UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

GRADO EN INGENIERÍA INFORMÁTICA EN INGENIERÍA DEL SOFTWARE

Trabajo Fin de Grado

Integración de datos IoT en portales Opendata

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

GRADO EN INGENIERÍA INFORMÁTICA EN INGENIERÍA DEL SOFTWARE

Trabajo Fin de Grado

Integración de datos IoT en portales Opendata

Autor: Daniel García Pérez

Tutor: Adolfo Lozano Tello

ÍNDICE GENERAL DE CONTENIDOS

1.Introducción 11

1.1 Principios Open Data 11

1.2 Motivación 13

1.3 Alcance 14

1.4 Objetivos 14

2. Planificación Del Trabajo 16

3. Estado del arte 19

3.1. Características de la arquitectura 19

3.2. Comparación de características con respecto a Open Data Cáceres 20

3.2.1 Conclusiones Comparativas 21

4. Análisis y Diseño 23

4.1 Solución Propuesta 23

4.2 Requisitos la Aplicación de Geolocalización Móvil 25

4.2.1 Funcionales 26

4.2.2 No Funcionales 26

4.3 Casos de Uso la Aplicación de Geolocalización Móvil 27

4.3.1 Descripción de Actores 27

4.3.2 Diagrama y Descripción de Casos De Uso 27

4.4 Requisitos la Aplicación de Visualización de datos 29

4.4.1 Funcionales 30

4.4.2 No Funcionales 31

4.5 Casos de Uso la Aplicación de Visualización de datos 31

4.5.1 Descripción de Actores 31

4.5.2 Diagrama y Descripción de Casos De Uso 31

5. Implementación 38

5.1 Herramientas 38
5.1.1 Herramientas De Control De Versiones 38
5.1.2 Herramientas De Desarrollo 40
5.2 Componentes 44
5.2.1 Aplicación Android 44
5.2.2 Montaje y configuración de placa Arduino 46
5.2.3 Servidores Heroku 54
5.2.4 Aplicación Web para visualizar datos 63
5.2.5 Programa para convertir los datos a RDF e inclusión de datos en Virtuoso. 73
5.3 Problemas Encontrados 78
5.3.1 Aplicación Android 78
5.3.2 Montaje y configuración de placa Arduino 78
5.3.3 Servidores Heroku 80
5.3.4 Aplicación Web para visualizar datos 80
6. Manuales De Usuario 81
6.1 Aplicación Android Geolocalización 81
6.1.1 Requisitos 81
6.1.2 Guía De Funcionalidad 81
6.2 Aplicación Web De Visualización De Datos 84
6.2.1 Requisitos 84
6.2.2 Guía De Funcionalidad 84
7. Mejoras 91
8. Conclusiones 92
8.1 Objetivos Técnicos 92
8.2 Conclusiones Personales 92
9. Agradecimientos 94

10.Referencias Bibliográficas (Según norma ISO690) 95

ÍNDICE DE TABLAS

Tabla 1 Horas Esperadas Y Horas Reales 18
Tabla 2 Requisitos Funcionales App Android 26
Tabla 3 Requisitos no funcionales App Android 26
Tabla 4 Caso De Uso Visualizar Ubicación En Mapa 28
Tabla 5 Caso De Uso Envío De Datos A Servidor 29
Tabla 6 Requisitos Funcionales Aplicación Web 30
Tabla 7 Requisitos no funcionales Aplicación Web 31
Tabla 8 Caso De Uso Visualizar Datos Gráfica Área 32
Tabla 9 Caso De Uso Visualizar Datos Gráfica Línea 33
Tabla 10 Visualizar Datos Mapa Calor 34
Tabla 11 Visualizar Datos Mapa 34
Tabla 12 Consulta Datos Geolocalización Rango 35
Tabla 13 Consulta Datos Temperatura Mes 36
Tabla 14 Consulta Datos Temperatura Fecha Concreta 36

ÍNDICE DE FIGURAS

Ilustración 1 Logo Heroku 24
Ilustración 2 Actor App Web 27
Ilustración 3 Diagrama De Casos De Uso App Android 28
Ilustración 4 Diagrama De Casos De Uso Aplicación Web 32
Ilustración 5 Git 39
Ilustración 6 Heroku Cli 40
Ilustración 7 Logo Android Studio 40
Ilustración 8 Logo Arduino Ide 41
Ilustración 9 Logo Sublime Text 42
Ilustración 10 Logo Eclipse 43
Ilustración 11 Logo Java 44
Ilustración 12 Logo Spring 44
Ilustración 13 Clases App 45
Ilustración 14 Archivos XML App 45
Ilustración 15 Pantalla Bienvenida App 46
Ilustración 16 Google Map Ubicacion 46
Ilustración 17 Placa Arduino MKR 1000 47
Ilustración 18 Sensor Temperatura Arduino TMP 36 47
Ilustración 19 Placa Protoboard y Cables 48
Ilustración 20 Esquema Conexión Placa y Sensor 49
Ilustración 21 Esquema Visual Conexion Placa y Sensor 50
Ilustración 22 Gestor Tarjetas Arduino IDE 51
Ilustración 23 Gestor Librerías Arduino IDE 52
Ilustración 24 Código Arduino IDE 53
Ilustración 25 Formato Base JSON-Server Temperatura 55
Ilustración 26 Entrada Principal Json Server Temperatura 57
Ilustración 27 API Json Server Temperatura 58
Ilustración 28 Formato Base JSON-Server Localizaciones 59
Ilustración 29 Entrada Inicial Json Server Localizaciones 60
Ilustración 30 API Json Server Localizaciones 61
Ilustración 31 Árbol Ficheros Eclipse 62
Ilustración 32 Código Application.java 62

Ilustración 33 Clase FechaHora.java 63
Ilustración 34 Petición Fecha y Hora Heroku 63
Ilustración 35 Pantalla Principal Web 64
Ilustración 36 Código Gráficas HTML 64
Ilustración 37 Gráficas Temperatura 65
Ilustración 38 Código HTML Formulario Temperatura 66
Ilustración 39 Formularios HTML Temperatura 66
Ilustración 40 Codigo HTML mapas 67
Ilustración 41 Mapa HTML Localización 67
Ilustración 42 Código HTML Mapa Calor 68
Ilustración 43 Mapa Calor HTML 68
Ilustración 44 Formularios Mapas HTML 69
Ilustración 45 Formularios HTML Mapas 69
Ilustración 46 Código Gráficas Morris 70
Ilustración 47 Creación Mapa Normal 72
Ilustración 48 Creación Mapa Calor 72
Ilustración 49 Ontologías Utilizadas Temperatura 74
Ilustración 50 Archivo TTL 74
Ilustración 49 Ontologías Utilizadas Localizaciones 76
Ilustración 50 Archivo TTL Localizaciones 77
Ilustración 51 Actualizar Controlador Wifi 79
Ilustración 52 Permisos App 82
Ilustración 53 Instalación App 82
Ilustración 54 Pantalla Bienvenida App 83
Ilustración 55 Mapa Ubicación Usuario 83
Ilustración 56 Página Principal Web 84
Ilustración 57 Pantalla Gráficas 85
Ilustración 58 Filtro Fecha Exacta 86
Ilustración 59 Filtrado De Ejemplo 86
Ilustración 60 Filtrado Por Mes 87
Ilustración 61 Ejemplo Filtrado Mes 87
Ilustración 62 Pantalla Google Maps 88
Ilustración 63 Alejar/Acercar Zoom 88
Ilustración 64 Zoom Extremadura 89

Ilustración 65 Mapa Calor Google 89
Ilustración 66 Filtrado Por Rango 90
Ilustración 67 Ejemplo Filtrado Por Rango 90

RESUMEN

En la actualidad, el avance de la tecnología está propiciando que surjan nuevas formas de relacionar la sociedad y el entorno que le rodea, y puedan ser capaces de interactuar gracias a dichos avances. Gracias a estos avances, nacen movimientos que tienen como propósito obtener información del entorno utilizando dispositivos que surgen gracias a los avances de la tecnología y poner esa información a disposición de la sociedad de una manera libre y abierta. Este movimiento es el llamado movimiento Open Data, cuyo propósito es otorgar a la sociedad de datos referentes al entorno donde viven las personas.

El objetivo de este trabajo de fin de grado es diseñar una arquitectura que contribuya con la sociedad suministrando más datos para que pueda mejorar su día a día.

La arquitectura que propongo consta de cuatro tipos de componentes: Recogida de datos, Almacenamiento de datos, Visualización de datos e Incorporación de dichos datos a un portal Open Data.

La recogida de datos se lleva a cabo configurando dispositivos para recoger datos de Temperatura mediante una placa Arduino y datos de Geolocalización diseñando una aplicación Android.

Los datos recogidos serán almacenados en un servidor alojado en la plataforma Heroku.

Se ha diseñado una aplicación web para visualizar los datos recogidos.

El último componente de la arquitectura será el que tiene como objetivo convertir e incorporar los datos capturados a un formato reconocible para un portal Open Data.

1.Introducción

Con la explosión y auge de las nuevas tecnologías, la sociedad está experimentando cambios con respecto a la gestión de sus recursos. Estos cambios, tienen por objetivo mejorar la calidad de vida de la sociedad en cuanto al conocimiento de su entorno.

Gracias a estos cambios, surgen movimientos de mano de los ciudadanos, en los que ponen en disposición de la sociedad datos recogidos por los dispositivos que surgen gracias a estas nuevas tecnologías.

Un movimiento significativo, puede ser el movimiento Open Data, que tiene como objetivo poner a disposición de la sociedad datos que recogen unos dispositivos conectados con el entorno, para posteriormente hacer un tratamiento de estos datos y tener información de nuestro entorno para tener más información a la hora de tomar decisiones que lo mejoren, un pilar fundamental del movimiento es que son datos abiertos, a disposición de cualquiera que quiera consultarlos.

Este Trabajo de Fin de Grado, tiene como objetivo principal, recoger datos de una manera automatizada, procesarlos, exponerlos y posteriormente incorporar dichos datos a un fichero Open Data con el fin de compartir información y conocimientos con la sociedad.

1.1 Principios Open Data

El movimiento Open Data, con el paso del tiempo ha ido adquiriendo cada vez más fuerza. Cada vez son más los profesionales de las tecnologías de la información que tienen inquietudes y ayudan a la comunidad en este tema. Debido al auge del movimiento, un grupo de expertos decidió reunirse en 2007 para fijar unas normas comunes que sirvieran de consenso entre profesionales para hacer más sólidos los cimientos del movimiento. Los principios básicos son los siguientes:

- Completos

Este principio se encarga de que los datos sean públicos y que no estén sujetos a limitaciones de privacidad, seguridad o privilegios válidos. Además, pretende tener a la sociedad informada para que los dirigentes puedan tomar decisiones que mejoren nuestro entorno.

- Primarios

Según este principio, los datos que se recojan deben estar detallados y ser capturados de manera que los datos no hayan pasado ningún filtro o algún tipo de proceso. El principio se asegura de que se facilite información acerca de la forma en la que se han obtenido los datos.

- Oportunos

Este principio asegura que los datos estén actualizados en todo momento y los usuarios puedan disponer de ellos en cualquier momento.

- Accesibles

Los datos deben estar disponibles para la gama más amplia de usuarios con la más amplia gama de propósitos.

- Automatizados

Los datos se estructuran razonablemente para permitir el procesamiento automatizado. Para que los datos abiertos puedan ser utilizados, deben estar debidamente codificados.

- No Discriminatorios

Los datos deben estar disponibles para cualquier persona sin necesidad de registro.

- No Propietarios

Los datos disponibles están en un formato sobre el cual ninguna entidad tiene el control exclusivo.

- Libres De Licencia

Este principio se asegura de que los datos no están sujetos a ningún derecho de autor, patente, marca o regulación. Se permiten restricciones razonables de privacidad y seguridad.

1.1 Motivación

Las motivaciones que han promovido mi interés por desarrollar este trabajo de fin de grado han sido:

- Supone un reto para mí como futuro profesional de la informática, enfrentarme a un problema real y cotidiano como puede ser planificar, diseñar y desarrollar un proyecto real como este. Me considero un entusiasta de la tecnología y el hecho de enfrentarme a tecnologías como puede ser el uso y configuración de placas Arduino, temática desconocida por mí, supone un reto personal. Por otro lado, también me gusta el desarrollo web y otras temáticas como puede ser la gestión de aplicaciones en entornos Cloud.

- La realización de un proyecto a este nivel supone una gran experiencia en desarrollo web y diversos temas que tienen que ver con mi futura carrera profesional, de manera que este proyecto va a suponer que mi próxima incursión en el mercado laboral sea más fácil para mí.

- Con el desarrollo de este proyecto, la idea es seguir promoviendo el movimiento Open Data y fortalecer el ecosistema de datos actuales de Open Data Cáceres. Cada vez son más los desarrolladores que dedican su vida profesional al movimiento Open Data, puede ser un buen nicho de mercado.

- En el actual sistema social y político, cada vez son más las mentes que intentan diseñar ideas que puedan beneficiar a la sociedad, por lo tanto, gracias al movimiento Open Data, pueden diseñarse nuevas formas de beneficiar a la sociedad.

- La sociedad en general no tiene tantas herramientas para conocer su entorno, por lo tanto, desde este proyecto, quiero promover la idea de que la sociedad acabe teniendo más facilidades para conocer su entorno y pueda tomar

decisiones que mejoren su día a día. En este ámbito puede encuadrarse la parte de captura de temperatura que comentaré en el siguiente capítulo.

1.3 Alcance

En este apartado, voy a exponer hasta donde quiero llegar con este proyecto:

- Definir y desarrollar una arquitectura que permita la captura de datos de diferentes temáticas, como por ejemplo la captura de datos de temperatura mediante una placa Arduino y un sensor de medición de temperatura y, por otro lado, la captura de datos de geolocalización mediante una aplicación Android. Una vez estos datos sean capturados, serán remitidos a un servidor en la nube.
- Desarrollar una aplicación web que permita capturar los datos recogidos por los dispositivos mencionados anteriormente y sea capaz de mostrar dichos datos de diferentes formas.
- Desarrollar un sistema que permita que los datos capturados sean incorporados de manera automática al portal Open Data Cáceres.

1.4 Objetivos

El objetivo general de este proyecto es crear una arquitectura que permita la recogida, exposición y envío de datos a un portal Open Data haciendo uso de los conocimientos adquiridos durante mi vida universitaria. Por otro lado, otro objetivo es facilitar a los usuarios información para que puedan tomar decisiones.

En cuanto al desarrollo e investigación los objetivos son:

- Hacer un estudio de las distintas formas de capturar datos de temperatura.
  1. Analizar sus características.
  2. Identificar sus características.
  3. Elegir la mejor opción.
- Hacer un estudio de las distintas formas de capturar datos de Geolocalización en aplicaciones Android teniendo en cuenta las características del dispositivo móvil, este estudio se lleva a cabo documentándome en varias fuentes y decidiendo cual es la mejor opción para capturar esos datos.
- Documentarme acerca de las plataformas Cloud existentes para la recepción de datos de los dispositivos anteriormente mencionados.
- Elegir la mejor forma de presentación de los datos recogidos.
- Informarme acerca de los mecanismos disponibles para enviar la información al portal Open Data

2. Planificación Del Trabajo

La planificación para el correcto desarrollo del proyecto se ha dividido en distintas fases acorde a la temática a tratar en ese momento, todas estas fases estarán debidamente detalladas en el capítulo 3:

Fase 1: Documentación acerca de Open Data y sus distintas características, premisas y pilares fundamentales.

Fase 2: Documentación acerca de las arquitecturas actuales para el envío y gestión de datos captados por dispositivos a portales Open Data, es decir, ver las posibles alternativas para resolver la problemática del Trabajo De Fin De Grado y elegir una de ellas con la supervisión del tutor.

Fase 3: Estudio y comprensión de métodos para la configuración de un dispositivo físico para capturar datos de Temperatura. Me decanto por la opción de utilizar una placa Arduino MKR 1000 y un sensor de temperatura TMP 36.

Fase 4: Obtención y configuración de dispositivos físicos para la captura de datos de Temperatura.

Fase 5: Estudio y comprensión de métodos para el desarrollo de aplicación Android para la captura de datos de Geolocalización. La elección para esta fase es sencilla, aplicación Android que capture la posición actual del dispositivo cada 10 minutos.

Fase 6: Desarrollo de aplicación Android para la obtención de datos de Geolocalización.

Fase 7: Estudio y evaluación de distintas plataformas Cloud para almacenar los datos capturados por los dispositivos elegidos en la fase anterior. Para esta fase me decanto por desplegar servidores json en Heroku.

Fase 8: Implementación de los servidores en Cloud que recibirán los datos capturados.

Fase 9: Estudio de alternativas referentes a la exhibición de los datos capturados. Me decanto por realizar una aplicación con HTML y JavaScript.

Fase 10: Desarrollo de aplicación web para la exhibición de dichos datos.

Fase 11: Estudio y reuniones acerca de las diferentes alternativas para incorporar los datos capturados al portal Open Data Cáceres.

A continuación, se detalla una tabla explicativa con el número de horas esperadas y horas reales para la realización de este proyecto.

Tabla 1 Horas Esperadas Y Horas Reales

Fase	***Horas Esperadas***	***Horas Reales***
Fase 1	10	10
Fase 2	20	20
Fase 3	5	4
Fase 4	55	65
Fase 5	10	8
Fase 6	75	100
Fase 7	10	3
Fase 8	5	5
Fase 9	10	10
Fase 10	75	90
Fase 11	5	5
Totales	**280**	**320**

3. Estado del arte

En este capítulo se procede a detallar la información de cada una de las fases detalladas en el capítulo anterior, de manera que se explicarán la arquitectura que he desarrollado para este proyecto, así como sus componentes y la relación que existe entre ellos.

3.1. Características de la arquitectura

Las características que ha de tener la arquitectura que se diseñe para la captura y gestión de datos será la siguiente:

- Los datos recogidos tienen que ser referentes a
    - Temperatura: Mediante a una placa Arduino o similar.
    - Geolocalización: Mediante una aplicación en un dispositivo Android.
- Los datos se encontrarán un servidor en la nube que permita su visualización en algún tipo de formato JSON, CSV, XML o similar.
- Visualizar los datos obtenidos de forma gráfica y amigable para el usuario.
- Estos datos serán almacenados en un portal Open Data para su compartición con la comunidad.

Al final de la realización del proyecto, tiene que haber una serie de componentes que estarán relacionados en torno a estas características.

La arquitectura resultante tendrá el siguiente aspecto:

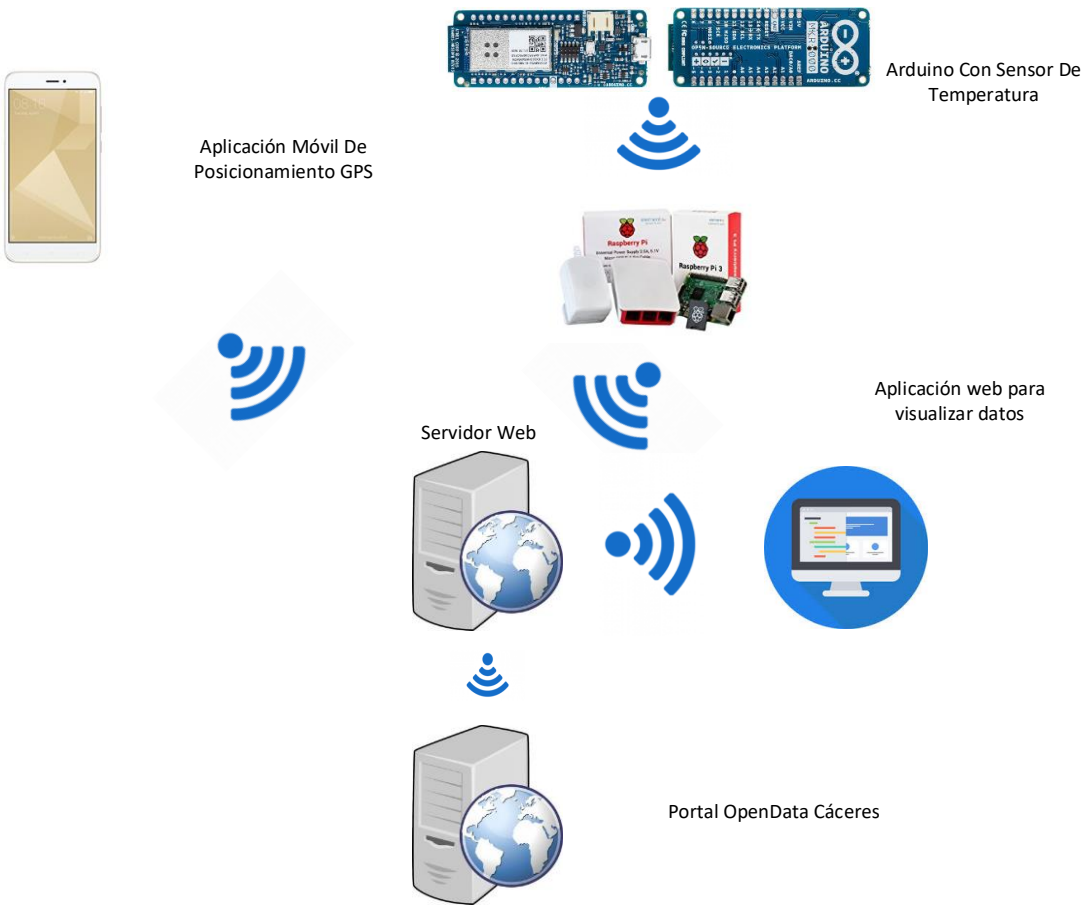

Ilustración 1 Esquema Arquitectura

Los elementos detallados y su funcionamiento se detallarán en el siguiente capítulo, en el apartado *"Solución propuesta"*.

3.2. Comparación de características con respecto a Open Data Cáceres

El objetivo de este apartado es comparar las características que he propuesto con las características de un sistema parecido y vigente como es Open Data Cáceres.

En este caso, se procede a establecer una comparativa entre la arquitectura que propongo y la arquitectura de Open Data Cáceres en cuanto a la recogida de información de sus autobuses.

Después de mantener una reunión con miembros del equipo de Open Data Cáceres puedo que extraer que el mecanismo que siguen en Open Data Cáceres la gestión de los datos referentes a las líneas y autobuses de Cáceres es el siguiente:

Los datos referentes a la localización de los autobuses están a su disposición mediante una URL proporcionada por la empresa que gestiona los autobuses.

Estos datos están en formato WSDL y son consultados cada 10 segundos mediante un programa escrito en lenguaje Python que genera un fichero de salida en formato RDF.

Este programa se está ejecutando durante todo el día menos en las horas nocturnas, ya que no hay servicio de autobuses en dicho horario.

Por otro lado, genera archivos que serán necesarios para componer la Especificación general de feeds de transporte público (GTFS). Esta especificación, es necesaria para poder ofrecer la información de tráfico a Google Maps, de manera que colaboran con la comunidad.

Para finalizar, sube los datos en formato RDF a la plataforma Virtuoso.

3.2.1 Conclusiones Comparativas

Tabla Comparativa:

Aspecto / *Arquitectura*	*Recogida*	*Alojamiento Externo*	*Formato De Datos*	*Aplicaciones Visualización*	*Formato RDF*	*Subida a Virtuoso*
Alumno Daniel García	✓	✓	JSON	✓	✓	✓
Open Data Cáceres / Vectalia		✓	WSDL	✓	✓	✓

Las conclusiones después de extraer todos los datos de ambas arquitecturas son:

Ambas recogen datos y los almacenan en un servidor disponible a través de una URL, salvo con la diferencia en la forma en la que se guardan los datos, ya que en la arquitectura que se define para solucionar este problema, los datos se almacenan en formato JSON y en la arquitectura que funciona en Open Data Cáceres guarda sus datos en formato WSDL.

Ambas tienen un software que muestra los datos recogidos al público en general. En el caso de Open Data Cáceres está el ejemplo de la aplicación Android de gestión de autobuses alojada en Play Store en el siguiente enlace : https://play.google.com/store/apps/details?id=com.cuatroochenta.subuscaceres&hl=es

Ambas generan ficheros en formato RDF utilizando programas escritos en Python.

Ambas suben los datos a la plataforma Virtuoso.

4. Análisis y Diseño

En este capítulo, procedo a explicar la solución propuesta detalladamente en cuanto a elementos y su comportamiento. Además, como es necesaria una aplicación web explicaré detalles que tienen que ver con el proceso de desarrollo software estudiado en las asignaturas de ingeniería de software e ingeniería de requisitos.

4.1 Solución Propuesta

La solución propuesta para la recogida de datos referentes a Temperatura y Geolocalización y posterior incorporación al fichero Open Data de Cáceres es la siguiente:

Se recogerán datos de dos fuentes:

Arduino con conexión wifi que envíe datos de Temperatura a un servidor Cloud.

Concretamente, se hará uso de una placa Arduino MKR 1000 con un sensor wifi incorporado y el sensor de temperatura TMP 36. Se procederá a realizar un programa que capture los datos y los envíe cada 5 minutos a un servidor desplegado en la plataforma Cloud Heroku, plataforma que será detallada más adelante. Junto con la Temperatura, se enviará una marca de tiempo, para su posterior gestión. Para obtener la marca de tiempo, se hace uso de un servidor de Fecha y Hora desplegado en Heroku mediante una aplicación Spring.

Aplicación Android para la recogida de datos de Geolocalización que envíe dichos datos a un Servidor Cloud.

En cuanto a la recogida de datos por parte de un dispositivo móvil, se diseñará una aplicación Android que vaya recogiendo datos de Geolocalización y los envíe cada 10 minutos a un servidor desplegado en la plataforma Heroku.

Una vez recogidos los datos, serán transmitidos a un servidor en Heroku, por lo tanto, procedo a explicar qué es Heroku:

¿Qué Es Heroku?

Ilustración 2 Logo Heroku

Heroku es una plataforma como servicio de computación en la Nube que soporta distintos lenguajes de programación.

Heroku es propiedad de Salesforce.com. Heroku, es una de las primeras plataformas de computación en la nube, que fue desarrollada desde junio de 2007, con el objetivo de soportar solamente el lenguaje de programación Ruby, pero posteriormente se ha extendido el soporte a Java, Node.js, Scala, Clojure y Python y (no documentado) PHP. La base del sistema operativo es Debian o, en la nueva plataforma, el sistema basado en Debian Ubuntu.

La idea propuesta con esta plataforma, es que los dispositivos anteriormente descritos envíen los datos que vayan capturando mediante una petición POST a un servidor alojado en la plataforma para su posterior uso.

El servidor alojado en dicha plataforma es un servidor llamado JSON-Server, cuya función es proporcionar una API Rest y un lugar donde albergar nuestros datos.

Se desplegarán 2 JSON-Server referentes a:

- Temperatura: Almacenará la temperatura junto con una marca de tiempo.
- Geolocalización: Almacenará las coordenadas y una marca de tiempo.

En Heroku, por otro lado, se desplegará una aplicación con Spring que mediante una petición GET devolverá la fecha y hora actuales, resolviendo la problemática que tiene la placa Arduino de que no almacena por sí misma la fecha actual.

Pero, por otro lado, en cuanto a la plataforma Heroku, tiene algún inconveniente a la hora de hacer uso de ella para usuarios que la usan de forma gratuita, y es que, si no se hace uso de una aplicación desplegada en Heroku durante 30 min, la plataforma reinicia la aplicación borrando los datos que se encuentren en ella.

Una vez se encuentren los datos en la plataforma, estarán a disposición de una Aplicación Web en HTML y JavaScript que mostrará dichos datos en la siguiente forma:

Temperaturas: se hará uso de la librería para crear gráficas Morris.js y se crearan gráficas de línea y de área para mostrar los datos. Además de poder filtrar las temperaturas existentes por fechas concretas o rango de meses.

Geolocalizaciones: se hará uso de la API de Google Maps para mostrar un mapa normal con las localizaciones que existan y de un mapa de calor con las zonas donde se encuentran datos. Dichos mapas, permitirán al usuario filtrar por un rango de fechas y ver qué dispositivos estuvieron conectados y su lugar en un rango de fechas concreto.

Los requisitos, actores, casos de uso y sus descripciones se detallarán en los siguientes apartados.

El último elemento de la arquitectura será un programa en Python que se encargará de hacer una petición GET a los servidores de Temperatura y Geolocalización en Heroku y partir de ellos, creará un fichero en formato RDF (extensión .ttl) que después se incorporará al fichero Open Data mediante Virtuoso.

Virtuoso, es un servidor de datos que aglutina muchas funcionalidades diferentes. Es un servidor web, una base de datos, una tripleStore, un gestor de puntos Sparql. Se usa para almacenar las tripletas en grafos y como punto Sparql para poder realizar consultas sobre las tripletas

A continuación, detallaré los requisitos de las 2 aplicaciones con las que el usuario podrá interaccionar con los datos.

4.2 Requisitos la Aplicación de Geolocalización Móvil

El objetivo de esta aplicación conlleva que el dispositivo mande datos cada 10 minutos a un servidor simplemente con el hecho de que el usuario la inicie, además, es necesario que el usuario sea capaz de ver su ubicación en un mapa. Para ello, se dota a la aplicación de una serie de requisitos según se especifica en cualquier proceso de desarrollo software. Para representar los requisitos, dividiremos los requisitos en dos tipos: Aquellos que tienen que ver con la funcionalidad de la aplicación, los llamados requisitos funcionales y, por otro lado, aquellos que tienen que ver con las características del sistema, los requisitos no funcionales.

4.2.1 Funcionales

A continuación, se expone una tabla con los requisitos funcionales:

Tabla 2 Requisitos Funcionales App Android

IDENTIFICADOR	DESCRIPCIÓN
RF1	El usuario tiene que ser capaz de ver su ubicación en un mapa.
RF2	Capaz de mandar su ubicación a un servidor externo cada 10 minutos

4.2.2 No Funcionales

A continuación, se expone una tabla con los requisitos no funcionales:

Tabla 3 Requisitos no funcionales App Android

IDENTIFICADOR	DESCRIPCIÓN
RNF1	La aplicación tiene que tener una interfaz sencilla
RNF2	Tiene que tener conexión a internet

4.3 Casos de Uso la Aplicación de Geolocalización Móvil

4.3.1 Descripción de Actores

Los actores especifican un rol jugado por un usuario o cualquier otro sistema que interactúa con la aplicación. Los actores pueden representar roles jugados por usuarios humanos, hardware externo, u otros sujetos.

Por lo tanto, para esta aplicación, vamos a tener como actor:

- **Usuario:** Su función será interactuar con la aplicación Android enviando peticiones y recibiendo datos en forma de mapas.

Ilustración 3 Actor App Web

4.3.2 Diagrama y Descripción de Casos De Uso

En este apartado, voy a exponer el diagrama de casos de uso junto la descripción de dichos casos en formato estandarizado, para la realización del diagrama se ha creado un proyecto en la herramienta Modelio, dicho proyecto se adjuntará a los recursos del proyecto:

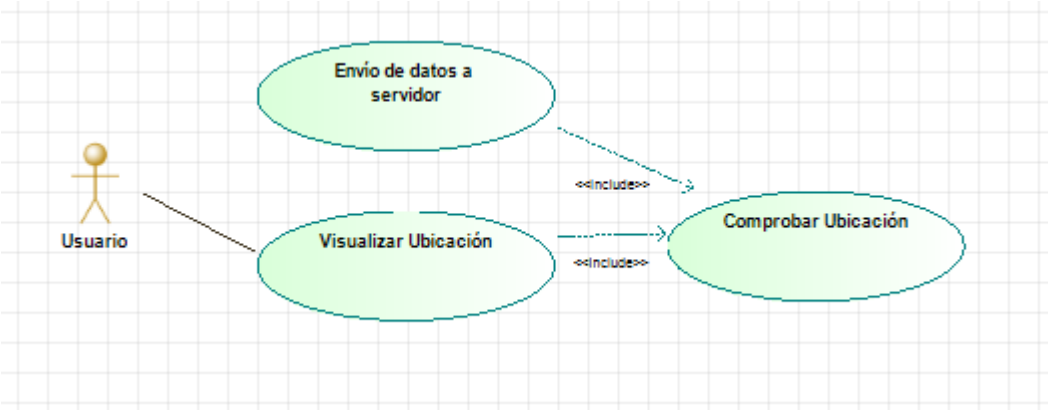

Ilustración 4 Diagrama De Casos De Uso App Android

A continuación, se expone la descripción de los casos de uso referentes a la aplicación Android:

Tabla 4 Caso De Uso Visualizar Ubicación En Mapa

CU1- RF1	**Visualizar Ubicación**	
Descripción	Cuando el usuario presione el botón "Ver mi ubicación" el sistema tiene que mostrarle al usuario un mapa con su ubicación	
Precondición	Aplicación encendida y Google Maps actualizado	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario enciende la aplicación
	2	El usuario hace click en el botón "Ver mi Ubicación"
Postcondición	El usuario podrá ver su ubicación en un mapa de Google Maps	
Excepciones	**Paso**	**Acción**

Tabla 5 Caso De Uso Envío De Datos A Servidor

CU2- RF2	**Envío de datos a servidor**	
Descripción	Cuando el usuario encienda la aplicación, se iniciará un servicio que enviará Geolocalización del usuario a un servidor en Heroku cada 10 minutos.	
Precondición	Aplicación encendida, tener la conexión a internet y el GPS del dispositivo activo	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario enciende la aplicación
Postcondición	El dispositivo enviará los datos del usuario a un servidor externo de forma totalmente transparente para el usuario.	
Excepciones	**Paso**	**Acción**

4.4 Requisitos la Aplicación de Visualización de datos

El propósito de esta aplicación es proporcionar al usuario un sitio web donde pueda consultar los datos capturados ya sea en forma de gráfica o mapa, según el tipo de datos. Para ello, es necesario dotar a la aplicación de una serie de requisitos según se especifica en cualquier proceso de desarrollo software, para representar los requisitos, dividiremos los requisitos en dos tipos: *funcionales* y *no funcionales*.

4.4.1 Funcionales

A continuación, expongo la tabla con los requisitos funcionales de la aplicación para visualizar los datos vía web.

Tabla 6 Requisitos Funcionales Aplicación Web

IDENTIFICADOR	DESCRIPCIÓN
RF1	El usuario puede visualizar todos los datos de temperatura mediante una gráfica de área
RF2	El usuario puede visualizar todos los datos de temperatura mediante una gráfica de línea
RF3	El usuario tiene la posibilidad de consultar en un mapa de calor las ubicaciones de todos los dispositivos que han mandado datos en algún momento al servidor.
RF4	El usuario tiene la posibilidad de consultar en un mapa las ubicaciones de todos los dispositivos que han mandado datos en algún momento al servidor.
RF5	El usuario puede filtrar por un rango de fecha y hora concreto los dispositivos conectados
RF6	El usuario puede filtrar por un mes concreto los datos de temperatura
RF7	El usuario puede elegir un momento concreto del tiempo y ver los datos de temperatura asociados a ese día y rango horario

4.4.2 No Funcionales

A continuación, expongo una tabla con los requisitos no funcionales:

Tabla 7 Requisitos no funcionales Aplicación Web

IDENTIFICADOR	DESCRIPCIÓN
RNF1	La aplicación tiene que tener una interfaz sencilla
RNF2	Alojada en algún servidor externo para la consulta de datos
RNF3	Capaz de obtener datos de una fuente externa

4.5 Casos de Uso la Aplicación de Visualización de datos

4.5.1 Descripción de Actores

El actor será aquel que interactúe con la aplicación, por lo tanto, para esta aplicación, se define al siguiente actor

- **Usuario:** Su función será interactuar con la aplicación web enviando peticiones y recibiendo datos en forma de gráficas o mapas de distinto tipo.

Ilustración 3 Actor App Web

4.5.2 Diagrama y Descripción de Casos De Uso

En este apartado, voy a exponer el diagrama de casos de uso junto la descripción de dichos casos:

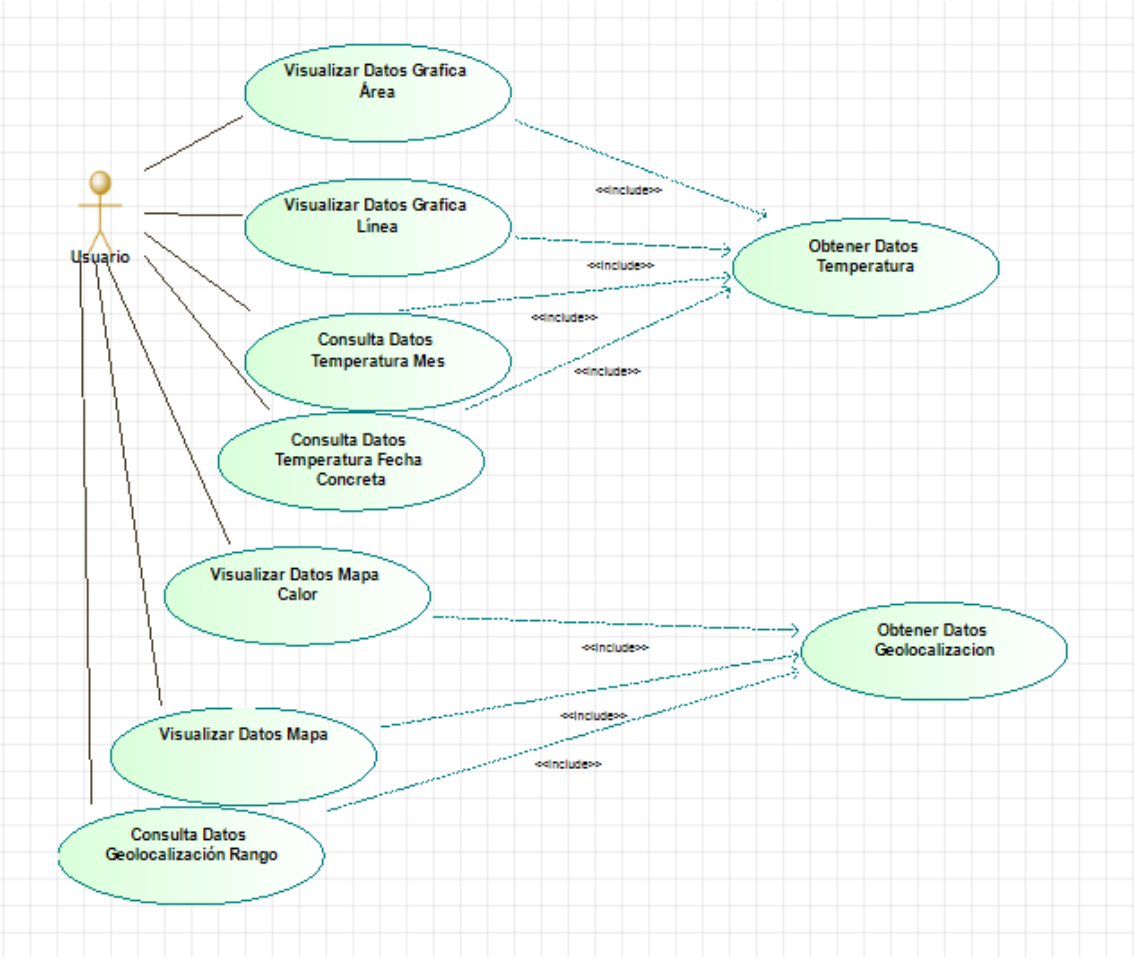

Ilustración 5 Diagrama De Casos De Uso Aplicación Web

Tabla 8 Caso De Uso Visualizar Datos Gráfica Área

CU1- RF1	**Visualizar Datos Grafica Área**	
Descripción	Cuando el usuario acceda a la aplicación, tiene que tener la posibilidad de ver una gráfica de área con todas las temperaturas existentes	
Precondición	Conexión a internet para acceder al sitio web	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario accede a la web
	2	El usuario hace click en el botón “Graficas Temperatura”

Postcondición	La aplicación mostrará al usuario una gráfica de área con las temperaturas disponibles.	
Excepciones	**Paso**	**Acción**

Tabla 9 Caso De Uso Visualizar Datos Gráfica Línea

CU2- RF2	**Visualizar Datos Grafica Línea**	
Descripción	Cuando el usuario acceda a la aplicación, tiene que tener la posibilidad de ver una gráfica de línea con todas las temperaturas existentes	
Precondición	Conexión a internet para acceder al sitio web	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario accede a la web
	2	El usuario hace click en el botón “Graficas Temperatura”
Postcondición	La aplicación mostrará al usuario una gráfica de línea con las temperaturas disponibles.	
Excepciones	**Paso**	**Acción**

Tabla 10 Visualizar Datos Mapa Calor

CU3- RF3	**Visualizar Datos Mapa Calor**	
Descripción	Cuando el usuario acceda a la aplicación, tiene que tener la posibilidad de ver un mapa de calor con todas las Geolocalizaciones existentes.	
Precondición	Conexión a internet para acceder al sitio web	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario accede a la web
	2	El usuario hace click en el botón “Mapa Calor”
Postcondición	La aplicación mostrará al usuario un mapa de calor con todas las Geolocalizaciones	
Excepciones	**Paso**	**Acción**

Tabla 11 Visualizar Datos Mapa

CU4- RF4	**Visualizar Datos Mapa**	
Descripción	Cuando el usuario acceda a la aplicación, tiene que tener la posibilidad de ver un mapa con todas las Geolocalizaciones existentes.	
Precondición	Conexión a internet para acceder al sitio web	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario accede a la web
	2	El usuario hace click en el botón “Mapa”

Postcondición	La aplicación mostrará al usuario un mapa con todas las Geolocalizaciones	
Excepciones	**Paso**	**Acción**

Tabla 12 Consulta Datos Geolocalización Rango

CU5- RF5	**Consulta Datos Geolocalización Rango**	
Descripción	Cuando el usuario acceda a la aplicación, tiene que tener la posibilidad de ver un mapa normal y un mapa de calor con todas las Geolocalizaciones dentro de un rango.	
Precondición	Conexión a internet para acceder al sitio web	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario accede a la web
	2	El usuario hace click en el botón “Mapa”
	3	El usuario elige una fecha mínima en un calendario
	4	El usuario elige una hora mínima en el calendario
	5	El usuario elige una fecha máxima en un calendario
	6	El usuario elige una hora máxima en un calendario
	7	Click al botón de “búsqueda por fecha”
Postcondición	La aplicación mostrará al usuario un mapa normal y un mapa de calor con todas las Geolocalizaciones dentro de ese rango	
Excepciones	**Paso**	**Acción**
	1	Si el usuario no hace click en el botón, la web cargará por defecto todos los datos que existan, sin haber aplicado ningún filtro

Tabla 13 Consulta Datos Temperatura Mes

CU6- RF6	**Consulta Datos Temperatura Mes**	
Descripción	Cuando el usuario acceda a la aplicación, tiene que tener la posibilidad de ver los 2 tipos de gráfica descritas con las temperaturas en un mes elegido	
Precondición	Conexión a internet para acceder al sitio web	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario accede a la web
	2	El usuario hace click en el botón “Graficas”
	3	El usuario elige mes
	4	El usuario introduce un rango horario
	5	Click al botón de “búsqueda por mes”
Postcondición	La aplicación mostrará al usuario gráficas de área y línea con los datos de un mes concreto	
Excepciones	**Paso**	**Acción**
	1	Si el usuario no hace click en el botón, la web cargará por defecto todos los datos que existan, sin haber aplicado ningún filtro

Tabla 14 Consulta Datos Temperatura Fecha Concreta

CU7- RF7	**Consulta Datos Temperatura Fecha Concreta**
Descripción	Cuando el usuario acceda a la aplicación, tiene que tener la posibilidad de ver los 2 tipos de gráfica descritas con las temperaturas en una fecha y rango horario elegidos
Precondición	Conexión a internet para acceder al sitio web

Secuencia Normal	**Paso**	**Acción**
	1	El usuario accede a la web
	2	El usuario hace click en el botón “Graficas”
	3	El usuario elige una fecha
	4	El usuario elige un rango horario
	5	Click al botón de “búsqueda por fecha”
Postcondición	La aplicación mostrará al usuario gráficas de área y línea con los datos de un mes concreto	
Excepciones	**Paso**	**Acción**
	1	Si el usuario no hace click en el botón, la web cargará por defecto todos los datos que existan, sin haber aplicado ningún filtro

5. Implementación

En este capítulo, se procede a tratar, exhibir y comentar los diferentes aspectos acerca de la implementación de los distintos componentes de la arquitectura, así como también se van a comentar los problemas que me han ido surgiendo y su posible solución durante esta etapa del proyecto.

Por otro lado, también procederé a detallar las herramientas que han sido utilizadas para el desarrollo de los distintos componentes que he diseñado para poder atacar el problema descrito en capítulos anteriores. Probablemente, éste sea uno de los capítulos más importantes de la documentación, ya que suponen el corazón del proyecto, sin la implementación de los componentes necesarios, no podría haberse hecho realidad este proyecto.

5.1 Herramientas

En este apartado, se procede a detallar las herramientas que se han utilizado para poder llevar a cabo el desarrollo de los componentes de la arquitectura, dichas herramientas y el control y conocimiento sobre ellas, son de suma importancia a la hora de realizar no solo este proyecto, si no para desarrollar cualquier proyecto en la vida profesional. El uso de las más actuales herramientas es sumamente importante para el desarrollo profesional, ya que, si no se eligen las herramientas más actuales, se pueden caer en problemas de compatibilidad en cuestiones técnicas, por lo tanto, uno de los trabajos más importantes que he tenido que desarrollar, ha sido el de elegir la herramienta concreta y actual para cada uno de los apartados.

5.1.1 Herramientas De Control De Versiones

Si nos ceñimos a la definición de control de versiones: gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración

del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

Por lo tanto, es necesario hacer uso de dichas herramientas, para este proyecto, he utilizado algunas herramientas para el control de versiones, *Git* y *Heroku Cli*, las cuales, procedo a detallar y exponer.

5.1.1.1 Git

Ilustración 6 Git

Git es un software de control de versiones, diseñado para mejorar la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de un ordenador y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

5.1.1.2 Heroku Cli

Ilustración 7 Heroku Cli

Heroku Cli es un software de control de versiones que utiliza comandos GIT, su propósito es ayudar al desarrollador con su labor de despliegue y gestión de aplicaciones en la plataforma Heroku. Dicho software me ha sido de gran importancia a la hora de gestionar mis aplicaciones y servidores desplegados en la plataforma Heroku.

5.1.2 Herramientas De Desarrollo

En este epígrafe, se detallarán las herramientas que he utilizado para el desarrollo de los componentes de la arquitectura.

5.1.2.1 Android Studio

Ilustración 8 Logo Android Studio

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android.

Está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de Android.

El uso que he hecho de este software ha sido principalmente para el desarrollo de la aplicación de Geolocalización en Android. Además, dicho software ha sido el utilizado en asignaturas cursadas durante el grado.

5.1.2.2 Arduino IDE

Ilustración 9 Logo Arduino Ide

Arduino IDE es el software de código abierto que se usa para programar las placas Arduino, es un programa ligero y que permite una amplia gama de funcionalidades. Por otro lado, gracias a la plataforma que existe a nivel mundial de Arduino, su uso es muy sencillo y ayuda y reduce la carga de trabajo de comprensión para los desarrolladores.

Con respecto a este software, ha sido utilizado para diseñar el programa que se cargará en la placa Arduino para la recogida de Temperaturas. En siguientes apartados explicaré el proceso de montaje y configuración de dicha placa.

5.1.2.3 Sublime Text

Ilustración 10 Logo Sublime Text

Sublime text es un editor de texto y editor de código fuente está escrito en C++ y Python.

Se puede descargar y evaluar de forma gratuita. Sin embargo, no es software libre o de código abierto3 y se debe obtener una licencia para su uso continuado, aunque la versión de evaluación es plenamente funcional y no tiene fecha de caducidad.

Con respecto a la utilidad que ha tenido Sublime Text en mi proyecto, ha sido de gran ayuda para desarrollar la aplicación web con los lenguajes HTML y JavaScript para la visualización de datos y para crear los Script en Python para transformar los datos en formato JSON en archivos en formato RDF (.ttl).

Lenguajes Utilizados.

- **HTML**: sigla en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web.
- **JavaScript**: es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,3 basado en prototipos, imperativo, débilmente tipado y dinámico.
- **Python**: lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Librerías Utilizadas

- **Morris**: Librería JavaScript que permite la creación de todo tipo de gráficas, para el desarrollo de este proyecto, he elegido las gráficas de área y línea.
- **JQuery**: Biblioteca multiplataforma de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.
- **Ajax**: acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.
- **Google Maps**: Librería de Google para gestionar mapas.

5.1.2.4 Eclipse

Ilustración 11 Logo Eclipse

Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrolladores.

He utilizado Eclipse para desarrollar un servidor web que mediante una petición devuelva la fecha y hora actual. Dicho servidor, se ha implementado en lenguaje Java y Spring

Java:

Ilustración 12 Logo Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.

Spring:

Ilustración 13 Logo Spring

Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java

5.2 Componentes

En este nuevo apartado, procedo a explicar la implementación de cada uno de los componentes desarrollado paso a paso, con capturas explicativas incluidas de manera que se puedan entender cada una de las decisiones que he ido tomando en cada uno de los pasos que he ido dando hacia la resolución del proyecto.

5.2.1 Aplicación Android

La aplicación Android, está desarrollada con Android Studio, el árbol de ficheros que la compone y de la cual explicaré la función de cada clase es el siguiente:

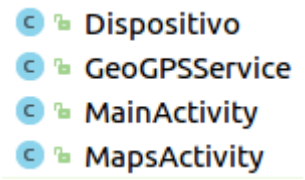

Ilustración 14 Clases App

- **Dispositivo:** Se encarga de crear el modelo de datos que construirá el objeto, que estará compuesto, por un identificador como el IMEI del dispositivo ya que es un valor único, la latitud, longitud, altura y la fecha actual.
- **GeoGPSService:** Se encarga de iniciar un servicio que va a estar mandando cada 10 minutos la Geolocalización del dipositivo al servidor en Heroku, lo hará mediante un objeto JSON.
- **MainActivity:** Es la encargada de arrancar la aplicación y de lanzar el servicio GeoGPSService, además de lanzar el mapa cuando el usuario haga click en el botón "Ver Mi Ubicación".
- **MapsActivity:** Esta clase se encarga de todo lo referente a la visualización del mapa con la ubicación del usuario.

Por otro lado, existen también archivos XML que son los que almacenan la interfaz de la aplicación, a continuación, el árbol y la organización de los mismos.

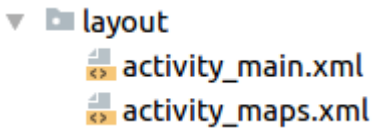

Ilustración 15 Archivos XML App

- **Activity_main.xml** es la encargada de organizar los elementos gráficos que verá el usuario en su pantalla, el aspecto de la pantalla es:

Ilustración 16 Pantalla Bienvenida App

- **Activity_maps.xml** se encarga de mostrar al usuario el mapa donde verá su ubicación, el aspecto del mapa es el siguiente:

Ilustración 17 Google Map Ubicacion

5.2.2 Montaje y configuración de placa Arduino

Con respecto al montaje y configuración de la placa Arduino, es necesario poseer una serie de hardware aparte del software necesario para otorgarle funcionalidad a la placa.

Con respecto al hardware serán necesarios:

1 Placa Arduino MKR 1000.
2 Sensor de Temperatura para Arduino TMP 36.
3 Placa Protoboard para conexión de el sensor de temperatura.
4 Cables que permitan conectar el sensor con la placa.
5 Cable de alimentación para la placa Arduino.

Con respecto al software, se hace uso de:

1 Arduino IDE.

Procedimiento:

Lo primero que necesitamos es tener operativa la placa Arduino MKR 1000, tal placa, puede observarse en la siguiente figura.

Ilustración 18 Placa Arduino MKR 1000

Por otro lado, es necesario poseer el sensor de temperatura TMP 36, en la siguiente figura se puede observar el sensor.

Ilustración 19 Sensor Temperatura Arduino TMP 36

Para poder conectar el sensor con la placa, es necesario el uso de una placa Protobard y cables de conexión como los que aparecen en la siguiente figura.

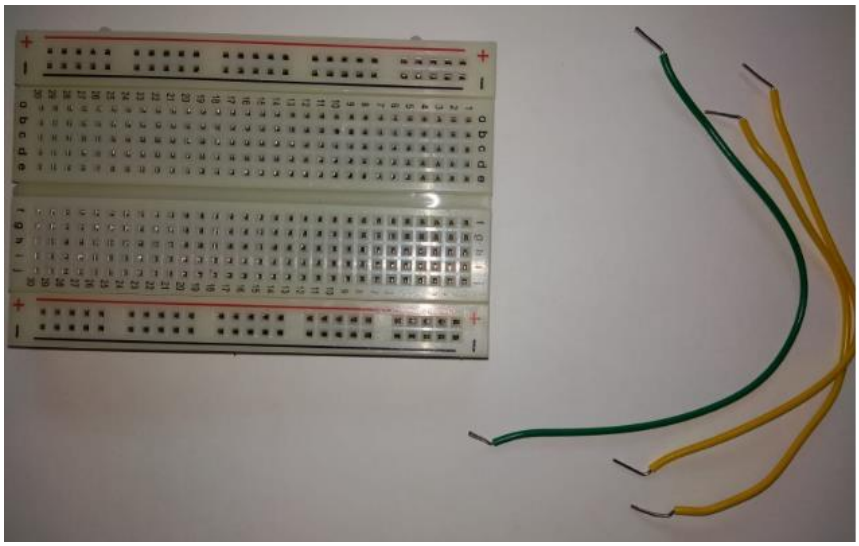

Ilustración 20 Placa Protoboard y Cables

El siguiente paso será el de conectar todos los dispositivos siguiendo el siguiente esquema de conexión:

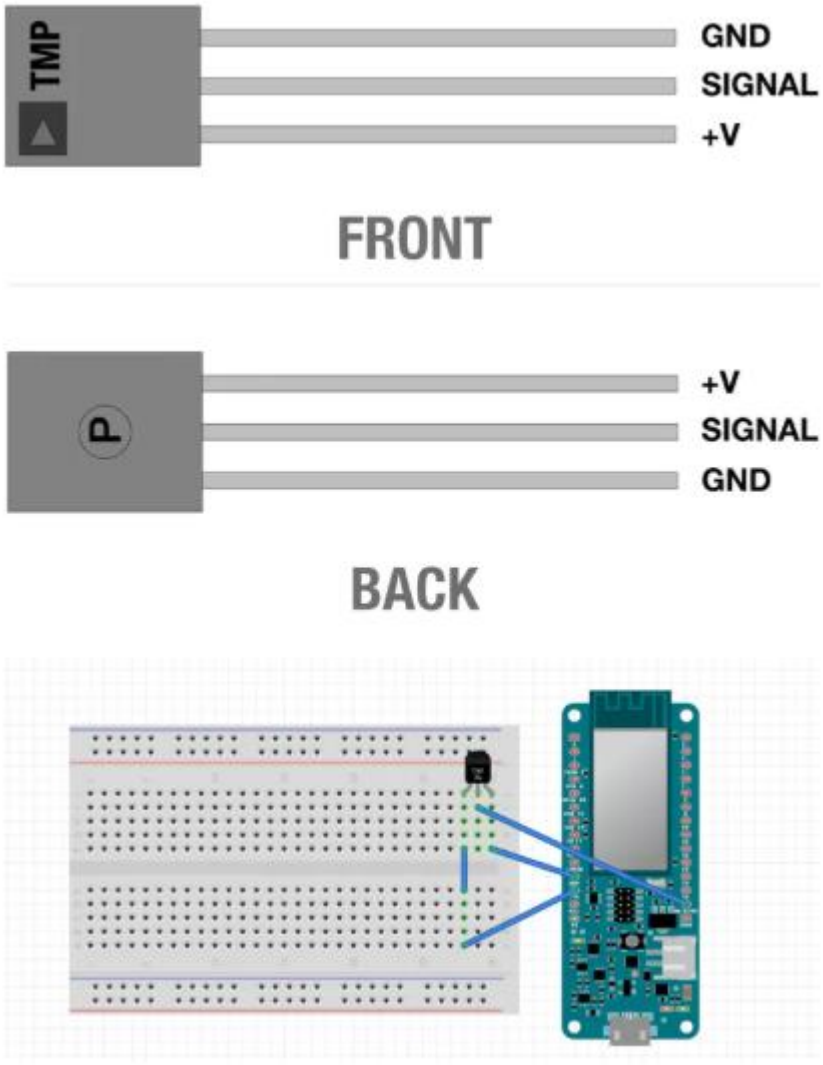

Ilustración 21 Esquema Conexión Placa y Sensor

TMP +v conecta con ***MKR1000 - VCC***

TMP GND conecta con ***MKR1000 - GND***

TMP SIG conecta con ***MKR1000 - A0***

El esquema llevado a la práctica quedaría conforme a la siguiente figura.

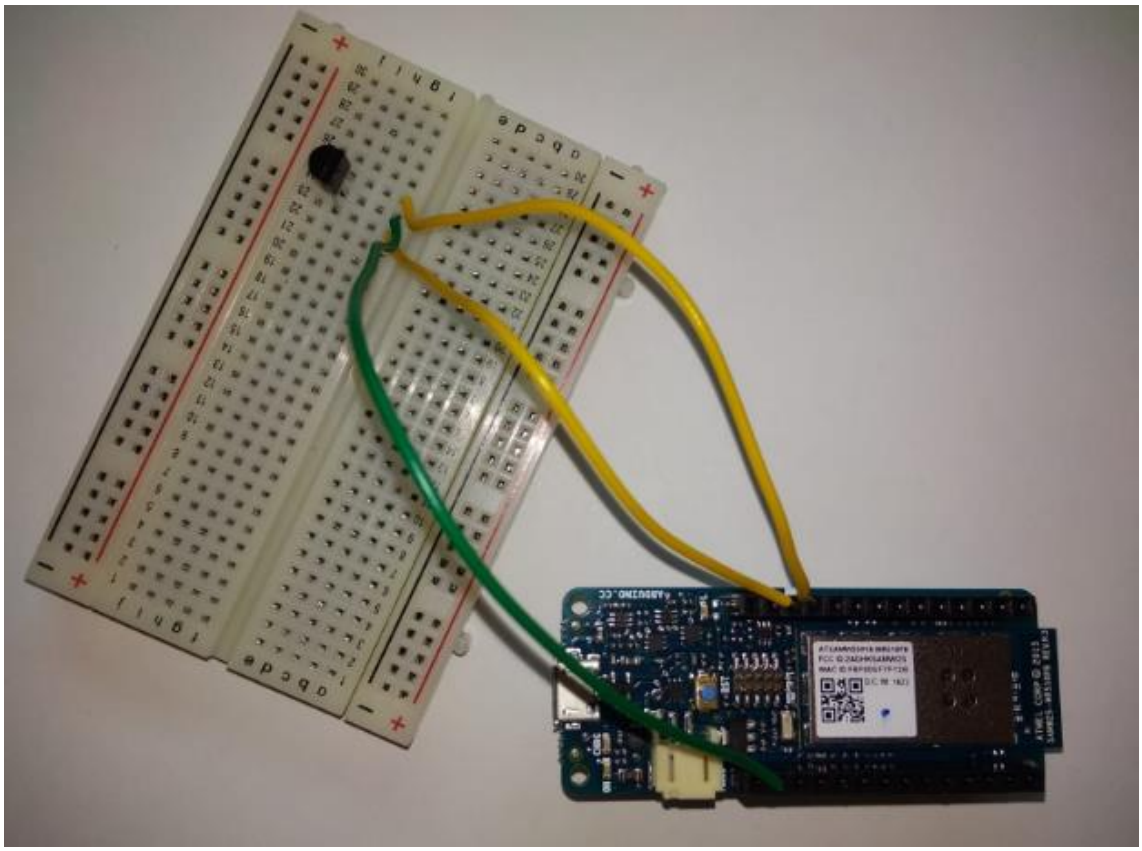

Ilustración 22 Esquema Visual Conexion Placa y Sensor

El último paso será conectar la placa al PC mediante un cable USB.

Configuración Software.

Para la configuración software será necesario tener instalado Arduino IDE.

Una vez instalado, será necesario instalar las librerías necesarias para hacer funcionar la placa MKR 1000.

Con la ventana de bienvenida de Arduino IDE, hacemos click en *Herramientas>Placa>Gestion De Tarjetas*, una vez dentro:

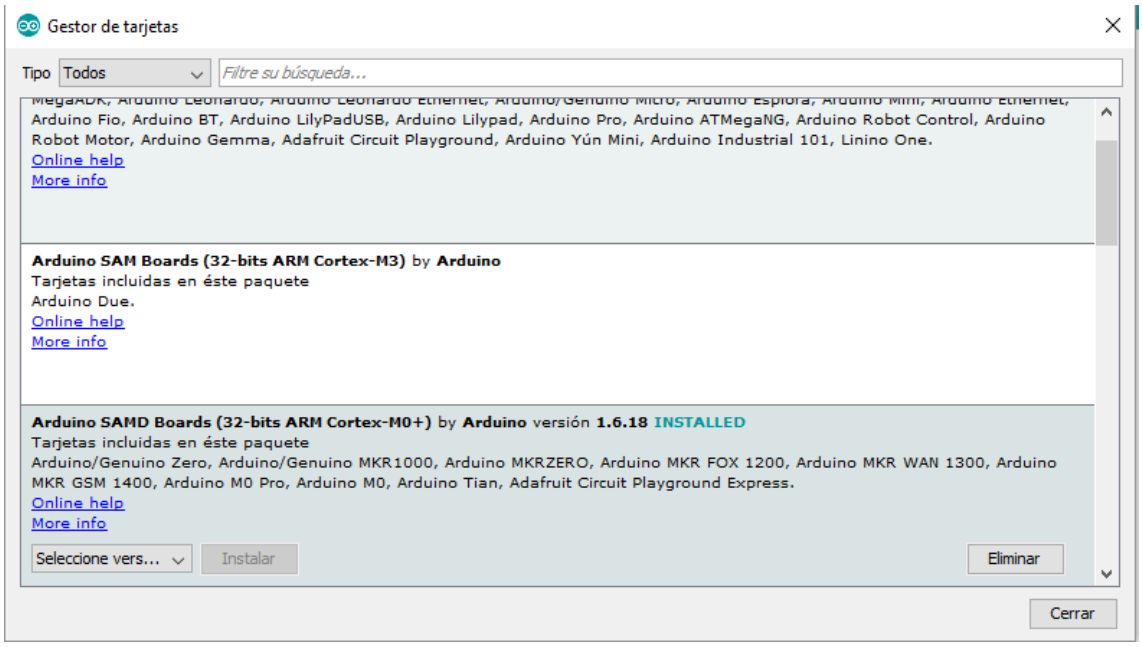

Ilustración 23 Gestor Tarjetas Arduino IDE

Buscamos la opción *Arduino SAMD Boards (32-bits ARM Cortex-M0+) by Arduino versión 1.6.18* y hacemos click en instalar.

Una vez instalado, volvemos a la pantalla de bienvenida del IDE y volvemos a pulsar sobre *Herramientas>Placa* y observando las opciones, vemos que ya nos aparece nuestra placa MKR 1000, la seleccionamos y ya tenemos instalada la placa, a partir de ahora nuestro IDE ya reconoce la placa.

El siguiente paso es instalar las librerías necesarias para que funcione el sensor Wifi integrado.

Para instalar la librería Wifi, hacemos click en *Programa>Incluir Librería>Gestionar Librerías*, y nos llevará a la siguiente ventana.

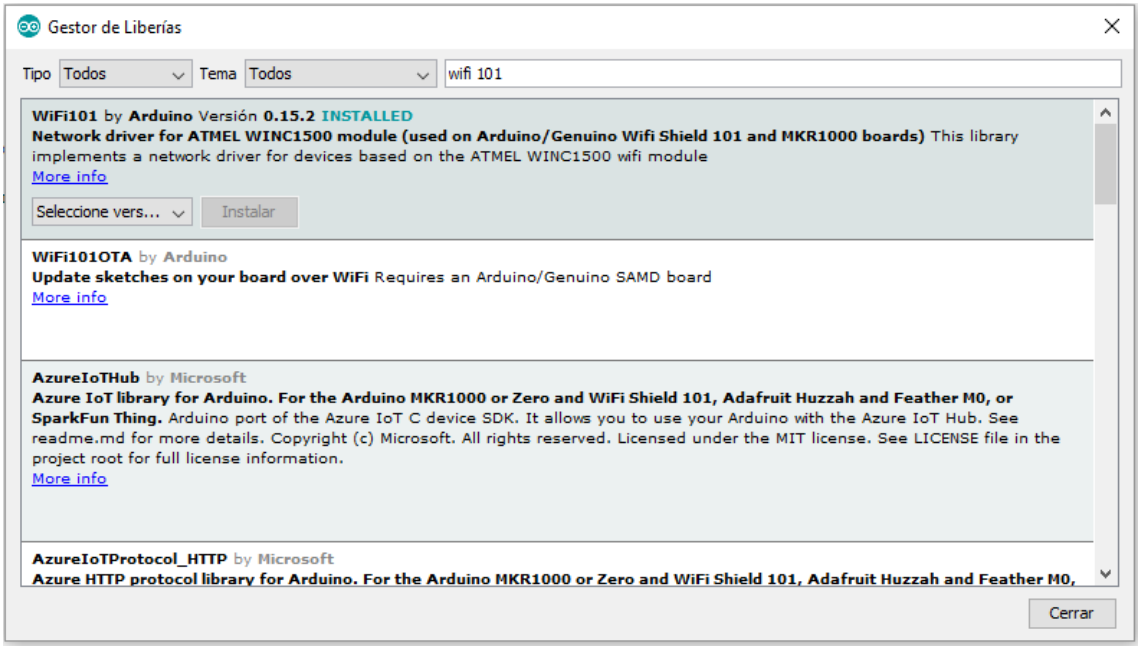

Ilustración 24 Gestor Librerías Arduino IDE

Buscamos Wifi 101 e instalamos.

Una vez instalada, podemos empezar a escribir el código que controle el funcionamiento de la placa.

Explicación del código que controla la placa.

El código de gestión se divide en 2 funciones *setup()* y *loop().*

Código Setup:

```
/*TFG Daniel Garcia Perez*/
#include <ArduinoHttpClient.h>
#include <WiFi101.h>
#include <ArduinoJson.h>


#include "arduino_secrets.h"
/////// Wifi Settings ///////
char ssid[] = "Livebox-5B06";
char pass[] = "G0D5n0zm";

const char serverAddress[] = "tmptfgdgarcia.herokuapp.com";  // server address
const char serverAddress2[] = "apptfgtempdaniel.herokuapp.com";  // server address

int port = 80;
int port2 = 80;
int sensorPin = 0;
float voltage=0;
int sensorValue = 0; // variable to store the value coming from the sensor
float temperatureC = 0.0;

WiFiClient wifi;
HttpClient client = HttpClient(wifi, serverAddress, port);
HttpClient client2 = HttpClient(wifi, serverAddress2, port2);
int status = WL_IDLE_STATUS;
int statusCode = 0;
String response;


void setup() {
  Serial.begin(9600);
  while(!Serial);
  while ( status != WL_CONNECTED) {
    Serial.print("Attempting to connect to Network named: ");
    Serial.println(ssid);                   // print the network name (SSID);

    // Connect to WPA/WPA2 network:
    status = WiFi.begin(ssid, pass);
  }

  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
}
```

Ilustración 25 Código Arduino IDE

El código referente al apartado *setup()* y declaración de variables globales, simplemente se encarga de establecer la conexión Wifi para poder conectar con el exterior a recoger o enviar datos según el caso. Además de establecer las conexiones necesarias para activar el sensor de temperatura.

Código Loop, al ser una función bastante extensa, procedo a explicarla de manera escrita:

Loop() se encarga de recoger la temperatura del sensor TMP 36 mediante la instrucción *float temperature = ((analogRead(A0) * (3300/1024)) -500).*

El siguiente paso, es hacer una petición GET al servidor de fecha y hora para que éste le otorgue la fecha y hora actual. Una vez tiene la hora, construye un objeto JSON con la temperatura y la fecha actual y mediante una instrucción POST envía el objeto JSON al servidor de temperaturas.

5.2.3 Servidores Heroku

El despliegue de servidores en la nube que permitan la recepción de datos y su posterior reenvío a cualquier dispositivo que realice una petición es algo fundamental, por lo que decido que necesito alguna solución que sea ligera y me permita esa comunicación. De manera que indagando, me decanto por implementar un sencillo JSON-Server, debido a que no necesito una infraestructura robusta para almacenar los datos como por ejemplo pueden ofrecerme los gestores de bases de datos mas conocidos como Oracle, MySql, Postgre… etc.

La elección de JSON-Server en lugar de una base de datos más conocida viene de la no necesidad de establecer relaciones entre las distintas entidades de datos, simplemente necesito un lugar donde almacenar las Temperaturas y por otro lado las Geolocalizaciones.

Además, voy a enviar los datos vía JSON, por lo que JSON-Server es capaz de gestionar datos en formato JSON. Sin más dilación, me decido a implementar cada uno de los servidores JSON.

Temperatura

El primer servidor que voy a implementar es un servidor que permita recibir la temperatura en el siguiente formato JSON.

1 Identificador: Es importante que todo dato que se envíe referente a la temperatura tenga un identificador para las posibles gestiones que tengan que hacerse con dicho dato.

2 Temperatura: Un campo importante, en él, recibiremos la temperatura que vaya capturando la placa Arduino.

3 Fecha: La fecha y hora en la que se ha capturado la temperatura, es indispensable que haya una marca de tiempo junto con la Temperatura.

A continuación, se procede a explicar cómo se ha desplegado dicho servidor en Heroku.

El primer paso es clonar de GitHub el servidor original para modificarlo ajustándolo a mis necesidades, por lo tanto, se abre una ventana de comandos y se ejecuta el comando:

```
git clone https://github.com/jesperorb/json-server-heroku.git
```

Una vez clonado, cambiamos de directorio y procedemos a cambiar el modelo de datos editando el archivo *db.json* y añadiendo el siguiente contenido:

```
1  {
2    "temperatura":[
3      {
4        "id" : 0,
5        "temperatura": 20,
6        "fecha": "5/24/18 17:40 PM"
7      },
8      {
9        "id" : 1,
10       "temperatura": 20,
11       "fecha": "5/24/18 17:45 PM"
12     },
13     {
14       "id" : 2,
15       "temperatura": 21,
16       "fecha": "5/24/18 17:50 PM"
17     }
18
19   ]
20 }
```

Ilustración 26 Formato Base JSON-Server Temperatura

Una vez modificado el archivo, es necesario actualizar los cambios ejecutamos las instrucciones

```
git add .
git commit -am “Mensaje De Commit”
heroku login
heroku create nombreProyecto
git push heroku master
heroku open
```

Las instrucciones anteriores se encargan de añadir al repositorio Git los archivos modificados con *“add .”,* de confirmar la actualización mediante *“commit”.*

Para poder desplegarlos en la plataforma heroku, es necesario ejecutar las instrucciones *“heroku login”* para iniciar sesión en Heroku, con *“create”* creamos un nuevo espacio para nuestra aplicación, mediante el comando *“push”* subimos el proyecto local en Git a Heroku y se despliega en la nube. Por último, mediante la instrucción *“open”* se abre la aplicación en un navegador comprobando que todo ha ido bien.

El resultado es el siguiente:

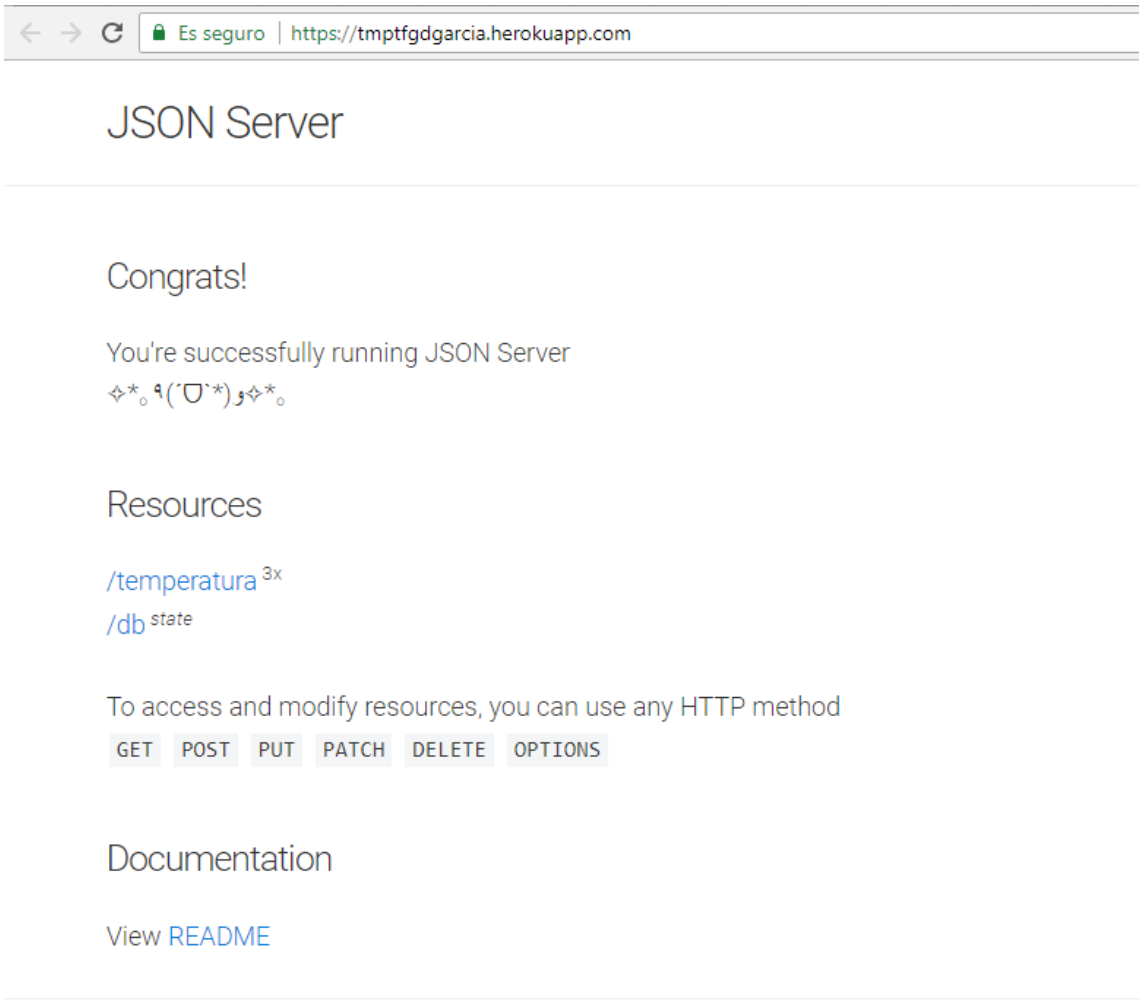

Ilustración 27 Entrada Principal Json Server Temperatura

Por último, para ver los datos que hay en dicho servidor de una forma pública y abierta, de manera que se cumpla uno de los principios fundamentales del movimiento Open Data, podemos visitar la dirección:

https://tmptfgdgarcia.herokuapp.com/temperatura

En dicha dirección, podremos ver los datos que almacena el servidor en el momento actual, dichos datos servirán para que la aplicación que expondré en siguientes apartados, pueda surtirse de datos y funcionar:

Es seguro | https://tmptfgdgarcia.herokuapp.com/temperatura


```
[
  {
    "id": 0,
    "temperatura": "20",
    "fecha": "5/24/18 17:40 PM"
  },
  {
    "id": 1,
    "temperatura": "20",
    "fecha": "5/24/18 17:45 PM"
  },
  {
    "id": 2,
    "temperatura": "21",
    "fecha": "5/24/18 17:50 PM"
  }
]
```


Ilustración 28 API Json Server Temperatura

Localización

Para implementar el servidor de Localizaciones, he seguido el mismo procedimiento que para implementar el servidor de Temperatura, con el único cambio del archivo *db.json*.

El procedimiento que he seguido ha sido:

El primer paso es clonar de GitHub el servidor original para modificarlo ajustándolo a mis necesidades, por lo tanto, se abre una ventana de comandos y se ejecuta el comando:

```
git clone https://github.com/jesperorb/json-server-heroku.git
```

Una vez clonado, cambiamos de directorio y procedemos a cambiar el modelo de datos editando el archivo *db.json* y añadiendo el siguiente contenido:

```
1  {
2    "localizaciones":[
3      {
4        "latitud": "38.9650771",
5        "altitud": "0.0",
6        "longitud": "-5.8575522",
7        "fecha": "Thu May 24 17:28:44 CEST 2018",
8        "imei": "865199027752595",
9        "id": 1
10     },
11     {
12       "latitud": "38.9650771",
13       "altitud": "0.0",
14       "longitud": "-5.8575522",
15       "fecha": "Thu May 24 17:28:44 CEST 2018",
16       "imei": "865199027752595",
17       "id": 2
18     },
```


Ilustración 29 Formato Base JSON-Server Localizaciones

Una vez modificado el archivo, es necesario actualizar los cambios ejecutamos las instrucciones

```
git add .
git commit -am “Mensaje De Commit”
heroku login
heroku create nombreProyecto
git push heroku master
heroku open
```

Las instrucciones anteriores se encargan de añadir al repositorio Git los archivos modificados con *“add .”*, de confirmar la actualización mediante *“commit”*.

Para poder desplegarlos en la plataforma heroku, es necesario ejecutar las instrucciones *“heroku login”* para iniciar sesión en Heroku, con *“create”* creamos un nuevo espacio para nuestra aplicación, mediante el comando *“push”* subimos el proyecto local en Git a Heroku y se despliega en la nube. Por último, mediante la instrucción *“open”* se abre la aplicación en un navegador comprobando que todo ha ido bien.

El resultado es el siguiente:

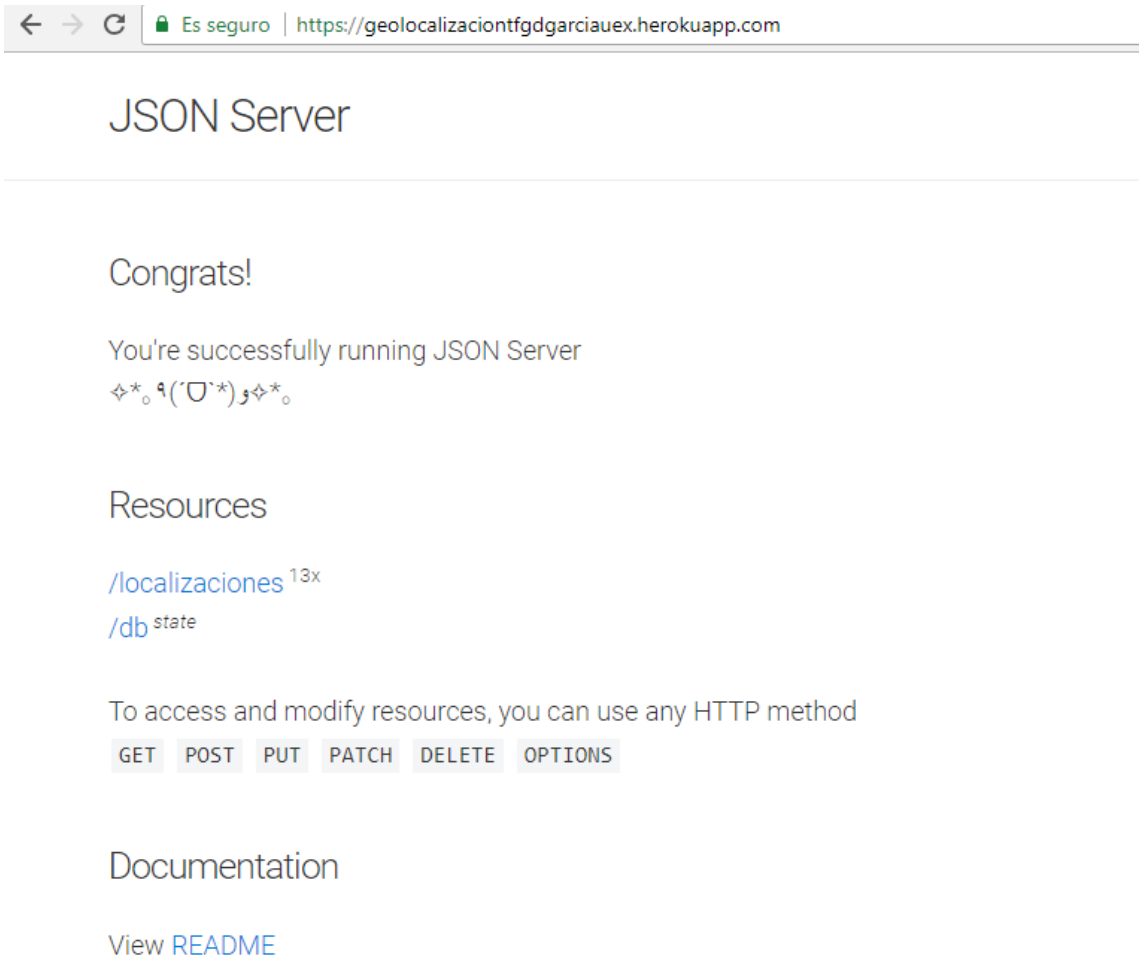

Ilustración 30 Entrada Inicial Json Server Localizaciones

Por último, para ver los datos que hay en dicho servidor de una forma pública y abierta, de manera que se cumpla uno de los principios fundamentales del movimiento Open Data, podemos visitar la dirección:

https://geolocalizaciontfgdgarciauex.herokuapp.com/localizaciones

En dicha dirección, podremos ver los datos que almacena el servidor en el momento actual, dichos datos servirán para que la aplicación que expondré en siguientes apartados, pueda surtirse de datos y funcionar:

Es seguro | https://geolocalizaciontfgdgarciauex.herokuapp.com/localizaciones


```
    "id": 8
  },
  {
    "latitud": "38.9651093",
    "altitud": "0.0",
    "longitud": "-5.8575344",
    "fecha": "Thu May 24 18:28:38 CEST 2018",
    "imei": "865199027752595",
    "id": 9
  },
  {
    "latitud": "38.9651046",
    "altitud": "0.0",
    "longitud": "-5.8575339",
    "fecha": "Thu May 24 18:29:12 CEST 2018",
    "imei": "865199027752595",
    "id": 10
  },
  {
    "longitud": "-6.3438783333333335",
    "latitud": "39.47900833333333",
    "imei": "868767027276703",
    "altitud": "85.5",
    "fecha": "Thu Jun 07 14:03:21 CEST 2018",
    "id": 11
  },
  {
    "longitud": "-6.3438783333333335",
    "latitud": "39.47900833333333",
    "imei": "868767027276703",
    "altitud": "85.5",
    "fecha": "Thu Jun 07 14:52:09 CEST 2018",
    "id": 12
  },
  {
    "longitud": "-6.3438783333333335",
    "latitud": "39.47900833333333",
    "imei": "868767027276703",
    "altitud": "85.5",
    "fecha": "Thu Jun 07 16:49:03 CEST 2018",
    "id": 13
  }
]
```


Ilustración 31 API Json Server Localizaciones

Fecha y Hora

Diseñar un servidor de Fecha y Hora no entraba dentro de mis planes iniciales, puesto que no lo consideraba necesario. Una vez que iba avanzando en el proyecto, llego al punto en el que haciendo uso de la placa Arduino, esta debía mandar los datos de temperatura junto con una marca de tiempo, aquí es donde surge la problemática de que las placas Arduino por sí solas, no manejan el tiempo actual, solamente cuentan el tiempo mientras están encendidas, es decir; cada vez que se apagan, su reloj interno se pone a 0. Debido a esta problemática decido implementar mediante Spring, en Eclipse,

un servicio que dada una petición devuelva la fecha y hora actual, de esta forma, la placa Arduino, ya tiene la herramienta que le permita incluir una marca de tiempo en el paquete JSON que le envíe al servidor de Temperatura.

El proyecto resultante en eclipse es bastante sencillo, basta con tener el framework Spring instalado y hacer las llamadas pertinentes en las clases.

El árbol de clases es

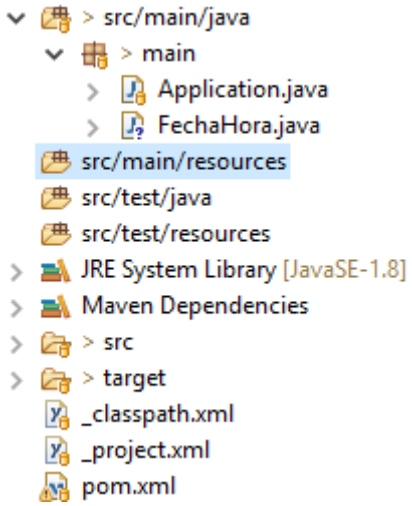

Ilustración 32 Árbol Ficheros Eclipse

En la clase Application.java simplemente se hace la llamada a SpringBootApplication para que pueda iniciarse Spring.

```
1  package main;
2
3  import org.springframework.boot.SpringApplication;
5
6  @SpringBootApplication
7  public class Application{
8      public static void main(String[] args){
9          SpringApplication.run(Application.class, args);
10     }
11 }
12
```

Ilustración 33 Código Application.java

Por su parte, la clase FechaHora.java contiene el método que responderá cada vez que se pregunte por la ruta "/fechas" que será la fecha actual.

```java
package main;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

import org.json.simple.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class FechaHora{

    @SuppressWarnings("unchecked")
    @RequestMapping(value = "/fechas", method = RequestMethod.GET, produces = {"application/json"})
    public @ResponseBody JSONObject getFecha(){
     JSONObject answer = new JSONObject();
     Date d=new Date();
     SimpleDateFormat dateFormatMadrid = new SimpleDateFormat();
     dateFormatMadrid.setTimeZone(TimeZone.getTimeZone("Europe/Madrid"));
     answer.put("fecha",dateFormatMadrid.format(d));
        return answer;
    }
```

Ilustración 34 Clase FechaHora.java

Dicha clase, se encarga de obtener la fecha actual e introducirla en un objeto JSON para que devolverlo cuando sea hecha la petición.

Una vez implementado, será desplegado en Heroku pudiendo consultarse en la siguiente dirección: https://apptfgtempdaniel.herokuapp.com/fechas

Y el resultado será el siguiente

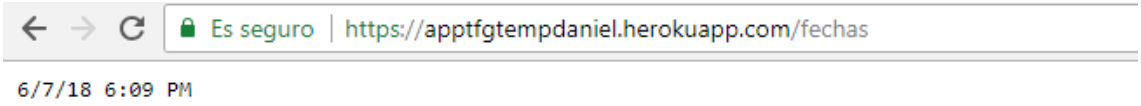

Ilustración 35 Petición Fecha y Hora Heroku

5.2.4 Aplicación Web para visualizar datos

La aplicación Web para visualizar datos se divide en 2 apartados bien diferenciados, un apartado donde implementa la interfaz gráfica de la página con HTML y la parte que gestiona el funcionamiento en JavaScript. Toda la aplicación web está implementada usando la herramienta de diseño Sublime Text

Comienzo explicando las partes fundamentales de la interfaz:

Consultado la web: https://webpageiottfg.herokuapp.com/

Obtendremos el siguiente resultado:

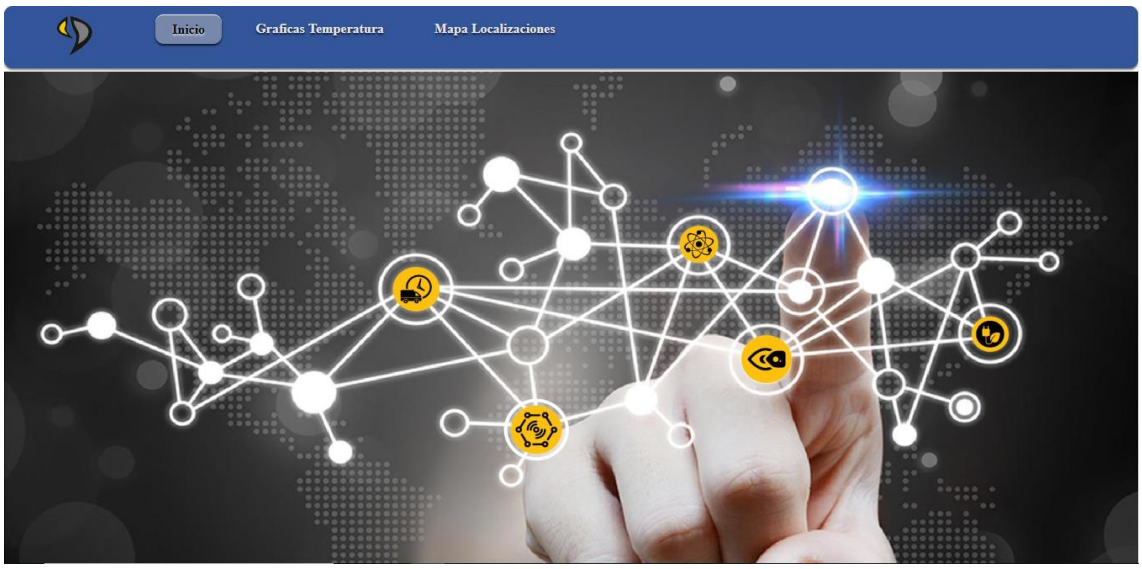

Ilustración 36 Pantalla Principal Web

Si observamos el menú superior, vemos que hay 2 apartados claramente diferenciados *Gráficas Temperatura* y *Mapa Localizaciones.*

Haciendo click en Gráficas Temperatura la página nos redirecciona a una página donde nos mostrará las gráficas correspondientes.

El código que concierne a la muestra de las gráficas es el siguiente:

```
<div class="row">
    <div class="col-md-6">
        <h2>Grafica de Linea</h2>
        <hr>
        <div id="myfirstchart"></div>
    </div>

    <div class="col-md-6">
        <h2>Grafica de Area</h2>
        <hr>
        <div id="mysecondchart"></div>
        </div>
    </div>
</div>
```

Ilustración 37 Código Gráficas HTML

En la ilustración superior vemos las etiquetas HTML que se refieren a la creación de capas para la muestra de gráficas, mediante los atributos id le damos

nombre a las capas para que luego el apartado JavaScript configure los datos que tiene que mostrar.

El resultado será el siguiente.

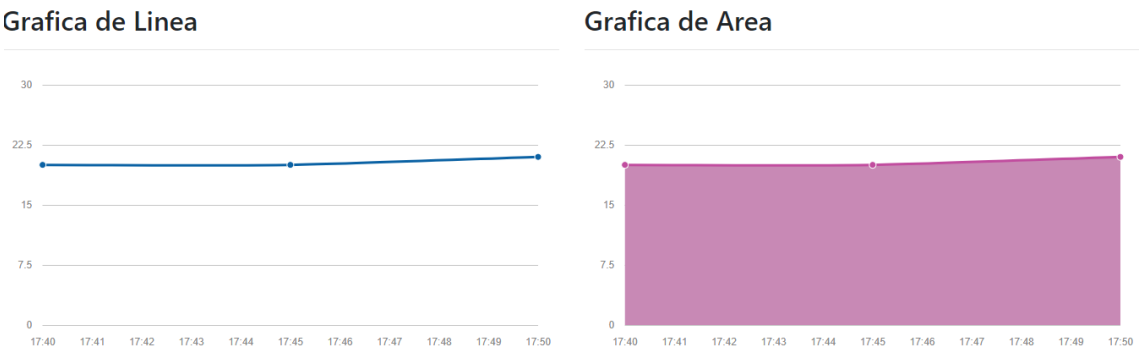

Ilustración 38 Gráficas Temperatura

Por otro lado, en la misma página, se encuentran los formularios que nos permitirán filtrar los datos que se van a mostrar en las gráficas. Su código es el siguiente:

```html
<div>
    <h2>Busqueda por fecha exacta</h2>
<form name="myForm" >
  <div>

    <input type="date" id="fechaC" name="fechaC" min="2018-04-01" max="2019-04-20" ><br/>
    Intruduce hora minima: <input type="time" id="rangoC" name="rangoC"><br/>
    Intruduce hora maxima: <input type="time" id="rangoM" name="rangoM">
    <span class="validity"></span>
  </div>
  <div>
    <button type="button"class="btn btn-secondary" id="boton">Busqueda por fecha</button>
  </div>
</form>
</div>
</td>
```

```html
<div>
    <h2>Busqueda por mes</h2>
<form name="myForm2">
  <div>

    <select id="month" name="month" class="select_month cart_select italic light" />
    <option selected hidden>Mes</option>
    <option value="01">Enero</option>
    <option value="02">Febrero</option>
    <option value="03">Marzo</option>
    <option value="04">Abril</option>
    <option value="05">Mayo</option>
    <option value="06">Junio</option>
    <option value="07">Julio</option>
    <option value="08">Agosto</option>
    <option value="09">Septiembre</option>
    <option value="10">Octubre</option>
    <option value="11">Noviembre</option>
    <option value="12">Diciembre</option>
    </select><br/>
    Intruduce hora minima: <input type="time" id="rangoC1" name="rangoC1"><br/>
    Intruduce hora maxima: <input type="time" id="rangoM1" name="rangoM1">
    <span class="validity"></span>
  </div>
  <div>

    <button type="button" class="btn btn-secondary" id="boton2">Busqueda por meses</button>
  </div>
</form>
```

Ilustración 39 Código HTML Formulario Temperatura

En la ilustración superior se define un formulario en HTML con listas desplegables, calendarios y horas para filtrar los datos.

El resultado en el navegador es el siguiente:

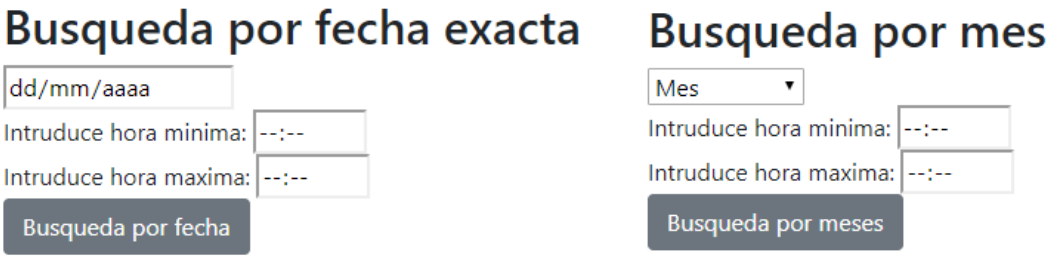

Ilustración 40 Formularios HTML Temperatura

A continuación, observamos el menú superior y pasamos a la explicación de la interfaz que tiene que ver con los mapas. Si pasamos el puntero por encima de Mapa Localizaciones podremos observar que se despliegan dos submenús *Mapa* y *Mapa Calor*.

Haciendo click en Mapa, obtendremos el mapa con las localizaciones que tenemos guardadas en el servidor. A continuación, muestro el código HTML que se encarga de mostrar los datos.

```
<h2>Mapa</h2>
<hr>
<div id="map"></div>
</div>
```

Ilustración 41 Codigo HTML mapas

Simplemente, se encarga de cargar una capa con la etiqueta DIV para que el archivo JavaScript se encargue de darle valores. El resultado será el siguiente:

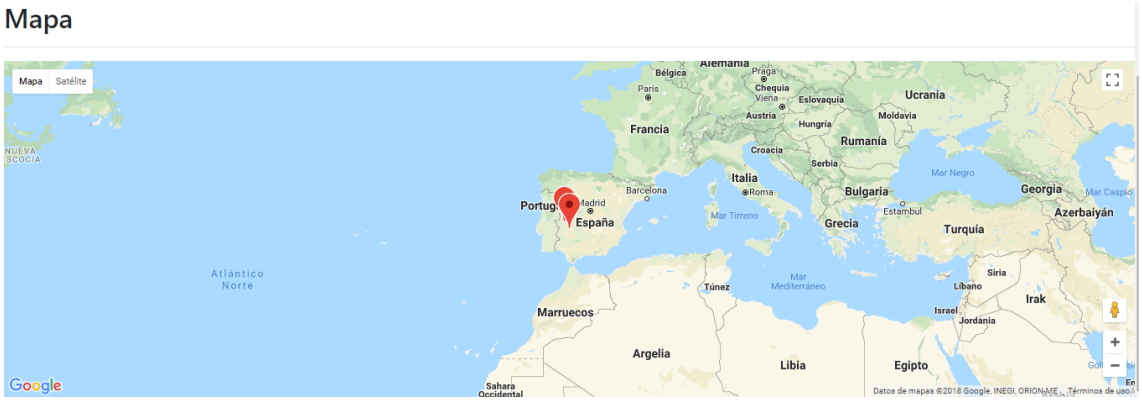

Ilustración 42 Mapa HTML Localización

Por otro lado, si hacemos click en el submenú *Mapa Calor*, obtendremos un mapa de calor con las localizaciones:

Su código:

```
<div class="col-md">
<h2>Mapa de Calor</h2>
<hr>
<div id="map"></div>
</div>
```

Ilustración 43 Código HTML Mapa Calor

Simplemente, se encarga de cargar una capa con la etiqueta DIV para que el archivo JavaScript se encargue de darle valores. El resultado será el siguiente:

Ilustración 44 Mapa Calor HTML

Y para finalizar con el apartado HTML, ambas páginas, la del mapa normal y el mapa de calor, tienen un formulario en el que da la posibilidad al usuario de elegir un rango de tiempo para mostrar los datos filtrados.

El código asociado es el siguiente:

```
<form name="myForm" >
<table>
    <tr>
        <td>
  <div>
    <p>Fecha minima
    <input type="date" id="fechaC" name="fechaC" min="2018-04-01" max="2019-04-20" ><br/>
    Intruduce hora minima: <input type="time" id="rangoC" name="rangoC"><br/>
    </p>
    <span class="validity"></span>
  </div>
    </td>
   <td>
   <div float="left">
    <p>Fecha maxima
    <input type="date" id="fechaCF" name="fechaCF" min="2018-04-01" max="2019-04-20" ><br/>
    Intruduce hora maxima: <input type="time" id="rangoC1" name="rangoC1"><br/>
    </p>
    <span class="validity"></span>
  </div>

  <div>

  </div>
  </td>
</tr>
<tr><td><button type="button"class="btn btn-secondary" id="boton1" name="boton1">Busqueda por fecha</button>
</td></tr>
  </table>
```

Ilustración 45 Formularios Mapas HTML

El código de la imagen superior se encarga de definir los campos del formulario que se encargan de gestionar las fechas que establecen el rango de búsqueda para que el usuario filtre los datos de localización.

El resultado es el siguiente:

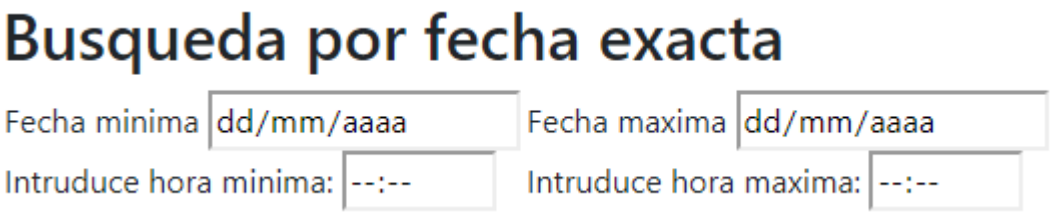

Ilustración 46 Formularios HTML Mapas

A continuación, se procede a explicar el código JavaScript asociado a cada una de las partes HTML:

Gráficas

El código asociado a la creación de las gráficas es el siguiente:

```
15 ▼    let graph= new Morris.Line({
16        element: 'myfirstchart',
17        data: tmps,
18        xkey: 'x',
19        ykeys: ['y'],
20        labels: ['Temperatura'],
21        hideHover: 'auto',
22        resize: true,
23        xmin:0
24      });
25
26
27
28 ▼ let graph2= new Morris.Area({
29        element: 'mysecondchart',
30        data: tmps,
31        xkey: 'x',
32        ykeys: ['y'],
33        labels: ['Temperatura'],
34        //pointSize: 2,
35        hideHover: 'auto',
36        resize: true,
37        xmin:0,
38        lineColors: ['#C14D9F', '#2CB4AC','#2CB4AC']
39      });
```

Ilustración 47 Código Gráficas Morris

Las gráficas se obtienen con la librería Morris.js, mediante la instrucción "Line" se crea la gráfica de línea y mediante "Area" la gráfica de área y se inicializan con los siguientes parámetros:

- **Element:** Elemento HTML donde se va a mostrar la gráfica en cuestión.
- **Data:** Datos que se van a mostrar, en este caso el array de Temperaturas llamado "tmps".
- **Xkey:** Valores que se mostrarán en el eje X.
- **Ykey:** Valores que se mostrarán en el eje Y.
- **Labels y hideover**: Etiqueta que se mostrará al usuario al pasar por encima de un punto concreto.
- **Resize:** Con esta propiedad, la gráfica se autoajustará al número de puntos existentes.

- **Line Colors:** Esta propiedad se encarga de otorgarle color a la línea en cuestión.

Por defecto, el sistema muestra todos los datos mediante una función llamada *conseguirTemperaturas()* que se ejecuta de manera automática al cargar la página.

La función se encarga de obtener mediante una petición GET, los datos en formato JSON referentes a la temperatura para después, en las siguientes líneas dentro dentro de un bucle "for" reconstruir la fecha del JSON y meter después los datos en un array de temperaturas, al final, se actualizan las gráficas para que se muestren en la parte de la interfaz.

Cuando el usuario hace una búsqueda por fecha exacta y pulsa el botón "Buscar" se ejecuta una función llamada *conseguirTempFechaC()* cuya labor es obtener los datos y filtrarlos según las peticiones del usuario.

El código de dicha función encarga de hacer la petición a la url y obtener los datos del formulario que haya introducido el usuario para después comprobar si la fecha que le llega desde el JSON está dentro del rango indicado por el usuario, si es así se mostrará en ambas gráficas.

En el caso de que el usuario quiera filtrar la temperatura de un mes en concreto, se ejecutará la función *conseguirTempMes().* La función se encargará de filtrar las temperaturas que se encuadren dentro del mes que el usuario especifique.

Mapa

En este apartado, se procede a explicar el código referente a la creación e incursión de puntos o puntos de calor en el mapa normal o mapa de calor según corresponda.

```
52  const dibujarMapa = (obj)=>{
53      let mapa =new google.maps.Map(document.getElementById('map'),{
54          center:obj,
55          zoom:4
56      })
57      let marcadorUsuario = new google.maps.Marker({
58          position:obj,
59          title:'Tu ubicacion'
60      })
61      marcadorUsuario.setMap(mapa)
62      let marcadores = lugaresInfo.map(lugar =>{
63          return new google.maps.Marker({
64              position:lugar.posicion,
65              title:lugar.imei,
66              map:mapa
67          })
68      })
69  }
```

Ilustración 48 Creación Mapa Normal

En el código expuesto en la ilustración superior, podemos observar como se crea un mapa asociado al elemento con id *"map"* incrustado en el HTML. Y por otro lado se crea una nueva marca dentro del mapa, además de que se incluye la ubicación del dispositivo que lo ejecuta par que el usuario sepa su oposición.

```
73  const dibujarMapaCalor = (obj)=>{
74      map = new google.maps.Map(document.getElementById('map'), {
75        center: obj,
76        zoom: 4,
77        mapTypeId: 'satellite'
78      });
79      var heatmap = new google.maps.visualization.HeatmapLayer({
80        data: lugaresInfo2
81      });
82      heatmap.setMap(map);
83  }
```

Ilustración 49 Creación Mapa Calor

En el código expuesto en la ilustración superior, podemos observar cómo se crea un mapa de calor asociado al elemento con id *"map"* incrustado en el HTML. Y por otro lado se crea una nueva marca de calor dentro del mapa.

Por defecto, el sistema muestra todos los datos mediante una función llamada *conseguirLugares()* que se ejecuta de manera automática al cargar la página.

Procedo a explicar el código:

La función se encarga de obtener mediante una petición GET, los datos en formato JSON referentes a la localización para después, en las siguientes líneas dentro de un bucle “for” reconstruir la fecha del JSON y meter después los datos en un array de localizaciones, al final, se actualizan el mapa correspondiente con los datos.

Cuando el usuario hace una búsqueda por rango de fechas y presiona el botón “Buscar” se ejecuta una función llamada *conseguirLugaresFecha()* cuya labor es obtener los datos y filtrarlos según las peticiones del usuario.

El código se encarga de hacer la petición a la url y obtener los datos del formulario que haya introducido el usuario para después comprobar si la fecha que le llega desde el JSON está dentro del rango indicado por el usuario, si es así se mostrará en el mapa que corresponda.

5.2.5 Programa para convertir los datos a RDF e inclusión de datos en Virtuoso.

En este apartado, trato el problema de convertir los datos en formato JSON a formato RDF y posterior subida en dicho formato a Virtuoso.

El procedimiento se lleva a cabo en lenguaje Python para generar el fichero en formato RDF de cada uno de los tipos de datos con los que se trabaja y para la subida de los ficheros en formato RDF generados se hace uso de un programa que se encarga de la subida de datos a Virtuoso.

Programa Python para Gestionar Temperaturas.

Se define un grafo que va a contener por un lado las ontologías necesarias para identificar cada uno de los campos del JSON y el valor de dicho campo.

Para elegir ontologías de forma que futuros desarrolladores puedan identificarlas para según el tipo de datos a tratar he consultado la web http://lov.okfn.org/dataset/lov

Según la consulta a la web, la ontología más utilizada para representar datos que capta un sensor de temperatura es

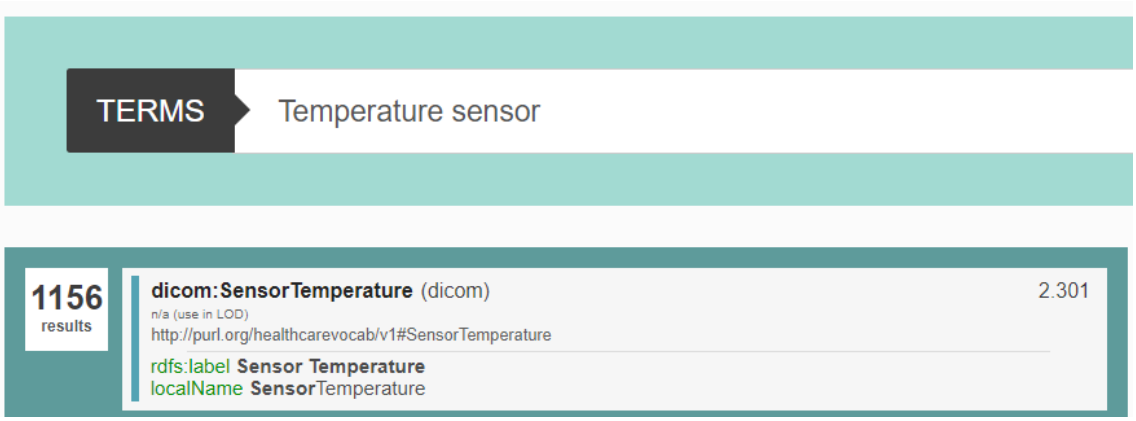

Ilustración 50 Ontologías Utilizadas Temperatura

Para incluir la ontología al grafo lo hacemos con la siguiente instrucción

graph.bind("aws","http://purl.oclc.org/NET/ssnx/meteo/aws#TemperatureSensor"), y para añadir datos cuando se consulten mediante la URL al grafo la siguiente instrucción *graph.add((uri, AWS['], Literal(key['temperatura'], datatype = XSD.Decimal)))*.

El siguiente paso será hacer una consulta a la URL https://tmptfgdgarcia.herokuapp.com/temperatura, que devolverá los datos en formato JSON. Dichos datos serán procesados y agregados al Grafo.

Una vez tengamos los datos en el grafo, será la hora de escribirlos en un fichero con extensión TTL.

Después de la ejecución del programa se generará un fichero TTL con una línea por cada dato en formato JSON capturado:

```
<https://tmptfgdgarcia.herokuapp.com/temperatura/0> ns1:TemperatureSensor "20"^^xsd:decimal ;
    time:at "5/24/18 17:40 PM"^^xsd:dateTime .
```

Ilustración 51 Archivo TTL

En la imagen podemos observar que aparece la URL donde se encuentra el dato, la temperatura en formato decimal y la fecha de captura del dato.

Una vez que se ha generado el fichero TTL, es necesario subir ese archivo a Virtuoso, se va a generar un programa que genere y suba esos datos una vez al día a Virtuoso.

Esta parte, al ser un asunto interno de Open Data Cáceres no se va a implementar en este proyecto, pero me dispongo a detallar las características de ese programa. Igualmente, se han realizado pruebas que aseguran el funcionamiento del código

Una opción es subir a Virtuoso los datos de forma gráfica, pero se va a desarrollar un programa que va a subir dicha información.

En primer lugar, es necesario borrar el grafo que se va a actualizar para que no haya conflictos con los datos o duplicados de los mismos mediante la instrucción:

SPARQL CLEAR GRAPH < https://tmptfgdgarcia.herokuapp.com/temperatura>;

El siguiente paso es limpiar la cola de carga para asegurarnos de que no vamos a introducir nada incorrecto mediante la instrucción *delete from DB.DBA.load_list;*

Seguidamente, se procede a cargar el fichero RDF en la cola de carga, indicando el directorio donde se encuentra el archivo a subir, el archivo y el grafo en el que queremos almacenar los datos.

ld_dir('dir_to_upload/',temperatura.ttl','https://tmptfgdgarcia.herokuapp.com /temperatura ');

Por último, se ejecuta el loader, que será el que en última estancia añada los datos a la triplestore para que puedan ser consultados *rdf_loader_run();* y se limpia la cola de carga *delete from DB.DBA.load_list;*

Programa Python para Gestionar Localizaciones.

Se define un grafo que va a contener por un lado las ontologías necesarias para identificar cada uno de los campos del JSON y el valor de dicho campo.

Para elegir ontologías de forma que futuros desarrolladores puedan identificarlas para según el tipo de datos a tratar he consultado la web http://lov.okfn.org/dataset/lov

Según la consulta a la web, la ontología más utilizada para representar datos que capta un sensor de temperatura es

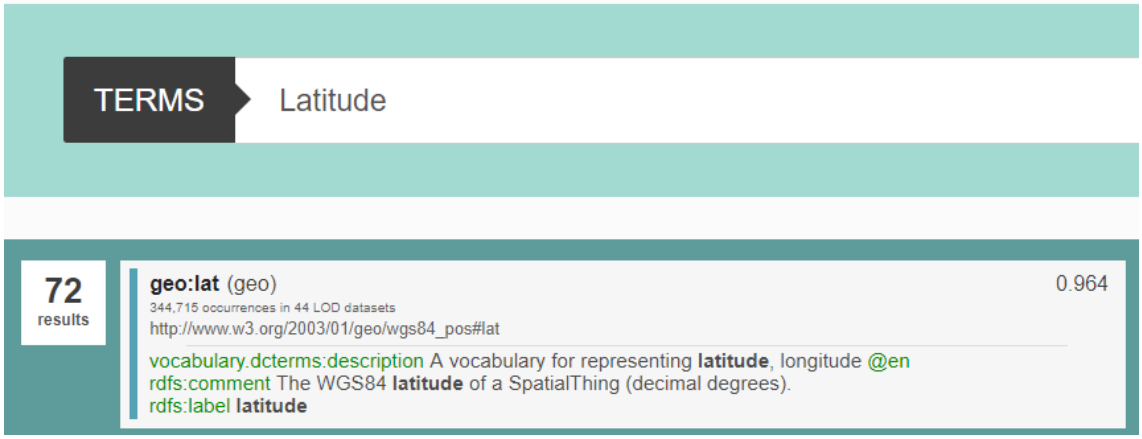

Ilustración 52 Ontologías Utilizadas Localizaciones

Para incluir la ontología al grafo lo hacemos con la siguiente instrucción

graph.bind("geo","http://www.w3.org/2003/01/geo/wgs84_pos#"), y para añadir datos cuando se consulten mediante la URL al grafo las siguientes instrucciónes

graph.add((uri, FOAF['name'], Literal(key['imei'])))

graph.add((uri, GEO['lat'], Literal(key['latitud'], datatype = XSD.Decimal)))

graph.add((uri, GEO['long'],Literal(key['longitud'],datatype = XSD.Decimal)))

graph.add((uri, GEO['alt'],Literal(key['altitud'],datatype = XSD.Decimal)))

graph.add(((uri, TIME['at'], Literal(key['fecha'], datatype=XSD.dateTime))))

El siguiente paso será hacer una consulta a la URL https://geolocalizaciontfgdgarciauex.herokuapp.com/localizaciones, que devolverá los datos en formato JSON. Dichos datos serán procesados y agregados al Grafo.

Una vez tengamos los datos en el grafo, será la hora de escribirlos en un fichero con extensión TTL.

Después de la ejecución del programa se generará un fichero TTL con una línea por cada dato en formato JSON capturado:

```
<https://geolocalizaciontfgdgarciauex.herokuapp.com/localizaciones/1> a geo:SpatialThing ;
    geo:lat "38.9650771"^^xsd:decimal ;
    geo:long "-5.8575522"^^xsd:decimal ;
    time:at "Thu May 24 17:28:44 CEST 2018"^^xsd:dateTime ;
    foaf:name "865199027752595" .
```

Ilustración 53 Archivo TTL Localizaciones

En la imagen podemos observar que aparece la URL donde se encuentra el dato, la Geolocalización compuesta de longitud, latitud y altura y por otro lado, la fecha de captura del dato.

Una vez que se ha generado el fichero TTL, es necesario subir ese archivo a Virtuoso, se va a generar un programa que genere y suba esos datos una vez al día a Virtuoso.

Esta parte, al ser un asunto interno de Open Data Cáceres no se va a implementar en este proyecto, pero me dispongo a detallar las características de ese programa.

Una opción es subir a Virtuoso los datos de forma gráfica, pero se va a desarrollar un programa que va a subir dicha información.

En primer lugar, es necesario borrar el grafo que se va a actualizar para que no haya conflictos con los datos o duplicados de los mismos mediante la instrucción:

SPARQL CLEAR GRAPH<https://geolocalizaciontfgdgarciauex.herokuapp.com/localizaciones>;

El siguiente paso es limpiar la cola de carga para asegurarnos de que no vamos a introducir nada incorrecto mediante la instrucción *delete from DB.DBA.load_list;*

Seguidamente, se procede a cargar el fichero RDF en la cola de carga, indicando el directorio donde se encuentra el archivo a subir, el archivo y el grafo en el que queremos almacenar los datos.

ld_dir('dir_to_upload/',temperatura.ttl',' https://geolocalizaciontfgdgarciauex.herokuapp.com/localizaciones/');

Por último, se ejecuta el *loader*, que será el que en última estancia añada los datos a la triplestore para que puedan ser consultados *rdf_loader_run();* y se limpia la cola de carga *delete from DB.DBA.load_list;*

Cabe destacar, que, para el correcto funcionamiento de este apartado, se han realizado pruebas en Open Data Cáceres con los datos recogidos, de manera que, aunque no estén operativos para el usuario al finalizar este proyecto, el objetivo es que en un futuro inmediato puedan ser alojados en Open Data Cáceres.

5.3 Problemas Encontrados

5.3.1 Aplicación Android

- **Problema con dispositivos Android con API superior a la versión 23:** A partir de la API 23, la gestión de los permisos necesarios para hacer uso de algún componente del dispositivo móvil cambia su filosofía, en versiones anteriores a la 23 bastaba con añadir los permisos al fichero *manifest.xml*, pero a partir de la versión 23 de la API, los permisos se conceden también en tiempo de ejecución. El problema viene al consultar el IMEI del dispositivo, ya que la aplicación necesita los permisos de lectura de parámetros del dispositivo y por la definición de la aplicación, no es posible solicitar permisos en tiempo de ejecución, esto es debido a que hago uso de una clase que implementará un servicio para consultar el IMEI del dispositivo y realizar el envío de datos. Para total compatibilidad con las versiones más actuales de API, sería necesario cambiar toda la estructura básica de la aplicación, de manera que se contempla como una posible mejora.

5.3.2 Montaje y configuración de placa Arduino

- Controlador Wifi desactualizado: El primer problema que encontré al montar la placa y realizar las primeras pruebas de testeo, fue que el controlador Wifi de la placa estaba desactualizado, por lo tanto, tuve que encontrar una solución para este problema. Explico dicha solución paso a paso.

    - El primer paso es descargar el software cuyo nombre es *WiFi101 Firmware Updater* y se encarga de la actualización de dicho controlador, puede descargarse desde la siguiente web:

https://github.com/arduino-libraries/WiFi101-FirmwareUpdater-Plugin/releases/tag/v0.9.2

- Una vez descargado, es necesario crear una carpeta llamada *tools* en el espacio de trabajo Arduino.
- Descomprimir el archivo descargado en la carpeta *tools*, en Arduino IDE hacemos click e Herramientas>Wifi101 Firmware Updater cuyo resultado será:

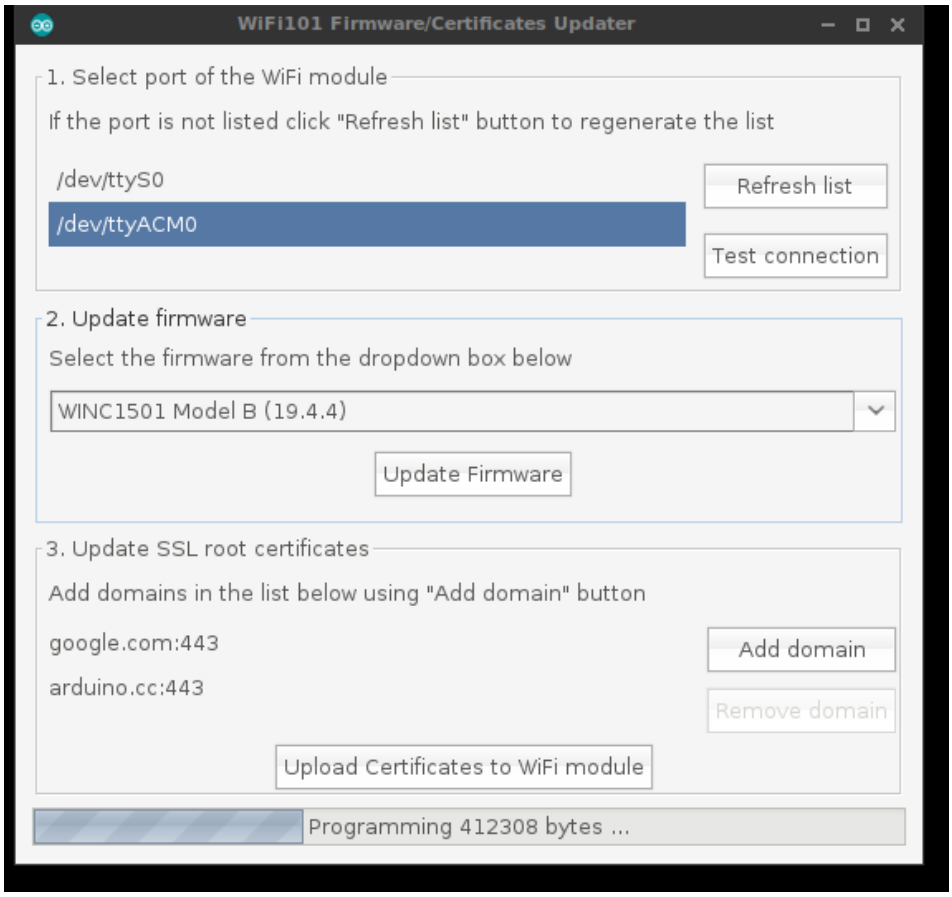

Ilustración 54 Actualizar Controlador Wifi

- En esta pantalla, elegimos el puerto donde se encuentra conectada nuestra placa Arduino, la versión a instalar y hacemos click en *Update Firmware*, una vez instalado nos aparecerá un aviso con la confirmación.

- Para comprobar que realmente se ha actualizado el controlador, podemos hacer uso de un programa de ejemplo que nos proporciona la librería WiFi101. Podemos ejecutar dicho ejemplo haciendo click en

Archivo>Ejemplos>WiFi101> CheckWifi101FirmwareVersion y ejecutando el proyecto con la placa Arduino conectada.

5.3.3 Servidores Heroku

No se han detectado problemas en el desarrollo de los servidores desplegados en Heroku.

5.3.4 Aplicación Web para visualizar datos

- Los problemas detectados en este apartado tienen que ver con la comprensión del flujo de datos JavaScript y de las distintas formas de comunicar la interfaz HTML con los archivos JavaScript, además de el estudio y conocimiento de las distintas librerías para la gestión de datos de forma gráfica y visual.
  Solución: Horas de estudio.

6. Manuales De Usuario

En este apartado serán detallados los distintos manuales de usuario referentes a la utilización de los componentes desarrollados que podrán ser utilizados por los usuarios.

Se detallarán los manuales de usuario referentes a la aplicación Android de Geolocalización.

6.1 Aplicación Android Geolocalización

Este apartado trata sobre el manual de usuario de la aplicación Android para obtener la Geolocalización.

6.1.1 Requisitos

Los requisitos para la utilización de dicha aplicación serán los siguientes:

- ✓ Conexión a internet.
- ✓ GPS.
- ✓ Espacio de almacenamiento disponible de 9.32MB.

6.1.2 Guía De Funcionalidad

Lo primero que hay que hacer es descargar el archivo .apk desde el siguiente enlace:https://drive.google.com/open?id=10c4uVyqN-mq2fukiSCyebHCxHapfLKsC

El siguiente paso es instalar la aplicación, una vez descargado el archivo .apk es necesario hacer click sobre él para lanzar su instalación:

Es necesario que usted revise los permisos que va a conceder a la aplicación, por lo tanto, se proceden a detallar los mismos:

Ilustración 55 Permisos App

Ilustración 56 Instalación App

Si estamos de acuerdo con los permisos que se van a conceder a la aplicación, hacemos click en *Instalar*.

Una vez instalada, será avisado mediante un mensaje de confirmación de instalación. Si hace click en ejecutar le aparecerá la siguiente pantalla

Ilustración 57 Pantalla Bienvenida App

En la imagen superior, usted podrá observar el diseño de la pantalla de bienvenida de la aplicación, desde ese preciso momento, la aplicación, está mandando su ubicación cada 10 minutos a un servidor externo para hacer un uso abierto de sus datos. Si usted hace click sobre el botón *"Ver Mi Ubicación"*, le será mostrado un mapa con su localización actual como el de la siguiente imagen:

Ilustración 58 Mapa Ubicación Usuario

6.2 Aplicación Web De Visualización De Datos

Este apartado trata sobre el manual de usuario de la aplicación web para la visualización de datos referentes a la Temperatura y Geolocalización.

6.2.1 Requisitos

Los requisitos para la utilización de dicha aplicación serán los siguientes:

- ✓ Conexión a internet.
- ✓ Navegador Recomendado *Google Chrome*.

6.2.2 Guía De Funcionalidad

Lo primero que usted puede hacer es acceder a la aplicación web mediante la URL: https://webpageiottfg.herokuapp.com/

En cuanto acceda, esta será su pantalla principal:

Ilustración 59 Página Principal Web

Vemos que en la parte superior de la página hay un menú en el que aparecen las diferentes opciones para visualizar datos, a continuación, se proceden a explicar ambos:

6.2.2.1 Temperaturas

Haciendo click sobre el apartado del menú *Gráficas Temperatura*, obteniendo el siguiente resultado:

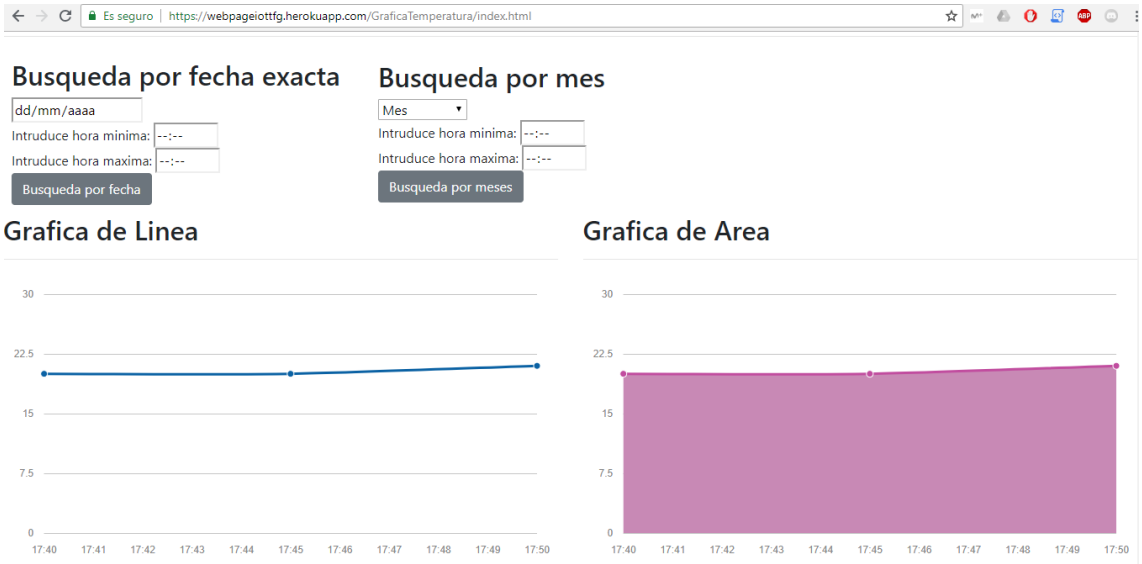

Ilustración 60 Pantalla Gráficas

En dicha pantalla, podemos observar cómo se nos muestran unas gráficas con los datos referentes a la temperatura, dichas gráficas son una gráfica de línea a la izquierda y una gráfica de área a la derecha, estos datos son todos los que tiene el servidor en el momento de la consulta.

Observamos que en la parte superior de la web se encuentran dos apartados llamados *"Búsqueda Por Fecha Exacta"* y *"Búsqueda Por Mes"* que serán detallados a continuación.

6.2.2.1.1 Filtrado Por Búsqueda Exacta

En dicha pantalla, podemos observar que se muestra el siguiente formulario que nos va a permitir filtrar datos de temperatura mediante una fecha exacta:

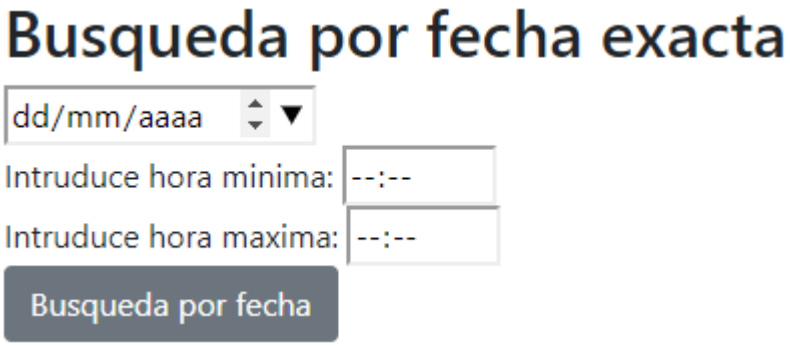

Ilustración 61 Filtro Fecha Exacta

De manera que usted va a poder introducir una fecha exacta y un rango horario mínimo y máximo para visualizar los datos. *A modo de ejemplo, se introduce la fecha 24/05/2018 entre las 17:00 y las 17:45 con el resultado siguiente:*

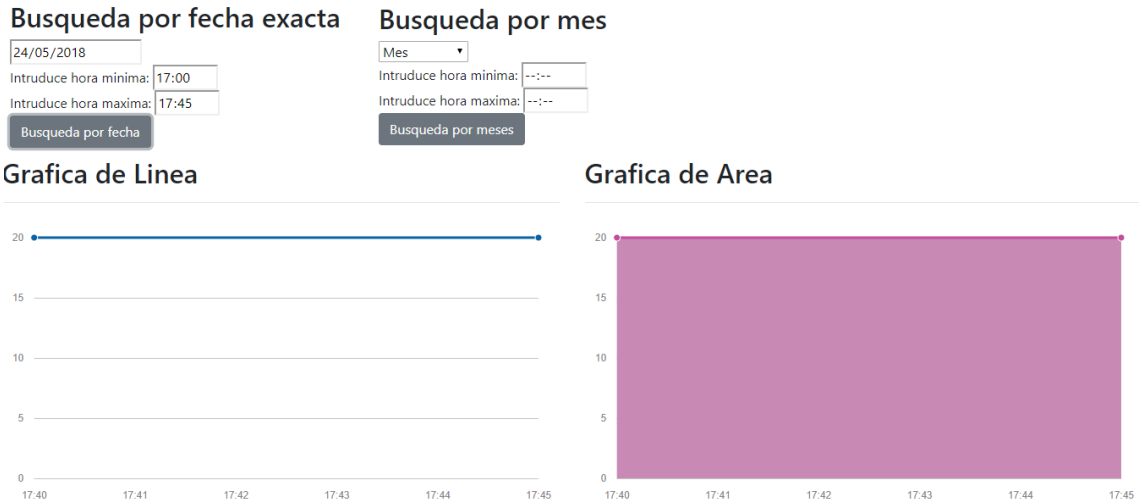

Ilustración 62 Filtrado De Ejemplo

6.2.2.1.2 Filtrado Por Mes.

En dicha pantalla, podemos observar que se muestra el siguiente formulario que nos va a permitir filtrar datos de temperatura mediante un mes concreto:

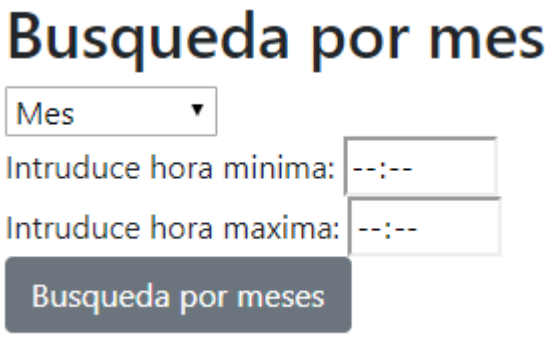

Ilustración 63 Filtrado Por Mes

De manera que usted va a poder introducir un mes exacto para visualizar los datos. *A modo de ejemplo, se introduce el mes de Mayo entre las 17:00 y las 18:45 con el resultado siguiente:*

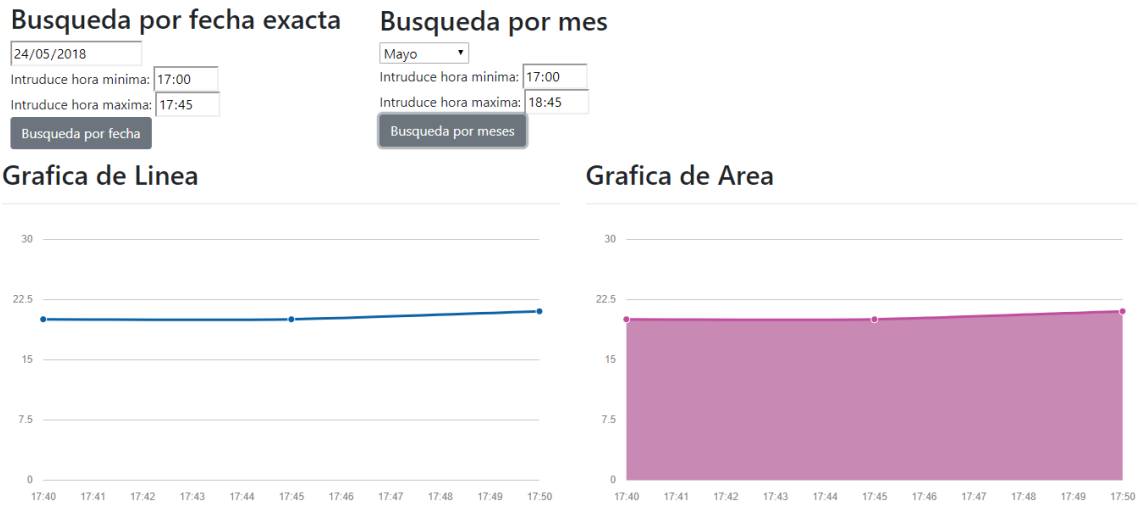

Ilustración 64 Ejemplo Filtrado Mes

6.2.2.1 Mapas Localización

Haciendo click sobre el apartado del menú *Mapas Localización>Mapa*, se obtiene el siguiente resultado:

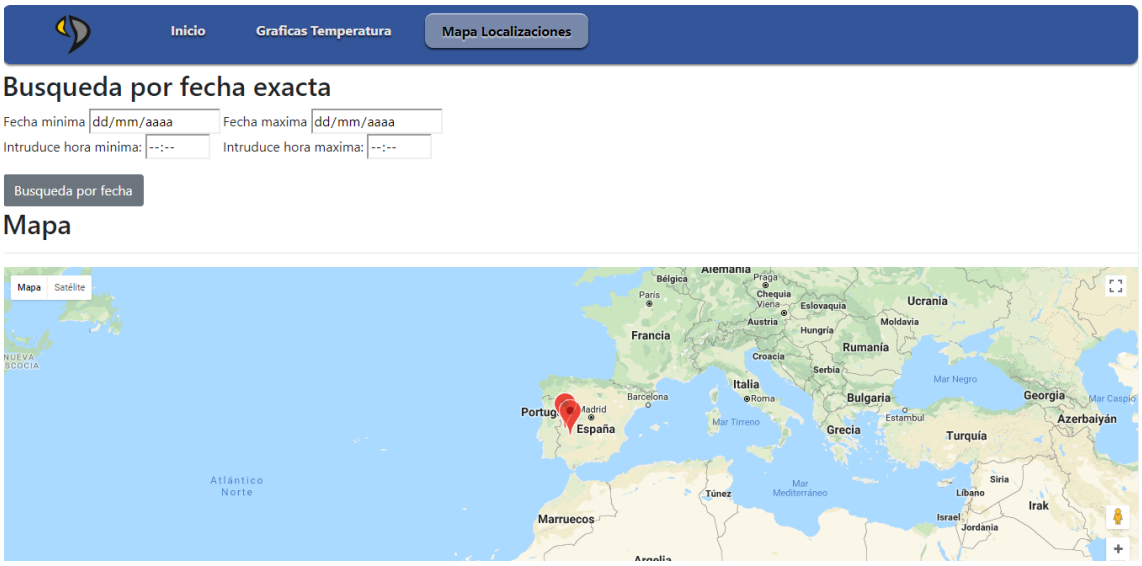

Ilustración 65 Pantalla Google Maps

En dicha pantalla, podemos observar cómo se nos muestra un mapa de Google Maps con Localizaciones, estos datos son todos los que tiene el servidor en el momento de la consulta.

Observamos que en la parte superior de la web se encuentran dos apartados llamados “Búsqueda Por Fecha Exacta” que serán detallados a continuación.

Otra funcionalidad que le brinda la aplicación web es poder hacer zoom sobre alguna posición del mapa mediante los controles:

Ilustración 66 Alejar/Acercar Zoom

Probamos a hacer zoom sobre la comunidad de Extremadura:

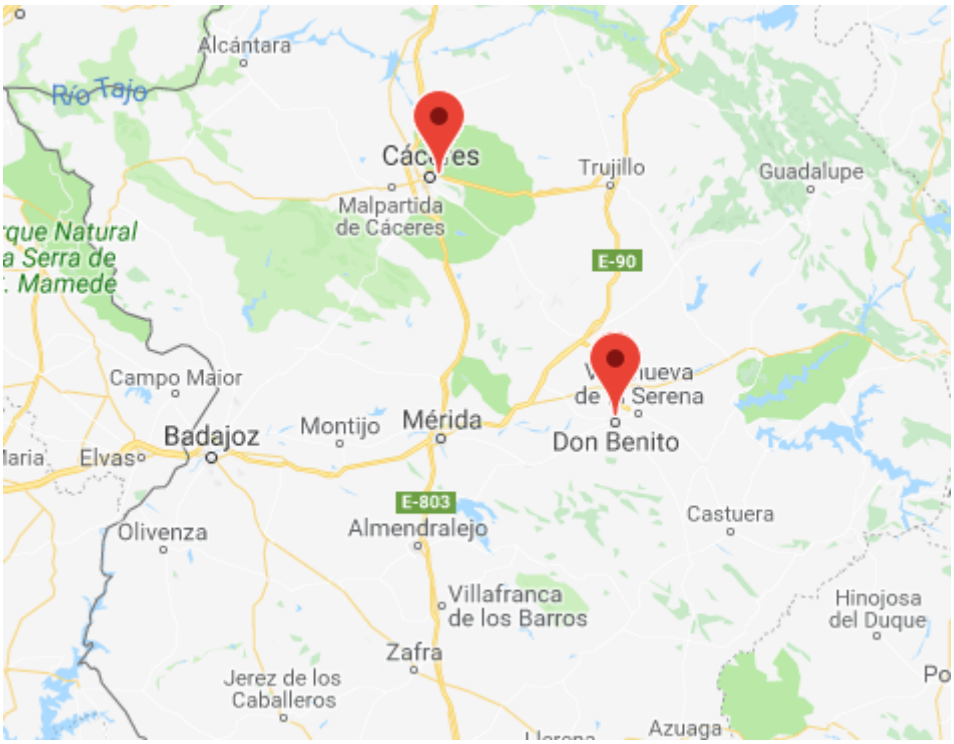

Ilustración 67 Zoom Extremadura

Observamos como aparecen puntos sobre Cáceres y Don Benito, si siguiéramos haciendo zoom, cada vez nos mostraría mas puntos sobre según nos fuésemos acercando.

Por otro lado, si usted hace click en Mapa Localizaciones>Mapa De Calor obtendrá la siguiente información:

Ilustración 68 Mapa Calor Google

6.2.2.1 Filtrado Por Rango De Fechas

El siguiente ejemplo es funcional tanto con un Mapa como con un Mapa de calor.

En dicha pantalla, podemos observar que se muestra el siguiente formulario que nos va a permitir filtrar datos de Geolocalización que estén dentro de un rango de fechas:

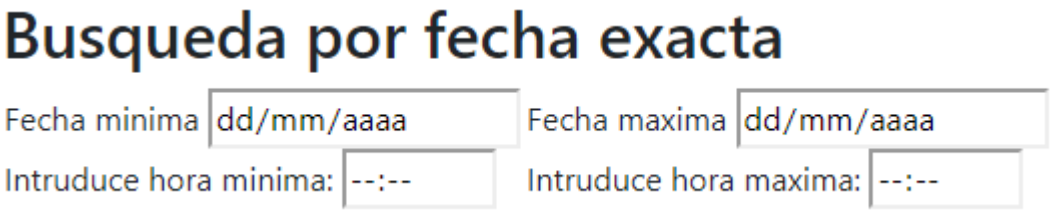

Ilustración 69 Filtrado Por Rango

De manera que usted va a poder introducir un mes exacto para visualizar los datos. *A modo de ejemplo, se introduce el 08/06/2018 a las 10:00 y las 10:00 del 09/06/2018:*

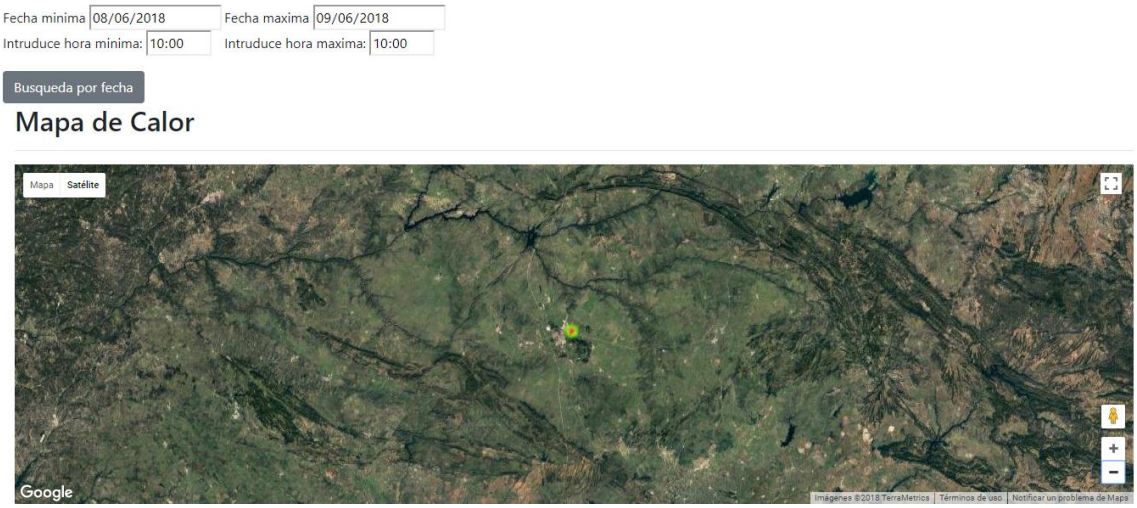

Ilustración 70 Ejemplo Filtrado Por Rango

7. Mejoras

Este apartado trata sobre las posibles mejoras que se pueden aplicar sobre la arquitectura, dichas mejoras van desde la inclusión de nuevos dispositivos de captura de datos hasta la mejora de algún servicio que se utiliza:

- **Utilización de otro tipo de dispositivo para capturar datos de temperatura:** Podría utilizarse algún otro tipo de dispositivo de datos para captar los datos referentes a la temperatura, como podría ser un dispositivo del perfil de Beacon, éste es un dispositivo de pequeño tamaño capaz de capturar datos y transmitirlos utilizando una conexión Bluetooth. El cambio de dispositivo no sería una mejora como tal, pero cambiaría el funcionamiento de la arquitectura.
- **Inclusión de nuevos dispositivos de captura de datos:** Puede ser una gran mejora para la arquitectura añadir algún nuevo sensor que capture más datos aparte de los que ya existen. A modo de ejemplo, me gustaría comentar la posibilidad de añadir un sensor a la placa Arduino que capte la humedad del ambiente, este sensor será un complemento perfecto para los datos de temperatura y daría bastante potencia en cuanto a la gestión de datos a la arquitectura diseñada.
- **Contratar un servidor que permita almacenar todos los datos y no sean eliminados:** Tal y como funciona la arquitectura actual, si las URLS de Temperaturas y Localizaciones no reciben ninguna petición durante 30 minutos, los datos son borrados. Esto es debido a que los servidores están alojados con un plan gratuito en Heroku. De manera que, si se contratase un plan premium, los datos no serían borrados.
- **Modificar la aplicación Android de Geolocalización:** Si se modificara la aplicación Android para que funcionara en dispositivos a partir de la API 23 en adelante supondría que la aplicación podría llegar al 100% de los dispositivos Android del mercado

8. Conclusiones

El capítulo consta de una serie de conclusiones obtenidas después de realizar este proyecto, por un lado, hablare de las conclusiones u objetivos técnicos y por otro, de las conclusiones personales.

8.1 Objetivos Técnicos

En cuanto a los objetivos técnicos puedo concluir lo siguiente:

- Con respecto a la captura de datos de temperatura he aprendido bastante acerca de las posibles alternativas para capturar dichos datos, además de ser capaz de elegir una de ellas en base a criterios de consenso con respecto a la comunidad, además de aprender cómo funcionan las placas Arduino.
- Para la captura de datos con un dispositivo Android, he aprendido como se desarrollan componentes nuevos como por ejemplo la implementación de servicios o la captura de datos referentes a información del dispositivo.
- He comprendido como se gestionan las plataformas Cloud con el uso de Heroku.
- Adquisición de nuevos conocimientos acerca del desarrollo JavaScript y el uso de nuevas librerías.
- Aprendizaje de conceptos básicos del lenguaje Python y cómo funciona alguna funcionalidad de Virtuoso.

8.2 Conclusiones Personales

- Las conclusiones personales que obtengo después de la realización de este proyecto no pueden ser más positivas, por un lado, el aspecto de que no tenga unos plazos establecidos de entregas como he tenido en las asignaturas que he cursado durante mi vida universitaria me ha beneficiado mucho como futuro profesional de las tecnologías de la información, ya que he aprendido a planificarme en el trabajo y marcar un ritmo constante de trabajo.

- Otro de los aspectos con los que más contento estoy es el del gran aprendizaje que he adquirido en cuestiones técnicas. Antes de realizar este proyecto, no tenía apenas conocimiento acerca de las placas Arduino o por ejemplo, no tenía conocimientos prácticos de alojar aplicaciones en la nube, y después de realizar este trabajo de fin de grado soy capaz de montar y configurar una placa Arduino y perfectamente capaz de alojar y gestionar aplicaciones en la nube.
- Una parte esencial y que me llena de satisfacción es la de aumentar mis conocimientos referentes al desarrollo software y al ciclo de vida asociado. Por ejemplo, he aumentado mi conocimiento acerca de tecnologías de desarrollo web como puede ser JavaScript, del que ya tenía algunos conceptos previos gracias a cursar la asignatura de Programación En Internet, pero la flexibilidad que da dicho lenguaje hace que sea muy versátil y haya muchas formas de programar con él. También me alegra haber fortalecido y aumentado conocimientos acerca de desarrollo Android.
- Y por último, aunque no tenga mucho que ver con el desarrollo de este trabajo de fin de grado, estoy muy contento por haber elegido formarme como ingeniero de software, ya que durante estos 4 años he crecido muchísimo como futuro profesional de la informática y cómo persona, ya que han sido unos años muy duros pero han fortalecido mis hábitos de trabajo y me han ayudado a conocer mis límites y ser capaz de aumentar mis conocimientos de manera exponencial.

9. Agradecimientos

Como último apartado, me gustaría agradecer a las siguientes personas el tiempo dedicado a ayudarme con la realización de este proyecto.

Tutor Del Proyecto: Adolfo Lozano Tello.

Personal de Open Data: Julio Acedo Durán y Marcos Fernández Sellers.

Sin su ayuda en algunos de los temas más fundamentales, no podría haber desempeñado este trabajo.

10.Referencias Bibliográficas (Según norma ISO690)

Principios Open Data: https://www.viavansi.com/blog-xnoccio/es/8-principios-de-los-datos-abiertos/

Author	**Azahara Carrillo**
Article title:	Los 8 principios básicos de los Datos Abiertos | Xnoccio.com : Tips, desarrollo de software, firma digital, open data, etc.
Website title:	Xnoccio.com : Tips, desarrollo de software, firma digital, open data, etc.
URL:	https://www.viavansi.com/blog-xnoccio/es/8-principios-de-los-datos-abiertos/

Heroku: https://es.wikipedia.org/wiki/Heroku

Article title:	**Heroku**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/Heroku

JSON Server: https://github.com/typicode/json-server

Article title:	**typicode/json-server**
Website title:	GitHub
URL:	https://github.com/typicode/json-server

Control De Versiones: https://es.wikipedia.org/wiki/Control_de_versiones

Article title:	**Control de versiones**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/Control_de_versiones

Git: https://es.wikipedia.org/wiki/Git

Article title:	**Git**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/Git

Heroku Cli: https://devcenter.heroku.com/articles/heroku-cli

Article title:	**The Heroku CLI | Heroku Dev Center**
Website title:	Devcenter.heroku.com
URL:	https://devcenter.heroku.com/articles/heroku-cli

Android Studio: https://es.wikipedia.org/wiki/Android_Studio

Article title:	**Android Studio**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/Android_Studio

Arduino Ide: https://www.arduino.cc/en/Main/software

Article title:	**Arduino - Software**
Website title:	Arduino.cc
URL:	https://www.arduino.cc/en/Main/software

Sublime Text: https://es.wikipedia.org/wiki/Sublime_Text

Article title:	**Sublime Text**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/Sublime_Text

Esquema funcionamiento placa Arduino: https://www.hackster.io/foxlab/temperature-monitoring-with-arduino-mkr1000-and-artik-cloud-84c351

Author	**Salman Faris**
Article title:	Temperature Monitoring with Arduino MKR1000 and ARTIK Cloud
Website title:	Hackster.io
URL:	https://www.hackster.io/foxlab/temperature-monitoring-with-arduino-mkr1000-and-artik-cloud-84c351

Puesta en marcha la placa MKR 1000: https://create.arduino.cc/projecthub/charifmahmoudi/arduino-mkr1000-getting-started-08bb4a

Article title:	**Arduino MKR1000 Getting Started**
Website title:	Arduino Project Hub
URL:	https://create.arduino.cc/projecthub/charifmahmoudi/arduino-mkr1000-getting-started-08bb4a

JSON-Server: https://github.com/jesperorb/json-server-heroku

Article title:	**jesperorb/json-server-heroku**
Website title:	GitHub
URL:	https://github.com/jesperorb/json-server-heroku

Eclipse: https://es.wikipedia.org/wiki/Eclipse_(software)

Article title:	**Eclipse (software)**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/Eclipse_(software)

Java: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

Article title:	**Java (lenguaje de programación)**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

Python: https://es.wikipedia.org/wiki/Python

Article title:	**Python**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/Python

Spring: https://es.wikipedia.org/wiki/Spring_Framework

Article title:	**Spring Framework**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/Spring_Framework

JQuery: https://es.wikipedia.org/wiki/JQuery

Article title:	**JQuery**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/JQuery

Ajax: https://es.wikipedia.org/wiki/AJAX

Article title:	**AJAX**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/AJAX

HTML: https://es.wikipedia.org/wiki/HTML

Article title:	**HTML**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/HTML

JavaScript: https://es.wikipedia.org/wiki/JavaScript

Article title:	**JavaScript**
Website title:	Es.wikipedia.org
URL:	https://es.wikipedia.org/wiki/JavaScript

WiFi101 Firmawre Updater: https://github.com/arduino-libraries/WiFi101-FirmwareUpdater-Plugin

Article title:	**arduino-libraries/WiFi101-FirmwareUpdater-Plugin**
Website title:	GitHub
URL:	https://github.com/arduino-libraries/WiFi101-FirmwareUpdater-Plugin

WiFi101 Firmware Updater Descarga: https://github.com/arduino-libraries/WiFi101-FirmwareUpdater-Plugin/releases/tag/v0.9.2

Article title:	**arduino-libraries/WiFi101-FirmwareUpdater-Plugin**
Website title:	GitHub
URL:	https://github.com/arduino-libraries/WiFi101-FirmwareUpdater-Plugin/releases/tag/v0.9.2