented and thus determine if it is necessary to improve their accessibility by public transport, to increase cleaning services, etc. Ideally, it would be very interesting if this information could also be consumed by other tourists in order to detect crowded points of interest, and better plan their routes. One could also think of a crowd sensing system in which users can interact with other users, for example, sending questions to know whether some place is still open.

We shall focus on the emergency system as an example running throughout this communication to show the benefits of the proposed framework. This information system could be implemented using a pure client–server architectural style. The mobile application provided by the smartcity would only have to get the required information and upload it to the cloud. There are several tools facilitating this development. For example, OpenAPI can be used to design the API and generate its skeleton; with AWS, one can define microservices as Lambda[21] functions and deploy them on a cloud.[22] However, this architecture has some disadvantages that lessen the usefulness of the system. Dynamic contextual information about each citizen has to be constantly posted to the cloud (such as their location, vital signs, etc.) in order to be useful for creating the evacuation plan. This constant data flow would increase the network overhead (and hence the latency and response times) and the resource consumption of mobile devices,[6] affecting the user's satisfaction and quality of experience. In addition, a huge amount of information would have to be stored in the cloud and processed to provide useful information to the different services, thus increasing the operating costs. For example, if an emergency coordinator wants to obtain the location of the citizens affected by an emergency, all the locations have to be processed. But if, after that, they want to get those who also have some mobility problem, this requires the information to be processed again, wasting more computational resources and, which is more important, lengthening the response time. Finally, the storage of people's sensitive information in a cloud environment may pose a risk to privacy, since users usually do not have control over who consumes that information and when.

Instead, in order to address these problems and to reduce the cloud workload, we may rely on the mobile, mist, and edge computing paradigms in which some tasks are distributed to users' devices, generating closer and more direct communication, also following the same service-oriented architecture that improves the software management and maintenance. If the emergency system is migrated to this architecture, all the information gathered (e.g., the exact location of the user) would be directly stored in companion devices, or even not stored at all, being obtained at run-time only when needed. Figure 1 shows this system and how the information is provided and consumed. Citizens would then have an API deployed on their devices with different microservices that either provide very specific information, both static (such as personal data, health data, etc.) and dynamic (location, mobility history, tastes and preferences, etc.), or that allow the citizens to receive alerts, warnings, etc. Access to the user's data would be determined by the microservices offered, allowing it to be far more personalized and individually adapted since, for instance, only the microservices of the citizens affected by a specific emergency would be invoked and consumed. This would impact positively on latency, response times, operating costs and stress on the cloud and network infrastructure since there is no information constantly being sent to cloud environments, only the data consumed in the microservices actually involved. In addition, it would allow users to have greater control over the data stored and consumed from their mobile devices.

However, development of such a solution requires a great effort on the part of developers since there are currently no tools supporting the design, specification, and implementation of these services on end devices. This means that developers have to work on creating the system they envision with the skills, utilities, mechanisms, etc. that they themselves know, thus generating very personalized code particularly adapted to the system they are developing, which will be hard to maintain and to evolve in the future.

For this reason, we propose Human Microservices as a way to integrate humans into the IoT loop, providing a framework in which they can be developed more rapidly and easily. The tools involved are capable of generating these microservices using the same technology that developers are currently using for cloud environments, thus shortening their learning curve, and promoting and favouring the use of SOC architectures and tools. In addition, Human Microservices is a framework that facilitates the application of the mobile, mist and edge computing paradigms, allowing to easily deploy computation services in devices closers to the end user.

3 | HUMAN MICROSERVICES

Human Microservices is a framework based on the People as a Service (PeaaS)[14] and the Internet of People (IoP)[23] models previously proposed by the authors of this paper. Both models identify smartphones as the core elements in integrating people into the Internet. Indeed, smartphones are currently people's communication interface for interacting with other people or with IoT devices. But smartphones are usually just mere consumers of services deployed on cloud environments. They obtain information from their users through sensors and other connected devices, and store it in the cloud, acting as simple gateways. This leads to increased resource consumption and limits the users' control over their data and its privacy.

PeaaS and IoP encourage smartphones to leverage their full computing power to store the user interactions and all the data gathered in order to create a virtual profile of their user. This profile could then be consumed directly from the smartphone by other IoT devices, thereby reducing network overhead, response time, and operating costs.[6] Furthermore, it empowers users to manage their personal information and its privacy.[24]

Building on our previous work, we focus with Human Microservices on providing developers with techniques, based on existing standards and tools, that can reduce the effort required to develop APIs through which to offer the users' virtual profile to other entities. In turn, this improves the user's integration into the Internet and the system's maintainability, and reduces response time and network overhead.

This section details how to specify, develop, and deploy APIs on mobile devices. In order to perform a uniform deployment of APIs, a process needs to be defined specifying a set of steps, guidelines, and standards to be followed to design, document, build, and deploy those APIs. In addition, in order not to place an added burden on the developer, that process must be based on existing standards and activities already used for other environments.

In the following subsections, we shall first detail the process of generating and deploying APIs on mobile devices. Second, we shall briefly explain the standard (OpenAPI Specification) on which we base the definition and modeling of the APIs. Third, we explain the scaffolding of the API designed for mobile devices (based on the Android operating system). It is an extension of an existing source code generator. And fourth, we explain how the microservices deployed are invoked.

3.1 | Deployment process

In order to perform a uniform deployment of APIs on mobile devices, developers need to be able to rely on a well-defined process. The different steps proposed are shown in Figure 2. As can be seen, these are the activities that are usually carried out in designing and implementing APIs for cloud environments.

1. **API definition.** First, the API to be deployed must be designed. In this case, this has to be carried out using the standard OpenAPI Specification (OAS) which is widely used in the industry for the design and implementation of APIs that are to be deployed in cloud environments.

   The definition follows the same notation and logic as if it corresponded to deployment on a cloud environment (i.e., the specification defines the different microservices that will be available for consumption).
2. **Generation of source code.** In this step, the skeleton of the API is generated. Currently, there are different tools allowing the scaffolding of OAS-defined APIs for a client–server architecture. Examples are OAS-Generator,[25] Guardrail,[26] and Swagger Codegen.[27] These tools do not support the generation of APIs for mist or mobile computing paradigms. Therefore, an extension of the OpenAPI Generator has been developed to create and deploy APIs on Android mobile devices.

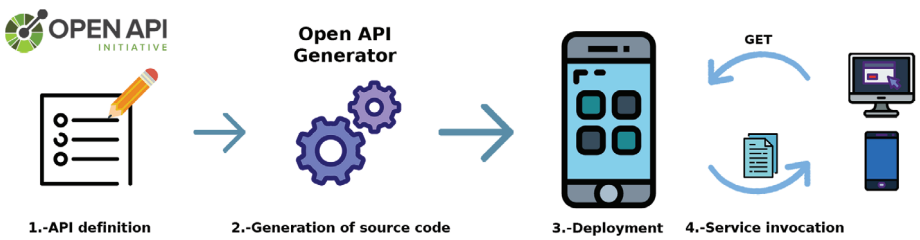


FIGURE 2 Deployment process to generate mobile APIs

3. **Deployment.** Before being able to deploy the API, the developer must finish the implementation by adding the business logic—that is, the behavior of the different endpoints defined (obtaining information from the sensors or the stored virtual profile)—and the configuration of the communications protocol. Then, the API can be deployed following the process defined by the manufacturer. In Section 4.1, the API of the scenario described above is developed in full, from its design to its full deployment, being fully operational for use.
4. **Service invocation.** Finally, with the API deployed, any device with the necessary permissions will be able to consume the defined microservices, even chaining calls to the microservices to obtain progressively more refined information from a user, an essential aspect in order to have IoT devices and information systems capable of reacting in demanding times to the user's needs. The endpoints provided can be consumed by other servers or devices, using a peer-to-peer schema. These invocations will be detailed in Section 3.4.

3.2 | The OpenAPI specification

In order to deploy an API, it first has to be defined and designed. As mentioned above, we propose the use of OpenAPI since it is a language widely used in the industry. This subsection will be concerned with the most important elements of the OAS to be considered for the definition and design of mobile device APIs. Some of them are shown in Figure 3(A).

Firstly, the **OpenAPI** element gives the version of the specification. Currently, the process and tools that are defined support version 3.0.0 or higher.

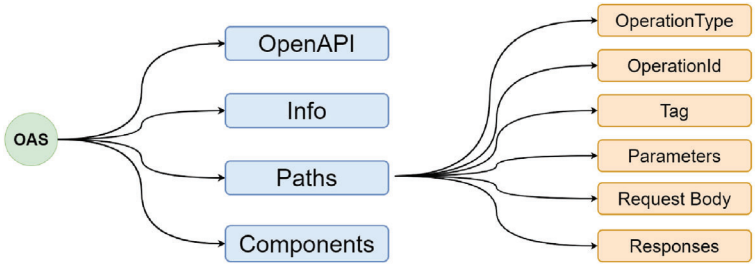
Secondly, **Info** contains data about the API such as name, description, contact, terms and services, etc.

Thirdly, **Paths** indicates the different endpoints or API operations, that is, the different Human Microservices that can be invoked. To define an endpoint, one has to indicate its name, its properties, etc. Figure 3(B) shows the most important properties of this element that are needed to generate mobile device APIs. *OperationType* indicates the type of the operation *(GET, POST, PUT, DELETE)*. Obviously, Human Microservices supports all of them, for example, information can be consumed using *GET* operations, but it can also be sent to the user using a *POST* operation. *OperationId* indicates the ID of the operation which will later be used to identify the source code generated. *Tag* is used to group the different endpoints depending on their goals, that is, tags will allow the operations to be organized into groups. *Parameters* defines the different parameters that the operation needs. It includes the name and type of each parameter. *RequestBody* contains the body of the operation if it is a *POST* or a *PUT*. Usually, the body is defined in the *Components* element in order to be reused by other microservices. *Responses* includes the different responses that can be returned as a result of the operation. As in *RequestBody*, it is possible to define responses in *Components* and reuse them.

And fourthly, in the **Components** element, different parts of the specification are declared as schemas, responses, etc. For example, the schemas allow input and output data types to be defined. These types may be primitives or arrays as well as objects. Once declared in this element of the OAS, they can be referenced in any other operation.

3.3 | Scaffolding of mobile device APIs

Once the API has been designed using OpenAPI, its skeleton has to be generated. As mentioned above, we extended OpenAPI Generator to support the generation of code for Android mobile devices. The process of generation and deployment is similar for all these devices, and to avoid being redundant, in this paper our focus will be on the whole process for mobile devices with Android OS.



(A) The principal elements of the OAS. (B) The principal properties of the OAS's Path element.

FIGURE 3 The OpenAPI specification's (OAS's) essential structure. (A) The principal elements of the OAS; (B) The principal properties of the OAS's Path element

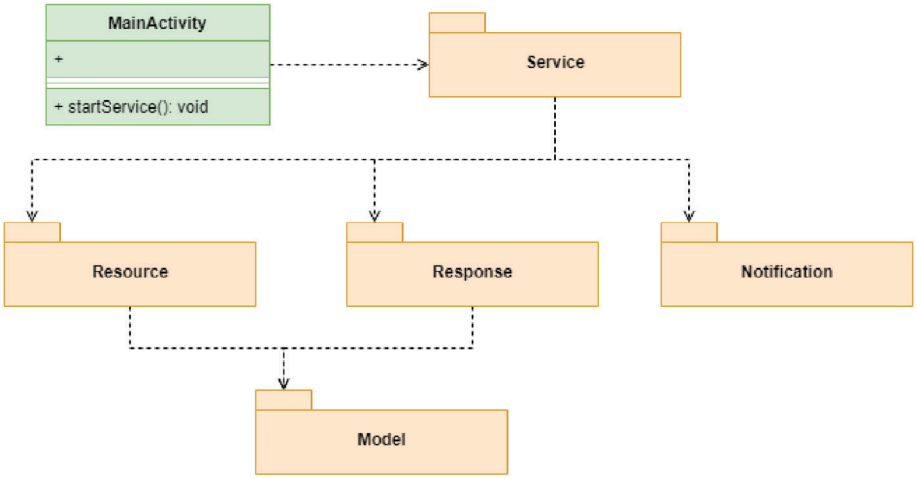


FIGURE 4 The Android API's main structure.

This tool is denoted APIGEND (API Generator for End Devices). APIGEND is a Web application developed with Spring* that uses Mustache† templates for the code generation. At present, we are hosting it on Heroku‡, and it is available to be used by any developer.

Currently, it allows the generation of APIs for Android devices with an Android API level 26 or higher. One of the main problems with the deployment of APIs on Android devices is their consumption. Normally, due to security restrictions of the operating systems or to the mobility of these devices (which can lead to intermittent Internet connection and changes in the IP address), the consumption of the deployed services can not be based on synchronous communication. Therefore, an asynchronous communication schema has also been supported in the extension developed which, in particular, includes different communication protocols such as Firebase Cloud Messaging (FCM)[28] and MQTT.[29] In Section 3.3, we shall explain the extension of the generator that was developed. One advantage of this framework is that we have added all the libraries required for deploying an API and to support different communication schemes for automatic addition and inclusion in the skeleton generated.

The objective of this extension is to generate a native API—in order to increase efficiency and not expend the resources of computing-limited devices—that supports the provision of Human Microservices. To this end, the API generated also follows the structure and good practices defined by the manufacturer, that is, Android§.

Figure 4 shows the packages diagram of the API generated by APIGEND in which all the API's business logic is located. In the following paragraphs, we shall explain the contents of each package.

Model contains the Java classes that implement the schemes of the specified communication objects. Each class automatically contains serializers (to map the different parameters to the Responses), the Getter, Setter, etc., in order to assist developers with the implementation.

Resource contains the classes in which the endpoints or Human Microservices are defined. Their purpose is to contain the business logic of each endpoint. Each class contains the set of endpoints associated with a *Tag* in the form of Java methods. These methods are empty in the absence of any implementation because it is for the developers to complete the specific business logic of each endpoint.

Response contains classes with serializers for mapping requests. When a request is received, it is in JSON format, so that great effort is required on the part of the developer to manually extract each parameter and identify its type. These serializers map requests, transforming the received data into instances of the defined Java primitives, objects, arrays, etc., thus saving developers time.

Service contains the classes that implement the communication middleware selected (*MQTT* or *Firebase Cloud Messaging*). In the case of MQTT, it comprises several classes that are automatically generated with the configuration of this communication framework. The only one that the developer must modify is *MQTTConfiguration* in order to define the IP address and the port of the MQTT broker(s). The rest of the classes are responsible for managing the workflow associated with the invocations of the microservices—that is, when the API receives a message, through the Responses, it is mapped

*https://spring.io
†https://mustache.github.io
‡https://openapi-generator-spilab.herokuapp.com/
§https://developer.android.com/studio

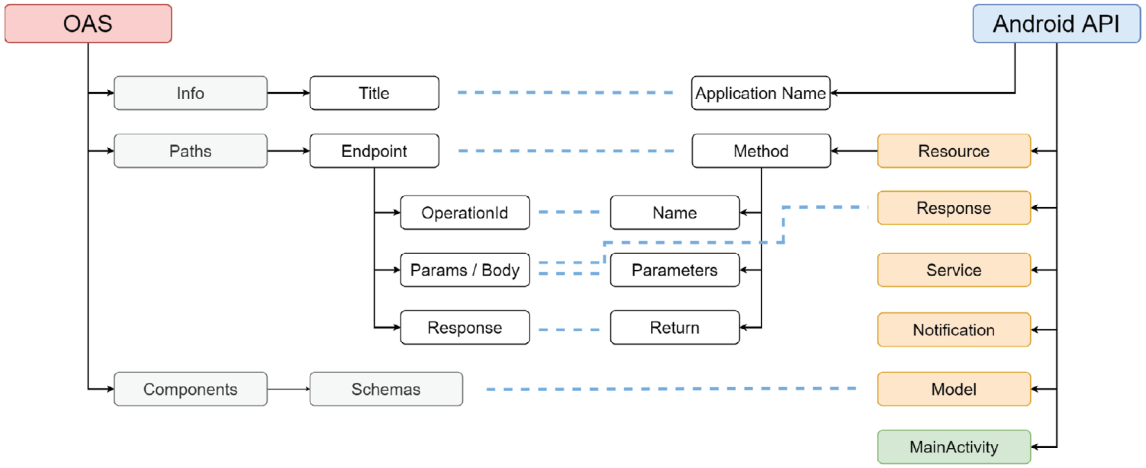


FIGURE 5 A synthesis of the conversions between OpenAPI Specification and the Android API's main structure

and derived to the correct Resource managing the microservice invoked. In the case of Firebase, only the *FirebaseService* is generated with which to manage this workflow. The application-specific configuration can be completed through the Android IDE[¶].

Notification contains classes for storing the requests received by the mobile device in a small database, so that they can be presented in a list. This package allows, firstly, developers to know that the request has been successfully received, and secondly, users to evaluate the request received in order to increase their control over the data consumed.

MainActivity starts the communication services and the different elements of the main view to start the list in which the notifications are presented. It also has the corresponding methods defined so that the API can receive requests even if it is in the background on the device.

Finally, Figure 5 shows a diagram synthesizing the different conversions made between the OpenAPI specification and the Human Microservices. It details the different elements of the specification and the set of directories and classes generated for the API. The blue dashed lines represent the relationships connecting them.

The name of the API is determined by the title of the specification. In the specification, each Tag groups a set of endpoints that are similar because they reuse parameters, object schemas, etc., and/or because of their purpose. Therefore, the classes of the Resource package are generated from the Tags defined. Each class takes the name and contains the endpoints associated with the corresponding *Tag* in the form of Java methods. These methods have the name of the OperationId and the parameters of the specification transformed to Java data types. In addition, the object that the microservice has to return as answer according to the design is also specified. This allows for better organization of the code. For example, a tag groups together several human microservices in which the device's database has to be accessed to retrieve personal information, health information, etc. For the developers, implementation will be easier and optimal since, in this case, they can reuse the same connection and part of the code to access and retrieve the information from the database.

The same is the case with the classes of the Response package. They are also generated and grouped depending on the tag with which they are associated. Within each class, all the parameters defined in the endpoints annotated by the same tag are grouped together and correctly transformed to Java data types. This also allows developers to reuse code since, on many occasions, the endpoints that are grouped under the same tag use the same parameters or objects for their invocation. For example, if one wants to obtain different information about an individual's location (i.e., current location, past location, etc.) through Human Microservices, the developer will most likely need, among other data, some unique identifier as a parameter to be able to provide that information.

The classes of the Model package are generated from the object schemes detailed in the Components section, taking the name of the scheme as a name. Each class contains the attributes that have been specified in the schemes, but transformed to the Java data types. These models will be used by the different endpoints as parameters for the invocation or as a response.

[¶]https://firebase.google.com/docs/cloud-messaging/android/client

3.4 | Consuming Human Microservices

This subsection explains how to invoke the microservices deployed on mobile devices. As noted above, the source code generated also allows two asynchronous communication protocols to be used, Firebase cloud messaging (FCM) and MQTT, in order to overcome some of the restrictions of mobile operating systems and devices. However, their use differs somewhat. In the following, we shall explain how to invoke the microservices in each of the two cases.

3.4.1 | Firebase cloud messaging

FCM can send messages through the Firebase Admin SDK or through HTTP or XMPP protocols. These messages can be received through push notifications to iOS, Android, or Web (JavaScript) devices. This system allows one to send a message directly to a single device or a set of them through "*topics*." FCM has the property of storing messages in a queue for a time until the device has Internet connection.

The communication flow is as follows. By default, when the API is generated it is automatically configured to be subscribed to a main topic, by which it will receive the consumers' requests. Once the microservice has been processed, the response can be sent using FCM (by default) or other communication protocols. FCM provides a unique *"id"* called *token* for each user, so that the response is directly sent to the consumer. Consumers can also use the tokens to send the invocations, but that would require them to know the provider ID. Therefore, by default, the Human Microservices request is made through the main topic. For instance, the request can be sent using FCM's API[#]. Nevertheless, in order to invoke a specific endpoint, a structure has been defined for invoking a microservice:

- **Body**: The content of the request is in JSON format. The different parameters that must be specified are:
  - **To**: This parameter corresponds to the name of the topic or the ID of the device to send the request to as if it were the *"IP Address of the API"*.
  - **Data**: This parameter contains all the information regarding the request, resource, microservice to be executed, parameters, or body of the operation. We have defined these parameters in order for them to be processed by the framework presented.
    * **Resource**: Name of the *Tag* associated with the endpoint to be invoked. This parameter is used to discern between the other resources/tags that group the endpoints in order to optimize the response time.
    * **Method**: OperationId of the endpoint to be invoked.
    * **Sender**: This parameter is automatically included in the Response of the microservice. It is used to indicate the consumer's ID in order to send the reply.
    * **Id Request**: This is automatically included to identify each request.
    * **Params**: The request body or parameters required for the endpoint to be invoked.

Listing 1 shows an example of an API request. It is based on the case study of the emergency alert system explained in Section 2. This microservice can obtain the location of all the users within an area (latitude, longitude, and radius) around where an emergency has occurred. The users within the delimited area will return their current location in order for an efficient evacuation plan to be developed.

```
{
    "to": "/topics/CitizensE",
    "data":{
        "resource": "User",
        "method": "getLocation",
        "sender": "employee3748",
        "params": {
            "latitude": 38.514233
            "longitude": -6.847281
            "radius": 2000
        }
    }
}
```

Listing 1: Content of an Emergency Alerts API request using FCM

[#]https://firebase.google.com/docs/cloud-messaging/http-server-ref

3.4.2 | MQTT

MQTT is also an asynchronous communication method implementing a publish/subscribe pattern. A customer can publish a message on a topic. Other clients can be subscribed to this topic, and the broker will send them the messages published.

The process is practically the same as with FCM. The provider is subscribed to the main topic, by which it will receive the requests from the consumers. Then the reply is sent directly to the consumer using their ID or a personal topic. The content of the request is in JSON, and its structure is similar to that defined for FCM. Since the *"data"* and the *"to"* fields in the FCM request are specific to the FCM API, MQTT only uses the structure that is defined within the former of those fields. The **MQTT topic** for sending requests to the API by default is the *title* of the specification without spaces.

4 | VALIDATION

This section evaluates the Human Microservices framework and tools. Section 4.1 explains the development and deployment of Human Microservices through a real case study used to monitor the activity of people during the COVID-19 pandemic. Then Section 4.2 evaluates the proposed framework through a survey of different developers who were trained in its use.

4.1 | Case study: Monitoring people's activity

This subsection explains the development and deployment of an API with Human Microservices on citizen's mobile devices. This case study focuses on the scenario presented in Section 2 in which an emergency alert system was proposed, using the information citizens provided to better control and monitor their health in emergency situations. In addition, this information could be used by other citizens to quickly help their neighbors.

This case study corresponds to a real environment in which the application was developed. The application, called Contigo[‖], was developed during the initial stage of the COVID-19 pandemic for the institution SEPAD[**](Extremadura Service for the Promotion of Autonomy and Attention to Dependence) to monitor elderly people during this pandemic. Contigo is especially aimed at elderly people who live alone, and therefore for whom it is found more difficult to know what their health status is.

The following sections detail the development and deployment of this case study using the tools that have been presented above.

4.1.1 | API definition

The first step in the process proposed is to design the API for deployment on mobile devices. The following paragraphs explain the different microservices that make up the API. These microservices will be defined using OAS so that their skeleton can subsequently be automatically generated by APIGEND.

- **Post alert:** This microservice can be used to inform about critical situations that may occur near the user's location. To do this, the mobile device can for instance display a message informing about the details of the alert.
- **Get user:** This allows the SEPAD (or the emergency control service) to know whether a user is within a critical area. If so, it provides the citizen's id so that this ID can in the future be used to get more personalized data.
- **Get user status:** This provides information about the health situation of a user that is gathered passively using their mobile device's built-in sensors. This service stores, for instance, information about the time that the screen is active, or whether the user is doing some specific activity through the use of the accelerometer sensor. The stored information is provided to external entities.

[‖]https://contigo.spilab.es/contigo/
[**]https://saludextremadura.ses.es/sepad/inicio

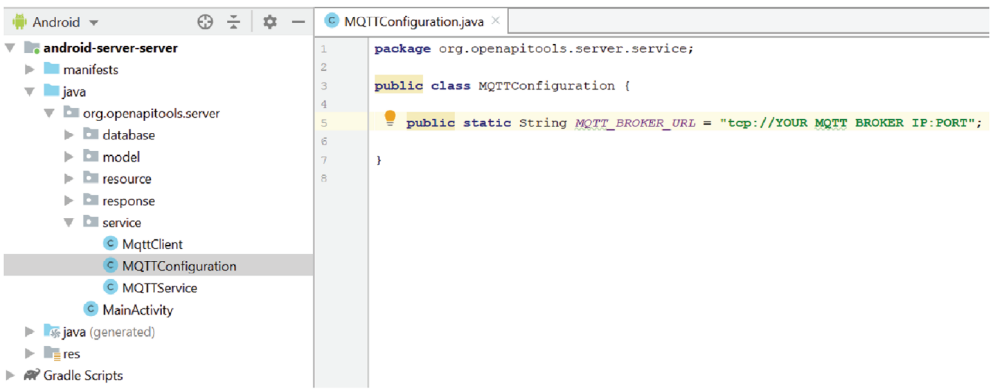


FIGURE 6 MQTT configuration file to connect to the broker

- **Post status:** This microservice can be used for an active check in which the user must respond. For example, if it has been detected that a user has low activity, they can be asked about their status, and they can answer. The answer is also stored in the virtual profile, so that other devices can get the user's status.
- **Get user location:** This microservice provides a citizen's current location. Latitude and longitude are obtained so that it is possible to identify where the citizen is for possible evacuation if low activity has also been detected.
- **Get user health:** This microservice provides a brief report of citizens' health problems. As with the previous microservice, this information is vital for emergency situations, for example, to prioritize the evacuation of some users or to know which specific medication or other treatments might be required.

These microservices can be used by both the SEPAD (nurses, specialists, etc.) and the citizens themselves (neighbors, family members, etc.). However, this could be a problem for the users' privacy. To resolve this problem, we are currently working on a framework to empower users to also manage the privacy of their data, indicating which, when, and how other entities can consume it.[24]

This API, with the endpoints described, was defined using OpenAPI. Its complete specification is available on GitHub[††].

4.1.2 | Source code generation and deployment

After defining the API, the second step is the generation of its skeleton by means of APIGEND. In the generation options, one has to indicate the operating system (in this case, *android-server*), the URL where the specification is stored, and the communication protocol that should be used. For this case study, we chose MQTT as the communication protocol in order to reduce the dependencies with external systems and to include security protocols for communication encryption. Appendix A describes the scaffolding process in detail.

In the third step, one has to complete the API. This begins with completing the configuration of the communication layer for which one only has to edit the MQTTConfiguration file inside the Service package, indicating the IP and port of the broker to which we have to connect to send and receive the messages. Figure 6 shows the MQTTConfiguration file. Then, the business logic of each endpoint must be developed. As mentioned above, these endpoints are defined in the Resource package. For example, the *Get User Location* microservice provides a citizen's current location. To finish this microservice, it is necessary to just include the source code that gets the current location of the mobile device directly through the GPS sensor.

Each API has different endpoints and behaviors, which means differences in their development. Also, each developer can make changes in the skeleton generated such as adding new classes, dependencies, etc. Nevertheless, the tool that has been developed allows developers to reuse existing standards, thus improving the software's quality. Also, the user's virtual profile can easily be provided to be consumed by third parties.

Finally, the deployment process is unique for each API, and depends on the process defined by the manufacturer.

[††] https://github.com/slasom/human-microservices

```
public void getLocation () throws MqttException, UnsupportedEncodingException{
        fusedLocationClient.getLastLocation().addOnSuccessListener((Activity) context, new
            OnSuccessListener<Location>() {
          @Override
          public void onSuccess(Location location) {
              if (location != null)
                  myLocation = new LatLng(location.getLatitude(),location.getLongitude());
          }
      });

    //Return Location on reply.
    mqtt.publishMessage(MQTTService.getClient(),toJson(myLocation),1,userResponse.getSender());
  }
```

Listing 2: Get User Location microservice code

4.1.3 | The Contigo mobile application

In this subsection, we intend to demonstrate the benefits of the system designed with Human Microservices. Contigo has been deployed with most of the microservices explained above. Two more of them are planned for future releases.

Contigo monitors whether a person used their mobile phone and which physical activities they performed during the last few hours. Figure 7 shows the mobile application's interface with which the user can observe the time of the set of activities detected during the last 24 h.

The health and activity information is stored only on the mobile device, and offered through Human Microservices. Currently, it can be consumed from a web application developed by SEPAD that aggregates and coordinates all the information stored in the devices.

This web application obtains the condition of the different users, aggregates that data, and displays the information on a map. In this way, health services can observe which users have been using mobile devices with a certain degree of normality and which have behaved in an unusual way. In addition, these microservices can also be consumed by authorized persons such as neighbors or relatives of an elderly person. Figure 8 shows a screenshot of the result obtained from a request to the devices that are within the area delimited on the map. Through the sidebar, one can adjust different parameters such as the radius of the area, the minimum activity, visualization settings, etc. The upper bar allows one to generate a document with the list of people whose activity has been below the minimum.

4.1.4 | Resource consumption analysis

One factor to take into account is the consumption of resources (battery, data traffic, etc.). Resource consumption is a fundamental pillar for the success of almost any application deployed on battery-powered devices,[6] as is the case of

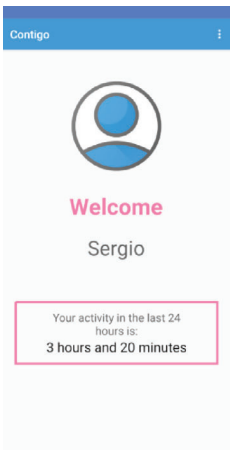


FIGURE 7 Contigo APP screenshot

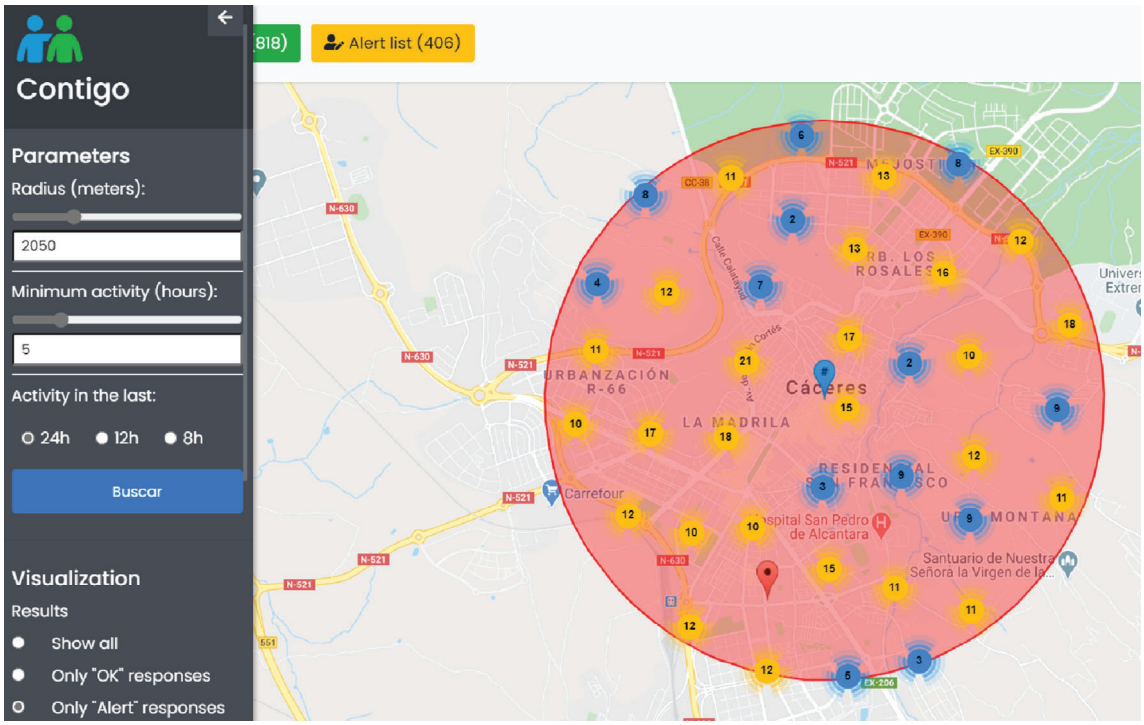


FIGURE 8 Contigo website

the devices proposed for the deployment of Human Microservices. Being deployed on a mobile device or on a cloud environment entails different consumption patterns. Obviously, in the former case, that device has to process and provide all the information. In the latter case nevertheless, the device has to constantly be sensing and sending all the information. In this section, we evaluate the resource consumption of the API designed, taking the Contigo app as referent. Specifically, we shall evaluate and compare the battery consumption and data traffic when the microservice *Get User Status* is invoked.

Each access to this microservice requires retrieving from the device's storage the user's activity during a time frame indicated as input parameter (e.g., activity time during the last 12 h), the user's basic data (name, address, etc.) and their current location. The average size of this data is 1 kB.

We have measured the execution of this microservice on seven Android devices. These devices are two Xiaomi Mi 9, two Xiaomi Mi 9T, one Huawei Mate 20, one OnePlus 6T, and one Xiaomi Mi Mix 2. In order to determine the consumption of resources, the battery and data traffic dimensions were measured. Battery consumption measurements were obtained with Batterystats[‡‡]which is an Android framework tool that collects device battery data in the background. To analyze the battery consumption provided by Batterystats, we used another Android tool called Battery Historian[§§]. In order to analyze data traffic, we used the tool Profiler[¶¶], which allows us to inspect network traffic in Android devices. This tool is included in the Android Studio IDE[##]. Figure 9 shows the results obtained from these measurements, including battery consumption, data traffic consumption, and operating costs generated, comparing the deployment of microservices using Human Microservices and a cloud environment.

The mean battery consumptions were 10.35 μAh for computing the request and 18.31 μAh for sending it. The data consumption for sending the information was 2.07 KB (since the communication protocol headers have to be included). Therefore, the average total consumption for each request was 28.66 μAh and 2.07 KB.

With Human Microservices, this access would occur only when authorized persons (nurses, family members, etc.) invoke the microservice. Let us assume that on average they are going to check the user's condition about four times a day (three invocations by family members and one by the nurse), which would amount to 124 invocations per month. This makes a total consumption per month of 3553.84 μAh of battery and 256.68 KB of data traffic.

[‡‡]https://developer.android.com/topic/performance/power/setup-battery-historian
[§§]https://developer.android.com/topic/performance/power/battery-historian
[¶¶]https://developer.android.com/studio/profile/network-profiler
[##]https://developer.android.com/studio

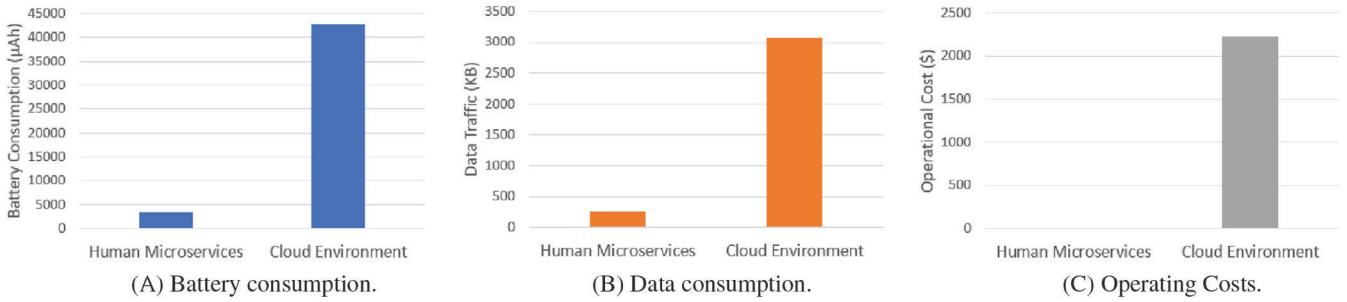


FIGURE 9 Human Microservices versus cloud environment resource consumption for the defined case study. (A) Battery consumption; (B) Data consumption; (C) Operating costs

In contrast, if Human Microservices were deployed in a cloud environment, the data has to be up-to-date to be useful. Therefore, mobile devices have to be constantly computing and sending the information to the cloud environment at a stable frequency. If one assumes that the data is updated every half an hour, there will be an average of 48 accesses per day and 1488 accesses per month. Given the consumption measurements, the total consumption would be 42,646.08 Ah of battery and 3080.16 KB of data traffic. As can be observed in Figure 9(A), (B), this means that to have up-to-date data with a cloud environment, a consumption of both battery and data traffic of 12 times more than with Human Microservices would be required.

Another type of resource consumption is that of resources in the cloud. Human Microservices are deployed on mobile devices and do not lead to any increase in operating costs. However, if they are deployed in a cloud environment, the data has to be stored. In accordance with the above measurements, the average transfer size is 1 KB (without headers) per device. If this system were hosted by a medium sized city with 300,000 inhabitants, it would mean a data volume of 446 MB for each invocation.

Let us suppose that we store the processed data in an Amazon S3[||||] database. This has a monthly cost of 0.023$/month per GB for the first layer. In addition, only 1 month of the location history will be saved. If storing the sensed data for all the users only once requires 446 MB, and in one month the information is obtained and stored 1488 times, this would be 663.6 GB for a cost of 15.26$/month. However, it is necessary to add the cost of writing to the database. The standard S3 database has a price of 0.005$/1000 requests. If in one month 1488 invocations are made for an average of 300,000 inhabitants, this results in a sum of 2232$/month whereas Human Microservices would not generate these operating costs since the microservices are deployed on the users' mobile devices, as can be seen in Figure 9(C).

Consuming Human Microservices that provide highly personalized and up-to-date information entails the consumption of many resources. As shown in this section, its deployment on the users' smartphones reduces resource consumption by up to twelve times and does not generate any operational costs compared to their deployment in a cloud environment.

4.2 | Training developers in Human Microservices

The Human Microservices framework has also been evaluated by developers. This validation consisted of a training course divided into three parts. The first part consisted of a survey inquiring into the problems of current architectural designs and the existing tools for deploying APIs on end devices. The second part focused on a training course explaining the concepts of the mobile computing architectures and the use of the Human Microservices framework. And in the third part we conducted another survey to inquire about the usefulness of the tools presented.

4.2.1 | Problems, hypotheses, and questions

Currently, most mobile applications use a client–server architecture in which smartphones consume information from cloud environments. Other architectural styles such as peer-to-peer or mobile computing have lower application rates. This may be due to a lack of training in these architectural styles or to the absence of any tools simplifying their

[||||] https://aws.amazon.com/s3/

application. Both situations lead to greater effort on the part of developers to apply them, and to greater costs for the software company.

These problems led us to the following hypotheses and questions on which we based the survey.

- **Hypothesis 1:** If more training in these architectural styles were offered, there would be an increase in the application rate. In order to check this hypothesis, three research questions (RQ1, RQ2, and RQ3) related to training in peer-to-peer and mobile computing architectural styles were posited:
  - **RQ1 - In general, the training received in the tools supporting the peer-to-peer style is …** (a. greater than that received for the Client–Server style, b. similar to that received for the Client–Server style, c. smaller than that received for the Client–Server style). This question allowed us to determine the degree of training received.
  - **RQ2 - Have you received any training for the development of mobile apps following a peer-to-peer architectural style?** This question was raised to determine the training received in the peer-to-peer architectural style.
  - **RQ3 - What do you think it would be necessary for applying this style more frequently?** This question allowed us to determine which factor is the most important one in order to apply a peer-to-peer style.

- **Hypothesis 2:** If there were a larger set of tools facilitating the development of these architectural styles, this would mean a decrease in effort and cost, thus increasing the application rate. To check this hypothesis, three research questions (RQ4, RQ5, and RQ6) related to development knowledge and tools facilitating the development process were posited:
  - **RQ4 - How many mobile apps have you developed following a peer-to-peer style?**This question attempted to corroborate the lack of applications applying the peer-to-peer style.
  - **RQ5 - How many tools do you know facilitating the application of the peer-to-peer style?** With this question we tried to check if there is a lack of tools supporting this architectural style.
  - **RQ6 - Is it essential for you to have tools that support the peer-to-peer style?**This question allowed us to determine the importance of the existence of tools for applying the peer-to-peer style.

This survey was completed by the developers at the beginning of the training course. Subsequently, we trained them in the Human Microservices framework. Finally, three additional research questions (RQ7, RQ8, and RQ9) were posited to check its usefulness:

- **RQ7 - Did you understand the concepts explained in the course?** This question allowed us to determine whether the concepts explained in the training were understood.
- **RQ8 - The framework presented for developing Human Microservices is …** (a. Very simple, b. Simple, c. Normal, d. Complicated, e. Very Complicated). This question was raised in order to determine the complexity of the presented framework.
- **RQ9 - Have you been able to deploy the case study with Human Microservices?** This question attempted to verify the difficulty of deploying Human Microservices.

For reasons of space and readability, all the questions, possible responses, and results have been assigned to Appendix B. The following is an analysis with the most important results.

4.2.2 | Surveys and results

The training course was given in three universities (University of Malaga, University of Granada, and University of Valladolid) and at the International Conference on Web Engineering to students of different master's degrees, developers, and researchers. In total, there were 79 participants. Figure 10 shows the percentages of the profiles of the participants in the survey.

The results for the questions related to Hypothesis 1 confirm it. The participants found it to be essential to receive training in an architectural style for its application, advantages, and disadvantages to be better taken into account in the

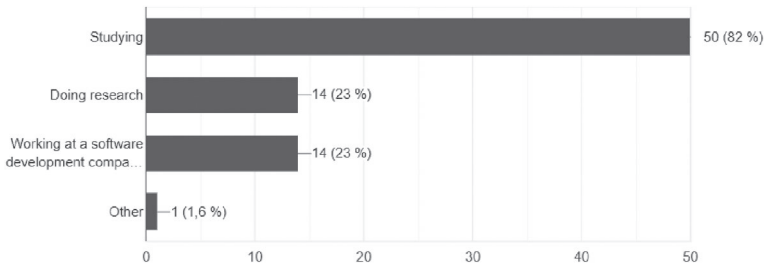
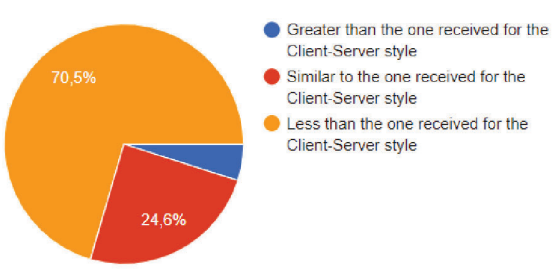


FIGURE 10 Profiles of training course participants

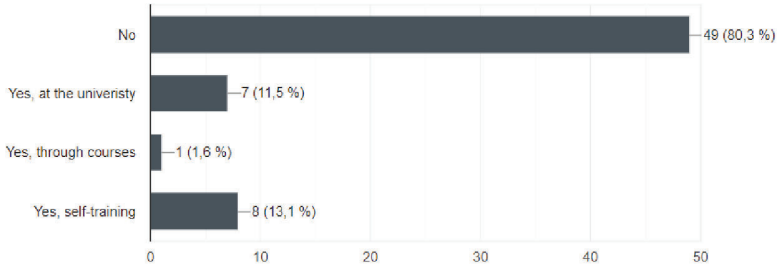
decision-making process. Figure 11(A),(B) shows the results of RQ1 and RQ2 where it can be observed that the majority of participants (around 70.5%) mainly received training in the client–server architectural style, since it is a style much more adopted in the industry. Nevertheless, in the responses to Question RQ3 (Figure 12(A)), 40.5% attributed this low application rate to the lack of both training and tools that facilitate development. This is related to what it is posited in Hypothesis 2.

The results for the questions related to Hypothesis 2 also confirm this hypothesis. For RQ4 (results shown in Figure 12(B)), almost 80% of the participants had never developed any peer-to-peer application. Moreover, as can be identified from the results of RQ5 (Figure 13(A)), almost 70% do not know any tool that facilitates the development of this architectural style. The results of both questions (RQ4 and RQ5) can be related. The lack of tools to facilitate the development of peer-to-peer applications is one of the reasons why the majority of participants have never developed an application following this architectural style. This is also validated in the responses obtained from RQ6 (shown in Figure 13(B)). Almost 80% find it essential to have tools that facilitate the development using this this architectural style. Therefore, this could be one of the determining factors for its application. While this paper proposed the concept of Human Microservices as microservices focused on providing information about people from their own devices, tools and processes are also required, as is reflected in the confirmation of Hypothesis 2. For this reason, we also proposed a suitable framework as the first step for its application.

Finally, with respect to the inquire into the usefulness of the tools presented for Human Microservices, in general the results were positive. Figure 14(A) shows the results of RQ7, detailing that the majority of participants (almost 93%) clearly

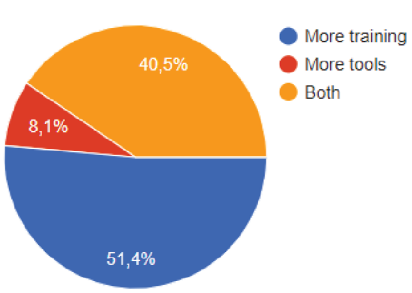


(A) Results of RQ1.

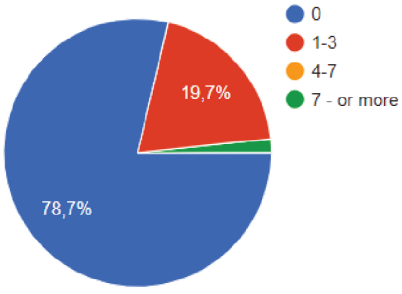


(B) Results of RQ2.

FIGURE 11 Results obtained for RQ1 and RQ2 belonging to Hypothesis 1. (A) Results of RQ1; (B) Results of RQ2

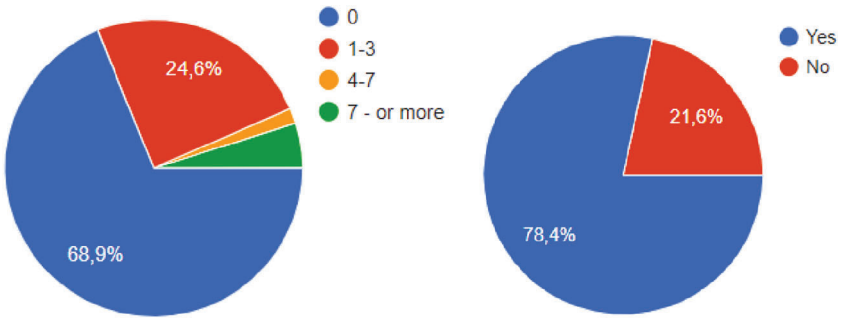


(A) Results of RQ3.



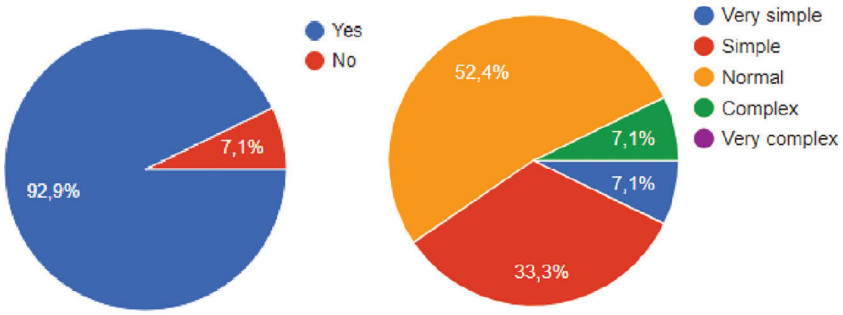
(B) Results of RQ4.

FIGURE 12 Results obtained for RQ3 and RQ4 belonging to Hypotheses 1 and 2, respectively. (A) Results of RQ3; (B) Results of RQ4



(A) Results of RQ5. (B) Results of RQ6.

FIGURE 13 Results obtained for RQ5 and RQ6 about the tools supporting the mobile computing paradigm. . (A) Results of RQ5; (B) Results of RQ6



(A) Results of RQ7. (B) Results of RQ8.

FIGURE 14 Results of RQ7 and RQ8 focused on the training course and the ease of use of the framework presented. (A) Results of RQ7; (B) Results of RQ8

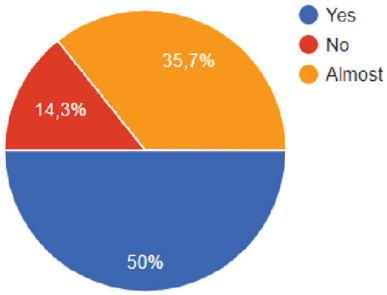


FIGURE 15 Result for RQ9 aimed at checking the applicability of Human Microservices

understood the concepts presented on peer-to-peer and mobile computing architectures, and Human Microservices. This is also reflected in the results of RQ8 (shown in Figure 14(B)), where more than 50% did not find the tool difficult to use, and indeed 33% found it simple. A very positive result, since the participants of the survey had never used the tool before and they managed to use it in only 1 h. Finally, similar results were obtained with the deployment of the case study developed with Human Microservices (RQ9, shown in Figure 15), 50% were able to deploy it without any kind of problem, and almost 35% achieved this goal—a fairly positive result given the short time in which the case study was developed and deployed. This indicates that the deployment of Human Microservices is simple.

5 | DISCUSSION

This section discusses the use of Human Microservices as a framework for providing personal and contextual information about people through APIs deployed on mobile devices.

Firstly, Human Microservices details a process guiding the development and deployment of APIs on mobile devices. This process comprises mechanisms based on existing technologies already used by developers, which shortens the learning curve. In contrast, currently, if developers want to provide information directly from mobile devices, they have to create ad hoc systems without following or applying any standard. Although similar functionalities could be implemented, the software would be harder to maintain and to integrate with other systems. In addition, the effort, and therefore the cost, of implementation would be greater.

Secondly, Human Microservices enables the use of the SOC paradigm and its technologies to implement and provide information from mobile devices. In addition to the benefits that it provides for the system's maintainability and modularity, it also allows the role of mobile devices to be changed, breaking the server dependence of the vast majority of mobile applications which are nothing more than clients. When the server is migrated to a newer version to add new functionalities, the clients also have to be upgraded or they may no longer work. Human Microservices changes the role of mobile devices, and makes them independent of the server. In addition, they can also be versioned (like any other API for server environments). Therefore, even if a new version cannot be deployed on a mobile device because of the computing capabilities required, an older version can be deployed in order to provide the basic services. Thus, different information systems consuming the older services can still work. In this way, the API and the mobile devices are completely detached from the server. For example, the API presented in this paper was focused on an emergency system, but it can also be used by other health services or even by smartcities to evaluate the user's location.

Thirdly, Human Microservices allows a developer to provide personal, very specific, and up-to-date contextual information. Some contextual information can be obtained at run-time from the different sensors and then provided to the consumers. For example, if one wants to get the location of a citizen, the microservice just accesses the GPS sensor and gets that information. Of course, some of the high level of availability that cloud environments provide may be lost. These devices may have intermittent connection problems, so that a mixed system could be designed in which some services can also be provided from edge or fog layers to provide the information when the device is unavailable.

Fourthly, Human Microservices allows developers to provide services of low granularity. Providing such microservices from a cloud environment would require implementing complex filtering techniques. For example, if one wants to provide the location of a citizen, search techniques would have to be implemented to identify the citizen as user, access their location history and retrieve the last one. Nonetheless, cloud environments also have several advantages when information from different sources has to be aggregated. For example, if one wants to know the number of citizens in a neighborhood, one would access the database, locate the users, and do a straightforward count. Instead, with Human Microservices, the location on each device would have to be checked. Each design has its advantages depending on the particular case and the type of information (historical and aggregate, or personal and recent) that needs to be consumed.

And fifthly, with regard to resource consumption, in Section 4.1.4 it was seen that the use of Human Microservices deployed on mobile devices had lower resource consumption (battery and data traffic) than deployment of these microservices on a cloud. In addition, the deployment of Human Microservices on mobile devices can lead to reduced network use since, as we have shown, considerably less data has to be sent. In an IoT environment, in which users are surrounded by Internet-connected devices, these devices can directly access the user's personal information. In addition to reducing network saturation, this improves response times, with faster adaptation of the devices and therefore a better user experience.

Edge and fog architectures and similar are being enhanced for the deployment of microservices close to the user so as to improve the quality of service (such as response time and latency). However, the personal information continues to be relegated to external nodes. The user therefore loses control over that data. With the present proposal, personal information is computed and provided from where it is generated. Human Microservices and the tools presented allow developers to change the role of companion and personal devices in order for them to store, compute, and provide information which is personal, thus improving the integration of users into the Internet, as demonstrated by the Contigo application.

6 | RELATED WORK

There exist few commercial tools that support other architectural styles, such as mobile-based styles or edge-based styles. AWS Greengrass[30] and Azure IoT Edge[31] allow developers to deploy an application in an edge environment. These have meant a major increase in quality in this area, providing processes and mechanisms that facilitate the development of this type of architecture for developers. However, they are always subjugate to the cloud environment, with a client–server schema which generates extra latency with the cloud. While their focus is on deploying APIs at the edge of the network, not on end devices, their proximity with the latter means that they could be complemented to provide some routing mechanisms that would give a higher quality of service. At the research level however, there is growing interest in proposals that encourage the use of these architectural styles to facilitate the development of novel applications.

In this sense, in Reference 32, the authors provide a systematic literature review on the game theory computation offloading approaches for the mobile edge computing environments in the form of a classical taxonomy to recognize the state-of-the-art mechanisms on this topic. In Reference 33, the authors propose an extension of the Business Process

Model Notation for modeling IoT-aware applications and systems, and an architecture for an IoT-aware business process deployment into hybrid fog/cloud. In Reference 34, the authors utilized learning automata as a decision-maker to offload the incoming dynamic workloads into the edge. Also, they propose an edge server provisioning approach using a long short-term memory model to estimate the future workload and reinforcement learning technique to make an appropriate scaling decision. Similarly, in Reference 35, the authors introduce a framework for the deployment of applications across multiple domains of cloud-fog providers while guaranteeing resources locality constraints. These works focused on defining different techniques for deploying/offloading services in computing devices closer to users (fog and edge environments). The presented proposal further delves in the deployment of services closer to the user by defining techniques and tools that help developers to define and deploy them on the users' personal devices.

As we have shown, the OpenAPI specification is very useful for defining applications, including the generation of new styles by creating new or extending their existing code generators. In Reference 36, the authors propose WoTDL2API as a tool to generate a RESTful API with a Web interface from an OAS specification. The API generated aims to generate and deploy a web-based RESTful interface obtained from the WoTDL descriptions to provide interoperability between different IoT devices that interact with different communication protocols (Zigbee, Bluetooth, RFID, …). The source code generated can be deployed in peripheral devices such as gateways, routers, etc. It cannot, however, be deployed on end devices such as smartphones, which does not allow it to use their capabilities to store and provide personal and contextual information.

In Reference 19, the authors presented a modular and scalable architecture based on lightweight virtualization techniques. Modularity, combined with the orchestration capabilities of Docker, simplifies administration and allows for distributed deployments, creating a highly dynamic system. Moreover, features such as fault tolerance and system availability are achieved by distributing the application logic across different layers. In (34), virtualization of IoT devices is proposed for improving the response time of smartcity applications. The proposal applies a shared use of microservices and a dynamic scaling of resources to improve the use of computer resources and the quality of the application. While these two proposals distribute the computing load between different layers, they store the data in a cloud environment, so they do not allow the provision of data from the outer layers. Furthermore, they do not provide tools to facilitate developers the application of the proposed frameworks, which has a major impact on their application as the validation section has shown. In contrast, Human Microservices allow information to be exposed directy from end devices, such as mobile devices, using conventional tools in order to shorten the learning curve and ease its application.

In Reference 18, the authors present the pillars on which to build fog health care applications, together with the evaluation of a working prototype called Spark IoT Platform that collects ECGs from an end medical device, and uses the patient's smartphone as a fog gateway for securely sharing those ECGs with other authorized entities. The information stored on mobile devices can be consumed through a health platform deployed in a cloud environment. This prototype allows patients to share information out to their physicians, monitor their health status independently, and notify the authorities in real-time in emergency situations. The smartphone communicates with medical devices through an API defined via secure Bluetooth wireless connection. The mobile application stores all the data, and is also capable of analyzing it using various algorithms available for the Android operating system. The cloud platform is responsible for providing access to authorized users to monitor patient data. However, it does not detail how medical information is shared from the smartphone to external users—a key process that we have covered in this paper with Human Microservices.

In Reference 37, the authors present PACO (Programming Abstraction for Contextual On-loading), an architecture enabling the storage of spatial-temporal data of the users in their mobile phones. Specially, they focus on location and time, although they provide flexibility to store additional contextual data. The user retains individual, direct, and physical control of the stored spatial-temporal information; no third party service "owns" or "controls" this highly personal information. To access the stored data, PACO provides an API that allows user applications on the device to freely consult the stored spatiotemporal data, retrieving any view of the raw data that is desired. In order to get access to the stored data, developers have to implement spatial-temporal queries that are provided to the PACO's API, which hinders its reusability and the use of a standard language to consume the exposed information. Human Microservices is based on OpenAPI specifications to clearly expose the services provided and how to consume them. Nevertheless, both approaches could be perfectly be integrated to, first, improve how personal data is stored and, second, how it is exposed to and consumed by external entities.

Tim Berners-Lee, inventor of the World Wide Web, has developed a project, called Social Linked Data,[38] in which he proposes conventions and tools to build decentralized social applications where users can store their personal data and control the access to them. This proposal defines how information can be stored and computed in mobile or web applications but does not specify how this information can be provided and consumed by external entities. It is the

decision and work of developers to carry out this task. In contrast, Human Microservices defines a process for designing and implementing the exposed information and, secondly this process is based on standards. Again, both proposals could be integrated in order to better address users' privacy issues, and to improve how the exposed information can be consumed.

In[39] the authors propose a novel service model called nanoEdge, in which end nodes with adequate capabilities (e.g. smart TVs, cars, smartphones, etc.) collaborate to provide the services needed for data and network management, processing, storage, and security functions, without heavily relying on centralised servers. The proposal also presents a proof-of-concept (PoC) implementation to demonstrate that the model is real-world achievable, and that performance and resource efficiency are feasible. Their vision is to make it part of IoT scenarios in which users interact with local and global digital services mainly through intelligent environments. However, the proof of concept shown is an example developed specifically for the use case presented in the paper, and the different technologies used are explained, but no tool is provided to facilitate the application of the proposed model to other scopes. In our proposal, the detailed framework is specially oriented to allow users to expose the information gathered by their mobile phones in order to be easily consumed by IoT devices. The APIs exposed with Human Microservices can be integrated in order to improve the behavior or IoT devices depending on the users' context.

Finally, in Reference 40, the authors propose a scalable IoT architecture that exploits transparent computing. The aim is to provide scalable services on lightweight IoT devices to build scalable IoT platforms by distributing computing and storage among different layers according to demand. This provides improved response times, and support for context-aware services, inter alia. The authors present a case study in which TCwatch wearables are end devices, a smartphone acts as the edge server, and a high performance PC is the cloud server. TCwatch devices can select an application from the Applist on demand, and send the sensed data (and the application data) to the smartphone for edge storage and processing. Nonetheless, this case study is very specific for the devices that are used. Neither are details provided on what kind of application can be deployed, nor are any tools provided with which to easily apply this approach to other use cases.

7 | CONCLUSIONS

During the last few years, the deployment of mobile devices and IoT has grown considerably, generating a large market of mobile and IoT applications which require information which is more specific and personalized in order to automatically change their behavior depending on the context and the requirements of each user. These application are usually based on a client–server architecture in which all the information gathered is sent, stored, and computed in cloud environments. Currently, different architectures are being proposed to bring these storage and processing tasks closer to the users in order to improve the quality of service and to exploit the location-awareness and the locality of the data so as to also integrate the human into the IoT loop.

Currently, this integration requires developers to use ad hoc techniques, and firms to invest a vast amount of resources, in order to implement these techniques. Developers and firms need known paradigms, tools, and standards to be adapted in order to facilitate the distribution of computation among layers closer to the end-users. In addition, these techniques should empower users to manage their own data and its privacy.

In this paper, we have proposed Human Microservices as a framework to turn humans into data providers, offering personal and contextual information from their personal devices. This framework is based on the SOC paradigm and on existing tools and standards for the development of APIs for cloud environments, thus shortening the learning curve and reducing development costs.

As future work, we are working on further improving the quality of service, and the fault tolerance for system availability. Currently, end devices present and consume information through a single communication node. If, however, that node is located far from the point where the data is presented or consumed, the quality of service may be affected. Therefore, we are working on a distribution of communication nodes in different layers (edge, mist, etc.) where the devices can be connected to several nodes and some routing mechanisms can be implemented to achieve a higher quality of service.

ACKNOWLEDGEMENTS

This work was supported by the projects RTI2018-094591-B-I00, PGC2018-094905-B-I00 (MCI/AEI/FEDER,UE), the RCIS research network (RED2018-102654-T), the 4IE+ Project (0499-4IE-PLUS-4-E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the project UMA18-FEDERJA-180 (FEDER/Junta de Andalucia), by

the Department of Economy, Science and Digital Agenda of the Government of Extremadura (GR18112, IB18030), and by the European Regional Development Fund.

ORCID

Sergio Laso https://orcid.org/0000-0001-8911-9371
Javier Berrocal https://orcid.org/0000-0002-1007-2134

REFERENCES

1. Statista Mobile internet usage worldwide; 2019. https://www.statista.com/study/21391/mobile-internet-usage-statista-dos%sier/. Accessed February 26, 2020.
2. Miorandi D, Sicari S, De Pellegrini F, Chlamtac I. Internet of things: vision, applications and research challenges. *Ad Hoc Netw*. 2012;10(7):1497-1516.
3. Dragoni N, Giallorenzo S, Lafuente AL, et al. Microservices: yesterday, today, and tomorrow. *Present and Ulterior Software Engineering*. New York, NY: Springer; 2017:195-216.
4. Bass L, Clements P, Kazman R. *Software Architecture in Practice*. Boston, MA: Addison-Wesley Professional; 2003.
5. Barbera MV, Kosta S, Mei A, Perta VC, Stefa J. Mobile offloading in the wild: findings and lessons learned through a real-life experiment with a new cloud-aware system. Paper presented at: Proceedings of the IEEE INFOCOM 2014-IEEE Conference on Computer Communications, Toronto, Canada; 2014:2355-2363
6. Berrocal J, Garcia-Alonso J, Vicente-Chicote C, et al. Early analysis of resource consumption patterns in mobile applications. *Pervasive Mob Comput*. 2017;35:32-50.
7. Initiative O. The OpenAPI specification repository. contribute to OAI/OpenAPI-specification development by creating an account on GitHub; 2020 https://github.com/OAI/OpenAPI-Specification. Accessed March 3, 2020.
8. Petrov V, Mikhaylov K, Moltchanov D, et al. When IoT keeps people in the loop: a path towards a new global utility. *IEEE Commun Mag*. 2018;57(1):114-121.
9. Rosenberger P, Gerhard D. Context-awareness in industrial applications: definition, classification and use case. *Proc CIRP*. 2018;72:1172-1177.
10. Stojkoska BLR, Trivodaliev KV. A review of internet of things for smart home: challenges and solutions. *J Clean Prod*. 2017;140:1454-1464.
11. Yu R, Xue G, Zhang X. Application provisioning in fog computing-enabled Internet-of-Things: a network perspective. Paper presented at: Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, HI; 2018:783-791.
12. Satyanarayanan M. Mobile computing: the next decade. *ACM SIGMOBILE Mob Comput Commun Rev*. 2011;15(2):2-10.
13. Yousefpour A, Fung C, Nguyen T, et al. All one needs to know about fog computing and related edge computing paradigms: a complete survey. *J Syst Arch*. 2019;98:289-330.
14. Guillen J, Miranda J, Berrocal J, Garcia-Alonso J, Murillo JM, Canal C. People as a service: a mobile-centric model for providing collective sociological profiles. *IEEE Softw*. 2013;31(2):48-53.
15. De Moor K, Ketyko I, Joseph W, et al. Proposed framework for evaluating quality of experience in a mobile, testbed-oriented living lab setting. *Mob Netw Appl*. 2010;15(3):378-391.
16. Papazoglou MP, Georgakopoulos D. Service-oriented computing. *Commun ACM*. 2003;46(10):25-28.
17. Statista. Internet of Things (IoT) active device connections installed base worldwide from 2015 to 2025; 2020. https://www.statista.com/statistics/1101442/iot-number-of-connected-dev%ices-worldwide/. Accessed June 26, 2020.
18. Akrivopoulos O, Chatzigiannakis I, Tselios C, Antoniou A. On the deployment of healthcare applications over fog computing infrastructure. Paper presented at: Proceedings of the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), Turin, Italy; 2017;2:288-293; IEEE.
19. Alam M, Rufino J, Ferreira J, Ahmed SH, Shah N, Chen Y. Orchestration of microservices for iot using docker and edge computing. *IEEE Commun Mag*. 2018;56(9):118-123.
20. Linaje M, Berrocal J, Galan-Benitez A. Mist and edge storage: fair storage distribution in sensor networks. *IEEE Access*. 2019;7:123860-123876. https://doi.org/10.1109/ACCESS.2019.2938443.
21. Amazon Amazon web service lambda; 2020. https://aws.amazon.com/lambda/. Accessed March 3, 2020.
22. Amazon Amazon web service API gateway; 2020. https://aws.amazon.com/api-gateway/. Accessed March 3, 2020.
23. Miranda J, Mäkitalo N, Garcia-Alonso J, et al. From the internet of things to the internet of people. *IEEE Internet Comput*. 2015;19(2):40-47.
24. Herrera JL, Berrocal J, Murillo JM, Chen HY, Julien C. A privacy-aware architecture to ShareDevice-to-device contextual information Juan Luis Herrera. Paper presented at: Proceedings of the 2020 IEEE International Conference on Smart Computing (SMARTCOMP), Bologna, Italy: IEEE; 2020.
25. Group I OAS generator; 2020. https://github.com/isa-group/oas-generator. Accessed March 21, 2020.
26. Twilio Guardrail; 2020. https://github.com/twilio/guardrail. Accessed March 21, 2020.
27. Swagger. Swagger codegen; 2020. https://swagger.io/tools/swagger-codegen/. Accessed March 21, 2020.
28. Google Firebase cloud messaging; 2020. https://firebase.google.com/docs/cloud-messaging/. Accessed January 20, 2020.
29. MQTT. http://mqtt.org/. Accessed January 20, 2020.
30. Amazon AWS IoT greengrass; 2020. https://aws.amazon.com/greengrass/. Accessed February 28, 2020.

31. Microsoft Azure IoT edge. https://azure.microsoft.com/services/iot-edge/. Accessed February 28, 2020.
32. Shakarami A, Shahidinejad A, Ghobaei-Arani M. A review on the computation offloading approaches in mobile edge computing: a game-theoretic perspective. *Softw Pract Exper*. 2020;50(9):1719-1759.
33. Kallel A, Rekik M, Khemakhem M. IoT-fog-cloud based architecture for smart systems: prototypes of autism and COVID-19 monitoring systems. *Softw Pract Exper*. 2021;51(1):91-116.
34. Shahidinejad A, Ghobaei-Arani M. Joint computation offloading and resource provisioning for e dge-cloud computing environment: a machine learning-based approach. *Softw Pract Exper*. 2020;50(12):2212-2230.
35. Faticanti F, Savi M, De Pellegrini F, Kochovski P, Stankovski V, Siracusa D. Deployment of application microservices in multi-domain federated fog environments. Paper presented at: Proceedings of the 2020 International Conference on Omni-layer Intelligent Systems (COINS), Barcelona, Spain; 2020:1-6; 2020.
36. Noura M, Heil S, Gaedke M. Webifying Heterogenous internet of things devices. Paper presented at: Proceedings of the International Conference on Web Engineering; 2019:509-513; Springer, New York, NY.
37. Wendt N, Julien C. Paco: a system-level abstraction for on-loading contextual data to mobile devices. *IEEE Trans Mob Comput*. 2018;17(9):2127-2140.
38. Sambra AV, Mansour E, Hawke S, et al. Solid: a platform for decentralized social applications based on linked data. Technical report, MIT CSAIL & Qatar Computing Research Institute; 2016.
39. Harjula E, Karhula P, Islam J. Decentralized Iot Edge Nanoservice Architecture for Future Gadget-Free Computing. *IEEE Access*. 2019;7:119856–119872. http://dx.doi.org/10.1109/access.2019.2936714.
40. Ren J, Guo H, Xu C, Zhang Y. Serving at the edge: a scalable IoT architecture based on transparent computing. *IEEE Netw*. 2017;31(5):96-105.

How to cite this article: Laso S, Berrocal J, García-Alonso J, Canal C, Manuel Murillo J. Human microservices: A framework for turning humans into service providers. *Softw Pract Exper*. 2021;51:1910–1935. https://doi.org/10.1002/spe.2976

APPENDIX A. GENERATE API SOURCE CODE

APIGEND is an application that can be accessed using microservices or a Web platform[***]. In this Appendix, we shall explain the Web platform. When a developer accesses the Web interface, they will see an interface similar to the one shown in Figure A1. At the top, there are a series of suggestions for the correct generation of the source code. Just below, there are three sections (Gen-Api-Controller, Clients, and Servers).

For the generation of APIs or the server side of applications, developers have to access the section "Servers". In this section, the endpoint called *'Generates a server library'* (Figure A2) allows them to automatically generate the skeleton of an OAS-designed API.

To generate the API for Android devices, one must select the *Try it Out* option. This will allow the parameters to generate the API to be edited. Specifically, one has to select from the *framework* drop-down list the language or the operating system for which the API is to be generated. In this case, the **android-server** option has to be selected.

In the parameter section, one has to indicate the configuration of the API to be generated in a JSON format. This configuration contains different options such as the URL of the specification *(openAPIUrl)*, security parameters, other options such as the library used for communication, etc. To use MQTT, developers have to indicate the *library* with value **mqtt**, as shown in Figure A3. Otherwise, to use FCM, the developer just has to write **firebase**, as shown in Figure A4.

To generate the API, one just has to click on the *Execute* button. If there is no error in the specification, a JSON is obtained as a result. Inside the JSON, there is a link to download the application, as shown in Figure A5. By copying and pasting the link into the browser's navigation bar, a *.zip* file with the API generated is automatically downloaded.

This guide has shown how to generate the source code of an API for Android devices, although developers must finish the implementation. Nevertheless, with this tool, developers are saved time and effort as it is a simple process, instead of their having to implement an API from scratch.

[***] openapi-generator-spilab.herokuapp.com

FIGURE A1 APIGEND - Web interface

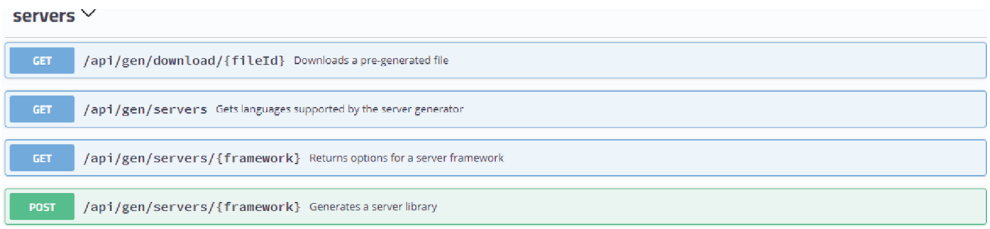


FIGURE A2 APIGEND - endpoint for generating APIs

FIGURE A3 Parameters to generate an application using MQTT

FIGURE A4 Parameters to generate a mobile application using Firebase Cloud Messaging middleware

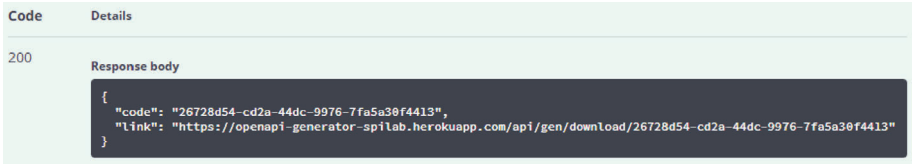


FIGURE A5 Result obtained from APIGEND generating the skeleton

APPENDIX B. SURVEYS

In this section, the survey and the results obtained with it are presented in detail to confirm the posited hypotheses and the usefulness of the framework that has been presented.

Questions

With regard to the first hypothesis *"If more training in these architectural styles were offered, there would be an increase in the application rate"*, we posed the following research questions:

- **RQ1 - In general, the training received in the tools supporting the peer-to-peer style is …** This question allowed us to determine the degree of training received concerning the typical client-server architectural style. As answers, three options were proposed.
  - greater than that received for the Client–Server style.
  - similar to that received for the Client–Server style.
  - less than that received for the Client–Server style.
- **RQ2 - Have you received any training for the development of mobile apps following a peer-to-peer architectural style?** This question allowed us to determine the previous training received in the peer-to-peer architectural style. As answers, four options were proposed.
  - No.
  - Yes, at university.
  - Yes, through courses.
  - Yes, self-training.
- **RQ3 - What do you think it would be necessary for applying this style more frequently?**This question allowed us to determine which factor is the most important in order to apply a peer-to-peer style. As answers, three options were proposed.
  - More training.
  - More tools.
  - Both.

Concerning the second hypothesis *"If there were a larger set of tools facilitating the development of these architectural styles, this would mean a decrease in effort and cost, thus increasing the application rate"*, we asked the following research questions:

- **RQ4 - How many mobile apps have you developed following a peer-to-peer style?**This question attempts to corroborate the lack of applications applying the peer-to-peer style. As answers, four options were proposed.
  - 0.
  - 1–3.
  - 4–7.
  - 7 or more.
- **RQ5 - How many tools do you know facilitating the application of the peer-to-peer style?**With this question we tried to check if there is a lack of tools supporting this architectural style. As answers, four options were proposed.
  - 0.
  - 1–3.
  - 4–7.
  - 7 or more.
- **RQ6 - Is it essential for you to have tools that support the peer-to-peer style?** This question allowed us to determine if the use of these tools is essential. As answers, two options were proposed.
  - Yes.
  - No.

During the second part of the training course, a case study with Human Microservices was developed and deployed with APIGEND in 1 h. With this question, we tried to corroborate the difficulty and usability of the tool. Finally, the questions asked to check the usefulness of the framework presented were the following:

- **RQ7 - Did you understand the concepts explained in the course?** This question allowed us to determine if the concepts explained about peer-to-peer, mobile computing architectures, and Human Microservices were understood. As answers, two options were proposed.
  - Yes.
  - No.
- **RQ8 - The framework presented for developing Human Microservices is …** As answers, five options were proposed.
  - Very simple.
  - Simple.
  - Normal.
  - Complicated.
  - Very complicated.
- **RQ9 - Have you been able to deploy the case study with Human Microservices?** This question attempts to check the deployment process of Human Microservices. As answers, three options were proposed.
  - Yes.
  - No.
  - Almost.

Results

In the following, the results of RQ1, RQ2, and RQ3, belonging to Hypothesis 1, are detailed and analyzed.

- **RQ1 - In general, the training received in the tools supporting the peer-to-peer style is …** Figure 11(A) shows the results obtained for RQ1. One can observe that a vast majority of participants (70.5%) have received less training in a peer-to-peer architectural style than in a client-server style.
- **RQ2 - Have you received any training in the development of mobile apps following the peer-to-peer style?** Figure 11(B) shows the results obtained for this question. In this question, one can detect the cause of the previous question's results. A great majority of participants (80.3%) have received less training in peer-to-peer architectures because they have simply have not been trained in them at all.
- **RQ3 - What do you think is required to apply this style more frequently?** Figure 12(A) shows the results obtained for RQ3. One can observe that 51.4% thought that receiving more training is very important to apply architectural styles such as peer-to-peer, although 40.5% thought that both training and tools are important to apply these styles.

The following are the results obtained for RQ4, RQ5, and RQ6, belonging to Hypothesis 2.

- **RQ4 - How many mobile apps have you developed following a peer-to-peer architectural style?** Figure 12(B) shows the results obtained. One can observe that 78.7% of the participants never developed a mobile application following this architectural style.
- **RQ5 - How many tools do you know facilitating the application of the peer-to-peer style?** Figure 13(A) shows the results obtained for this question. The vast majority of participants (68.9%) did not know any tool that facilitates the development or application of this style. These results could be related to those of RQ4 as most had never developed a peer-to-peer application, so one of the causes may be the lack of tools facilitating the development of such applications.
- **RQ6 - Is it essential for you to have tools that support the peer-to-peer style?** Following the previous answers, due to the lack of tools, the participants, by a large majority (78.4%), thought that it is essential to have tools facilitating the development of peer-to-peer or mobile computing applications (Figure 13(B)).

The following are the results for the questions focused on evaluating the usefulness of the Human Microservices framework.

- **RQ7 - Did you understand the concepts explained in the course?** Figure 14(A) shows the results obtained for this question. One can observe that almost all the participants (92.9%) understood the concepts presented on the peer-to-peer and mobile computing architectures and Human Microservices, which leads us to say that the processes and tools presented were easy for developers to understand.
- **RQ8 - The framework presented for developing Human Microservices is …** Figure 14(B) shows the results obtained. One can observe that 52.4% thought that the tool is normal in difficulty, while 33% considered it simpler. Their sum shows that the great majority of the participants were able to use the tool presented to deploy Human Microservices. This is a very positive result since they had never used it before, and they managed to learn it in only 1 h.
- **RQ9 - Were you able to deploy the case study with Human Microservices?** 50% of the participants were able to deploy the case study using the Human Microservices framework, while 35.7% almost did so (Figure 15). Therefore, as in the previous question, the results are very positive, and indicate that the process for deploying Human Microservices is very easy to apply.

According to the results obtained from the questions related to Hypotheses 1 and 2, we can state that it is of vital importance to provide both training and tools to increase and facilitate the application of mobile computing and peer-to-peer architectural styles. In this survey, we evaluated the usefulness of the Human Microservices framework, concluding that it is easy to apply and is accepted by developers.

3.4 Perses: A framework for the continuous evaluation of the QoS of distributed mobile applications

Authors: *Sergio Laso, Javier Berrocal, Pablo Fernández, Antonio Ruiz-Cortés and Juan M Murillo.*

Publication type: *Journal paper.*

Journal: *Pervasive and Mobile Computing.*

DOI: *10.1016/j.pmcj.2022.101627.*

Quality indicator: *JCR Q2.*

Pervasive and Mobile Computing 84 (2022) 101627

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Perses: A framework for the continuous evaluation of the QoS of distributed mobile applications

Sergio Laso [a,b,*], Javier Berrocal [a], Pablo Fernández [c], Antonio Ruiz-Cortés [c], Juan M. Murillo [a]

[a] Universidad de Extremadura, Cáceres, Spain
[b] Current Affiliation: Global Process and Product Improvement S.L., Cáceres, Spain
[c] I3US Institute, SCORE Lab. University of Sevilla, Sevilla, Spain

ARTICLE INFO

Article history:
Received 28 March 2022
Received in revised form 3 June 2022
Accepted 6 June 2022
Available online 18 June 2022

Keywords:
Distributed Computing
Mobile applications
Quality of Service
Evaluation
Virtual scenarios

ABSTRACT

The increasing capabilities of mobile devices have led to the emergence of new paradigms exploiting them. These paradigms foster the onload and distribution of functionalities on mobile devices, allowing the development of distributed mobile applications. This distribution reduces the latency and the data traffic overhead and improves privacy. As in any other mobile application, their success largely depends on the quality of service (QoS) they offer. Nevertheless, the evaluation of distributed mobile applications is particularly complex due to the number, heterogeneity, and interactions between the devices involved. Current techniques allow developers to assess the quality of a single device, but they are not designed for highly heterogeneous, distributed, and collaborative environments. This paper presents a framework called Perses, which allows the creation of virtual scenarios with multiple heterogeneous mobile devices to launch end-to-end tests to evaluate not only each device but also the interactions among them. The framework was evaluated against a real deployment, showing that the behavior and the quality attributes measured are similar to those of the real deployment, allowing developers to evaluate these applications before launching them. Finally, Perses was integrated into a DevOps methodology to automate its execution and further facilitate its adoption by software companies.

1. Introduction

We have witnessed a massive deployment of mobile applications since the arrival of the Apple App Store and Google Play [1]. This has generated a proliferation of companies dedicated to the development of mobile applications, leading to a great economic impact [2]. The success or failure of mobile applications largely depends on the quality of experience (QoE) [3] they offer, i.e., the satisfaction of a user with the provided functionality. The QoE is usually considered a broad term evaluating both the quality of service (QoS) [4] – quantitative measures of the performance of a service – and the user experience (UX) — efficiency of the interaction between the end-user and the service. Many of the companies behind mobile applications are startups so that the success of an application is an omen of the future of the company.

To ensure their success, mobile applications usually follow a pure client–server architecture. All computing demanding components (back end) are offloaded to cloud environments. Only the user interface and some basic components (front

* Corresponding author.
E-mail address: slasom@unex.es (S. Laso).

https://doi.org/10.1016/j.pmcj.2022.101627

end) are deployed on mobile devices. This architectural style allows developers to reduce the resource consumption on these devices; thus, allowing their deployment on almost any device with a minimum set of characteristics — limiting the problems caused by the great heterogeneity of devices in this market.

To evaluate the required quality, developers can assess each side (back end and front end) independently, but also end-to-end (E2E) testing are also performed. E2E tests assess the correct integration of the two sides and the correct functioning of the main functionalities, replicating the behavior of the users [5].

Nevertheless, in recent years, mobile devices have considerably increased their computing and storage capabilities [6], enabling the development of more computing-demanding applications in order to meet more stringent QoS requirements. New paradigms and architectural designs have emerged for developing them with distributed computing (in the following distributed mobile applications), such as Human Microservices [7], the Internet of People [8] or mist computing [9]. In addition, these paradigms allows one to address some challenges such as privacy awareness [10] and distribute the learning and intelligence among several nodes [11]. In these new paradigms, a functionality, or parts of it, may be distributed among different devices (mobile devices, IoT devices, fog or edge nodes, etc.). To consume this distributed functionality, usually some of the involved devices should interact to get all the required data and complete its workflow.

Therefore, these paradigms allow developers to relegate more functionality and on-load some computationally demanding components on mobile devices to further improve the QoS by reducing the latency caused by the network communication and data traffic overhead, increasing the control of the user over their data, and improving their privacy.

However, the QoS of distributed mobile applications is complex to evaluate. End-to-end evaluation in such architectures is not as trivial as in more traditional architectures. In a client–server architecture, it is only necessary to have the back end available and execute the end-to-end tests from a device. With these new paradigms, it is necessary to deploy a considerable set of heterogeneous devices close to real scenarios to properly evaluate the QoS — the latency and the execution time can vary greatly depending on the devices involved. This is especially relevant in today's ecosystem of mobile devices, due to the market fragmentation. This makes it even more necessary to know the expected quality in advance.

The application that does not meet the expected QoS is more likely to be rejected by users, with the financial and image impact that this may cause to a company. Currently, some commercial tools allow the evaluation of applications on different devices. AWS Device Farm [12] or Azure App Center Test [13] are platforms that provide a farm of real mobile devices. However, they do not allow the deployment of several mobile devices and the launch of E2E tests triggering different interactions among them, which is required to properly evaluate these distributed applications. Therefore, new techniques and tools are needed to evaluate the QoS of distributed mobile applications with E2E testing, enabling large-scale simultaneous evaluation of several highly interacting mobile devices and considering the heterogeneity of today's ecosystem.

In this paper, we present a framework called Perses. This framework allows the creation of virtual scenarios for the large-scale simultaneous deployment of virtualized mobile devices. Developers can simulate the deployment of distributed mobile applications in these heterogeneous scenarios. To evaluate the QoS correctly, the framework launches E2E tests that allow the whole system to be tested. In addition, this tool can be fully integrated into a software development process such as DevOps, automating the entire process of creating, deploying, and launching E2E tests, which reduces the effort invested by software companies to validate the QoS of their applications before deploying them into the production environment.

The main contributions of this works are as follows:

1. Perses allows to evaluate the QoS of distributed mobile applications through E2E tests in heterogeneous virtual scenarios with large-scale mobile devices.
2. Integration into common software development practices. Different methods have been developed to ease the integration of Perses into a DevOps methodology and in the Continuous Integration practices. This reduces the effort invested by software companies to validate the QoS of their applications before deploying them in the production environment.
3. The presented framework has been evaluated with a real application, comparing the real vs the virtual scenarios. In addition, different scalability tests have been performed, identifying that Perses is highly scalable and that the performance is maintained when the number of simulated devices increases.

The rest of the paper is structured as follows. Section 2 describes the motivations for this work. Section 3 explains the Perses framework. Section 4 validates the framework through a case study. Section 5 presents several related works. Finally, Section 6 details the conclusions.

2. Motivation

The growth of the mobile applications market has generated a multitude and variety of applications. Among the different companies that develop applications, there is a fierce fight for their applications to be the most downloaded and used to obtain greater benefits through different forms of revenue. There are different dimensions, such as advertising, originality, and virality for an application to be more or less successful. This success also lies in end-user satisfaction by offering a good QoE [14].

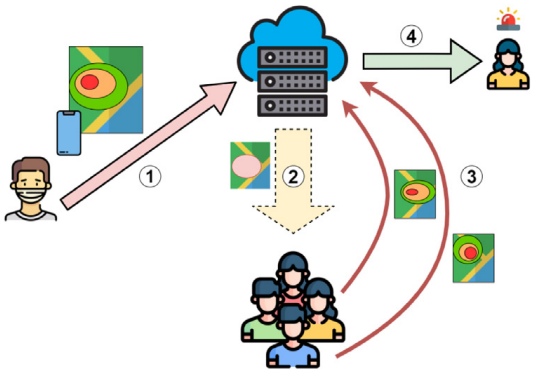


Fig. 1. COVID-Heatmaps app.

To achieve this QoE, it is necessary to apply software development methodologies and tools that allow developers to evaluate the system and to obtain rapid feedback. The QoE not only depends on the usability and accessibility that the interface consumes by end users but also depends on the QoS offered by the whole application and the infrastructure supporting it, especially in the case of distributed mobile applications in which computing is not primarily performed in a single location (cloud environment in traditional architectures) but is performed on a set of distributed devices.

During the last few years, we have witnessed applications (or specific releases) that either failed or had poor acceptance by users due to the QoS provided. For instance, Pokemon Go, a fairly successful mobile augmented reality game, may have failed in its early days. After an avalanche of downloads of the game in its presentation due to people's interest, the servers where the game's functionalities were hosted were saturated, causing the vast majority of users to be unable to play [15]. Another example is Auctionata — Online, which proposed live-streamed auctions of fine art and collectibles by broadcasting bids via mobile devices. The early attempts at broadcasting events failed to meet the expected QoS, limited by slow broadband speeds and delivery concerns [16]. In the latter case, the company was a start-up and had to close down.

These problems can be avoided by evaluating the QoS of the whole system before deployment. In client–server architectures, QoS is usually evaluated at the back-end, for example, by performing load tests simulating the connection of multiple clients with tools such as Apache JMeter[1] or Postman,[2] as this is where most of the computation and storage is located. Nevertheless, in distributed mobile applications, the computation and storage tasks are performed on different devices (smartphones, IoT devices, edge or cloud nodes, or any other device managing the system). Therefore, QoS evaluation must be performed jointly, involving all the different (or, at least, a good representation) devices that collaborate in the real deployment. This will allow developers to obtain results that can more accurately predict the expected behavior in a real production environment.

As a running example, let us present a distributed mobile application to detect close contacts of positive COVID-19 users, called COVID-Heatmaps. This application is composed, on the one hand, of a mobile application that stores the location of users on their devices and processes and generates heatmaps that are used to detect close contacts. On the other hand, a cloud node compares the heatmap generated by users with that of a COVID-19-positive user to detect whether they are close contacts. Fig. 1 shows an overview of how this application works. First (step 1), when a COVID-19-positive user is registered, the mobile application processes its heatmap with the stored traces and sends them to the cloud node. Second, the cloud node delimits the user's movement areas according to the heatmap and sends them to the other mobile devices (step 2). These devices check with their location history to determine whether they have been in the received areas. The devices that have been in these areas process and send their heatmap to the cloud node (step 3). The cloud node compares the heatmaps of the users with one of the COVID-19 positive, and the users who are considered close contacts are notified (step 4).

A potential solution for evaluating the QoS in these distributed applications is to use a real device farm that allows deploying the application on a large set of devices, and then, together with the cloud node, launch end-to-end tests, simulating the behavior of any user and the QoS obtained.

Several platforms offer physical mobile device farms. Among them are the AWS Device Farm and Azure App Center Test, which allow us to create highly customized test runs, indicating different types of devices, OS versions, etc. One of the disadvantages of these platforms is the number of devices that can be run simultaneously; they offer a very limited number and at a rather high cost. In addition, these platforms focus on launching UX tests, such as unit and adaptivity tests of graphical elements, navigation through different screens, compatibility with different OS versions, etc. (Appium,[3] Espresso,[4] etc.).

1 https://jmeter.apache.org.
2 https://www.postman.com.
3 http://appium.io.
4 https://developer.android.com/training/testing/espresso.

Furthermore, in distributed mobile applications, all the distributed components (mobile devices and cloud nodes in this case) must be fully available to execute E2E tests analyzing the QoS. For instance, to search for the close contacts of a COVID-19-positive user in our running example, the positive user identifies his or her movement areas in the previous few days to the cloud, the cloud sends these areas to the other users, these users validate whether they have been in these areas, and sends these evaluations to the cloud — which checks whether they are close contacts. This E2E test involves the evaluation of several components on different devices. Traditional testing platforms are not intended for testing scenarios with multiple interacting devices. They are mainly focused on running isolated tests on each device.

Therefore, new techniques are needed to evaluate the QoS of distributed mobile applications simulating a near-reality scenario. This will allow developers to launch E2E tests and measure the obtained QoS. These techniques will also be able to be integrated into a software development process, such as DevOps, so that they can be widely adopted by enterprises, saving the effort and cost required to perform these tests.

3. Perses: QoS evaluation in distributed applications

The estimation of QoS attributes in distributed architectures is particularly complex, and frameworks/tools are needed to help developers measure them to increase the success likelihood. Perses is a framework that allows developers to easily deploy a virtual scenario with multiple heterogeneous virtual mobile devices to evaluate the QoS of a distributed mobile application. These applications are characterized by distributed computing, in which some or all of the main computation that significantly affects QoS is performed on the end devices (executing activities such as data processing, running an AI model, etc.). In addition, these applications seek to enhance user privacy by storing data on end devices locally. Given an initial file with the configuration of the virtual scenario (infrastructure, network, devices, tests, etc.), Perses deploys it in a cloud instance.

Perses is fully scalable, allowing the evaluation of virtual scenarios formed by a large number of virtualized heterogeneous devices. The heterogeneity of virtual mobile devices is characterized by the possibility of deploying them with different hardware, operating system and configurations, being able to simulate close-to-real scenarios where there is a multitude of devices with different hardware and software characteristics.

Once deployed, Perses launches the tests, collects the logs from the devices, and analyzes the results. To support the evaluation of different dimensions of the QoE, Perses allows the execution of QoS tests and UX tests. First, it allows the configuration and definition of E2E tests evaluating different devices and the interactions among them, which are essential for evaluating QoS in distributed mobile applications. The UX tests are focused on evaluating the user interface with Espresso. Thus, Perses covers the most important dimensions evaluated by developers.

Finally, to save the time required for the manual execution of Perses, it is fully integrated into the DevOps methodology, automating the different steps during the evaluation process, and allowing software companies to easily integrate Perses into their continuous integration pipeline.

During the following subsections, first, the architecture of Perses is presented, detailing the characteristics of its modules; second, the configuration file is showed, where the characteristics of the virtual scenario, tests, etc. are defined, and finally, the integration of Perses in the DevOps methodology and its execution flow are described.

3.1. Architecture

Fig. 2 shows the architecture of Perses, formed by different modules focused on specific functionalities that complement each other. The most important modules are described below.

Setup: Perses requires a set of credentials and a configuration file as input. This file contains the characteristics of the virtual scenario to be deployed, the tests to be launched and the desired QoS attributes to be evaluated during the test execution. This module checks both the credentials to connect to the cloud infrastructure provider and the configuration file with the different parameters defined in the virtual scenario.

Deployment: The deployment module creates and deploys the entire virtual scenario taking the configuration file as input. To create and deploy the scenario, an abstraction layer is defined that encapsulates Terraform [17] — a framework with a high-level language that is used to define the deployment infrastructure of an application for cloud providers. Terraform has been extended so that in addition to deploying the cloud infrastructure, it also orchestrates the virtual scenario by installing the necessary resources, creating virtual mobile devices, and deploying distributed mobile applications on those devices. To host and deploy the virtual scenario, Amazon Web Services (AWS) is used as the cloud infrastructure provider. For the creation and deployment of virtual mobile devices, Docker[5] is used, where mobile devices are deployed in containers [18]. The management of these devices and the installation and deployment of the distributed application is performed through an Android Debug Bridge (ADB).[6]

Due to the network infrastructure provided by AWS, data is transmitted directly over the wired network. This is a significant difference to real mobile-cloud communication via WIFI or LTE. Kathara [19], a framework that enables network

[5] https://www.docker.com.

[6] https://developer.android.com/studio/command-line/adb.

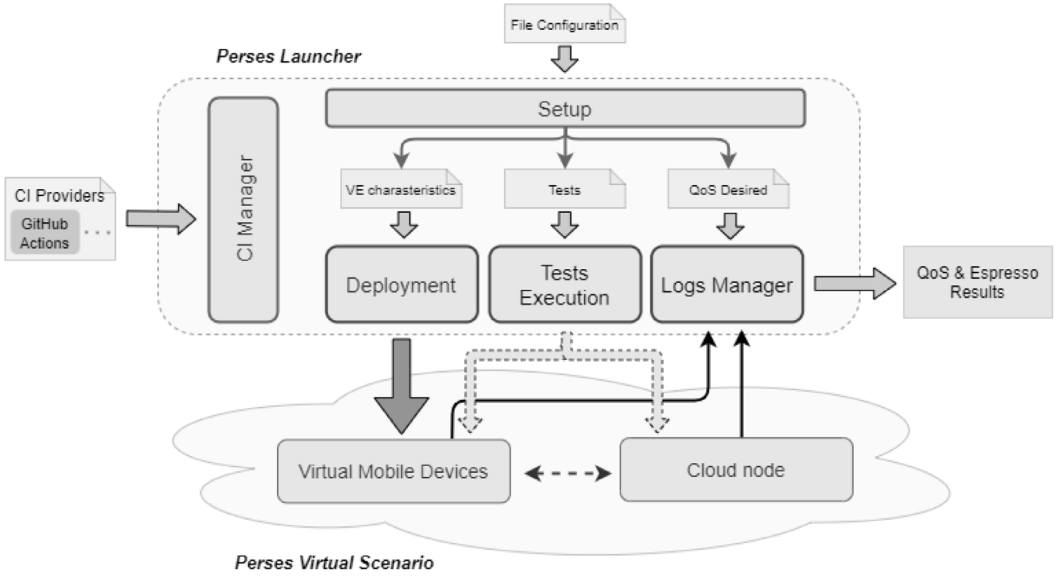


Fig. 2. General diagram of Perses.

emulation in Docker containers, has therefore been integrated. This allows us to emulate the network infrastructure of the virtual devices with the cloud environment to replicate real mobile-cloud communications.

Tests Execution: this module executes the tests defined in the configuration file. Perses allows two different types of tests to be run. *E2E tests*, these tests evaluate the QoS attributes of the application by executing the core functionalities that trigger the interactions among different devices. For this, Perses integrates APIPecker [20] — a simple API performance tester where different attributes (concurrent users, iterations and delay) can be defined to customize the tests, stress the devices, and obtain more information about the QoS. In addition, *user interface tests*, Perses allows developers to run Espresso tests to evaluate the UX and specific traditional functionalities.

Logs Manager: after launching the tests, this module collects the results obtained by aggregating the system logs of each of the virtual devices. After this, it analyzes the results and determines whether the desired QoS defined in the configuration file is achieved.

CI Manager: this module integrates with DevOps. This module autonomously manages the execution of the different Perses actions and modules. This makes it possible to automate the entire process of creating and deploying virtual scenarios and all the management related to the launch and analysis of tests. This automation is carried out through a workflow defined with GitHub Actions [21].

3.2. Defining virtual scenarios

Perses needs a configuration file (*.yaml* extension)[7] where the characteristics of the virtual scenario, the tests, and the desired QoS are defined. Fig. 3 shows a conceptual model with all the parameters in the configuration file. The explanation of each of the parameters and the user guide is detailed on Github.[8]

3.3. Integrating Perses in a DevOps methodology

One of the features of Perses is the full integration into the DevOps methodology. For this purpose, it is currently integrated with GitHub Actions, which makes it easy to automate software workflows applying CI/CD. To perform this integration, it is necessary to define a file with the different steps that the workflow must follow. This workflow has been developed to run any application, i.e., it is not necessary to define a workflow file for each application. Listing 1 shows an extract of the workflow, and the complete file is available on GitHub.[9]

Due to the integration of Perses in the DevOps methodology, the effort required by applying the defined framework is reduced. In addition, by automating the entire execution flow, the effort for developers is minimal, and repeatability is encouraged.

7 https://github.com/slasom/COVID-Heatmaps/.
8 https://github.com/perses-org/perses.
9 https://github.com/perses-org/gha/blob/master/workflow/perses-workflow.yml.

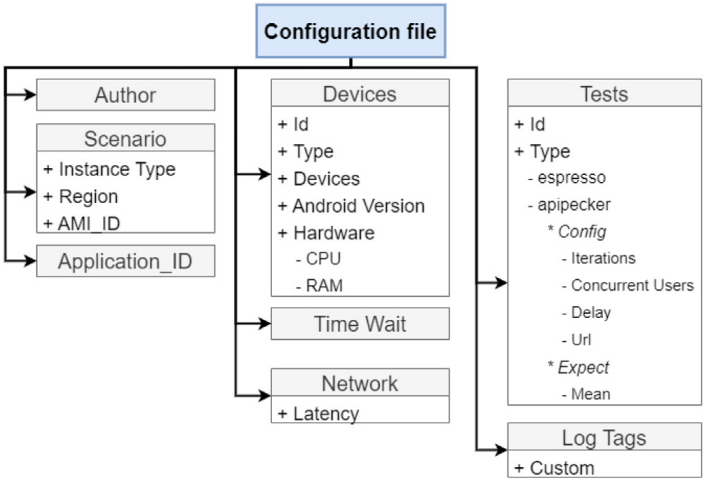


Fig. 3. Conceptual Model. Configuration File.

```
- name: "Build Android project"
uses: vgaidarji/android-github-actions-build@v1.0.1
with:
args: "./gradlew assembleDebug assembleAndroidTest"...
- name: "Perses Setup"
run: |
cd.perses_runner
node index -a setup -g../.perses-full.yml -c../.perses-credentials.yml
...
```

Listing 1: Perses Workflow

4. Experimental evaluation

Perses is a framework that allows the deployment of virtual scenarios to assess the QoS of distributed computing applications. However, although it can configure scenarios with heterogeneous devices, these devices are not real; they have virtualized images with limitations to consume a similar amount of resources as real devices. In this section, we present the evaluation of a case study in a real scenario with physical devices and with Perses to study the feasibility of the proposal and the accuracy of the results obtained.

In this section, first, the characteristics of both scenarios are presented (devices used, characteristics of the cloud node, characteristics of the Perses's virtual scenario, etc.). Second, the tests to be launched and the QoS parameters to be measured are explained. Finally, the results obtained, the comparison of both scenarios in terms of executing time, transfer time, etc., and their operational cost are analyzed and discussed.

The case study focuses on the evaluation of the viability of the proposal. For this purpose, the distributed computing application mentioned in Section 2 will be used in which several QoS parameters will be evaluated.

4.1. Experiment set-up

The experiment was carried out with seven physical mobile devices for the real scenario and seven virtual mobile devices for the virtual scenario. To make the test fair, each of the seven devices in each scenario was loaded with a specific set of locations so that the same volume of data was processed and transferred in both scenarios in the different tests.

The real scenario is composed of three Xiaomi Mi 9, two Mi 9T, OnePlus 6T, and Huawei Mate 20, which have similar hardware characteristics. All have 6 GB of RAM, and the processors are very similar (Qualcomm 855, 845, and 730 and Kirin 980). All of them have Android 9 as their OS.

The virtual scenario has been defined in the configuration file with all the features required to use Perses. More information about the configuration file can be found in the link.[10] To host and deploy the virtual scenario, we use a C5.metal EC2 instance of AWS. The set of virtual devices consists of seven devices, as in the real scenario. The hardware

10 https://github.com/slasom/Covid-Heatmaps/.

is limited to similar characteristics to those of real devices. Otherwise, as they are located in a fairly powerful instance, they clearly outperform physical devices. Each of them is composed of three CPUs, 6 GB of RAM, and Android 9 as the OS.

The cloud node is the same for both scenarios. It has been deployed on a T2.large EC2 of AWS. For communication and data transfer with mobile devices, the MQTT [22] communication protocol was used.

4.1.1. Scenario evaluation set-up

To evaluate the QoS parameters, a set of tests with different configurations is launched based on the main functionality of the application, i.e., to search for close contacts after a COVID-19-positive user registration.

The different configurations in the tests are linked to the definition of different 'positive users' with the set of synthetic locations that has been created to obtain more varied results. When a positive user is registered, one, three, five and seven devices real/virtual respond to the request with a heatmap of 100, 1,000, 5,000 and 10,000 location points. These increments are defined to evaluate Perses in the face of increased mobile device and data/computer complexity. Each of the tests was repeated four times to obtain consistent results. To check the dispersion of the mean of the test repetitions, the coefficient of variation was calculated, obtaining a maximum value of 0.078 among all the results.

Finally, three QoS parameters are measured that allow us to compare Perses with a real scenario. **Mobile-Side execution time** is the time it takes for mobile devices to process their heatmaps. **Transfer time** measures the time it takes to send the data from the cloud side to the mobile side and vice versa. **Aggregation time** captures the time the cloud spends processing and comparing the heatmaps to detect possible close contacts.

4.1.2. Results of the experiments

This last subsection presents the results obtained after the evaluation of the COVID-Heatmaps app in both scenarios.

Mobile-Side Execution time: Fig. 4(a) shows the computing time on real and virtual mobile devices for the different numbers of devices involved and location points exchanged. It can be seen that they follow the same trend in both scenarios. There is a small gap between both scenarios; this is due to the existing hardware inequality, and the processors of real mobile devices are less powerful due to their size, architecture, etc.

To better analyze the differences between the two scenarios and the existing gap, Figs. 4(b) and 4(c) show the difference between the plane for the real scenario and the plane for the virtualized scenario. Fig. 4(b) shows a 2D graph with different lines for the different data consumption as the number of devices increases. Fig. 4(c) shows different lines for the different number of mobile devices as the quantity of data to be processed increases.

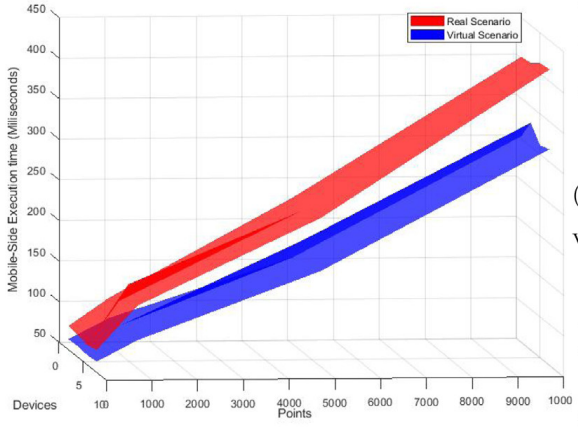
The graph 4(b) shows that for 100 location points, the difference is approximately 28 ms on average for the whole set of devices, as each device computes its heat map concurrently and therefore the computation time is very similar. The same is true for the other sets of points sent; for 1,000 location points, the difference is slightly higher, 46 ms. For 5,000 location points, the difference increases to 80 ms. Finally, for 10,000 location points sent, the difference is 134 ms on average. This increase can be better seen in Fig. 4(c), showing that for every configuration (number of devices), the execution time increases with the same trend. However, the increase is the same with respect to the total time, and the dashed pink line shows the percentage of the difference in the execution times between both scenarios. As seen in Figs. 4(b) and 4(c), percentage-wise, the difference is constant at approximately 34%. This gap cannot be reduced due to the minimum hardware requirements for virtual devices to be deployed. If they are further constrained to match real devices, they do not have sufficient capacity to deploy the docker container. However, as can be seen this gap constant, therefore extrapolation can be applied in order to obtain results closer to the real devices.

Finally, Fig. 5(a) shows the results obtained with Perses after a test with different sets of virtual mobile devices to evaluate scalability, deploying up to 50 simultaneous devices. An increase in execution time can be observed as the number of points increases. However, increasing the number of devices does not affect the performance because the execution is performed in parallel on each device.

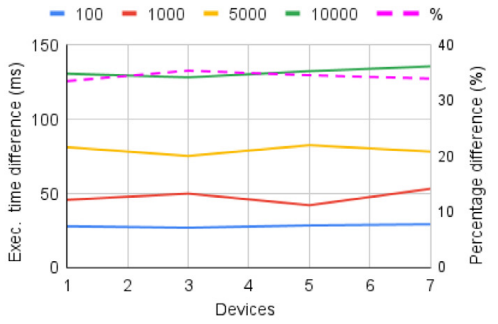
Transfer time: Fig. 6(a) shows the results obtained on the data transfer time. As with Mobile-Side Execution time, they follow the same trend with the constant difference between the two scenarios. In this case, the difference is minimal, the integration of Kathara with Perses allows emulating the communication of real mobile devices. In the real scenario, the mobile devices receive the request and transfer the data via WiFi technology plus the existing distance to the location of the cloud node deployed on the AWS servers. In the virtual scenario, the data is transmitted directly over the wired network. So without network emulation, the transfer time of virtual mobile devices would not come close to that of the real scenario.

As before, Figs. 6(b) and 6(c) show the difference between the real and virtualized scenario planes. Fig. 6(b) shows different lines for the different numbers of points. Fig. 6(c) shows different lines for each set of responding devices. Both figures show a difference between ± 25 ms. This means that in certain cases, the virtual scenario transfers data faster than the real scenario and vice versa. The percentage of difference between both scenarios is between 3%–5%, a minimal variation. Therefore, Perses allows developer to simulate the network infrastructure and execute the tests over it.

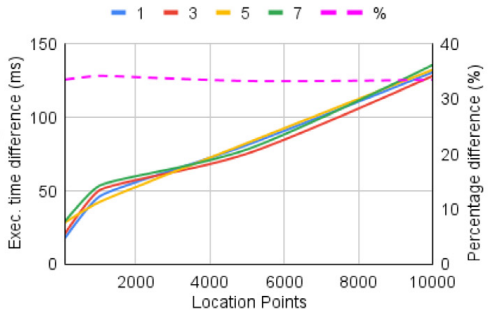
Finally, Fig. 5(b) shows the results obtained with the scalability test. We can observe a growth of the transfer time both when increasing the number of points and the number of virtual devices. The growth due to the increase in the amount of virtual devices is caused by the increase in the number of virtual devices and the increase in the coordination time of all the responses received by the cloud.



(a) Execution time

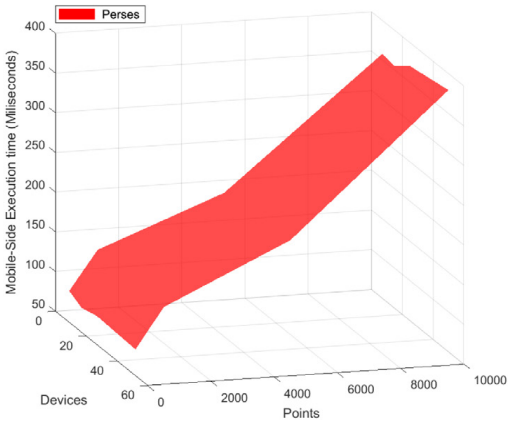


(b) Differences depending on the num of devices

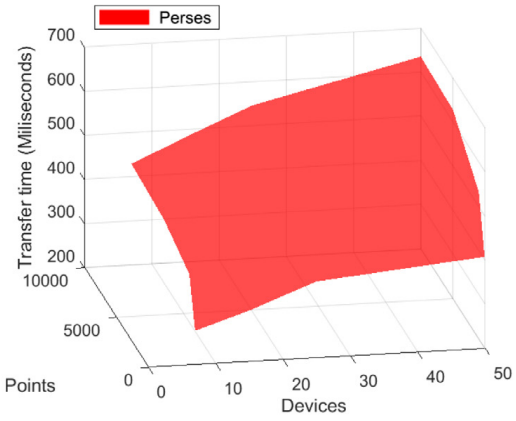


(c) Differences depending on the num of points

Fig. 4. Mobile-Side Execution time results.



(a) Mobile-Side Execution time large-scale



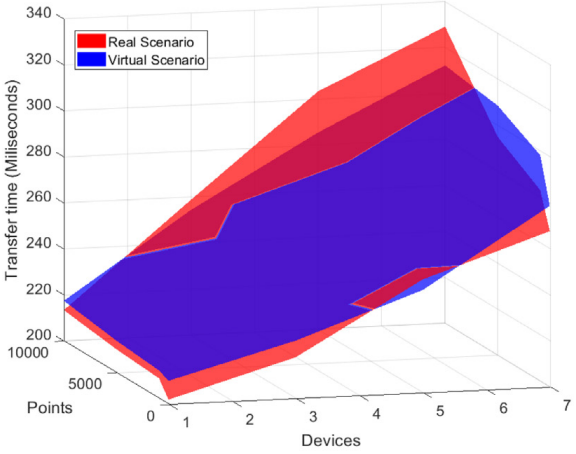
(b) Transfer time large-scale

Fig. 5. Large-scale test results.

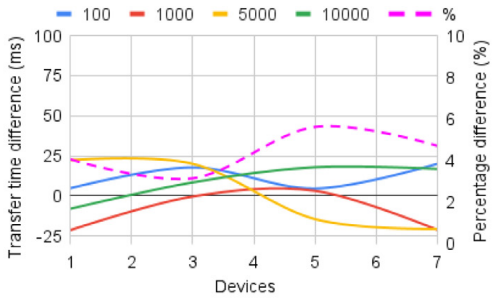
Aggregation time: Fig. 7(a) shows the aggregation time in the cloud node of the heatmaps for the different numbers of devices involved and location points sent. In this case, it can be seen that the planes obtained from both scenarios are practically the same. The cloud node is the same for both scenarios, the number of devices involved and the volume of data are the same. Figs. 7(b) and 7(c) show the differences in aggregation time between the two scenarios as a function of the number of devices and the amount of information shared, respectively. A constant trend is observed horizontally, and there is hardly any difference (between 0.01 and −0.01), as expected.

Operational costs: This last section shows the differences concerning the operational costs required for testing the application in both scenarios.

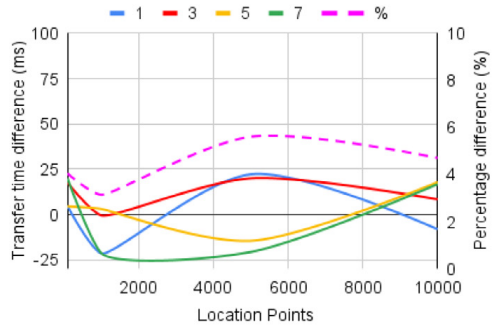
The real scenario had a cost composed of the physical smartphones and the AWS infrastructure. The price of the smartphones (launch price) gives a total amount of $2,804. With respect to the AWS infrastructure, a T2.large instance was used in the Ireland region at a cost of $0.1008/hours. The duration of the tests in this scenario lasted 5 min or 0.083 h, resulting in $0.009. Therefore, the real scenario had a cost of $2,804.01



(a) Transfer time

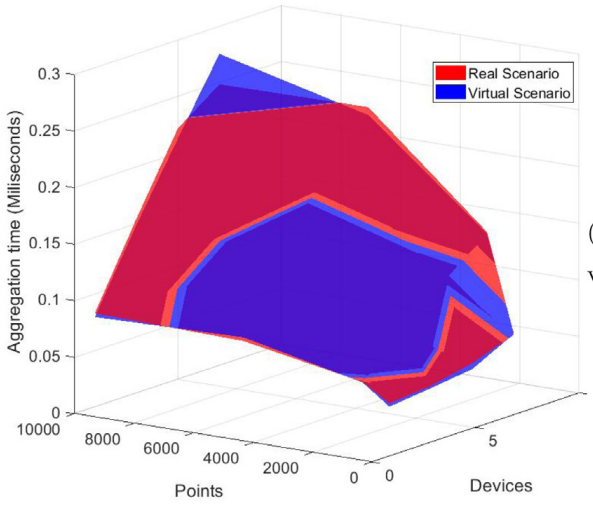


(b) Differences depending on the num of devices

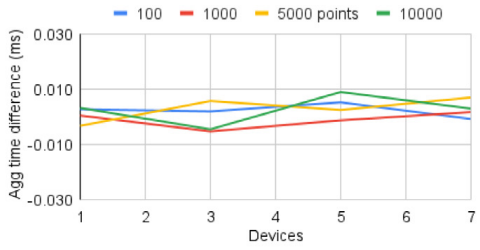


(c) Differences depending on the num of points

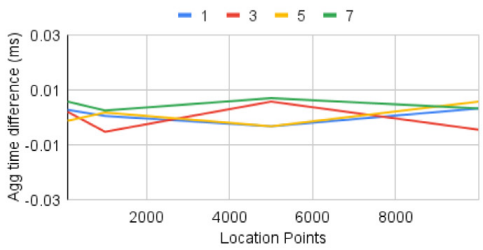
Fig. 6. Transfer time results.



(a) Aggregation time



(b) Differences depending on the num of devices



(c) Differences depending on the num of points

Fig. 7. Aggregation time results.

The virtual scenario had a cost composed of the AWS infrastructure used to host and use the scenario with the virtual devices, the cloud node, and the MQTT communication protocol. The virtual scenario was hosted on a C5.metal instance in the Ireland region for $4.608/hours. As before, the cloud node and the MQTT protocol were hosted on the same instance. These tests took 20 min or 0.33 h. Therefore, a cost of $1.536 for the virtual scenario and $0.033 for the cloud node and the MQTT protocol was obtained. Therefore, the virtual scenario had a cost of $1.569.

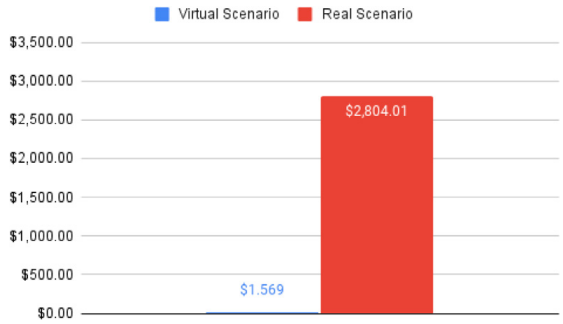


Fig. 8. Operational costs.

Fig. 8 shows the total costs for each of the scenarios. There is a large difference between testing in a real scenario and testing with the Perses framework. For the cost of deploying the real scenario, we performed Perses's evaluation on applications with similar characteristics 1,787 times.

5. Related works

There are different frameworks and commercial platforms for testing mobile applications. In addition to the aforementioned AWS Device Farm and Azure App Center Test, there are other tests, such as the Firebase Test Lab [23] and Perfecto [24]. All of them offer different functionalities and features for testing mobile applications with real and virtual devices (selection of different models and hardware characteristics, video reports of results, etc.). They are also easily integrated into different development tools and in CI/CD pipelines of the software development cycle of companies. However, they are focused on UI testing, i.e., they do not evaluate QoS parameters such as response time and latencies, as they are designed for mobile applications with client–server architectures.

At the research level, there are some proposals detailing frameworks, tools, and techniques for evaluating distributed applications focused on the IoT. In [25], the authors proposed a framework for developing and evaluating component-based distributed systems for heterogeneous scenarios, considering mobile and fixed networks. It is an interesting proposal and similar to our work focusing on distributed applications. However, it is not clear what the workflow of the platform was and what the development and evaluation process was. Moreover, it is not possible to evaluate standard applications, only those developed on the platform. The focus was specifically on the evaluation of applications developed with the proposed framework. Therefore, approaches are needed that can also evaluate any applications that can be deployed on the target devices.

In [26], the authors presented a framework for automated IoT application testing. The framework allows the automated execution of user-defined experiments and can generate virtual testbeds with adjustable network properties. To do so, they virtualize devices acting as fog and edge devices using the QEMU emulator. The authors developed a prototype of the framework and, in the experiment, performed experiments with physical and emulated Raspberry Pi 3. The framework is promising; however, it does not allow the creation of heterogeneous testbeds. All emulated devices run under the same operating system and the same defined features. Finally, they do not integrate the defined framework in a development cycle for companies.

In [27], the authors presented a simulator that enables the design and analysis of large-scale IoT systems for smart city applications consisting of mobile devices. It supports the simulation of devices, gateways, and applications. It also supports environment modeling (including movement, interference, etc.). They explain and simulate an IoT system that is deployed in Leuven. Nevertheless, it is intended for specific case studies of smart city applications.

In [28], the authors proposed a system for testing the usability and performance of Android mobile applications with virtual reality. This system simulates different network and mobility behaviors to evaluate applications in close-to-real environments with different network configurations. However, they focused on manually testing the application using a single device, which makes it impossible to evaluate distributed mobile applications. Furthermore, the tests performed focused on the user experience, and no QoS-related parameters were collected.

6. Conclusion

The market for mobile applications has grown at a frenetic pace. Moreover, the capacities of mobile devices have increased considerably. With this, new paradigms and distributed architectural designs have emerged for developing mobile applications exploiting these capabilities to further improve the QoS. However, new tools are needed to easily assess these distributed and highly heterogeneous environments and to ensure application quality.

In this paper, we presented a framework called Perses, which allows developers to create virtual scenarios and can deploy a large number of virtual mobile devices to correctly evaluate the QoS. This framework was fully integrated into a DevOps software development flow, which allows automating the tasks of creating and deploying the virtual scenario,

launching E2E tests, collecting results, etc., which reduces the effort required to perform these tests. This is a fundamental aspect so that it can be better embraced by software development companies. Finally, the framework was evaluated with a case study, and the results obtained were comparable to those obtained from a real scenario.

In future work, we are working on allowing the emulation of new devices (IoT devices). This will increase the heterogeneity of the scenarios. We are also working on providing different network topologies with Kathara (distribution of devices connected to different nodes simulating the connection to different WIFI or LTE connections). Finally, we are also working on automatically generating a cost analysis to allow developers to visualize the costs that will be generated before the deployment of the virtual scenario, being able to adjust it to the budget they have defined.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially funded by the DIN2020-011586 grant, funded by MCIN/AEI/10.13039/501100011033 and by the European Union "NextGenerationEU/PRTR", by projects RTI2018-094591-B-I00 (MCI/AEI/FEDER,UE) and RTI2018-101204-B-C21, the 4IE+ Project (0499-4IE-PLUS-4-E) funded by Interreg V-A España-Portugal (POCTEP) 2014–2020 program, by the RCIS network (TIN2016-81978-REDT), by the Department of Economy, Science and Digital Agenda of the Government of Extremadura (GR21133, IB18030), by the Government of Andalusian (US-1264651) and by the European Regional Development Fund.

References

[1] Statista Research Department, Number of apps available in leading app stores 2020, 2021, URL https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. (Accessed April 13 2021).
[2] J. Parker, 10 Years of growth of mobile app market, 2018, https://www.knowband.com/blog/es/mobile-app-es/10-years-of-growth-of-mobile-app-market/. (Accessed 13 February 2021).
[3] K. De Moor, I. Ketyko, W. Joseph, T. Deryckere, L. De Marez, L. Martens, G. Verleye, Proposed framework for evaluating quality of experience in a mobile, testbed-oriented living lab setting, Mob. Netw. Appl. 15 (3) (2010) 378–391.
[4] H.J. Kim, D.H. Lee, J.M. Lee, K.H. Lee, W. Lyu, S.G. Choi, The QoE evaluation method through the qos-qoe correlation model, in: 2008 Fourth International Conference on Networked Computing and Advanced Information Management, vol. 2, 2008, pp. 719–725, http://dx.doi.org/10.1109/NCM.2008.202.
[5] B. Lima, Automated scenario-based integration testing of distributed systems, in: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2018, pp. 956–958.
[6] J. Berrocal, J. Garcia-Alonso, C. Vicente-Chicote, J. Hernández, T. Mikkonen, C. Canal, J.M. Murillo, Early analysis of resource consumption patterns in mobile applications, Pervasive Mob. Comput. (ISSN: 1574-1192) 35 (2017) 32–50, http://dx.doi.org/10.1016/j.pmcj.2016.06.011, URL http://www.sciencedirect.com/science/article/pii/S1574119216300797.
[7] S. Laso, J. Berrocal, J. García-Alonso, C. Canal, J. Manuel Murillo, Human microservices: A framework for turning humans into service providers, Softw. - Pract. Exp. (2021).
[8] J. Miranda, N. Mäkitalo, J. Garcia-Alonso, J. Berrocal, T. Mikkonen, C. Canal, J.M. Murillo, From the internet of things to the internet of people, IEEE Internet Comput. 19 (2) (2015) 40–47.
[9] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J.P. Jue, All one needs to know about fog computing and related edge computing paradigms: A complete survey, J. Syst. Archit. (2019).
[10] Y. Qian, L. Hu, J. Chen, X. Guan, M.M. Hassan, A. Alelaiwi, Privacy-aware service placement for mobile edge computing via federated learning, Inform. Sci. 505 (2019) 562–570.
[11] K. Lin, C. Li, Y. Li, C. Savaglio, G. Fortino, Distributed learning for vehicle routing decision in software defined internet of vehicles, IEEE Trans. Intell. Transp. Syst. 22 (6) (2020) 3730–3741.
[12] Amazon Web Service, AWS device farm, 2021, https://aws.amazon.com/device-farm/. (Accessed 23 March 2022).
[13] M. Azure, App center test, 2021, https://docs.microsoft.com/en-us/appcenter/test-cloud/. (Accessed 28 May 2021).
[14] W.N. Picoto, R. Duarte, I. Pinto, Uncovering top-ranking factors for mobile apps through a multimethod approach, J. Bus. Res. 101 (2019) 668–674.
[15] A.P. Gamerant, Pokemon go worldwide launch halted to fix server problems, 2016, https://gamerant.com/pokemon-go-server-launch-problems/. (Accessed 28 October 2021).
[16] S. Insights, Top 7 biggest flops in the mobile app industry, 2019, https://www.smartinsights.com/mobile-marketing/top-7-biggest-flops-mobile-app-industry/. (Accessed 28 January 2021).
[17] HashiCorp, Terraform, 2021, https://www.terraform.io/. (Accessed 22 April 2021).
[18] Budtmo, Docker android, 2021, https://github.com/budtmo/docker-android. (Accessed 23 March 2022).
[19] M. Scazzariello, L. Ariemma, T. Caiazzi, Kathará: A lightweight network emulation system, in: NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2020, pp. 1–2.
[20] P. Fernandez, API pecker, 2020, https://www.npmjs.com/package/apipecker. (Accessed 10 March 2022).
[21] Github, GitHub actions, 2021, https://github.com/features/actions. (Accessed 27 July 2021).
[22] MQTT, MQTT, 2021, URL http://mqtt.org/. (Accessed 22 March 2021).
[23] Firebase, Firebase test lab, 2021, https://firebase.google.com/products/test-lab. (Accessed 13 March 2022).
[24] Perforce Software, Perfecto, 2021, https://www.perfecto.io/. (Accessed 23 March 2022).
[25] B. Richerzhagen, D. Stingl, J. Ruckert, R. Steinmetz, Simonstrator: Simulation and prototyping platform for distributed mobile applications, in: The 8th EAI International Conference on Simulation Tools and Techniques (ACM SIMUTOOLS 2015), IMDEA Networks Institute Publications Repository, 2015.

[26] I. Behnke, L. Thamsen, O. Kao, Héctor: A framework for testing IoT applications across heterogeneous edge and cloud testbeds, in: Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion, 2019, pp. 15–20.
[27] M. Provoost, D. Weyns, DingNet: A simulator for large-scale IoT systems with mobile devices, in: EWSN, 2019, pp. 267–269.
[28] T. Amano, S. Kajita, H. Yamaguchi, T. Higashino, M. Takai, Smartphone applications testbed using virtual reality, in: Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2018, pp. 422–431.

3.5 Elastic Data Analytics for the Cloud-to-Things Continuum

Authors: *Sergio Laso, Javier Berrocal, Pablo Fernández, José María García, Jose García-Alonso, Juan M. Murillo, Antonio Ruiz-Cortés and Schahram Dustdar.*

Publication type: *Journal paper.*

Journal: *IEEE Internet Computing.*

DOI: *10.1109/MIC.2021.3138153.*

Quality indicator: *JCR Q2.*

EDITOR: Schahram Dustdar, dustdar@dsg.tuwien.ac.at

DEPARTMENT: INTERNET OF THINGS, PEOPLE, AND PROCESSES

Elastic Data Analytics for the Cloud-to-Things Continuum

Sergio Laso, *Global Process and Product Improvement, 10160 , Cáceres, Spain*

Javier Berrocal, *University of Extremadura, 06006 Badajoz, Spain*

Pablo Fernandez and José María García, *Universidad de Sevilla, 41004 Sevilla, Spain*

Jose Garcia-Alonso and Juan M. Murillo, *University of Extremadura, 06006 Badajoz, Spain*

Antonio Ruiz-Cortés, *Universidad de Sevilla, 41004 Sevilla, Spain*

Schahram Dustdar, *TU Wien, 1040 Vienna, Austria*

The massive deployment of Internet-connected devices has led to an increase in the collection of data that are then used by companies to improve their decision-making processes. This growing trend demands more and more cloud and communications infrastructure. The limited resources, the need for sharing them, and the fact that many consumers are interested in the same data, call for an efficient management of the available resources. The cloud-to-things continuum can be used to execute different analytics closer to the data source so that infrastructure consumption and data circulation can be optimized. In this article, different dimensions for achieving elastic analytics and a framework for dynamically modifying their behavior are proposed.

Over the last few years, there has been a massive deployment of Internet of Things (IoT) devices, from heart rate sensors to location monitoring devices. This deployment has been especially driven by IoT applications that facilitate everyday tasks for every individual and organization. These tasks cover a wide range of use cases and applications, from managing an individual's personal health to understanding the flow of people and movement patterns in a city. This massive deployment of IoT devices has also led to the creation of networks of smart devices. This growing use of IoT devices is putting stress on current infrastructures in different dimensions.

On the one hand, it is putting stress on the amount of devices deployed. As IoT applications become more complex, they require combining devices from different domains for offering more useful functionalities. So far, new IoT applications and functionalities require the deployment of new devices. However, in this growing trend, the deployment of new IoT devices for sensing similar data is becoming unfeasible and unsustainable. For instance, a city that monitors the movement patterns of its citizens may also want to combine the location information with the heart rate in order to know what kind of activity citizens tend to do in each area of the city and, thus, provide more useful services and better plan their investments. For that purpose, instead of asking citizens to wear new heart rate monitors, it would be more feasible to ask them to share the data obtained by the monitors they already wear for other apps. Thus, in coming years, we will see how IoT applications will share large networks of devices already deployed.

On the other hand, it is putting stress on the infrastructure for data circulation and processing. Data, especially those coming from IoT devices, have become a strategic asset because of their availability to continuously monitor the environment without human intervention. IoT device networks are enablers of a new data economy in which more and more data are being exchanged not only within but also amongst

1089-7801 © 2022 IEEE
Digital Object Identifier 10.1109/MIC.2021.3138153
Date of current version 23 December 2022.

companies.[1] This fact boosts the massive deployment of IoT device networks. However, it also entails an increase in the circulation of information and the need for its processing and, consequently, a management model to support the optimal network usage. Such requirements will be even harder when different information systems require data from the same devices, for different needs and with different qualities. For example, a system to monitor an individual's heart rate during his/her activities may require very high information freshness; however, for a municipality to plan its services, such information does not have to be very fresh, but it has to be obtained from as many devices as possible.

In order to reduce the network overhead and improve the quality of service (QoS), IoT applications already attempt to take advantage of the cloud-to-things continuum to deploy services closer to information providers and consumers. However, such applications are still isolated systems that do not favor the sharing of data or analytics streaming. Thus, a sustainable data economy in this market[2] poses many challenges, among which we can count the following.

- First, the shared use of IoT devices so that data they are sensing can be used by different applications minimizing the drain on resources.
- Second, the optimization of the use of the cloud-to-things continuum infrastructure by enabling optimized deployment so IoT applications interested in the same data streaming and analytics can be deployed as close to the data as possible and together in the same nodes of this continuum.
- Third, elimination of duplicate streams by enabling applications interested in the same data or analytical streams to use the same datum.

In this article, we address these challenges by using elastic IoT data analytics whose behavior can be dynamically modified according to the quality requirements defined by each IoT system and the available resources of the cloud-to-thing infrastructure. For defining and deploying these analytics, we also provide a framework with an orchestrator managing the elasticity. This orchestrator evaluates the state of the infrastructure, the other analytics in progress, and the QoS required by each analytic. As a result, it reconfigures all the analytics to maximize the quality provided by each of them without exhausting the infrastructure nor the IoT devices.

In the following, we first discuss the different dimensions that should be taken into account for building elastic data analytics. Then, we present a framework for achieving elastic data analytics considering some of the defined dimensions. Finally, we present some conclusions and discussion about the elasticity required by IoT applications.

ENABLING ELASTIC DATA ANALYTICS

Data analytics allow companies to process large volumes of information in order to: find trends, eliminate unnecessary information, aggregate information, etc. In addition, they can enable efficient information management, as they may reduce the overhead of the transferred data. However, they may also require data storage and processing capabilities from the cloud-to-things continuum nodes. These analytics must be defined by each data consumer, since the results must be adapted to their requirements. Nevertheless, the circulation of information should be controlled to increase the efficiency in the use of both the system and the data.

Let us define a smart city scenario to better show the need of elastic data analytics. In order to maintain the simplicity, this case study is focused only on the citizens' location. Different institutions can analyze this information to identify their movement patterns, but with different goals. For instance, a healthcare system can use the real-time movements of the citizens to identify crowded areas and reduce the risk of infection for COVID-19. Instead, the city council may require long-term movement patterns to study future infrastructure investments (accessibility, bus lines, etc.).

The execution of these analytics impacts on several highly related dimensions, as Figure 1(a) shows, that must be controlled in a sustainable computing and information environment: data quality, resource consumption and cost.

First, one of the most important aspects for data consumers is the quality of the information. This quality must be appropriate for the specific functionalities they want to provide. While there are different properties to measure data quality, for information coming from IoT devices two properties are particularly relevant.[3]

1) *Accuracy:* It is the level to which data represent the real-world scenario. In this sense, in a network of IoT devices, the higher the number of devices involved in the data analytic the better they will represent the real world.
2) *Freshness* or the timeliness of the data: For some applications, the data have no value if it is

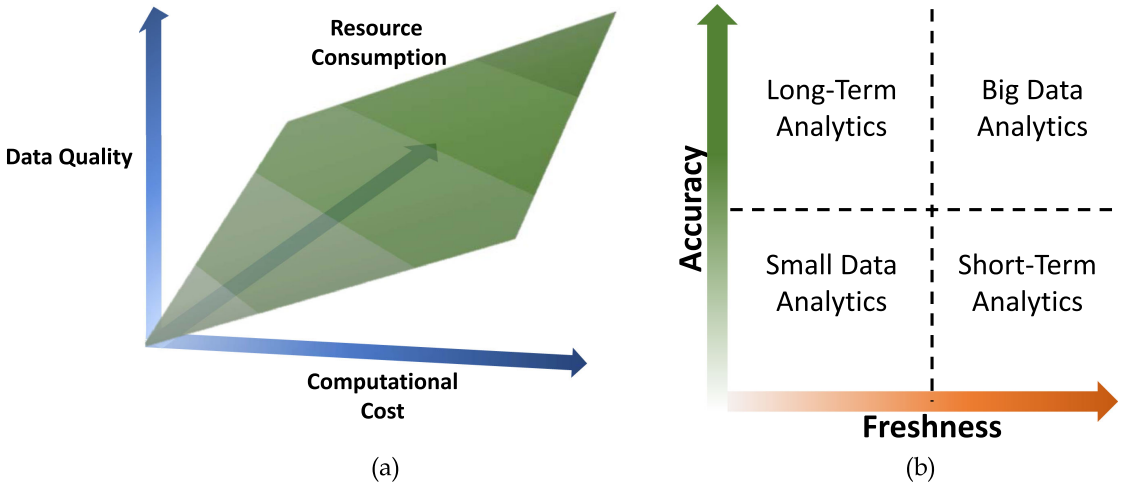


FIGURE 1. (a) Dimensions impacting the analytics. (b) Analytics depending on the data quality.

not available at the right moment. Freshness indicates the frequency at which IoT devices have to provide information for the analytic to provide value to the consumer.

Combining both dimensions, as Figure 1(b) shows, different types of analytics can be executed. When the accuracy and the freshness are low, small data analytics are executed in order to have a limited quantity of highly granular data that usually provide valuable information for the system. As the freshness and the accuracy increase, bigger data analytics are executed focusing on processing large volumes of information for business decisions. Instead, if only the accuracy increases, long-term data analytics are executed for predictive decision-making processes. In addition, if mainly the freshness is important for achieving an acceptable data quality, short-term data analytics are usually executed for developing reactive processes. For instance, for our case study, the healthcare system would require short-term analytics while the city council would require long-term analytics.

Second, this type of analytics has a direct impact on the resource consumption of the nodes in the cloud-to-things continuum. A greater accuracy means that more IoT devices will be involved in sending information, consuming their resources, and increasing the number of information flows. Edge and fog nodes can be used to distributedly process such analytics, reducing and aggregating the information flows. However, the greater the dispersion, the greater the number of nodes involved. Therefore, a high accuracy usually leads to a greater distribution of the resource consumption over the entire network. For our case study, the city council may require to use different nodes to cover the whole city, storing and aggregating the movement patterns whilst the healthcare system would only require a sample of some areas.

Finally, in addition to the data cost (which may depend on each data producer), its processing also entails some infrastructure costs associated with the use of edge and fog nodes.[4] Thus, the greater the freshness and frequency at which the information must be processed, the greater the need for processing this information in the fog or edge nodes, and the higher the infrastructure cost of the analytics. This cost may be limited to a small number of fog or edge nodes if the accuracy is low or to a larger one as the accuracy increases. For instance, for the healthcare analytics only the nodes involved in crowded areas would incur a higher cost. Furthermore, in a leveraged scenario with an open market of sensors,[5] this cost could also be dynamic depending on the fluctuations of the market.

In environments where multiple IoT applications are consuming similar information and using the same computing resources, data analytics should be able to have an elastic behavior.[6] They should be able to increase or decrease the provided quality depending on the available resources (and within the consumer's requirements). In addition, a framework would be needed to efficiently manage the distribution of all the requested analytics on the cloud-to-things continuum. For each analytic, this framework should identify in which fog or edge nodes it should be deployed to meet the accuracy and freshness requirements, without overloading the infrastructure. Likewise, when a new analytic is requested, the framework should use the defined elasticity to reconfigure the already existing analytics in order to achieve an efficient use of the

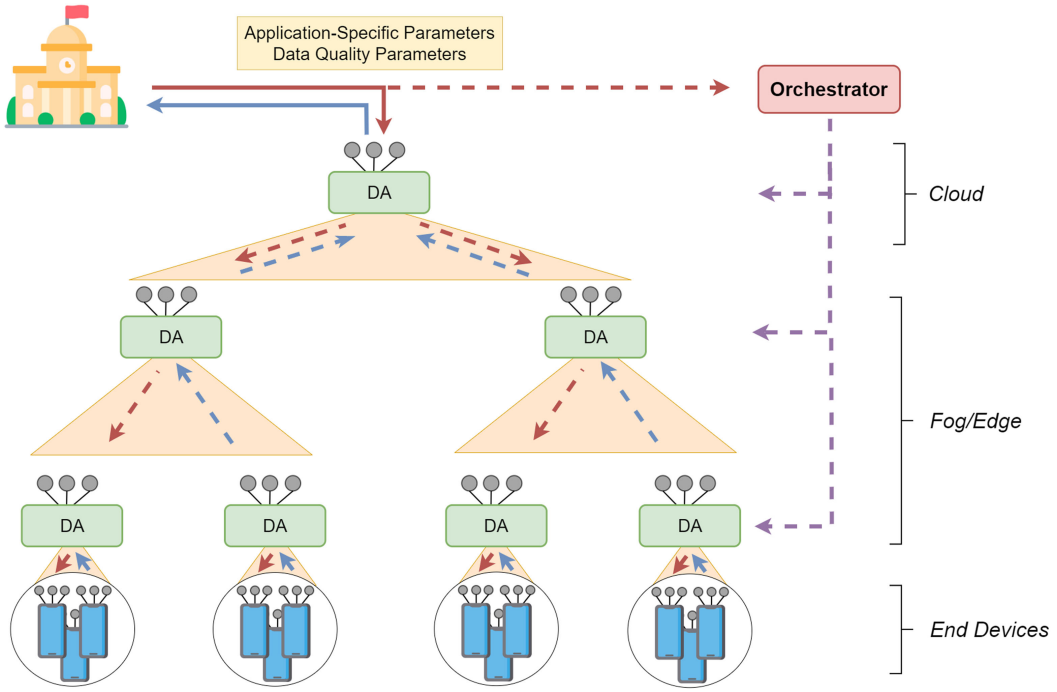


FIGURE 2. Architecture of the elastic data analytic framework.

computing resources, while maximizing the provided quality. Moreover, this distribution should also consider if several analytics should be deployed together, because they can share the same sensed data and, thus, increase the general efficiency of the system.

REALIZATION AND APPLICATION OF THE FRAMEWORK

To achieve elastic data analytics, we propose a framework orchestrating the analytics required by different data consumers and leveraging the cloud-to-things continuum to distribute them depending on the available resources and the required quality. The source code of the framework[a] for the presented case study and a video[b] showing the achieved elasticity are publicly available.

Figure 2 shows an example of the infrastructure that can be used for our smart city case study. It shows the different layers of the cloud-to-things continuum. Note that we limited the number of layers to improve readability. Data consumers (healthcare system and city council, for our case study) can request different analytics through the entry point to the infrastructure, the cloud. The deepest layer is composed by the end devices sensing and sending information to the cloud. This information goes through the fog and edge nodes on its way to the cloud where it is provided to data consumers.

Data analytics can be requested with different parameters. First, the application-specific parameters (for instance, the area to monitor in the smart city) and, second, the data quality parameters (accuracy and freshness). For the current implementation, the accuracy and the freshness can take different values from a range (low, medium, and high). The accuracy relates to the ratio of end devices involved, and the freshness to the frequency at which the information is obtained.

Figure 2 also defines the most important components of the proposed framework. First, cloud, fog, and edge nodes are composed by a data aggregator (DA) component. This component processes the deployed analytics. To that end, it requests the required information to the lower nodes depending on the data quality (mainly the freshness) required by the most demanding analytic, caches the obtained data to be reused by the other analytics, and processes it. In addition, the obtained results are also cached and always available, waiting for the higher levels to request them.

In addition, an elasticity orchestrator is proposed to balance the load of the whole infrastructure, modify the behavior of the elastic analytics depending on the previously defined dimensions, and to provide the best QoS to all parties. When a new analytic is required, it is analyzed by the orchestrator to evaluate the desired quality

[a][Online]. Available: https://doi.org/10.5281/zenodo.5793214
[b][Online]. Available: https://doi.org/10.5281/zenodo.5793189

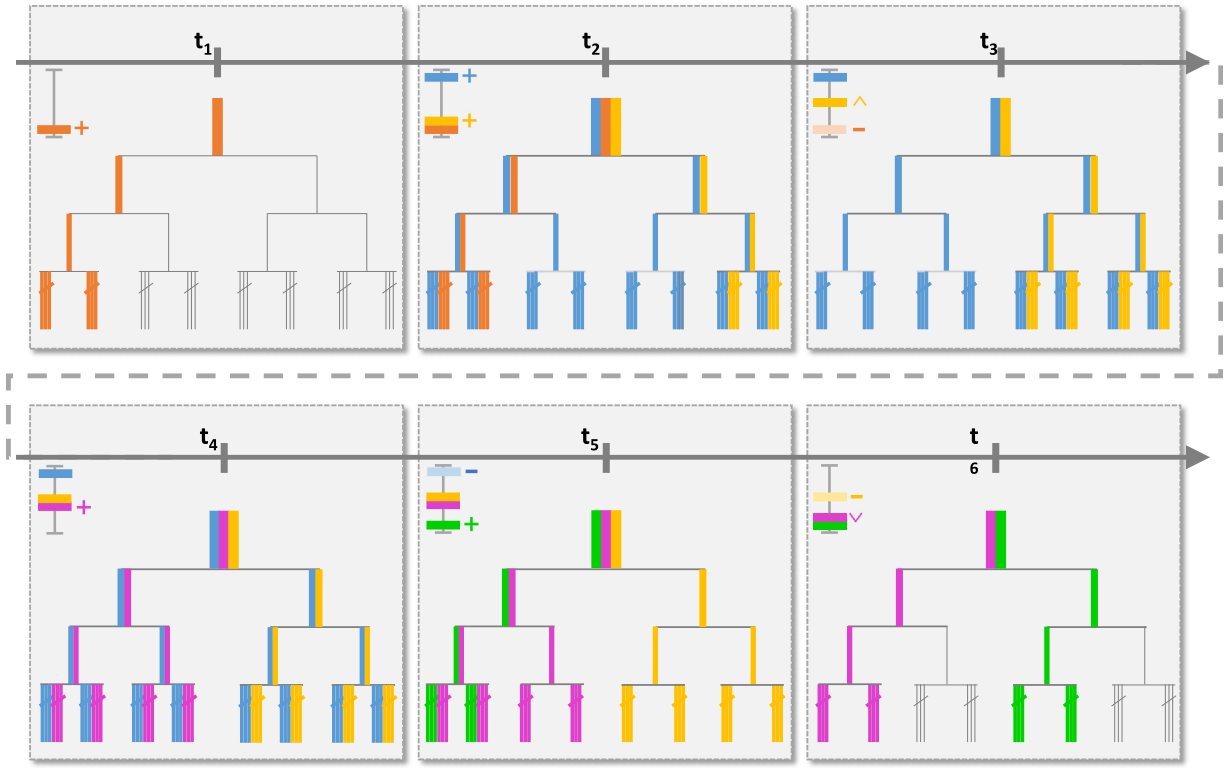


FIGURE 3. Elasticity dynamics.

(accuracy and freshness), the analytics already deployed in the architecture, and the workload of the different nodes. If there are enough resources, it directly deploys it providing the highest possible quality. If there are not enough resources, it checks if the analytics already deployed can be reconfigured to make room for the new one. This reconfiguration can be provided by redeploying existing analytics on other nodes to have a more efficient distribution or, if possible due to the quality defined by consumers, reducing the accuracy and freshness of some analytics to free up some resources.

Finally, all the DAs have the same communication interface (API). Thus, the infrastructure can be easily increased or reduced without heavily impacting the orchestrator. This also facilitates the work of developers and operators, avoiding having to create *ad hoc* solutions.

Figure 3 shows a working example of the provided elasticity. The example consists of a timeline with six

Analytic 1: Healthcare system, identification of crowded areas

Analytic 2: City council, study of movement patterns

FIGURE 4. Analytics obtained for the smart city case study.

DATA MANAGEMENT IN THE CLOUD-TO-THINGS CONTINUUM

Cloud computing has not only been a revolution at the business level to reduce IT costs. It has also fostered the massive deployment of new Internet-connected devices, with reduced computing resources but with huge capabilities for sensing and interacting with the environment. This has led to a steady increase in the data flows between these connected devices and the cloud. Paradigms such as fog and edge computing have brought the cloud environments, and hence the information processing, closer to the data sources. In fog computing, the processing is done on servers residing between the cloud and end devices, while edge computing focuses on processing the information at the edge of the network (gateways and end devices). By processing the information closer to the data source, the infrastructure overhead and the QoS can be improved.[1]

These paradigms allow the generation of distributed clouds making use of the cloud-to-thing continuum for the deployment of services throughout the entire infrastructure.[2] These services, some of them focused on data processing, can dynamically migrate from one layer to another depending on the volume of information to be processed, the workload of the nodes, or the QoS desired. Different technologies even orchestrate these distributed services to execute complex workflows.[3]

The incorporation of intelligence to the cloud-to-thing continuum is allowing researchers to take a further step in the processing of these huge data flows in order to get predictions and automate the decision-making process. However, fog and edge nodes have limited computing resources, hindering the execution of common artificial intelligence techniques. New techniques are being defined for reducing the rigidity of these mechanisms, allowing their distribution along the cloud-to-thing infrastructure.[4]

Finally, the cloud-to-thing continuum should not only manage the distribution of resources for a single IoT application but also support the integration of various applications, even from different domains. However, the different data semantics, requirements, and qualities of the different consumers complicate the interoperability and turn the collaboration across multiple vertical systems into a challenging task. New architectures are being defined to manage the infrastructure resources and share services between different systems.[5] Nevertheless, an elastic management of the information flows in the cloud-to-thing continuum is needed. This elasticity must consider the limited infrastructure resources, the interactions between applications, the data that can be shared between them, and the quality required.

REFERENCES

1. T. Lynn, J. G. Mooney, B. Lee, and P. T. Endo, *The Cloud-to-Thing Continuum: Opportunities and Challenges in Cloud, Fog, and Edge Computing*. Berlin, Germany: Springer, 2020, doi: 10.1007/978-3-030-41110-7.
2. J. L. Herrera, J. Galán-Jiménez, J. Berrocal, and J. M. Murillo, "Optimizing the response time in SDN-fog environments for time-strict IoT applications," *IEEE Internet Things J.*, vol. 8, no. 23, pp. 17172–17185, Dec. 2021, doi: 10.1109/JIOT.2021.3077992.
3. M. Villari, M. Fazio, S. Dustdar, O. Rana, D. N. Jha, and R. Ranjan, "Osmosis: The osmotic computing platform for microelements in the cloud, edge, and Internet of Things," *Computer*, vol. 52, no. 8, pp. 14–26, 2019, doi: 10.1109/MC.2018.2888767.
4. D. R. Torres, C. Martín, B. Rubio, M. Díaz, "An open source framework based on Kafka-ML for DDNN inference over the cloud-to-things continuum," *J. Syst. Archit.*, vol. 118, 2021, Art. no. 102214, doi: 10.1016/j.sysarc.2021.102214.
5. N. Chen, Y. Yang, J. Li, and T. Zhang, "A fog-based service enablement architecture for cross-domain IoT applications," in *Proc. IEEE Fog World Congr.*, 2017, pp. 1–6, doi: 10.1109/FWC.2017.8368533.

instants (t). At each instant, an action is triggered that has an impact on the infrastructure. At the top left of each instant, a small legend appears with a vertical bar. This legend indicates for each analytic (each colored horizontal bar) if it is inserted/removed from the system (+/−) and, depending on its height in the vertical bar, the accuracy that it will provide. Note that we only represent the accuracy dimension to improve readability. At *t1*, the orange analytic is requested, requiring low accuracy. The orchestrator, therefore, assigns it to only a part of the infrastructure. At *t2*, two new analytics (yellow and blue)

are inserted, with low and high accuracy, respectively; the orchestrator deploys the blue analytic throughout the whole infrastructure while the yellow one is deployed only in the less loaded part. At *t3,* the orange analytics ends and, because there are enough resources, the orchestrator upgrades the yellow one to a medium accuracy. At *t4,* the purple analytic is inserted and the orchestrator again assigns it the lowest loaded part of the infrastructure. At *t5*, the blue analytic ends and a new one with a low accuracy is deployed. Finally, at *t6*, the yellow analytics ends and the rest are reorganized to better distribute the resource consumption.

Finally, to show how the elasticity may affect the results obtained, Figure 4 shows two different analytics for the smart city case study. For each analytic, the data quality parameters (freshness and accuracy) are detailed. The results are the citizens' locations, which are shown with heatmaps for an easier decision-making process. The first analytic is focused on quickly identifying crowded areas for the healthcare system. Therefore, it has been requested with low accuracy and high freshness. Consequently, the orchestrator has only ordered the execution of the analytics to a part of the infrastructure and fewer citizens are involved but with real-time data. On the other hand, the second analytic would be interesting for the city council to study future investments. In this case, it has been requested with high accuracy and low freshness. Therefore, the orchestrator assigns the entire infrastructure to execute them and, therefore, gets the information from all the available citizens in the area.

CONCLUSION

The deployment of a swarm of sensors and Internet-connected devices is giving rise to a new data-driven economy. We envision an environment in which both the data to be consumed and the existing resources must be efficiently managed. In this article, we propose elastic data analytics, and we define the different dimensions affected by them. We also outline a framework to manage the dynamicity of these analytics in real time depending on the context.

Nevertheless, there are still some open challenges to be addressed to fully implement elastic data analytics. First, different policies should be generated for changing the management of elasticity depending also on the cost. For instance, trying to maximize the available computing resources, or minimize the costs to foster a sustainable economy. Second, more dimensions should be analyzed to identify how the elasticity affects them, such as the cost of the data. Currently, we work on including artificial intelligence to define self-managed analytics.

ACKNOWLEDGMENTS

This work was supported in part by the DIN2020-011586 Grant, funded by MCIN/AEI/10.13039/501100011033 and the European Union "NextGenerationEU/PRTR," in part by the Projects RTI2018-094591-B-I00, RTI2018-101204-B-C21 (MCIU/AEI/FEDER, UE) and 0499_4IE_PLUS_4_E (Interreg V-A España-Portugal 2014-2020), in part by the Department of Economy, Science and Digital Agenda of the Government of Extremadura under Grants GR18112, IB18030 in part by Junta de Andalucía under projects APOLO (US-1264651) and EKIPMENT-PLUS (P18-FR-2895), in part by the European Regional Development Fund, and in part by the RCIS Network under Grant TIN2016-81978-REDT.

REFERENCES

1. A. Opher, A. Chou, A. Onda, and K. Sounderrajan, *The Rise of the Data Economy: Driving Value Through Internet of Things Data Monetization*. New York, NY, USA: IBM Corporation: Somers, 2016.
2. F. Xhafa, B. Kilic, and P. Krause, "Evaluation of IoT stream processing at edge computing layer for semantic data enrichment," *Future Gener. Comput. Syst.*, vol. 105, pp. 730–736, 2020, doi: 10.1016/j.future.2019.12.031.
3. J. Medd, "Data quality fundamentals for data and analytics technical professionals," Gartner, Aug. 17, 2020. [Online]. Available: https://www.gartner.com/en/documents/3989182/data-quality-fundamentals-for-data-and-analytics-technic
4. A. Brogi, S. Forti, and A. Ibrahim, "Deploying fog applications: How much does it cost, by the way?," *Small*, vol. 1, no. 2, 2018, Art. no. 20, doi: 10.5220/0006676100680077.
5. J. M. García, P. Fernandez, A. Ruiz-Cortés, S. Dustdar, and M. Toro, "Edge and cloud pricing for the sharing economy," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 78–84, Mar./Apr. 2017, doi: 10.1109/MIC.2017.24.
6. S. Dustdar, P. Fernandez, J. M. García, and A. Ruiz-Cortés, "Elastic smart contracts in blockchains," *IEEE CAA J. Autom. Sinica*, vol. 8, no. 12, pp. 1901–1912, Dec. 2021, doi: 10.1109/JAS.2021.1004222.

SERGIO LASO is currently with Global Process and Product Improvement, Cáceres, Spain, where he is working toward the industrial Ph.D. degree in computer science. Contact him at slasom@unex.es.

JAVIER BERROCAL is currently an associate professor with the Department of Informatics and Telematics System Engineering, University of Extremadura, Badajoz, Spain. Contact him at jberolm@unex.es.

PABLO FERNANDEZ is currently an associate professor with the Applied Software Engineering Group, Smart Computer Systems Research and Engineering Lab, Research Institute of Computer Engineering, University of Sevilla, Seville, Spain. Contact him at pablofm@us.es.

JOSÉ MARÍA GARCÍA is currently an associate professor with the Applied Software Engineering Group, Smart Computer Systems Research and Engineering Lab, Research Institute of Computer Engineering, University of Sevilla, Seville, Spain. Contact him at josemgarcia@us.es.

JOSE GARCIA-ALONSO is currently an associate professor with the Department of Informatics and Telematics System Engineering, University of Extremadura, Badajoz, Spain. Contact him at jgaralo@unex.es.

JUAN M. MURILLO is currently a full professor of Software Engineering with the University of Extremadura, Badajoz, Spain. Contact him at juanmamu@unex.es.

ANTONIO RUIZ-CORTÉS is currently a full professor and head of the Applied Software Engineering Group (ISA) with the Smart Computer Systems Research and Engineering Lab, Research Institute of Computer Engineering, University of Sevilla, Seville, Spain. Contact him at aruiz@us.es.

SCHAHRAM DUSTDAR is currently a full professor of computer science (informatics) with the Distributed Systems Group, TU Wien, Vienna, Austria. Contact him at dustdar@dsg.tuwien.ac.at.

"Si se cree y se trabaja, se puede"
"If you believe and work, you can"

Diego Pablo Simeone

Chapter 4

Conclusions

This chapter closes the thesis. Section 4.1 discusses the results with respect to the goals proposed in this thesis. Section 4.2 presents the general conclusions on the results obtained throughout the PhD. In Section 4.3 we discuss future lines of research. Finally, in Section 4.4, some personal reflections on the context of this thesis are given.

Contents

4.1 Discussion

In this section, we review how our relevant proposals have contributed to the state of the art, taking into account the proposed goals.

SG1: *Providing tools and processes that speed up the implementation of emerging architectural styles such as Edge, Fog, and Mist Computing.* This goal was focused on covering the implementation or Code phase according to the DevOps lifecycle of IoT applications in the Cloud-to-thing continuum. Three proposals were presented, Human Microservices, SOLID in Smartphones, and Distributed API Composer. The first two respectively are focused on supporting the Mist layer. Human Microservices [13] provided a framework for the design and implementation of APIs for devices belonging to these layers to provide their own microservices; and SOLID in smartphones

[32] was focused on providing a model for the storage of personal data, having control of access to the data by external entities. The last of the proposals called Distributed API Composer (API) [35] aimed to cover the Cloud, Fog, and Edge layers, completing the whole spectrum of the Cloud continuum [92]. This component has the goal to perform the whole process of coordinating invocations, obtaining the data, and aggregating it from the APIs distributed in the end devices. The proposal provides a model and a process for its design and implementation. It also has the ability to be deployed in any of the mentioned layers, which allows developers to decide where to deploy them according to their requirements. During the research in this phase, we realized that there were proposals for the same purpose [64, 93, 94] but they are ad-hoc solutions for specific examples, not allowing us to generate them in other projects with different requirements. One of the advantages of the presented proposals is the use of OpenAPI [34] a standard for the design and development of APIs that reduces the impact on developers and companies because they do not have to learn a new language. This allows the learning curve for developers to be lower. And SOLID [30], is another well-known initiative to provide alternative models for decentralized data storage.

SG2: *Providing tools and processes that facilitate the evaluation of the QoS of IoT applications that apply architectural styles making use of paradigms such as Edge, Fog, and Mist Computing.* This goal was focused on covering the evaluation or Test phase of the DevOps methodology. In this phase, we presented Perses [14], a framework that allows to deploy of virtual scenarios with heterogeneous devices to evaluate the QoS of distributed applications, i.e. applications deployed in the Cloud continuum. It also allows to integrate into a software development process, such as DevOps, by automating the tasks of creating and deploying the virtual scenario, launching tests, collecting results, etc. During the research in this phase, we realized that such a tool was needed when trying to evaluate and compare a distributed architecture versus a traditional cloud architecture for another research work [95]. However, during the literature review, as in the previous phase, we realized that there were proposals that focused, for example, on evaluating applications but developed with a specific framework [96, 97] or tools that allowed automatic testing but in scenarios without heterogeneity [98]. Commercial tools that allow testing with real or virtual devices but are focused on user interface testing, i.e. they do not evaluate QoS [57, 99]. Our proposal currently also has some limitations: Currently only mobile device virtualization is supported, but we are working on extending the range of end devices (microcontrollers, sensors, etc). Regarding network emulation, currently, we can define the type of connectivity (Wi-Fi or ISP) that the mobile devices have but we do not take into account the network topology (groups of devices connected to different networks) or mobility, which is an important aspect to consider [100].

SG3: *Provide mechanisms so that an application can dynamically change the architecture topology to adapt to the context needs or resources required at the time.* The last goal was focused on covering the deployment or Operate phase according to DevOps. The contributions that have been presented were aimed at the implementation of applications so that the deployment of the Cloud continuum infrastructure can be dynamically adapted to the needs of the application context (e.g. communication, transmission, and data quality requirements, etc.). On the one hand, we presented the Oppnets [42], a solution based on an opportunistic network algorithm that enables the deployment of the Cloud-to-Thing communication technology solutions in isolated areas where connectivity is limited. The solution consisted of a DTN-based system applied to both elderly healthcare and industrial activities in rural areas. It was based on cooperation between nodes by dynamically adapting the communication according to the type of data and the receiver's interests. There are proposals to use DTN for deployment in isolated and specific areas where Internet connection is not possible [101, 102, 103]. However, our proposal improves these solutions by dynamically adapting the communication according to the interests of the nodes. On the other hand, our proposal Elastic Data Analytics [43] is more focused on favoring data sharing or analysis flow, in those applications that use the same type of data. Using Elastic Data Analytics, applications can launch analytics by defining their quality requirements, whose behavior can be dynamically modified according to these defined quality requirements (freshness and accuracy) and the available resources of the cloud-to-thing infrastructure. For this purpose, we provide a framework with an orchestrator that manages the elasticity produced by the variation of the analysis quality requirements. This orchestrator evaluates the state of the infrastructure, the other ongoing analytics, and the QoS required by each scan. As a result, it reconfigures all analytics to maximize the quality provided by each analytic without exhausting the resources of the cloud-to-thing infrastructure. Currently, there are other proposals on elasticity but applied in other areas such as Blockchain [104] or Service Level Objectives (SLOs) [105].

4.2 Research Conclusions

In recent years, the number of IoT devices has increased considerably. Their computational capabilities have also increased to get more information from the environment where there are a wide variety of IoT applications that use microservices deployed in the Cloud to adapt the physical environment to users' needs and preferences. This wide variety of applications usually has strict QoS and QoE requirements to provide end users with a good experience and be better than the competition.

However, QoS and QoE can be affected by the demand for network traffic of the infrastructure from all Internet-connected devices [106], as well as higher operating costs due to increased scalability in their cloud environments.

In order to try to meet these requirements, researchers have proposed new architectural paradigms such as Fog Computing, Edge Computing, and Mist Computing. All these proposals are within the so-called Cloud-to-thing continuum, which are all infrastructures that extend from the cloud to the end devices closest to the users. These paradigms aim to distribute computing and storage services in different layers. Among other benefits, these paradigms reduce network load and dependency on the cloud environment, improving response time and providing a better user experience.

Currently, some tools or frameworks allow reducing the effort and development time for IoT applications based on Cloud Computing. However, this does not happen with these distributed computing paradigms so that they can be applied in a real development environment. This implies that companies must devote more effort to the development of these systems.

Therefore, this thesis has focused on investigating and contributing to the development of these emerging architectural paradigms such as Fog, Edge, and Mist along the Cloud-to-thing continuum. To this end, different proposals have been presented to help facilitate the development of IoT applications based on these paradigms focused on implementation, evaluation, and deployment activities within software development methodologies such as DevOps. This integration with development methodologies will facilitate the introduction of these architectural paradigms in developing IoT applications in enterprises.

4.3 Future Work

This section describes some future lines that this thesis leaves open. Each of them is listed below.

- *Human Microservices and SOLID:* Both proposals currently have the same drawbacks, which is the limitation of supported devices. Therefore, the next steps for these proposals would be to increase support for more end devices for both API deployment and secure storage on them. Finally, with respect to Human Microservices, end devices currently provide and consume information through a single communication node. However, if that node is located far from the point where data is presented or consumed, the QoS may be affected. Therefore, we are

working on a distribution of communication nodes in different layers (edge, fog, etc.) where devices can be connected to several nodes and some routing mechanisms can be implemented to achieve a higher QoS.

- *Distributed API Composer:* We are working, first, on improving and evolving the DAC functions, and second, on conducting experiments to compare with traditional systems and to know the perception of analysts and developers regarding the use of this system. The whole proposal and its improvements are being reflected in the preparation and writing of an article to be submitted to a journal.

- *Perses:* We plan to enable the emulation of new devices (IoT devices). This will increase the heterogeneity of the scenarios. It is also planned to provide different network topologies with Kathara [59] in the same scenario (distribution of devices connected to different nodes simulating the connection to different WIFI or LTE connections) and the mobility of the simulated nodes. Finally, we have also planned the automatic generation of a cost analysis that allows developers to visualize the expenditures that will be generated before the deployment of the virtual scenario, being able to adjust it to the budget they have defined.

- *OPPNets:* The next step considered for this proposal is to provide the proposed algorithm with intelligent behavior employing a machine learning model, also taking into account the trajectory in the movement of the nodes.

- *Elastic Data Analytics:* This proposal is the most current and the most challenging to achieve. We are currently working on the development of a framework to generate scenarios (supported by Perses) with different conditions (number of devices involved, data granularity, data quality requirements, etc.), to evaluate different parameters (QoS, costs, etc.) between an infrastructure with elastic data analytics and a traditional cloud and determine at which moments of an IoT application it is more beneficial to use one or the other depending on the context conditions. We have also considered generating different policies to change the management of elasticity depending also on the cost. For example, try to maximize the available computing resources, or minimize costs to promote a sustainable economy. Finally, we have also raised the inclusion of Artificial Intelligence to define self-managed analytics.

4.4 Personal Reflection

Finally, in this section, I present some personal reflections on the work of this thesis and thoughts on the objectives and possible future implications.

The present thesis is the culmination of a whole body of work that began with my master's studies. Although it is not the end either, since with the future works described above there is still a long way to go in this field of research.

The aim of this thesis is to show the potential of distributed computing and the benefits it can bring. Especially for the coming years, where it is estimated that the number of devices connected to the Internet will increase considerably [2]. This implies that in order to continue maintaining infrastructure with the quality requirements as the current one, it will be necessary to continue increasing resources, which implies higher costs but above all higher energy consumption[107].

We are currently facing different problems related to the extraction of fossil wastes caused by climate change, scarcity of resources, wars, etc. being the main source of energy generation in the world [108]. This implies that important efforts are being made in new ways of energy generation but also in how to manage it as efficiently as possible [109]. In my opinion, I believe that these new distributed computing technologies can be beneficial for the optimization of resource consumption and therefore lower energy consumption [107].

In 2020 in my master's thesis, I said *"I believe that in the near future, we will have to manage all resources as efficiently as possible, so that in addition to obtaining better results in terms of QoS, lower costs, etc., it will have the least possible impact on the environment"*. Although only a short time has passed so far, I believe that this moment has already arrived.

Appendix A

Supporting Publications

Contents

A.1 Providing Support to IoT Devices Deployed in Disconnected Rural Environment

Authors: *Sergio Laso, Daniel Flores-Martín, Juan Luis Herrera, Carlos Canal, Juan M. Murillo and Javier Berrocal.*

Publication type: *Conference paper (long paper).*

Workshop: *International Workshop on Gerontechnology, Cáceres, Spain. 2019.*

Available online: *10.1007/978-3-030-41494-8_14.*

Providing Support to IoT Devices Deployed in Disconnected Rural Environment

Sergio Laso[1][0000−0001−8911−9371], Daniel Flores-Martín[1][0000−0002−2554−2194], Juan Luis Herrera[1], Carlos Canal[2][0000−0002−8002−0372], Juan Manuel Murillo[1][0000−0003−4961−4030], and Javier Berrocal[1][0000−0002−1007−2134]

[1] University of Extremadura, Quercus Software Engineering Group, Avda. de la Universidad s/n, Cáceres, 10003, Spain
{slasom, dfloresm, jlherrerag, juanmamu,jberolm}@unex.es
[2] University of Málaga, Spain
canal@lcc.uma.es

Abstract. The increase in the end and near-to-the-end devices capabilities has led to the development of paradigms such as the Internet of Things, Fog Computing and Edge Computing. These devices require an internet connection for sending the sensed or processed data, and for getting specific requests. Many of these devices are intended to make people's live easier and they can also be used to better monitor people's health. One of the problems of these devices is that they require internet connection and in most rural areas there is a poor internet infrastructure, making their use almost unfeasible. This paper presents a tool for generating applications with a mobile-centric architectural style that takes advantage of the capabilities of the end devices for processing and storing the sensed data. This style allows devices to process the information requests asynchronously, reducing the Internet requirements and allowing the use of these technologies in areas where the Internet connection is poor and intermittent.

Keywords: Internet of Things · Server-Centric · Health · Mobile-Centric.

1 Introduction

End and near-to-the-end devices capabilities have increased enormously. During the last years, computational and storage capabilities have been multiplied with the aim of obtaining more information from the environment and performing increasingly complex operations. This increase has favored the development of paradigms such as the Internet of Things (IoT) [14], where they have a greater integration in the Internet.

IoT and end devices are designed to be connected to the Internet all the time. Usually, they are integrated into the Internet following a server-centric architecture [7], in which the final devices act as simple clients obtaining information through their sensors and interfaces, and sending it to

the back-end. Currently, with the capabilities that many devices have, other architectures, such as mobile-centric, p2p, and so on, can be followed with the aim of limiting the use of Internet [11], being useful in many cases.

For instance, in rural environments, it is common to find areas with poor and limited access to 3G / 4G communication networks [6]. These connections are often of poor quality and intermittent. Therefore, the use of server-centric architectural designs can provide a bad user experience. If we have a set of IoT devices collecting information and sending it back to the server or cloud, it is more than likely that in this environment some of the information will be lost generating a bad user experience and a undesirable behaviour that can jeopardize the success of almost any application.

To cover these shortcomings, the use of mobile-centric architectural designs have been proposed by some researchers [10]. These designs focus on making use of the devices' capabilities for computing and storing the gathered information. The stored information is provided by the device itself to any other entity to be consumed in an asynchronous manner, limiting the need for constant Internet connection. Nevertheless, the lack of tools assisting developers in the application of these designs, hinders and limits the use of these architectural styles [8].

In this paper, we present a tool that helps developers to implement APIs following a mobile-centric architectural design. This tool is focused on Android devices, since currently they usually are the devices with enough computing capabilities for computing, storing and providing the sensed information. To assist in the development of these APIs, this tool requires an OpenAPI specification [4] of the designed application, and it generates the scaffolding of the application and the whole communication interface. In this way, the effort to develop this type of architectures is reduced.

This type of design is very useful in rural environments that have an intermittent connection since the gathered data is no longer sent to cloud environment, reducing the loss of information.

The rest of the paper is structured as follows. Section 2 describes the motivations for this work and describes briefly two tools used in this work. Section 3 explains the tool for generating mobile-centric applications. Section 4 shows several related works. Finally, Section 5 details the conclusions.

2 Motivation

Elderly people are increasingly accessing new technologies, and specially smartphone and IoT devices, to live independently and be in contact with their relatives [17]. With the massive deployment of IoT devices and the mHealth paradigm [16], they also are starting to use their smartphones to monitor and control their health status [13]. Usually, they have different IoT devices, such as tensiometers, oximeters, smartbands, or the smartphone

itself, to gather information about their health like heart rate, oxygen levels, etc. Once this information is gathered, it is sent to the smartphone that acts as a gateway with the cloud environment, where it is processed and stored.

This designs leads to some advantages, since the information is stored in the cloud and can be remotely evaluated by relatives and healthcare professionals. Nevertheless, it also has some drawbacks. For instance, one important problem is raised in rural areas, where the Internet coverage is poor and intermittent. In these situations, these IoT devices and the smartphone cannot send the gathered data to a server or cloud, being lost in some situations. This may also lead to some health alarms to be raised because, for instance, the heart rate value has not being received by the cloud during a couple of days, generating unnecessary worries to relatives and healthcare professionals.

To better show this problem, we are going to describe a scenario. This scenario will be used as running example in the rest of the paper to show the benefits of this work.

Pedro is 74 years old and lives in Casares de las Hurdes in the province of Cáceres. His sons purchased him a device for monitoring different health dimensions (heart rate, blood pressure, oximeter, ...) and a smartphone with an application to receive data from the device and to get the GPS position of the smartphone. When the application receives the data, it connects to a server and sends all the sensed data so that it can be reviewed by some healthcare professionals. Casares de las Hurdes is a village with poor accessibility and communications networks, therefore, the coverage is quite bad being null or intermittent in many locations. Pedro's sons indicated him some places in his home where there is some coverage, but they are not the usual place where he stays. So, Pedro usually forgets to move to another location to take the measurements and sometime they are lost when they unsuccessfully sent to the server.

Figure 1 shows a scheme of the current architecture.

As can be seen, there are several problems apart from the obvious one that is the bad Internet connection, starting with the task of being aware of the coverage of the device and having to move to places where there is coverage. Another problem and it is quite serious, is the loss of data that occurs when it cannot be sent correctly to the server.

There are different architectural styles and designs to try to solve this problem, in this paper we will use a mobile-centric architecture. This architecture is focused on taking advantage of mobile devices' capabilities in order to compute, store and provide services or functionalities that are invoked or consumed by a third entity, that is, as if we had our own server in the smartphone itself. In this architecture, the interactions with the mobile devices could be asynchronous and its connection to the Internet could be limited. Therefore, if the application described above is updated with this

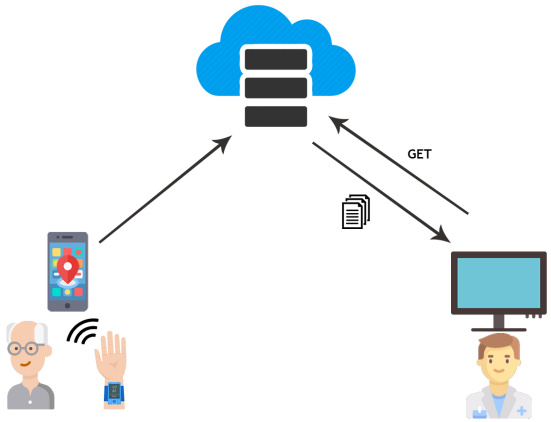


Fig. 1. Server-Centric architecture.

style, the values captured by the IoT devices are processed and stored on the smartphone and the request to obtain the data are sent and processed directly by it.

The problem that exists with these alternative architectures is that there are not tools supporting them and facilitating their development [8]. Most of the current development tools are focused on providing support to the server-centric architectural style, since it is the most used one. For example, through an OpenAPI specification [4], developers can generate part of the source code of both the back-end and the client of the application. However, these same tools are not yet available for these alternative architectures. This is a great disadvantage for developers using such designs, since it entails an extra effort for its development.

To facilitate the development of these designs, in Section 3, we shall present a tool that assist developers in the implementation of applications following a mobile-centric architecture. This development tool is focused on using OpenAPI for specifying the app's behaviour and Firebase Cloud Messagging (FCM) [1] for supporting for communication and interactions with it.

2.1 OpenAPI

Currently, there are many tools that facilitate the specification and deployment of applications. One of them is OpenAPI, which was created to standardize the description of RESTful APIs [12]. With it one can specify APIs using its standard, and through code generators such as OpenAPI Generator [3], get the skeleton of the application from both the server and the client. However, these facilities are not available for other architectural styles, such

as mobile-centric. The current capabilities of mobile devices allow them to also be mobile cloud environment having different APIs deployed on them processing and storing data and serving the information to third entities. This is especially useful in rural environment that have an intermittent connection.

2.2 Firebase

One of the major problems of mobile-centric applications, especially those deployed in closed or not completely open operating systems, is the communication between consumers and the providers, as it is limited by privacy and security policies. To solve this problem, services offered by Google, Firebase Cloud Messaging (FCM) [1], can been used to interact with the devices through Push notifications.

There are other similar messaging protocols such as MQTT [2] or RabbitMQ [5]. These can also be used is this design. In fact, MQTT is already included, but it has not been specified because the focus of this paper is in the general workflow. The advantage of FCM is that its integration is easier in Android-based applications.

FCM can send messages through the Firebase Admin SDK or through HTTP or XMPP protocols. These messages can be received through push notifications to iOS, Android or Web (JavaScript) devices. This system allows one to send a message directly to a single device or a set of them through '*topics*'. FCM has the property of storing messages in a queue for a while until the device has Internet connection. It is an asynchronous communication method, therefore, it can be used in places with an intermittent internet connection.

3 Generation of Mobile-Centric Applications

In this section, the process to generate mobile-centric applications and APIs for Android devices is explained. As can be seen in Figure 2, in the first step *"API Definition"* the API features are defined, this task will be done through the OAI (OpenAPI) Specification following the same notation as if it were to be developed and deployed on a cloud environment. The API specification will describe the different resources and endpoints that it will have available to be invoked.

In the second step *"Generate base code"* generates the skeleton code of the API. Currently, there are no tools to facilitate this task for this type of architecture as discussed above. In order to offer this functionality, the OpenAPI Source Code Generator has been extended so that the skeleton of services or functionalities can also be generated to be deployed on Android-based mobile devices. Thus, these services can be consumed by third entities

as if they were deployed in an cloud environment. To that end, the source code to simulate the communication logic of an APIRest is also generated, reducing the effort required by developers. This communication logic has being simulated using Firebase Cloud Messaging, Thus, first, any third party can invoke the offered services and, second, the services can also be consumed asynchronously, providing support to the consumption of this information in environment with an intermittent Internet connection.

Therefore, having the application generated in the second step, in the third step "*Deployment*", developers only have to finalize the implementation of each feature and configure the Firebase Cloud Messaging services to be able to deploy it.

Finally in the fourth step "*Requests*" with the application deployed, we invoke the resources defined through any service invocation tool.

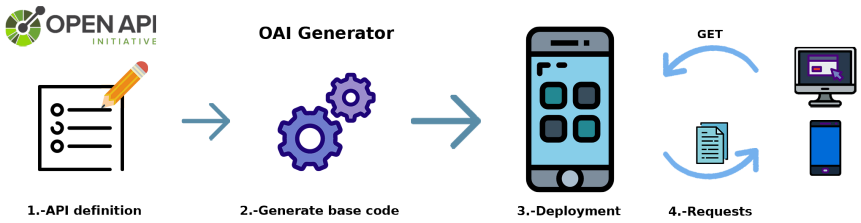


Fig. 2. Steps to Generate Mobile-Centric Application.

Following the described scenario, with this new proposal, as can be seen in Figure 3, all the data collected from both the devices and the smartphone itself will be stored on the smartphone. To access this data, relatives and healthcare professionals will only have to invoke the necessary smartphone's services through a client application to obtain the sensed information. The services invocation is done asynchronously, so if Pedro does not have Internet coverage when the request is made, the request (asynchronous) will be sent and it will be stored in the Firebase's queue waiting for Pedro to have connection. Once he has Internet connection, it will reach Pedro's phone and the requested data will be sent to the doctor's app.

Below an extract of the OpenAPI specification for the *Health Monitoring* application of the described scenario can be seen. As discussed earlier, this specification uses the same notation that could be used to implement it in a cloud environment. Obviously, the main difference is that the services have to be designed to offer and provide the specific information stored in a mobile device.

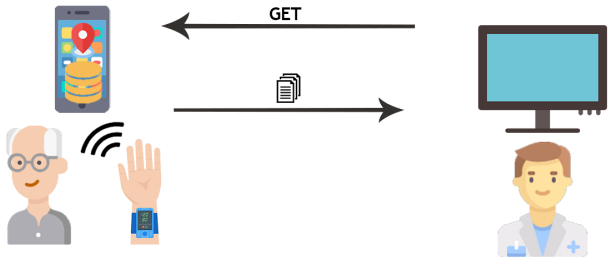


Fig. 3. Mobile-Centric Architecture.

```yaml
openapi: 3.0.1
info:
    title: Health Monitorization
    description: This application monitors the health constants
    of users to keep track of their health.
    version: '1.0'
    termsOfService: 'https://healtmonitorization.spilab.es/terms'
    contact:
    name: Health Monitorization
    url: 'https://spilab.es'
    email: info@spilab.es
paths:
  /health:
    get:
      tags:
        - Health
      summary: Gets a health constants.
      description: Get several health constants from a patient.
      operationId: getHealthConstants
      parameters:
        - name: patientID
          in: query
          required: true
          schema:
            type: number
            format: double
      responses:
        '200':
          description: Successful response
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/HealthConstants'
```

```
34          '404':
35            $ref: '#/components/responses/404NotFound'
```

With the specification shown above, the OpenAPI Source Code Generator performs the conversion and generates a project with the API's skeleton. The generation of the source code is done as follows: the title of the specification "*Health Monitorization*" is used as the identifier and name of the application; each tag, in this case only *Health*, groups all the endpoints it contains (in this case only *getHealthConstans*); then, each tag is transformed into a resource or Android service (Class) that exposes those endpoints; finally, for each endpoint a specific method is created for developers to implement the concrete behaviour.

Thus the HealthResource class that is shown in listing 3, groups all the endpoints belonging to the *Health* tag. This class is formed by a first method called "*executeMethod*", which contains a switch to redirect the request to each specific endpoint. In this case, the OAI generator only generated one endpoint *getHealthConstans* (as was defined in the specification). This is the only method that have to be implemented by the developer, in addition to the Firebase configuration. Finally, the generated API also contains an auxiliary method to return the request results.

Listing 1.1. Code generated by new version of OpenAPI Generator.

```
public class HealthResource {
    ...
    public void executeMethod(HealthResponse response){
        healthResponse=response;
      switch (response.getMethod()){
         case "getHealthConstants":
          getHealthConstants(response.getParams().getuserID());
         break;
      }
    }
    public HealthConstants getHealthConstants (Double userID){
        //TODO
        return null;
    }
    ...
}
```

Firebase Cloud Messaging (FCM) cannot be directly used to implement the same communication logic as an APIRest, the communication between consumers and providers have to be asynchronous. For environment with an intermittent connection, an asynchronous communication is an advantage. FCM saves the requests in a queue for a while, so if the from which

we want to get information do not have Internet connection, there will be no problem, requests will reach devices once they have it. In addition, the integration with this communication method is generated automatically by the tool when the mobile application is generated. An implementation for synchronous communication could also be generated, but it is out of focus of the paper.

Having the application deployed, it is possible to invoke the services using any REST client. The following listing shows the structure to invoke the service from which one can obtain the vital signs of a patient using the defined API. The structure general of the request is the same for every application, the *Url* and the type of *Method* don't change. In the *Header*, it is necessary to put the Authorization Key that Firebase provides in the project configuration details. Finally, the structure of the *Body* is also the same for every application, obviously, it has to be adapted to its corresponding methods, parameters, etc .

- Url: https://fcm.googleapis.com/fcm/send
- Method: POST
- Headers:
  - Content-Type: application/json
  - Authorization: key=<Key obtained from Firebase configuration>
- Body:

```
{
"to": "tokenId or topic",
        "data":{
                "resource": "Health", --Endpoint
                "method": "getHealthConstants", --Method (operationId) of Endpoint
                "sender": "158.49.112.1/result", --Address to send reply
                "params":{ --Parameters of the method
                        "patientID": 2458372
                }
        }
}
```

The tools presented in this paper, facilitate the development of applications following a mobile-centric style, promoting their use and also covering problems, such as those presented in disconnected environments.

4 Related Work

There are few commercial tools that support other architectural styles, such as mobile-centric, peer-to-peer or other fog-based styles. However, at the research level, more and more proposals are presented fostering the use of the available specification and resources to generate the source code and facilitate the development process of any application following these alternative styles.

As we have seen, the popular OAI specification is very useful for defining applications, including generating new styles by creating or expanding its code generators. In [15], the authors propose WoTDL2API. WoTDL2API is a tool to generate a RESTful API with a web interface from an OAI specification. The generated API has the goal of generating and deploying a web-based RESTful interface obtained from the WoTDL descriptions to provide interoperability between different IoT devices that communicate with different communication protocols (Zigbee, Bluetooth, RFID, ...).

However, this solution doesn't cover our connection problem in rural environment. Although it is possible to deploy the API on a local network not relying on 3G / 4G connections, it requires Internet coverage to receive request from third entities that are not connected to the same local network.

On the other hand, in [9] they propose a system to download generalized applications with strict delay requirements as stateless functions on edge nodes with available computing capabilities. The execution of functions passes through edge nodes with sending capabilities, which are ideally located as close as possible to end users and is highly scalable. They have shown that this distributed architecture works better than a centralized approach but still does not cover the problem regarding the bad or intermittent connection that occurs in most rural areas.

With these works, we see that tools are needed to support the deployment of applications in environments with intermittent connectivity and following a mobile-centric style.

5 Conclusions

The increase of the computing capabilities of end and neart-to-the-end devices opens the possibility to the application of architectural styles that until now could only be used in very specific scenarios. Thanks to these emerging architectural styles, one can solve some problems that exist today. This paper exposes the lack of tools helping in the development of these alternative architectures.

In this paper, we have presented a tool that automates the creation of the skeleton of an application following a mobile-centric style, thus promoting its use and advantages such as, for example, in disconnected rural environment.

Currently we work, first, on analyzing the impact of using these tools and if they would increase the application of these alternative architectural styles. In addition, we are also working on providing support to other emerging architectures such as Fog Computing.

6 Acknowledgment

This work was supported by the 4IE+ project (0499_4IE_PLUS_4_E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the Spanish Ministry of Science, Innovation and Universities (MCIU) (RTI2018-094591-B-I00) (MCIU/AEI/FEDER, UE), by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), and by the European Regional Development Fund.

References

1. Firebase Cloud Messaging, https://firebase.google.com/docs/cloud-messaging
2. Mqtt, http://mqtt.org/
3. Openapi Generator, https://github.com/OpenAPITools/openapi-generator
4. The OpenAPI Specification Repository. Contribute to OAI/OpenAPI-Specification development by creating an account on GitHub, https://github.com/OAI/OpenAPI-Specification
5. Rabbitmq, https://www.rabbitmq.com/
6. Bahia, K.: Gsma - state of mobile internet connectivity 2018 (2019), https://www.gsma.com/mobilefordevelopment/wp-content/uploads/2018/09/State-of-Mobile-Internet-Connectivity-2018.pdf
7. Barbera, M.V., Kosta, S., Mei, A., Perta, V.C., Stefa, J.: Mobile offloading in the wild: Findings and lessons learned through a real-life experiment with a new cloud-aware system. In: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications. pp. 2355–2363 (April 2014). https://doi.org/10.1109/INFOCOM.2014.6848180
8. Berrocal, J., García-Alonso, J., Murillo, J.M.: Architectures server-centric vs mobile-centric for developing wot applications. In: Bakaev, M., Frasincar, F., Ko, I. (eds.) Web Engineering - 19th International Conference, ICWE 2019, Daejeon, South Korea, June 11-14, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11496, pp. 578–581. Springer (2019). https://doi.org/10.1007/978-3-030-19274-7_48, https://doi.org/10.1007/978-3-030-19274-7_48
9. Cicconetti, C., Conti, M., Passarella, A.: Low-latency distributed computation offloading for pervasive environments. In: Pervasive Computing and Communications (PerCom), 2019 IEEE International Conference on. IEEE (2019)
10. Guillén, J., Miranda, J., Berrocal, J., Garcia-Alonso, J., Murillo, J.M., Canal, C.: People as a service: A mobile-centric model for providing collective sociological profiles. IEEE Software **31**(2), 48–53 (2014)
11. Hirsch, M., Mateos, C., Zunino, A.: Augmenting computing capabilities at the edge by jointly exploiting mobile devices: A survey. Future Generation Computer Systems **88**, 644–662 (2018)
12. Masse, M.: REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. " O'Reilly Media, Inc." (2011)
13. Mena, L.J., Félix, V.G., Ochoa, A., Ostos, R., González, E., Aspuru, J., Velarde, P., Maestre, G.E.: Mobile personal health monitoring for automated classification of

electrocardiogram signals in elderly. Computational and mathematical methods in medicine **2018** (2018)
14. Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I.: Internet of things: Vision, applications and research challenges. Ad hoc networks **10**(7), 1497–1516 (2012)
15. Noura, M., Heil, S., Gaedke, M.: Webifying heterogenous internet of things devices. In: International Conference on Web Engineering. pp. 509–513. Springer (2019)
16. Organization, W.H., et al.: mhealth: new horizons for health through mobile technologies. mHealth: new horizons for health through mobile technologies. (2011)
17. Zainal, A., Abdul Razak, F.H., Ahmad, N.A.: Older people and the use of mobile phones : An interview study (12 2013). https://doi.org/10.1109/ACSAT.2013.83

A.2 Yourpantry: Food monitoring through pantry analysis using the smartphone and making use machine learning and deep learning techniques

Authors: *Enrique Moguel, José García-Alonso and Sergio Laso .*

Publication type: *Conference paper (long paper).*

Workshop: *International Workshop on Gerontechnology, Cáceres, Spain. 2019.*

Available online: *10.1007/978-3-030-41494-8_1.*

YourPantry: Food Monitoring through Pantry Analysis Using the Smartphone and Making Use Machine Learning and Deep Learning Techniques.

Enrique Moguel[1][0000−0002−4096−1282], José García-Alonso[1], and Sergio Laso[1]

University of Extremadura. Cceres, Spain.
enrique@unex.es

Abstract. Food is one of the fundamental pillars of our lives, and that of the elderly as well. In many cases, elderly people do not follow a balanced diet according to their vital needs. Nowadays, there are technological solutions that help in the daily activities of the elderly and indicate which foods they should eat. But all of these solutions require attention and manual registration. Therefore, we propose an application for smartphones that assists in feeding the elderly through the realization of a photograph of their pantry. This application uses Artificial Intelligence techniques such as Machine Learning and Deep Learning.

Keywords: Elderly · Smartphone · Food-inatake monitoring

1 Introduction

At present, the world is experiencing population-aging, a trend that is both pronounced and historically unprecedented. Over the past six decades, countries had experienced only a slight increase in the share of people aged 65 years and older, from 8% to 10%. However, in the next four decades, this group is expected to rise to 22% of the total population, a jump from 800 million to 2 billion people [1].

This trend is even more worrying in rural regions, for example, Extremadura in Spain or Alentejo in Portugal. This kind of regions have lower population density than the average, and they keep losing its young population to more socioeconomically developed regions. Therefore, they have a higher-than-average aged population while being economically disadvantaged with a fragile cultural and socioeconomic context. Additionally, due to low population density and youth migration to richer regions, elders frequently live alone [2].

Aging in these regions is not a problem by itself, at least not directly. However, as people become older, they are more prone to diseases such as cognitive impairment, diabetes, hypertension, and cardiovascular problems. Different studies indicate that a significant number of these diseases related to aging have their origin in deficient nutrition [3].

Elders are a particularly disadvantaged group with respect to nutrition, especially if compared with the rest of the population in developed countries [3]. Frequently, the elderly suffer changes in their nutritional patterns that, in some cases, can cause significant damages to their physical condition. For instance, some older people change their dietary habits, increasing their intake of greasy and salty food, or decreasing the total ingested food. This change of patterns usually leads to important-nutrient losses that directly influence the health of older people [4]. Deficient eating habits can cause serious problems in the function and cognitive status of elders, in addition to a higher rate of mortality, such as due to cardiovascular problems or anorexia episodes [5].

During the last few years, important research efforts have been devoted to various aspects related to feeding, not only for aging populations but also for the general population, addressing different nutritional disorders. One of the main objectives of the scientific community is to precisely monitor and identify nutritional patterns and ingested by elderly people. Different studies have shown that elders' nutritional patterns are a valid parameter for predicting their quality of life [6] [7] [8]. Specifically, there are approaches in this area to propose new algorithms, techniques, or systems for improving food-intake monitoring.

Food-intake monitoring is intended to acquire information, such as the number of vitamins, minerals, and other substances ingested by a person [9]. This information is then used for the identification of nutritional patterns and the detection of nutritional problems. Most works in this field focus on the state of nutrition and their relationship with different diseases, such as obesity [10], Alzheimer's disease [11], depression [12], and metabolic syndrome [13].

In general, these approaches propose to periodically carry out different surveys to the elderly in order to know their food-intake patterns, what their illnesses are, and the evolution of both. In this process, the prevalent method of diet monitoring is manually recording survey results [14] [15] [16]. However, this is a tedious process that ends with a low adherence rate, reducing its long-term effectiveness [17] [18]. To address the problem presented by manual recording, numerous technological solutions have been proposed. These solutions introduce the use of a wide range of devices, technologies, and algorithms to automatically identify different aspects of the food-intake process, like the type of food being eaten and the amount of ingested calories, or identifying the person ingesting the food.

However, food-intake monitoring solutions have additional difficulties when taking into account the circumstances of an aging population, particularly when the monitored elders live in rural environments like the ones mentioned above. The lack of infrastructure, the low average technical skills of people living on these regions, the loneliness of elders, etc. hinder the deployment and use of food-intake monitoring systems. A study of the existing proposals is necessary to know if they can be deployed in these environments.

Continuous nutrition monitoring is essential to influence positively the nutrient content of the food supply and meet the changing nutrition needs of the population. Nutrition scientists in the food industry use nutrition monitoring

data in a variety of ways that include developing nutrition communications for consumers and health professionals, guiding product development and reformulation, and applying research applications [19].

In this paper, a application making use machine learning and deep learning techniques for food-intake monitoring is presented focused on the coverage of the requirements the elderly living in rural and low populated regions. This research is part of an European project (International Institute for Research and Innovation in Aging - 0045-4IE-4-P) granted to improve the quality of life of elder people living in rural environments.

The rest of the paper is structured as follows. Section 2 presents the motivation for food-intake monitoring of aging populations in rural areas. Section 3 describes the proposed solution for food-intake monitoring making use a smartphone. And finally, Section 4 concludes this work.

2 Motivation

Monitoring food-intake in an aging population is important for detecting nutritional problems and, thus, be able to prevent related diseases. As mentioned above, one of the cheapest and less intrusive ways to perform this monitoring is through technological solutions, because methods relying on external supervisors can be complex, expensive and inaccurate, as the supervisors themselves are not involved in the eating activity and mostly rely on visual observation.

Existing monitoring systems usually focus on technical aspects related to automating the monitoring process, and improving precision in food and intake detection. Aspects like overall user impression, social acceptance, or system outputs are often considered. However, most works do no take into account the specific context in which the systems are deployed. This is particularly relevant in the case of elders living in rural regions, since several aspects have to be taken into account to make them viable for these environments.

Fig. 1. YourPantry Application.

3 Proposed solution

The details related to the design and development of the proposed application will be detailed below. The architectural aspects of the software application, the navigation diagrams, the visualization screens and the details of the development of the solution proposal will be detailed below.

3.1 Architectural Design

As can be seen in Figure 2, the architecture is composed of three layers: Presentation layer, Domain Logic layer and Data Source layer.

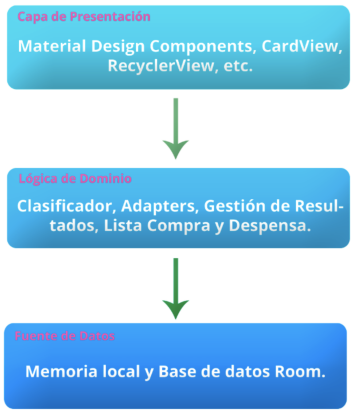


Fig. 2. Architectural design

The three-layer architecture is a type of architecture used in the large majority of systems. It is usually used in systems that implement a business model such as an online store, an application to manage certain data, etc..

The complete data management system will have a database to store that data and a user interface that will be the interface with which users interact. In addition, a part of the system will be in charge of processing the data and managing what is done with it. The three-layer architecture divides the system into three distinct parts, so that each layer only communicates with the lower one.

– **Presentation Layer**. All kinds of utilities offered in Android will be used in order to have a clean and minimalist user interface. Basically it constitutes the whole visual part of the application. Its only function is to pass the user's actions to the business layer.

- **Domain Logic**. This layer manages the application logic. It is where they say what to do with the data. It will carry out the business logic of the application, where it will communicate the user interface with the data needed for the application. This will be reflected in the Classifier (will be responsible for analyzing the image and processing before showing them), will also perform a results management (edit, delete and insert) as well as a management of the pantry and shopping list. Other elements will also appear here such as the adapters that will help us to show the data in the different lists of the application through (in this case) the RecyclerViews. It will be connected to the persistence layer in order to perform its functions.
- **Data Source**. This layer is in charge of saving the data. It will be where you manage everything related to the database and the creation, editing and deletion of data from it. The application will have local persistence, we will store all the information of the pantry and the shopping list in a database.

3.2 Navigation Diagram

Next, as can be seen in figure 3, the navigation diagram is detailed, that is, the different screens that the application will have:

- *Home screen*: it is the main screen of the application and from it the user can access the other screens through the different buttons.
- *Analysis screen*: in this screen, the results obtained by the classifier are shown, in this screen we will be able to edit the results if there was some error, before inserting it in the pantry or in the shopping list.
- *Pantry contents screen*: the contents of the pantry are shown in a list.
- *Shopping list contents screen*: this screen shows in a list, the contents of the shopping list, each product will contain a check to mark the products we have purchased and have a control.

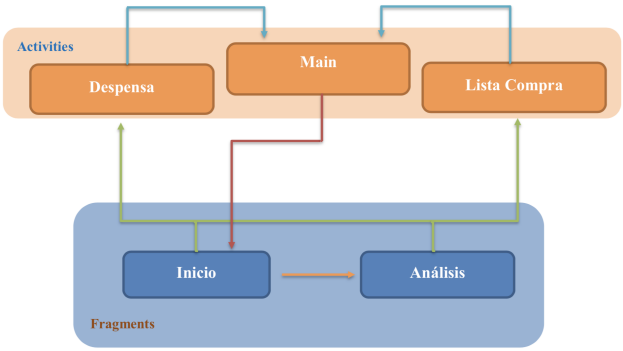


Fig. 3. Navigation diagram.

3.3 Visualization Screens

Next, as can be seen in figure 4, the screens of the YourPantry application are shown following the navigation diagram shown above. The application's interface is simple and intuitive for anyone, including seniors, can use it.

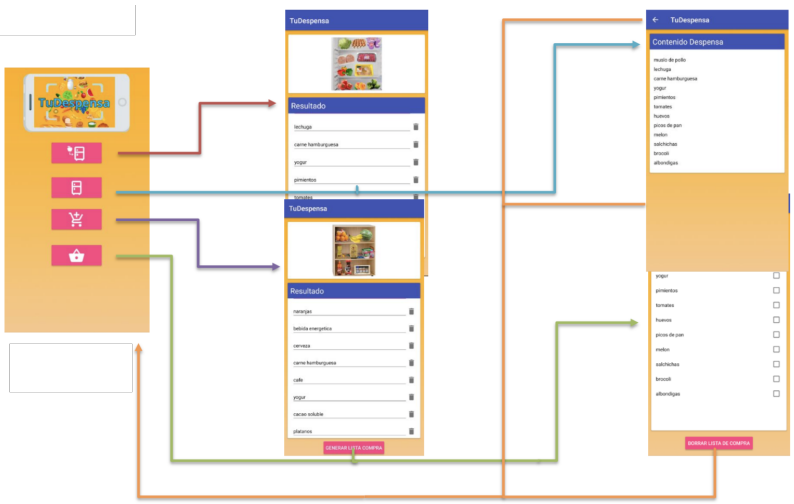


Fig. 4. Visualization screens.

3.4 Development Solution

Once we have described the system architecture, the navigation diagram and the screen visualization of the application, we will proceed to detail the development process. In which a process of preparation, training and optimization of the Deep Learning model has been carried out with Tensorflow.

The current image recognition models have millions of parameters. Training them from scratch requires a lot of training data and a lot of computing capacity (hundreds of hours of processing with a GPU).

In the same way, Transfer Learning is a technique that greatly simplifies this training process by taking a model that has already been trained and reusing it in a new one.

For this work we will reuse the feature extraction capabilities of the powerful image classifiers trained in ImageNet (MobileNet in this case) and form a new classification layer at the top for our purpose.

To start training the model, it is necessary to have a set of images so that it can be trained, in this work, following the proposed requirements, the model must recognize a considerable amount of food products.

The images used in this work have been obtained from ImageNet. For the retraining of the MobileNet model a script has been used that provides us with Tensorflow in a Codelab.

Once the model has been re-trained, the mobile application is generated with this model, and then the comparison is made with the images taken from the camera.

4 Conclusions

Today, society's food consumption is losing quality. The diet followed by people does not follow the recommendations of health experts, leading to different diseases, such as diabetes or cardiovascular disease. And in the elderly sector it is no different.

This paper presents an application for smartphones for the food monitoring and for the recommendation of diets according to the needs of each elder, making use Machine Learning and Deep Learning techniques.

The project is in the verification and validation phase, although satisfactory results have already been obtained with the first tests carried out.

Acknowledgements

This work was supported by 4IE project (0045-4IE-4-P) and 4IE+ project (0499 _4IE_PLUS_4_E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the Spanish Ministry of Science, Innovation and Universities (RTI2018-094591-B-I00), by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), and by the European Regional Development Fund.

References

1. Bloom, D.E., Chatterji, S., Kowal, P., Lloyd-Sherlock, P., McKee, M., Rechel, B., Rosenberg, L., Smith, J.P.: Macroeconomic implications of population ageing and selected policy responses. The Lancet **385**(9968), 649–657 (feb 2015). https://doi.org/10.1016/S0140-6736(14)61464-1, http://linkinghub.elsevier.com/retrieve/pii/S0140673614614641
2. Berry, E.H., Kirschner, A.: Demography of Rural Aging. In: Rural Aging in 21st Century America, pp. 17–36. Springer Netherlands, Dordrecht (2013). https://doi.org/10.1007/978-94-007-5567-3_2, http://www.springerlink.com/index/10.1007/978-94-007-5567-3{_}2
3. Morley, J.E., Silver, A.J.: Nutritional Issues in Nursing Home Care. Annals of Internal Medicine **123**(11), 850 (dec 1995). https://doi.org/10.7326/0003-4819-123-11-199512010-00008, http://annals.org/article.aspx?doi=10.7326/0003-4819-123-11-199512010-00008

4. Burris, M., Kihlstrom, L., Arce, K.S., Prendergast, K., Dobbins, J., McGrath, E., Renda, A., Shannon, E., Cordier, T., Song, Y., Himmelgreen, D.: Food Insecurity, Loneliness, and Social Support among Older Adults. Journal of Hunger & Environmental Nutrition pp. 1–16 (mar 2019). https://doi.org/10.1080/19320248.2019.1595253, https://www.tandfonline.com/doi/full/10.1080/19320248.2019.1595253
5. Evans, C.: Malnutrition in the elderly: a multifactorial failure to thrive. The Permanente journal **9**(3), 38–41 (2005), http://www.ncbi.nlm.nih.gov/pubmed/22811627http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3396084
6. Sayer, A.A., Cooper, C.: Early diet and growth: impact on ageing. The Proceedings of the Nutrition Society **61**(1), 79–85 (feb 2002), http://www.ncbi.nlm.nih.gov/pubmed/12002798
7. Darnton-Hill, I., Nishida, C., James, W.P.T.: A life course approach to diet, nutrition and the prevention of chronic diseases. Public health nutrition **7**(1A), 101–21 (feb 2004), http://www.ncbi.nlm.nih.gov/pubmed/14972056
8. GARIBALLA, S.: Malnutrition in hospitalized elderly patients: when does it matter? Clinical Nutrition **20**(6), 487–491 (dec 2001). https://doi.org/10.1054/clnu.2001.0477, http://www.ncbi.nlm.nih.gov/pubmed/11883996http://linkinghub.elsevier.com/retrieve/pii/S0261561401904777
9. World Health Organization: Methods of monitoring food and nutrient intake. In: World Health Organization (ed.) Preparation and use of food-based dietary guidelines, chap. 3. Geneva, nutrition programme edn. (1996), http://www.fao.org/3/x0243e/x0243e05.htm
10. Ledikwe, J.H., Smiciklas-Wright, H., Mitchell, D.C., Jensen, G.L., Friedmann, J.M., Still, C.D.: Nutritional risk assessment and obesity in rural older adults: a sex difference. The American Journal of Clinical Nutrition **77**(3), 551–558 (mar 2003). https://doi.org/10.1093/ajcn/77.3.551, http://www.ncbi.nlm.nih.gov/pubmed/12600842https://academic.oup.com/ajcn/article/77/3/551/4689684
11. Droogsma, E., Van Asselt, D.Z.B., Scholzel-Dorenbos, C.J.M., Van Steijn, J.H.M., Van Walderveen, P.E., Van Der Hooft, C.S.: Nutritional status of community-dwelling elderly with newly diagnosed Alzheimer's disease: Prevalence of malnutrition and the relation of various factors to nutritional status. The journal of nutrition, health & aging **17**(7), 606–610 (aug 2013). https://doi.org/10.1007/s12603-013-0032-9, http://www.ncbi.nlm.nih.gov/pubmed/23933871http://link.springer.com/10.1007/s12603-013-0032-9
12. Boulos, C., Salameh, P., Barberger-Gateau, P.: Malnutrition and frailty in community dwelling older adults living in a rural setting. Clinical Nutrition **35**(1), 138–143 (feb 2016). https://doi.org/10.1016/j.clnu.2015.01.008, http://www.ncbi.nlm.nih.gov/pubmed/25649256http://linkinghub.elsevier.com/retrieve/pii/S026156141500028X
13. Khanam, M.A., Qiu, C., Lindeboom, W., Streatfield, P.K., Kabir, Z.N., Wahlin, Å.: The metabolic syndrome: prevalence, associated factors, and impact on survival among older persons in rural Bangladesh. PloS one **6**(6), e20259 (2011). https://doi.org/10.1371/journal.pone.0020259, http://www.ncbi.nlm.nih.gov/pubmed/21697988http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3115931
14. Fuchs, K.L., Haldimann, M., Vuckovac, D., Ilic, A.: Automation of Data Collection Techniques for Recording Food Intake: a Review of Publicly Available and Well-Adopted Diet Apps. In: 2018 International Conference on Informa-

tion and Communication Technology Convergence (ICTC). pp. 58–65. IEEE (oct 2018). https://doi.org/10.1109/ICTC.2018.8539468, https://ieeexplore.ieee.org/document/8539468/

15. Müller, A.M., Alley, S., Schoeppe, S., Vandelanotte, C.: The effectiveness of e-& mHealth interventions to promote physical activity and healthy diets in developing countries: A systematic review. The international journal of behavioral nutrition and physical activity **13**(1), 109 (2016). https://doi.org/10.1186/s12966-016-0434-2, http://www.ncbi.nlm.nih.gov/pubmed/27724911http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5057225
16. L Holm, A., Laursen, M.B., Koch, M., Jensen, J., Diderichsen, F.: The health benefits of selective taxation as an economic instrument in relation to IHD and nutrition-related cancers. Public health nutrition **16**, 1–8 (2013). https://doi.org/10.1017/S1368980013000153
17. Doulah, A., Yang, X., Parton, J., Higgins, J.A., McCrory, M.A., Sazonov, E.: The importance of field experiments in testing of sensors for dietary assessment and eating behavior monitoring. In: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). pp. 5759–5762. IEEE (jul 2018). https://doi.org/10.1109/EMBC.2018.8513623, https://ieeexplore.ieee.org/document/8513623/
18. Farooq, M., Doulah, A., Parton, J., McCrory, M., Higgins, J., Sazonov, E., Farooq, M., Doulah, A., Parton, J., McCrory, M.A., Higgins, J.A., Sazonov, E.: Validation of Sensor-Based Food Intake Detection by Multicamera Video Observation in an Unconstrained Environment. Nutrients **11**(3), 609 (mar 2019). https://doi.org/10.3390/nu11030609, https://www.mdpi.com/2072-6643/11/3/609
19. Susan J. Crockett, Rosemary C. Tobelmann, Ann M. Albertson, Brenda L. Jacob: Nutrition Monitoring Application in the Food Industry. Nutrition Today **37**(3), 130–135 (2002), https://journals.lww.com/nutritiontodayonline/Abstract/2002/05000/Nutrition{_}Monitoring{_}Application{_}in{_}the{_}Food.13.aspx

A.3 Deployment of APIs on Android Mobile Devices and Microcontrollers

Authors: *Sergio Laso, Marino Linaje, Jose Garcia-Alonso, Juan M Murillo and Javier Berrocal.*

Publication type: *Conference paper (demo and artifact paper).*

Conference: *International Conference on Pervasive Computing and Communications (PERCOM), Austin, EEUU. 2020.*

Available online: *Demo: 10.1109/PerComWorkshops48775.2020.9156208. Artifact: 10.1109/PerCom45495.2020.9127353.*

Quality indicator: *GGS class (rating) 1 (A+).*

Deployment of APIs on Android Mobile Devices and Microcontrollers

Sergio Laso*, Marino Linaje*, Jose Garcia-Alonso*, Juan M. Murillo* and Javier Berrocal*
*University of Extremadura
Caceres, Spain
Email:{slasom, mlinaje, jgaralo, juanmamu, jberolm}@unex.es

***Abstract*—The high penetration and acceptance of smart devices has encouraged the development of IoT applications. The increase in the capabilities of these final devices has also led to the development of paradigms such as Fog and Edge Computing. Through these new architectural paradigms, developers can exploit the capabilities of the end devices to store, process and provide services, in order to improve the application by reducing the response time and the network overload. Currently, most applications are developed following a Server-Centric architecture because there are a lot of tools facilitating their development. However, these types of tools are not available for these emerging paradigms, which means an extra effort for developers. This paper shows a tool that semi-automatize the generation of applications based on the Edge Computing paradigm, thus reducing the developers' works and the application of these new architectural paradigms.**

***Index Terms*—Microservices, Android, Microcontroller, OpenAPI, Edge Computing**

I. Introduction

During the last few years, the capabilities of Internet-connected devices have increased enormously. Their processing and storage capacity have multiplied in order to be able to sense and/or process more information [1]. This increase has led to the development of the Internet of Things paradigm [2], in which these devices are used and coordinated to make people's life easier.

IoT applications require devices to be almost constantly sensing the environment, processing the gathered data and executing some actions for adapting the environment. For these applications to be really useful and accepted by end-users, they have to fulfill very stringent requirements regarding response time, location-awareness, etc. [3].

To meet these requirements, new paradigms such as Fog Computing, Edge Computing or, even, Mobile Centric architectures [4] have arisen for deploying applications closer to the end-user [5]. Fog Computing aims to to provide computing, storage and networking services between end devices and traditional cloud computing data centers. On the other hand, Edge Computing brings computational and storage capacities to the edge devices. Among other benefits, these paradigms reduce the network load and dependency on the cloud environment; improving the response time and providing better user experience. For example, using data closer or even inside the end device.

Currently, the predominant paradigm for deploying applications or APIs has been the Server-Centric or Cloud-Centric. In which the data gathered by IoT devices is sent to the cloud where it is stored and processed. Initially, that architectural style was applied because Internet-connected devices did not have enough capabilities for processing the gathered data [6]. It is still the most applied and used architectural style. This is because currently there is a myriad of tools and services facilitating the development of applications applying this style. For instance, OpenAPI [7] allows one to generate some of the code for both the server and the client from an application specification. Likewise cloud environment providers such as AWS [8] or Google [9] provide different tools.

Nevertheless, the number of tools providing support for the application of Fog, Edge or Mobile-Centric styles is much lower. This is because end devices or IoT devices were initially conceived as simple clients, not as cloud environment. In addition, in order to improve the security, they have a closed operating system. Therefore, implementing application making use of these paradigms require an extra effort by developers and a higher cost for software companies.

In this paper, we present a tool that helps developers to implement and deploy APIs on end devices, in order to easily apply the Edge and Mobile-Centric paradigms [10]. Concretely, it allows developers to deploy an API on Android devices and microcontrollers. To that end, common tools that are usually used by developers when they apply a Server-Centric paradigm have been adapted, reducing the impact and the learning curve. The presented approach requires an application designed with the OpenAPI specification. That design is used for generating the scaffolding of the application and all the communication interface required for deploying it on end devices.

The rest of the paper is structured as follows. Section II shows several related works. Section III explains the tool. Subsection III-A explains the process of deploying applications to end devices. Section IV describes a demonstration example. Finally, Section V details the conclusions.

II. Related Work

Few commercial tools support architectural styles such as Edge or Fog-based styles like AWS IoT Greengrass [11] or Azure IoT Edge [12]. However, at the research level, more and more proposals are presented fostering the use of the available specifications and resources to facilitate the development process of any application following these paradigms. For

instance, Noura et al. propose the WoTDL2API tool, which automatically generates a RESTful API for controlling different IoT devices [13]. This API is generated from an OpenAPI-based specification and using its toolchain for code generation. The generated source code can be deployed on peripheral devices, such as gateways, routers, etc. Nevertheless, it cannot make use of the capabilities of end devices.

In [14], the authors present a virtualization of IoT devices to improve response time for intelligent city applications. In the proposal, they apply a shared use of microservices and dynamic scaling of resources to improve the use of computing resources and the quality of the application. In [15], the authors also present a modular and scalable architecture based on lightweight virtualization. The modularity provided, combined with the orchestration provided by Docker, simplifies the administration and enables distributed deployments, creating a highly dynamic system. Both approaches distribute the computing load among different layers. However, the data is stored in a cloud layer. Besides, they do not provide tools for developers to easily apply their approaches.

In this paper, we present the modification of existing tools for applying the server-centric architectural style in order to also be able to deploy APIs on end devices, making the application of the Edge paradigm easier.

III. APIGEND - API Generator for End Devices

Currently, the specification and development of microservices are supported by a large number of tools that facilitate the work of the developer. Specifications such as OpenAPI are widely used to detail an API, generate documentation, perform tests and even, generate the API skeleton. This type of tools can generate the source code for different technologies such as Node.JS, Kotlin, JAX-RS, etc., but they focus on deploying the API on a server or on a cloud environment.

This section shows an extension of the OpenAPI Generator [16] to create and deploy an API on Android-based end devices (from API 23) and microcontrollers (ESP32), simplifying the development of these applications. In this way, these APIs can be consumed by third parties as if they were deployed on a cloud environment.

A. Deployment Process

APIGEND is part of a set of steps you need to follow to deploy APIs on end devices. This process can be seen in Figure 1. These steps are:

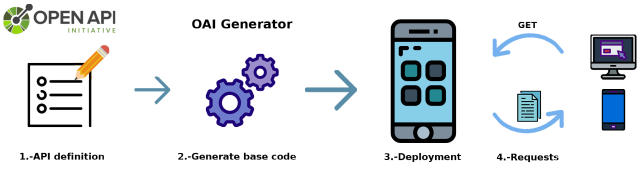


Fig. 1. Deployment Process.

1) **API Definition.** In this step, the API characteristics are defined, this task will be done through the OpenAPI Specification following the same notation as if it were developed and deployed on a cloud environment. The API specification will describe the different resources and endpoints that will be available to be invoked.
2) **Generate Base Code.** In this step, the API base code is generated with the presented tool. The tool also generates the source code to simulate the communication logic of an API Rest, reducing the effort required by developers. This communication logic has been implemented using Firebase Cloud Messaging [17] (for Android devices only) and MQTT [18], being able to decide during the generation of the source code what type of communication should be used.
3) **Deployment.** In this step, the developers only have to finish the implementation of the generated API and configure the messaging services.
4) **Requests.** Finally, with the deployed application, one can invoke the defined resources.

IV. Demo Case

In this demo, we will show attendees how to perform the whole process to deploy an API in an Android device and ESP32 Microcontroller and invoke the available services. We are going to define an API called Event Alerts. This API serves to post different cultural events within an area. You can also get the preferences of different users to organize events more adapted to them. Each device will represent a fictitious user and, therefore, each one will be able to alert with an event to the other users. They will also be able to obtain the preferences of each one to know which event is the most suitable to organize.

The steps to be realized in the demonstration are described below (following the steps described in Subsection III-A).

1) **API Definition.** We define the API with the OpenAPI Specification (OAS). This API will contain two microservices or endpoints:
   a) *Post Events:* This microservice is a POST type operation in which in the *body* of the request an object Event is included, which contain the information of the event and the area (lat,long,rad) in which we want to publish the event.
   b) *Get Users:* This microservice is a GET type operation that will take as parameters the area from which we want to get the users' preferences. This microservice will return the the user's information in an object detailing the user's id and a list of strings with their preferences.
2) **Generate Base Code.** In this step, the API base code is generated with *APIGEND*. For this demo, the MQTT communication interface will be used, as is shown in Figure 2.
3) **Deployment.** In this case, the MQTT communication interface of both devices will be configured, indicating the IP address and port of the MQTT broker and a small implementation will have to be made to send a predefined request from both devices.

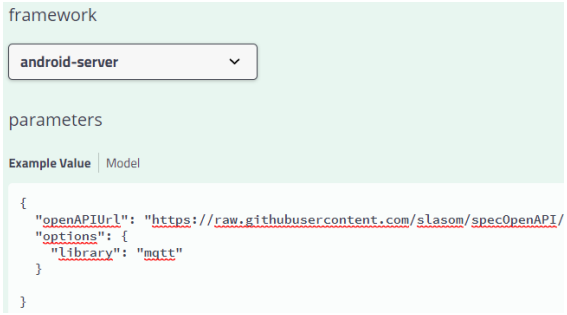


Fig. 2. Parameters to generate the Android application using the MQTT library.

4) **Requests.** After the deployment, we will be able to make requests. In this demo, both devices will be able to send predefined requests, the Android device will do it through a button in the application and the microcontroller through a physical button. As for the reception, the Android device will receive a push notification and will display the information in the main screen of the application as shown in Figure 3. The microcontroller will display the alert information a screen associated to the microcontroller.

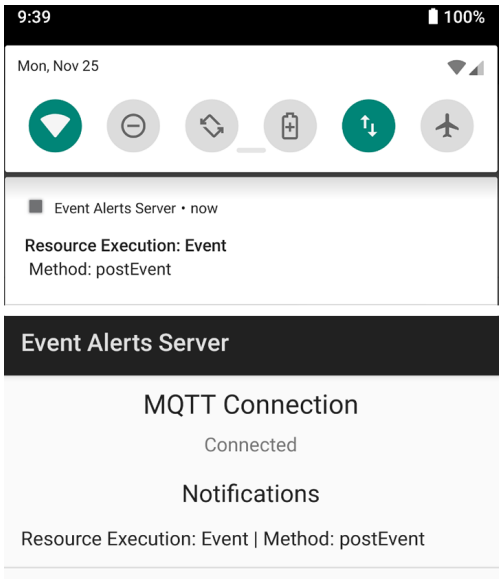


Fig. 3. Notifications received on the Android device

V. Conclusions

The increase of the computing capacities of the end devices opens the possibility to the application of architectural paradigms that until recently could only be used in very specific scenarios, obtaining benefits such as lower latency and lower consumption of network traffic. This paper discusses the lack of tools to help in the development of these alternative architectures. Therefore, it presents a tool that automates the creation of the skeleton of an API in order to be easily deployed on final devices such as Android devices or microcontrollers. This approach is based on tools weel-known by developers, so it reduces the learning curve and expands the target audience that could apply it.

Acknowledgment

This work was supported by the 4IE+ project (0499_4IE_PLUS_4_E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the Spanish Ministry of Science, Innovation and Universities (MCIU) (RTI2018-094591-B-I00) (MCIU/AEI/FEDER, UE), by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), and by the European Regional Development Fund.

References

[1] J. Berrocal, J. Garcia-Alonso, C. Vicente-Chicote, J. Hernández, T. Mikkonen, C. Canal, and J. M. Murillo, "Early analysis of resource consumption patterns in mobile applications," *Pervasive and Mobile Computing*, vol. 35, pp. 32 – 50, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119216300797

[2] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[3] H. Qian and D. Andresen, "Extending mobile device's battery life by offloading computation to cloud," in *2015 2nd ACM International Conference on Mobile Software Engineering and Systems, MOBILESoft 2015, Florence, Italy, May 16-17, 2015*, A. Abadi, D. Dig, and Y. Dubinsky, Eds. IEEE, 2015, pp. 150–151.

[4] J. Guillén, J. Miranda, J. Berrocal, J. Garcia-Alonso, J. M. Murillo, and C. Canal, "People as a service: A mobile-centric model for providing collective sociological profiles," *IEEE Software*, vol. 31, no. 2, pp. 48–53, 2014.

[5] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *2017 Global Internet of Things Summit (GIoTS)*. IEEE, 2017, pp. 1–6.

[6] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, "A survey on fog computing for the internet of things," *Pervasive and Mobile Computing*, vol. 52, pp. 71 – 99, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119218301111

[7] "The OpenAPI Specification Repository. Contribute to OAI/OpenAPI-Specification development by creating an account on GitHub." [Online]. Available: https://github.com/OAI/OpenAPI-Specification

[8] "Amazon Web Service." [Online]. Available: https://aws.amazon.com/

[9] "Amazon Cloud." [Online]. Available: https://cloud.google.com/

[10] "Openapi Generator Spilab." [Online]. Available: https://openapi-generator-spilab.herokuapp.com/

[11] "Aws IoT Greengrass." [Online]. Available: https://aws.amazon.com/greengrass/

[12] "Azure IoT Edge." [Online]. Available: https://azure.microsoft.com/services/iot-edge/

[13] M. Noura, S. Heil, and M. Gaedke, "Webifying heterogenous internet of things devices," in *International Conference on Web Engineering*. Springer, 2019, pp. 509–513.

[14] K. Ogawa, K. Kanai, K. Nakamura, H. Kanemitsu, J. Katto, and H. Nakazato, "Iot device virtualization for efficient resource utilization in smart city iot platform," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2019, pp. 419–422.

[15] M. Alam, J. Rufino, J. Ferreira, S. H. Ahmed, N. Shah, and Y. Chen, "Orchestration of microservices for iot using docker and edge computing," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 118–123, 2018.

[16] "Openapi Generator." [Online]. Available: https://github.com/OpenAPITools/openapi-generator

[17] "Firebase Cloud Messaging." [Online]. Available: https://firebase.google.com/docs/cloud-messaging

[18] "MQTT." [Online]. Available: http://mqtt.org/

Artifact Abstract: Deployment of APIs on Android Mobile Devices and Microcontrollers

Sergio Laso*, Marino Linaje*, Jose Garcia-Alonso*, Juan M. Murillo* and Javier Berrocal*
*University of Extremadura
Caceres, Spain
Email:{slasom, mlinaje, jgaralo, juanmamu, jberolm}@unex.es

***Index Terms*—Microservices, Android, Microcontroller, OpenAPI, Edge Computing**

I. Introduction

This artifact is a guideline for the generation of APIs through the APIGEND (API Generator for End Devices) tool. This tool is an extension of the OpenAPI Generator [1]. It originally allows developers to create both the client and server side through an OpenAPI Specification with a Server-Centric style in different languages. The extension developed also allows one to generate APIs for end devices, specifically for Android devices and ESP32 Microcontrollers, making the application of the Edge [2] and Mobile-Centric [3] paradigms easier.

II. Requirements and Deployment

APIGEND is a web application developed with Spring[1]and uses Mustache's templates[2] for code generation. APIGEND can be deployed locally using the development environment by running the *openapi-generator-online* module as shown in Figure 1, on a Docker[3] container as it has a DockerFile or directly on a cloud environment. This last option has been followed for deploying APIGEND on Heroku[4] and which can be accessed through the following link:

https://openapi-generator-spilab.herokuapp.com

The source code can be dowloaded from the following public Bitbucket repository:

https://bitbucket.org/spilab/openapigenerator

III. Generating APIs

This section shows the complete process to generate APIs, specifically for Android devices with MQTT [4] as communication protocol, although to generate APIs with other languages, the procedure is practically the same.

When we access the web interface we have three sections (Gen-Api-Controller , Clients and Servers). We will have to access the Servers section and the endpoint called *"Generates a server library"* as shown in Figure 2.

[1]https://spring.io
[2]https://mustache.github.io
[3]https://www.docker.com
[4]https://heroku.com

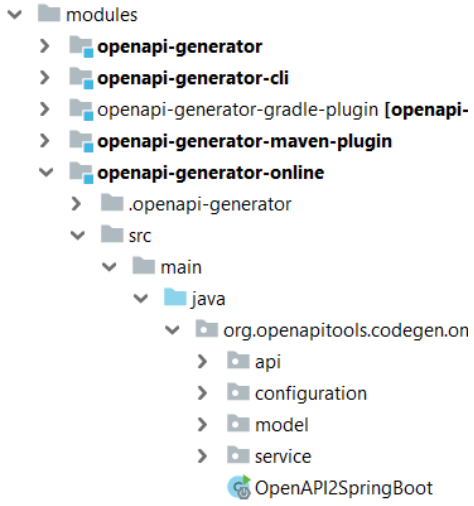


Fig. 1. Module Package.

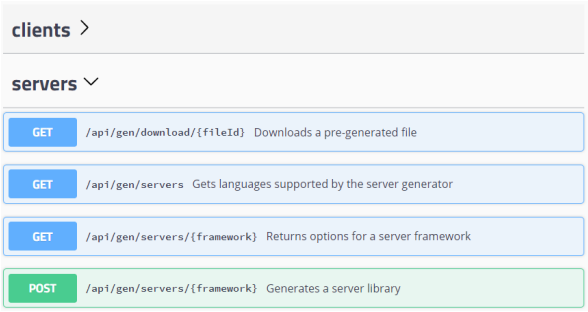


Fig. 2. Web Interface.

To generate the API for an Android device with MQTT, we must select the option *Try it Out*. This will allow us to specify the parameters required for generating the API. Concretely, we have to specify in *framework* the *"android-server"* option. In the *parameters* section, we have to indicate in *"openAPIUrl"* the url of the API specification and in "options" we have to indicate as *library*, *"mqtt"* (if no library is indicated, by default it is generated using Firebase Cloud Messagging [5]). Figure 3 shows an example for the use case that will be followed during the demo.

To generate the API, click on the *Execute* button and if

978-1-7281-4657-7/20/$31.00 ©2020 IEEE

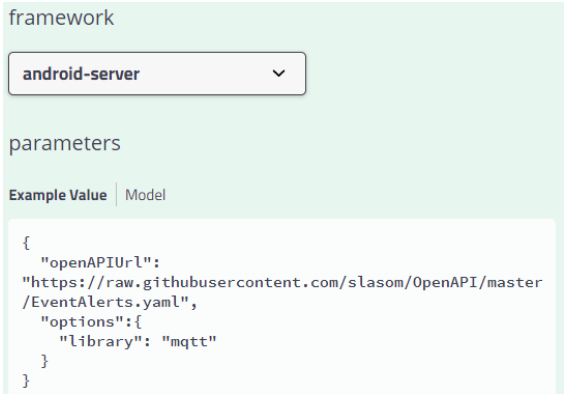


Fig. 3. Parameters to generate the Android application using the MQTT library.

there is no error in the specification, we will get as a result a JSON. Inside the JSON, the link option specify the URL for downloading the application as Figure 4 shows. If we copy and paste it into the browser bar, a *.zip* file with the API generated will be downloaded automatically.

Fig. 4. Correct Generation.

IV. Deploying APIs on end devices

In this section, we will proceed to show how the deployment could be done for Android devices. Having downloaded the .zip file, it have to be unzipped in order to finish the implementation of the different endpoints defined in the specification and also to configure the MQTT communication protocol. To do this, it is necessary to access the *MQTTConfiguration.java* file located in the following path:

`src/main/java/org/openapitools/server/service/`

In the file, we must introduce the IP and the port of the MQTT broker to which we want to connect the API for the reception of requests.

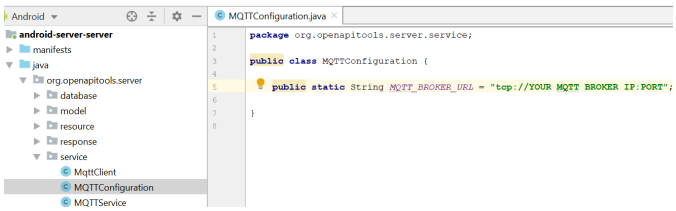


Fig. 5. Android Configuration.

V. Requests on end devices

This section explains the structure that has to be followed to invoke an endpoint employing MQTT. The Listing 1 shows the content of a request to one of the endpoints declared in the Event Alerts API[5]. This is the specification that will be used during the defined demo.

The different parameters that have to be specified are:

- **resource**: this parameter corresponds to the names of the *Tags* created in the specification. There are two in this specification, *Event* and *Location*.
- **method**: corresponds to the *operationId* of each endpoint in the specification.
- **sender**: this parameter is automatically included in the Response of the API (it is not necessary to indicate it in the specification), it can be useful to introduce, to which client we want to respond to. In this case, the sender's topic is specified to receive the reply.
- **params**: in this parameter are included the objects or parameters that go in the request. In this method, the specification has a request body with the *Event* object schema as a parameter.

The **MQTT topic** for sending requests to the API is the *title* of the specification without spaces. In this case, *EventAlerts*.

Listing 1. Content of an Event Alerts API request.

```
{
    "resource": "Event",
    "method": "postEvent",
    "sender": "client3248",
    "params": {
        "event": {
          "id": 1,
          "title": 'Football Match!',
          "description": 'Football match at 11:00 in CC',
          "location":{
                "latitude": 38.514377,
                "longitude":-6.844325,
                "radius": 200
                }
          }
        }
}
```

Acknowledgment

This work was supported by the Innovation and Universities (MCIU) (RTI2018-094591-B-I00) (MCIU/AEI/FEDER, UE), by Government of Extremadura (GR18112, IB18030), and by the European Regional Development Fund.

References

[1] "Openapi Generator." [Online]. Available: https://github.com/OpenAPITools/openapi-generator
[2] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *2017 Global Internet of Things Summit (GIoTS)*. IEEE, 2017, pp. 1–6.
[3] J. Guillén, J. Miranda, J. Berrocal, J. Garcia-Alonso, J. M. Murillo, and C. Canal, "People as a service: A mobile-centric model for providing collective sociological profiles," *IEEE Software*, vol. 31, no. 2, pp. 48–53, 2014.
[4] "MQTT." [Online]. Available: http://mqtt.org/
[5] "Firebase Cloud Messaging." [Online]. Available: https://firebase.google.com/docs/cloud-messaging

[5]https://raw.githubusercontent.com/slasom/OpenAPI/master/EventAlerts.yaml

A.4 SOLID and PeaaS: Your Phone as a Store for Personal Data

Authors: *Manuel Jesús-Azabal, Javier Berrocal, Sergio Laso, Juan Manuel Murillo and Jose Garcia-Alonso.*

Publication type: *Conference paper (long paper).*

Conference: *International Conference on Web Engineering, Helsinki, Finland. 2020.*

Available online: *10.1007/978-3-030-65665-2_1.*

Quality indicator: *GGS class (rating) 3 (B-).*

SOLID and PeaaS: your phone as a store for personal data

Manuel Jesús-Azabal, Javier Berrocal, Sergio Laso, Juan Manuel Murillo and Jose Garcia-Alonso

QSEG, Universidad de Extremadura, Avda. Universidad S/N, 10003 Cáceres (Spain)
{manuel, jberolm, slasom, juanmamu, jgaralo}@unex.es

Abstract. Advertising has become the most important source of income for a significant number of web-based companies. This income is usually dependent on the personal information that companies gather from their users which has led them to create very rich profiles of their users. However, these profiles do not follow any standard and are usually incomplete in the sense that users provide different subsets of information to each platform. Thus, the quality and quantity of the data varies between applications and tends to inconsistency. In this context, the SOLID initiative proposes an alternative to decentralize the user information giving them complete ownership of their information. In this demo, we propose a proof of concept in which SOLID is used to store the user information in their mobile device, following the People as a Service paradigm to provide this information as a service to third parties.

Keywords: SOLID · Mobile-centric architecture · Advertisement · Personal Data · Smartphones.

1 Introduction

Advertisement on the Internet is based on very specific targeting, so the right products are proposed to the appropriate public. Therefore, it is critical to identify user preferences and interests [1]. Moreover, corporations invest a lot of money in obtaining relevant data about their users. This background is used to better understand how to focus the marketing campaigns and advertisement of trademarks and products. Enterprises, like Facebook or Google, mainly base their business model on the management of this kind of information [2].

Companies using this business model are under the obligations imposed by the different personal data protection laws existing around the world. However, the potential of these companies to infer more complex information [3], which can be used for private ends or politics interests [4] is a current concern. Also, the lack of any storage standard implies that each enterprise maintains their own set of information, causing a big duplicity of resources. Additionally, data leaks suppose a potential threat for users who can find their data exposed. This problems were already brought to the public attention in cases like Cambridge Analytica [5]. An example of these problems can be music streaming platforms.

A user that plays music 90% of the time in an application (e.g. Spotify) and 10% of the time in another (e.g. YouTube), would have accurate recommendations in the first one but poor in the second one. The user would also have two music listening profiles, each of them incomplete.

To address this, different projects are trying to establish an alternative model for personal data management. One of the most relevant works in this area is the SOLID initiative [6], a movement conceived by the World Wide Web ideator, Tim Berners-Lee. This project proposes the use of PODS (Personal Online Data Stores) to store personal information. Thus, external applications which require the information must be authorized by the PODS manager and they have to actively request the information.

In this demo, we present a prototype that integrates a SOLID PODS into a smartphone. This idea is based on the People as a Service (PeaaS) [7] paradigm. PeaaS propose to use smartphones as a virtual representation of their owners. Thus, the paper is organised as follows. First, Section 2 briefly describes some alternatives for massive data storage, explaining the relevance of SOLID and PeaaS. Second, Section 3 specifies the details about the presented demo and how the PODS is integrated to allow the smartphone to serve the user information as a service. Finally, Section 4 draws some conclusions about the paper.

2 Background and Related Work

The management and storage of personal information have elicited multiple works which try to bring transparency and control for the users. The SOLID initiative [6] is the most popular focused on personal information management, but it is not the only one which introduces an alternative models for data storage. There are also several works based on a distributed network where data storage is decentralized. Examples of these are [8], [9], [10] or [11].

The HAT project [8] (Hub of All Things) is similar to SOLID in the sense that it proposes a centralized entity to store the personal information. This centralized microserver provides the users' personal information. Thus, external applications consume the provided data without having to privately store it. It is a proposal which shares the purpose with SOLID, nevertheless, it is more centered on how to distribute the information with the external entities while HAT brings a model based on the combining, production and exchange of information.

Freenet [9] was originally born to fight censure and guarantee anonymous navigation. Thus, it distributes the information along with the network components, sharing bandwidth and storage space. Moreover, Freenet is based on the same principles as blockchain and support the delocalization of massive storage.

The Dat Foundation [10] promotes universal access and distribution of knowledge, avoiding data monopolies and private massive storage. The initiative has mainly boosted the Dat Protocol, which provides a peer-to-peer tool for dataset sharing. Furthermore, the system implements the tracking of data version [12], following the free software philosophy.

The Threefold Network [11] is an ecosystem for technology development which provides communication tools based on three main concepts: equality, freedom and sustainability. Thus, the project proposes a peer-to-peer model where data is only stored by the owner entity.

These three approaches bring alternative models for information storage and interactions between network elements. Nevertheless, the collaborative nature of the projects and the data scalability do not provide a concrete pattern to standardize personal information storage. SOLID is a proposal focused on changing the paradigm of data management and bringing real privacy to users. For that purpose, the users' personal information is kept in PODS, entities which work as a user profile and manage requests from external apps. The PODS is managed by the own user, who specifies the personal information and the applications which can access to this data. Thus, the user must approve external demands, becoming able to revoke the access anytime. As a result, personal information is separated from applications which must actively demand the data and remain authorized.

In this paper, we present a solution which integrates the SOLID philosophy into PeaaS [13], a paradigm that proposes the use of smartphones as a virtual representation of their owners. Due to the amount of sensors present in those devices and their pervasive presence in society they are the perfect tool to gather users information. Moreover, thanks to their computation capabilities, the user profiles gathered in smartphones can be used in a variety of ways.

Additionally, as detailed above, SOLID provides a solution to manage the user information in a single profile. This goal is perfectly aligned with the PeaaS paradigm, which needs a storage mechanism for the user virtual profile. Thus, this demo brings an integration between these two works, drawing on the possibilities of the smartphone. In the next section, we present an overview of the proposed solution and how external applications interact with the system.

3 Storing and accessing SOLID information in PeaaS

This paper proposes a proof of concept for personal data storage on smartphones using Solid under the PeaaS paradigm. The proposed solution deploys a PODS in a smartphone, serving as provider of the personal data to authorized applications which demand it. As a result, the personal information is stored in the user's device, managing external accesses and getting a full control of the data.

The proposed solution is based on two main elements: an smartphone, which serves the data and performs the storage process, authorization control and definition of the information; and an API gateway, an intermediate layer which is in charge of communicating the external requests with the user device.

The smartphone is in charge of executing a local SOLID PODS. This element is originally designed to be deployed in web servers. However, for this demo we have deployed it on an smartphone (technical details to replicate this demo can be found in [1]). Once this process is finished, the deployed application provides to

[1] https://bitbucket.org/spilab/solidsituational.context/src/master/

the user all mechanisms to specify the personal information and manage external requests. To adapt the behaviour of the PODS to the PeaaS paradigm, it is necessary to install a PeeaS Application (also included in the above mentioned repository). This program will be in charge of executing a callback method in the smartphone when a call from the API Gateway is received. The API Gateway works as a common layer for all devices, providing the external request with the communication mechanism to reach the SOLID PODS. Figure 1 shows a step-by-step guide of the solution.

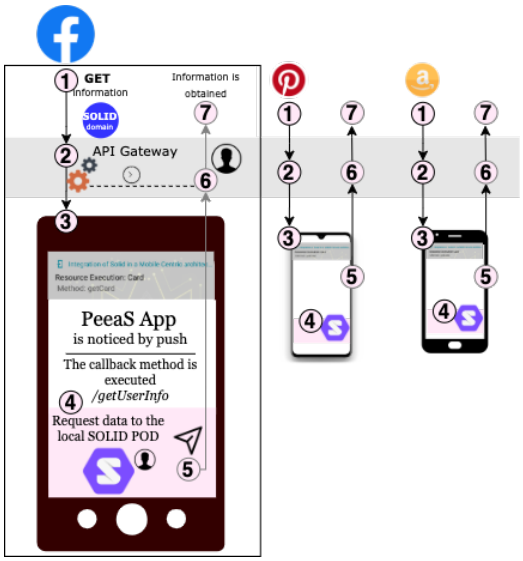


Fig. 1. Serving data from a smartphone based PODS.

The process begins when an external application requires information from the user. Since the data is stored in the local PODS, the API Gateway is in charge of communicating the request to the device. Thus, the process begins when the external application provides the individual SOLID domain of the data owner (1). Next, the API Gateway maps the SOLID domain with the corresponding device, notifying the request and waiting for the smartphone response (2). When the smartphone receives the notification (3), it executes a callback method in the PeeaS Application (4) which internally requests the personal data to the PODS deployed in the phone. When the personal profile information is obtained (5), the smartphone returns the data to the API Gateway (6), which responds to the external application (7). This way, the full communication process is completed. It is important to take into account the API Gateway is a common entity for all calls and interactions with the phones, becoming a relevant intermediate element.

Following this working scheme, an example case of use is explained in Figure 2: Facebook wants to know the geolocalization position and interests of a concrete user to display custom advertisement. First, it is important that the user specifies Facebook as a friendly domain in the PODS, otherwise, the information request would not be allowed. This way, Facebook application knows the SOLID Domain of the user, value used to indicate the smartphone involved in the request. This variable is provided by the user when registration process is made. Thus, Facebook marks the GPS position and user interests set as desired information to the API Gateway, performing a GET petition and waiting for the response. Internally, the API Gateway maps the provided SOLID Domain and notifies the PeeaS application in the smartphone about the petition. Once the device is informed, authorization checks are made in the PODS. In the case the requester application is accepted, a callback method is executed and the SOLID PODS replies with the GPS position and interests set. As a result, the smartphone returns the data to the API Gateway, which provides Facebook with the demanded information. This working scheme is followed every time an external app requests a concrete data.

Fig. 2. Process to obtain GPS value and interests data from user PODS.

The proposed solution is based on the interoperation of multiple entities. As a result, the SOLID PODS works autonomously on the user smartphone, achieving a real separation between personal information and external applications. Thus, the governance of the data is entirely placed on the user. As a consequence, the personal information is managed by the user, becoming able to control the accesses from external parties.

4 Conclusions

Data governance has become an important issue, specially for online advertisement. Personal information has become an appreciated coin and enterprises invest important money amounts to obtain the users' personal information [1].

In this paper, a demo which joins the SOLID philosophy into the PeaaS paradigm has been provided. Therefore, smartphone devices become personal

PODS for data storage. This way, the users keep the personal information on the phone and external applications must request it. Thanks to this, personal data is integrated into the individual device, allowing the user to keep a full control over the stored information and the access of external applications. Joining these two paradigms encourages a change of mind in privacy philosophy while providing a vanguard technological solution.

Acknowledgment This work has been partially funded by the 4IE+ project (0499-4IE-PLUS-4-E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the Spanish Ministry of Science, Innovation and Universities (RTI2018-094591-B-I00), by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), and by the European Regional Development Fund.

References

1. Minna Ruckenstein and Julia Granroth. Algorithms, advertising and the intimacy of surveillance. *Journal of Cultural Economy*, 13(1):12–24, 2020.
2. Christian Fuchs. The political economy of privacy on facebook. *Television & New Media*, 13(2):139–159, 2012.
3. Sandra C Matz, Jochen I Menges, David J Stillwell, and H Andrew Schwartz. Predicting individual-level income from facebook profiles. *PloS one*, 14(3), 2019.
4. Vladlena Benson and Tom Buchanan. Social big data and its integrity: The effect of trust and personality traits on organic reach of facebook content. In *Cyber Influence and Cognitive Threats*, pages 145–158. Elsevier, 2020.
5. Ikhlaq ur Rehman. Facebook-cambridge analytica data harvesting: What you need to know. *Library Philosophy and Practice*, pages 1–11, 2019.
6. The SOLID Foundation. The solid project. https://solid.mit.edu. Accessed: 2020-02-27.
7. Joaquin Guillen, Javier Miranda, Javier Berrocal, Jose Garcia-Alonso, Juan Manuel Murillo, and Carlos Canal. People as a service: a mobile-centric model for providing collective sociological profiles. *IEEE software*, 31(2):48–53, 2013.
8. Irene CL Ng. Can you own your personal data? the hat (hub-of-all-things) data ownership model. 2018.
9. Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing privacy enhancing technologies*, pages 46–66. Springer, 2001.
10. The DAT Foundation. The dat foundation. https://dat.foundation. Accessed: 2020-04-15.
11. The Threefold Network. The threefold network. https://threefold.io. Accessed: 2020-04-15.
12. Maxwell Ogden, Karissa McKelvey, Mathias Buus Madsen, et al. Dat-distributed dataset synchronization and versioning. *Open Science Framework*, 10, 2017.
13. Javier Berrocal, Jose Garcia-Alonso, Carlos Canal, and Juan M Murillo. Situational-context: a unified view of everything involved at a particular situation. In *International Conference on Web Engineering*, pages 476–483. Springer, 2016.

A.5 FoodScan: Food monitoring app by scanning the groceries receipts

Authors: *Beatriz Sainz-De-Abajo, José Manuel García-Alonso, José Javier Berrocal-Olmeda, Sergio Laso-Mangas and Isabel De La Torre-Díez.*

Publication type: *Journal paper.*

Journal: *IEEE Access (2020).*

Available online: *10.1109/ACCESS.2020.3046031.*

Quality indicator: *Q2.*

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 27, 2020, accepted December 16, 2020, date of publication December 21, 2020,
date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3046031

FoodScan: Food Monitoring App by Scanning the Groceries Receipts

BEATRIZ SAINZ-DE-ABAJO[1], JOSÉ MANUEL GARCÍA-ALONSO[2], (Member, IEEE), JOSÉ JAVIER BERROCAL-OLMEDA[2], (Member, IEEE), SERGIO LASO-MANGAS[2], AND ISABEL DE LA TORRE-DÍEZ[1]

[1]Department of Signal Theory, Communications and Telematics Engineering, University of Valladolid, 47011 Valladolid, Spain
[2]Department of Computer Science and Telematic Systems Engineering, University of Extremadura, 10003 Caceres, Spain

Corresponding author: Beatriz Sainz-de-Abajo (beasai@tel.uva.es)

This work was supported by the Interreg V-A España-Portugal 2014-2020, under Project 0499_4IE_PLUS_4_E.

ABSTRACT In the mobile device market there is a large number of applications to help people monitor intake or provide suggestions to lose weight and manage a healthy diet. However, the vast majority of these apps consume a lot of time by having to introduce food one by one. This paper presents the work to develop and pilot test a new Android application, FoodScan, aimed at people over 70, specially those from rural environments or with limited technical knowledge, to manage their food from the items that appear on their grocery receipts, avoiding the obligation to introduce one by one those foods, and generating recommendations. To achieve this final objective, specific objectives have been completed as indicated in the methods section. We conducted a review of current calorie control applications to learn about their weaknesses and strengths. Different algorithms were tested to expedite the introduction of food into the application and the most suitable for the FoodScan application was selected. Likewise, several options were taken into account to create the knowledge base of food, taking into account dietary recommendations for people over 70 years. Once developed, a pilot evaluation was carried out with a convenience sample of 109 volunteers in rural areas of Caceres and Valladolid (Spain) and Alentejo (Portugal). They tested FoodScan for a month after which they completed a user satisfaction survey. 93 % (101/109) believed that the app was easy to download and install, 66 % (72/109) thought that it was easy to use, 47 % (51/109) noted that the charts with the recommendations helped them with diet control and 49 % (53/109) indicated that FoodScan helped them improve healthy eating habits. One-month pilot evaluation data suggested that most users found the app somewhat helpful for monitoring food intake, easy to download and easy to use.

INDEX TERMS MHealth, android, food intake monitoring, elderly, automatic dietary assessment, Web scraping, optical character recognition algorithm (OCR), user evaluation.

I. INTRODUCTION

The guidelines to follow in any healthy diet are summarized in: (1) variety in foods; (2) reduce fat consumption; (3) increase the consumption of complex carbohydrates such as fruits and vitamins; (4) increase the consumption of fiber-rich foods; (5) reduce sugar and salt; (6) maintain the intake of vitamin D and calcium; (7) hydrate; (8) moderate alcohol consumption; (9) make 4 or 5 meals a day reducing the amounts in each one; and (10) exercise regularly. Technology is a highly effective tool for controlling diet and physical activity [1], [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Shen Yin.

Overweight, which affects all age groups, has increased to alarming levels as a consequence not only of excessive calorie intake but also of being more sedentary [3], [4]. The increase in the production of processed foods, the rapid urbanization and the change in lifestyles have led to a change in eating habits. More and more people are far from their ideal weight, which is a serious health problem due to related heart diseases, diabetes, stroke, etc. Eating a healthy diet helps protect us from malnutrition in all its forms, as well as the diseases mentioned above. It is necessary to change people's lifestyles, controlling the amount of food we eat and start eating healthier.

In rural areas, with an increasingly aging population and with medical services distant from their homes, it is observed

that many elderly people who live alone are suffering from nutritional deficiencies that, in some cases, can cause significant damage to their physical condition. To determine the health of the elderly population and avoid these problems, we should have accurate information about their nutritional patterns to predict their quality of life [5]–[7].

The objective the FoodScan application is to facilitate the control of the elder diet, specially of those living in rural areas. These people have limited technical knowledge and prefer simple devices. In the development of the application, it was taken into account that connectivity can also be a problem. Sometimes the Internet connection does not arrive in optimal conditions to certain rural areas or they do not even have any signal. FoodScan is a simple application that collects the information by scanning the purchase groceries receipts and that does not require an Internet connection for its management. We can store food in an integrated way without having to do it one by one. It has been developed for the Android Operating System, since it has a higher penetration rate than other operating systems.

The steps that we have carried out throughout the work have been: (1) to review existing Calorie Control Apps; (2) to analyse algorithms registering the purchased groceries and select a food collection to compare; (3) to develop the FoodScan App; and (4) to conduct a user evaluation.

II. RELATED WORK

Technology can help people to maintain a healthy diet [8]. A large number of applications have been designed for this purpose [9], [10]. Some manage sports activities or physical exercise plans [11]–[14]. Others focus on food control [15]. Nevertheless, most of those applications unify both aspects: control of food, caloric intake and sports activities.

Generally, applications for food control have the main function of registering and storing meals throughout the day. The user must enter, one by one, the type of food ingested. This controls the calories ingested in a day and the specific nutrients (proteins, carbohydrates, fats, vitamins), and allows comparisons with the data recorded from previous days. Applications can incorporate other functions such as: (1) set goals to lose weight over a period of time; (2) store healthy food recipes; or (3) include a step counter to complement the record of calories ingested with those that have been burned throughout the day doing physical exercises.

Mobile applications have been developed to scan purchase receipts [16], [17]. We found a study to measure the effects of front-of-pack interpretive nutrition labels on grocery shopping using label scanning[18]. In another article, the authors tried to know, by registering receipts, the behavior of buying food in households [19]. In addition, that will be effective if there is a commitment by the user to scan the tickets. However, we have not found any mobile application that photographs and records the groceries receipts to compare them over a period of time, and based on that comparison show recommendations for a balanced diet.

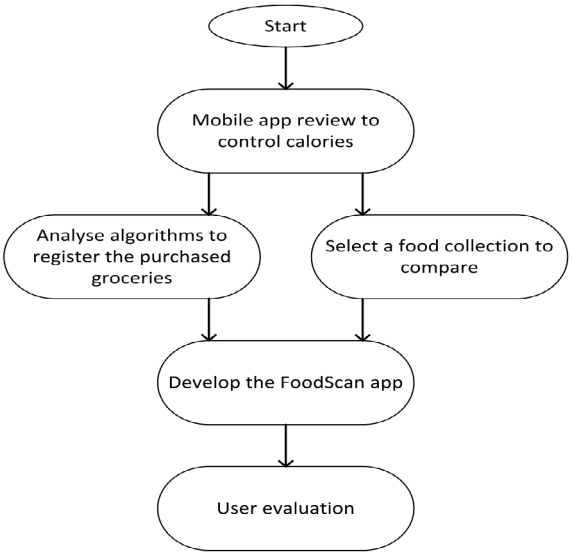


FIGURE 1. Flow chart.

III. METHODS

A. WORK FLOW

As a starting point, we carry out a review of mobile applications that offer characteristics similar to those we try to implement in our app. After this comparative review, several algorithms were analysed to speed up the introduction of food into the application. It was also very important to choose the most appropriate food knowledge base, taking into account the food recommendations for people over 70 years old living in rural areas. After achieving these objectives, we developed the application and finally carried out a satisfaction survey among end users. Figure 1 shows the flow chart followed in this work.

B. REVIEW OF CURRENTLY AVAILABLE MOBILE APPS TO CONTROL CALORIES

A search was performed through Google, Scopus and Web of Science. The terms used were: ''best calorie counter apps'', ''apps to control calorie intake'', ''apps to control diet'', and ''apps to control feeding''. After an exhaustive examination of mobile applications, we focused on the three that had the higher number of downloads in order to analyse their functionalities and features: Lose It! [20], Fat Secret [21] and My Fitness Pal [22]. Five people downloaded these three free applications and tested them for a period of approximately two months. They tested them simultaneously by entering the same data in all three applications. It was decided that the evaluators would analyse characteristics of the applications such as ease of downloading, user registration in the application, graphical interface, simplicity of the food registration process, possibility of scanning food barcodes, and using graphs and charts for providing the results. After this trial period, opinions on the functionality of the applications were shared.

C. ANALYSIS OF ALGORITHMS REGISTERING THE PURCHASED GROCERIES AND SELECTION OF A FOOD COLLECTION TO COMPARE

Three options were tested: (1) Web Scraping; (2) object recognition and; (3) Optical Character Recognition (OCR).

Web Scraping allows one to extract large amounts of information from web pages in an automated and dynamic way. In order to prove its effectiveness, online shopping websites were crawled with the aim of analysing and comparing the user's different purchases over a period of time. The technique is not efficient for our purpose because pages have different structures. In addition, data protection problems must be taken into account [23]–[25]. Finally, people over 70 years old living in rural areas do not use online shopping websites.

Image recognition identifies objects in an image or video sequence and facilitates their subsequent classification [26]–[28]. To test the algorithm, photographs of the products of the refrigerator were taken several days apart to see the food that the user consumed during that time interval. Comparing the photographs, and from a healthy diet scheme or template, it can be concluded if the user followed a balanced diet or needs to buy or consume certain foods. Although it is a good strategy to compare images, the amount of comparisons is very large. The vectorization process consumes a large amount of resources, and the methodology to recognize the similarity between these objects is very complex. If we add difficulty in taking a clear and clean photograph of the items in the refrigerator, it is concluded that this technique does not turn out to be the most efficient for our target population.

The Optical Character Recognition (OCR) algorithm consists of the digitalization of texts that are automatically identified from an image, symbols or characters that belong to a certain alphabet [29], [30]. There is a wide range of tools or libraries that make the use of OCR algorithms easier for developers. Asprise OCR SDK provides an API to recognize text and image barcodes. The problem is that the demo has very limited functions [31], [32].

As a search method to select a food collection to compare, we analysed online bibliographic references. It is important to know the recommendations of the specialists, and differentiate the types and groups of food that are classified according to the distribution of the food pyramid for people with the age range above 70 years [33]. And we must keep it in mind to avoid the nutritional disorders of the elderly.

D. APP STRUCTURE AND DEVELOPMENT

FoodScan has been developed for a minimum Android 6.0 version, equivalent to the API level 23, and the version with which the project has been compiled is Oreo 8.1, equivalent to the API level 27. Support for older APIs is provided since the target users of the application tend to own low-end devices that are not frequently updated and the latest data shows that more than 84 % of Android devices run a version 6.0 or higher. Nevertheless, the proposed application has been tested in new and more powerful devices with similar results. The functions allowed by the application are:

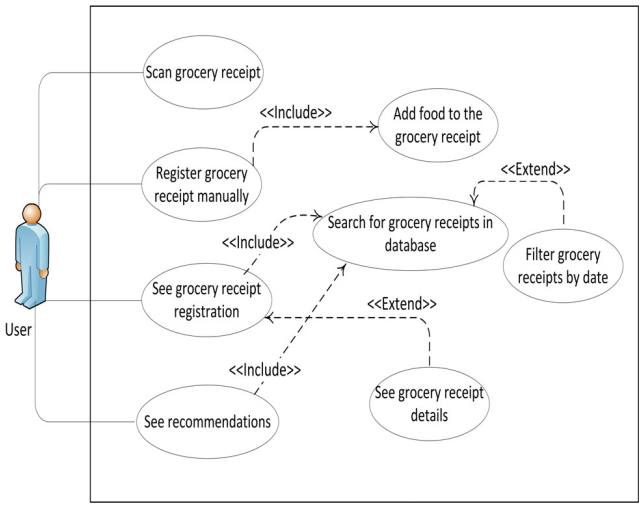


FIGURE 2. FoodScan use-cases diagram.

1) Scan grocery receipts for food registration.
2) Manually add a food to an already registered grocery receipt, or create a new one.
3) View the record of stored grocery receipts, and search and filter by date.
4) See summaries of statistics, by groups of food consumed in the week or month and those that remain to be consumed, through graphs.

The domain of the FoodScan ontology would be "Food" as the main class. Different food groups are established as subclasses, namely: Legumes, Vegetables, Fruits, Cereals and derivates, Dry fruits, Milk and derivates, Eggs, Meats, and Fish. In the case of meat and fish, new subclasses are established based on the proportion of fat: fatty and lean. Finally, more than 100 individuals, specific products that can be automatically recognized by the application, has been included in the ontology, distributed along the different classes. The relationships that appear in this ontology are only belonging. A food belongs to a certain class, regardless of relationships between classes. We save and export the ontology in a file of the RDF / XML type. The file uses different brands or labels to indicate classes, individuals and relationships. It is possible to modify this file, to add new individuals to some class, because it is not static. For the implementation of the ontology we used the editor ''Protégé''[34].

The use-cases diagram shows the structure of the application, the actors involved and the different functions performed by those actors. These types of diagrams are designed to show the interaction between the system and the user, or another system. User interactions with the application allows them to: (1) scan a grocery receipt; (2) register a grocery receipt manually; (3) see records; and (4) see recommendations. See in Figure 2.

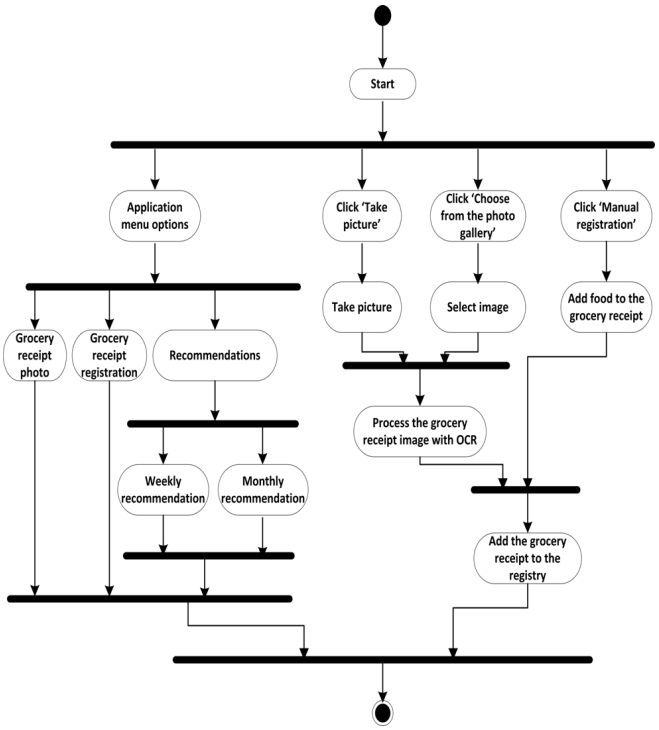


FIGURE 3. FoodScan Activity Diagram.

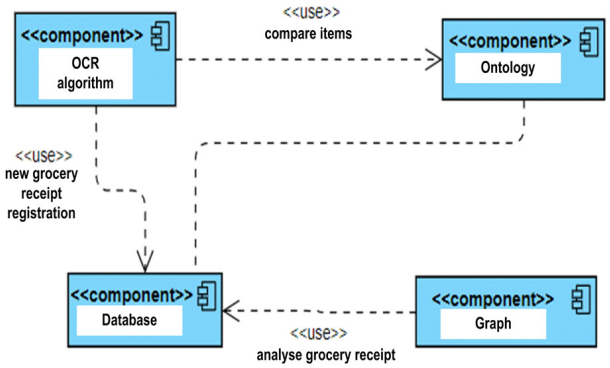


FIGURE 4. FoodScan architecture.

The activity diagram (see Figure 3) shows the workflow that is performed and details the different routes between the events of the application.

Two tables are created for the design of the relational database and implemented using the SQLite database. The "grocery receipts" table contains the grocery receipt data in the app, storing an automatically generated grocery receipt code that identifies it, the date on which this receipt was added to the database and a text with the address where the image is stored of the grocery receipt. The "food" table contains in each of its records the name of the recognized food and the code of the grocery receipt to which it belongs.

In Figure 4, one can observe the relationships between the modules that integrate FoodScan: (1) OCR algorithm; (2) food ontology; (3) groceries receipt database; and (4) functions of the result graph.

Since the application data is personal and unique for that application it is not necessary to upload it to a remote server. That is why we have chosen to use SQLite as the internal database for the Android application. Data management is done locally and, therefore, personal data is not sent to a remote server to be analysed, improving the privacy of the proposed solution.

To generate the graphics, we use MPAndroidChart [35]. We established the recommended weekly quantities for each food group, and they were all stored together in a numerical array. Each bar of the graph is composed of two sections. The first section indicates the food bought. The second section of the bar presents the food that remains to be ingested. This amount is the result of the operation: Recommended amount - Ingested amount.

E. END USER EVALUATION

A small group of volunteers visited rural areas from the provinces of Caceres and Valladolid (Spain) and Alentejo (Portugal) to show FoodScan. The population was notified about the investigation by means of posters placed in the civic centers and the town hall. The number of people who attended the first demonstration meeting exceeded 270 among all the rural areas visited. Volunteers gave a practical demonstration of how to download and enter food data into the FoodScan app to get graphs of results.

About a month after the first meeting, volunteers returned to conduct the web based survey among those who used the FoodScan application. A considerably smaller number of users attended this second meeting: 109 Spanish and Portuguese users. All of them tested the application and were interested in answering the satisfaction survey. Please, note that no incentive were offered by the volunteers.

The questions were created by the researchers. It is not based on an established user satisfaction survey. A person in charge of the Privacy Policy Service of the University of Valladolid reviewed the survey and recommended to eliminate the questions that could identify the respondents. After following the recommendations, the survey was approved. In addition, participants were not asked to provide demographic characteristics. We did not consider requesting these data because we did not aim to know the health of the participants. We were only interested in the age of the respondents because the application is aimed to people over 70 years old. All the people recruited for the study were over 50 years old with medium-low technological knowledge.

Although the data collected is anonymous and therefore it is not necessary to include transparency clauses on data protection in the survey, an informed consent was also collected from those who completed the survey. Participants were informed by letter of the content of the survey and asked about their implicit consent. Participants who gave ethical approval were offered the possibility to use tablets to conduct the survey.

TABLE 1. User satisfaction survey.

Question	User response			
What weight loss apps have you used?	Open question			
What level of difficulty did you have during the download and installation?	Difficult	Average	Easy	Very easy
Has it been easy to use?	Difficult	Average	Easy	Very easy
Do the charts with recommendations help control the diet?	Little	Somewhat	Average	A lot
Does it help improve your healthy diet habits?	Little	Somewhat	Average	A lot
What would you be willing to pay for FoodScan?	≤$4.99	$5 to $9.99	$10 to $14.99	$15 to $19.99
Would you recommend this application?	Yes	No		

The tool for the development of the survey was Google Forms due to its simplicity and because a license is not required for its use.

A survey was carried out with 7 questions (see Table 1), accessible through a link provided to the users of the application. We were especially interested in the answers to questions regarding the usability and ease of downloading the application. We decided that the number of questions in the survey should be small, so that people would be encouraged to complete it. Thanks to the simplicity of the questionnaire, all the people who carried out the survey answered all the questions. These results were transferred to an Excel spreadsheet in order to better analyse the results and generate graphs.

IV. RESULTS

A. SUMMARY OF THE REVIEW OF CURRENTLY AVAILABLE APPS

All three applications offer similar features and interfaces. The user can set the ideal weight (s)he wants to achieve and how quickly (s)he hopes to achieve it. Depending on their age, weight, height and sex, the application indicates a daily calorie regime. The user has to record the daily food and physical exercises, can visualize in a graph the calorie and nutrient intake, follow the progress in their marked objectives, modify the quantities or portions of the foods, and see suggestions of healthy recipes. The success of these three applications compared to others is that it is possible to scan the food barcode with the camera of the mobile device and register the food. The scanned item is recognized if it is stored in the database. If the food is not stored, it can be manually included. They are visual applications and have different sections. Graphs show the values of calories and nutrients in quantities and percentages. It also offers a simple and navigable interface, with the possibility of configuring the app in several languages.

But not everything are advantages. The main problem of these applications is that the user must introduce food and physical activities one by one. This can be a tedious and time-consuming task, since they have to provide the food, the quantity and the time of day at which it is ingested. In the long run, there is an increased likelihood of abandonment due to the high burden related to entering foods into apps. Therefore, in order to increase the adherence, users need applications registering the food intake without requiring introducing these foods one by one.

B. ANALYSIS OF ALGORITHMS REGISTERING THE PURCHASED GROCERIES AND SELECTION OF A FOOD COLLECTION TO COMPARE

The one that was finally used in FoodScan is the OCR algorithm. The OCR character recognition services of web pages were tested. Some of those consulted were Newocr.com, Onlineocr.net, Ocr.space, and Tess4j together with tesseract [36]. In the Android Studio development environment, there were difficulties in making the Tess4j library compatible. Finally, Google Vision was used, which allows us to understand the content of an image by encapsulating powerful machine learning models in a simple and easy-to-use way [37]–[39].

Foods are categorized into different groups. It is a data store that allows us to compare the scanned and recognized foods of the grocery receipts using the OCR algorithm. The considered alternatives were: (1) UNESCO Thesaurus [40]; (2) Database; and (3) Ontologies. For FoodScan we use an ontology, because it offers several functions that allow us to build a simple and complete collection. It can be adjusted to the needs of FoodScan, to help and advise people who use it in the diet, offering recommendations based on purchases made within a certain period of time.

C. FOODSCAN APPLICATION

The FoodScan home screen is shown in Figure 5. It offers three options to add or register a grocery receipt in the app's database: (1) take a picture; (2) choose an image from the gallery; and (3) add a new groceries receipt by selecting foods manually. The 3 horizontal lines icon, in the upper left of the screen, takes us to the menu with the functions of the app: (1) back to the beginning; (2) open the record of the groceries receipts stored in the database; and (3) initiate recommendations or statistics.

Figure 6 shows the grocery receipts registration screen. It allows one to insert the "start date" and the "end date". The "Filter" button shows only the grocery receipts between these dates. If the dates are not selected correctly, an error message will be displayed on the screen. If the dates are not selected, all the grocery receipts registered in the database are shown. By clicking on a grocery receipt, a pop-up menu will appear with 3 options: (1) see details, which offers groceries receipt details, the food it contains and an image if it exists; (2) delete, which allows the grocery receipt to be deleted; or (3) cancel, to close the menu without changes. The user can manually add a food to the grocery receipt among those stored in the application. This is done sequentially as can be seen in Figure 7.

Regarding the recommendations of FoodScan based on the recorded data, we decided to provide them using visual techniques such as graphs. FoodScan has the advantage of

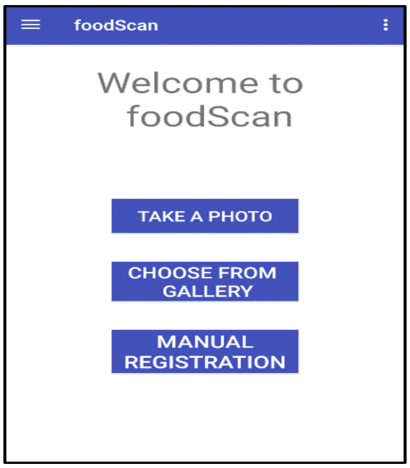


FIGURE 5. FoodScan home screen.

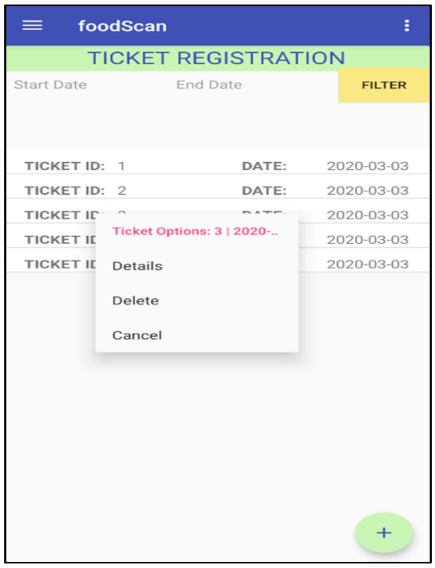


FIGURE 6. Menu of the groceries receipt list.

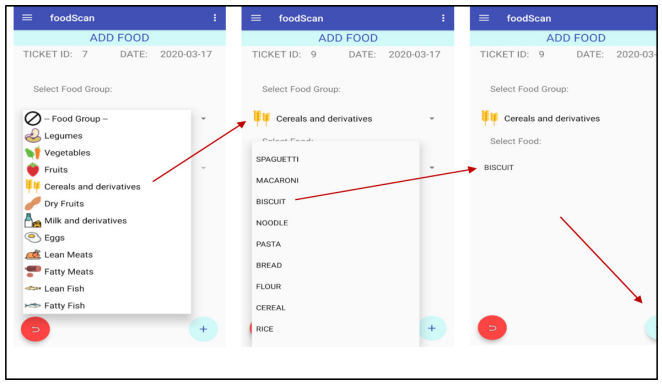


FIGURE 7. Sequence to manually add a food to the groceries receipt.

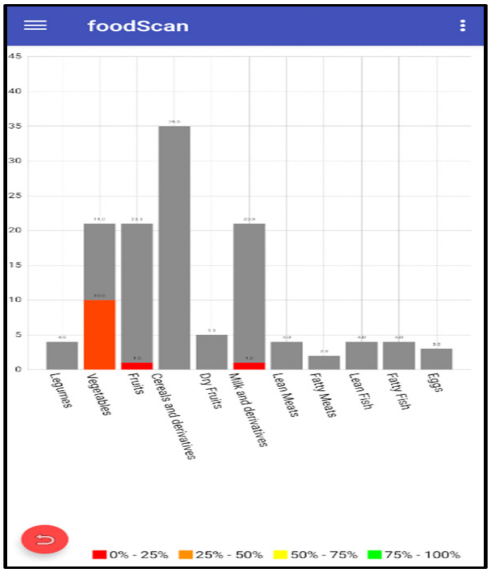


FIGURE 8. Graph with recommendations.

offering weekly or monthly recommendations. In the case of the weekly recommendation, the user must select the day on which that week begins, and the application will take that day and the next six. If it is monthly, the user must select the month and year. In both cases, the app will search the database for grocery receipts included in this period of time, and if there are grocery receipts on those dates, it will show the graph with the results of the analysis. If there is not any grocery receipt in the database, the process is canceled and the app displays a message to the user warning of the error.

FoodScan offers a visual analysis of the foods of each group consumed in a given time, comparing them with the amounts recommended by specialists, to follow a balanced diet. If there are groceries receipts stored for that period of time, a bar graph is shown for each food group with the amount of food consumed, and those that remain to be consumed according to the established recommended quantities. A possible limitation of FoodScan is that it only tracks the food bought, not the one consumed. As a criterion of colors, we consider the "traffic light system", whose colors range from green to dark red, and which mark the different levels of risk. A Color Code has been established based on the percentage of food consumed: (1) between 0-25 %, red color; (2) between 25-50 %, orange color; (3) between 50-75 %, yellow color; and (4) between 75-100 %, green color. If the color of the bar for a certain food group is not green, the user should adjust their diet to meet the recommended amounts (in grey color). In Figure 8, we see that only three food groups were added to the application database: (1) vegetables; (2) cereals and derivatives; and (3) milk and derivatives. All in insufficient quantities and that is why the bars are in orange and red. In a balanced diet, Figure 8 would show all the bars in green. Therefore, from the visualization of this graph, the user knows that the consumption of food should be increased, if (s)he wants to follow a healthy diet, according to the recommendations of the dietitians.

D. RESULTS OF THE END USER EVALUATION

The opinion of the users allows us to know if it has a friendly interface. In addition, the information collected will help us to improve the application in the future.

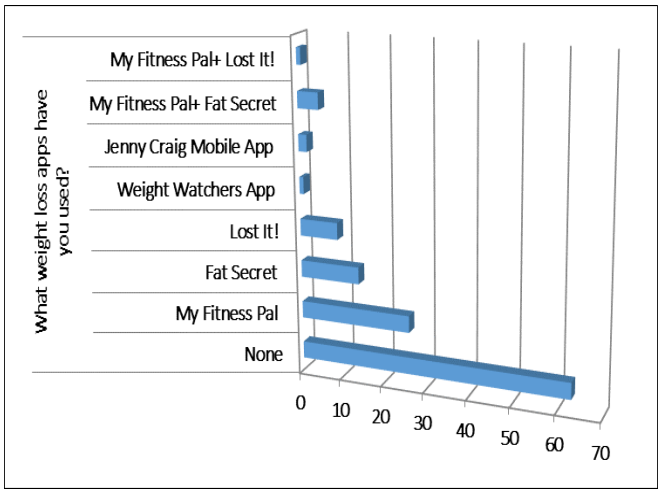


FIGURE 9. Weight Loss Apps usage.

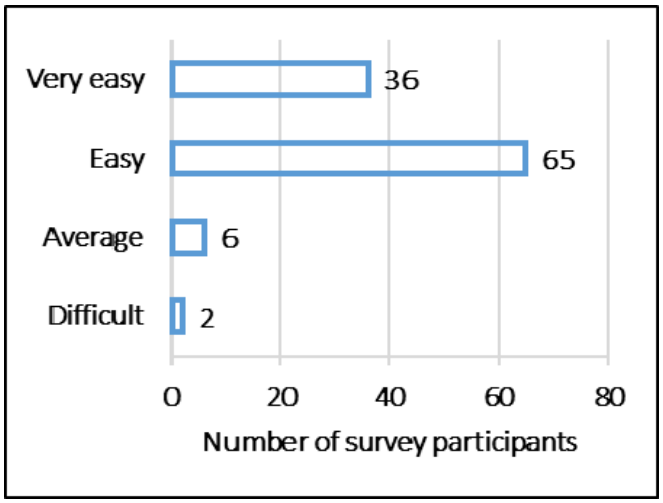


FIGURE 10. Problems downloading and installing.

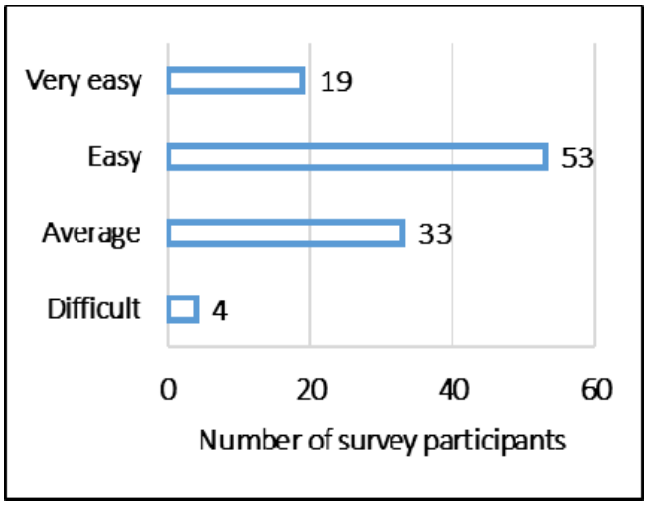


FIGURE 11. Ease of use.

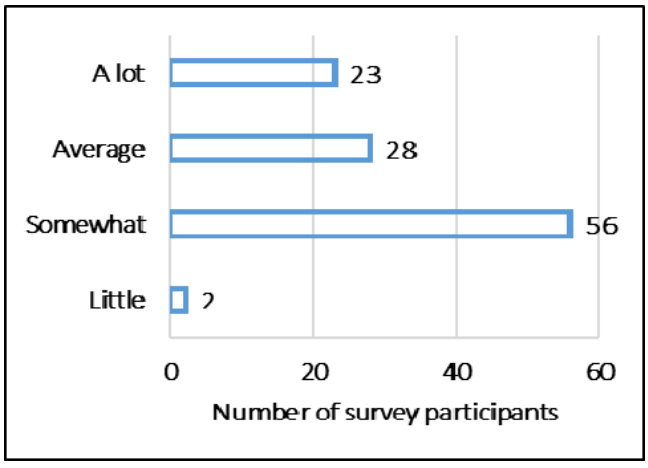


FIGURE 12. Do the charts with recommendations help control the diet?.

Regarding the use of applications for weight loss, more than half users answered that they did not use any before trying FoodScan. Those who had used weight loss applications indicated that the most used were My Fitness Pal, Fat Secret and Lost It! Few users reported having used more than one application. See Figure 9.

Regarding the download and installation of FoodScan, 65 out of 109 participants (60 %) answered that it was easy and 36 out of 109 (33 %) very easy (see Figure 10). As can be seen in Figure 11, regarding ease of use, 4 out of 109 participants (4 %) found it difficult, 33 out of 109 (30 %) average difficulty, 53 out of 109 (49 %) easy and 19 out of 109 (17 %) very easy.

As can be seen in Figure 12, with the exception of 2 of the total of 109 polled (2 %) who do not consider it useful, the public appreciates the recommendations shown in the charts. As shown in Figure 13, users generally agree with the fact that the app helps them improve their healthy eating habits. In Figure 14, one can see the price users would be willing to pay if it were marketed. It is an application that, in general, has liked. 101 out of 109 participants (93 %) would recommend it to other users (see Figure 15).

V. DISCUSSION AND CONCLUSION

A. PRINCIPAL RESULTS

In this work, we started from the analysis of mobile applications that offer similar characteristics to those that we try to implement in FoodScan. After this comparative review, various algorithms were analysed to accelerate the introduction of food and the most suitable food knowledge base for people over 70 years old. Next, we developed the application and finally we carried out a satisfaction survey among users who used the application for a month. Below we summarize the objectives achieved:

1) Review the currently available apps. The analysis allowed us to better understand its characteristics, weaknesses and strengths. The main weakness identified is that the information of the food intake has to be manually provide to the analysed apps. Table 2 compares the features of FoodScan with other apps.

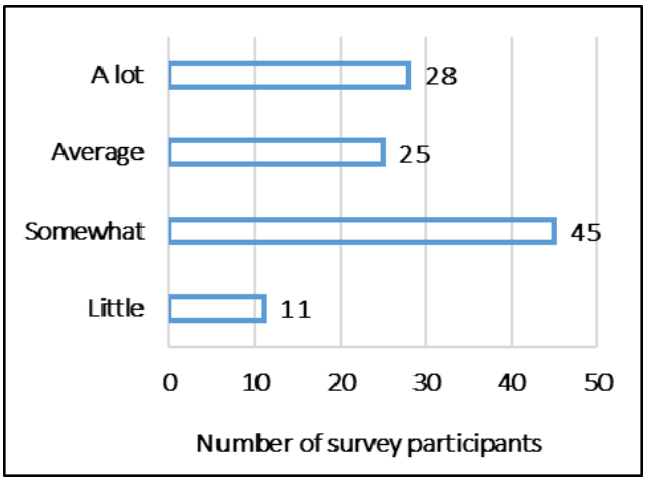


FIGURE 13. Does it help improve your healthy diet habits?.

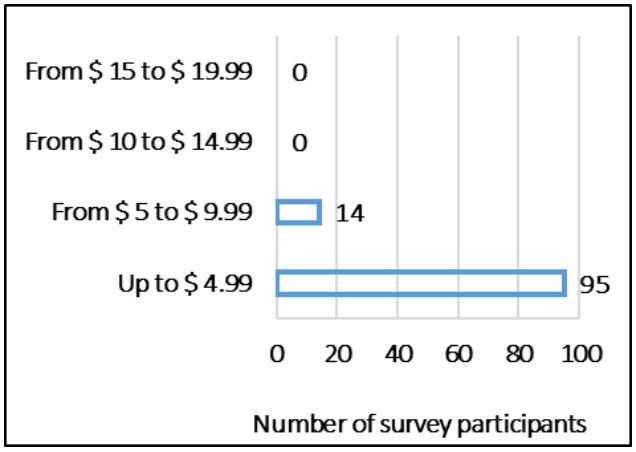


FIGURE 14. What amount of money would you be willing to pay?.

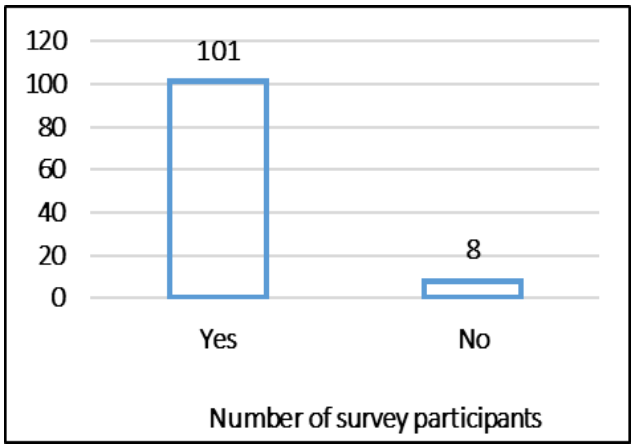


FIGURE 15. Would you recommend it?.

Only our application allows you to register the food included in the ticket by scanning.

2) Analysis of algorithms registering purchased foods. Three options were tested: (1) Web Scraping; (2) object recognition and; (3) Optical Character Recognition (OCR). OCR was finally used because of the number of services available and its accuracy. Also, we selected a food collection to compare. The following alternatives that were considered: (1) UNESCO Thesaurus; (2) Database; and (3) Ontologies. We used ontologies because it allowed us to easily build a complete collection.
3) Development of the FoodScan application. We based on the premise that our target audience is people over 70 years old, with limited technical knowledge and living in rural areas. This population group is generally reluctant to use technologies and, therefore, the need to create a simple, intuitive, friendly application that offers visual results that are very easy to understand is crucial to increase the user experience. In addition, FoodScan works without being connected to the Internet, so it does not require any communication for using it. FoodScan is an Android application that meets the characteristics and functionalities necessary for these users to control their food, in an easy and intuitive way. It allows users to add the food products they purchased by scanning grocery receipts. By means of an OCR algorithm, the grocery receipts are analysed and the food that matches to those stored is added to a grocery receipts register. With these grocery receipts stored, the app offers users a visual and understandable analysis of the product bought in each group during a period of time, comparing them with the amounts recommended by the specialists.
4) End user evaluation. The results of the satisfaction survey, completed by users who tried the FoodScan tool, highlight its ease of installation and handling.

TABLE 2. Comparison of the features of FoodScan and other commercial apps.

Diet control apps	Record the foods one by one and quantities	Register groceries receipts	Record physical exercise	Daily progress in calorie intake
Lose It!	√	×	√	√
Fat Secret	√	×	√	√
My Fitness Pal	√	×	√	√
FoodScan	√	√	×	√

B. LIMITATIONS

The implemented application and this study have some limitation that should be highlighted:

1) The satisfaction survey do not ask for the demographics information of the participants. We did not consider their inclusion because the only requirement for participating in this study was to be over 50 years old. They all met the demand.
2) FoodScan tracks what is purchased, not what is consumed. Therefore, it cannot be used to estimate the daily nutritional intake, since some of the purchased groceries could be shared or thrown away. These

drawbacks were explained during the demonstration of the application and the users were given correct guidelines. Also, consideration should be given to whether the person lives alone or not.

3) The user should follow a good practice in using the application, but if (s)he shares, throws away and buys more than necessary, the results will not be realistic. We are not able to specify what a person has eaten at a certain time, but we get a general idea in a period of a week or a month.
4) Further research is required to determine if users are able to persist in app use over the long term and if the app leads to improvement in dietary intake. We are also interested in knowing the lack of adherence to the application. The responses obtained in a short period of use of the application cannot offer reliable data on the improvement in diet habits or weight loss.

C. CONCLUSION

The elderly who are not used to new technologies may find it difficult to use mobile applications to control their diet. The disadvantage of many of these applications is that they require a lot of time to introduce the food consumed throughout the day. From the user's point of view, it is not practical and, in the long run, most of them will abandon this control. Our app is useful and easily manageable and avoids the obligation to introduce one by one those foods. The analysis of the purchased food is offered through a bar graph with different colors, so it is simple and intuitive. The elderly, or the people in their care, can control how they are eating or what food deficiencies they may have.

It can be used by all kinds of public, regardless of their technical knowledge. Although it is geared towards serving people over the age of 70, we hope that it will also be used by anyone in their daily lives and that they will be loyal in the long term. Only by adopting healthy habits we can improve our quality of life.

FoodScan is an application that we have just developed and will continue to improve thanks to the contributions of users. We will regularly monitor the 109 people who completed the survey on the regular use of FoodScan to control their diet.

D. FUTURE LINES OF WORK

One of the aspects of the application to consider in the future is to extend it and implement a more complete ontology and a bigger food collection by adding more quantity and variety. Furthermore, we are already working to expand the food ontology and include users of a wider age range. Another important point would be to improve the design of the application based on user ratings. This application is configured in Spanish, Portuguese and English. It is planned to adapt it to other languages to reach a greater number of users.

To make the application more powerful, artificial intelligence should be added, capable of learning what items refer to a specific food, since a direct comparison is currently made. We will have to weigh that it does not affect the performance and/or speed of the groceries receipt analysis, so that the user does not have to wait a long time. Another improvement to be developed is the possibility of making the food analysis more complete by offering, a more specific study of nutrients and calories ingested.

REFERENCES

[1] S.-W. Chen, D.-L. Chiang, T.-S. Chen, H.-Y. Lin, Y.-F. Chung, and F. Lai, "An implementation of interactive healthy eating index and healthcare system on mobile platform in college student samples," *IEEE Access*, vol. 6, pp. 71651–71661, 2018.

[2] M. Chukwu, "Personalized mobile monitor for assisted healthy-living," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2011, pp. 18–22.

[3] *Obesity: Preventing and Managing the Global Epidemic Report of a WHO Consultation*., World Heal. Organ., Geneva, Switzerland, 2000.

[4] M. A. Subhi, S. H. Ali, and M. A. Mohammed, "Vision-based approaches for automatic food recognition and dietary assessment: A survey," *IEEE Access*, vol. 7, pp. 35370–35381, 2019.

[5] E. Moguel, J. Berrocal, and J. García-Alonso, "Systematic literature review of food-intake monitoring in an aging population," *Sensors*, vol. 19, no. 15, p. 3265, Jul. 2019.

[6] A. A. Sayer and C. Cooper, "Early diet and growth: Impact on ageing," *Proc. Nutrition Soc.*, vol. 6, no. 1, pp. 79–85, 2002.

[7] I. Darnton-Hill, C. Nishida, and W. James, "A life course approach to diet, nutrition and the prevention of chronic diseases," *Public Health Nutrition*, vol. 7, no. 1a, pp. 101–121, Feb. 2004.

[8] S. Pagoto, K. Schneider, M. Jojic, M. DeBiasse, and D. Mann, "Evidence-based strategies in weight-loss mobile apps," *Amer. J. Preventive Med.*, vol. 45, no. 5, pp. 576–582, Nov. 2013.

[9] B. Y. Laing, C. M. Mangione, C.-H. Tseng, M. Leng, E. Vaisberg, M. Mahida, M. Bholat, E. Glazier, D. E. Morisky, and D. S. Bell, "Effectiveness of a smartphone application for weight loss compared with usual care in overweight primary care patients," *Ann. Internal Med.*, vol. 161, no. 10, p. 5, Nov. 2014.

[10] J. Tang, C. Abraham, E. Stamp, and C. Greaves, "How can weight-loss app designers' best engage and support users? A qualitative investigation," *Brit. J. Health Psychol.*, vol. 20, no. 1, pp. 151–171, Feb. 2015.

[11] J. Stephens and J. Allen, "Mobile phone interventions to increase physical activity and reduce weight: A systematic review," *J. Cardiovascular Nursing*, vol. 28, no. 4, pp. 320–329, 2013.

[12] G. Flores Mateo, E. Granado-Font, C. Ferré-Grau, and X. Montaña-Carreras, "Mobile phone apps to promote weight loss and increase physical activity: A systematic review and meta-analysis," *J. Med. Internet Res.*, vol. 17, no. 11, p. e253, Nov. 2015.

[13] G. Bleser, D. Steffen, M. Weber, G. Hendeby, D. Stricker, L. Fradet, F. Marin, N. Ville, and F. Carré, "A personalized exercise trainer for the elderly," *J. Ambient Intell. Smart Environ.*, vol. 5, no. 6, pp. 547–562, 2013.

[14] P. K. Diwakar, Y. Keun Oh, S.-H. Park, and Y.-R. Yoon, "Personal digital exercise trainer for managing, monitoring and recording the exercise," in *Proc. IEEE Eng. Med. Biol. 27th Annu. Conf.*, 2005, pp. 3720–3723.

[15] C. M. Wharton, C. S. Johnston, B. K. Cunningham, and D. Sterner, "Dietary self-monitoring, but not dietary quality, improves with use of smartphone app technology in an 8-Week weight loss trial," *J. Nutrition Edu. Behav.*, vol. 46, no. 5, pp. 440–444, Sep. 2014.

[16] A. Jäckle, J. Burton, M. P. Couper, and C. Lessof, "Participation in a mobile app survey to collect expenditure data as part of a large-scale probability household panel: Coverage and participation rates and biases," *Surv. Res. Methods*, vol. 13, no. 1, 2019.

[17] B. Read, "Respondent burden in a mobile app: Evidence from a shopping receipt scanning study," *Surv. Res. Methods*, vol. 13, no. 1, pp. 45–71 2019.

[18] E. Volkova, B. Neal, M. Rayner, B. Swinburn, H. Eyles, Y. Jiang, J. Michie, and C. N. Mhurchu, "Effects of interpretive front-of-pack nutrition labels on food purchases: Starlight randomised controlled trial," *Obesity Res. Clin. Pract.*, vol. 8, pp. 110–111, Dec. 2014.

[19] S. A. French, S. T. Shimotsu, M. Wall, and A. F. Gerlach, "Capturing the spectrum of household food and beverage purchasing behavior: A review," *J. Amer. Dietetic Assoc.*, vol. 108, no. 12, pp. 2051–2058, Dec. 2008.

[20] *Lose It*. Accessed: Oct. 27, 2020. [Online]. Available: https://www.loseit.com/

[21] *Fat Secret*. Accessed: Oct. 27, 2020. [Online]. Available: https://www.fatsecret.com/

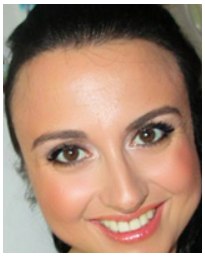
[22] *Myfitnesspal*. Accessed: Oct. 27, 2020. [Online]. Available: https://www.myfitnesspal.com/
[23] O. Castrillo-Fernández, "Web scraping: Applications and tools," Eur. Public Sector Inf. Platform (EPSI platform), Tech. Rep. 10, 2015, pp. 1–31. [Online]. Available: https://pdfslide.net/documents/web-scrapingapplications-and-tools.html
[24] F. Polidoro, R. Giannini, R. Lo Conte, S. Mosca, and F. Rossetti, "Web scraping techniques to collect data on consumer electronics and airfares for Italian HICP compilation," *Stat. J.*, vol. 31, no. 2, pp. 165–176, 2015.
[25] R. Penman, T. Baldwin, and D. Martinez, "Web scraping made simple with SiteScraper," *Penman Web Scraping*, pp. 1–10, May 2009. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.654.745&rep=rep1&type=pdf
[26] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, 2013.
[27] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *Int. J. Comput. Vis.*, vol. 61, no. 1, pp. 55–79, 2005.
[28] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, vol. 2, Sep. 1999, pp. 1150–1157.
[29] A. Chaudhuri, K. Mandaviya, P. Badelia, and S. K. Ghosh, "Optical character recognition systems," *Stud. Fuzziness Soft Comput.*, vol. 352, pp. 9–41, Dec. 2017.
[30] L. Converso and S. Hocek, "Optical character recognition," *J. Vis. Impair. Blind.*, vol. 84, no. 10, pp. 507–509, Dec. 1990.
[31] M. K. Ugale and M. S. Joshi, "Improving optical character recognition for low resolution images," *IJCSN Int. J. Comput. Sci. Netw.*, vol. 6, no. 25, pp. 18–20, 2017.
[32] A. Farooq, A. K. Khan, and G. Raja, "Implementation of a speech based interface system for visually impaired persons," *Life Sci. J.*, vol. 10, no. SPEC. no. 9, pp. 398–400, 2013.
[33] R. M. Russell, H. Rasmussen, and A. H. Lichtenstein, "Modified Food Guide Pyramid for People over Seventy Years of Age," *J. Nutr.*, vol. 129, no. 3, pp. 751–753, Mar. 1999.
[34] *Protégé*. Accessed: Oct. 27, 2020. [Online]. Available: https://protege.stanford.edu/
[35] P. Jahoda. (2016). *MPAndroidChart*. Accessed: Oct. 27, 2020. [Online]. Available: https://www.webci.tation.org/5mt5TH6pg
[36] C. Patel, A. Patel, and D. Patel, "Optical character recognition by open source OCR tool tesseract: A case study," *Int. J. Comput. Appl.*, vol. 55, no. 10, pp. 50–56, Oct. 2012.
[37] R. Hyam, "Automated image sampling and classification can be used to explore perceived naturalness of urban spaces," *PLoS ONE*, vol. 12, no. 1, Jan. 2017, Art. no. e0169357.
[38] A. Bechmann, "Keeping it real: From faces and features to social values in deep learning algorithms on social media images," in *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, 2017, pp. 1–9. [Online]. Available: https://scholarspace.manoa.hawaii.edu/bitstream/10125/41372/1/paper0223.pdf
[39] R. Arief, A. Benny, T. Maulana, and H., "Automated extraction of large scale scanned document images using Google vision OCR in apache Hadoop environment," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 11, pp. 112–116, 2018.
[40] *UNESCO Thesaurus*. Accessed: Oct. 27, 2020. [Online]. Available: http://vocabularies.unesco.org/browser/thesaurus/es/?clang=en

BEATRIZ SAINZ-DE-ABAJO received the Ph.D. degree *(summa cum laude)* from the University of Cordoba, in 2009. She is currently an Associate Professor in Telecommunications Engineering with the University of Valladolid, Spain. Her fields of action are the development and evaluation of e-health systems, m-health, medicine 2.0., and cloud computing, focuses on topics related to electronic services for the information society. She belongs to the GTe Research Group, integrated within the UVa Recognized Research Group Information Society. Among the lines of research, the group works to develop innovative solutions in the field of health that help patients improve their quality of life and facilitate the work of health professionals.

JOSÉ MANUEL GARCÍA-ALONSO (Member, IEEE) received the Ph.D. degree in software engineering from the University of Extremadura, in 2014. He is currently an Associate Professor with the University of Extremadura, and a Co-Founder of Gloin, Software-Consulting Company. His research interests include software engineering, mobile computing, pervasive computing, eHealth, and gerontechnology.

JOSÉ JAVIER BERROCAL-OLMEDA (Member, IEEE) received the Ph.D. degree in computer science from the University of Extremadura, Spain, in 2014. In 2016, he obtained an Associate position at the University of Extremadura. He is currently a Co-Founder of company Gloin, which is a Software-Consulting Company. His main research interests include mobile computing, context awareness, pervasive systems, crowd sensing, the Internet of Things, and fog computing.

SERGIO LASO-MANGAS is currently pursuing the Ph.D. degree with the University of Extremadura, Spain. He is also working with the Computing and Telematics Systems Department. His research interests include mobile computing, pervasive systems, context-awareness, edge computing, and the Internet of Things.



ISABEL DE LA TORRE-DÍEZ is currently a Professor with the Department of Signal Theory and Communications, University of Valladolid, Spain. She is also a Leader of the GTe Research Group. She is the author or coauthor of more than 200 articles in SCI journals, peer-reviewed conferences proceedings, books, and international book chapters. She has coauthored 21 registered innovative software. She has been involved in more than 65 Program committees of international conferences until 2020. She has participated/coordinated in 37 funded European, national, and regional research projects.

A.6 OPPNets and rural areas: an opportunistic solution for remote communications

Authors: *Manuel Jesús-Azabal, Juan Luis Herrera, Sergio Laso and Jaime Galán-Jiménez.*

Publication type: *Journal paper.*

Journal: *Wireless Communications and Mobile Computing.*

Available online *10.1155/2021/8883501.*

Quality indicator: *JCR Q3.*

Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8883501, 11 pages
https://doi.org/10.1155/2021/8883501

WILEY | Hindawi

Research Article

OPPNets and Rural Areas: An Opportunistic Solution for Remote Communications

Manuel Jesús-Azabal, Juan Luis Herrera, Sergio Laso, and Jaime Galán-Jiménez

Department of Computing and Telematics Systems, University of Extremadura, Avda. de la Universidad S/N. 10003 - Caceres, Spain

Correspondence should be addressed to Manuel Jesús-Azabal; manuel@unex.es

Received 12 March 2020; Revised 20 March 2020; Accepted 12 December 2020; Published 15 January 2021

Academic Editor: Nathalie Mitton

Many rural areas along Spain do not have access to the Internet. Despite the huge spread of technology that has taken place during recent years, some rural districts and isolated villages have a lack of proper communication infrastructures. Moreover, these areas and the connected regions are notably experiencing a technological gap. As a consequence of this, the implementation of technological health solutions becomes impracticable in these zones where demographic conditions are especially particular. Thus, inhabitants over 65 suppose a large portion of such population, and many elderly people live alone at their homes. These circumstances also impact on local businesses which are widely related to the agricultural and livestock industry. Taking into account this situation, this paper proposes a solution based on an opportunistic network algorithm which enables the deployment of technological communication solutions for both elderly healthcare and livestock industrial activities in rural areas. This way, two applications are proposed: a presence detection platform for elderly people who live alone and an analytic performance measurement system for livestock. The algorithm is evaluated considering several simulations under multiple conditions, comparing the delivery probability, latency, and overhead outcomes with other well-known opportunistic routing algorithms. As a result, the proposed solution quadruples the delivery probability of Prophet, which presents the best results among the benchmark solutions and greatly reduces the overhead regarding other solutions such as Epidemic or Prophet. This way, the proposed approach provides a reliable mechanism for the data transmission in these scenarios.

1. Introduction

Communication technologies have experienced an exponential spread in the last years. The increasing interest of users on the Internet has unleashed a big competition between Internet Service Providers (ISPs) to provide coverage to as many areas as possible. Furthermore, the lowering prices of broadband access have allowed small businesses and homes to have access to the Internet. Nowadays, the percentage of the population in Spain with access to high-speed Internet connectivity has extraordinarily increased up to values of 81% in 2019 [1]. In Figure 1, the evolution of the percentage of homes with FTTH (Fiber To The Home) coverage in Spain is shown for recent years. These numbers highlight an average increase of 12.98% per year, which is expected to continue growing in the next decade. The constant expansion of technology has promoted the improvement of infrastructures and the rise of connected places. Nevertheless, these encouraging statistics are not the same in all regions. Territories like Madrid, Barcelona, or the Basque Country possess the highest numbers of connected homes. However, regions like Extremadura, Castile and León, or Galicia constitute the lowest rates and the biggest "shadow surfaces" [1]. These districts, where Internet infrastructures are minimal, are normally rural and isolated areas in which network operators are not interested in deploying their solutions due to the low benefits they acquire. These circumstances aggravate the problem. Furthermore, along the technology gap, rural areas face additional challenges like healthcare.

Demography in rural areas is commonly characterized by a significant percentage of people who are over 65. In this manner, rural regions face the challenge of providing

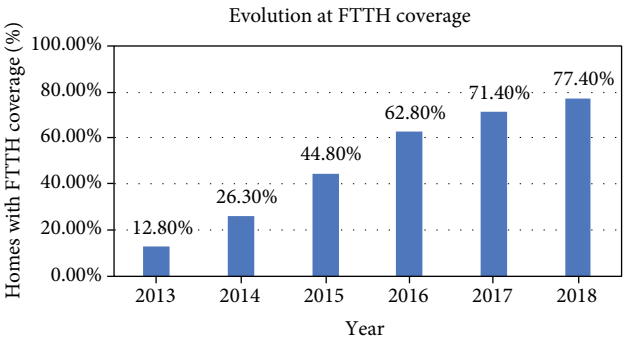


Figure 1: Evolution of FTTH coverage in Spain.

solutions to enhance the lives of elders in this challenging context. Loneliness is one of the main features in elderly people since a meaningful portion of them live alone in their homes. These circumstances can become a severe problem for health, especially in elderly people who are under clinical treatments. As a consequence of the lack of Internet infrastructures, technological solutions to help elderly people in their day-to-day routines cannot be applied, since monitoring systems and telemedicine platforms, which actively depend on the Internet, cannot be deployed. This way, these situations impact on the life of elderly people in rural areas.

On the other hand, technological constraints affect not only the elderly people but the local businesses too. Rural areas are filled with business activities like livestock or agriculture, and they are especially impacted by the conditions of the zone. Technological solutions for industry performance have become disruptive tools in production improvement. These systems provide mechanisms for monitoring and analyzing production processes, being implemented consistently in daily operation. Many of these systems provide a set of tools for production monitoring. In this manner, data processing becomes a relevant feature which requires specific communication infrastructures. In remote rural areas, where there is no Internet access, these procedures cannot be easily deployed.

Taking this context into account, rural areas that lack communication infrastructures face difficult challenges. The specific needs and the technological isolation motivate the spreading of alternative technologies which explore mechanisms for information transmission and system communication. Based on this idea, this paper proposes a DTN-based communication system for healthcare monitorization and livestock data transmission. The proposed platform brings a solution for elderly monitorization without an Internet connection. Thus, presence information about the elderly is transmitted to detect possible dangerous situations derivated from the physical inactivity. As a consequence, mortality rates in elderly people who live alone may decrease, taking into account that emergency detection becomes a crucial factor in healthcare. Moreover, since the platform provides a communication mechanism for isolated areas, local livestock businesses can transmit performance and production data. As a result, the technological gap is addressed while the system provides a healthcare solution for elderly people. The proposed approach exploits the use of SACAR OCVN [2], a routing algorithm based on the cooperation among nodes with different interests. In particular, this algorithm provides the mechanism to forward the information by adapting it depending on the type of data and on the interests of the receiver.

This article proposes the solution structuring the content as follows: firstly, Section 2 consists of a brief study of technological approaches to the communication challenges and rural areas. Section 3 describes our proposal, as well as the system model with the elements composing it. Section 4 shows the experimental results and the algorithm performance, by comparing the obtained results with the outcomes of other well-known opportunistic routing algorithms. Finally, Section 5 draws some conclusions and the next steps in this research line.

2. Related Work

During the last years, the research community has been actively working on solutions for communication in isolated areas. These ideas propose an alternative behaviour regarding the classical network operation, using the close proximity between devices and the movement between places. The most popular network technologies based on this idea are Delay Tolerant Networks (DTN). DTN technology is a communication paradigm based on the physical proximity of devices to reach information transmission [3]. This way, low energy interfaces are used to provide communication and broadcast information between the network components or gateways.

DTN are notably deployed in isolated and concrete areas where the Internet connection is not possible. However, the possibilities of DTN are numerous. Hostile-communication contexts like subaquatic communications, spatial transmission, or intermittent connections are some of these opportunities. Moreover, some communication applications do not require a constant information flow like wildlife tracking [4] or low-priority information traffic [5]. This way, DTN provides a suitable option for the deployment of applications and data exchange. Nevertheless, one of the most relevant applications of DTN is the reaction and deployment in natural disaster scenarios. Since the communication architecture is usually critically damaged, DTN provides a useful solution for simple data transmission. The application of DTN in isolated rural areas has led to multiple works in this field. In the early 2000s, many proposals based on DTN brought the connection to remote areas. Projects like Zhang et al. [6] or Pentland et al. [7] are good examples of technology applied in these isolated areas.

In Zhang et al. [6], the authors propose a communication system based on DTN which allows the population at Swedish Lapland to be connected to the Internet. This project addresses the requirement of providing a solution which adapts to the nomadic living conditions of the inhabitants. Following the same line, Pentland et al. [7] brought a low-cost communication infrastructure to remote areas using wireless communications and public transport mobility. Besides, the article includes the successful results of the deployment in isolated parts of India and Cambodia. Both

proposals are mainly oriented to provide inhabitants with communication services like email or access to documents.

As DTN is improving, recent works explore new prospects. Some projects like Berrocal et al. [8] bring new paradigms like the hybrid model conformed by SDN [9] and DTN, using the movement of the cars and intermediate nodes to relay information to gateways. Other ideas like Galán-Jiménez et al. [10] focus on providing Internet coverage to rural and low-income areas based on 5G architectures, using Unmanned Aerial Vehicles (UAVs) in remote zones, and exploring optimal energy consumption [11].

Of course, the purpose of these systems is significantly varied. Technological healthcare is especially relevant in rural areas but is arduous to match with the possibilities of DTN. Usually, the healthcare system requires constant communication and Internet connection, factors which are hard to provide using DTN. However, several projects already proposed eHealth solutions using this network paradigm. Works like Galán-Jiménez et al. [12] proposes the use of the DTN model to monitor patients. This specific scheme provides opportunistic communication between the patient and the doctor, using body sensors and smartphones. Moreover, Galán-Jiménez et al. [13] keeps a similar behaviour but benefiting from intermittent connections. As a result, this architecture brings viable ideas which propose a solution for isolation and healthcare in rural areas.

The work presented in this paper is different from the ones introduced above in the sense that technological healthcare is addressed through the integration of SACAR OCVN [2], an opportunistic routing algorithm based on the interests of the nodes and that it is applied in rural areas to provide health monitoring and communication infrastructure for local businesses. Therefore, the proposed model forwards two kinds of information: health info from the ageing inhabitants and production data from smart livestock. Thus, the solution faces the problem of healthcare monitoring through the presence detection of the elderly at home and provides a mechanism to deal with the technological gap in rural industries. In the next section, the behaviour of our proposed algorithm is explained.

3. OPPNets and Rural Areas

As previously introduced, this work proposes a solution for healthcare monitoring and data transmission in rural isolated areas. The possibilities of DTN enable the deployment of schemes which allow communication in places where the Internet is not available. Therefore, the provided network scheme benefits from vehicular traffic and physical encounters to transmit two kinds of information: presence data from elderly people's homes and sensor records from livestock. These two types of variables are quite relevant in the proposed scenario.

Presence detection in homes of the elderly is critically relevant. A large number of elderly people live alone at home in isolated rural areas, which becomes a severe health issue in the case of an emergency. This way, the detection of a dangerous situation becomes arduous without human supervision. Nevertheless, there are some relevant clues for possible health risks like interaction within the home environment. Taking this into account, the proposed scheme is based on obtaining data from sensors distributed around the house. These devices provide information about actions like door opening or lights operation. Hence, this presence information is transmitted through the network.

On the other hand, the gateways receive additional information: livestock analytic reports. The usual isolation of many areas of agricultural exploitation and many livestock farms becomes a significant technological gap compared with those businesses which can access the Internet. These industry holdings are especially suitable for the deployment of sensors and production analytics. Nevertheless, the lack of communication infrastructures affects these technological approaches. Thus, using the communication capability of the proposed network, reports from sensors are transmitted into the platform to end gateways.

The network behaviour follows the scheme provided in Figure 2, which identifies several components: (i) sender nodes, (ii) intermediate nodes, and (iii) gateways.

(i) Sender nodes refer to the homes of the elderly people and the smart livestock. These elements generate and transmit new information toward gateway nodes which are connected to the Internet. Thus, the messages are sent into the network in specified intervals of time. This way, the nodes send the messages to reachable intermediate nodes, which serve as data mules

(ii) Intermediate nodes are in charge of forwarding messages to the gateway. Therefore, cars and pedestrians receive, carry, and deliver the information as they move through the streets and roads. These nodes decide the information which they prefer to obtain and broadcast: presence info or livestock performance data. Additionally, there is another fundamental element in the scenario: *throwboxes*. These devices are placed on the main points of the road path and can store messages from any other intermediate node. Thus, throwboxes work as a "meeting point" for data and distribute it to other interested reachable nodes

(iii) Gateways are the destination elements of the messages generated by the sender nodes. They are connected to the Internet and are in charge of processing and transmitting information to the Cloud. As a result, the collected data can be externally processed, enabling the detection of anomalous patterns in the elderly activity and recognizing a possibly dangerous situation. In the same way, the performance information about livestock can be processed when it is finally delivered

The opportunistic behaviour of the network is mainly based on the routing algorithm we propose to perform distributed communication among the aforementioned elements. In this paper, the idea behind the SACAR OCVN algorithm defined in [2] is applied to the rural scenario in

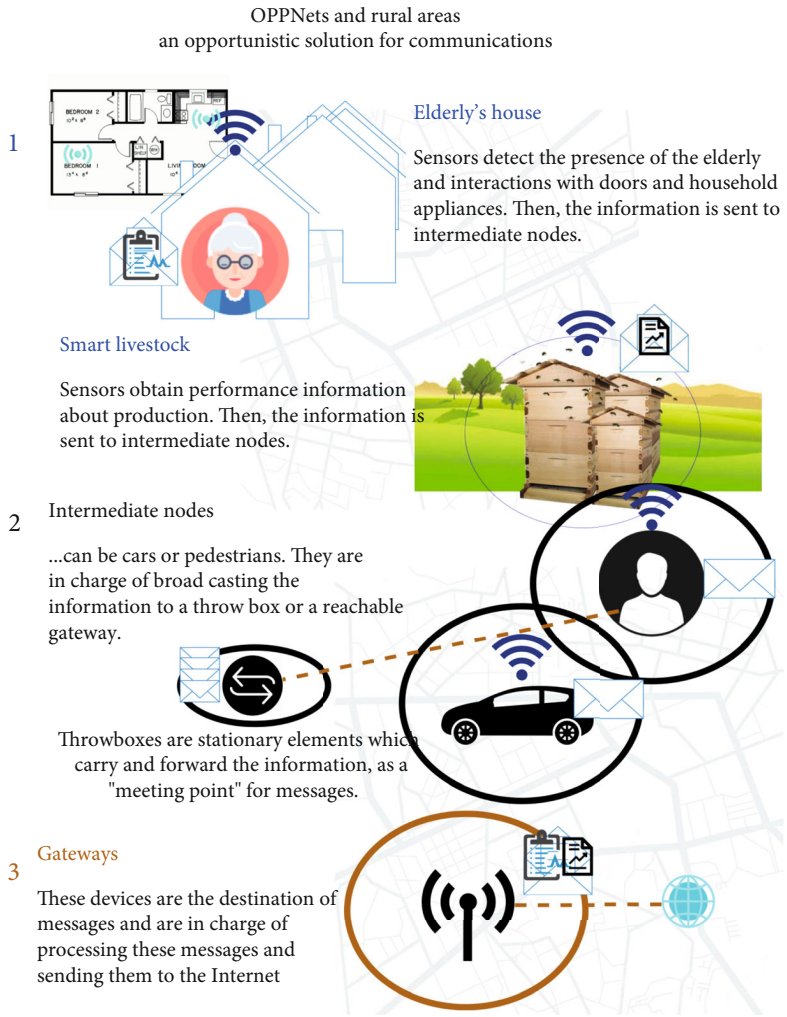


Figure 2: Working scheme of OPPNets in rural areas.

order to relay on cooperating nodes to successfully deliver the information from elders and livestock. The behaviour of the nodes in the scenario is based on the preferences of the nodes, following the Situational-Context model [14]. This paradigm proposes an automatic communication scheme for devices, using a virtual profile which defines the node's capabilities and preferences.

SACAR OCVN [2] applies the Situational-Context scheme as a means to automatise the nodes' encounter process. Thus, the nodes in the scenario have a virtual profile which identifies them as sender nodes, intermediate nodes, or gateways. The different roles available on the virtual profile are Goals and Skills: the first one defines the information preferences, while the Skills indicate the actions the node is able to perform. This model, once it is adapted to the scenario, is used to distinguish between sender elements (represented in Figure 3), carrying nodes (depicted in Figure 4), and end gateways (Figure 5).

The communication takes place when two nodes encounter each other. This way, by using low-energy technologies, the information is exchanged within the corresponding range. Thus, the communication process follows three main steps: (1) devices encounter each other, (2) virtual profiles

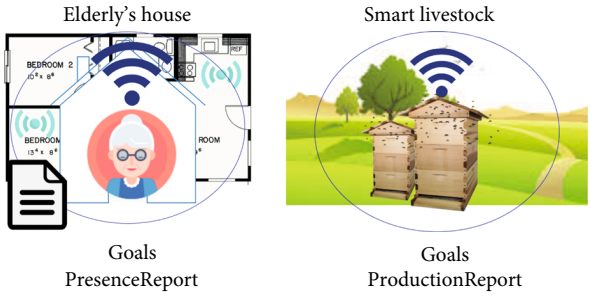


Figure 3: Virtual profile of sender nodes.

are gathered, and (3) information is exchanged. All these steps are shown in Figure 6. The proposed scenario raises an opportunistic context where communication provides an improvement in the day-to-day routines of local inhabitants. The idea has been executed as a simulation which has tested the algorithm under different situations, trying to define the most appropriate context for interactions. The next section details the simulation process and the parameters that have been tuned.

Figure 4: Virtual profile of intermediate nodes.

Figure 5: Virtual profile of gateway nodes.

4. Experimental Results

The result section is organized as follows: firstly, the selected scenario is described. Then, the results are analyzed focusing on two aspects: (i) performance analysis of SACAR OCVN algorithm on the scenario and (ii) comparison with other relevant opportunistic routing solutions.

4.1. Simulated Scenario. Simulations have been carried out using The ONE simulator [15]. This tool provides a development environment for DTNs, allowing the specification of a detailed scenario, movements, and protocols. Thus, the performance of the routing algorithm can be obtained through execution reports. In this work, the simulated scenario recreates a rural area with low connectivity, which reflects a village community and the surrounding roads. Figure 7 represents the simulation map.

The simulation process follows an execution scheme which varies the context inputs and node behaviour. Table 1 summarizes the input configuration. The number of total nodes is $N = 146$, where $N_s = 3$ is the number of destination nodes, $N_i = 66$ is the number of intermediate nodes, and $N_g = 80$ are referred to sender nodes. Destination locations have been established far away from the rural village in order to analyze the effectiveness of the communication. Intermediate nodes are distributed along paths in an area of $A = 81000\,\text{m}^2$ with different movements patterns, P, which specify how the nodes move. The type of interest settings is composed of the Skill set, S, and the Goal set, G. The first group keeps the capabilities of the nodes, in this case, store-and-carry presence information or performance data. Also, the Skill of transmitting messages to the Internet is possible. On the other hand, the Goal set defines, in this case, the kind of information the node generates. The maximum number of Goals in the nodes is $g_n = 2$ and $s_n = 2$ in the case of Skills. Simulation duration is set to $T = 28000$ s.

Message generation interval, ω, is in charge of establishing the waiting time before the sender nodes generate new information. It is a key parameter in the simulations since the number of created messages depends actively on it. In this scenario, three times are considered: $\omega = \{900, 1800, 3600\}$s. These intervals adapt the message generation to real values since presence data requires a fluent transmission. In the next section, simulation results are analysed, as well as compared with other well-known opportunistic routing algorithms under different contexts.

4.2. Parameter Setting. Four main outcomes are analyzed after running the simulations: (1) delivery probability (d_{prob}), (2) overhead ratio (θ), (3) average latency (τ), and (4) average number of hops (γ):

(i) d_{prob} is the percentage of messages which reached the destination node. This way, it is a critical value which reflects the success of communication and forwarding. This value is calculated as the relation between the number of messages originally sent, d_{sent}, and the number of messages received at destinations, d_{received}

(ii) θ is the relation between duplicated messages and received messages. It reflects the use of the network

(iii) τ reflects the average time needed to receive a message once it is sent

(iv) γ represents the average number of intermediate nodes needed to reach the destination

These results are obtained when the simulation process is finished. This way, the scenario receives two main inputs which vary the final results. These two variables are interval message generation (ω) and interest distribution at intermediate nodes.

(1) Interval message generation, ω, specifies the waiting time before new messages are created. Thus, it defines a variable which plays an important role in simulation. The considered times are 900, 1800, and 3600 seconds

(2) Since SACAR OCVN is based on node interests, three different distributions of interests are considered: (i) the percentage of nodes only interested in carrying elderly presence information (I_{elderly}), (ii) nodes only interested in carrying industrial production information (e.g., from livestock) (I_{industry}), and (iii) nodes interested in carrying both types of information (I_{hybrid}). Moreover, there are also nodes which are not interested in carrying any type of information (I_{empty}). Table 2 shows the selected values for the three considered scenarios

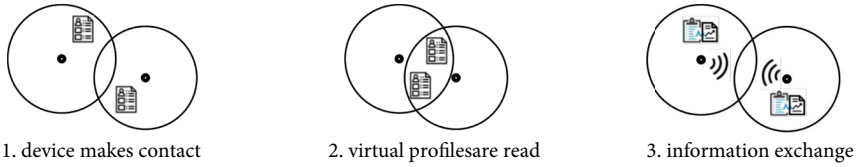


Figure 6: Detailed communication process.

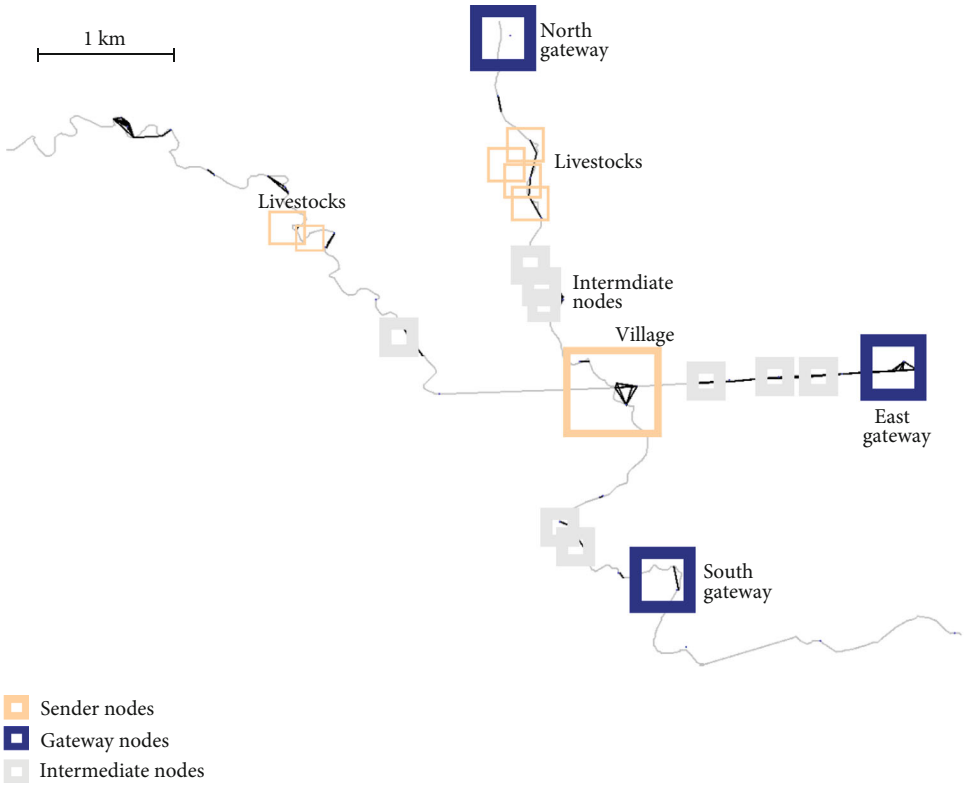


Figure 7: Rural scenario simulated in The ONE.

Table 1: Parameter setting for the rural scenario.

Parameter	Value
A	81000 [m^2]
N	146
N_s	3
N_i	66
N_g	80
S	{SendPresenceInfo, SendproductionInfo, StoragePresenceInfo, StorageProductionInfo}
G	{PresenceReport, ProductionReport}
s_n	2
g_n	2
P	{StationaryMovement for N_s, StationaryMovement for N_g, MapRouteMovement for N_i with stationary time periods in the range of t = [300,500] s.}
P	
T	28000 [s]
ω	{900,1800,3600} [s]

Table 2: Percentage of nodes contemplated for scenario simulations.

Scenario	I_{elderly} (%)	I_{industry} (%)	I_{hybrid} (%)	I_{empty} (%)
Elderly interest scenario	50	20	20	10
Factory interest scenario	20	50	20	10
Hybrid interest scenario	35	35	20	10

5. SACAR OCVN Performance Analysis

The proposed algorithm is executed on each scenario distribution, varying the message generation interval, ω. As Figure 8 reflects, the delivery probability (d_{prob}) obtained by SACAR OCVN is very relevant. Since it is a critical value which captures the success at message transmission, a high percentage is needed. This way, the algorithm experiences the lowest rates when the message generation interval is low. On the other hand, the best results are obtained when

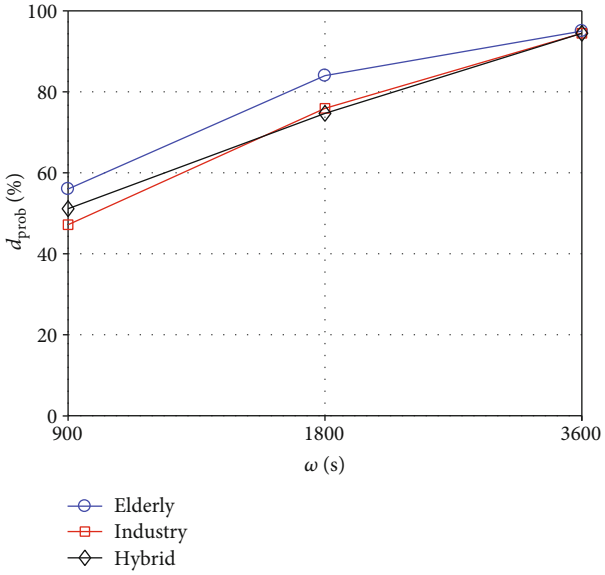


Figure 8: Delivery probability in SACAR OCVN.

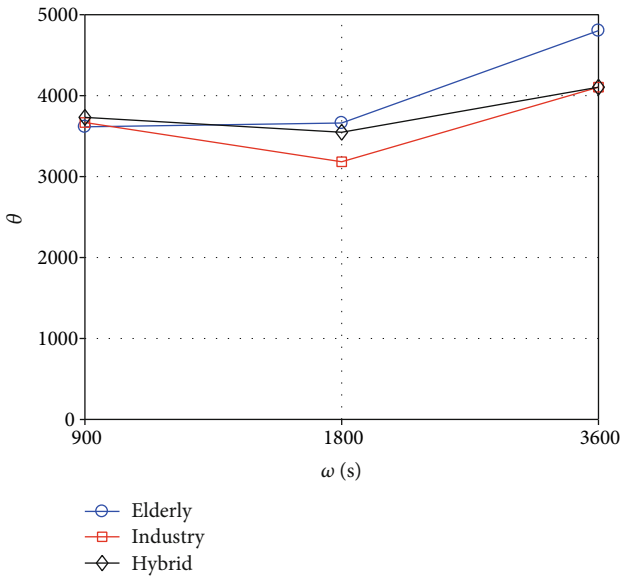


Figure 9: Overhead ratio in SACAR OCVN.

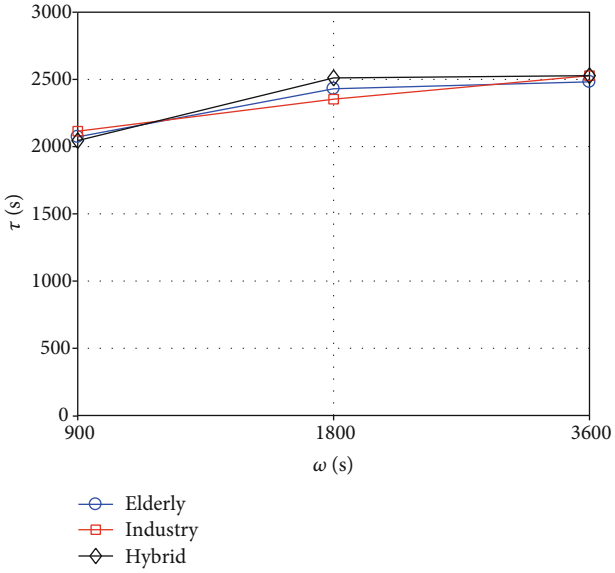


Figure 10: Average latency in SACAR OCVN.

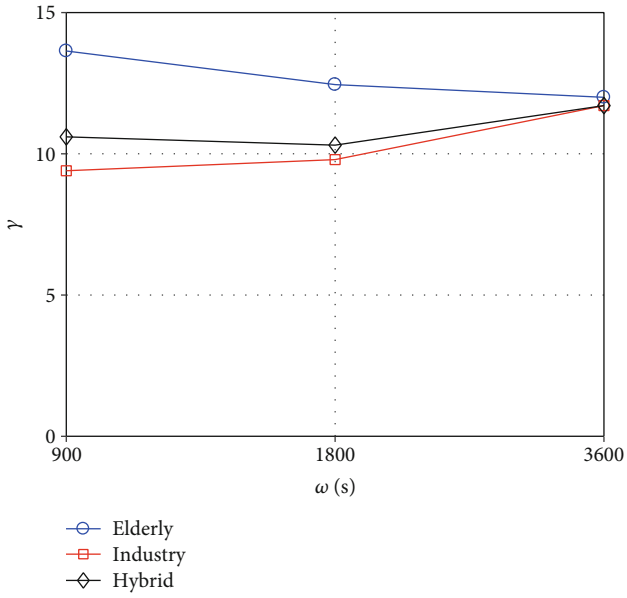


Figure 11: Average hops in SACAR OCVN.

the value is increased, reaching $d_{prob} = 0.945$ when $\omega = 3600$ s . This is essentially due to low interval values which involve a larger number of messages to be transmitted.

The overhead ratio results are included in Figure 9. This value is a suitable parameter to know the practical usage of the bandwidth in the network. SACAR OCVN experiences an overhead increase when the predominant interest is the information from elderly people. On the other hand, when industry interest is predominant, the value of θ decreases. As a result, the algorithm experiences an average result when the scenario is mixed.

Since the transmitted information is elderly-care related, the average latency (τ) is critical to minimize the needed time to reach the destination. As Figure 10 shows, SACAR OCVN presents very similar results in each scenario with an average latency of around $\tau = 2500$ s.

Finally, the average number of hops, γ, is the median number of intermediate nodes needed by the messages to reach the destination. The latency value is usually related to the hop rate since the encounters and information transmission at hops improve possibilities of delivery. This way, as Figure 11 shows, SACAR OCVN average hop outcomes follow an inversely proportional trend w.r.t. latency values.

As a conclusion, SACAR OCVN results are positive. The delivery probability in $\omega = 3600$ s is $d_{prob} = 0.945$, guaranteeing a high rate at message broadcasting. Besides, the message generation interval which provides a better performance is $\omega = 3600$ s. This successful result means that the algorithm

provides a good communication mechanism for the exposed scenario. As a consequence, the data of presence from the elderly's homes is fluently transmitted through the network. Thus, the system allows the remote detection of possible dangerous situations, derivated from the inactivity of the elderly. Besides, the local industries can transmit performance data from the exploitation.

Once SACAR OCVN best results are obtained, they are compared under the same context conditions with the rest of the DTN algorithms included in The ONE. Furthermore, this second stage of result analysis is addressed.

6. Comparison with Other Opportunistic Routing Solutions

SACAR OCVN experiences better results when the message generation interval is $\omega = 3600$ s. This way, using the same scenario conditions, in this section, we compare these values with the ones obtained when running other well-known opportunistic routing algorithms in The ONE. Note that the analysis uses the same set of parameters described in the previous section. Next, benchmark algorithms are briefly described. DirectDeliveryRouter [16] works based on straight delivery between the sender and the receiver node. Thus, intermediate elements are ignored. EpidemicRouter [17], on the other hand, belongs to the flood algorithm family. Thus, the behaviour is based on duplicating the messages with every encountered node. MaxPropRouter [18] uses the previous node encounters to define the most appropriate path to the destination. ProphetRouter [19] works based on a probabilistic scheme while SprayAndWaitRouter [20] replies to messages by creating copies which can be specified by the user.

The executions take place with $\omega = 3600$ s for the different interest distributions over the same scenario described in Figure 7. Delivery probability is the most significant parameter in performance comparison. As Figure 12 shows, the best delivery probability is $d_{\text{prob}} = 0.95$ and belongs to the SACAR OCVN execution. The other algorithms provide a low delivery rate in the three scenarios.

Overhead ratio results are reflected in Figure 13. DirectDeliveryRouter is based on communicating sender nodes and destination, without using intermediate elements. Thus, message traffic is low. Besides, the lowest values belong to SprayAndWaitRouter, which provides the communication process with better use of the bandwidth than EpidemicRouter and ProphetRouter. SACAR OCVN, in turn, keeps close values to SprayAndWait.

Figure 14 shows the average latency in executions. SACAR OCVN is the algorithm which has the highest latency value, mainly because of the largely supported message traffic, as well as the low hop rates. On the other hand, EpidemicRouter and ProphetRouter keep the results below these numbers. Besides, SprayAndWaitRouter provides the lowest rate.

The average number of hops in each of the algorithms is presented in Figure 15. DirectDeliveryRouter bases its behaviour on senders delivering messages straight to the destination; thus, it is always one hop. On the other hand, SACAR OCVN keeps an average hop number quite low regarding EpidemicRouter and ProphetRouter results, which represent the highest in the hybrid interest scenario. Again, SprayAndWaitRouter provides a good performance and draws on few hops to reach the destination.

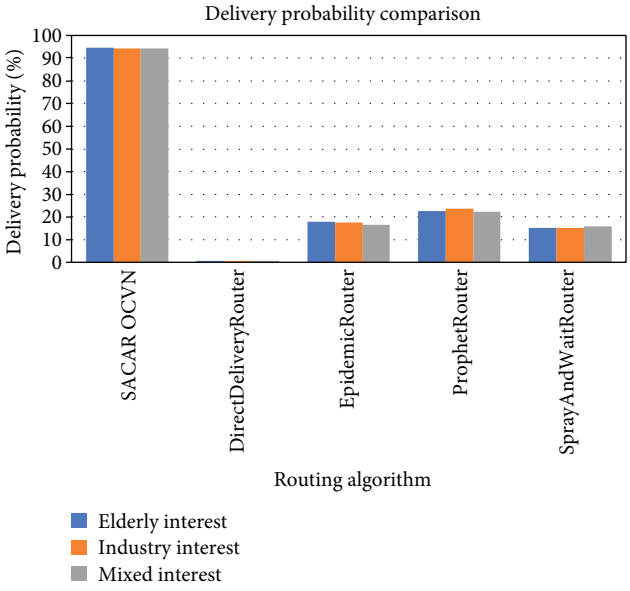


Figure 12: Delivery probability comparison.

The comparison process clearly shows the performance of SACAR OCVN compared with the opportunistic routing algorithms included in The ONE. The high delivery probability value guarantees the applicability of the solution in the described scenario, becoming a powerful tool in the implementation of the project. It is important to remark that SprayAndWaitRouter keeps good results at θ, τ, and γ, but d_{prob} is too low to provide a reliable communication. The possibilities of the router are relevant and follow a promising work line.

7. Discussion

Rural areas represent a big percentage of the population over 65. Only in Spain, around 30% of inhabitants in small villages are older adults. These demographic conditions become a challenge for health, since issues like personal assistance, solitude, and isolation play a big role in the daily routines of seniors. Technology has become a very significant ally for healthcare, providing mechanisms and tools such as eHealth, remote monitoring, and smart systems which ease communication and quick emergency detection. However, many of these rural spaces lack Internet connection, where no infrastructures to deploy eHealth solutions can be exploited.

In this paper, a response to this context is provided. Making use of alternative communication mechanisms like opportunistic networks, a reliable system for remote areas is introduced. Thus, the proposal supplies villages with an architecture for message transmission which allows the detection of possible emergencies of seniors who live alone

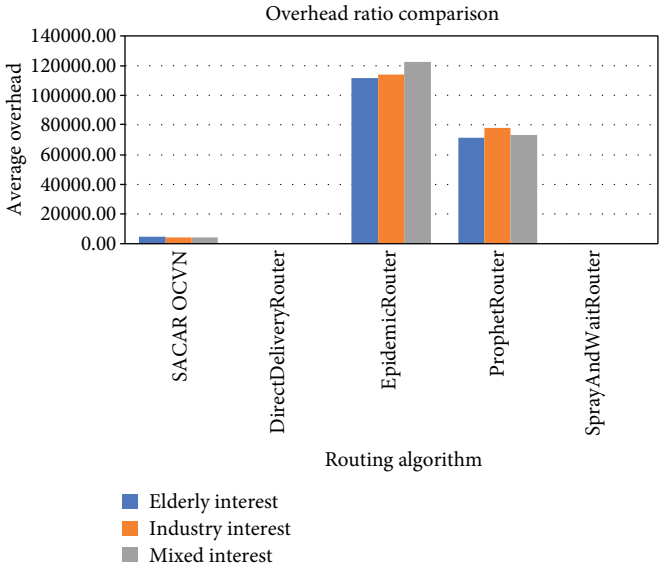


Figure 13: Overhead ratio comparison.

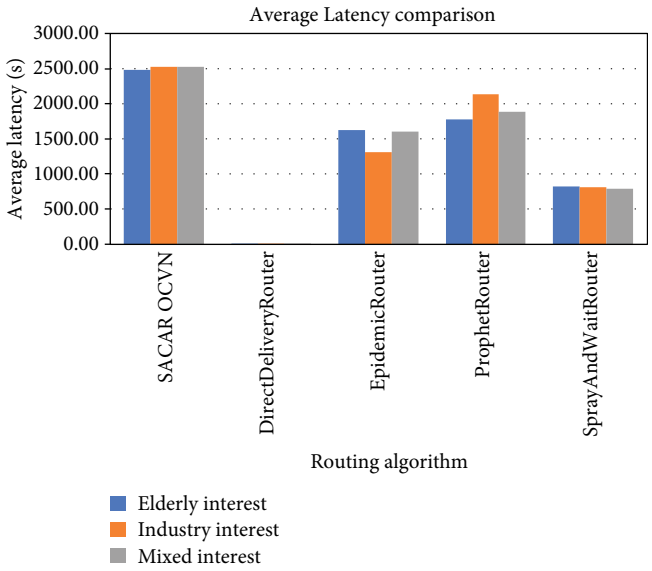


Figure 14: Average latency comparison.

while local exploitations are also able to transmit performance reports.

On the one hand, information about the presence habits of seniors who live alone is transmitted. This way, several sensors are installed along the houses, recording the presence of the elderly and sending such information toward the opportunistic network. The possibility of determining the presence patterns of elderly people allows the detection of potentially risky situations that can happen in the solitude of the home. As a result, mortality risks in the elderly who live alone are reduced by exploiting the detection of anomalous behaviour.

On the other hand, the local exploitations in isolated rural areas face the challenge of the lack of Internet infrastructure. The technological gap reduces the competitive advantages compared to the connected industries, and therefore, their maximum performance capabilities cannot be reached. In order to tackle this problem, the proposed solution exploits the use of sensors at the exploitation place like livestocks or hives, playing roles like weight measurement, animal tracking, or food consumption.

The data collected by sensors is broadcasted toward the destination making use of BLE technology and intermediate nodes. These nodes, which serve as mules, can be different types of devices like cars, smartphones, smartwatches, or throwboxes. These lasts are devices installed on the roads, fed by solar panels, which store and forward the information to the devices in their range.

As a result, the proposed solution in this paper is aimed at improving the quality of life in remote areas.

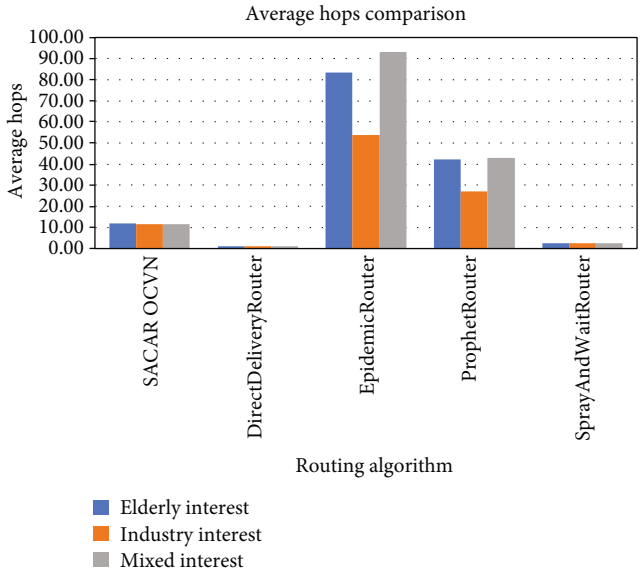


Figure 15: Average hops comparison.

Communication possibilities in these scenarios go further and provide an eHealth architecture for eldercare and local business improvement. In this way, the opportunities of deploying a prototype in a real isolated zone could bring a brand new autonomous model for well-being, alliance, and cooperation.

8. Conclusions and Future Works

Many isolated rural areas do not have an Internet connection. These limitations are often induced by the remote situation and the geographical issues, as well as the lack of interest from telecommunication companies. Thus, inhabitants cannot keep connected in a world where the Internet expansion is exponentially growing. Furthermore, the population in rural areas is usually over 65 years old. Thus, isolated regions face the difficult challenge of providing services and health attention, especially when these elderly people live alone. However, this context can be improved by using technological solutions. Digital elder healthcare is an active work line where many researchers and companies are working on. Nevertheless, the lack of an Internet infrastructure prevents the deployment of many of these solutions. Moreover, the technological gap also affects the local industry and livestock, which are not generally able to implement solutions to monitor and improve performance.

In order to face these challenges, DTN solutions become a suitable technology to provide connectivity in remote places. This paper proposes a DTN routing algorithm based on information interests which transfer elderly presence information and industry performance data to gateway points.

Our proposed algorithm has been simulated on a realistic isolated region of Spain. The obtained results are positive and guarantee a high rate of successful message delivery under different conditions. Furthermore, the outcomes of our proposed algorithm are also compared with other well-known opportunistic routing solutions, outperforming them. Therefore, the proposed scheme brings a solution to the technological gap in isolated rural areas, enabling the monitoring of elders' activity and providing a reliable communication system to improve the performance of livestock industries.

The authors are working on providing the proposed algorithm with a smart behaviour through a machine learning model. Moreover, trajectory prediction in nodes' movement presents other key aspects to be explored.

Data Availability

Data are available on request. Research results are available on request. Thus, in order to be provided with the data report, please, write to Manuel Jesús-Azabal (manuel@unex.es).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by 4IE+ project (0499_4IE_PLUS_4_E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by RTI2018-094591-B-I00 Project (MCI/AEI/FEDER,UE), by the Department of Economy, Science and Digital Agenda of the Government of Extremadura (GR18112, IB18030), and by the European Regional Development Fund.

References

[1] J. Burgess, B. Gallagher, D. D. Jensen, and B. N. Levine, "Maxprop: routing for vehicle-based disruption-tolerant networks," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pp. 1–11, Barcelona, Catalunya, Spain, 2006.

[2] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev*, vol. 7, no. 3, pp. 19-20, 2003.

[3] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[4] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-Tolerant Networking, WDTN*, vol. 5, pp. 252–259, New York, NY, USA, 2005, ACM..

[5] A. Vahdat and D. Becker, *Epidemic Routing for Partially-Connected Ad Hoc Networks*, Duke University Durham, Technical report, 2000.

[6] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, "Hardware design experiences in zebranet," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems, SenSys*, vol. 4, pp. 227–238, New York, NY, USA, 2004, ACM.

[7] L. Amorosi, L. Chiaraviglio, and J. Galán-Jiménez, "Optimal energy management of UAV-based cellular networks powered by solar panels and batteries: formulation and solutions," *IEEE Access*, vol. 7, pp. 53698–53717, 2019.

[8] J. Berrocal, J. Garcia-Alonso, C. Canal, and J. M. Murillo, "Situational-context: a unified view of everything involved at a particular situation," in *International Conference on Web Engineering*, pp. 476–483, Springer, 2016.

[9] E. Borgia, R. Bruno, and A. Passarella, "Making opportunistic networks in IoT environments CCN-ready: a performance evaluation of the mobccn protocol," *Computer Communications*, vol. 123, pp. 81–96, 2018.

[10] Ministerio de Asuntos Económicos y Transformación Digital (es), "El 81% de la población española dispone de cobertura de Internet a más de 100 Mbps," October 2020 https://cutt.ly/YgmAT2N.

[11] A. Doria, M. Uden, and D. P. Pandey, "Providing connectivity to the saami nomadic community," *International Conference on Open Collaborative Design for Sustainable Innovation*, vol. 1, 2002.

[12] J. Galán-Jiménez, J. Berrocal, J. Garcia-Alonso, and M. J. Azabal, "A novel routing scheme for creating opportunistic context-virtual networks in IoT scenarios," *Sensors*, vol. 19, no. 8, p. 1875, 2019.

[13] J. Galán-Jiménez, L. Chiaraviglio, L. Amorosi, and N. Blefari-Melazzi, "Multi-period mission planning of UAVs for 5G coverage in rural areas: a heuristic approach," in *2018 9th International Conference on the Network of the Future (NOF)*, pp. 52–59, IEEE, Poznań, Poland, 2018.

[14] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," in *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pp. 96–107, San Jose, California, USA, 2002.

[15] E. Max-Onakpoya, A. Jacobs, and C. E. Baker, "An opportunistic mhealth architecture for remote patient monitoring," in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, pp. 169–169, Santa Cruz, CA, USA, 2019.

[16] E. Max-Onakpoya, S. Madamori, and C. E. Baker, "Utilizing opportunistic social networks for remote patient monitoring in rural areas," in *Proceedings of the 1st ACM International Workshop on Technology Enablers and Innovative Applications for Smart Cities and Communities*, pp. 46–49, New York, NY, USA, 2019.

[17] A. Pentland, R. Fletcher, and A. Hasson, "Daknet: rethinking connectivity in developing nations," *Computer*, vol. 37, no. 1, pp. 78–83, 2004.

[18] K. Shin, K. Kim, and S. Kim, "Traffic management strategy for delay-tolerant networks," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1762–1770, 2012.

[19] Y. Teranishi, T. Kimata, E. Kawai, and H. Harai, "Hybrid cellular-DTN for vehicle volume data collection in rural areas," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 276–284, Milwaukee, Wisconsin, USA, 2019.

[20] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE simulator for DTN protocol evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009ICST.

A.7 Virtual environment for evaluating the QoS of distributed mobile applications

Authors: *Sergio Laso, Javier Berrocal, Pablo Fernández, Antonio Ruiz-Cortés and Juan M Murillo.*

Publication type: *Conference paper (Demo and Artifact paper).*

Conference: *International Conference on Pervasive Computing and Communications (PERCOM), Kassel, Germany. 2021.*

Available online: *Demo: 10.1109/PerComWorkshops51409.2021.9431133. Artifact: 10.1109/PerComWorkshops51409.2021.9431135.*

2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops) | 978-1-6654-0424-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/PERCOMWORKSHOPS51409.2021.9431133

Virtual Environment for Evaluating the QoS of Distributed Mobile Applications

Sergio Laso*, Javier Berrocal*, Pablo Fernández†, Antonio Ruiz-Cortés† and Juan M. Murillo*
*University of Extremadura
Cáceres, Spain
Email:{slasom, jberolm, juanmamu}@unex.es
†I3US Institute, SCORE Lab. Universidad de Sevilla
Sevilla, Spain
Email:{pablofm, aruiz}@us.es

***Abstract*—The increasing capabilities of end devices has led to a wider distribution of the computation and the massive deployment of distributed mobile applications. The success of these applications is highly dependent on the Quality of Service they provide. This quality is especially difficult to assess due to the large number of entities involved and their heterogeneity. Current tools are usually focused on evaluating the QoS provided by a single entity. Nevertheless, the QoS of distributed applications not only depend on the QoS of each entity, the interactions among entities has also to be evaluated. Therefore, new techniques are required to perform a comprehensive evaluation of the expected QoS of these applications before their production deployment. This paper presents a framework, called Perses, for launching virtual environments to simulate and test the execution of distributed mobile applications. This simulation provides results of the QoS achieved. Moreover, the framework has been integrated into a DevOps methodology in order to automate its execution.**

***Video showcase*— https://youtu.be/wpIApe_sPFE**

***Index Terms*—Distributed Mobile Applications; Quality of Service; DevOps; Virtual Environment**

I. Introduction

The success or failure of any mobile application depends on many dimensions such as advertising, virality or the Quality of Service (QoS) provided [1]. The QoS is a key dimension that can be controlled and evaluated during the development and operation of the application [2]. A poor quality will lead users to quickly reject the application.

These applications have been usually developed using a client-server architectural style to achieve a maximum QoS. The most demanding computing components were usually deployed in cloud environments due to their large computing and storage capabilities. The basic components were deployed at the client-side, especially the user interface, as their computing requirements are usually low and can be executed by almost any mobile device with a minimum set of functionalities.

During the last few years, mobile devices computing and storage capabilities have significantly increased [3], leading to the emergence of new paradigms focused on exploiting them, such as Human Microservices [4], Mist Computing [5] or Federated Learning [6]. They allow developers to on-load some computing functionalities on mobile devices to further improve the QoS and be more competitive at the enterprise level. For environment, such as Healthcare or Industry 4.0, with stringent QoS requirements (such as response time or latency) this on-load of functionalities is crucial [7]. These paradigms allow developers to locally store and compute the sensed information in order to be consumed by local applications or nearby devices.

In client-server applications, QoS-sensitive functionalities are deployed in cloud environments, in which cloud providers can guarantee a minimum quality of the infrastructure and also scale it up if more resources are required. In such a context, QoS is mainly evaluated at the server-side, whilst the tests at the mobile-side are typically focused on the user experience (functionality and adaptability of graphic elements, navigation through different screens, etc.).

In distributed applications, both the cloud and the mobile-side have computing and storage components. Moreover, QoS-sensitive functionalities can be deployed at both sides. Therefore, the QoS evaluation must combine the integrated assessment of the server-side, the mobile-side, and the interaction among devices. Only evaluating the three dimensions in an integrated would allow one to predict with adequate accuracy the expected QoS. In addition, these are usually highly heterogeneous environments (different mobile devices, with different capabilities, etc.), being even more difficult to predict the expected QoS. Therefore, technologies and methods are required to evaluate the QoS of distributed applications in heterogeneous environments.

Currently, there are different environments that can be used to evaluate the behaviour of non-distributed mobile applications in heterogeneous environments, such as AWS Device Farm [8] and Azure App Center Test [9]. They allow developers to create customized environments composed of devices with different hardware, versions of the operating system, configuration, etc. However, these platforms focus on evaluating and launching user interface tests (using Appium,[1] Espresso,[2] etc.). Therefore, they are not designed to measure the QoS attributes integrating the cloud and the mobile-side,

[1]http://appium.io/
[2]https://developer.android.com/training/testing/espresso

and their interactions.

Likewise, there are some research efforts towards the evaluation of the QoS of distributed applications. In [10] the authors propose a framework for the development and evaluation of distributed systems based on components for heterogeneous scenarios, taking into account mobile and fixed networks. However, this evaluation is limited to applications developed using the proposed framework. Therefore, approaches are needed that can also evaluate any application that can be deployed in the targeted devices. In [11], the authors propose a system for testing the usability and performance of Android mobile applications with Virtual Reality. This system simulates different network and mobility behaviours in order to evaluate applications in close to real environments. However, this approach is focused on manually testing the application using a single device, which hinder the evaluation of distributed applications deployed on heterogeneous devices. Furthermore, the tests carried out are focused on user experience.

In this paper, we present a framework, called Perses [12], to evaluate the QoS of distributed mobile applications. To that end, it allows the definition and deployment of customized virtual environments, with multiple heterogeneous virtual devices, in which developers can simulate the deployment of distributed mobile application and launch performance tests to evaluate different QoS attributes, namely computing time, response time or latency.

Moreover, this framework has been fully integrated into a DevOps methodology [13] integrating the virtual environment deployment and the tests execution in a continuous integration pipeline, reducing the effort of evaluating the QoS before deploying the application into production.

The rest of the paper is structured as follows. Section II explains the characteristics of Perses. Subsection II-A details its architecture and Subsection II-B describes a demo use case. Finally, the conclusions are presented in Section III.

II. Perses

In distributed mobile applications, the holistic evaluation of the QoS is crucial to predict their success. In these applications, some components are deployed on the server and others on the mobile-side. If the QoS is evaluated independently in each entity and, then, aggregated, the results can be far from reality. The interactions among entities, and devices heterogeneity must also be taken into account.

Perses is a framework that allows developers to easily deploy a virtual environment with multiple heterogeneous virtual devices to evaluate the QoS of a distributed application. To use it, one only has to define a *configuration file* where the desired QoS and the characteristics of the environment where it is to be tested are indicated.

Perses is fully integrated into a DevOps methodology, being able to automate all the different steps that should be executed during the evaluation process, from the deployment of the environment to the collection and checking of the results obtained to determine whether or not the application achieves the desired quality. This is an important feature for development companies since they can easily integrate Perses in their continuous integration pipeline.

Perses is fully scalable, allowing one to evaluate applications with a large number of virtualised devices. It also allows customizing their characteristics such as hardware, OS, etc, being able to easily test different configurations.

A. Architecture

Perses is a framework based on different tools to facilitate the creation and deployment of the virtual environments with heterogeneous virtual mobile devices simulating a real deployment environment. This section explains the different modules of this framework, how they have been developed using existing tools, and how they have been integrated.

Figure 1 shows a general diagram of Perses's architecture. Perses consists of two components: *Perses Launcher*, focused on the definition and configuration of the virtual environment, and *Perses Virtual Environment*, responsible for creating the heterogeneous environment and executing the defined tests. Each component has different modules that are explained below:

- **Setup:** this module allows developers to create a *configuration file* that is used for defining the characteristics of the virtual environment to be deployed, the tests to be launched and the desired QoS attributes to be evaluated during the test execution.
- **Deployment:** this module takes as input the *configuration file* and is responsible for creating and deploying the whole virtual environment. To that end, an abstraction layer has been defined that encapsulates Terraform [14] –a framework with a high-level language that can be used to define the deployment infrastructure of an application for cloud providers. This module, in addition to deploying the cloud infrastructure, orchestrates the virtual environment obtaining and installing the necessary resources, creating the virtual mobile devices and deploying the distributed mobile application.
- **Tests Execution:** this module is in charge of launching and analysing the tests defined in the *configuration file*. Two different kinds of tests can be executed: performance and user interface tests. The performance tests evaluates the QoS of the application. To that end, APIPecker [15] – a simple API performance tester in which different attributes can be defined (concurrent users, iterations and delay) – is used. For the user interface tests, Perses allows developers to execute UI tests developed with Espresso. After launching the tests, this module collects, analyses, and aggregates the system log of every virtual devices and the general performance metrics to determine if the desired QoS is achieved.
- **CI Manager:** this module is in charge of the integration with DevOps. It automates the whole deployment process of Perses and the execution of the different modules. This automation is carried out through a workflow defined with GitHub Actions [16]. Once the process is completed, the test results are provided to the developers.

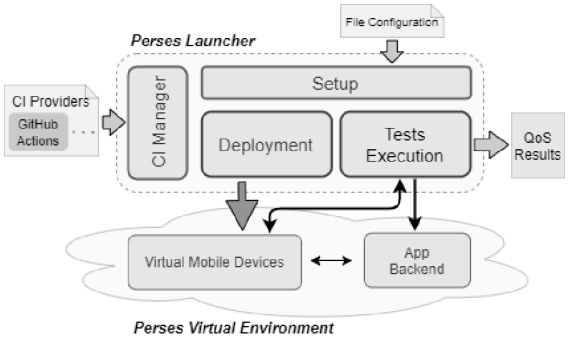


Fig. 1. General diagram of Perses.

B. Demo Use Case

In this demonstration, we will show attendees how the Perses framework works by evaluating the QoS of a distributed mobile application. For this purpose, we will evaluate an application to help curb the Covid-19 pandemic. This application provides the contagion risk of a person.

The application is made up of a server and a mobile-side. The server-side is the access point for users to obtain the risk percentage and, therefore, is also responsible for calculating this risk. The mobile-side is in charge of saving the traces of each user and providing on-demand to the cloud the places/regions in which they have been on.

To know the contagion risk, the user provides the regions where he has been on during the last three days to the server. Then, the server compares them with the regions visited by recently diagnosed positive users in order to calculate the infection risk.

To make the application suitable for the production deployment, an average *response time* of no more than two seconds is required. For this purpose, Perses will create a virtual environment with a set of devices simulating a real situation to evaluate the QoS. The environment will be composed of virtual mobile devices that already have a location history stored. Some of them will act as users with Covid-19 to evaluate the most adverse cases where a user has moved through areas where there have been several positive people.

III. Conclusion

QoS is one of the key dimensions for the success or failure of any application. In recent years, the capabilities of end devices have increased considerably, leading to the emergence of new paradigms focused on the exploitation of these capabilities. This gives rise to distributed applications where both the cloud and the mobile-side can have computing and storage components with the aim of further improving the QoS. For environments with strict QoS attributes this distribution of functionalities is crucial. Moreover, with the advance of IoT, these applications are going to be more and more common. Nevertheless, this quality depends on different entities and their interactions, which should be carefully analyzed.

In this paper, we presented a framework for the deployment of a customized virtual environment for assessing the QoS of distributed mobile applications. We currently work on analysing the impact of the use of this framework. We are also working on using Perses to evaluate different architectural styles and which one achieves a better QoS.

Acknowledgment

This work has been partially funded by the projects RTI2018-101204-B-C21 and TIN2015-70560-R (MCI/AEI/FEDER,UE), the 4IE+ Project (0499-4IE-PLUS-4-E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the RCIS network (TIN2016-81978-REDT), by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), by the Government of Andalusian (US-1264651) and by the European Regional Development Fund.

References

[1] H. J. Kim, D. H. Lee, J. M. Lee, K. H. Lee, W. Lyu, and S. G. Choi, "The qoe evaluation method through the qos-qoe correlation model," in *2008 Fourth International Conference on Networked Computing and Advanced Information Management*, vol. 2, 2008, pp. 719–725.

[2] W. N. Picoto, R. Duarte, and I. Pinto, "Uncovering top-ranking factors for mobile apps through a multimethod approach," *Journal of Business Research*, vol. 101, pp. 668–674, 2019.

[3] J. Berrocal, J. Garcia-Alonso, C. Vicente-Chicote, J. Hernández, T. Mikkonen, C. Canal, and J. M. Murillo, "Early analysis of resource consumption patterns in mobile applications," *Pervasive and Mobile Computing*, vol. 35, pp. 32 – 50, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119216300797

[4] S. Laso, M. Linaje, J. Garcia-Alonso, J. M. Murillo, and J. Berrocal, "Artifact abstract: Deployment of apis on android mobile devices and microcontrollers," in *2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2020, pp. 1–2.

[5] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, 2019.

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[7] S. Shukla, M. F. Hassan, L. T. Jung, and A. Awang, "Architecture for latency reduction in healthcare internet-of-things using reinforcement learning and fuzzy based fog computing," in *International Conference of Reliable Information and Communication Technology*. Springer, 2018, pp. 372–383.

[8] A. W. Service, "Aws device farm," https://aws.amazon.com/device-farm/, (Accessed on 28/10/2020).

[9] M. Azure, "App center test," https://docs.microsoft.com/en-us/appcenter/test-cloud/, (Accessed on 28/10/2020).

[10] B. Richerzhagen, D. Stingl, J. Ruckert, and R. Steinmetz, "Simonstrator: Simulation and prototyping platform for distributed mobile applications," in *The 8th EAI International Conference on Simulation Tools and Techniques (ACM SIMUTOOLS 2015)*. IMDEA Networks Institute Publications Repository, 2015.

[11] T. Amano, S. Kajita, H. Yamaguchi, T. Higashino, and M. Takai, "Smartphone applications testbed using virtual reality," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2018, pp. 422–431.

[12] Perses-org, "Perses," https://github.com/perses-org/perses, note = (Accessed on 28/10/2020).

[13] L. Bass, I. Weber, and L. Zhu, *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[14] HashiCorp, "Terraform," https://www.terraform.io/, (Accessed on 28/10/2020).

[15] P. Fernandez, "API Pecker," https://www.npmjs.com/package/apipecker, (Accessed on 28/10/2020).

[16] Github, "Github actions," https://github.com/features/actions, (Accessed on 28/10/2020).

2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops) | 978-1-6654-0424-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/PERCOMWORKSHOPS51409.2021.9431135

Artifact: Virtual Environment for Evaluating the QoS of Distributed Mobile Applications

Sergio Laso*, Javier Berrocal*, Pablo Fernández†, Antonio Ruiz-Cortés† and Juan M. Murillo*
*University of Extremadura, Cáceres, Spain
Email:{slasom, jberolm, juanmamu}@unex.es
†I3US Institute, SCORE Lab. Universidad de Sevilla, Sevilla, Spain
Email:{pablofm, aruiz}@us.es

I. Introduction

This artifact is a guideline for the use of Perses. Perses is a framework for evaluating the Quality of Service (QoS) of distributed mobile applications. Perses allows developers to define and deploy customized environments with heterogeneous virtual mobile devices in order to simulate the deployment of these applications and analyze the monitored QoS attributes. Perses has been integrated into a DevOps methodology, automating its execution before its deployment in a real environment.

II. Perses Setup

This section explains how to configure Perses to evaluate a distributed mobile application. The main requirements are an account on Github [1] and Amazon Web Service (AWS) [2]. GitHub is required to maintain a repository storing the developed source code of the distributed mobile application. Every new commit launch the virtual environments defined with Perses by means of GitHub Actions [3]. AWS is used by Perses as an infrastructure provider to host all the virtual mobile devices to be deployed. In addition, Perses requires high-capacity AWS *.metal* instances. These enable the nested virtualisation required for Android device virtualisation. This artifact will use C5.metal (Perses takes care of managing it automatically) so you should verify if your account can instantiate that kind of instances before starting the process.[1]

To test Perses, a distributed mobile application is required. Currently, the mobile side is only compatible with Android mobile applications. In order to facilitate the evaluation of Perses, the distributed mobile application of the use case explained in [4] is provided.

Both components are accessible in public repositories [2],[3].

The process that should be followed to configure Perses and evaluate the application's QoS is:

1) **Deployment of the server-side:** To get started, it is necessary to deploy the server-side. In its repository (*Readme.md*) are the instructions for the deployment.
2) **Create the application repository:** It is also necessary to create a personal repository on GitHub to upload the mobile application. Please, note that in order to run gradle-based applications, the gradlew script must have execution rights [4]). In order to ease the testing of Perses, just a fork or a clone of the provided repository can be created that already has the execution rights.
3) **Virtual environment definition:** A configuration file, called *.perses.yml*, must be created in the mobile repository to define: the characteristics of the virtual environment in which the behaviour of the distributed mobile application should be evaluated, the tests to be launched and the desired QoS attributes that have to be checked during the test execution. This file must be created in the root of the mobile application repository.
   For the running example, this file is already provided. In particular, three sets of Android mobile devices have been defined, each one consisting of 8 devices. The difference lies on their hardware, simulating mobile devices with different CPU and RAM. In addition, 3 performance tests have been defined for evaluating the general response time of the distributed mobile application with different number of concurrent users. Finally, the QoS objective of a response time lower that two seconds is also defined, in order to provide an adequate user experience. An interface test with Espresso [5] has also been defined to evaluate the main functionality of the application in which the user clicks on the button to calculate the infection risk percentage.
4) **Automating the execution Workflow:** The execution of Perses is automated with GitHub Actions. For this, the different steps to perform (compilation of the application, installation of the necessary resources, deployment of the virtual mobile devices, execution of the tests, etc.) should be defined in a pipeline. This pipeline is already defined and can be reused to automate the execution of any project.[5]. To do this, it should be copied into the root of the mobile application repos-

[1]https://github.com/perses-org/perses/blob/master/MetalVerification.md
[2]Mobile application: https://doi.org/10.5281/zenodo.4476671
[3]Server: https://doi.org/10.5281/zenodo.4476371
[4]https://stackoverflow.com/questions/17668265/gradlew-permission-denied
[5]https://raw.githubusercontent.com/perses-org/gha/master/workflow/perses-workflow.yml

itory following the structure: *.github/workflows/perses-workflow.yml*. Please, note that this file is already also included in the provided example.

5) **AWS Credentials:** In order for Perses to be able to deploy the virtual devices in AWS, it needs credentials to access and create the necessary infrastructure. To that end, developers have to create an user and obtain the EC2 Key Pair. To create the user, the AWS tutorial should be followed.[6]. Please, note that this user must have *Programmatic* access and *AdministratorAccess* permission. After creating the user, download the provided *.csv* file for later use. To create the EC2 Pair Key, again, the AWS tutorial should be followed.[7].Please note that this key has to be created in the eu-west-1 region (this is to make it easier the artifact guide) otherwise the pipeline execution will not be successful. Finally, the generated key should be downloaded as a *.pem* file.
6) **GitHub Secrets:** to automatically deploy the virtual environment and execute the defined test when a new commit is done, the encrypted environment variables on GitHub, called Secrets, are used to include the AWS credentials in the application repository. To create them, in the application repository, developers have to access to *Settings* and *Secrets*, creating the following variables:
   - **AWS_ACCESS_KEY**. In this variable, developers should add the **Access key ID** provided in the the previously downloaded *.csv* file.
   - **AWS_SECRET_KEY** is used to insert the **Secret access key** provided in the *.csv* file.
   - **KEY_NAME**, which contains the *name* of the EC2 pair key (not the ID).
   - **KEY_PEM**, which stores a copy of the **all** content of the *.pem* file.

Once these steps are completed, Perses should be ready to deploy the virtual environment to evaluate the QoS.

III. Evaluating the QoS

Perses is integrated in a DevOps and a Continuous Integration process. To launch Perses, it is necessary to make some changes in the repository (e.g. modify the Readme.md file by including a sentence). After saving the change, Github will automatically launch Perses.This launch will execute the steps defined in the above mentioned pipeline. The process of evaluating the QoS can be reviewed accessing to the Action tab in the GitHub repository. Figure 1 shows an example of different executions.

In order to thoroughly analyze the results obtained from each Perses execution, developers can access to each of the listed Perses runs by clicking on *perses-test* in the *Jobs* section. The Github Action console will be opened showing the results of each of the steps defined in the pipeline such as building the project, creation of the infrastructure, deployment of the virtual devices, test execution, etc. Figure 2 shows an excerpt of the

[6]https://amzn.to/3nVDoQb
[7]https://amzn.to/3sCr3Ut

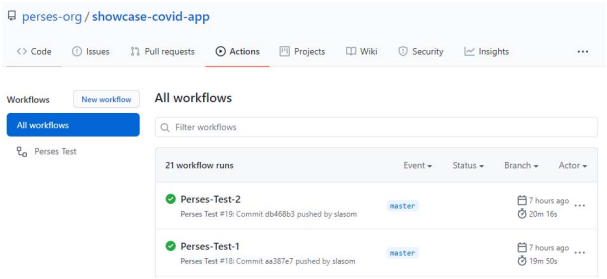


Fig. 1. GitHub Actions Main Page.

content of the step *Check Test Results*, where the results of the performance tests are evaluated to check if they are under the defined response time requirement. This console also shows if the Espresso tests were successfully passed on each device.

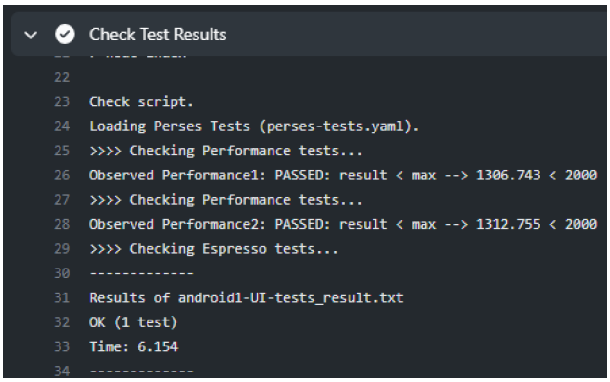


Fig. 2. Perses workflow.

Acknowledgment

This work has been partially funded by the projects RTI2018-094591-B-I00, RTI2018-101204-B-C21 and TIN2015-70560-R (MCI/AEI/FEDER,UE), the 4IE+ Project (0499-4IE-PLUS-4-E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the RCIS network (TIN2016-81978-REDT), by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), by the Government of Andalusian (US-1264651) and by the European Regional Development Fund.

References

[1] Github, "Github," https://github.com, (Accessed on 19/01/2021).
[2] A. W. Service, "Amazon web service," https://aws.amazon.com/, (Accessed on 19/01/2021).
[3] Github, "Github actions," https://github.com/features/actions, (Accessed on 28/10/2020).
[4] S. Laso, J. Berrocal, P. Fernández, A. Ruiz-Cortés, and J. M. Murillo, "Virtual environment for evaluating the qos of distributed mobile applications," in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2021, pp. 1–3.
[5] Android, "Espresso - Android Developer," https://developer.android.com/training/testing/espresso, (Accessed on 19/01/2021).

A.8 Pushed SOLID: Deploying SOLID in Smartphones

Authors: *Manuel Jesús-Azabal, Enrique Moguel, Sergio Laso, Juan Manuel Murillo, Jaime Galán-Jiménez, and José García-Alonso.*

Publication type: *Journal paper.*

Journal: *Mobile Information Systems.*

Available online: *10.1155/2021/2756666.*

Quality indicator: *JCR Q4.*

Hindawi
Mobile Information Systems
Volume 2021, Article ID 2756666, 13 pages
https://doi.org/10.1155/2021/2756666

Research Article

Pushed SOLID: Deploying SOLID in Smartphones

Manuel Jesús-Azabal, Enrique Moguel, Sergio Laso, Juan Manuel Murillo, Jaime Galán-Jiménez, and José García-Alonso

Department of Computer Systems and Telematics Engineering, University of Extremadura, Avda. de la Universidad, S/N. 10003 Cáceres, Spain

Correspondence should be addressed to Manuel Jesús-Azabal; manuel@unex.es

Received 25 May 2021; Revised 15 July 2021; Accepted 6 August 2021; Published 17 August 2021

Academic Editor: Alejandro Fernández

Personal information has become one of the most valuable coins on the Internet. Companies gather a massive amount of data to create rich profiles of their users, trying to understand how they interact with the platform and what are their preferences. However, these profiles do not follow any standard and are usually incomplete in the sense that users provide different subsets of information to distinct platforms. Thus, the quality and quantity of the data vary between applications and tends to inconsistency and duplicity. In this context, the Social Linked Data (SOLID) initiative proposes an alternative to separate the user's information from the platforms which consume it, defining a unique and autonomous source of data. Following this line, this study proposes Pushed SOLID, an architecture that integrates SOLID in the user's smartphone to store and serve their information from a single entity controlled by the users themselves. In this study, we present an implementation of the Pushed SOLID proposal with the aim of experimentally assessing the technical viability of the solution. Satisfactory performance results have been obtained at battery consumption and response time. Furthermore, users have been interviewed about the proposal, and they find the solution attractive and reliable. This solution can improve the way data are stored on the Internet, empowering users to manage their own information and benefiting third party applications with consistent and update profiles.

1. Introduction

Internet applications have become an important part of our daily routines. Every day, thousands of users interact with social networks, sharing new content, consuming posts, and communicating with others. Platforms are concerned on keeping users interested on the services they supply, so that one of the main challenges is providing the appropriate content to the right audience. Successful social webs such as Facebook (https://www.facebook.com), Twitter (https://www.twitter.com), Amazon (https://www.amazon.com), Netflix (https://www.netflix.com), or Spotify (https://www.spotify.com) provide customised services for their users based on their preferences and habits [1]. For this, the storage of personal data becomes critical for applications which benefits from fidelity to enhance the experience and competency [2]. However, these practises are not always favourable both for users and companies. The centralisation and privatisation of the information storage can drive to poor services and risks among other factors.

The privatisation and centralisation of the information is a common practise for applications and companies. Data provide a significant competitive advantage which enables a deeper understanding of the market and users. Services can be improved and adapted to the changing demands of audience which can be analysed and studied to infer these conclusions [3]. Besides, targeted advertisement makes data collection lucrative [4]. However, this privatisation trend drives sites to form information silos, so that data storage is performed independently and autonomously. Hence, the access for other applications is disabled, and users do not enjoy much control over their data. As a consequence, these policies result in potential privacy problems, inconsistencies, duplicity, and insecurity issues.

The lack of any storage standard for information and the unwillingness to share data between platforms have direct

consequences on users and companies. Reutilisation of information is not a common mechanism in this kind of application. Users who create an account on a new platform do not find easy by default to port information, friend lists, and interests in new profiles. It is true that many services allow login with Facebook or Google credentials; however, this drives to another point: privacy and dependency. Many of these platforms require accepting terms and conditions which involve targeted publicity, empowering these companies [5]. Therefore, many users place blind confidence in these policies [6], which eventually can result in data leaks [7] and in the use of information for private ends [8–10]. This illusion of control is evident when information and profiles cannot be deleted definitively even when users deactivate their accounts [11]. However, the inconsistencies and duplicity are also significant collateral effects of the privatisation.

Considering most of the applications are reluctant to share their stored information, there is a large set of data duplicated between services [12]. Information about identification, address, contact, and interests are common topics in registration processes. For example, fields such as name, phone, sex, or birthday are required for new profiles in platforms such as LinkedIn (https://www.linkedin.com/signup), Facebook (https://es-es.facebook.com/r.php), or Twitter (https://twitter.com/i/flow/signup). This way, the inputted data are duplicated, resulting in future inconsistencies. A little change in any of these parameters would oblige the user to update the information one by one. As a result, it is common that services recommend content according to past values such as addresses or interests. Likewise, this is typical on multimedia platforms such as music streaming services which recommend content according to previous reproductions. Thus, e.g., if two music streaming platforms are used in unbalanced times, the performance will not be accurate in any of them. Thus, inconsistencies and duplicity affect the user experience, which becomes wasted and results in a bad service for both application and costumer.

As a response for this, there are proposals which attempt to provide alternative models for data storage. Following the basis of data decentralisation, some of the most popular are WebBox [13] or Diaspora [14]; however, Social Linked Data (SOLID) initiative [15] becomes the most relevant among these solutions. SOLID proposes the decentralisation of the information to separate applications and personal data [16]. This way, users store their information in autonomous entities called Personal Online DataStores (PODS). Therefore, applications would adopt a model where they do not store data but request it from PODS. Thus, users are able to control accesses and authorise or deny petitions. Adopting this model provides users and companies an enriched experience based on accuracy, privacy, and control, giving response to information duplication and inconsistencies. This way, PODS becomes useful elements to enhance the experience for users and to improve companies' services.

In the present work, we propose the deployment of SOLID PODS on smartphones. This combination exploits the pervasive and contextual characteristics of these devices to provide a new storage model for personal data. Smartphones are appropriate devices to store data since most of the information involved in the applications processes are obtained from them [17, 18]. They become relevant sources of information, so that they can be easily integrated in the data flows without requiring further changes in applications. Therefore, smartphones are suitable devices to store PODS with the user profile, keeping personal information and relevant data for applications. Thus, external apps request access which the user can authorise or deny. As a result, the user is able to know which applications are consuming information and when, whereas companies benefit from a reliable update and consistent data source.

To present this work, the rest of the study is organized in four main sections. Next, some alternative proposals for data storage are introduced, and the differences with our approach are analysed. Then, our proposal is described in detail, including a reference implementation. Following, an assessment of the prototype is presented to study the acceptation and technical viability of the solution. And finally, last section draws some conclusions about the study.

2. Related Works

As the present work, many approaches have brought solutions for alternative data storage. The management and storage of personal information have elicited multiple works which try to bring transparency and control for the users. In this line, even some big companies manifest concern and corporate responsibility about the data government and propose mechanisms to help users. This is the case of Apple and its tool to enable or disable apps to track the user's activity [19]. This function increases the control over the data and its diffusion. However, the tool does not provide a mechanism to avoid data replication and inconsistencies. For this, there are other works which provide further mechanisms and privacy from external accesses. Some proposals focus on the way the data is shared through a set of entities, while others focus on centralising the storage on just one point. Examples of these can be the HAT project [20], Freenet [21], the DAT Foundation [22], the Threefold Net [23], the ActivityPub [24], the Safe Network [25], or the BBC Databox [26]. However, recently, the SOLID initiative has become one of the most relevant.

Works such as by Paulos [27], Eisenstadt et al. [28], Ramachandran et al. [29], and Mannens et al. [30] take SOLID as a base for implementations which exploit the potential of the tool and develop new functions and mechanisms.

The work developed by Paulos et al. [27] brings a deep usage of SOLID for health purposes. Taking into account the massive amount of health data collected by wearables and training devices, this proposal study an alternative storage architecture based on SOLID. The project investigates about the decentralisation of the records stored by companies such as Fitbit, Apple, or Google into a model based on SOLID deployed in the smartphone. This way, issues such as privacy, consistency, and private management are addressed. However, in contrast with our proposal, the solution stores

health data to protect privacy and consistency, while our project provides a tool to centralize information and serve any kind of information to third applications.

In the case of Eisenstadt et al. [28], the work centres on providing a reliable certification method for COVID-19 test results. This way, the project brings an architecture for storing tamper-proof and privacy-preserving certification that allows the identification of people who has submitted to Coronavirus tests. For this purpose, the study makes use of SOLID PODS running on phones, guarantying the privacy of the contained information. Eventhough the project makes use of smartphones to deploy PODS, the purpose centres on providing a validation and certification tool. Thus, the solution does not implement services to store and serve personal information. However, our work offers the mechanism to manage the individual's data as services for third parties.

In the same way, Ramachandran et al. [29] also proposed SOLID as a tool for data verification and confidentiality. Thus, the project connects SOLID PODS and Blockchain to create a decentralized architecture which provides a reliable mechanism for data storage, keeping integrity and privacy.

The project proposed by Mannens et al. [30] presented a model to streamline governmental processes making use of the SOLID PODS. The idea comes up as a response to the large amount of personal information that governments store and how institutions keep multiple copies. Issues such as consistency, access control, and privacy are addressed through the use of PODS, following legal frameworks. Therefore, citizens manage their information storing the data in their private PODS and allowing public institution the access. As a result, citizens get control over their data while responding to store inconsistencies between institutions. However, the solution is not a universal tool for information gathering but centred on bureaucracy and official institutions.

These approaches implement SOLID on smartphones. The solutions equip PODS with new behaviour and possibilities, defining a decentralized architecture based on individual devices. The proposals integrate SOLID for concrete purposes such as certification, validation or, in the case of Paulos [27], storing. However, even though these implementations exploit SOLID, they do not draw a global mechanism for data storage. This way, evaluating the different alternatives, we believe it is interesting to equip smartphones with a real relevance at information management and align their possibilities with SOLID mechanisms. Thus, this proposal places smartphones as the core of data storage to serve a new paradigm where privacy and decentralisation suppose the key element of the information government.

Pushed SOLID provides an information profile which could serve as potential tool for multiple purposes. Projects [17, 31–33] which require a profile to manage, process, and export data can be easily integrated with the architecture. In the same way, solutions [34–36] could find in the present proposal a reliable mechanism for implementation.

In the next section, the details of the proposed work are explained, and the internal mechanisms of the solution are provided.

3. Pushed SOLID

This study proposes an architecture which combines the intimate nature of smartphones and the powerful philosophy of the SOLID initiative. The solution mixes these two lines into a project which situates the user in the centre of the information management. Thus, smartphones become the store of their data, empowering the devices to serve as providers of the information to the external applications which require it. This proposal aims to provide a response to the increasing interest in data governance and presents a solution for inconsistencies and duplicity of information on personal profiles.

3.1. System Definition. The proposed architecture relies on a set of components which collaborate to provide the information to external requests as shown in Figure 1. This scheme explains the complete process to communicate a petition from an external application (ε) to the SOLID PODS (P_{solid}) of the user, executed on its smartphone. This way, three main entities shape the architecture: the SOLID PODS (P_{solid}) and Pusher app (P_{app}) in the smartphone (S) and API Gateway (G) in the server. Additionally, Firebase (F) is used to communicate petitions between the server and the smartphone and the external application (ε) is the one which begins the process. Next, the elements of the architecture are detailed and explained.

The SOLID PODS is the entity in charge of storing the personal information. This element is executed locally in the user's smartphone. It has been designed to be deployed on a server; however, in the current proposal, they have been adapted to be deployed on smartphones. Thus, the PODS stores data locally in the device, independently from any external entity. In the current proof of concept of the solution, users can interact with the platform using the default web interface in the phone. Considering the main goal of this implementation is analysing the viability of the proposal, the interactions can be done using the default front.

Once the SOLID PODS is configured, it is ready to provide information. However, the data store is locally deployed, so it can not be accessed externally, and only the own phone visualises the platform. Nevertheless, we want P_{solid} to provide the requested information to external apps (ε), so that it is required an additional mechanism to perform this. Hence, we turn to the Pusher application (P_{app}) to manage external requests. This application provides communication between the API Gateway (G) and the SOLID Server (P_{solid}) which runs on the smartphone. The API Gateway is required to effectively resolve the petitions and to map the SOLID domain requests with the address of the device. Nevertheless, the adoption of this intermediate layer does not work against the decentralized philosophy since the API Gateway does not store further data than the tuples related to addressing the petitions. In order to perform this communication between the API Gateway and the smartphone, Firebase (F) [37] is used to connect the petitions from G with the device.

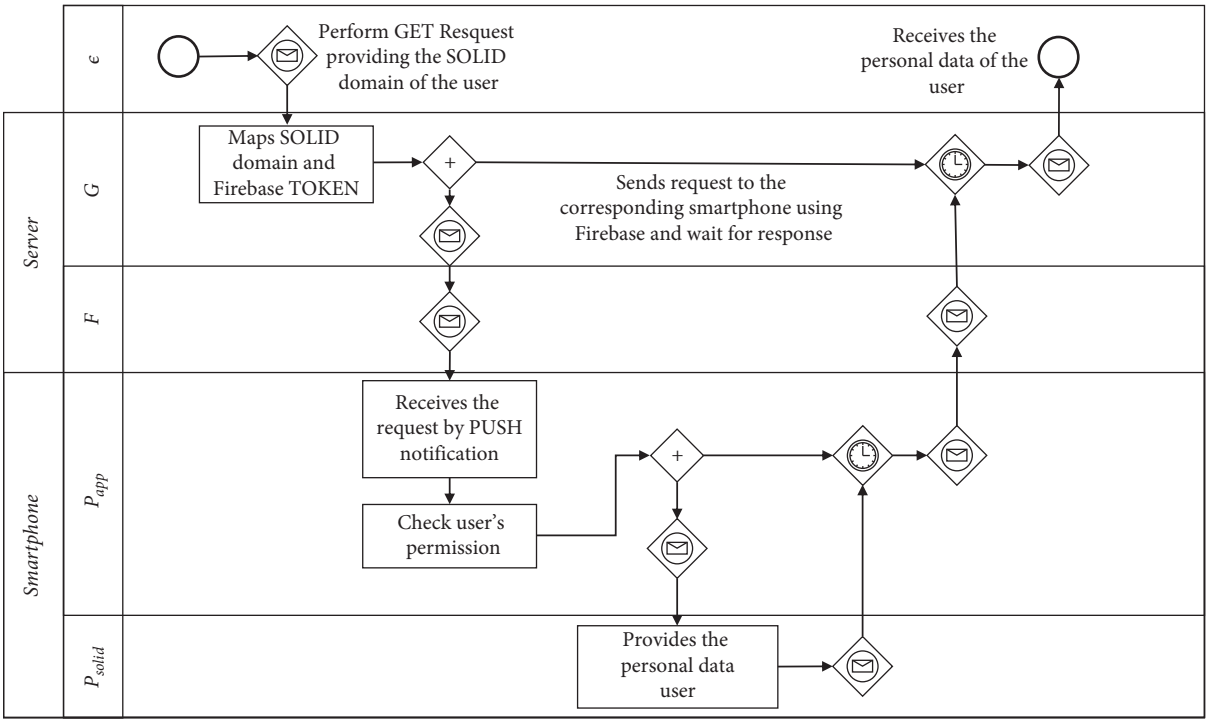


Figure 1: Pushed SOLID architecture.

Firebase is a technology developed by Google which enables a simple push-based communication between entities regardless of the technology used for Internet connectivity (Wi-Fi, 4G, or others). We have considered Firebase as a communication component for its simplicity but can be easily replaced by a similar open-source mechanism like MQTT [38], which would allow information transfer between the layers. Nevertheless, Firebase provides reliable performance and its terms of usage specify how the data stored by the technology are basic information required to operate [39].

Once Firebase communicates the petitions received in the API Gateway, this component communicate the request with the smartphone executing a callback method of Pusher app. Therefore, the method executed in Pusher app asks for the required information to the SOLID PODS, which selects the corresponding data from the profile and returns the values. As a result, Pusher app provides the requested data to the API Gateway. Considering the callback method identifies the requester entity and SOLID PODS counts with access control; security and validation tasks can be easily integrated in the process. As a result, the technical details to replicate the implementation process can be found in the project repository (https://bitbucket.org/spilab/solidsituational.context).

Considering these three components of the system, the way they are coordinated begins from the installation and configuration of the SOLID PODS and the Pusher application. Once these two elements are operative, the Pusher app communicates the addition with the API Gateway, which will be in charge of mapping incoming petitions to the corresponding smartphone. For this, the token ID of Firebase is updated from the Pusher app, in the case it changes. Therefore, the API Gateway keeps updated with the latest value required to operate.

The detailed working of Pushed SOLID is shown in Figure 2, which identifies all steps and collaborations between the different layers. First, the external application (ϵ) requires a concrete set of information from a user (step 1) (e.g., a new web platform where a user is creating an account). This way, instead of having to input all their personal data, the new member provides its SOLID Domain (2). This variable is the public username of its SOLID PODS and serves as main identification in the architecture. Therefore, the application only have to demand the required data to the API Gateway (G), providing the SOLID domain. The API Gateway is the only entity which truly knows where the information is stored. Thus, this layer uses Firebase technology to easily communicate with smartphones through push notifications. Moreover, the API Gateway maps the SOLID Domain with the Firebase ID and notifies the corresponding smartphone (S) about the petition (3). As a result, the device receives the notification and asks the user for acceptance to allow the external app access to the information (4). This transaction is carried out with the Pusher application (P_{app}) which executes a callback method invoked when the API Gateway receives interactions. Then, the user is asked to allow or deny petitions. This way, the permissions can be configured, defining concrete requester as reliable. This operation can be performed straightly in the SOLID PODS or using the Pusher app. In the case the permission is favourable, the local SOLID PODS (P_{solid}) provides the required information to the Pusher app. Then, the application responds to the API Gateway with the data (5).

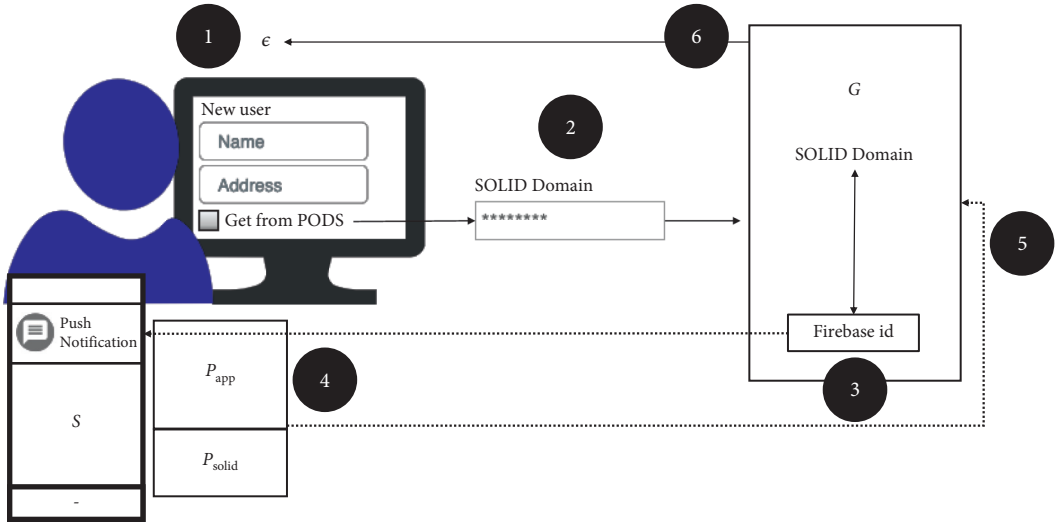


Figure 2: Serving data from a smartphone-based PODS.

Finally, the API Gateway returns the answer to the requesting platform (ε) and the process is completed (6).

Considering this procedure, Pushed SOLID becomes a suitable tool to manage the information both for users and applications. Users are able to keep all their personal information in just one place: their phone, which physically stores the data. Furthermore, it is possible to identify all applications and entities which access the profile. Thus, the user can authorise and revoke petitions at any time. On the other hand, applications benefit from the centralisation of the data in the PODS. This solution guarantees the reliability of the information and that the personal profiles are complete and updated. Besides, inconsistencies and data replication are solved, providing uniformly a common source of data for all applications. As a result, information can be shared between platforms, and it is possible to track the entities which consume our data and the storage becomes standardised. Hence, the way personal information is stored in the Internet could change, bringing a new philosophy based on control and governance for the user.

3.2. Use Case. We have already approached to some of the issues which affect users experience due to the independence of the data between platforms. One of the most recurrent is enjoying a service using multiple platforms, such as music streaming, where users tend to distribute reproductions among several applications [40]. This section compares the ordinary scenario for a user who does not use Pushed SOLID and how the proposal can improve the experience.

For this example, we consider two of the most used music streaming platforms [41]: Amazon Music and Spotify. Both applications count with a recommendation system based on historical reproductions and favourite lists, so the more a user uses a given platform, the more accurate suggestions are got from that platform. However, these applications store the information privately and they do not share it with other services. This way, these politics become inconvenient when users consume more than one streaming platform. Thus, the irregular time spent in each service drives to inaccurate recommendations. Considering this, the proposed use case represents a possible scenario (Figure 3): a user spends 90% of time (t = 90%) listening to music with Spotify ($\varepsilon_{\text{Spotify}}$) and 10% ($t$ = 10%) using Amazon Music ($\varepsilon_{\text{Amazon}}$). As a result, the knowledge about preferences is unbalanced, so that Amazon Music cannot provide accurate recommendations while Spotify lacks on a 10% of the music reproductions. Therefore, this context can highly affect the user experience (ω_1).

As a response for this recurrent situation, Pushed SOLID proposes the unification of the knowledge generated from platforms in the SOLID PODS (ω_2). Thus, the information about historical reproductions and favourite music is stored in a single entity: the SOLID profile (PODS). This way, music streaming platforms read and modify the same music preferences independently from the usage time. As a result, all applications provide recommendations as if they would be used every time. This improvement is reflected on Figure 3 where both Amazon Music and Spotify consume the same information about music preferences.

As a result, Pushed SOLID improves the recommendation performance of the music streaming platforms and provides a better experience, sharing data sources and democratizing the custom experiences possibilities. In the same way, the user is not the only beneficiary since applications will be able to perform more accurate recommendations independently from the time of usage. Thus, Pushed SOLID becomes a very interesting solution for information management, whereas the data governance befalls completely on the user. Considering the proposal uses the smartphone as a data provider, performance and consumption become two main issues for the technical viability. Moreover, next section evaluates the implementation of the architecture and analyses the response of the solution under daily use, checking the technical viability of the solution.

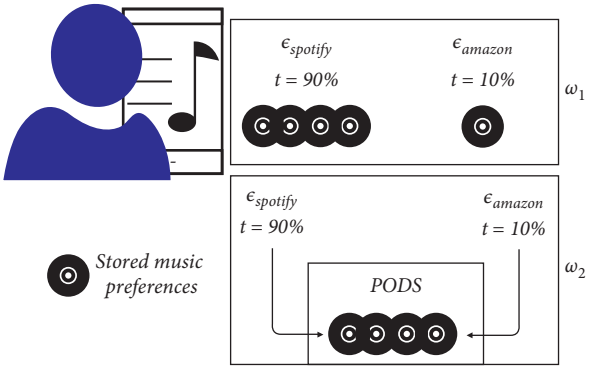


Figure 3: Music streaming with (ω_2) and without (ω_1) situation SOLID.

4. Pushed SOLID Validation

The architecture proposes the use of the smartphone as the centre of all our profiles on the Internet. This means that the device is in charge of managing all incoming data petitions and satisfying them with the corresponding information. Luckily, the technical capabilities of the smartphones have been constantly improving during the last years. However, the high requirements of the proposal suppose a challenge for devices, involving factors such as battery consumption and response time. In addition, it is relevant to consider the possible controversy nature of the project. Taking into account the goal of the platform is managing the large set of personal information distributed along the Internet, user's perception is critical to build a reliable solution. Considering this two points, an experimental evaluation about user's opinion and a system assessments about the performance and technical viability of the proposal has been performed. For this evaluation, the implementation of Pushed SOLID counts with PODS which stores a profile with personal information, historical music played, and main interests (Table 1).

The assessments performed in the experiment followed two dimensions: the perception and usability of the solution and the technical performance of the implementation.

4.1. Usability Tests. The perception and feeling of the users is a critical variable for the proposal. Considering the main goal is managing all personal information, it becomes clearly relevant to study the reaction and opinion from users when they use the prototype. Thus, several steps were considered to perform usability tests. Therefore, a methodology based on user testing [42] was applied to obtain reliable answers. Following this guideline, we were able to elaborate a testing context which allowed us to analyse the perception of the idea and the first opinions after its use. This way, a group of users participated and answered two different surveys.

The first step was to select a group of sixteen users, who are not related with the project, and to introduce them to the purpose of Pushed SOLID. The background of the participants varied, with only six technicians. This way, we try to avoid bias.

Table 1: Personal information stored in a PODS.

Name
Role
Organization name
Address
E-mail
Telephone
Historical music played
Main interests

Then, a first survey about the perception of the idea was done, addressing privacy politics, data centralisation, and opinions, from a neutral point of view. This way, the questions raised clearly the issue, approaching current tendencies and the concept that users have understood about the proposal. The answers were based on ranges: each question (Q) raised a topic and testers had to provide a score (S_p) from $0 \leq S_p \leq 5$, according to the relevance they give to that specific topic. Additionally, a comments box gathered opinions and suggestions. The questions included in the perception survey are given in Table 2. Thus, an average of the obtained results from the survey is shown in Figure 4. Besides, since Q_5 is not a scoring topic but a text parameter, the most recurrent words in the answers are considered. Additionally, a control question was introduced to check if any respondent answered randomly. As a result, all users answered correctly.

Figure 4 shows the obtained scores in the perception survey. The results manifest the interest from users about the proposal and about the phone as the centre of information management. In the same way, testers also showed distrust regarding data management policies in companies (Q_1), especially in those participants who were technicians. For the questions concerning the project concept, users reflected interest for centralisation (Q_2), defining the overall valued dimension. In the same way, testers also think that public would approve and adopt the proposal (Q_3). However, the values define this characteristic as the lowest rated regarding the work. In the case of the main purpose of Pushed SOLID, users think this can become a solution for store issues like replication or inconsistencies (Q_4). Additionally, the answers in Q_5 are analysed considering the recurrence of words. For this, we have considered as common concepts those words which are included in the abstract description of the solution. Therefore, concepts like smartphone (appeared 14 times in responses, considering "phone" as synonym), Internet (13), privacy (11), data (10), and information (6) are ignored. As a result, the words which repeat the most and are not common are centralisation (10), companies (9), security (7), trust (6), change (6), and music (4). Hence, the messages were mainly focused on manifesting the relevance of the idea as a disruptive tool which proposes a great change at data management through centralisation. Besides, the tendency manifests how there is a widespread concern about providing security mechanisms to guarantee a safe storage in the smartphone, defining this as the main core of the project. Also, there were comments which support that companies should take part in the proposal, manifesting the importance

Table 2: User perception survey.

	Q	S_p
Q_1	Do you trust the current politics for storage methods for personal information at big companies?	[0,5]
Q_2	Do you like the idea of keeping all your data in your phone?	[0,5]
Q_3	Do you think current users will embrace this model?	[0,5]
Q_4	Do you think this model solves main management issues? (consistency, duplicity...)	[0,5]
Q_5	How would you improve the platform?	[text]

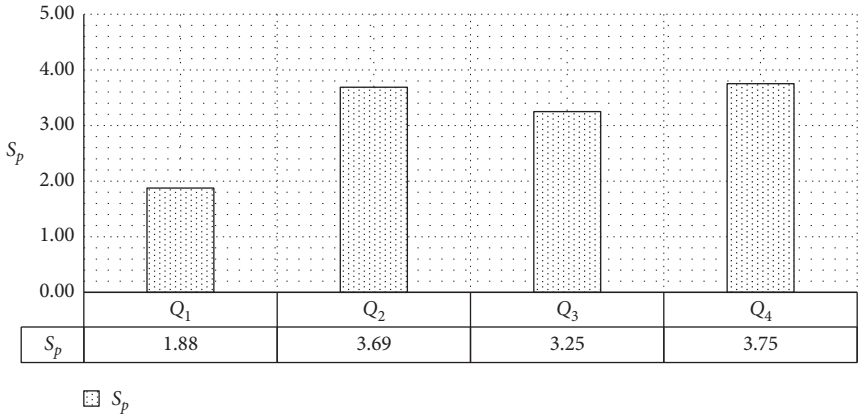


Figure 4: Score (S_p) of the questions (Q_n) about the perception of the project.

of providing a good experience also for enterprises. At last, some contributions were focused on potential applications like music services. Considering the results obtained in the first interaction with users, the survey discloses significant interest in the proposal by the testers.

Once the first approach was finished, testers interacted with the platform. The tasks they performed mainly focused on user experience, so that they created a PODS and inputted information. Besides, they supervise information petitions from external applications. In the same way as the previous step, testers answered a survey about the user experience using the proposal implementation. Hence, testers scored (S_x) from $0 \leq S_x \leq 5$, the satisfaction and user-friendliness of the solution. The questions included in the survey are given in Table 3. As a result, Figure 5 shows the obtained average results.

Figure 5 shows the obtained results from the user experience survey. In average, the scores denote testers have had a successful experience, specially technicians. For Q_5, the average of answers situates the information input with 2.69, around one point below Q_7. The most rated parameter in the survey was Q_6 with 3.69. Hence, these results show a great acceptance for a novel proposal. Testers mainly highlight the simplicity at using the platform, especially for the registration process and configuring the SOLID PODS. Additionally, the suggestions and comments in Q_8 are also interesting. In the same way as previous survey analysis, the opinions in the comments field are addressed with the most recurrent words. Thus, the words which repeat the most were user interface (11 times), easy (10), comfortable (9), battery (8), performance (5), and integration (4). These recurrences manifest some of the most favourable points of the proof of concept. Users talked about the user interface and the way the application is easy to configure. Also, testers place a value on the comfort of checking the information as well as the incoming petitions from external applications. In the same way, performance and battery were two of the most recurrent issues, standing out the importance of providing a good efficiency in the execution. As a result, the implemented solution provides a successful user experience with an easy-to-use interface.

Considering the obtained results in the usability tests, the proposal fulfils the requirements of usability and provides a disruptive solution to manage information. However, it is important to highlight the performance requirements that the prototype meets. For this, the next section assesses the technical behaviour of the solution, evaluating the consumed battery and the response time and drawing a discussion of the results.

4.2. Technical Performance. One of the main dimensions of the project is the technical viability. Considering the architecture proposes the smartphones as the provider for all information required by applications, the solution must guarantee a good performance. This way, parameters such as the energy consumption and response time become critical variables to analyse. In this section, the provided implementation of the Pushed SOLID is assessed in a laboratory context where random petitions are simulated to evaluate battery consumption in the smartphone devices and the time required to obtain a concrete data from the PODS. Next, the setup configuration is explained and the results are discussed.

Table 3: User perception survey.

	Q	S_x
Q_5	Satisfaction at information input in SOLID PODS	[0,5]
Q_6	Simplicity at the registering process	[0,5]
Q_7	Simplicity at specifying SOLID PODS parameters	[0,5]
Q_8	Comments field to suggest changes	[Text]

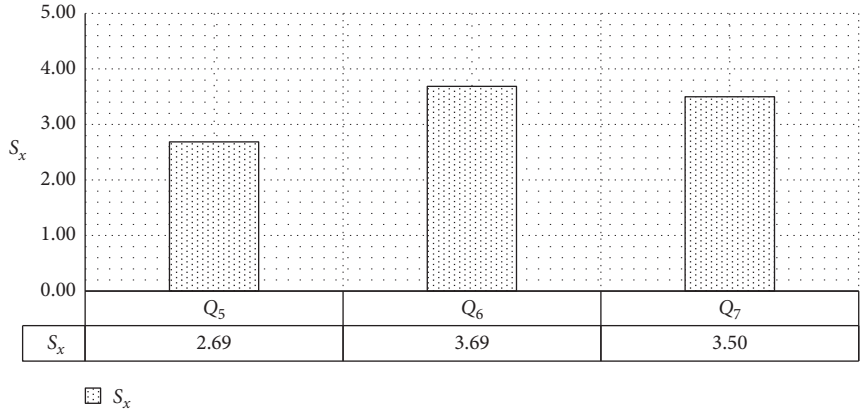


Figure 5: Score (S_x) of the questions (Q_n) about the user experience.

4.2.1. Assessment Setup. In order to evaluate the platform, an assessment setup is built. This scenario is composed by two different contexts: the laboratory and the smartphones. Each of them is in charge of deploying any of the three different entities involved in the architecture: external application which performs petitions (E_{app}), the API Gateway which communicates petitions with smartphones (G), and smartphones which executes the Pusher app (P_{app}) and the SOLID PODS to store the information (P_{solid}). Considering this, an experimental context is built to simulate a heavy charge of petitions to the architecture. Figure 6 represents graphically the process.

The petitions are generated in the laboratory context. It is shaped by two computers, computer $A(C_A)$ and $B(C_B)$ (Table 4). The first one deploys the API Gateway (G), whereas the second one send petitions to it, simulating the activity of an external application (E_{app}). C_B follows Algorithm 1 to request the data to the API Gateway, which communicates with the smartphones. The smartphone context is shaped by two devices: smartphones $A(S_A)$ and $B(S_B)$ (Table 5). Both devices are submitted to a daily use by their owners while executing a SOLID PODS and answering automatically the petitions generated by C_B. As a result, the raised scenario proposes a experimental context where the performance of the platform can be easily tested. Every time C_B performs a petition, the time lapsed between the request and the response is registered. In the same way, both smartphones S_A and S_B monitor the battery consumed by serving the data from the PODS.

The proposed execution scheme is performed during three days continuously, following the instructions of Algorithm 1. This way, there are two programmed peaks of petitions during the first two days. These events try to check the capability of the platform to provide response to a large

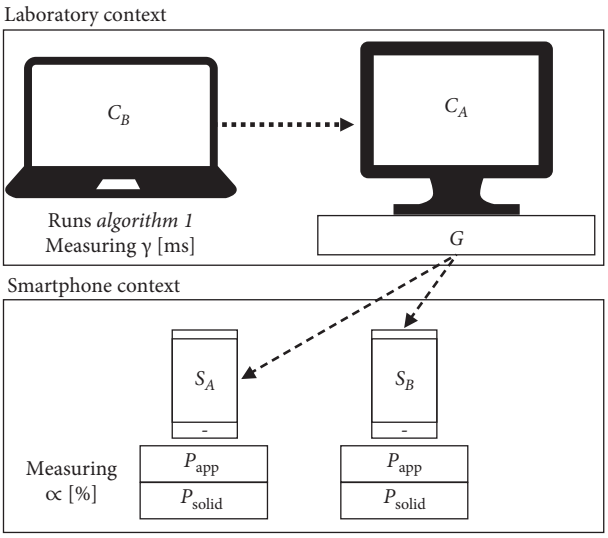


Figure 6: Evaluation setup.

set of requests, while analysing the impact on the response time and battery average. The number of petitions performed during an hour in the experiment ranges from zero to twelve. Therefore, the traffic does not follow a clear pattern and changes randomly. This idea corresponds with the goal of providing an irregular and realistic flow of messages. For this, a generated number and a waiting time of five minutes is used. As a result, this behaviour tries to provide a changing number of requests based on the average number of notifications that smartphones use to receive in a day [43]. Once the execution concludes, outcomes are obtained. The next section discusses the results and extracts conclusions.

Table 4: Technical specifications of the computers used in the experiment (C_A and C_B).

Specifications	C_A	C_B
Operative system	Ubuntu 19.10	MacOS Catalina 10.1
RAM memory	8 Gb	16 Gb
Processor	IntelCore i5-3570 (3.4 GHz)	IntelCore i5-7360U (2.3 GHz)

Require: returns false if simulation time is over, $S(\,)$; petition probability, P; petition type, P_t; returns true if it is peak time, P; get the SOLID DOMAIN from smartphone, SOLID (smartphone); generates a number between [0.0, 1.0], R; wait for five minutes, $W(\,)$; returns the type of data to request, $T(\,)$; perform the request of a concrete data to a concrete smartphone, $L(S_n, T)$.

```
(1) do
(2)   P←R
(3)   if P > = 0.5 then
(4)     do
(5)       P_t←T
(6)       L(SOLID(S_A), P_t)
(7)       L(SOLID(S_B), P_t)
(8)     while P = True
(9)   end if
(10)  W
(11) while S = True
```

Algorithm 1: Pseudocode used to generate random petitions, executed in C_B.

Table 5: Technical specifications of the smartphones used in the experiment (S_A and S_B).

Specifications	S_A	S_B
Model	LG G5 Titan	Xiaomi Mi A2 Lite
Android version	Android 6.0 Marshmallow	Android X
Battery (mAh)	2800	4000
Processor	Snapdragon 820 (2.15 GHz)	Snapdragon 625 (2.0 GHz)

4.2.2. Result Analysis. After three days of executions, the results are obtained. There are two sources of results: computer B (C_B), which has been monitoring the response time, and smartphones A (S_A) and B (S_B), which have monitored the battery level evolution. These two variables are specially relevant to qualify the performance of the platform, since they reflect critical values for the purpose of the solution. This way, the battery consumption and response time are introduced.

On the one hand, the battery consumption (α_s (%) with s as the corresponding smartphone) represents the percentage of battery consumed by the application execution. Smartphones run two main components: the SOLID PODS and the Pusher app. The first one waits for petitions from Pusher app which, at the same time, receives requests from the API Gateway, using Firebase. Therefore, the consumption of energy is an important part of the project. Considering that Internet application would require information to the smartphone, it is important to provide availability in the service. For this, the energy consumed by the architecture must be low in smartphones. In order to obtain realistic results, S_A and S_B have been submitted to daily use by their owner with tasks such as surfing the net, messaging, multimedia, and geopositioning. The experiment is focused on analysing how Pushed SOLID can be embed in the day-to-day routines and address the quality of its integration with the rest of functions. Therefore, we discarded the assessment in a controlled environment, focusing in results obtained from a real context. This way, some inaccuracies are assumed, such as distinguishing between consumptions in WiFi or cellular network. For this task, the android Battery Monitoring Tool [44, 45] has been used. As a result, the obtained values correspond with a realistic context.

On the other hand, the response time (γ (ms)) represents the time elapsed between a petition is done by an external app and the information is received by the entity. This way, this parameter becomes especially relevant since the information must be provided quickly to the multiple demanding applications. Factors such as the CPU usage, the memory available, and the Internet connection can affect the results, but this also enriches the outcomes. Considering the idea is integrating the service with an average use of the smartphone, it is interesting to study the response of the architecture when the device is submitted to a real use. Once the executions are completed, the results are obtained. Thus, Figures 7 and 8 show the outcomes.

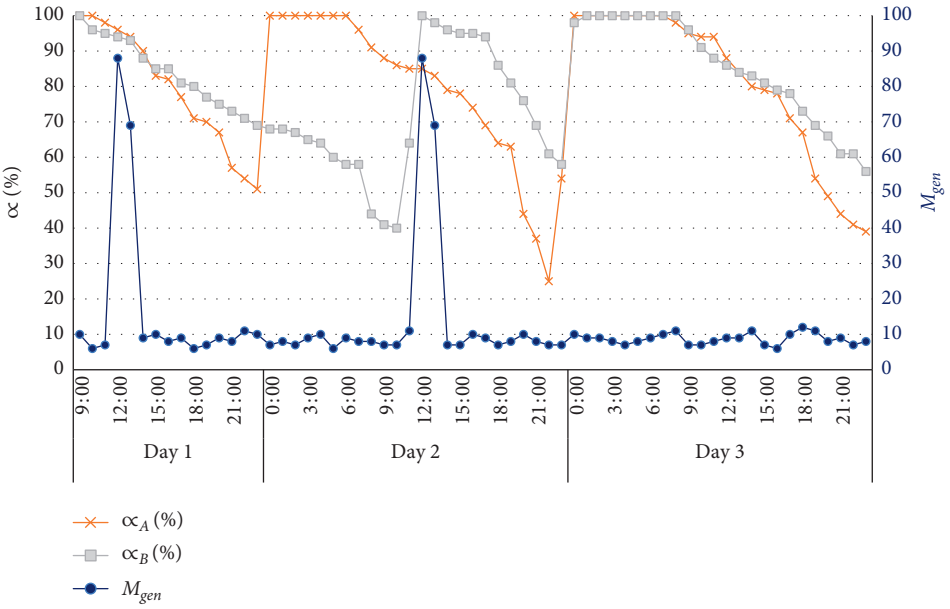


Figure 7: Battery evolution (α) and messages generated (M_{gen}) of smartphones.

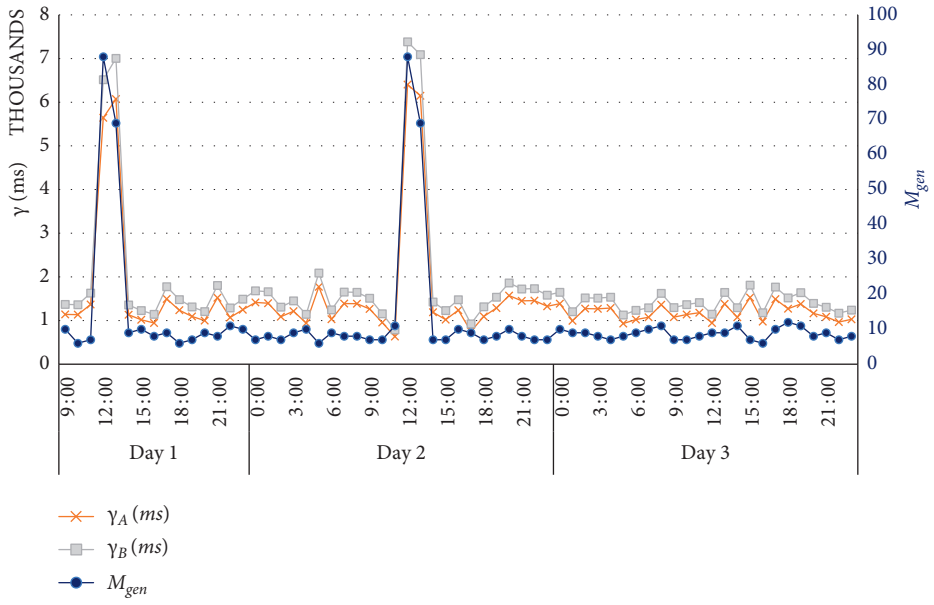


Figure 8: Response time (γ) evolution and messages generated (M_{gen}) in smartphones.

Figure 7 shows the evolution of the battery in S_A and S_B during the three days of executions. The figure represents three different variables as a function of the execution time (T): the battery evolution of both smartphones (α_a and α_b) and the number of generated requests (M_{gen}). As can be seen, there is not a clear connection between M_{gen} and the battery evolution (α_a and α_b). There are two programmed peaks in M_{gen} at $T = 11.00$ on days 1 and 2. These events were preset to find more evident impact on battery. However, as can be seen, the battery is not directly affected by the requests. Therefore, we can deduce that the energy consumption derived from the architecture operation on the smartphones is very low. Additionally, the Android Battery Monitoring Tool indicates consumption below 0% for Pusher app, a report which matches with the battery consumption trajectory.

In the case of γ, Figure 8 shows the recorded times to respond to external applications as a function of the execution time (T). Thus, for this case, we can appreciate a connection between M_{gen} and γ. The average of response time rounds $\gamma = 2152.25$ (ms) with a median number of petitions of $M_{\text{gen}} = 12.87$. The programmed peaks of requests have a direct impact on the time required to solve the petitions. Thus, γ increases from the average value up to more than 7000 (ms), evidencing that the solution is sensitive to a large number of requests. Moreover, in the case of

the rest of the petitions, there are some values above one second. The size of the transmitting messages varies, depending if it is a petition or it is a response, between 582 bytes and 612–1024 bytes, respectively. Therefore, it is difficult to identify the size as a bottleneck. Nevertheless, This delay can be introduced by the API Gateway, which performs a waiting time to receive information from the smartphone. As a result, the time required to retrieve information from the smartphone can be optimized. However, considering the main goal of the assessment is analysing the viability of the proposal, the delays can be accepted.

The prototype developed as a proof of the concept of Pushed SOLID provides a valid solution to store our personal data on phones. The approach has been tested to study the technical viability of the project and the results are favourable. Two of the main features the proposal must provide are acceptable energy consumption, which does not affect significantly the battery, and a reasonable response time which guarantee an effective service for external apps. Thus, once the tests have been performed, the results are favourable. On the one hand, the deployment of the SOLID PODS and Pusher app in the smartphones does not have a significant impact on the battery. In fact, the obtained results manifest how low is the energy consumed by the solution. Besides, the petition peaks do not specially affect the battery performance. On the other hand, the response time is sensitive to the simultaneous requests. However, the average time required to provide information is low and satisfies the requirements of the service.

Considering the two dimensions of the assessment, usability and technical performance, we can consider the prototype as a good implementation of the architecture. The implemented solution becomes a reliable method to identify the possibilities of the project and its scope. As summary, the involved testers manifest significant interest for the proposal and the qualitative evaluations show its technical viability. Therefore, the project represents a successful work line with a promising approach to a new data paradigm. In order to define some considerations in the assessment, the next subsection draws the threats to the validation process, identifying some relevant considerations about the results. Furthermore, the last section brings conclusions about the proposal and the advances that the work implements.

4.3. Threats to Validity. In this section, we address the most relevant threats to validity which can condition the performed assessment. For this, two points are approached: tests with users and the smartphone operability.

One of the most relevant stages in the validation of Pushed SOLID is the usability tests. These activities involved a set of research studies to know the concern and opinion of potential users about the proposal. The results manifested a good response from research studies who considered the system as an appropriate alternative option for data storage. However, it must be considered that six of the sixteen users who participated in the experiment were technicians, acquainted with brand new solutions and novel technologies. In the case of other participants, who are ten, they were not technical profiles, so they provide more reliable results for the user testing.

On the other hand, the operability of the smartphone becomes also a threat. The smartphone which executes Pushed SOLID can loose connectivity or its battery can get discharged, so that petitions from external applications can be missed. For this, we rely on the inherent usage patterns of the user and how smartphones use to be always connected and alive. According to studies like [46–48], most users keep their phones with battery during all the day, as well as connected to the Internet. Therefore, we consider the proposal can likely be operative in almost all situations.

5. Conclusion

Personal data have become a very valuable coin for Internet companies. The custom services and advertisement are a relevant issue for enterprises which find publicity as an important source of incomes. As a result, companies keep private their data from other apps, lacking of any standard, and bringing also replication and inconsistencies between platforms. In response to this, the SOLID initiative proposes an alternative storage philosophy which centralises information on uniquely individual entities called PODS. Thus, users keeps all their information while controlling the external accesses to the data. In this framework, we propose Pushed SOLID, an architecture which deploys SOLID PODS on user's smartphone. Thus, the information is stored in the device, which serves as provider of the data to external requests. This way, the user can easily authorise or deny accesses to the PODS.

With the objective of studying the possibilities of the work and the technical viability, a prototype of the proposal was implemented. Thus, a group of testers were interviewed about their opinion of the project, obtaining very good feedback. Users considered the model as a suitable response to the uncontrolled and massive storage of personal data. Besides considering the solution situates the smartphone as the centre of the implementation, it is relevant to guarantee the technical viability. For this, performance assessments were performed to study the impact of the solution on battery consumption and the response time of the requests. As a result, the experimental tests showed a good response of the prototype for both parameters, battery and response time. As a result, the obtained results manifest the relevance of the proposal and the technical viability.

Considering the assessments and evaluations of the proposal, Pushed SOLID can be considered as a valid solution for data storage. The possibility of keeping all personal data on the Internet in just one device enhances the users control over their own information. This way, both users and enterprises benefit from the work. On the one hand, users are empowered with a reliable knowledge about the entities which access to their data, being able to refuse requests while keeping a track on the consumed information. On the other hand, enterprises obtain updated and consistent information while issues like replication are solved. As a result, Pushed SOLID proposes a new paradigm on the Internet which

becomes a new way of understanding privacy, while users and enterprises are benefited.

Data Availability

The implementation is available on the repository of the project: https://bitbucket.org/spilab/solidsituational. context. Furthermore, results and obtained data are also included.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the 4IE+ Project (0499-4IE-PLUS-4-E) funded by the Interreg V-A España-Portugal (POCTEP) 2014–2020 program, the project RTI2018-094591-B-I00 (MCI/AEI/FEDER,UE) by the Department of Economy, Science and Digital Agenda of the Government of Extremadura (GR18112 and IB18030), and by the European Regional Development Fund.

References

[1] M. Ruckenstein and J. Granroth, "Algorithms, advertising and the intimacy of surveillance," *Journal of Cultural Economy*, vol. 13, no. 1, pp. 12–24, 2020.

[2] S. C. Matz, J. I. Menges, D. J. Stillwell, and H. A. Schwartz, "Predicting individual-level income from Facebook profiles," *PLoS One*, vol. 14, no. 3, Article ID e0214369, 2019.

[3] A. Esteve, "The business of personal data: Google, Facebook, and privacy issues in the EU and the USA," *International Data Privacy Law*, vol. 7, no. 1, pp. 36–47, 2017.

[4] J. Sheth, "New areas of research in marketing strategy, consumer behavior, and marketing analytics: the future is bright," *Journal of Marketing Theory and Practice*, vol. 29, no. 1, pp. 3–12, 2021.

[5] J. G. Cabañas, Á. Cuevas, A. Arrate, and R. Cuevas, "Does Facebook use sensitive data for advertising purposes?" *Communications of the ACM*, vol. 64, no. 1, pp. 62–69, 2020.

[6] C. Niu, Z. Zheng, F. Wu, S. Tang, X. Gao, and G. Chen, "Unlocking the value of privacy: trading aggregate statistics over private correlated data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2031–2040, London, UK, August 2018.

[7] D. Lupton, "Thinking with care about personal data profiling: a more-than-human approach," *International Journal of Communication*, vol. 14, no. 19, 2020.

[8] D. Hunter and N. Evans, "Facebook emotional contagion experiment controversy," *Research Ethics*, vol. 12, no. 1, pp. 2-3, 2016.

[9] M. Klenk, "Digital well-being and manipulation online," *Philosophical Studies Series, Ethics of Digital Well-Being*, Springer, Berlin, Germany, 2020.

[10] I. U. Rehman, "Facebook-Cambridge analytica data harvesting: what you need to know," *Library Philosophy and Practice*, vol. 1, no. 1, pp. 1–11, 2019.

[11] J. P. Black, "Facebook and the future of fair housing online," *Oklahoma Law Review*, vol. 72, no. 3, p. 711, 2020.

[12] M. U. Tahir, M. R. Naqvi, S. K. Shahzad, and M. W. Iqbal, "Resolving data de-duplication issues on cloud," in *Proceedings of the 2020 International Conference on Engineering and Emerging Technologies (ICEET)*, pp. 1–5, IEEE, Lahore, Pakistan, February 2020.

[13] M. Van Kleek, D. A. Smith, N. Shadbolt et al., "A decentralized architecture for consolidating personal information ecosystems: the Webbox," 2012.

[14] Diaspora Foundation, 2021, https://diasporafoundation.org/.

[15] The SOLID Foundation, "The solid project," 2021, https://solid.mit.edu.

[16] A. Vlad Sambra, E. Mansour, S. Hawke et al., "Solid: a platform for decentralized social applications based on linked data," MIT CSAIL & Qatar Computing Research Institute, Tech. Rep., 2016.

[17] J. Garcia-Alonso, J. Berrocal, J. M. Murillo, D. Mendes, C. Fonseca, and M. Lopes, "Situational-context for virtually modeling the elderly," in *Proceedings of the International Symposium on Ambient Intelligence*, pp. 298–305, Springer, Toledo, Spain, June 2018.

[18] E. Moguel, J. Berrocal, J. M. Murillo et al., "Enriched elderly virtual profiles by means of a multidimensional integrated assessment platform," *Procedia computer science*, vol. 138, pp. 56–63, 2018.

[19] Apple Inc, "Data tracking by applications," 2021, https://developer.apple.com/app-store/user-privacy-and-data-use/.

[20] I. C. L. Ng, "Can you own your personal data? the hat (hub-of-all-things) data ownership model," 2018.

[21] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: a distributed anonymous information storage and retrieval system," in *Designing Privacy Enhancing Technologies*, pp. 46–66, Springer, Berlin, Germany, 2001.

[22] The DAT Foundation, https://dat.foundation, 2021.

[23] The Threefold Network, https://threefold.io, 2021.

[24] ActivityPub, https://activitypub.rocks/, 2021.

[25] SafeNetwork, https://safenetwork.org/, 2021.

[26] BBC, "BBC box," 2021, https://www.bbc.co.uk/rd/projects/databox.

[27] J. Paulos, *Investigating decentralized management of health and fitness data*, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2020.

[28] M. Eisenstadt, M. Ramachandran, N. Chowdhury, A. Third, and J. Domingue, "COVID-19 antibody test/vaccination certification: there's an app for that," *IEEE Open Journal of Engineering in Medicine and Biology*, vol. 1, pp. 148–155, 2020.

[29] M. Ramachandran, N. Chowdhury, T. Allan, J. Domingue, K. Quick, and M. Bachler, "Towards complete decentralised verification of data with confidentiality: different ways to connect solid pods and blockchain," in *Proceedings of the Companion Proceedings of the Web Conference 2020*, pp. 645–649, Taipei, Taiwan, April 2020.

[30] E. Mannens, R. Verborgh, and T. Berners-Lee, "Streamlining governmental processes by putting citizens in control of their personal data," in *Proceedings of the Electronic Governance and Open Society: Challenges in Eurasia: 6th International Conference, EGOSE 2019*, vol. 1135, p. 346, Springer Nature, St. Petersburg, Russia, November 2019.

[31] M. Jesús-Azabal, J. Rojo, E. Moguel et al., "Voice assistant to remind pharmacologic treatment in elders," in *Proceedings of the International Workshop on Gerontechnology*, pp. 123–133, Springer, Evora, Portugal, October 2019.

[32] E. Moguel, M. Jesus Azabal, D. Flores-Martin, J. Berrocal, J. Garcıa-Alonso, and J. M. Murillo, "Asistente de voz para el recordatorio de tratamiento farmacológico," 2021.

[33] J. Rojo, D. Flores-Martin, J. Garcia-Alonso, J. M. Murillo, and J. Berrocal, "Automating the interactions among iot devices

using neural networks," in *Proceedings of the 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 1–6, IEEE, Austin, TX, USA, March 2020.

[34] X. Dong, Y. Guo, F. Li, L. Dong, and A. Khan, "Combination model of heterogeneous data for security measurement," *Journal of Universal Computer Science*, vol. 25, no. 3, pp. 270–281, 2019.

[35] A. Nikiforova, J. Bicevskis, Z. Bicevska, and I. Oditis, "User-oriented approach to data quality evaluation," *Journal of Universal Computer Science*, vol. 26, no. 1, pp. 107–126, 2020.

[36] A. Choi and H. Shin, "Longitudinal healthcare data management platform of healthcare iot devices for personalized services," *Journal of Universal Computer Science*, vol. 24, no. 9, pp. 1153–1169, 2018.

[37] L. Moroney, Moroney, and Anglin, *Definitive Guide to Firebase*, Springer, Berlin, Germany, 2017.

[38] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," *The Journal of Open Source Software*, vol. 2, no. 13, p. 265, 2017.

[39] Google Inc, "User agreement firebase," 2021, https://firebase.google.com/support/privacy?hl=es-419.

[40] R. Prey, "Nothing personal: algorithmic individuation on music streaming platforms," *Media, Culture & Society*, vol. 40, no. 7, pp. 1086–1100, 2018.

[41] Statista, "Most used music streaming services," 2021, https://www.statista.com/statistics/798125/most-popular-us-music-streaming-services-ranked-by-audience/.

[42] F. Z. Ghazizadeh and S. Vafadar, "A quantitative evaluation of usability in mobile applications: an empirical study," in *Proceedings of the 2017 International Symposium on Computer Science and Software Engineering Conference (CSSE)*, pp. 1–6, IEEE, Shiraz, Iran, October 2017.

[43] M. Pielot, K. Church, and R. De Oliveira, "An in-situ study of mobile phone notifications," in *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services*, pp. 233–242, Toronto, ON, Canada, September 2014.

[44] S. K. Datta, C. Bonnet, and N. Nikaein, "Android power management: current and future trends," in *Proceedings of the First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT)*, pp. 48–53, IEEE, Seoul, South Korea, June 2012.

[45] Google Inc, "Battery monitoring android," 2021, https://developer.android.com/training/monitoring-device-state/battery-monitoring.

[46] D. M. Gezgin, "Understanding patterns for smartphone addiction: age, sleep duration, social network use and fear of missing out," *Cypriot Journal of Educational Sciences*, vol. 13, no. 2, pp. 166–177, 2018.

[47] É. Duke and C. Montag, "Smartphone addiction, daily interruptions and self-reported productivity," *Addictive behaviors reports*, vol. 6, pp. 90–95, 2017.

[48] J. Kuem, S. Ray, P.-F. Hsu, and L. Khansa, "Smartphone addiction and conflict: an incentive-sensitisation perspective of addiction for information systems," *European Journal of Information Systems*, vol. 1, no. 1, pp. 1–22, 2020.

A.8. PUSHED SOLID: DEPLOY SOLID IN SMARTPHONES203

Authors: *Sergio Laso, Javier Berrocal, Pablo Fernández, Antonio Ruiz-Cortés and Juan M Murillo.*

Publication type: *Conference paper (demo paper).*

Conference: *Jornadas de Ciencia e Ingeniería de Servicios (JCIS), Málaga, Spain. 2021.*

Available online: *11705/JCIS/2021/046.*

Perses: Un framework para evaluar la Calidad de Servicio en aplicaciones móviles distribuidas

Sergio Laso[1], Javier Berrocal[1], Pablo Fernández[2], Antonio Ruiz-Cortés[2] and Juan M. Murillo[1]

[1] University of Extremadura, Spain
{slasom, jberolm, juanmamu}@unex.es
[2] University of Sevilla, Spain
{pablofm, aruiz}@us.es

Abstract. Las crecientes capacidades de los dispositivos finales han llevado al despliegue masivo de aplicaciones móviles distribuidas. El éxito de estas aplicaciones depende en gran medida de la Calidad del Servicio (QoS) que ofrecen. Esta calidad es especialmente difícil de evaluar debido al gran número de entidades implicadas y a su heterogeneidad. Las herramientas actuales de evaluación no dan soporte a este tipo de aplicaciones, suelen centrarse en la evaluación de la QoS proporcionada por una sola entidad. Sin embargo, la QoS de las aplicaciones distribuidas no sólo depende de la QoS de cada entidad, también hay que evaluar las interacciones entre las entidades. En este artículo se presenta un framework, denominado Perses, para lanzar entornos virtuales que permitan simular y evaluar la ejecución de aplicaciones móviles distribuidas. Esta simulación proporciona resultados de la QoS alcanzada. Además, el framework se ha integrado en una metodología DevOps para automatizar su ejecución.
Vídeo de presentación— https://youtu.be/wpIApe_sPFE

Keywords: Aplicaciones Móviles Distribuidas; Calidad de Servicio; DevOps; Entorno Virtual

1 Introducción

El éxito o el fracaso de cualquier aplicación móvil depende de dimensiones, como la publicidad, la viralidad o la Calidad del Servicio (QoS) proporcionada [7].Una mala calidad llevará a los usuarios a rechazar rápidamente la aplicación.

Estas aplicaciones se desarrollan normalmente utilizando un estilo arquitectónico cliente-servidor. Los componentes informáticos más exigentes se despliegan normalmente en entornos cloud debido a su gran capacidad de computación y almacenamiento. Los componentes básicos se despliegan en el lado del cliente, ya que sus requisitos de computación suelen ser bajos y pueden ser ejecutados por casi cualquier dispositivo móvil.

Durante los últimos años, las capacidades de computación y almacenamiento de los dispositivos móviles han aumentado significativamente [3], provocando la aparición de nuevos paradigmas centrados en su explotación, como los Microservicios Humanos [8] o Mist Computing [11]. Estos paradigmas permiten a los

desarrolladores cargar algunas funcionalidades de computación en los dispositivos móviles para mejorar la QoS y ser más competitivos a nivel empresarial.

En las aplicaciones distribuidas, tanto el cloud como el lado cliente/móvil tienen componentes de computación y almacenamiento. Por lo tanto, la evaluación de la QoS debe combinar la evaluación conjunta del lado del servidor y del lado móvil, así como de la interacción entre ambos. Sólo la evaluación conjunta de las tres dimensiones permitiría predecir con suficiente precisión la QoS esperada. Además, suele tratarse de entornos heterogéneos (diferentes dispositivos móviles), siendo aún más difícil predecir la QoS esperada.

Actualmente, existen diferentes plataformas que permiten evaluar el comportamiento de aplicaciones móviles no distribuidas en entornos heterogéneos. AWS Device Farm [10] y Azure App Center Test [1] permiten a los desarrolladores probar diferentes dispositivos con diferentes configuraciones. Sin embargo, se centran en la evaluación de tests de interfaz de usuario (mediante Appium[3], Espresso [4], etc.). Por lo tanto, no están diseñadas para medir los atributos de QoS.

En este artículo, presentamos un framework, denominado Perses [9], para evaluar la QoS de las aplicaciones móviles distribuidas. Para ello, permite definir y desplegar entornos virtuales personalizados, con múltiples dispositivos virtuales heterogéneos, en los que los desarrolladores pueden simular el despliegue de la aplicación móvil distribuida y lanzar tests de rendimiento para evaluar diferentes atributos de la QoS.Además, este framework se ha integrado completamente en una metodología DevOps [2], reduciendo el esfuerzo de evaluación de la QoS antes de desplegar la aplicación en producción.

El resto del artículo se estructura de la siguiente forma: La Sección 2 explica las características de Perses y su arquitectura. La Subsección 2.1 describe un caso de uso para la demo. Finalmente, la Sección 3 presenta las conclusiones.

2 Perses

Perses es un framework que permite a los desarrolladores desplegar fácilmente un entorno virtual con múltiples dispositivos virtuales heterogéneos para evaluar la QoS de una aplicación distribuida. Para utilizarlo, sólo hay que definir un *fichero de configuración* con las características del entorno, los tests a ejecutar y los umbrales deseados de los atributos de calidad.

Perses está totalmente integrado en una metodología DevOps, siendo capaz de automatizar todos los diferentes pasos que se deben ejecutar durante el proceso de evaluación.Además es totalmente escalable, permitiendo evaluar aplicaciones con un gran número de dispositivos virtualizados.

La Figura 1 muestra un diagrama general de la arquitectura de Perses. Perses consta de dos componentes: *Perses Launcher*, centrado en la definición y configuración del entorno virtual, y *Perses Virtual Environment*, encargado de crear el entorno heterogéneo y ejecutar los tests definidos. Cada componente tiene diferentes módulos que se explican a continuación:

[3] http://appium.io/
[4] https://developer.android.com/training/testing/espresso

- **Setup:** este módulo se encarga de validar y extraer las caracteristicas del *fichero de configuración* para los demás módulos.
- **Deployment:** este módulo toma como entrada las caracteristicas obtenidas en el módulo anterior y se encarga de crear y desplegar todo el entorno virtual. Este módulo tiene una capa de abstracción que encapsula Terraform [6]. Además, orquesta el entorno virtual instalando los recursos necesarios, creando los dispositivos móviles virtuales y desplegando la aplicación.
- **Tests Execution:** este módulo se encarga de ejecutar y analizar los tests definidos en el *fichero de configuración*. Perses permite ejecutar tests de rendimiento y de interfaz de usuario. Los tests de rendimiento evalúan la QoS de la aplicación utilizando APIPecker [4]. Los tests de interfaz de usuario permiten ejecutar tests desarrollados con Espresso.
- **CI Manager:** este módulo se encarga de la integración con DevOps. Automatiza la ejecución de los diferentes módulos. Esta automatización se realiza a través de un flujo de trabajo definido con GitHub Actions [5].

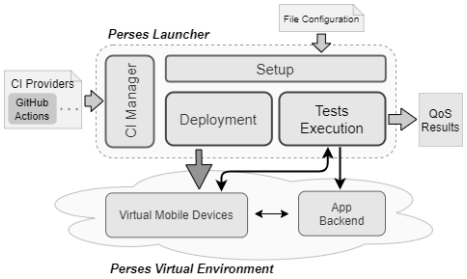


Fig. 1. Arquitectura de Perses.

2.1 Caso de uso de la demo

En esta demostración, mostraremos a los asistentes cómo funciona el marco Perses evaluando la QoS de una aplicación móvil distribuida. Para ello, evaluaremos una aplicación para ayudar a frenar la pandemia del Covid-19. Esta aplicación proporciona el riesgo de contagio de una persona.

La aplicación se compone de un servidor y el lado móvil. El lado del servidor es el punto de acceso para que los usuarios obtengan el porcentaje de riesgo y, por lo tanto, también es responsable de calcular este riesgo. El lado móvil se encarga de guardar la localización de cada usuario y proporcionar al cloud los lugares/regiones en los que ha estado.

3 Conclusiones

La QoS es una de las dimensiones clave para el éxito o el fracaso de cualquier aplicación. En los últimos años, las capacidades de los dispositivos móviles han

aumentado considerablemente, dando lugar a aplicaciones con computación distribuida en las que tanto el cloud como el lado móvil pueden tener componentes de computación y almacenamiento con el objetivo de mejorar aún más la QoS. Sin embargo, esta calidad depende de diferentes entidades y sus interacciones, que deben ser analizadas cuidadosamente.

En este trabajo, presentamos un framework para el despliegue de un entorno virtual personalizado para evaluar la QoS de las aplicaciones móviles distribuidas. Actualmente, trabajamos en el análisis del impacto del uso.

Agradecimientos

Este trabajo ha sido parcialmente financiado por los proyectos RTI2018-094591-B-I00 (MCI/AEI/FEDER,UE), RTI2018-101204-B-C21, el Proyecto 4IE+ (0499-4IE-PLUS-4-E) financiado por Interreg V-A España-Portugal (POCTEP) 2014-2020, por la red RCIS (TIN2016-81978-REDT), por la Consejería de Economía, Ciencia y Agenda Digital de la Junta de Extremadura (GR18112, IB18030), por la Junta de Andalucía (US-1264651) y el Fondo Europeo de Desarrollo Regional.

References

1. Azure, M.: App center test. `https://docs.microsoft.com/en-us/appcenter/test-cloud/`, (Accessed on 28/10/2020)
2. Bass, L., Weber, I., Zhu, L.: DevOps: A software architect's perspective. Addison-Wesley Professional (2015)
3. Berrocal, J., Garcia-Alonso, J., Vicente-Chicote, C., Hernández, J., Mikkonen, T., Canal, C., Murillo, J.M.: Early analysis of resource consumption patterns in mobile applications. Pervasive and Mobile Computing **35**, 32 – 50 (2017). https://doi.org/https://doi.org/10.1016/j.pmcj.2016.06.011, `http://www.sciencedirect.com/science/article/pii/S1574119216300797`
4. Fernandez, P.: API Pecker. `https://www.npmjs.com/package/apipecker`, (Accessed on 28/10/2020)
5. Github: Github actions. `https://github.com/features/actions`, (Accessed on 28/10/2020)
6. HashiCorp: Terraform. `https://www.terraform.io/`, (Accessed on 28/10/2020)
7. Kim, H.J., Lee, D.H., Lee, J.M., Lee, K.H., Lyu, W., Choi, S.G.: The qoe evaluation method through the qos-qoe correlation model. In: 2008 Fourth International Conference on Networked Computing and Advanced Information Management. vol. 2, pp. 719–725 (2008). https://doi.org/10.1109/NCM.2008.202
8. Laso, S., Berrocal, J., García-Alonso, J., Canal, C., Manuel Murillo, J.: Human microservices: A framework for turning humans into service providers. Software: Practice and Experience (2021)
9. Perses-org: Perses. `https://github.com/perses-org/perses`, note = (Accessed on 28/10/2020)
10. Service, A.W.: Aws device farm. `https://aws.amazon.com/device-farm/`, (Accessed on 28/10/2020)
11. Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., Jue, J.P.: All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture (2019)

A.9 Una Propuesta para la Composición de APIs Distribuidas

Authors: *Sergio Laso, David Bandera, Javier Berrocal, José García-Alonso, Juan Manuel Murillo and Carlos Canal.*

Publication type: *Conference paper (long paper).*

Conference: *Jornadas de Ciencia e Ingeniería de Servicios (JCIS), SISTEDES. Málaga, Spain. 2019.*

Available online: *11705/JCIS/2021/043.*

Una Propuesta para la Composición de APIs Distribuidas

Sergio Laso[1], David Bandera[2], Javier Berrocal[1], Jose Garcia-Alonso[1], Juan Manuel Murillo[1] and Carlos Canal[2]

[1] University of Extremadura, Spain
{slasom, jberolm, jgaralo, juanmamu}@unex.es
[2] University of Málaga, Spain
{dbandera, canal}@lcc.uma.es

Abstract. El incremento de las capacidades de computación de distintos dispositivos (elementos de la red, dispositivos finales, etc.) finales ha dado lugar a paradigmas como Fog, Edge, Mist o Crowd computing que tienen como objetivo explotar dichas capacidades para almacenar y procesar información, proporcionándola al entorno mediante APIs y servicios. Esta distribución de la computación permite mejorar la calidad de servicio, sobre todo en entornos con requisitos estrictos. Sin embargo, el uso de APIs y servicios desplegados de forma distribuida conlleva un esfuerzo extra al desarrollador, por la necesidad de controlar y coordinar la invocación a las distintas APIs y los resultados que proporcionan. En este artículo presentamos un compositor de APIs distribuidas (DAC), un sistema el cual permite recopilar y agregar la información de las APIs desplegadas en distintos dispositivos. Con el objetivo de, reducir el esfuerzo de su implementación, se ha definido una extensión de la especificación OpenAPI para facilitar su desarrollo y despliegue.

Keywords: APIs · Agregación · Internet Of Things · Computación Distribuida

1 Introducción

El incremento de las capacidades computacionales tanto de las redes como de los dispositivos finales ha dado lugar a una alta distribución de las aplicaciones IoT y sus correspondientes APIs. Paradigmas como Fog, Edge, Mist o Crowd computing [15] permiten a los desarrolladores replicar diferentes partes de estas APIs en nodos más cercanos al usuario final, o incluso en sus propios dispositivos, aprovechando las capacidades computacionales de estos dispositivos para minimizar el tiempo de respuesta y mejorar la Calidad del Servicio [7]. Esta distribución de la carga computacional permite desplegar aplicaciones IoT con una Calidad del Servicio rigurosa, siendo especialmente necesario en entornos con requisitos de QoS estrictos, como la Industria 4.0. o el sector de sanidad [13].

Para consumir estas APIs distribuidas, es necesario disponer de un único punto de acceso a ellas, ya que no es razonable acceder a cada uno de los dispositivos de forma individual. Esta tarea conlleva una sobrecarga significativa

al desarrollo de sistemas distribuidos. Gestionar este proceso no es trivial y aumenta de forma considerable el tiempo y esfuerzo de desarrollo a la hora de implementar el sistema.

Un ejemplo de estos sistemas son las aplicaciones de crowd sensing [9]. Estas aplicaciones involucran a un gran número de individuos que recogen y extraen datos de forma colectiva de áreas de interés específico. Cada individuo es independiente de los demás, pero tienen la opción de compartir los datos obtenidos para colaborar hacia un objetivo común. En este caso, las APIs están replicadas y distribuidas lo más cercano posible a la fuente de los datos con el objetivo de obtener una gran cantidad de información a la vez que se minimiza la sobrecarga computacional del proceso. Sin embargo, para recopilar todos estos datos hace falta acceder a todos estos dispositivos uno por uno y una vez recopilados, es necesario realizar un post procesado para asegurar la integridad, coherencia y reducir la redundancia de estos datos. Este proceso de gestión de datos debería requerir el mínimo esfuerzo posible para los desarrolladores.

Por ello, en este artículo presentamos una propuesta llamada DAC (Distributed API Composer), que realiza todo el proceso de coordinación de las invocaciones, obtención de los datos y su agregación de las APIs distribuidas. Para ello, hemos diseñado un modelo conceptual para plasmar todos sus conceptos y características. Este modelo nos ha servido de apoyo para desarrollar una extensión del lenguaje OpenAPI [5] con el objetivo de facilitar a los desarrolladores su implementación y despliegue.

Para detallar las aportaciones aquí presentadas, este artículo se estructura de la siguiente forma: La Sección 2 hablamos sobre las motivaciones para realizar esta investigación en este campo y los problemas que pretendemos solventar. La Sección 3 explicamos con detalle nuestra propuesta. En la Sección 4 realizamos un repaso de trabajos relacionados con nuestra propuesta. Finalmente en la Sección 5, presentamos las conclusiones de nuestro trabajo.

2 Motivación

Tal como hemos comentado anteriormente, existen diferentes paradigmas como la computación Fog, Edge o Mist, que permiten explotar las capacidades de los nodos más cercanos a los usuarios finales para almacenar y procesar los datos con el objetivo de mejorar la calidad de la experiencia.

Existen frameworks basados en estos paradigmas [8] que permiten el despliegue de APIs en diferentes dispositivos finales, los cuales se convierten en proveedores de servicios, pudiendo ser consumidos directamente por otros dispositivos conectados a Internet como si estuvieran desplegados en la nube a través de una API.

El principal problema presente al realizar este despliegue distribuido consiste en que para recopilar los datos de cada dispositivo, tendríamos que ir accediendo individualmente a cada uno de ellos para recoger la información. Para mostrar mejor este problema vamos a describir un escenario, el cual se utilizará como ejemplo en el resto del documento para mostrar los beneficios de este trabajo.

Un centro comercial pretende mejorar la experiencia de los usuarios, controlando diferentes aspectos a través de una API desplegada en los dispositivos móviles de los mismos. Esta API está formada por diferentes microservicios que exponen información almacenada en un perfil virtual [2,6] de cada usuario. Este perfil virtual contiene información sobre las preferencias del usuario y los datos que ha recopilado con su dispositivo.

El centro comercial pretende utilizar los siguientes microservicios con el objetivo de personalizar y mejorar la visita de los usuarios:

- Temperatura: este microservicio obtiene la temperatura que el usuario especifica como deseada. Con ella se pretende adaptar la climatización, conociendo las condiciones de los usuarios presentes en diferentes puntos del centro.
- Localización: con la localización pretenden generar mapas de calor e indicar qué zonas son las más concurridas del centro comercial para que los usuarios y personal de seguridad puedan visualizar si existen aglomeraciones, sobretodo para evitar contagios por Covid-19.
- Música: este microservicio expone los géneros más escuchados por el usuario. Con ello, se pretende reproducir los estilos musicales preferidos por los asistentes para que la visita sea más agradable.

Para intentar solventar este problema, sería necesario un sistema, que, de forma coordinada, obtuviese los datos de las APIs de los usuarios y agregarlos para obtener el resultado final (temperatura media, mapa de calor, etc). También tiene que tener en cuenta la limitación y disponibilidad de muchos de los dispositivos finales, debido a las restricciones de seguridad de sus sistemas operativos o a la movilidad de estos (que pueden provocar una conexión intermitente a Internet y cambios en la dirección IP).Por lo tanto, el sistema debe contemplar diferentes condiciones y protocolos en la comunicación para obtener una Calidad del Servicio aceptable.

Sin embargo, nos encontramos con otro problema principal, para desarrollar este sistema, los desarrolladores tienen que utilizar todas sus habilidades y conocimientos para diseñar, implementarlo y desplegarlo. Esto conlleva en primer lugar, un aumento del tiempo y el coste de desarrollo y, en segundo lugar, una reducción de la capacidad de mantenimiento. Por lo tanto, son necesarios mecanismos y herramientas que faciliten el desarrollo de este sistema.

Existen diversas posibles soluciones a este problema. Una solución podría ser la utilización de un API Gateway que agrupe todos los microservicios para que solo exista un único punto de entrada [16]. Sin embargo, no es una solución del todo satisfactoria, puesto que la funcionalidad de un API Gateway consiste en exponer de forma integrada un conjunto de microservicios de diferentes APIs alojadas en lugares diferentes, pero que se integran de una forma lógica para proporcionar una funcionalidad más compleja. En nuestro caso, las APIs que queremos exponer están desplegadas en diferentes dispositivos, pero son todas iguales. Es necesaria una forma de coordinar su invocación y agrupar automáticamente todos los resultados de la misma API.

En la siguiente sección, se explica con detalle las características de nuestra propuesta para recopilar y agregar la información de APIs distribuidas. Además, también presentamos una extensión del lenguaje OpenAPI con el objetivo de facilitar el diseño y la implementación a los desarrolladores. Por último se describe el proceso de desarrollo, tomando como ejemplo el caso de estudio.

3 Distributed API Composer (DAC)

DAC tiene como objetivo principal coordinar la invocación para recuperar y agregar la información de una API final desplegada en multitud de dispositivos.

Para recuperar la información, DAC realiza una invocación de forma paralela hacia las APIs finales desplegadas, estas responden con la información pedida y el DAC recopila y agrega toda la información para exponerla como resultado final. La agregación del DAC no solo se basa en recuperar y exponer la información en bruto sino realizar algún procesamiento para exponerla lo más adaptada posible al resultado final y mejorar el cumplimiento de los requisitos de la Calidad del Servicio. Para facilitar la agregación de los datos, se han definido diferentes operaciones para procesar los datos y que pueden ser reutilizadas entre los diferentes endpoints. Además, también permite definir condiciones en la comunicación con las APIs finales para ofrecer una Calidad del Servicio aceptable.

DAC actúa como una API REST habitual, con sus endpoints y parámetros y la misma forma de invocación. La interfaz de un DAC para una API es la misma que la de la API final para que su invocación y uso sea totalmente transparente para cualquier usuario. Se puede desplegar tanto en el cloud como en nodos Fog y Edge más cercanos a las APIs para ofrecer una mejor Calidad del Servicio o reducir el tráfico de información.

La Figura 1 muestra un esquema general del caso de estudio, desplegando las APIs en los móviles de los usuarios y el DAC para ser utilizados en el centro comercial. Por ejemplo, las entidades que tengan permiso (personal del centro, seguridad o dispositivos IoT), pueden consultar la temperatura media deseada por los usuarios a través del endpoint del DAC, que invoca de forma coordinada las APIs para obtener la temperatura de cada usuario y agrega la información (la media aritmética de temperaturas) para devolver como resultado la temperatura media para controlar el sistema de climatización.

Para desarrollar DAC con todas sus características, en esta sección se presenta en primer lugar el modelo conceptual de este sistema, detallando sus características y limitaciones. En segundo lugar, con el objetivo de facilitar la creación de DAC y la integración con las herramientas comunes de desarrollo de microservicios, se ha extendido el estándar OpenAPI para dar soporte a los nuevos conceptos definidos. En último lugar se detalla el proceso de implementación y despliegue, desarrollando el caso de estudio mencionado en la sección anterior.

3.1 Modelo conceptual de DAC

Esta sección muestra los diferentes conceptos y características necesarios para definir un Compositor de APIs distribuidas. Así, como las relaciones entre los

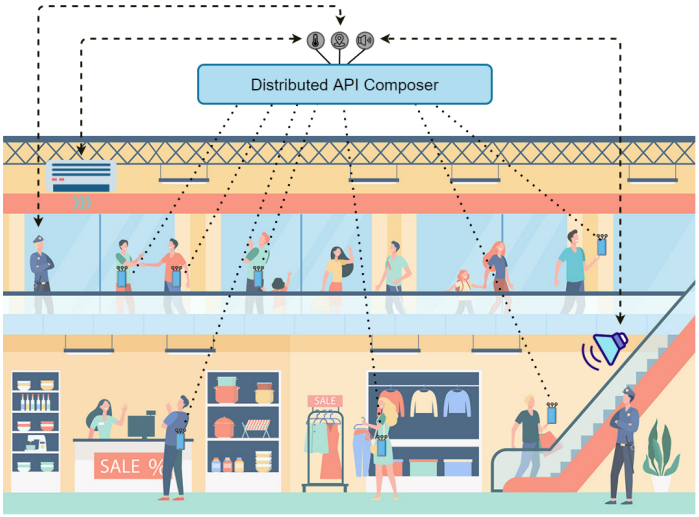


Fig. 1. Caso de estudio: Despliegue de DAC y APIs en un centro comercial

distintos conceptos. La Figura 2 muestra el modelo conceptual diseñado para mostrar dichos conceptos y relaciones.

DAC: es el elemento principal del modelo, puede soportar un protocolo de comunicación y está compuesto por varios endpoints.

- Communication Protocol: permite definir el protocolo concreto que será utilizado para interaccionar con la API que se despliega en los dispositivos finales. Actualmente, DAC implementa el protocolo de comunicación MQTT [10].
- Endpoint: es la interfaz de interacción con DAC, define los datos que se intercambian de entrada y salida. Contiene los mismos endpoints que las APIs agregadas, con la diferencia en que su funcionalidad se basa en agregar las respuestas de dichas APIs.
  - Data Object: son los tipos de datos que se utilizan para la entrada y salida de los endpoints. Estos datos pueden ser utilizados como operandos o estar compuestos de ellos si son más complejos.
  - Operand: permite indicar a qué dato se le quiere aplicar la regla de agregación en un endpoint.
  - Operator: permite definir las distintas reglas que serán utilizadas para agregar la información al invocar el mismo endpoint de las APIs distribuidas. Actualmente, DAC soporta distintos tipos de operadores:
    - *Logical Operator:* se trata de operadores lógicos básicos que se pueden aplicar sobre el operando. Por ejemplo, AND, OR, etc.
    - *Math Operator:* se trata de operadores básicos que se pueden aplicar sobre el operando. Por ejemplo, media aritmética, suma, etc.

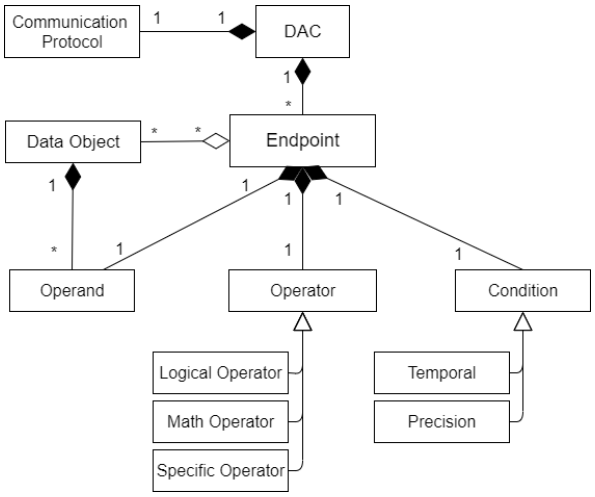


Fig. 2. DAC - Modelo conceptual

  - ∗ *Specific Operator:* se trata de operadores que realizan un procesamiento más complejo y no se pueden realizar con los operadores anteriores. El desarrollador puede implementar sus propias funciones de agregación.
- Conditions: permite definir las condiciones en la comunicación para ofrecer una Calidad del Servicio aceptable. Se pueden definir dos:
  - ∗ *Temporal:* permite definir un tiempo máximo de respuesta (en milisegundos) para finalizar la comunicación. Si no se conoce el número de dispositivos conectados, es importante definir esta condición para finalizar la recopilación de datos y pasar a la agregación.
  - ∗ *Precisión:* permite definir un número máximo de respuestas que va a recibir de las APIs. Pueden existir endpoints en los que no sea necesario recopilar la información de todos los dispositivos, sino una muestra representativa, permitiendo reducir el tiempo de respuesta.

3.2 Integración de DAC con OpenAPI

OpenAPI es un estándar ampliamente utilizado en la industria que permite diseñar y documentar APIs (conjuntos de microservicios) tanto para el cloud como para dispositivos finales [8] que pueden utilizarse para generar código fuente, ahorrando tiempo de desarrollo. Actualmente, con la especificación existente no tenemos las capacidades necesarias para definir las propiedades de DAC. También es un lenguaje extensible, permite describir funcionalidades adicionales que no están cubiertas por la especificación. Por ello, en esta subsección mostramos la creación de una extensión que permite definir las distintas operaciones y condiciones del DAC para cada endpoint.

Para describir nuevas propiedades en la especificación OpenAPI se insertan campos precedidos de *"x-"*, es la nomenclatura definida por el estándar para añadir las extensiones sin impactar en las herramientas de validación y generación de código ya existentes. La extensión se ha definido siguiendo el modelo conceptual. Así, para definir un DAC, se debe declarar usando la etiqueta *"x-dac"* en cada endpoint. Dentro de la etiqueta se especifica el resto de los conceptos definidos en el modelo.

Siguiendo con el caso de estudio, para definir los microservicios descritos, los definimos con la especificación OpenAPI, que será la misma para definir tanto el DAC como la API, pero añadiendo también la extensión .

El listado 1.1 muestra parte de la definición de la especificación donde se define un microservicio del caso de estudio. El DAC recibe de las APIs un objeto llamado User, en el DAC se indica que en la propiedad *User.temperature* se aplique el operador matemático de la media aritmética. Como condiciones de la comunicación, se espera recibir la información de 200 dispositivos sin que supere un máximo de 1500 milisegundos.

Listing 1.1. Extensión de OpenAPI para la definición del DAC.

```
{   paths:
      /temperature:
        get:
          summary: Gets the environment temperature
          operationId: getTemperature
          tags:
          - User
          x-dac:
            operand: User.temperature
            operator:
              type: math
              name: AVG
            conditions:
              precision: 200
              temporal: 1500
          responses:
        ...    }
```

Por último, existen diferentes generadores como OpenAPI Generator [4] o APIGEND [14] que permiten generar el código fuente tanto para el lado cloud, como para el lado cliente en diferentes lenguajes y dispositivos finales, a través de una especificación OpenAPI. Sin embargo, no soportan la generación del DAC. Por ello, también ha sido necesario extender, en este caso, el generador APIGEND para que soporte la creación de DAC.

3.3 Proceso de despliegue

En esta última subsección explicamos todo el proceso para desplegar la aplicación IoT del caso de estudio. A continuación, se enumeran los distintos pasos:

1. **Diseño de la API:** En primer lugar, hay que diseñar la API con la especificación OpenAPI. La definición sigue la misma notación y lógica que si la desplegáramos en el cloud, definiendo los microservicios que estarán disponibles para su consumo. Además, hay que incluir la extensión definida

para poder generar el DAC. Esta especificación será válida tanto para generar el código para las APIs de los dispositivos finales como para generar el DAC. En GitHub [3] se encuentra la especificación OpenAPI del caso de estudio.
2. **Generación del código fuente:** En segundo lugar, se genera el código fuente de las distintas partes de la aplicación (API y el DAC). El generador APIGEND soporta la generación de ambas entidades. En la sección de *Servers* del generador, se indica la URL de la especificación en el campo *openAPIUrl* y en el lenguaje, se selecciona *Android-Server* para generar la API para los dispositivos finales y *DAC-Node* para generar el DAC.
3. **Despliegue:** En tercer lugar, para poder desplegar las APIs y el DAC, los desarrolladores deben finalizar la implementación. En ambas entidades es necesario configurar el protocolo de comunicaciones MQTT para su interconexión. Además, en la API, tienen que añadir la lógica de negocio a los microservicios. En el DAC también tienen que añadir la lógica de negocio en aquellos microservicios donde se hayan definido *Specific Operator*.
4. **Invocación de servicios:** Finalmente, con el DAC desplegado, la invocación de los microservicios se realiza de la misma manera que una API REST desplegada en el cloud. Por cada invocación que reciba, el DAC se encargará en segundo plano de realizar la invocación de las APIs distribuidas.

4 Trabajos relacionados

El auge de los dispositivos IoT ofrece oportunidades de investigación en el campo de microservicios y crowd sensing, especialmente con la aparición de la tecnología de comunicaciones 5G. Este elevado número de dispositivos aumenta la cantidad de datos generados. En trabajos como [11] y [1], se comenta la necesidad de gestionar esta cantidad de información de forma inteligente, ya que las comunicaciones inalámbricas conllevan un consume energético importante. Por ello, desarrollan la idea de asignar dispositivos como agregadores, los cuales se encargan de agrupar información generada por dispositivos locales cercanos, y enviarla a un nodo principal de forma conjunta. Esta idea es similar a nuestra propuesta. Sin embargo, estos trabajos están enfocados a la optimización del consumo de energía y tráfico en redes de comunicaciones, en contraste con nuestra idea de ofrecer herramientas para facilitar el proceso de agregación de datos desde el punto de vista del desarrollo de los sistemas.

Otros trabajos relacionados con agregación en IoT se centran en la identificación de los dispositivos en los que se despliegan las APIS. En trabajos como [17], los autores proponen una arquitectura de agregación de la información en entornos IoT. Esta información está centrada en las características e identificación de los dispositivos en sí, y no de la información generada por las aplicaciones ejecutadas por estos dispositivos.

En artículos como [3], [12] se habla de la gestión de aplicaciones en plataformas PaaS heterogéneas. Utilizando una interfaz middleware llamada COAPS API, se abstrae la implementación concreta de las distintas plataformas PaaS

[3] https://github.com/slasom/JCIS2021/blob/main/shopping-center.yaml

para ofrecer a los desarrolladores una forma unificada de desplegar y gestionar sus aplicaciones, independientemente de la plataforma elegida. Esta idea se alinea con nuestra intención de ofrecer herramientas que faciliten la implementación y despliegue de APIs distribuidas. Sin embargo, está centrada en la computación en la nube. Nuestro trabajo tiene un enfoque centrado en la computación IoT, cuyos nodos se encuentran altamente distribuidos.

5 Conclusiones

Durante los últimos años, el despliegue de dispositivos móviles e IoT ha aumentado considerablemente, dando lugar a paradigmas como Fog, Edge o Mist computing que permiten explotar sus capacidades con el objetivo de proporcionar arquitecturas distribuidas a aplicaciones y servicios que requieren de unos requisitos estrictos de calidad de servicio. El consumo de estas aplicaciones y servicios con arquitecturas tradicionales está altamente superditado por APIs, algo que se está también extendiendo hacia las nuevas arquitecturas distribuidas. Sin embargo, supone una tarea compleja consumir estas APIs para obtener y computar la información que está distribuida en varios nodos.

En este artículo hemos presentado DAC, un compositor de APIs distribuidas. Este sistema tiene como objetivo coordinar la invocación hacia esas APIs para obtener y agregar la información de cada una de ellas. Para agregar la información, DAC permite definir diferentes tipos de operaciones que pueden ser reutilizadas. Su interfaz es igual a las APIs distribuidas, por lo que no cambia su forma de uso para los usuarios que lo utilicen. Además, también presentamos un conjunto de mecanismos y procesos que facilitan el desarrollo del DAC, utilizando herramientas comunes de desarrollo de microservicios.

Actualmente estamos trabajando, primero, en mejorar y evolucionar las funciones del DAC, y segundo, en la realización de experimentos para comparar con sistemas tradicionales y conocer la percepción de analistas y desarrolladores con respecto al uso de este sistema.

Agradecimientos

Este trabajo ha sido parcialmente financiado por los proyectos RTI2018-094591-B-I00, PGC2018-094905-B-I00 (MCI/AEI/FEDER,UE), la red de investigación RCIS (RED2018-102654-T), el proyecto 4IE+ (0499-4IE-PLUS-4-E) financiado por el programa Interreg V-A España-Portugal (POCTEP) 2014-2020, por el proyecto UMA18-FEDERJA-180 (FEDER/Junta de Andalucia), por la Consejería de Economía, Ciencia y Agenda Digital de la Junta de Extremadura (GR18112, IB18030), y por el Fondo Europeo de Desarrollo Regional.

References

1. Dehkordi, S.A., Farajzadeh, K., Rezazadeh, J., Farahbakhsh, R., Sandrasegaran, K., Dehkordi, M.A.: A survey on data aggregation techniques in iot sensor networks. Wireless Networks **26**(2), 1243–1263 (2020)

2. Guillen, J., Miranda, J., Berrocal, J., Garcia-Alonso, J., Murillo, J.M., Canal, C.: People as a service: a mobile-centric model for providing collective sociological profiles. IEEE software **31**(2), 48–53 (2013)
3. Hossny, E., Khattab, S., Omara, F., Hassan, H.: A case study for deploying applications on heterogeneous paas platforms. In: 2013 International Conference on Cloud Computing and Big Data. pp. 246–253 (2013). https://doi.org/10.1109/CLOUDCOM-ASIA.2013.13
4. Initiative, O.: Openapi Generator, https://github.com/OpenAPITools/openapi-generator. Accessed April 10, 2021
5. Initiative, O.: The OpenAPI Specification Repository. Contribute to OAI/OpenAPI-Specification development by creating an account on GitHub, https://github.com/OAI/OpenAPI-Specification, https://github.com/OAI/OpenAPI-Specification. Accessed March 3, 2020
6. Jesús-Azabal, M., Berrocal, J., Laso, S., Murillo, J.M., Garcia-Alonso, J.: Solid and peaas: Your phone as a store for personal data. In: International Conference on Web Engineering. pp. 5–10. Springer (2020)
7. Kim, H.J., Lee, D.H., Lee, J.M., Lee, K.H., Lyu, W., Choi, S.G.: The qoe evaluation method through the qos-qoe correlation model. In: 2008 Fourth International Conference on Networked Computing and Advanced Information Management. vol. 2, pp. 719–725 (2008). https://doi.org/10.1109/NCM.2008.202
8. Laso, S., Berrocal, J., García-Alonso, J., Canal, C., Manuel Murillo, J.: Human microservices: A framework for turning humans into service providers. Software: Practice and Experience (2021)
9. Marjanović, M., Antonić, A., Žarko, I.P.: Edge computing architecture for mobile crowdsensing. IEEE Access **6**, 10662–10674 (2018)
10. MQTT: MQTT, http://mqtt.org/, http://mqtt.org/. Accessed March 20, 2021
11. Orsino, A., Araniti, G., Militano, L., Alonso-Zarate, J., Molinaro, A., Iera, A.: Energy efficient iot data collection in smart cities exploiting d2d communications. Sensors **16**(6) (2016). https://doi.org/10.3390/s16060836, https://www.mdpi.com/1424-8220/16/6/836
12. Sellami, M., Yangui, S., Mohamed, M., Tata, S.: Paas-independent provisioning and management of applications in the cloud. In: 2013 IEEE Sixth International Conference on Cloud Computing. pp. 693–700. IEEE (2013)
13. Shukla, S., Hassan, M.F., Jung, L.T., Awang, A.: Architecture for latency reduction in healthcare internet-of-things using reinforcement learning and fuzzy based fog computing. In: International Conference of Reliable Information and Communication Technology. pp. 372–383. Springer (2018)
14. SPILAB: APIGEND - API Generator for End Devices, https://openapi-generator-spilab.herokuapp.com. Accessed April 10, 2021
15. Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., Jue, J.P.: All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture (2019)
16. Zhao, J., Jing, S., Jiang, L.: Management of api gateway based on micro-service architecture. In: Journal of Physics: Conference Series. vol. 1087, p. 032032. IOP Publishing (2018)
17. Zhu, T., Dhelim, S., Zhou, Z., Yang, S., Ning, H.: An architecture for aggregating information from distributed data nodes for industrial internet of things. In: Cyber-Enabled Intelligence, pp. 17–35. Taylor & Francis (2019)

A.10 Sistema IoT para la Prevención de Riesgos Laborales en una EDAR

Authors: *Sergio Laso, Daniel Flores-Martín, Juan Pedro Cortés-Pérez, Miguel Soriano Barroso, Alfonso Cortés-Pérez, Javier Berrocal and Juan M. Murillo.*

Publication type: *Conference paper (long paper).*

Conference: *Jornadas de Ciencia e Ingeniería de Servicios (JCIS), SISTEDES. Santiago de Compostela, Spain. 2022.*

Available online: *11705/JCIS/2022/019.*

Sistema IoT para la Prevención de Riesgos Laborales en una EDAR

Sergio Laso[1], Daniel Flores-Martin[2], Juan Pedro Cortés-Pérez[2], Miguel Soriano Barroso[3], Alfonso Cortés-Pérez[4], Javier Berrocal[2], y Juan M. Murillo[2]

[1] Global Process and Product Improvement S.L.
slasom@unex.es
[2] Universidad de Extremadura
Badajoz, España
{dfloresm,jpcortes,jberolm,juanmamu}@unex.es
[3] INYGES CONSULTORES S.L.
msoriano@inyges.es
[4] AC2 INNOVACIÓN S.L.
info@ac2sc.es

Abstract. Los peligros biológicos y el riesgo de ruidos en las estaciones depuradoras de aguas residuales resultan una preocupación real. Estas estaciones generan riesgos de inhalación de gases y exposición a ruidos elevados peligrosos. Las estaciones cuentan con sensores que son capaces de monitorizar los niveles de gases en el ambiente y el nivel de ruido. Esto no es suficiente y es necesario la geolocalización de los operarios. Sin embargo, la geolocalización en interiores sigue siendo un problema debido a precisión limitada del GPS. Existen alternativas como el bluetooth, que permiten obtener una geolocalización más exacta. En este trabajo se presenta la demostración de un sistema IoT basado en una aplicación móvil que permite geolocalizar a los operarios de una estación de aguas residuales a través de balizas bluetooth y cruzarlo con la información de los sensores de gases y ruidos para prevenir riesgos laborales.
Vídeo de presentación— https://youtu.be/wRDxQF7gqkk

Keywords: Monitorización; Beacons; EDAR; IoT; Prevención de riesgos laborales

1 Introducción

Las estaciones depuradoras de aguas residuales (EDAR) son infraestructuras que velan por la correcta conservación del medio ambiente. Estas estaciones se encargan de filtrar los residuos de las aguas utilizadas por los seres humanos.

Las EDAR implican diferentes peligros para las personas que trabajan allí, por la cantidad de carga biológica existente, gases tóxicos, o el ruido de los equipos [2]. Estos peligros son mitigados y monitorizados por equipos expertos de prevención de riesgos laborales. Además de las medidas de seguridad tradicionales como el uso de casco, guantes, etc., también existen diferentes tipos de sensores de presión del agua, de CO2, o de ruido, para controlar no superen

niveles elevados que sean peligrosos. La exposición a estos elementos solo es perjudicial cuando se superan determinados niveles y tiempos. Por lo tanto, para prevenir los riesgos de las personas, es necesario conocer la localización para comprobar si han estado en el área donde actúa un sensor y por cuanto tiempo.

La localización en interiores supone un problema debido a la baja precisión que proporcionan las tecnologías basadas en GPS. Como alternativa, existen otras tecnologías que permiten esta localización de manera más precisa, como las basadas en bluetooth y dispositivos BLE [1]. La geolocalización por bluetooth permite triangular la posición de una persona en función de los dispositivos bluetooth que tenga cerca, permitiendo así una localización más precisa.

En esta demostración presentamos un sistema IoT que ayuda a la automatización de la prevención de riesgos laborales en la EDAR de Coria, Extremadura [1].

Por un lado, se utilizan técnicas existentes de geolocalización en interiores a través de bluetooth en base a balizas electrónicas bluetooth (beacons) colocadas estratégicamente en la estación depuradora y, por otro, se monitorizan los niveles de gases tóxicos y ruidos por las zonas donde se mueven los operarios. Esto se consigue a través de una aplicación móvil desarrollada para monitorizar la actividad de los operarios de manera automática para alertar ante posibles niveles de peligros biológicos, denominada **DSIndoorLocation**. De esta forma, se consiguen prevenir ciertos riesgos de manera automática.s

El resto del documento se estructura de la siguiente manera: la Sección 2 detalla los aspectos más importantes de la aplicación DSIndoorLocation y un caso de ejemplo. Y la Sección 3 presenta las conclusiones de este trabajo.

2 Aplicación DSIndoorLocation

La aplicación DSIndoorLocation geolocaliza a los operarios junto con el control de los sensores para evitar posibles riesgos que se pueden producir en la planta.

En primer lugar, se realizó un análisis de la cobertura de los beacons, que después se analizó en el plano de la planta para colocarlos de forma estratégica para que cubriese todo el área de la planta. En la Figura 1 se muestra la colocación de los beacons con una distancia entre ellas no superior a 10 metros tras los análisis de su cobertura. En segundo lugar, se han colocado sensores de gases y ruidos en zonas especificas de la planta recomendadas por los expertos de la planta. Los valores obtenidos de los sensores son enviados al Cloud donde serán leídos posteriormente por la aplicación móvil.

Para obtener la localización, la aplicación utiliza la librería de neXenio [3], que utiliza el algoritmo de Multilateración para triangular la localización a partir de las distancias con respecto a los beacons y su ubicación. Por lo que es necesario que se detecten al menos 3 beacons (colocados estratégicamente en la planta) y conocer su ubicación. Para ello, la aplicación permite gestionar los beacons, permitiendo añadir y editar su información (identificador y ubicación).

La interfaz de la aplicación es simple. La Figura 2a muestra la interfaz principal compuesta por el mapa de la planta, la ubicación de los beacons (cuadrados

[1] `https://goo.gl/maps/cCJyoGCWz6ikjc8W6`

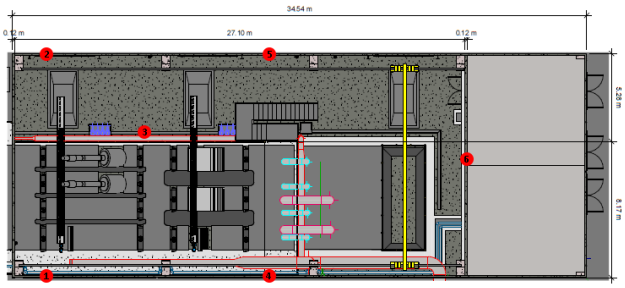


Fig. 1: Ubicación de los beacons en la planta EDAR

grises) y la del propio operario (círculo naranja) en tiempo real. En la parte inferior hay dos botones, "Comenzar/Finalizar" y "Salir". El botón "Comenzar/Finalizar" se utiliza para que el operario lo pulse cuando vaya a empezar o terminar sus labores de trabajo. El botón "Salir" detiene todos los servicios que se estén ejecutando en segundo plano y finaliza la aplicación.

El funcionamiento de la aplicación es el siguiente. Cuando el operario comienza sus labores de trabajo y pulsa el botón "Comenzar", la aplicación comienza a monitorizar su localización, a la vez que los valores de los sensores. Este proceso lo realiza en segundo plano por lo que el operario no tiene que interactuar más con el dispositivo hasta que finalice sus labores. Cuando el operario se acerque a un sensor con niveles elevados, la aplicación mostrará una notificación informando al operario de dicha situación como se muestra en la Figura 2b.

2.1 Caso de uso de la demo

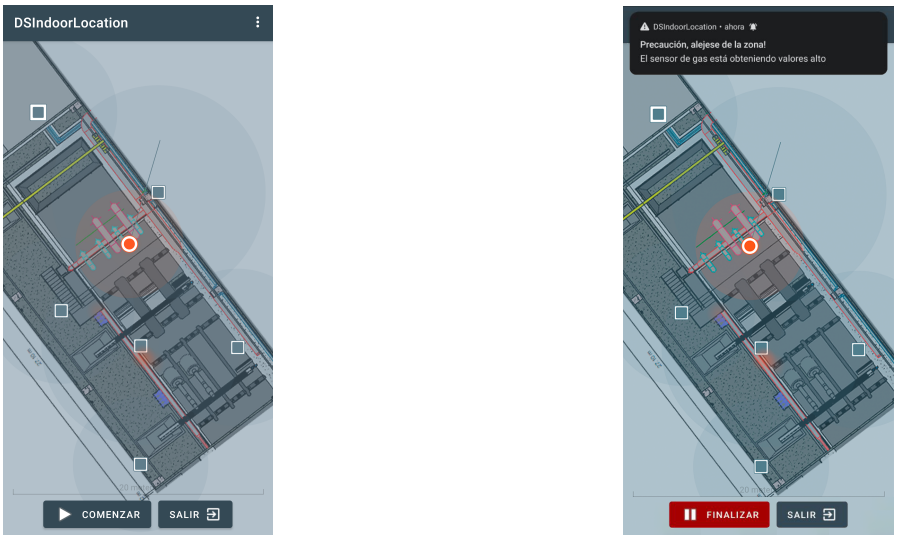
En esta demostración, mostraremos a los asistentes cómo funciona la aplicación DSIndoorLocation. Además mostraremos un caso de uso real donde se monitorizan las labores de un operario de la planta de la EDAR.

Para ello, se monitoriza el recorrido de un operario el cuál va realizando sus tareas de mantenimiento en la planta a la vez que va pasando por los diferentes sensores de gases. Gracias a la geolocalización, se pueden detectar situaciones en las que los sensores por donde pasa han detectado niveles altos, ayudando al operario a evitar una posible intoxicación.

El tiempo de desarrollo de la aplicación ha durado 2 meses y actualmente se encuentra en fase de pruebas con los operarios para afinar la precisión.

3 Conclusiones

En este trabajo, presentamos un sistema IoT que permite la automatización y control de la prevención de riesgos laborales en la EDAR de Coria, Extremadura. Este sistema permite monitorizar a los operarios de la planta a través de la geolocalización en interiores a la vez que se vigila el nivel de diferentes sensores



(a) Pantalla principal (b) Alerta de la aplicación

Fig. 2: Aplicación móvil DSIndoorLocation

de gases tóxicos por donde trabajan, alertando en los casos de posibles niveles de peligro. Actualmente, estamos trabajando en ampliar este sistema extendiendo la monitorización de los operarios también en el exterior de la planta.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la ayuda DIN2020-011586, financiada por MCIN/AEI/10.13039/501100011033 y por la Unión Europea "Next GenerationEU/PRTR", por el Ministerio de Ciencia, Innovación y Universidades (proyecto RTI2018-094591-B-I00 y por la ayuda FPU17/02251), por el proyecto 4IE+ (0499-4IE-PLUS-4-E) financiado por el programa Interreg V-A España-Portugal (POCTEP) 2014-2020, por la Consejería de Economía, Ciencia y Agenda Digital de la Junta de Extremadura (GR21133, IB18030) y el Fondo Europeo de Desarrollo Regional.

References

1. Bai, L., Ciravegna, F., Bond, R., Mulvenna, M.: A low cost indoor positioning system using bluetooth low energy. Ieee Access **8**, 136858–136871 (2020)
2. Granados, I.C.G.: Generación, caracterización y tratamiento de lodos de EDAR. Ph.D. thesis, Universidad de Córdoba (2015)
3. neXenio: BLE Indoor Positioning. `https://github.com/neXenio/BLE-Indoor-Positioning`, (Accessed on 25/04/2022)

Bibliography

[1] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad hoc networks*, 10(7):1497–1516, 2012.

[2] Transforma Insights. Number of internet of things (iot) connected devices worldwide from 2019 to 2021, with forecasts from 2022 to 2030 (in billions).

[3] Mohammad Hasan. State of iot 2022: Number of connected iot devices growing 18% to 14.4 billion globally.

[4] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. In *Present and ulterior software engineering*, pages 195–216. Springer, 2017.

[5] Yi Wei and M Brian Blake. Service-oriented computing and cloud computing: Challenges and opportunities. *IEEE Internet Computing*, 14(6):72–75, 2010.

[6] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[7] Biljana L Risteska Stojkoska and Kire V Trivodaliev. A review of internet of things for smart home: Challenges and solutions. *Journal of Cleaner Production*, 140:1454–1464, 2017.

[8] Ruozhou Yu, Guoliang Xue, and Xiang Zhang. Application provisioning in fog computing-enabled internet-of-things: A network perspective. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 783–791. IEEE, 2018.

[9] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 2019.

[10] Hoang T Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, 13(18):1587–1611, 2013.

[11] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358, 2017.

[12] Malika Bendechache, Sergej Svorobej, Patricia Takako Endo, and Theo Lynn. Simulating resource management across the cloud-to-thing continuum: A survey and future directions. *Future Internet*, 12(6):95, 2020.

[13] S. Laso, J. Berrocal, J. García-Alonso, C. Canal, and J. Manuel Murillo. Human microservices: A framework for turning humans into service providers. *Software - Practice and Experience*, 2021.

[14] Sergio Laso, Javier Berrocal, Pablo Fernández, Antonio Ruiz-Cortés, and Juan M Murillo. Perses: A framework for the continuous evaluation of the QoS of distributed mobile applications. *Pervasive and Mobile Computing*, 84:101627, 2022.

[15] Abhijit Sen. Devops, devsecops, aiops-paradigms to it operations. In *Evolving technologies for computing, communication and smart world*, pages 211–221. Springer, 2021.

[16] Joan Ernst Van Aken. Management research as a design science: Articulating the research products of mode 2 knowledge production in management. *British journal of management*, 16(1):19–36, 2005.

[17] Vijay Vaishnavi and William Kuechler. Design research in information systems. 2004.

[18] Ken Peffers, Marcus Rothenberger, Tuure Tuunanen, and Reza Vaezi. Design science research evaluation. In *International Conference on Design Science Research in Information Systems*, pages 398–410. Springer, 2012.

[19] Paul Johannesson and Erik Perjons. *An introduction to design science*. Springer, 2014.

[20] APIGenerator for End Devices (APIGEND). https://openapi-generator-spilab.herokuapp.com. Last accessed September 2022.

[21] PERSES. https://github.com/perses-org/perses. Last accessed September 2022.

[22] Alan R Hevner. A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4, 2007.

[23] Statista. Mobile internet usage worldwide, 2019. `https://www.statista.com/study/21391/mobile-internet-usage-statista-dossier/`. Accessed October 26, 2022.

[24] Sergio Laso, Daniel Flores-Martin, Juan Luis Herrera, Carlos Canal, Juan Manuel Murillo, and Javier Berrocal. Providing support to iot devices deployed in disconnected rural environment. In *International Workshop on Gerontechnology*, pages 140–150. Springer, 2019.

[25] Sergio Laso, Marino Linaje, Jose Garcia-Alonso, Juan M Murillo, and Javier Berrocal. Artifact Abstract: Deployment of APIs on Android Mobile Devices and Microcontrollers. In *2020 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–2, 2020.

[26] S Laso, M Linaje, J Garcia-Alonso, J M Murillo, and J Berrocal. Deployment of APIs on Android Mobile Devices and Microcontrollers. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2020*, 2020.

[27] Minna Ruckenstein and Julia Granroth. Algorithms, advertising and the intimacy of surveillance. *Journal of Cultural Economy*, 13(1):12–24, 2020.

[28] Chaoyue Niu, Zhenzhe Zheng, Fan Wu, Shaojie Tang, Xiaofeng Gao, and Guihai Chen. Unlocking the value of privacy: Trading aggregate statistics over private correlated data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2031–2040, 2018.

[29] Jagdish Sheth. New areas of research in marketing strategy, consumer behavior, and marketing analytics: the future is bright. *Journal of Marketing Theory and Practice*, 29(1):3–12, 2021.

[30] MIT. SOLID. https://solidproject.org/. Last accessed 2022/10/02.

[31] Manuel Jesús-Azabal, Javier Berrocal, Sergio Laso, Juan Manuel Murillo, and Jose Garcia-Alonso. SOLID and PeaaS: Your Phone as a Store for Personal Data. In *International Conference on Web Engineering*, pages 5–10. Springer, Cham, 2020.

[32] Manuel Jesús-Azabal, Enrique Moguel, Sergio Laso, Juan Manuel Murillo, Jaime Galán-Jiménez, and José Garcia-Alonso. Pushed SOLID: Deploying SOLID in Smartphones. *Mobile Information Systems*, 2021, 2021.

[33] Martina Marjanović, Aleksandar Antonić, and Ivana Podnar Žarko. Edge computing architecture for mobile crowdsensing. *IEEE Access*, 6:10662–10674, 2018.

[34] The OpenAPI Specification Repository. Contribute to OAI/OpenAPI-Specification development by creating an account on GitHub.

[35] Sergio Laso, David Bandera, Javier Berrocal, Jose Garcia-Alonso, Juan Manuel Murillo, and Carlos Canal. Una Propuesta para la Composición de APIs Distribuidas. *Jornadas de Ciencia e Ingeniería de Servicios (JCIS), SISTEDES*, 2021.

[36] Winnie Ng Picoto, Ricardo Duarte, and Inês Pinto. Uncovering top-ranking factors for mobile apps through a multimethod approach. *Journal of Business Research*, 101:668–674, 2019.

[37] Bruno Lima. Automated scenario-based integration testing of distributed systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 956–958, 2018.

[38] Sergio Laso, Javier Berrocal, Pablo Fernández, Antonio Ruiz-Cortés, and Juan M Murillo. Artifact: Virtual Environment for Evaluating the QoS of Distributed Mobile Applications. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 440–441. IEEE, 2021.

[39] Sergio Laso, Javier Berrocal, Pablo Fernández, Antonio Ruiz-Cortés, and Juan M. Murillo. Virtual environment for evaluating the qos of distributed mobile applications. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 407–409, 2021.

[40] Sergio Laso, Javier Berrocal, Pablo Fernández, Antonio Ruiz-Cortés, and Juan M Murillo. Perses: Un framework para evaluar la Calidad de Servicio en aplicaciones móviles distribuidas. *Jornadas de Ciencia e Ingeniería de Servicios (JCIS), SISTEDES*, 2021.

[41] Dinesh Kumar, Ashish Kumar Maurya, and Gaurav Baranwal. Iot services in healthcare industry with fog/edge and cloud computing. In *IoT-Based Data Analytics for the Healthcare Industry*, pages 81–103. Elsevier, 2021.

[42] Manuel Jesús-Azabal, Juan Luis Herrera, Sergio Laso, and Jaime Galán-Jiménez. OPPNets and Rural Areas: An Opportunistic Solution for Remote Communications. *Wireless Communications and Mobile Computing*, 2021, 2021.

[43] Sergio Laso, Javier Berrocal, Pablo Fernandez, José María García, Jose Garcia-Alonso, Juan M. Murillo, Antonio Ruiz-Cortés, and Schahram Dustdar. Elastic data analytics for the cloud-to-things continuum. *IEEE Internet Computing*, 26(6):42–49, 2022.

[44] GII-GRIN-SCIE (GGS) Conference Rating. `https://scie.lcc.uma.es/`. Accessed September 21, 2022.

[45] Daniel Flores-Martín Sergio Laso Jose García-Alonso, Javier Berrocal, Juan M. murillo. Estimación y Generación Temprana de Aplicaciones para la IoT. In *Jornadas de Ciencia e Ingeniería de Servicios (JCIS), SISTEDES*, 2019.

[46] Enrique Moguel, José Garcia-Alonso, and Sergio Laso. Yourpantry: food monitoring through pantry analysis using the smartphone and making use machine learning and deep learning techniques. *Communications in Computer and Information Science*, 1185:3–10, 2019.

[47] Sergio Laso, Daniel Flores, Jose Garcia-Alonso, Juan Manuel Murillo, and Javier Berrocal. Deploying APIs: Edge vs Cloud Environments. *MMTC Communications-Frontiers*, 2019.

[48] Daniel Flores-Martin, Sergio Laso, Javier Berrocal, Carlos Canal, and Juan M Murillo. Allowing IoT devices collaboration to help elderly in their daily lives. In *International Workshop on Gerontechnology*, pages 111–122. Springer, Cham, 2019.

[49] Daniel Flores-Martin, Sergio Laso, Javier Berrocal, and Juan M Murillo. Detecting and Monitoring Depression Symptoms According to People's Behaviour Through Mobile Devices. In *International Workshop on Gerontechnology*, pages 3–10. Springer, Cham, 2020.

[50] Beatriz Sainz-De-Abajo, José Manuel Garcia-Alonso, José Javier Berrocal-Olmeda, Sergio Laso-Mangas, and Isabel De La Torre-Diez. FoodScan: Food Monitoring App by Scanning the Groceries Receipts. *IEEE Access*, 8:227915–227924, 2020.

[51] Sergio Laso, Javier Berrocal, José Garcia-Alonso, Carlos Canal, and Juan M Murillo. Service Oriented Computing for Humans as Service Providers. In *Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing of the Future*, pages 111–122. Springer, Cham, 2021.

[52] Daniel Flores-Martin, Sergio Laso, Javier Berrocal, and Juan M Murillo. Detección de Comportamientos Anómalos Basados en la Utilización del Smartphone. *Jornadas de Ciencia e Ingeniería de Servicios (JCIS), SISTEDES*, 2022.

[53] Sergio Laso, Daniel Flores-Martin, Juan Pedro Cortes-Perez, Miguel Soriano Barroso, Alfonso Cortes-Perez, Javier Berrocal, and Juan M Murillo. Sistema iot para la prevencion de riesgos laborales en una edar. *Jornadas de Ciencia e Ingeniería de Servicios (JCIS), SISTEDES*, 2022.

[54] Daniel Flores-Martin, Sergio Laso, Javier Berrocal, and Juan M Murillo. Contigo: Monitoring People's Activity App for Anomalies Detection. In *International Workshop on Gerontechnology*, pages 3–14. Springer, Cham, 2022.

[55] SPILAB. APIGenerator for End Devices (APIGEND). https://bitbucket.org/spilab/openapigenerator. Last accessed September 2022.

[56] OpenAPI Initiative. OpenAPI Generator. `https://github.com/OpenAPITools/openapi-generator`. Last accessed September 2022.

[57] Google. Firebase Cloud Messaging. https://firebase.google.com/docs/cloud-messaging. Last accessed 2022/10/21.

[58] MQTT. http://mqtt.org/. Last accessed 2022/11/02.

[59] Mariano Scazzariello, Lorenzo Ariemma, and Tommaso Caiazzi. Kathará: A lightweight network emulation system. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–2. IEEE, 2020.

[60] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *IEEE Software*, 33(3):94–100, 2016.

[61] Mayank Gokarna and Raju Singh. Devops: A historical review and future works. In *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 366–371, 2021.

[62] Michael P Papazoglou and Dimitrios Georgakopoulos. Service-oriented computing. *Communications of the ACM*, 46(10):25–28, 2003.

[63] Orestis Akrivopoulos, Ioannis Chatzigiannakis, Christos Tselios, and Athanasios Antoniou. On the deployment of healthcare applications over fog computing infrastructure. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 288–293. IEEE, 2017.

[64] Muhammad Alam, Joao Rufino, Joaquim Ferreira, Syed Hassan Ahmed, Nadir Shah, and Yuanfang Chen. Orchestration of microservices for iot using docker and edge computing. *IEEE Communications Magazine*, 56(9):118–123, 2018.

[65] Marino Linaje, Javier Berrocal, and Alfonso Galan-Benitez. Mist and edge storage: Fair storage distribution in sensor networks. *IEEE Access*, 7:123860–123876, 2019.

[66] ISA Group. OAS Generator. `https://github.com/isa-group/oas-generator`. Accessed October 21, 2022.

[67] Twilio. Guardrail. `https://github.com/twilio/guardrail`. Last accessed September 2022.

[68] OpenAPI Initiative. Swagger Codegen. `https://swagger.io/tools/swagger-codegen/`. Last accessed September 2022.

[69] Asunción Esteve. The business of personal data: Google, facebook, and privacy issues in the eu and the usa. *International Data Privacy Law*, 7(1):36–47, 2017.

[70] Max Van Kleek, Daniel Alexander Smith, Nigel Shadbolt, et al. A decentralized architecture for consolidating personal information ecosystems: The webbox. 2012.

[71] Andrei Vlad Sambra, Essam Mansour, Sandro Hawke, Maged Zereba, Nicola Greco, Abdurrahman Ghanem, Dmitri Zagidulin, Ashraf Aboulnaga, and Tim Berners-Lee. Solid: a platform for decentralized social applications based on linked data. *MIT CSAIL & Qatar Computing Research Institute, Tech. Rep.*, 2016.

[72] H. J. Kim, D. H. Lee, J. M. Lee, K. H. Lee, W. Lyu, and S. G. Choi. The qoe evaluation method through the qos-qoe correlation model. In *2008 Fourth International Conference on Networked Computing and Advanced Information Management*, volume 2, pages 719–725, 2008.

[73] Saurabh Shukla, Mohd Fadzil Hassan, Low Tan Jung, and Azlan Awang. Architecture for latency reduction in healthcare internet-of-things using reinforcement learning and fuzzy based fog computing. In *International Conference of Reliable Information and Communication Technology*, pages 372–383. Springer, 2018.

[74] Shangguang Wang, Yali Zhao, Lin Huang, Jinliang Xu, and Ching-Hsien Hsu. Qos prediction for service recommendations in mobile edge computing. *Journal of Parallel and Distributed Computing*, 127:134–144, 2019.

[75] Katrien De Moor, Istvan Ketyko, Wout Joseph, Tom Deryckere, Lieven De Marez, Luc Martens, and Gino Verleye. Proposed framework for evaluating quality of experience in a mobile, testbed-oriented living lab setting. *Mobile Networks and applications*, 15(3):378–391, 2010.

[76] Amazon Web Service. AWS Device Farm. `https://aws.amazon.com/device-farm/`. (Accessed on 28/10/2022).

[77] Microsoft Azure. App Center Test. `https://docs.microsoft.com/en-us/appcenter/test-cloud/`. (Accessed on 28/10/2022).

[78] HashiCorp. Terraform. `https://www.terraform.io/`. (Accessed on 22/10/2022).

[79] Budtmo. Docker Android. `https://github.com/budtmo/docker-android`. (Accessed on 28/10/2022).

[80] Pablo Fernandez. API Pecker. `https://www.npmjs.com/package/apipecker`. (Accessed on 28/10/2022).

[81] Github. Github Actions. `https://github.com/features/actions`. (Accessed on 27/11/2022).

[82] Jaime Galán-Jiménez, Javier Berrocal, Jose Garcia-Alonso, and Manuel Jesús Azabal. A novel routing scheme for creating opportunistic context-virtual networks in iot scenarios. *Sensors*, 19(8):1875, 2019.

[83] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.

[84] Pei Zhang, Christopher M. Sadler, Stephen A. Lyon, and Margaret Martonosi. Hardware design experiences in zebranet. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 227–238, New York, NY, USA, 2004. ACM.

[85] Amin Vahdat and David Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Duke University Durham, 2000.

[86] John Burgess, Brian Gallagher, David D. Jensen, and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–11, 2006.

[87] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, July 2003.

[88] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, WDTN '05, pages 252–259, New York, NY, USA, 2005. ACM.

[89] Albert Opher, Alex Chou, Andrew Onda, and Krishna Sounderrajan. The rise of the data economy: driving value through internet of things data monetization. *IBM Corporation: Somers, NY, USA*, 2016.

[90] Fatos Xhafa, Burak Kilic, and Paul Krause. Evaluation of iot stream processing at edge computing layer for semantic data enrichment. *Future generation computer systems*, 105:730–736, 2020.

[91] Gartner. Data quality fundamentals for data and analytics technical professionals. . Last accessed September 2022.

[92] Maryam Karimi Makiabadi. Edge computing, fog and mist architecture analysis, application, and challenges. 2021.

[93] Jason Paulos. *Investigating decentralized management of health and fitness data*. PhD thesis, Massachusetts Institute of Technology, 2020.

[94] Tao Zhu, Sahraoui Dhelim, Zhihao Zhou, Shunkun Yang, and Huansheng Ning. An architecture for aggregating information from distributed data nodes for industrial internet of things. In *Cyber-Enabled Intelligence*, pages 17–35. Taylor & Francis, 2019.

[95] Javier Berrocal, Jose Garcia-Alonso, Pablo Fernandez, Alejandro Perez-Vereda, Juan Hernandez, Carlos Canal, Juan Manuel Murillo, and Antonio Ruiz-Cortes. Early evaluation of mobile applications' resource consumption and operating costs. *IEEE Access*, 8:146648–146665, 2020.

[96] Björn Richerzhagen, Dominik Stingl, Julius Ruckert, Ralf Steinmetz, et al. Simonstrator: Simulation and prototyping platform for distributed mobile applications. In *The 8th EAI International Conference on Simulation Tools and Techniques (ACM SIMUTOOLS 2015)*, pages 99–108, 2015.

[97] Michiel Provoost and Danny Weyns. Dingnet: A self-adaptive internet-of-things exemplar. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 195–201. IEEE, 2019.

[98] Ilja Behnke, Lauritz Thamsen, and Odej Kao. Héctor: A framework for testing iot applications across heterogeneous edge and cloud testbeds. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, pages 15–20, 2019.

[99] Perforce Software. Perfecto. `https://www.perfecto.io/`. (Accessed on 28/10/2022).

[100] Kun Cao, Guo Xu, Junlong Zhou, Tongquan Wei, Mingsong Chen, and Shiyan Hu. Qos-adaptive approximate real-time computation for mobility-aware iot lifetime optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(10):1799–1810, 2018.

[101] Amin Vahdat, David Becker, et al. Epidemic routing for partially connected ad hoc networks, 2000.

[102] Pei Zhang, Christopher M Sadler, Stephen A Lyon, and Margaret Martonosi. Hardware design experiences in zebranet. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 227–238, 2004.

[103] Lavinia Amorosi, Luca Chiaraviglio, and Jaime Galan-Jimenez. Optimal energy management of uav-based cellular networks powered by solar panels and batteries: Formulation and solutions. *IEEE Access*, 7:53698–53717, 2019.

[104] Schahram Dustdar, Pablo Fernández, José María García, and Antonio Ruiz-Cortés. Elastic smart contracts in blockchains. *IEEE/CAA Journal of Automatica Sinica*, 8(12):1901–1912, 2021.

[105] Thomas Pusztai, Andrea Morichetta, Víctor Casamayor Pujol, Schahram Dustdar, Stefan Nastic, Xiaoning Ding, Deepak Vij, and Ying Xiong. Slo script: A novel language for implementing complex cloud-native elasticity-driven slos. In *2021 IEEE International Conference on Web Services (ICWS)*, pages 21–31. IEEE, 2021.

[106] Varun Gupta and Sandun Perera. Managing surges in online demand using bandwidth throttling: An optimal strategy amid the covid-19 pandemic. *Transportation Research Part E: Logistics and Transportation Review*, 151:102339, 2021.

[107] Mahdi Abbasi, E Mohammadi-Pasand, and Mohammad Reza Khosravi. Intelligent workload allocation in iot–fog–cloud architecture towards mobile edge computing. *Computer Communications*, 169:71–80, 2021.

[108] Behnam Zakeri, Katsia Paulavets, Leonardo Barreto-Gomez, Luis Gomez Echeverri, Shonali Pachauri, Benigna Boza-Kiss, Caroline Zimm, Joeri Rogelj, Felix Creutzig, Diana Ürge-Vorsatz, et al. Pandemic, war, and global energy transitions. *Energies*, 15(17):6114, 2022.

[109] Saeed H Alsamhi, Ou Ma, Mohd Ansari, Qingliang Meng, et al. Greening internet of things for greener and smarter cities: a survey and future prospects. *Telecommunication Systems*, 72(4):609–632, 2019.