



**UNIVERSIDAD DE EXTREMADURA
CENTRO UNIVERSITARIO DE MÉRIDA**

**GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA
INFORMACIÓN**

TRABAJO FIN DE GRADO

**Desarrollo de un punto de asistencia
inteligente basado en reconocimiento
de patrones de voz con tecnología
LoRaWAN**

Autor: Javier Serradilla González

Mérida, Junio 2023



**UNIVERSIDAD DE EXTREMADURA
CENTRO UNIVERSITARIO DE MÉRIDA**

**GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA
INFORMACIÓN**

TRABAJO FIN DE GRADO

**Desarrollo de un punto de asistencia
inteligente basado en reconocimiento
de patrones de voz con tecnología
LoRaWAN**

Autor: Javier Serradilla González

Fdo.:

Tutores:

Héctor Sánchez Santamaría

Fdo.:

Pedro José Pardo Fernández

Fdo.:

Agradecimientos

Quisiera agradecer a todas las personas que han hecho posible que hoy estés leyendo este documento.

A mis padres, compañeros y amigos, con los que siempre he podido contar y a los que hoy os dedico estas palabras, aun con vuestros más y vuestros menos habéis conseguido que hoy sea la persona que soy, muchas gracias por el esfuerzo y sacrificio que ha supuesto criarme y educarme, pero tened por seguro que sin esto, no habría llegado a ser lo que soy.

A mis familiares y amigos, os agradezco enormemente la paciencia, el respeto y la compañía que me dais sin pedir nada a cambio, sois el pilar que me ha mantenido a flote durante mis peores años y en el que me he podido apoyar cuando la situación lo requería y os aseguro que cuando me necesitéis encontraréis el mismo pilar que he encontrado yo en vosotros.

A mis profesores, sin vosotros estoy seguro de que ni académica ni personalmente podría haber logrado tanto, sois el eje principal que permite a nuestra sociedad continuar avanzando, no lo olvidéis nunca y muchas gracias por el esfuerzo y la motivación que transmitís en la enseñanza.

A mis tutores, Pedro José Pardo Fernández y Héctor Sánchez Santamaría, que además de ser profesores también han sido consejeros y orientadores en este trabajo, sin ellos no habría sido posible, gracias por guiarme durante el transcurso del mismo y ofrecerme todos los materiales necesarios sin ningún tipo de pega o tiempo de entrega, gracias a vuestra dedicación y esfuerzo este trabajo es como es.

A Juan Luis Vázquez Parra, cuyo TFG, Sistema de control de jardinería de interiores, me ha servido de referencia para organizar las secciones que se plantean en este trabajo.

Mis más humildes gracias a todos, espero poder seguir compartiendo muchos más años y experiencias a vuestro lado.

Javier Serradilla González

Resumen

El 25 de enero de 2023 ocurrió en Algeciras uno de los peores ataques a la sociedad española que han sucedido en 2023, un joven hirió a cuatro personas y mató a un sacristán en un atentado contra las parroquias de San Isidro y La Palma de Algeciras según recogió el periódico El País, este hecho podría haber terminado de otra forma si se hubiera contado con los puntos de asistencia inteligentes descritos en este trabajo.

En este documento se explicarán las diferentes etapas, desde la concepción de la idea del punto de asistencia inteligente a su implementación final, pasando por el análisis, planificación y diseño. Al finalizar se plantearán las conclusiones del mismo y los posibles trabajos futuros que posibilita este trabajo.

La motivación que dio lugar a este trabajo no es otra que la aumentar las opciones de asistencia mediante la distribución de puntos de asistencia inteligentes, de manera que brinden asistencia en distintas situaciones sin necesidad de Internet, empleando para ello la tecnología LoRaWAN y haciendo uso de algo tan común como la voz.

Para poder tener operativos estos puntos que no requieren de Internet, ha sido fundamental el empleo de un servicio recolector compatible con LoRaWAN como es el que ofrece *The Things Network* (TTN), además del desarrollo y entrenamiento de un asistente conversacional (Chatbot) basado en redes neuronales que recoja las peticiones que el usuario comunique.

Finalmente, estas peticiones serán recogidas del servicio recolector por Node-red, suministradas a la aplicación web y gestionadas por personal cualificado para ello, de una forma sencilla, visual y amigable al personal encargado de gestionar dichas peticiones.

Palabras clave: IoT, Redes Neuronales, LoRaWAN, Aplicación Web, Node-red

Abstract

On January 25, 2023, one of the worst attacks on Spanish society that have happened in 2023 occurred in Algeciras, a young man injured four people and killed a sacristan in an attack on the parishes of San Isidro and La Palma de Algeciras, according to the newspaper El País, this event could have ended differently if the intelligent assistance points described in this work had been available.

This document will explain the different stages, from the conception of the idea of the intelligent point of assistance to its final implementation, including analysis, planning and design. At the end, the conclusions of the paper will be presented, as well as the possible future work made possible by this work.

The motivation behind this work is none other than to increase assistance options by distributing intelligent assistance points, so that they can provide assistance in different situations without the need for Internet, using LoRaWAN technology and making use of something as common as voice.

In order to have these points that do not require the Internet operational, it has been essential to use a LoRaWAN-compatible collection service such as the one offered by *The Things Network* (TTN), in addition to the development and training of a conversational assistant (Chatbot) based on neural networks that collects the requests that the user communicates.

Finally, these requests will be collected from the service collected by Node-red, supplied to the web application and managed by qualified personnel, in a simple, visual and user-friendly way for the personnel in charge of managing these requests.

Keywords: IoT, Neural Networks, LoRaWAN, Web Application, Node-red

Índice general

1.	Introducción y Objetivos	1
1.1.	Antecedentes	1
1.2.	Contextualización	2
1.3.	Definición del problema	3
1.4.	Algunos dispositivos similares	4
1.5.	Redes LoRaWAN	5
1.6.	Objetivos del trabajo fin de grado	6
1.7.	Estructura del documento	7
2.	Análisis previo	8
2.1.	Análisis inicial	8
2.2.	Análisis de requisitos	9
2.3.	Soluciones propuestas	10
2.4.	La solución final	11
2.5.	Análisis de viabilidad	13
3.	Planificación del trabajo fin de grado	17
3.1.	Metodología iterativa incremental	17
3.2.	División en paquetes de trabajo	18
3.3.	Organización temporal del trabajo	19
3.4.	Presupuesto de comercialización del trabajo	20
4.	Diseño del punto de asistencia	22
4.1.	Diagramas de casos de uso	22
4.2.	Diagramas de secuencia	25
4.3.	Diseño de la base de datos	28
4.4.	Diseño de la aplicación web	31
4.5.	Diseño del dispositivo físico	44
4.6.	Diseño de las conexiones	48
5.	Implementación del trabajo fin de grado	49
5.1.	Desarrollo del componente físico	49
5.2.	Implementación <i>hardware</i>	52
5.3.	Implementación <i>software</i>	56
5.4.	Implementación de las conexiones	85
6.	Conclusiones y trabajos futuros	107
6.1.	Conclusiones del trabajo fin de grado	107
6.2.	Trabajos futuros	108

7.	Bibliografía	110
8.	Anexo I. Obtención del DevEUI	115
9.	Anexo II. Creación y configuración de un bot de Telegram	116
10.	Anexo III. Manuales de uso de la aplicación web	119
	10.1. Manual de uso del usuario gestor	119
	10.2. Manual de uso del usuario administrador	120
11.	Anexo IV. Instalación de Rasa NLU en Raspberry Pi 4 Modelo B	122

Índice de cuadros

1.	Desglose de elementos <i>hardware</i> necesarios para el punto de asistencia inteligente	16
2.	Distribución de los paquetes de trabajo.....	18
3.	División temporal de las tareas a desarrollar	19
4.	Asignación horaria teórica de los paquetes de trabajo.....	20
5.	Presupuesto de comercialización del producto final.....	21
6.	Diagrama 1 de casos de uso.....	23
7.	Diagrama 2 de casos de uso.....	23
8.	Diagrama 3 de casos de uso.....	24
9.	Diagrama 4 de casos de uso.....	24
10.	Diagrama 5 de casos de uso.....	25

Índice de figuras

1.	Esquema del trabajo fin de grado	12
2.	Vista del dispositivo final	13
3.	Diagrama de secuencia de la realización de peticiones	26
4.	Diagrama de secuencia de la recepción de peticiones.....	27
5.	Diagrama de secuencia de la atención de peticiones	28
6.	Diseño de la base de datos.....	30
7.	Diseño de la vista de inicio de sesión	32
8.	Diseño de la vista de visualización de peticiones no atendidas	32
9.	Diseño de la vista de atención de peticiones.....	33
10.	Diseño de la vista de visualización de peticiones finalizadas	34
11.	Diseño de la vista de inicio de sesión del administrador	35
12.	Diseño de la vista de administración de los dispositivos	36
13.	Diseño de la vista de administración de los gestores.....	37
14.	Diseño de la vista de administración de las peticiones	38
15.	Diseño de la vista de administración de los centros de estudio.....	39
16.	Diseño de la vista para añadir y modificar dispositivos.....	40
17.	Diseño de la vista para añadir y modificar gestores	41
18.	Diseño de la vista para añadir y modificar peticiones	42
19.	Diseño de la vista para añadir y modificar centros de estudio	43
20.	Diseño de la carcasa visto desde AutoCAD.....	44
21.	Diseño del ensamble de los componentes	45
22.	Corte láser de las piezas que componen la carcasa	49
23.	Vista trasera superior de la carcasa	50
24.	Bandeja interior de la carcasa	52
25.	Etapas inicial de la instalación en la carcasa.....	53
26.	Etapas intermedia de la instalación en la carcasa.....	54
27.	Etapas final de la instalación en la carcasa.....	55
28.	Clases que conforman el modelo de la aplicación web.....	58
29.	Clases que conforman las interfaces de servicio de la aplicación web.....	59
30.	Clases que conforman los servicios de la aplicación web	60
31.	Clases que conforman el repositorio de la aplicación web.....	60
32.	Clase controladora de la aplicación web	61
33.	Clase auxiliar para las conexiones internas de la aplicación web.....	62
34.	Diagrama de la implementación Modelo - Controlador de la aplicación web	63

35.	Vista común del inicio de sesión.....	64
36.	Vista de visualización de peticiones no atendidas.....	65
37.	Vista de atención de peticiones.....	65
38.	Vista de visualización de peticiones finalizadas.....	66
39.	Vista de inicio de sesión del administrador	66
40.	Vista de administración de los dispositivos	67
41.	Vista para añadir y modificar dispositivos	67
42.	Vista de administración de los gestores.....	68
43.	Vista para añadir y modificar gestores.....	68
44.	Vista de administración de las peticiones.....	69
45.	Vista para añadir y modificar peticiones.....	69
46.	Vista de administración de los centros de estudio.....	70
47.	Vista para añadir y modificar centros de estudio.....	70
48.	Directorio de trabajo de la implementación <i>software</i> de la Raspberry Pi del dispositivo físico.....	73
49.	Vista inicial de The Things Network.....	86
50.	Vista de servicios de The Things Network.....	86
51.	Creación de una aplicación de The Things Network	87
52.	Vista de una aplicación personal en The Things Network.....	87
53.	Vista de configuración de un dispositivo en The Things Network	88
54.	Vista de configuración del <i>payload</i> en The Things Network.....	89
55.	Vista de configuración de la API en The Things Network.....	90
56.	Vista de inicialización de Node-red.....	91
57.	Configuración de MQTT en Node-red I.....	92
58.	Configuración de MQTT en Node-red II.....	93
59.	Configuración de MQTT en Node-red III.....	94
60.	Recepción de mensajes en Node-red.....	95
61.	Configuración del elemento Mysql en Node-red I.....	96
62.	Configuración del elemento Mysql en Node-red II	97
63.	Vista de Node-red con los elementos de MySQL.....	100
64.	Configuración del elemento switch en Node-red	101
65.	Vista final del elemento MySQL con Node-red.....	102
66.	Configuración del elemento Telegram Sender con Node-red I	103
67.	Configuración del elemento Telegram Sender con Node-red II.....	104
68.	Vista completa de la implementación de Node-red.....	106
69.	Vista de la ejecución del archivo FirstConfiguration con el DevEUI.....	115
70.	Vista del Bot Father en Telegram I.....	116
71.	Vista del Bot Father en Telegram II.....	117

Índice de códigos

1.	Script de creacion de la base de datos	56
2.	Código de configuración del "application.properties"	71
3.	Código para comenzar a configurar Rasa	72
4.	Código de configuración del fichero "nlu.yml" I.....	74
5.	Código de configuración del fichero "nlu.yml" II	74
6.	Código de configuración del fichero "rules.yml".....	74
7.	Código de configuración del fichero "stories.yml"	75
8.	Código de configuración del fichero "config.yml"	75
9.	Código de configuración del fichero "endpoints.yml"	75
10.	Código de configuración del fichero "domain.yml"	76
11.	Código de configuración del fichero "actions.py"	77
12.	Código para comenzar a entrenar el modelo de inteligencia artificial en Rasa .	77
13.	Código del main.py	77
14.	Código del script "startRasaActions.sh"	82
15.	Código del script "startRasaServer.sh"	82
16.	Código del script "startRasaMain.sh"	82
17.	Código del servicio "iniciarNou.service"	83
18.	Códigos para sincronizar el servicio con el sistema.....	83
19.	Código de configuración del Arduino MKR 1310 WAN.....	84
20.	Código del decoder de The Things Network.....	89
21.	Código para instalar Node-red	91
22.	Código para iniciar Node-red.....	91
23.	Código para instalar el paquete MySQL en Node-red.....	96
24.	Código del elemento Insercion en Node-red	98
25.	Código del elemento Consultar Aula en Node-red.....	99
26.	Código del elemento Volcar Aula en Node-red	101
27.	Código para instalar los paquetes del bot de Telegram en Node-red.....	103
28.	Código del elemento Notificar en Node-red.....	105
29.	Código de creación del entorno virtual usando Conda.....	122
30.	Código que muestra las dependencias de Rasa para Conda	122
31.	Código para solucionar el error de Conda	124
32.	Código de activación del entorno virtual usando Conda.....	124
33.	Código para solventar el error de Protobuf.....	124
34.	Código para solventar el error de Packaging version	124

1. Introducción y Objetivos

Este capítulo tiene por objetivo presentar la idea de partida que nos permitirá llegar al resultado final, es decir, el punto de asistencia inteligente, para ello en primer lugar se describirán los antecedentes previos al resultado, la contextualización del mismo junto a algunos dispositivos que cumplen objetivos similares. Posteriormente, se tratarán los objetivos que busca alcanzar este trabajo fin de grado y finalmente con la estructura del documento se concluirá este capítulo.

1.1. Antecedentes

Desde que el ser humano tiene uso de razón comienza una vida en sociedad, la cual aunque en la mayoría de ocasiones ofrece un gran número de ventajas y oportunidades, en ocasiones suceden casos que nos generan situaciones de indefensión o miedo [1].

Para solventar este tipo de situaciones problemáticas, las distintas sociedades han recurrido a diferentes organismos, como son las fuerzas del orden, la administración de justicia, los servicios de emergencia, etc.

Sin embargo, no siempre tenemos la disponibilidad o la capacidad de dirigirnos a estos organismos especializados cuando la situación lo requiere. Si bien es cierto que a día de hoy casi cualquier individuo es portador de un teléfono móvil, en ocasiones puede no ser suficiente para poder afrontar ese caso de necesidad, ya que la cobertura, la falta de batería o mismamente un hurto pueden privarnos de la capacidad de emplearlo. Es por ello que los puntos de asistencia inteligentes que propone este trabajo fin de grado pueden reforzar los ya existentes sistemas de seguridad y emergencia de nuestras sociedades.

Cabe destacar que el concepto de inteligencia, aunque ha estado muy presente en nuestro día a día, desde los conocidos *Smartphones* o teléfonos inteligentes [2] realmente no se podría hablar de inteligencia desde un planteamiento más racional hasta que no se incorporaron las Inteligencias Artificiales a los dispositivos electrónicos en general [3].

Finalmente, no ha sido hasta la llegada de las tecnologías *Long Range Wide Area Network* o LoRaWAN [4] que la idea de un punto de asistencia inteligente, independiente de Internet, a la hora del envío de las peticiones de asistencia ha sido posible, permitiendo a los usuarios de los puntos de asistencia un envío de sus peticiones a las autoridades competentes, aún con el inconveniente de la cobertura de red. Pero las ventajas del uso de LoRaWAN no acaban con la posibilidad de transmitir información a grandes distancias, también puede desarrollar un papel fundamental en la comunicación entre dispositivos que sean compatibles con el concepto de *Internet of Things* o IoT [5].

1.2. Contextualización

Gracias al constante uso del IoT en la vida cotidiana surgen diferentes casos de uso y necesidades en diferentes ámbitos como son la domótica, agricultura y ganadería, salud, industria, etc. Y pese a ser ámbitos tan diversos, es posible agrupar en un denominador común las características de las necesidades de estos, las **peticiones**, que tratan de sintetizar mediante las llamadas **palabras clave** situaciones o elementos que poseen gran importancia en el mensaje y que han de ser transmitidos, similar al concepto de *Intents* que se emplean en Android [6].

La idea de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN trata de conseguir una asistencia efectiva y flexible de manera que se pueda adaptar a cualquier ámbito. Y es la modularidad de la arquitectura del sistema planteado la que la posibilidad esta adaptabilidad, puesto que al hacer uso de *Neural Networks* o Redes Neuronales [7] bastaría con un simple reentrenamiento de estas para adaptarse a las necesidades específicas del nuevo entorno en el que se trabaje.

Otra característica destacable del sistema propuesto es la alta escalabilidad que proporciona la infraestructura subyacente a LoRaWAN, ya sea mediante un único gateway es posible interconectar cientos de dispositivos y a su vez es posible incorporar más gateways a la infraestructura, con lo cual los dispositivos interconectados pueden ser virtualmente infinitos [8].

1.3. Definición del problema

El problema que este TFG trate de afrontar es el dar asistencia al usuario aun cuando este no tenga disponibilidad de conexión a Internet y con una interacción persona-dispositivo más cercana.

Para ello serán necesarios los siguientes elementos:

1. Un dispositivo *hardware* capaz de hacer de pasarela para el envío de la petición a la red de LoRaWAN. Este ha de poder asistir al usuario de manera inteligente, logrando que el usuario pueda transmitirle su petición sin necesidad de esfuerzo.
2. Una implementación para gestionar las peticiones, enviarlas a la base de datos y notificar al personal encargado de atender las peticiones por medio de un servicio de notificaciones.
3. Un servicio de notificación para que el personal encargado de atender las peticiones sean conscientes de la llegada de nuevas peticiones al servicio receptor.
4. Un servidor capaz de levantar la aplicación web para permitir el acceso del personal encargado de atender las peticiones a la propia aplicación web.
5. Una aplicación web que permita al personal encargado de atender las peticiones tener de manera clara y ordenada las mismas, basándose en la prioridad.
6. Un servicio compatible con LoRaWAN, capaz de recibir las peticiones del dispositivo y poder reenviar dichas peticiones al servidor web y servicio de notificación.

1.4. Algunos dispositivos similares

Existen en el mercado dispositivos como altavoces inteligentes o dispositivos de teleasistencia de uso en el hogar que podrían desempeñar una función similar a la del sistema planteado en este trabajo:

Amazon Alexa: Alexa es un producto similar al que se plantea en este TFG, pero con una perspectiva diferente, como tal Alexa es un servicio de voz ubicado en la nube de Amazon [9] partiendo de esta premisa estaríamos hablando de un servicio completamente Online, a diferencia del objetivo de este trabajo fin de grado que emplea la integración de LoRaWAN en el propio ecosistema para no depender de una conexión a Internet a la hora del envío de peticiones. Obviamente, el enfoque de Alexa es mucho más diverso y permite infinidad de funcionalidades que este trabajo fin de grado mucho más específico no alcanza ni lo pretende.

Google Home: Google Home es otro producto similar al que se plantea en este TFG, pero con un objetivo mucho más específico, en este caso Google Home es una aplicación que permite automatizar y gestionar dispositivos en tu hogar compatibles con este servicio de Google [10]. Es por ello, que a diferencia del objetivo de este trabajo fin de grado que no se enfoca principalmente en interconectar dispositivos, monitorizar y automatizar, Google Home cumple con un ámbito similar pero distinto, ya que el tratamiento de peticiones tal y como se entienden en este trabajo fin de grado se realiza para el ámbito específico del hogar y el objetivo del producto de Google abarca un rango de aplicación diferente y más extenso que el de este trabajo fin de grado.

Apple Home Pod: es un altavoz inteligente diseñado por Apple [11] similar al que plantea Google Home, pero funcionando con el enfoque de infraestructura de Apple, es decir, compatible con todos los servicios que la compañía ofrece. En comparación al sistema que propone este trabajo fin de grado, las diferencias son notorias, como el uso de Internet para utilizar sus funcionalidades, destacando la necesidad de Siri, su asistente de voz, de una conexión a sus servicios en la nube. El uso de unos dispositivos concretos seleccionados por Apple, etc.

Dispositivos de monitorización remota en atención sanitaria [12]: Se basan en la bioingeniería, concretamente en el área de la sensorización, su objetivo es el leer, medir y monitorizar parámetros bioquímicos y físicos del usuario a fin de tenerlo atendido y controlado de forma remota y en tiempo real. Estos dispositivos aportan seguridad al usuario y son una solución similar a la que se propone en este TFG; sin embargo, el enfoque es distinto, puesto que los puntos de asistencia inteligentes no se trasladan junto al paciente ni miden parámetros de ningún tipo del usuario, en su lugar tienen un uso más generalizado y adaptable a las necesidades del cliente que los solicite, pudiendo personalizarse y atender peticiones de diversa índole.

Sistema de ayuda a la navegación exterior para personas con discapacidad visual en transporte público [13]: A diferencia de los anteriores dispositivos referenciados, este sistema está desarrollado para ambientes exteriores, si se compara con el sistema propuesto en este trabajo, las diferencias principales serían el ámbito de aplicación, puesto que el sistema de navegación está dirigido al transporte público mientras que el descrito en este trabajo es de ámbito general y el público objetivo, ya que el sistema propone asistencia para personas que presentan discapacidad visual y el del sistema de asistencia inteligente es un público más generalizado, que podría o no presentar algún tipo de discapacidad.

1.5. Redes LoRaWAN

LoRaWAN es un protocolo de red de punto a multipunto, destacando ser un protocolo de capa de control de acceso medio (MAC) con algunos componentes de capa de red. Además, es completamente bidireccional y usa la tecnología *Long Range* o LoRa para redes de baja potencia que cubren un área extensa, se encuentra bajo el grupo de las *Low Power Wide Area Network* o LPWAN, redes que se caracterizan por requerir bajo consumo energético para funcionar (duración de las baterías de hasta 10 años) y cubrir áreas extensas. Además, permiten la comunicación y administración de dispositivos LoRa [14].

Las redes que emplean LoRaWAN se pueden dividir en dos apartados, o niveles, siendo estos:

Nivel físico

En lo que respecta al nivel físico, LoRaWAN presenta una serie de características particulares, siendo estas:

- **Frecuencia de trabajo:** 868 MHz en Europa, 915 MHz en América, y 433 MHz en Asia.
- **Licencia de uso:** Funciona bajo NetID y es gestionada por la *LoRa Alliance*, ofertando un estándar global abierto con capacidades de *roaming* completas [15].
- **Rango de alcance:** 5 km en interurbano y hasta 15 km en condiciones perfectas.

Nivel de Red

En lo referente al nivel de Red, existen compañías como The Things Network [16] que ofrecen un ecosistema global colaborativo de Internet de las cosas, en el que se crean redes, dispositivos y soluciones empleando para ello LoRaWAN. Destacando uno de sus múltiples usos como servicio recolector de peticiones, encargado de recibir las peticiones desde un dispositivo emisor compatible con LoRaWAN y transmitirlos a otros servicios en Internet.

1.6. Objetivos del trabajo fin de grado

Los objetivos que pretende conseguir este trabajo fin de grado son los siguientes:

Objetivo general

El objetivo principal de este trabajo fin de grado es construir un dispositivo de asistencia inteligente cuyo funcionamiento esté basado en reconocimiento de patrones de voz muy concretos y haga uso de tecnología LoRaWAN para brindar asistencia en áreas que carezcan de conexión a Internet.

El sistema ha de ser capaz de analizar la voz del usuario que requiere asistencia y transmitir la petición obtenida de dicha entrada por voz, de manera sintetizada a una red compatible con LoRaWAN, además de la localización de la petición, a fin de poder ser atendida.

Finalmente, se busca plantear un análisis de la eficiencia energética que permita arrojar algo de luz sobre la viabilidad de este producto en distintas situaciones y usos.

Objetivos específicos

A un nivel más profundo y concreto, el propósito del trabajo fin de grado es trabajar con la red neuronal, que es la encargada de la interacción entre persona y dispositivo, de manera que las respuestas sean lo más humanamente posibles, tratando de aportar un poco de tranquilidad y trato más cercano al usuario.

Esto es posible gracias a la capacidad de reentrenamiento de la propia red, la cual puede prepararse para que tome conversaciones previas con usuarios para su posterior mejora, pues mediante refuerzo positivo es capaz de ir perfeccionando las futuras conversaciones para conseguir lo antes propuesto.

Otro punto a tener en cuenta sería el emplear las tecnologías anteriormente descritas como Node-red, bots de Telegram, bases de datos SQL y aplicaciones Web.

1.7. Estructura del documento

El objetivo de este documento es que el lector sea capaz de entender secuencialmente desde la concepción de la idea del punto de asistencia hasta el momento de su creación e implementación del ecosistema subyacente, al mismo que permite la correcta recepción y atención de las peticiones.

En este **apartado introductorio** se ha hablado de los elementos previos necesarios que han de ser especificados previamente al desarrollo de la idea, o dicho de otra forma, se han establecido las bases orientativas de la conceptualización del trabajo fin de grado.

En el siguiente capítulo se realizará el **análisis previo del sistema** tratando de recolectar en un único apartado las tecnologías que se emplearán e integración en la solución definitiva a realizar. Esta será explicada detalladamente y expuesta junto a un análisis de viabilidad del producto desde la óptica del desarrollador de la idea.

El tercer capítulo será destinado a la **planificación del trabajo fin de grado**, desde la planificación teórica inicial, incluyendo la planificación temporal por paquetes de trabajo y el hipotético presupuesto que conllevaría implementar este trabajo fin de grado en la vida real.

El cuarto capítulo tendrá por objetivo el **diseño del punto de asistencia**. En el que se detallarán casos de uso del sistema con sus correspondientes diagramas de secuencia, a fin de transmitir de forma clara y concisa al lector, con qué fin se emplea el dispositivo de asistencia. También se detallará el hardware empleado y las conexiones entre los distintos dispositivos que componen el ecosistema del trabajo fin de grado.

El quinto capítulo o **implementación del trabajo fin de grado** explicará de forma detallada las configuraciones software de los distintos componentes hardware, de manera que el lector tenga la posibilidad de realizar su propia implementación basada en este trabajo fin de grado previamente.

En el capítulo final, **conclusiones y trabajos futuros**, se explicará de forma sintetizada el desempeño que ha supuesto este trabajo para el autor del trabajo fin de grado, así como posibles mejoras o nuevas áreas a explorar en el desarrollo del concepto que ha llevado a esta idea de trabajo fin de grado.

El apartado de la **bibliografía** incluirá de forma esquematizada las diferentes fuentes de información que se han empleado para el desarrollo y conceptualización de la idea de trabajo fin de grado.

Finalmente, en el apartado de **anexos** se especificarán detalles técnicos que ayudarán al lector a comprender elementos fundamentales asociados a los diferentes subapartados del capítulo de implementación del trabajo fin de grado. Así como una guía de instalación de los puntos de asistencia para su correcto funcionamiento.

2. Análisis previo

Este capítulo tiene por objetivo analizar el problema definido en el capítulo introductorio, para ello en un primer análisis, a grandes rasgos, se establecerá un punto de partida para posteriormente poder profundizar en los requisitos de funcionamiento que tendrán los distintos elementos del punto de asistencia. Posteriormente, se expondrán las diferentes alternativas de diseño por las que ha pasado el trabajo fin de grado hasta llegar al diseño final, que será el elegido para ser implementado. Este diseño final será desarrollado junto a su respectivo análisis de viabilidad del sistema, con el fin de comprobar que el dispositivo de asistencia sea una solución viable en lo que a la implementación técnica y económica se refiere en el mercado tecnológico actual.

2.1. Análisis inicial

El concepto de punto de asistencia recoge un amplio espectro de enfoques a simple vista, por ello una vez elegido el tipo de asistencia, en este caso por voz, es necesario establecer de que manera va a reaccionar el dispositivo a esa entrada por voz, además comienza a ser necesario hardware concreto, es decir, un micrófono capaz de registrar esa voz para su posterior tratamiento.

En primer lugar, será necesario determinar la interacción máquina - persona, aunque es posible realizar esta interacción mediante elementos visuales como podrían ser monitores, pantallas de uso general, etc. En este caso se ha optado por la respuesta mediante voz sintética, destacando en este caso el siguiente elemento hardware, un altavoz, uno de los puntos fuertes de esta decisión es que las personas prefieren interactuar con asistentes conversacionales por voz [17].

En segundo lugar, hay que decidir de qué manera será tratada la petición de este, para ello existen varias alternativas aunque la mayoría coinciden en un elemento, siendo este el empleo de un *Speech to Text* o conversor de voz a texto, llegados a este punto hay que recordar una de las premisas originales del dispositivo "brindar asistencia sin necesidad de que el usuario tenga conexión Internet", es en este punto donde las posibilidades que nos ofrece el mercado tecnológico actual se limitan, ya que la mayoría de cuota de mercado la recogen los asistentes que cuentan con acceso a Internet [18]. Es por ello que se va a optar por un conversor sin acceso a Internet.

Es hora de determinar de qué manera se va a realizar el tratamiento de la voz del usuario, una opción muy sencilla sería comparar la entrada textual obtenida de la voz con una serie de palabras claves y actuar en consecuencia, pero este tipo de interacción además de ser más

artificial al usuario carece de posibilidad de entrenamiento y mejora, es por ello que el empleo de una red neuronal se hace obligatorio.

Una vez hemos definido los elementos de interacción debemos definir el dispositivo físico, este ha de ser compatible con micrófono, altavoz y a su vez ser capaz de procesar la entrada por voz y dar una salida, es por ello que se ha optado por el uso de una Raspberry Pi de 8Gb modelo B, el único inconveniente es que de forma nativa este dispositivo es incompatible con LoRaWAN, pero sí que permite la conexión serial [19]. Es por ello que se requiere de un dispositivo que funcione como antena, es decir, capaz de transmitir la información procesada del dispositivo, la petición, mediante LoRaWAN, en este caso se emplea un Arduino MKR WAN 1310. Pero la petición ha de ser enviada a algún lugar, el organismo destinado a esta función será The Things Network, ya que ofrece servicio de forma gratuita y estable.

Tras haber enviado la petición por LoRaWAN, esta se encuentra en los servidores de The Things Network y más concretamente en nuestra sección de aplicaciones.

- Es en este punto donde haremos uso de Node-red para dos aspectos importantes:
 1. Poder transmitir la información a nuestro servidor para que a su vez se vuelque en la base de datos y mantenga la aplicación web actualizada.
 2. Notificar a todos los gestores de una nueva petición por vía telefónica, con el fin de aumentar la disponibilidad de recepción.

2.2. Análisis de requisitos

Partiendo de la base del análisis inicial, la cual emplea LoRaWAN como soporte para el envío de datos a través de The Things Network, el siguiente paso será conocer en detalle los elementos que darán funcionamiento al punto de asistencia inteligente, los cuales se detallarán a continuación basándose en su función.

Los elementos que permiten que la petición sea recogida y enviada son:

- **Rasa NLU:** Software encargado de extraer información estructurada de los mensajes de los usuarios. Esto suele incluir la intención del usuario y cualquier entidad que contenga su mensaje [20].
- **Raspberry Pi de 8Gb de RAM con 16Gb de ROM:** Dispositivo compatible con las especificaciones previas, micrófono y altavoz. Se encargará del procesamiento de las peticiones del usuario.

- **Arduino MKR WAN 1310:** Dispositivo encargado de la transmisión de la petición procesada por la Raspberry Pi haciendo uso de LoRaWAN a The Things Network.

Los elementos que permiten que la petición sea procesada y registrada son:

- **Nodo en Node-red:** Software encargado de la transmisión de la petición a la base de datos SQL que porta el servidor y posterior notificación de llegada de nueva petición a los gestores por vía telefónica.
- **Servidor:** Dispositivo encargado del funcionamiento continuo de la base de datos y consecuentemente de la aplicación web.

Los elementos que permiten que la petición sea atendida son:

- **Canal de Telegram:** Medio encargado de la recepción de las notificaciones entrantes del nodo de Node-red, es decir, las nuevas peticiones.
- **Aplicación web:** Aplicación no nativa que permite a los gestores la administración de las peticiones de forma clara y fiable.

2.3. Soluciones propuestas

La idea final que se ha optado por materializar en este trabajo ha sido fruto de múltiples versiones y variaciones que han conducido a la misma.

Una de las soluciones candidatas que podía haber sido elegida era prescindir de la red neuronal encargada del procesamiento de la entrada por voz del usuario y emplear únicamente el conversor de voz a texto y filtrar en dicho texto resultante por las palabras clave; sin embargo, esta opción aunque a simple vista hubiera resultado en una finalización mucho más temprana del punto de asistencia, hubiera desembocado en una pérdida de la experiencia de usuario.

Otra solución candidata era haber desarrollado el dispositivo de asistencia en otro componente hardware distinto a la Raspberry Pi debido a su coste elevado y baja disponibilidad de unidades disponibles a la venta [21]. Sin embargo, en esta decisión influyen la capacidad de adaptación y los plazos de entrega, si bien es cierto que pueden existir dispositivos candidatos, como son la Nvidia Jetson Nano, ASUS Tinker Board S, Orange Pi, etc. [22]. La posibilidad de adaptar este sistema a otro hardware es compleja y costosa temporalmente hablando.

Finalmente, cabe destacar la posibilidad de haber realizado una aplicación móvil en lugar de haber destinado esta tarea a los bots de Telegram, pero al igual que se mencionó en el punto

anterior y se explicará con más detalle en el capítulo 3, el tiempo es un factor determinante que impide destinar más recursos a este apartado.

2.4. La solución final

Basándonos en distintos factores como son la disponibilidad de material, el abaratamiento de costes y la personalización del trabajo fin de grado, los elementos elegidos han sido los siguientes:

1. El dispositivo sobre el que se fundamenta este trabajo, en este caso, ha sido una Raspberry Pi de 8Gb de memoria RAM con 16Gb de memoria ROM, a su vez conectado a un Arduino MKR1310 que hará de pasarela para el envío de la petición a la red de LoRaWAN de The Things Network.
2. Una implementación en Node-Red [23] para gestionar las peticiones, enviarlas a la base de datos MySQL [24] y notificar a los gestores de las peticiones por vía telefónica con un bot de Telegram [25].
3. Un bot de Telegram configurado para la interconexión con Node-Red.
4. Un servidor capaz de levantar la aplicación web para permitir el acceso del personal encargado de atender las peticiones a la propia aplicación web.
5. Una aplicación web basada en Java Spring Boot [26] y cuyos datos se almacenan en la anteriormente descrita base de datos MySQL.

El esquema de la aplicación hace uso de los elementos anteriormente citados, tal y como podemos observar en la siguiente figura (ver figura 1).

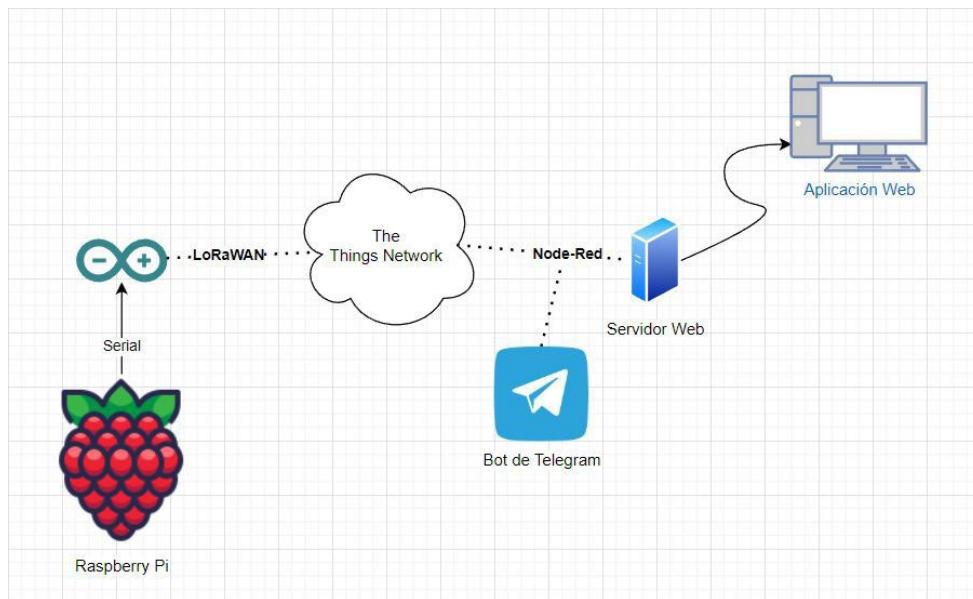


Figura 1: Esquema del trabajo fin de grado

El funcionamiento del esquema planteado en la **Figura 1** es el siguiente, una vez el usuario se acerca al dispositivo y envía su petición, este la procesa, la sintetiza y la envía a través de la infraestructura subyacente a LoRaWAN. Una vez ha llegado al servidor, este la deriva a dos elementos claves gestionados por Node-Red:

La integración con Telegram [27], que haciendo uso de los bots que nos ofrece la propia API o Interfaz de Programación de Aplicaciones de la aplicación, podemos notificar a los gestores de las peticiones por vía telefónica y sin coste adicional, recibiendo un simple mensaje de texto con el tipo de petición y el lugar donde se ha realizado.

La integración con MySQL, que realiza un volcado de la petición a la base de datos, la cual a su vez permitirá a la aplicación web [28] enfocada en la monitorización de las peticiones, obtener los datos actualizados y facilitar a los gestores de las peticiones la distribución y atención de las mismas.

Finalmente, la aplicación web será la encargada de facilitar el trabajo a los gestores, permitiendo que puedan administrar las peticiones de forma clara y confiable.

En la siguiente figura (ver figura 2) podemos observar la vista final del dispositivo físico.

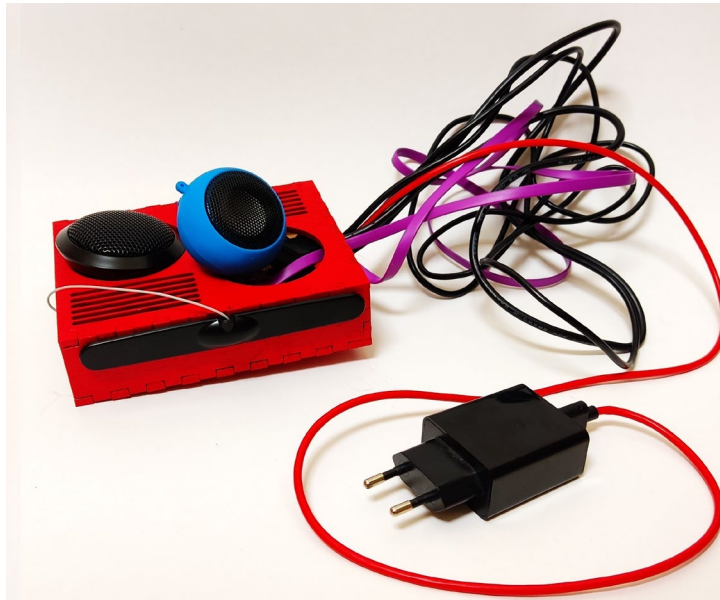


Figura 2: Vista del dispositivo final

2.5. Análisis de viabilidad

El análisis de viabilidad tiene por objetivo garantizar que este sea técnica, económica y legalmente factible, repercutiendo de manera directa en el éxito o fracaso eventual del producto a desarrollar.

En primer lugar, esto está altamente relacionado con los principios de calidad, eficiencia y pertenencia del trabajo fin de grado para el que se realiza. Si durante el desarrollo del producto estos principios son tenidos en cuenta y trabajado basándonos en ellos, se logra reducir el margen de error en la realización de este.

En segundo lugar, el análisis de viabilidad otorga múltiples mejoras en diferentes secciones del trabajo fin de grado, como son la reducción de tiempos innecesarios, aceleración del proceso de producción, reducción de sobrecostes, etc.

Destacan en el análisis de viabilidad [29] los siguientes criterios:

1. **Viabilidad técnica:** Este criterio trata de clarificar si los recursos actuales de carácter técnico son adecuados para el desarrollo del trabajo fin de grado de manera correcta.
2. **Viabilidad económica:** Este criterio trata de clarificar si es posible el desarrollo del trabajo fin de grado basándonos en el tiempo y dinero disponibles.
3. **Viabilidad operacional:** Este criterio trata de clarificar la disponibilidad de operarios encargados al desarrollo del trabajo fin de grado.

El coste de viabilidad de este trabajo fin de grado se basa en los distintos elementos que lo conforman, como son los componentes *hardware* que dan lugar al punto de asistencia, la oferta y demanda de los dispositivos de asistencia y los recursos que son necesarios para poder desarrollarse de manera adecuada [30].

Una vez obtenida una primera visión teórica del trabajo fin de grado, lo siguiente a realizar es comprobar si el producto que se desarrolla durante el trabajo fin de grado es realizable basándonos en la solución final obtenida, es decir, se debe comprobar que el producto sea viable tanto técnica como económicamente. En lo que respecta a la viabilidad operacional depende de contextos como el gubernamental u organizacional, siendo este tratado en apartados posteriores.

2.5.1. El producto y servicio ofertado

El estudio de la factibilidad del trabajo fin de grado incluye varios elementos como son los objetivos, el alcance, las restricciones a las que se enfrenta y el modelo lógico y sofisticado que supone el sistema actual. Esta factibilidad se clasifica en varios tipos al igual que ocurría anteriormente con la viabilidad:

1. **Factibilidad técnica:** Se basa en determinar si la tecnología planteada es la adecuada para el desarrollo del trabajo fin de grado.
2. **Factibilidad económica:** Se basa en analizar la relación que existe entre el coste y el beneficio que se da en el trabajo fin de grado.
3. **Factibilidad operacional:** Se basa en determinar si el trabajo fin de grado puede funcionar en la situación a la que se destine.

Para comenzar se debe cuestionar qué aporta este trabajo fin de grado con respecto a lo que ya se oferta en el mercado actual. En este caso, el dispositivo a comercializar se trata de un punto de asistencia inteligente basado en reconocimiento de patrones de voz que emplea tecnología LoRaWAN para brindar dicha asistencia.

La manera de dar asistencia del dispositivo consiste en que el usuario realice una petición al micrófono del dispositivo, este la procese y responda al usuario. Una vez la petición haya sido procesada será enviada al servidor mediante la conexión LoRaWAN, una vez llegue al servidor los servicios encargados de la gestión de peticiones darán solución a la misma.

Una de las ventajas competitivas que ofrece este servicio es la continua posibilidad de mejora mediante el reentrenamiento de la red neuronal subyacente que permite al dispositivo ofertar un trato más cercano al usuario. Además, al no depender de la conexión a Internet a la hora de enviar las peticiones, esto puede aumentar el alcance del producto, pues podría dar asistencia a usuarios que no cuenten con acceso a Internet.

Otra de las ventajas que presenta es la posibilidad de adecuación a distintas situaciones mediante la completa modularidad que oferta tanto por sus componentes *hardware* como *software*, dotando al punto de asistencia inteligente de una gran flexibilidad frente a distintos entornos.

Finalmente cabe destacar que los requerimientos técnicos del trabajo fin de grado son plenamente abarcables ya que la tecnología subyacente empleada como es LoRaWAN en el caso del dispositivo físico de asistencia o Internet en el caso del servidor son de aplicación cotidiana y lo suficientemente diversificadas como para que puedan desarrollarse plenamente.

2.5.2. La eficiencia energética

En lo que respecta a la eficiencia energética, la Raspberry Pi 4 Modelo B presenta un consumo de 0,6 A (Amperios) en reposo y 1,25 A trabajando a máxima potencia [31] con un voltaje de 5 V (voltios) presenta un consumo de 6,25 W (Vatios), mientras que el Arduino MKR WAN 1310 presenta un consumo de 0,104 mA [32] con un voltaje de 3,3 V presenta un consumo de 0,34 mW, es decir en conjunto el dispositivo consume **6,25034 W** en su pico de trabajo.

Esto se traduce, si se empleara el dispositivo de asistencia las 24 horas del día, en un consumo de **150,0082 W al día** y esto traducido en un año serían **54.753,0064 W al año** o lo que es lo mismo **54,75 kWh** (kilovatios) al año.

Tomando como ejemplo el día de hoy, el kWh en su momento de mayor coste en España costaría 0,14845 € el kWh [33] con lo que el coste energético del punto de asistencia sería de **8,12 € al año**.

2.5.3. Coste de los elementos *hardware* que componen el punto de asistencia

El siguiente cuadro tiene por objetivo proporcionar una idea aproximada del coste de implementación del trabajo fin de grado (ver cuadro 1).

Cuadro 1: Desglose de elementos *hardware* necesarios para el punto de asistencia inteligente

Elemento	Cantidad	Precio (en Euros)	Total
Micrófono de Condensador Omnidireccional de 360°	1	15.80	15.80
USB C Cargador con Interruptor para Raspberry Pi 4 Model B	1	15.88	15.88
Fonestar 3360 Negro altavoz (1.0 canales, Alámbrico, 3,5 mm, 8ohm)	1	15.00	15.00
Raspberry Placa Base PI 4 Modelo B / 8GB	1	264.00	264.00
Arduino MKR WAN 1310	1	39.60	39.60
Dispositivo de asistencia físico	1		350.28
Alquiler de servidor para la aplicación web	1	Precio anual	149.00
Punto de asistencia inteligente	1		499.28

2.5.4. Resultados y recomendaciones

Tras los anteriores puntos mencionados y con la financiación adecuada por parte del organismo interesado en la utilización de los puntos de asistencia inteligentes, el trabajo fin de grado podría ser realizado y puesto a punto en función de las necesidades del cliente, dando como resultado un **trabajo fin de grado viable de ser desarrollado**.

Es recomendable el estudio del caso de uso del trabajo fin de grado previo a ser instalado, definiendo una serie de palabras clave que permitan el desarrollo lo más adaptado al cliente posible, garantizando así una mejor experiencia de usuario y un funcionamiento óptimo.

Finalmente, en lo respectivo a la viabilidad económica, el producto es adaptable y sería posible, aunque no actualmente, a ser modificado basándonos en distintos presupuestos. Dicho presupuesto será comprobable en el siguiente capítulo.

3. Planificación del trabajo fin de grado

En este capítulo del trabajo fin de grado, una vez recogida la idea y estudiada su viabilidad, según ha quedado reflejado en el capítulo anterior 2, es el momento de comenzar a planificar el trabajo que se va a realizar, para ello se va a dividir el trabajo restante en una serie de "paquetes de trabajo"(EDT) [34]. Además, se explicará el uso de la metodología iterativa incremental en este trabajo fin de grado, la organización temporal que requerirá el trabajo fin de grado y el presupuesto de comercialización.

El objetivo de los paquetes de trabajo es facilitar la planificación teórica temporal necesaria para completar satisfactoriamente esos paquetes. Esta planificación teórica basada en un número de horas concreto tiene por objetivo delimitar el tiempo empleado en cada paquete, permitiendo un desarrollo uniforme y regulado de cada uno.

Finalmente, en el último apartado se mostrará un presupuesto aproximado en el caso de que un cliente quisiera disponer de un punto de asistencia inteligente, mostrando en el cuadro los gastos del sistema, gastos adicionales y mantenimiento, de manera que se daría por terminada la visión económica del trabajo fin de grado.

3.1. Metodología iterativa incremental

El primer paso a tener en cuenta es haber definido la solución final, tal y como se realizó en el capítulo anterior, una vez definida es momento de definir un contexto temporal que nos permita llegar a materializar dicha solución teórica.

En este caso la metodología empleada es la iterativa incremental [35], siendo su funcionamiento muy simple, el primer paso es definir una unidad mínima y sobre ella ir añadiendo funcionalidades, esto encaja perfectamente con la división en paquetes de trabajo, puesto que para alcanzar la idea final primero iremos realizando paso a paso las distintas secciones que la componen hasta dar finalmente con el producto completo.

Los paquetes de trabajo tienen una serie de procesos asociados a cada uno para que a la vez que se implementan cumplan con las políticas de calidad del producto.

Los procesos que se realizan en cada paquete son:

- **Análisis:** Se realiza una estimación de la tarea a realizar, en qué consiste y su funcionalidad. Permitiendo, una vez realizado de forma previa a los siguientes procesos, que se puedan realizar de manera iterativa e incremental.
- **Diseño:** Se planifica la tarea, ya sea mediante su implementación teórica o funcionalidad.

- **Implementación:** Se materializa la tarea y sus necesidades específicas.
- **Verificación:** Se pone a prueba y verifica su funcionalidad con respecto al resto de los elementos del sistema.

Cabe destacar la importancia del cliente en todos los procesos previamente descritos, puesto que al estar implicado en el desarrollo del producto, este a su vez determinará la manera de actuar del encargado del paquete, ya sea modificando el diseño previo, la implementación, etc.

Una ventaja clave de este tipo de manera de afrontar el trabajo es que permite que la personalización del punto de asistencia sea máxima, logrando que el cliente al estar presente durante el proceso de realización pueda obtener un resultado acorde a sus ideas y, por lo tanto, la experiencia de usuario se garantice y a su vez reduciendo los tiempos innecesarios en modificaciones débilmente especificadas.

3.2. División en paquetes de trabajo

Una vez definido el entorno de trabajo y la metodología, se han de definir los distintos paquetes de trabajo que han de ser realizados de manera que el punto de asistencia sea implementado con éxito.

Los paquetes de trabajo pueden ser visualizados en el siguiente cuadro (ver cuadro 2).

Cuadro 2: Distribución de los paquetes de trabajo

Paquete de trabajo	Tareas a realizar	Apartado
Preparación Raspberry Pi	Implementación de voz a texto Implementación de la red neuronal Síntesis de la petición del usuario Implementación de texto a voz	Dispositivo Físico
Preparación Arduino MKR 1310	Obtención de identificación de dispositivo Implementación del envío por LoRaWAN	Dispositivo Físico
Preparación de The Things Network	Creación de la aplicación de recepción Preparación de la conexión con Node-red	Recepción de peticiones
Preparación de la aplicación web	Diseño de la aplicación web Implementación de la aplicación web Testeo de la aplicación web	Administración de Peticiones
Preparación del servidor	Implementación de Node-red Lanzamiento de Aplicación web	Administración de Peticiones
Empleo de la aplicación web	Pruebas de uso con usuarios y peticiones reales	Atención de Peticiones
Resolución de imprevistos y actualización documental	Modificación de elementos erróneos y actualización de la documentación	Corección de errores

3.3. Organización temporal del trabajo

Tras haber identificado los paquetes de trabajo y haber definido las tareas a realizar en cada uno, es necesaria la asignación temporal de cada paquete para poder llegar al objetivo a implementar, el punto de asistencia inteligente, en plazo.

El siguiente cuadro mostrará la asignación temporal teórica, esto ayudará a la administración temporal del trabajo necesario para cumplir con la tarea programada (ver cuadro 3).

Cuadro 3: División temporal de las tareas a desarrollar

	septiembre	octubre	noviembre	diciembre	enero	febrero	marzo	abril	mayo
Preparación Raspberry Pi									
Preparación Arduino MKR 1310									
Preparación de The Things Network									
Preparación de la aplicación web									
Preparación del servidor									
Empleo de la aplicación web									
Resolución de imprevistos y actualización documental									

El trabajo fin de grado tiene una duración estimada de 9 meses y tal y como se puede observar en el cuadro anterior, se planea dejar los meses de abril y mayo para solventar inconvenientes no contemplados en la realización inicial teórica de la gestión temporal. La finalidad de esta decisión es la de tener un margen de previsión para, en caso de inconvenientes, tener capacidad de reacción, en caso contrario este espacio temporal puede emplearse para otras tareas como la revisión en profundidad de la documentación elaborada durante la realización del trabajo fin de grado.

Realizada la separación basándonos en los plazos de desarrollo se ha de realizar la asignación temporal horaria a cada paquete, esta queda reflejada en el cuadro de la página siguiente (ver cuadro 4).

Cuadro 4: Asignación horaria teórica de los paquetes de trabajo

Nombre del paquete	Asignación horaria
Preparación Raspberry Pi	45
Preparación Arduino MKR 1310	30
Preparación de The Things Network	15
Preparación de la aplicación web	45
Preparación del servidor	30
Empleo de la aplicación web	45
Documentación (Desde inicio hasta fin del trabajo fin de grado)	70
Resolución de imprevistos y actualización documental	20
Total	300 horas

3.4. Presupuesto de comercialización del trabajo

De acuerdo con lo mencionado al comienzo del capítulo, y tras haber revisado todos los puntos necesarios previos al diseño del sistema, es necesario presentar el apartado del presupuesto de comercialización del trabajo fin de grado, dando por cerrado el análisis económico de viabilidad del trabajo fin de grado y aportando información de coste de cara a una posible publicación en el mercado.

Este presupuesto incluye los costes del sistema, así como los costes adicionales necesarios para su desarrollo e instalación, tal y como se puede observar en el cuadro de la página siguiente (ver cuadro 5), es importante destacar que algunos de los valores mostrados en dicho cuadro han sido descritos en profundidad en el cuadro 1 (ver cuadro 1).

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Cuadro 5: Presupuesto de comercialización del producto final

Elemento	Cantidad	Precio (en Euros)	Total
Costes fijos			
Punto de asistencia inteligente	1	500	500
Instalación del punto de asistencia inteligente	1	50	50
Mantenimiento del punto de asistencia inteligente	1	400	400
Costes variables			
Preparación del punto de asistencia inteligente	1	Variable	Variable
Asistencia técnica	Según cliente	Variable	Variable
Información adicional			
* El coste de contratación de los gestores y equipos necesarios (smartphones y ordenadores) no se incluye en el presupuesto y dependerán del cliente.			
* La preparación del punto de asistencia inteligente tiene un coste variable debido a varios factores que se han de tener en cuenta a la hora de la implementación como la existencia de puntos de conexión a LoRaWAN cercanos, adaptación del sistema a las necesidades del cliente, etc.			
* El operario instalador se compromete a explicar a los gestores cómo trabajar con el sistema así como resolver dudas que se le planteen únicamente durante la instalación.			
* Los costes variables serán notificados y detallados al cliente antes del pago.			
* Los costes variables se calculan atendiendo a la siguiente fórmula: 0,19 € por km [36] + 105 € por noche de hospedaje [37] + 14,34 € por hora de trabajo [38].			
Producto final	1	950	950 + Variables

Tal y como se detalla en el cuadro anterior y a fin de solventar problemas a consecuencia de la subcontratación de personal especializado en la atención de este tipo de necesidades, los gestores, se le otorga al cliente la libertad de contratar al personal que desee. Esta consideración se toma debido a modificaciones que pueda suponer las leyes españolas vigentes sobre la financiación de subcontratas.

Cabe destacar que la asistencia técnica dependerá del cliente y, por lo tanto, no se puede estandarizar en el presupuesto.

4. Diseño del punto de asistencia

Este capítulo tiene por objetivo presentar el diseño del punto de asistencia inteligente, siendo este el paso previo necesario para poder llevar a cabo de manera exitosa la implementación del trabajo fin de grado.

La función del apartado de diseño es la de definir una manera de trabajo específica que marque el itinerario a seguir para poder desempeñar la implementación, haciendo uso de modelos teóricos en el proceso.

En los siguientes apartados se detallarán estos modelos teóricos previos a la implementación del punto de asistencia. Estos son necesarios para evitar que ocurran errores durante la implementación, ahorrando en el proceso tiempo y dinero, además permiten al cliente tener una muestra tangible del producto y redefinir apartados que en un futuro puedan generarle disconformidad.

4.1. Diagramas de casos de uso

En el sistema propuesto se especifican distintos tipos de usuario, como son los usuarios que requieren asistencia, es decir, aquellos que utilizan el dispositivo físico del punto de asistencia, los usuarios del sistema de gestión, que son aquellos contratados por el cliente para controlar y atender las peticiones que realizan los usuarios anteriormente citados y los usuarios administradores capaces de gestionar el sistema que emplearán los gestores.

Por tanto, es necesario especificar las acciones que van a poder realizar estos usuarios, para establecer una hoja de ruta necesaria para la futura implementación.

Es por ello que se emplearán diagramas de casos de uso para este cometido, a fin de poder representar de forma clara las distintas acciones posibles de estos usuarios [39]. Para ello en los siguientes cuadros se visualiza esta información (ver cuadro 6), (ver cuadro 7), (ver cuadro 8), (ver cuadro 9) y (ver cuadro 10)

Nota: A modo de facilitar la comprensión al lector, se ha adaptado el formato clásico de casos de uso a un formato de tabla.

Cuadro 6: Diagrama 1 de casos de uso

CASO DE USO 1: EL USUARIO REALIZA UNA PETICIÓN	
Caso de uso	El usuario realiza una petición al dispositivo físico
Condiciones Previas	El dispositivo físico ha de estar correctamente instalado. El servicio receptor ha de estar disponible.
Condiciones Posteriores	El servicio receptor ha de tener registrado la identificación del dispositivo físico.
Objetivo	La petición es enviada desde el dispositivo físico al sistema
Consideraciones	La petición se ha de poder enviar cada 2 minutos. El dispositivo físico ha de poder contar con alguna forma de finalización de la interacción persona-dispositivo.

Cuadro 7: Diagrama 2 de casos de uso

CASO DE USO 2: EL USUARIO GESTOR VISUALIZA LAS PETICIONES	
Caso de uso	El usuario gestor visualiza la lista de peticiones enviadas por el servicio receptor disponibles para ser atendidas.
Condiciones Previas	Una petición ha de haber sido enviada. El servicio Node-red ha de estar disponible. La base de datos de la aplicación web ha de estar disponible. La aplicación web ha de estar disponible.
Condiciones Posteriores	El servicio Node-red ha de estar vinculado al servicio receptor. El servicio Node-red ha de estar vinculado a la base de datos. El servicio Node-red ha de estar vinculado al bot de Telegram. El usuario gestor ha de estar registrado en la base de datos.
Objetivo	El usuario gestor puede visualizar la lista de peticiones para poder gestionarlas.
Consideraciones	El usuario gestor ha de poder visualizar las peticiones. El usuario gestor ha de poder trabajar en las peticiones. El usuario gestor ha de poder acceder al sistema y salir del mismo.

Una vez se han detallado los casos de uso del sistema es conveniente establecer un orden temporal en la utilización de este, puesto que el sistema así lo requiere, en caso contrario podrían darse sucesos que ocasionarían errores o funcionamientos deficientes del sistema como la inexistencia de un usuario gestor en la base de datos que quisiera abrir una sesión.

Para ello se hará uso de los diagramas de secuencia, siendo estos una solución de modelado

Cuadro 8: Diagrama 3 de casos de uso

CASO DE USO 3: EL USUARIO GESTOR ADMINISTRA LAS PETICIONES	
Caso de uso	El usuario gestor administra las peticiones de la lista de peticiones disponibles para ser atendidas.
Condiciones Previas	Una petición ha de haber sido enviada. El servicio Node-red ha de estar disponible. La base de datos de la aplicación web ha de estar disponible. La aplicación web ha de estar disponible.
Condiciones Posteriores	El servicio Node-red ha de estar vinculado al servicio receptor. El servicio Node-red ha de estar vinculado a la base de datos. El servicio Node-red ha de estar vinculado al bot de Telegram. El usuario gestor ha de estar registrado en la base de datos.
Objetivo	El usuario gestor puede administrar las peticiones de la lista de peticiones pudiendo marcarlas como en proceso o revocándolas.
Consideraciones	El usuario gestor ha de poder marcar las peticiones como en proceso. El usuario gestor ha de poder revocar las peticiones que ha marcado. Una petición revocada vuelve a la lista de peticiones disponibles. El usuario gestor ha de poder acceder al sistema y salir del mismo.

Cuadro 9: Diagrama 4 de casos de uso

CASO DE USO 4: EL USUARIO GESTOR FINALIZA LAS PETICIONES	
Caso de uso	El usuario gestor finaliza las peticiones de la lista de peticiones que previamente había marcado como en proceso.
Condiciones Previas	Una petición ha de haber sido marcada como en proceso. El servicio Node-red ha de estar disponible. La base de datos de la aplicación web ha de estar disponible. La aplicación web ha de estar disponible.
Condiciones Posteriores	El servicio Node-red ha de estar vinculado al servicio receptor. El servicio Node-red ha de estar vinculado a la base de datos. El servicio Node-red ha de estar vinculado al bot de Telegram. El usuario gestor ha de estar registrado en la base de datos.
Objetivo	El usuario gestor puede finalizar una petición que marca como en proceso para indicar que la ha atendido.
Consideraciones	El usuario gestor ha de poder finalizar las peticiones en proceso. El usuario gestor ha de visualizar las peticiones finalizadas. El usuario gestor ha de poder acceder al sistema y salir del mismo.

Cuadro 10: Diagrama 5 de casos de uso

CASO DE USO 5: EL USUARIO ADMINISTRADOR GESTIONA EL SISTEMA	
Caso de uso	El usuario administrador gestiona el sistema para su correcto funcionamiento.
Condiciones Previas	El servicio receptor ha de estar disponible. El servicio Node-red ha de estar disponible. La base de datos de la aplicación web ha de estar disponible. La aplicación web ha de estar disponible.
Condiciones Posteriores	El servicio Node-red ha de estar vinculado al servicio receptor. El servicio Node-red ha de estar vinculado a la base de datos. El servicio Node-red ha de estar vinculado al bot de Telegram. El usuario administrador ha de estar registrado en la base de datos.
Objetivo	El usuario administrador ha de poder tener control total del sistema.
Consideraciones	El usuario administrador ha de poder visualizar, añadir, eliminar y modificar las peticiones. El usuario administrador ha de poder visualizar, añadir, eliminar y modificar a los usuarios gestores. El usuario administrador ha de tener total acceso a la base de datos y ser el único con esta capacidad. El usuario administrador ha de poder acceder al sistema y salir del mismo.

dinámico popular en UML o Lenguaje unificado de modelado, con el fin de especificar la coexistencia de los objetos y procesos que a su vez coexisten junto a la información intercambiada entre ellos [40].

4.2. Diagramas de secuencia

El objeto de los diagramas de secuencia, tal y como se ha explicado con anterioridad, es el de mostrar una organización temporal de cómo se ha interaccionar con el sistema. Por ello se van a detallar los siguientes subapartados que tratarán de sintetizar la utilización del sistema:

4.2.1. Realización de una petición

La realización de peticiones es una parte fundamental del trabajo fin de grado, pues es la manera que tienen los usuarios necesitados de asistencia de comunicar esta necesidad a los gestores encargados de resolverlas. Por ello es de vital importancia que su recepción sea adecuada y efectiva.

En primer lugar, para que la petición pueda ser recogida en el servicio receptor, es necesario que dicho servicio receptor sea configurado por el administrador del sistema y para ello, ha de tener la información de identificación del dispositivo físico previamente.

En segundo lugar, el dispositivo físico ha de estar adaptado y configurado para poder atender las peticiones del usuario de manera adecuada.

Tras haber completado estos pasos previos, la petición llegará de forma correcta al servicio receptor. Tal y como se muestra en la siguiente figura (ver figura 3).

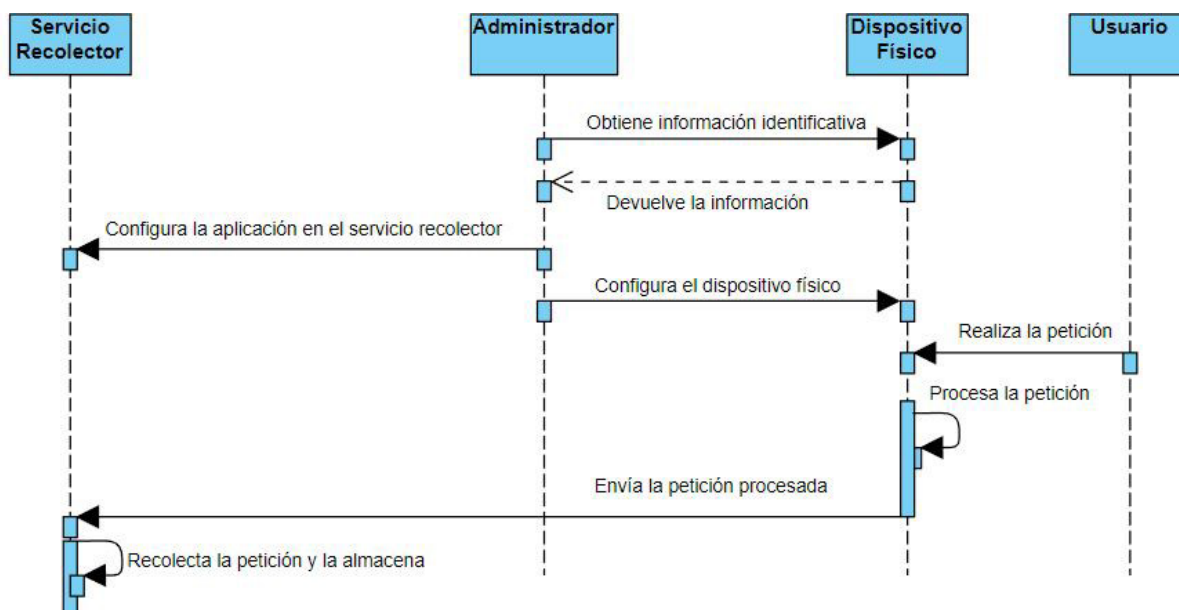


Figura 3: Diagrama de secuencia de la realización de peticiones

4.2.2. Recepción de una petición del servicio de recolección

En primer lugar, el administrador ha de configurar el canal de Telegram y bot asociado para su posterior uso a través de Node-red.

Posteriormente, de nuevo el administrador ha de configurar los servicios de Node-red previamente, y posteriormente su conexión con el servicio receptor, para poder ser transmitidos

a los servicios pertinentes una vez las peticiones son recolectadas por este.

Tras haber obtenido una petición en el servicio receptor, los servicios de Node-red se encargarán tanto de la carga en la base de datos como en el servicio notificador de los gestores (Telegram), tal y como se muestra en el siguiente diagrama (ver figura 4), de esta manera el usuario gestor será capaz tanto de conocer la llegada de las nuevas peticiones como de gestionarlas a través de la aplicación web.

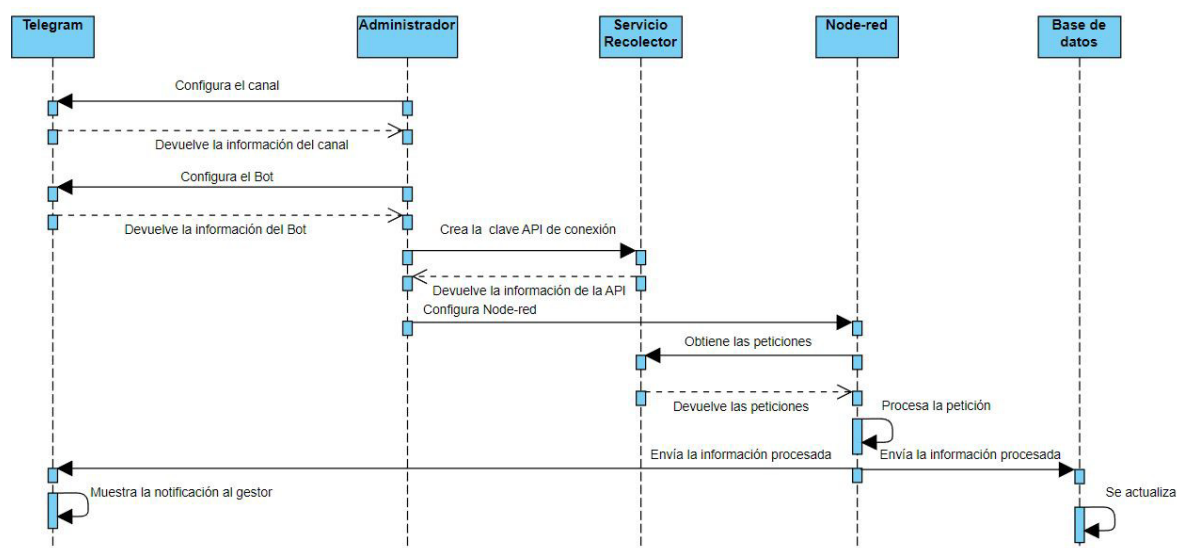


Figura 4: Diagrama de secuencia de la recepción de peticiones

4.2.3. Atención de una petición

En este último apartado la petición ya ha sido emitida, recolectada en el servicio y tanto cargada en la base de datos como notificada a los usuarios gestores. Es por ello que previamente el administrador debe de haber dado de alta a los usuarios gestores en la base de datos.

Una vez realizado esto, el gestor podrá realizar el tratamiento de la petición, es decir, visualizarla, marcarla como en proceso de realización (quedando actualizada la lista de peticiones disponibles) y darla por atendida una vez termine de atender, actualizando así la lista de peticiones terminadas tal y como se puede observar en el siguiente diagrama (ver figura 5).

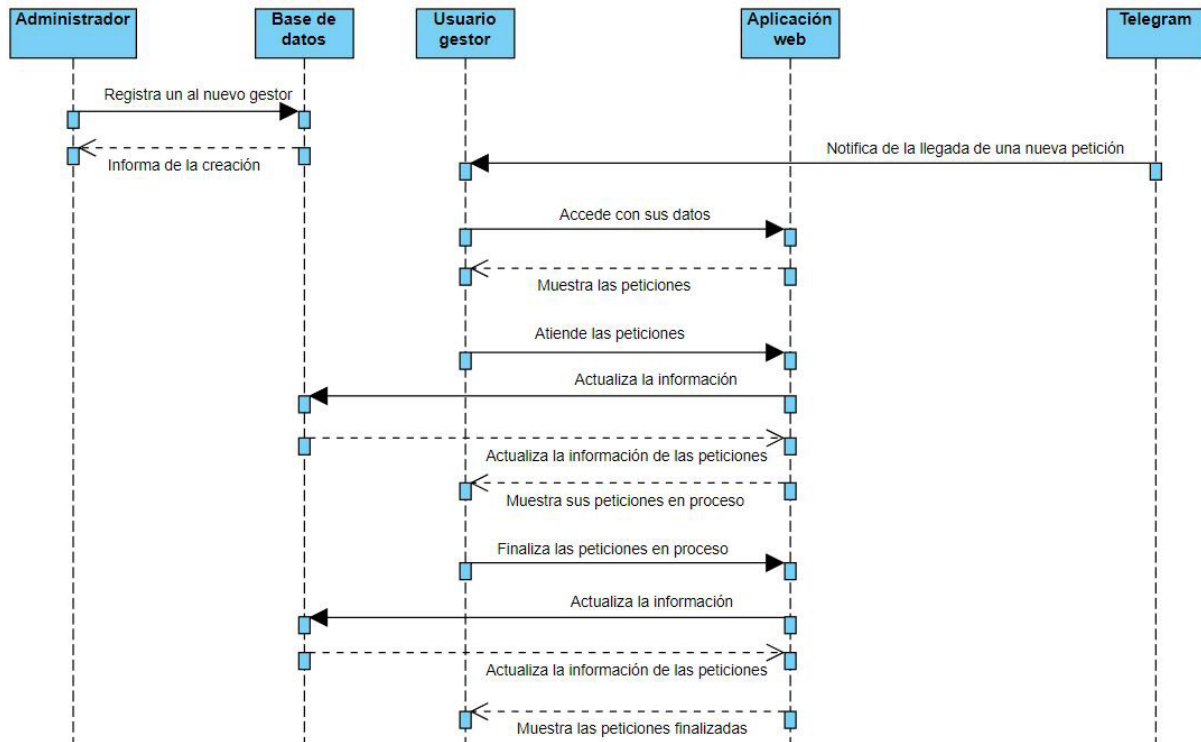


Figura 5: Diagrama de secuencia de la atención de peticiones

4.3. Diseño de la base de datos

Una vez hemos definido los diagramas de casos de uso y secuencia, es necesario diseñar los distintos elementos que serán necesarios para que el punto de asistencia inteligente funcione de forma adecuada.

Uno de estos elementos es la base de datos, encargada de almacenar la información que le envía Node-red y que a su vez proviene del servicio receptor, este punto será de vital importancia para que la aplicación web que utilizarán los usuarios gestores funcione correctamente, en este caso el tipo de base de datos será SQL concretamente bajo el *software* MySQL.

En primer lugar, para definir una base de datos hay que establecer que elementos van a formar parte de ella, en este caso va a ser una base de datos destinada a una aplicación concreta, gestionar las peticiones que ocurren en el Centro Universitario de Mérida, estas peticiones van a tener una serie de características concretas. Los usuarios gestores en este caso serán los bedeles.

Las peticiones a tratar serán de una de las siguientes categorías:

- **Fuego:** La finalidad de esta petición es la de informar de situaciones de fuego o incendio en el aula.
- **Socorro:** La finalidad de esta petición es la de informar de situaciones de necesidad o ayuda en el aula.
- **Mando del proyector:** La finalidad de esta petición es la de solicitar el mando del proyector del aula a los bedeles.
- **Llave de la persiana:** La finalidad de esta petición es la de solicitar las llaves de la persiana del aula a los bedeles.
- **Asistencia del bedel:** La finalidad de esta petición es la de solicitar a los bedeles su asistencia en el aula.
- **Emergencia:** La finalidad de esta petición es la de informar de situaciones de extrema necesidad.

Por lo tanto, los elementos serán los siguientes junto a sus atributos (ver figura 6).

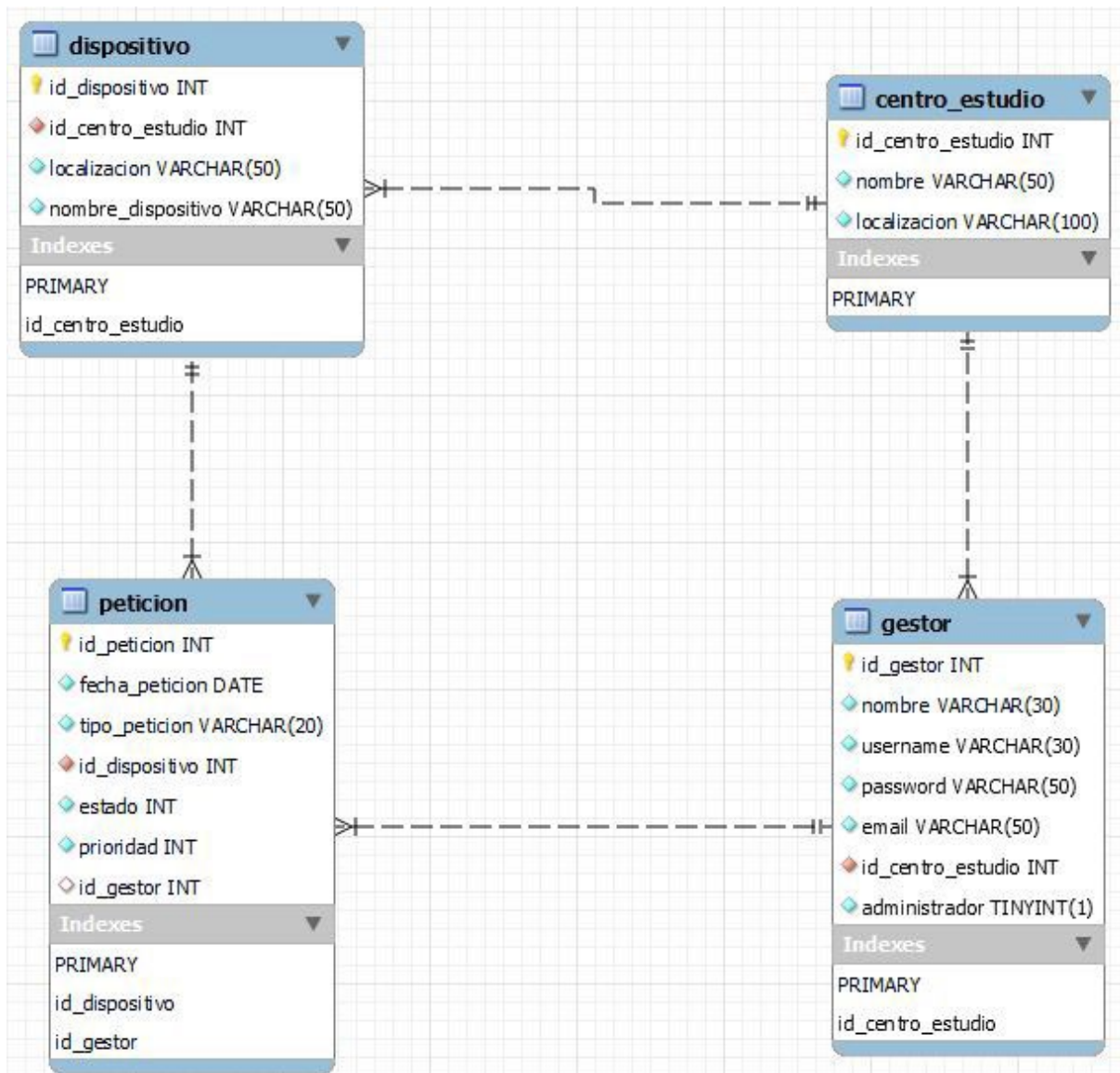


Figura 6: Diseño de la base de datos

La función de estos elementos es la siguiente:

- **Dispositivo:** Almacena la información necesaria del dispositivo, identificado por su nombre e identificado por su Id, tiene una relación con el centro de estudio en el que se aloja.
- **Centro estudio:** Almacena la información necesaria del centro de estudio en el que trabaja el gestor y se aloja el dispositivo.

- **Petición:** La finalidad de clase es almacenar el tipo de petición, la fecha en la que se realizó y su prioridad, siendo esta la máxima para emergencias, alta para fuego y socorro, media para solicitar la asistencia del bedel y baja para el resto.
- **Gestor:** Almacena la información necesaria del trabajador, será necesario el *username* para el acceso al sistema junto con el *password*.

4.4. Diseño de la aplicación web

Tras haber definido el tipo de las peticiones hay que definir que datos van a ser requeridos en la aplicación web, el lenguaje sobre el que se estructurará la aplicación web va a ser de tipo orientado a objetos, concretamente Java, con lo cual serán necesarias unas clases con sus respectivos atributos que darán forma a los objetos [41].

La POO o programación orientada a objetos ha sido elegida debido a su facilidad de implementación en este tipo de aplicaciones web, junto con el resto de herramientas que facilitarán la implementación de la aplicación web como Java Spring Boot.

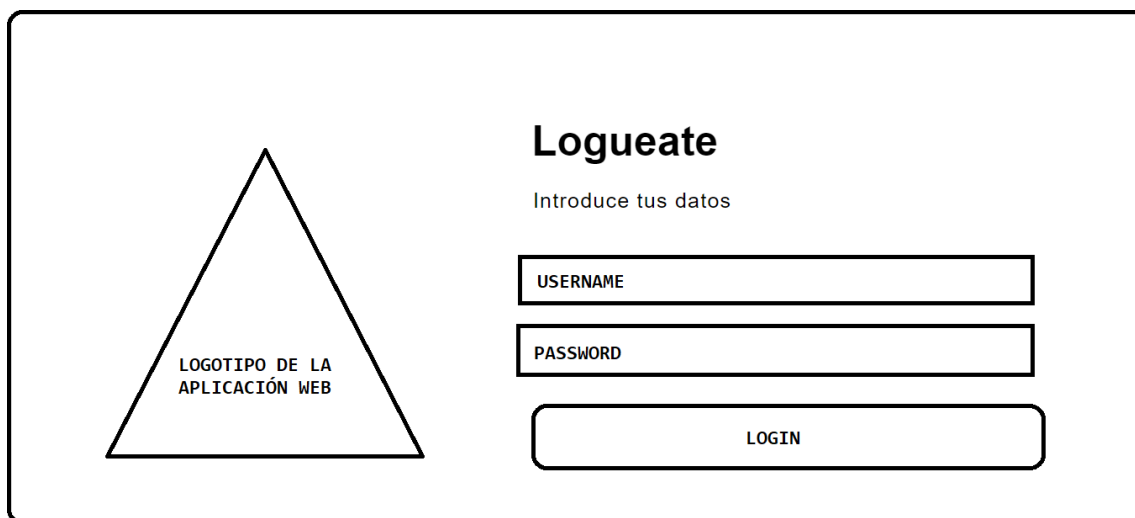
A consecuencia de lo anteriormente comentado, la elección de Java para el desarrollo de la aplicación web es debido a que el framework de desarrollo web Spring Boot, basado en Java, facilitará en gran medida el desarrollo de esta, además de la existencia de un conector entre Java y MySQL [42] que permitirá de forma cómoda la interacción entre ambas.

El primer paso para diseñar la aplicación web será el de definir que funcionalidades ha de tener, según lo recogido en los Diagrama 2 de casos de uso, Diagrama 3 de casos de uso, Diagrama 4 de casos de uso y Diagrama 5 de casos de uso.

Por ello, la aplicación web ha de ser capaz de:

- **Administrar la sesión:** El usuario gestor o administrador ha de poder iniciar y cerrar sesión.
- **Visualizar las peticiones:** Tanto el usuario gestor como el administrador.
- **Administrar las peticiones:** El usuario gestor ha de tener la posibilidad de atender, revocar y finalizar peticiones.
- **Administrar la información:** El usuario administrador ha de tener control total de los contenidos almacenados en la base de datos mediante la web, con lo cual ha de tener 4 secciones destinadas a la visualización, creación, eliminación y modificación de los distintos elementos pertenecientes al dispositivo, centro de estudio, petición y gestor.

Basándonos en las especificaciones previas, el diseño visual de la aplicación web variará en función de si el usuario es un gestor o un administrador. Aunque ambos compartirán la misma vista de inicio de sesión (ver figura 7).

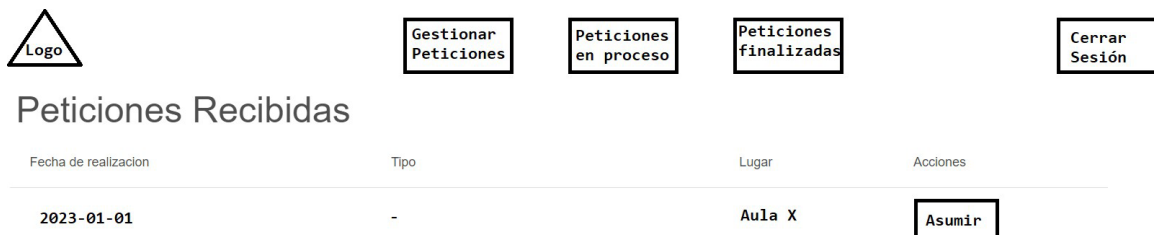


El diseño de la vista de inicio de sesión se muestra dentro de un recuadro con bordes redondeados. A la izquierda hay un triángulo que contiene el texto "LOGOTIPO DE LA APLICACIÓN WEB". A la derecha, el título "Logueate" está centrado, con el subtítulo "Introduce tus datos" debajo. Hay tres campos de entrada de texto: "USERNAME", "PASSWORD" y "LOGIN".

Figura 7: Diseño de la vista de inicio de sesión

4.4.1. Vista del usuario gestor

El usuario gestor tendrá la siguiente vista para la visualización de las peticiones sin atender (ver figura 8).



El diseño de la vista de visualización de peticiones no atendidas incluye un menú de navegación superior con un triángulo "Logo" y cuatro botones: "Gestionar Peticiones", "Peticiones en proceso", "Peticiones finalizadas" y "Cerrar Sesión". Debajo del menú, el título "Peticiones Recibidas" precede a una tabla con las siguientes columnas: "Fecha de realizacion", "Tipo", "Lugar" y "Acciones".

Fecha de realizacion	Tipo	Lugar	Acciones
2023-01-01	-	Aula X	Asumir

Figura 8: Diseño de la vista de visualización de peticiones no atendidas

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Gestionar peticiones:** Le mostrará al usuario gestor la vista reflejada en la figura 8.
- **Peticiones en proceso:** Le mostrará al usuario gestor la vista reflejada en la figura 9.
- **Peticiones finalizadas:** Le mostrará al usuario gestor la vista reflejada en la figura 10.
- **Asumir:** Marcará la petición como asumida por el usuario gestor y le mostrará la vista reflejada en la figura 9.
- **Cerrar sesión:** Finalizará la sesión activa y le mostrará al usuario gestor la vista reflejada en la figura 7.

La siguiente figura mostrará la vista para la atención de peticiones (ver figura 9).



Figura 9: Diseño de la vista de atención de peticiones

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Gestionar peticiones:** Le mostrará al usuario gestor la vista reflejada en la figura 8.
- **Peticiones en proceso:** Le mostrará al usuario gestor la vista reflejada en la figura 9.
- **Peticiones finalizadas:** Le mostrará al usuario gestor la vista reflejada en la figura 10.
- **Terminar:** Marcará la petición como finalizada por el usuario gestor y le mostrará la vista reflejada en la figura 10.
- **Revocar:** Devolverá la petición al estado de disponible y le mostrará al usuario gestor la vista reflejada en la figura 8.
- **Cerrar sesión:** Finalizará la sesión activa y le mostrará al usuario gestor la vista reflejada en la figura 7.

Finalmente la siguiente figura mostrará la vista para la visualización de peticiones terminadas (ver figura 10).



Figura 10: Diseño de la vista de visualización de peticiones finalizadas

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Gestionar peticiones:** Le mostrará al usuario gestor la vista reflejada en la figura 8.
- **Peticiones en proceso:** Le mostrará al usuario gestor la vista reflejada en la figura 9.
- **Peticiones finalizadas:** Le mostrará al usuario gestor la vista reflejada en la figura 10.
- **Cerrar sesión:** Finalizará la sesión activa y le mostrará al usuario gestor la vista reflejada en la figura 7.

La vista de las acciones Asumir, Revocar y Terminar se realizará mediante SweetAlert [43] a modo de reducir el número de vistas a implementar.

4.4.2. Vista del administrador

El usuario administrador tendrá la siguiente vista una vez inicie sesión (ver figura 11).

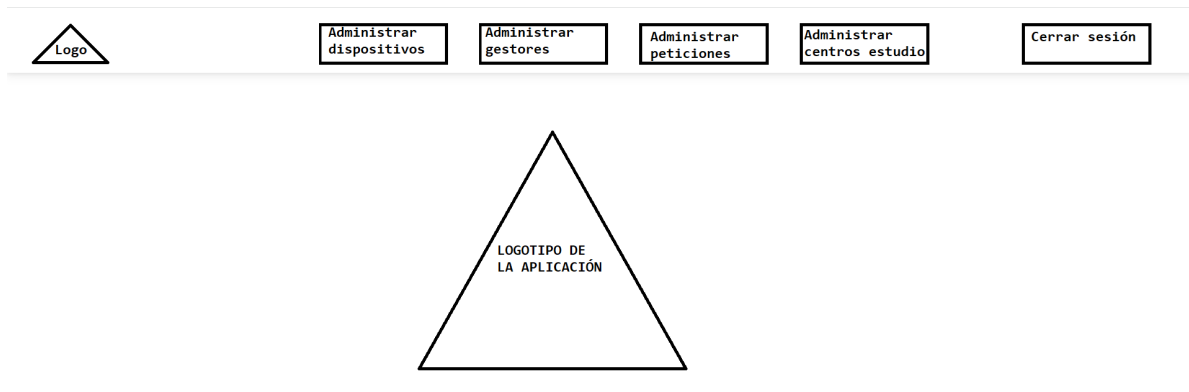


Figura 11: Diseño de la vista de inicio de sesión del administrador

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Administrar dispositivos:** Le mostrará al usuario administrador la vista reflejada en la figura 12.
- **Administrar gestores:** Le mostrará al usuario administrador la vista reflejada en la figura 13.
- **Administrar peticiones:** Le mostrará al usuario administrador la vista reflejada en la figura 14.
- **Administrar centros estudio:** Le mostrará al usuario administrador la vista reflejada en la figura 15.
- **Cerrar sesión:** Finalizará la sesión activa y le mostrará al usuario administrador la vista reflejada en la figura 7.

La siguiente vista mostrará todos los dispositivos y facilitará su administración (ver figura 12).

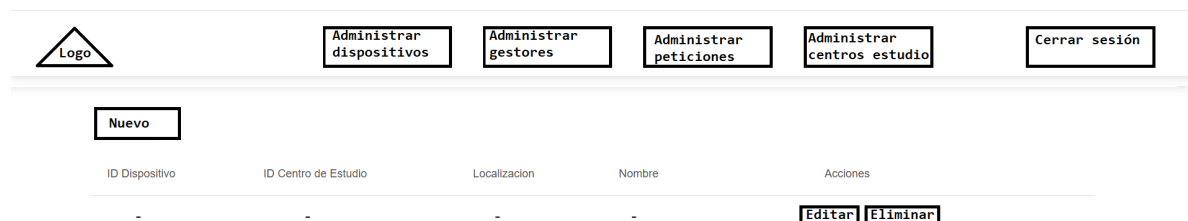


Figura 12: Diseño de la vista de administración de los dispositivos

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Administrar dispositivos:** Le mostrará al usuario administrador la vista reflejada en la figura 12.
- **Administrar gestores:** Le mostrará al usuario administrador la vista reflejada en la figura 13.
- **Administrar peticiones:** Le mostrará al usuario administrador la vista reflejada en la figura 14.
- **Administrar centros estudio:** Le mostrará al usuario administrador la vista reflejada en la figura 15.
- **Cerrar sesión:** Finalizará la sesión activa y le mostrará al usuario administrador la vista reflejada en la figura 7.
- **Nuevo y Editar:** Le mostrará al usuario administrador la vista reflejada en la figura 16. La diferencia es que en editar tomará los datos del dispositivo elegido.
- **Eliminar:** Se le mostrará al usuario administrador una interacción mediante SweetAlert y tras confirmar se eliminará el dispositivo seleccionado.

La siguiente vista mostrará todos los gestores y facilitará su administración (ver figura 13).

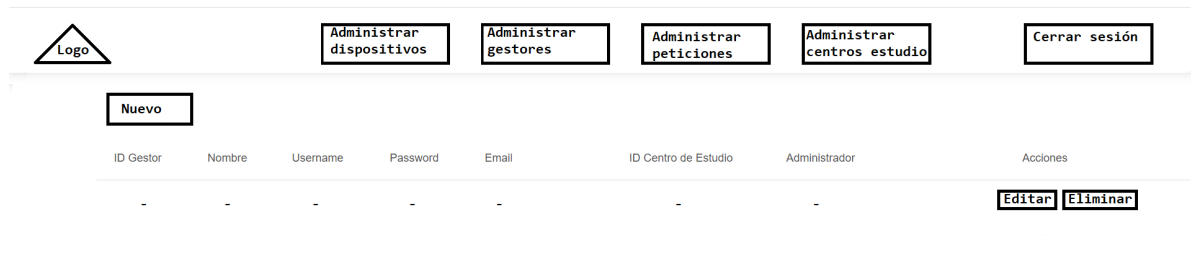


Figura 13: Diseño de la vista de administración de los gestores

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Administrar dispositivos:** Le mostrará al usuario administrador la vista reflejada en la figura 12.
- **Administrar gestores:** Le mostrará al usuario administrador la vista reflejada en la figura 13.
- **Administrar peticiones:** Le mostrará al usuario administrador la vista reflejada en la figura 14.
- **Administrar centros estudio:** Le mostrará al usuario administrador la vista reflejada en la figura 15.
- **Cerrar sesión:** Finalizará la sesión activa y le mostrará al usuario administrador la vista reflejada en la figura 7.
- **Nuevo y Editar:** Le mostrará al usuario administrador la vista reflejada en la figura 17. La diferencia es que en editar tomará los datos del gestor elegido.
- **Eliminar:** Se le mostrará al usuario administrador una interacción mediante SweetAlert y tras confirmar se eliminará el gestor seleccionado.

La siguiente vista mostrará todas las peticiones y facilitará su administración (ver figura 14).



Figura 14: Diseño de la vista de administración de las peticiones

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Administrar dispositivos:** Le mostrará al usuario administrador la vista reflejada en la figura 12.
- **Administrar gestores:** Le mostrará al usuario administrador la vista reflejada en la figura 13.
- **Administrar peticiones:** Le mostrará al usuario administrador la vista reflejada en la figura 14.
- **Administrar centros estudio:** Le mostrará al usuario administrador la vista reflejada en la figura 15.
- **Cerrar sesión:** Finalizará la sesión activa y le mostrará al usuario administrador la vista reflejada en la figura 7.
- **Nuevo y Editar:** Le mostrará al usuario administrador la vista reflejada en la figura 18. La diferencia es que en editar tomará los datos de la petición elegida.
- **Eliminar:** Se le mostrará al usuario administrador una interacción mediante SweetAlert y tras confirmar se eliminará la petición seleccionada.

La siguiente vista mostrará todos los centros de estudio y facilitará su administración (ver figura 15).



Figura 15: Diseño de la vista de administración de los centros de estudio

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Administrar dispositivos:** Le mostrará al usuario administrador la vista reflejada en la figura 12.
- **Administrar gestores:** Le mostrará al usuario administrador la vista reflejada en la figura 13.
- **Administrar peticiones:** Le mostrará al usuario administrador la vista reflejada en la figura 14.
- **Administrar centros estudio:** Le mostrará al usuario administrador la vista reflejada en la figura 15.
- **Cerrar sesión:** Finalizará la sesión activa y le mostrará al usuario administrador la vista reflejada en la figura 7.
- **Nuevo y Editar:** Le mostrará al usuario administrador la vista reflejada en la figura 19. La diferencia es que en editar tomará los datos del centro de estudios elegido.
- **Eliminar:** Se le mostrará al usuario administrador una interacción mediante SweetAlert y tras confirmar se eliminará el centro de estudios seleccionado.

En este caso el usuario administrador sí tendrá una vista compartida para añadir y modificar los elementos, siendo estas:

La siguiente vista para añadir y modificar dispositivos (ver figura 16).

The image shows a web application interface for adding and modifying devices. At the top, there is a navigation bar with a logo on the left and five menu items: 'Administrar dispositivos', 'Administrar gestores', 'Administrar peticiones', 'Administrar centros estudio', and 'Cerrar sesión'. Below the navigation bar is a form titled 'Datos del Nuevo Dispositivo'. The form contains three input fields: 'Selecciona Centro de Estudio' (a dropdown menu), 'Localizacion' (a text input), and 'Nombre del Dispositivo' (a text input). At the bottom of the form are two buttons: 'Guardar' and 'Cancelar'.

Figura 16: Diseño de la vista para añadir y modificar dispositivos

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Administrar dispositivos:** Le mostrará al usuario administrador la vista reflejada en la figura 12.
- **Administrar gestores:** Le mostrará al usuario administrador la vista reflejada en la figura 13.
- **Administrar peticiones:** Le mostrará al usuario administrador la vista reflejada en la figura 14.
- **Administrar centros estudio:** Le mostrará al usuario administrador la vista reflejada en la figura 15.
- **Seleccionar Centro de Estudio:** Le mostrará al usuario administrador un menú desplegable para elegir el centro de estudio asociado al dispositivo.
- **Guardar:** Realizará un guardado de dicho dispositivo en la base de datos o modificará el dispositivo con los nuevos datos.
- **Cancelar:** Devolverá al usuario administrador a la vista reflejada en la figura 12.

La siguiente vista muestra las funcionalidades de añadir y modificar gestores (ver figura 17).

Logo

Administrar dispositivos

Administrar gestores

Administrar peticiones

Administrar centros estudio

Cerrar sesión

Datos del Nuevo Gestor

Nombre

Username

Password

Email

Selecciona Centro de Estudio

Es administrador?

Guardar

Cancelar

Figura 17: Diseño de la vista para añadir y modificar gestores

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Administrar dispositivos:** Le mostrará al usuario administrador la vista reflejada en la figura 12.
- **Administrar gestores:** Le mostrará al usuario administrador la vista reflejada en la figura 13.
- **Administrar peticiones:** Le mostrará al usuario administrador la vista reflejada en la figura 14.
- **Administrar centros estudio:** Le mostrará al usuario administrador la vista reflejada en la figura 15.
- **Seleccionar Centro de Estudio:** Le mostrará al usuario administrador un menú desplegable para elegir el centro de estudio asociado al gestor.
- **Es administrador?:** Le mostrará al usuario administrador un menú desplegable para elegir si el usuario gestor es o no administrador.
- **Guardar:** Realizará un guardado de dicho gestor en la base de datos o modificará el gestor con los nuevos datos.
- **Cancelar:** Devolverá al usuario administrador a la vista reflejada en la figura 13.

La siguiente vista muestra las funcionalidades de añadir y modificar peticiones (ver figura 18).

The image shows a web application interface for managing requests. At the top, there is a navigation bar with a logo on the left and five buttons: 'Administrar dispositivos', 'Administrar gestores', 'Administrar peticiones', 'Administrar centros estudio', and 'Cerrar sesión'. Below the navigation bar is a form titled 'Datos de la Nueva Peticion'. The form contains four input fields: 'Tipo de Peticion', 'Selecciona Dispositivo' (with a dropdown arrow), 'Estado de la Peticion', and 'Prioridad de la Peticion'. At the bottom of the form are two buttons: 'Guardar' and 'Cancelar'.

Figura 18: Diseño de la vista para añadir y modificar peticiones

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Administrar dispositivos:** Le mostrará al usuario administrador la vista reflejada en la figura 12.
- **Administrar gestores:** Le mostrará al usuario administrador la vista reflejada en la figura 13.
- **Administrar peticiones:** Le mostrará al usuario administrador la vista reflejada en la figura 14.
- **Administrar centros estudio:** Le mostrará al usuario administrador la vista reflejada en la figura 15.
- **Seleccionar Dispositivo:** Le mostrará al usuario administrador un menú desplegable para elegir el dispositivo asociado a la petición.
- **Guardar:** Realizará un guardado de dicha petición en la base de datos o modificará la petición con los nuevos datos.
- **Cancelar:** Devolverá al usuario administrador a la vista reflejada en la figura 14.

Finalmente, la siguiente vista muestra las funcionalidades de añadir y modificar centros de estudio (ver figura 19).

The image shows a web application interface for managing study centers. At the top, there is a navigation bar with a logo on the left and five menu items: 'Administrar dispositivos', 'Administrar gestores', 'Administrar peticiones', 'Administrar centros estudio', and 'Cerrar sesión'. Below the navigation bar, a modal form titled 'Datos del Nuevo Centro Estudio' is displayed. The form contains two text input fields: 'Nombre' and 'Localizacion'. At the bottom of the form, there are two buttons: 'Guardar' and 'Cancelar'.

Figura 19: Diseño de la vista para añadir y modificar centros de estudio

Los elementos mostrados en la anterior figura cumplen con las siguientes funciones:

- **Administrar dispositivos:** Le mostrará al usuario administrador la vista reflejada en la figura 12.
- **Administrar gestores:** Le mostrará al usuario administrador la vista reflejada en la figura 13.
- **Administrar peticiones:** Le mostrará al usuario administrador la vista reflejada en la figura 14.
- **Administrar centros estudio:** Le mostrará al usuario administrador la vista reflejada en la figura 15.
- **Guardar:** Realizará un guardado de dicho centro de estudios en la base de datos o modificará el centro de estudios con los nuevos datos.
- **Cancelar:** Devolverá al usuario administrador a la vista reflejada en la figura 15.

4.5. Diseño del dispositivo físico

En este apartado se va a desarrollar el diseño del dispositivo físico, para ello el diseño se subdivide en los subapartados del diseño del componente físico y el diseño del componente *hardware* en el que se detallará manera de interconectar dichos elementos.

4.5.1. Diseño del componente físico

En lo respectivo al componente físico, se va a emplear una estructura externa o carcasa, que tendrá como función proteger en la medida de lo posible los componentes *hardware* del dispositivo, a su vez facilitará el transporte del mismo y facilitará su distribución.

El diseño de la carcasa se realizará empleando la herramienta online Boxes.py [44] y posteriormente se modificará con la herramienta AutoCAD [45], una herramienta de modelado de planos y modificación de estos.

El diseño resultante puede observarse en la siguiente figura (ver figura 20).

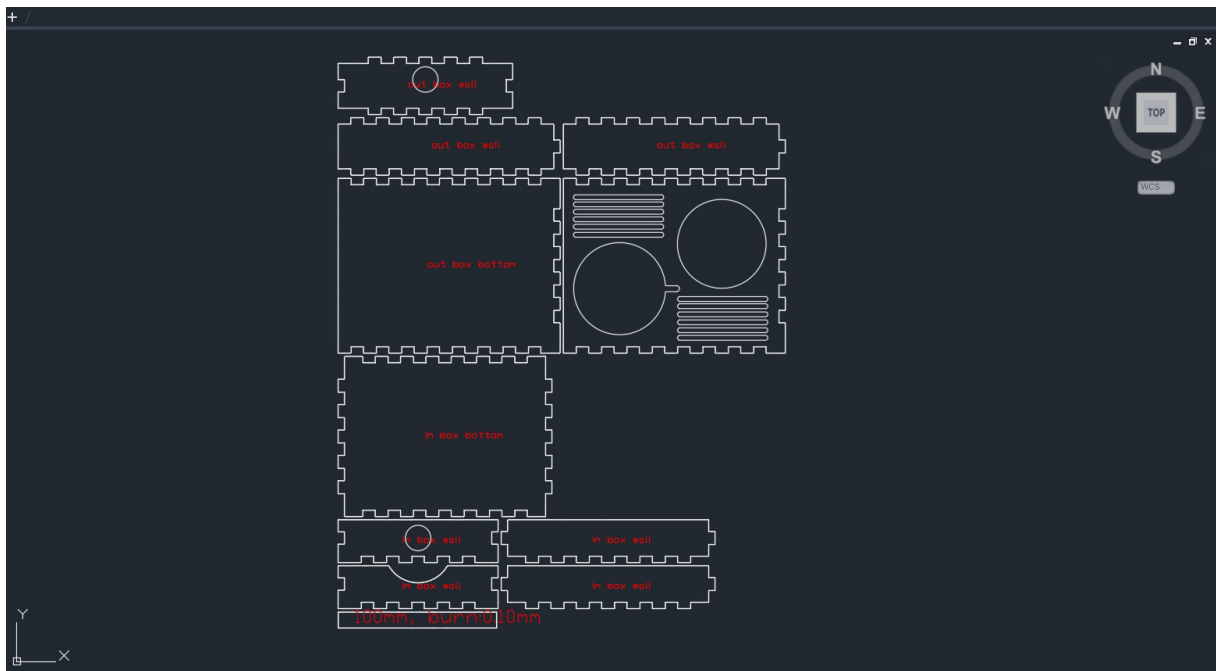


Figura 20: Diseño de la carcasa visto desde AutoCAD

4.5.2. Diseño del componente *hardware*

En lo que respecta a los componentes *hardware* del trabajo fin de grado [46] y según se ha recogido en "Desglose de elementos *hardware* necesarios para el punto de asistencia inteligente" mencionado en el capítulo 2, los componentes *hardware* de este son:

- **Micrófono:** Necesario para la entrada por voz de la petición del usuario.
- **Altavoz:** Necesario para la interacción con el usuario.
- **Raspberry Pi model B con 8Gb de RAM:** Componente que integrará los dos elementos anteriores y procesará la entrada de datos con la red neuronal para emitir la petición sintetizada al siguiente componente.
- **Arduino MKR 1310 WAN:** Necesario para enviar la petición sintetizada al servicio receptor, en este caso The Things Network mediante LoRaWAN.

Los componentes que se han descrito con anterioridad requieren de un ensamblado para que la información sea correctamente enviada entre ellos, a fin de que, en su conjunto, trabajen como el dispositivo físico del punto de asistencia inteligente anteriormente descrito.

El ensamblado de los componentes se refleja en la siguiente figura (ver figura 21).

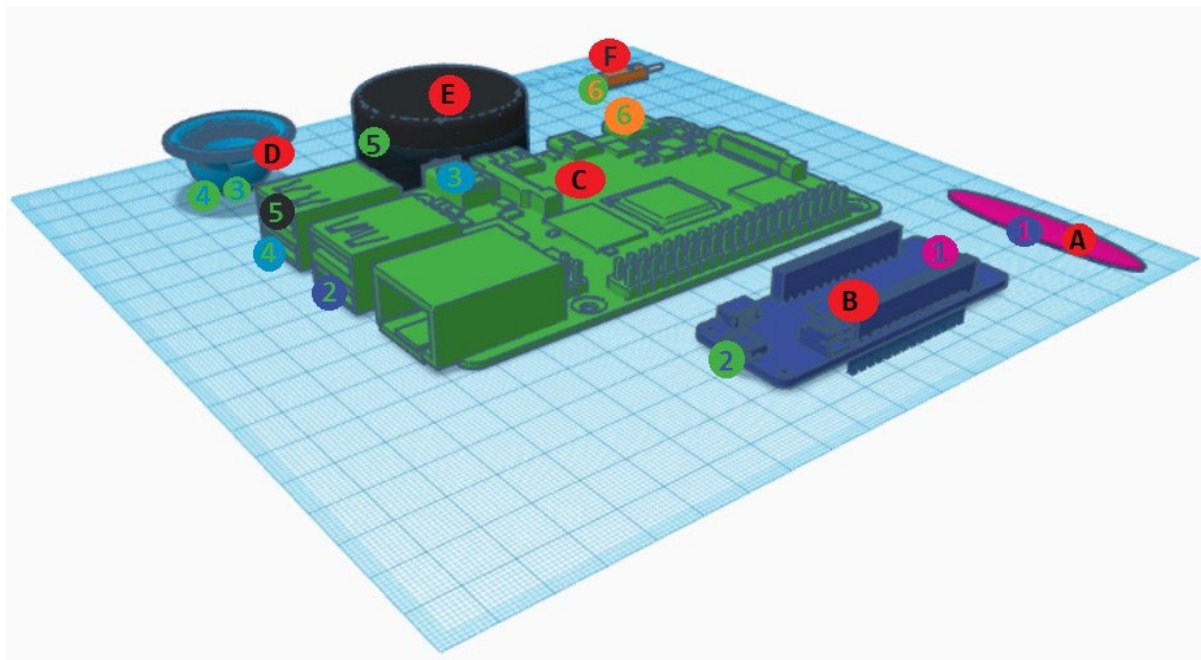


Figura 21: Diseño del ensamble de los componentes

Leyenda:

▪ **Elementos *hardware***

- A:** Antena del Arduino MKR 1310 WAN
- B:** Arduino MKR 1310 WAN
- C:** Raspberry Pi 4 Model B
- D:** Altavoz
- E:** Micrófono
- F:** Fuente de alimentación con interruptor

▪ **Conexiones entre elementos**

- 1:** Cable de antena (Arduino MKR 1310 WAN con Antena)
- 2:** Conexión usb a micro-usb (Raspberry Pi 4 Model B con Arduino MKR 1310 WAN)
- 3:** Conexión jack a jack (altavoz con Raspberry Pi 4 Model B)
- 4:** Conexión usb a mini-usb (Raspberry Pi 4 Model B con altavoz)
- 5:** Conexión usb a usb (Raspberry Pi 4 Model B con micrófono)
- 6:** Conexión usb a usb (Raspberry Pi 4 Model B con fuente de alimentación con interruptor)

Nota: El color de los elementos no hace referencia a los colores reales, es simplemente aclaratorio.

4.5.3. Diseño del componente *software*

En lo que respecta al *software* que se implementará en la Raspberry Pi destacan varios elementos clave, estos son:

- **Raspberry Pi Imager:** *Software* empleado en la preparación de la memoria microsd para contener el sistema operativo.
- **Ubuntu Server 22.04.02 LTS (64bits):** El sistema operativo de la Raspberry Pi compatible con todos los componentes que requiere el punto de asistencia inteligente.
- **Python 3.8:** Python [47] es el lenguaje de programación sobre el que se basa la implementación *software* de la Raspberry Pi y concretamente la versión 3.8 es compatible con todos los elementos descritos a continuación.

- **Miniforge3-Linux-aarch64:** Versión de Anaconda, un *software* de control de versiones para python y compatible con la Raspberry Pi.
- **Rasa NLU:** Rasa es una herramienta de procesamiento natural del lenguaje escrita en código abierto cuya funcionalidad es la extracción de *Intents* y *Entities* [48], la red neuronal se trabaja mediante esta herramienta debido a las facilidades que presenta para la misma, permite reentrenamiento, se puede usar sin conexión a Internet y trabaja mediante el lenguaje de programación Python.
- **Vosk:** Es una herramienta escrita en Python que haciendo uso de modelos previamente entrenados permite la conversión de voz a texto, necesaria para el procesamiento mediante Rasa NLU, además se puede usar sin conexión a Internet [49].
- **Arduino IDE:** Un *IDE* o Entorno de Desarrollo Integrado es una aplicación *software* que ayuda a los programadores a desarrollar código *software* de manera eficiente [50].

En lo que respecta al *software* que se implementará en la aplicación web, destacan los siguientes elementos:

- **Java:** [51] Java es un lenguaje de programación que se emplea muy a menudo en el diseño de aplicaciones web y ha tenido un uso muy continuado por parte de los usuarios en las últimas dos décadas, además es el lenguaje sobre el que se basa las herramientas descritas a continuación que emplearemos para el desarrollo de la aplicación web.
- **Java Spring Boot:** Es una herramienta *software* que se emplea para facilitar la incorporación de dependencias con Maven [52] y poder desplegar la aplicación en el servidor de manera sencilla.
- **Thymeleaf:** [53] Es una herramienta *software* que permite definir una plantilla que se emplea en conjunto a un modelo de datos para originar los documentos que permitirán controlar el funcionamiento de los distintos elementos que conformarán la aplicación web.

La aplicación web se implementará atendiendo al MVC o modelo-vista-controlador [54] el cual distingue los distintos elementos que conforman la aplicación web y facilita su implementación.

4.6. Diseño de las conexiones

El diseño de las conexiones es algo fundamental para que el envío de datos de forma inalámbrica se realicen de manera adecuada y eficiente, por ello las conexiones se ramifican en dos tipos diferentes, estas son:

4.6.1. Conexión a The Things Network

Esta conexión es necesaria para poder recolectar las peticiones procesadas procedentes del dispositivo físico, es decir funciona como un servicio receptor, para vincular ambos elementos es necesario previamente, crear una aplicación en The Things Network, una vez creada es necesario vincular un *end device*, el cual es el Arduino MKR 1310 WAN que estará conectado a la Raspberry Pi y cuya funcionalidad es ser la pasarela para el envío de datos entre el dispositivo físico y el servicio receptor. Este se realiza mediante un *DevEUI* o identificador de dispositivo único.

Para obtener el identificador de dispositivo único es necesario utilizar el Arduino IDE, este proceso está documentado en el **Anexo I. Obtención del DevEUI.**

Una vez obtenido el identificador hemos de introducirlo en la aplicación de The Things Network que hemos creado, de esta manera el envío de datos y posterior recolección podrán hacerse de manera satisfactoria.

4.6.2. Conexión a Node-red

La conexión con Node-red se explicará con más detalles en el capítulo siguiente, aunque en lo referente al diseño hemos de tener en cuenta varios detalles:

- **Requiere una vinculación con The Things Network**
- **Requiere una vinculación con la base de datos**
- **Requiere una vinculación con el bot de Telegram**

Al igual que ha ocurrido con la obtención del *DevEUI* el proceso de creación de un bot de Telegram también está documentado en el **Anexo II. Creación y configuración de un bot de Telegram.**

5. Implementación del trabajo fin de grado

Este capítulo tiene por objetivo explicar de manera estructurada y ordenada la implementación de los distintos paquetes de trabajo definidos en el capítulo 3: **Planificación del trabajo fin de grado** siguiendo las especificaciones definidas en el capítulo anterior: **Diseño del punto de asistencia**.

Debido a esto, la organización de este capítulo se dividirá en una serie de apartados para facilitar la comprensión del mismo, enfocándose cada uno en aquellos componentes *hardware*, *software* y de las conexiones respectivamente. Estos apartados a su vez se dividirán en subapartados cuando se traten varios componentes, como en el caso del *software* que se distinguen la aplicación web, de la implementación *software* del dispositivo físico.

5.1. Desarrollo del componente físico

El diseño descrito en el capítulo anterior será trasladado a un formato físico teniendo como componente base la madera, para ello será utilizará una cortadora láser, que permitirá que las piezas de la carcasa se obtengan de manera correcta. Una parte de este proceso de cortado se puede ver en la siguiente figura (ver figura 22).

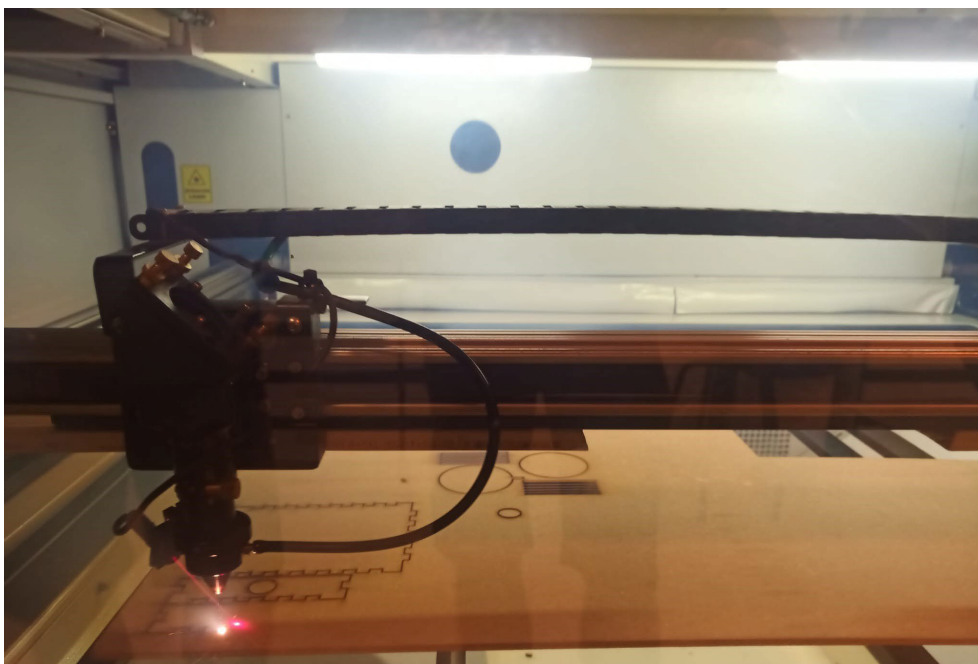


Figura 22: Corte láser de las piezas que componen la carcasa

Finalmente, las piezas de la carcasa debido al material por el que están constituidas, es decir la madera, es necesario un tratamiento posterior como el encolado, para que las piezas permanezcan unidas frente a adversidades externas, además se ha realizado un proceso de post-procesado con pintura roja para mejorar su apariencia final y barnizado lo que le confiere una superficie más adecuada al tacto además de proteger la pintura frente a posibles daños físicos, la carcasa resultante puede observarse en la siguiente figura (ver figura 23).



Figura 23: Vista trasera superior de la carcasa

La carcasa resultante será empleada en el siguiente apartado durante la etapa de Implementación *hardware*. Esta consiste en un armazón de madera con ranuras en la parte superior para facilitar el tránsito del aire con el objetivo de mejorar la refrigeración, los componentes que se encuentran en su interior, además está compuesta de 2 piezas divisibles que facilitan la instalación los componentes.

Nota: Aunque en este caso en concreto se emplee esta carcasa como diseño final, es necesario destacar que en producción el proceso puede variar significativamente, debido al empleo de otros materiales reciclables y por consiguiente el proceso de manufacturación, reestructuración de componentes, etc.

5.2. Implementación *hardware*

El *hardware* es la parte del trabajo fin de grado que podemos manipular físicamente, este apartado concreto tiene por objetivo detallar la implementación física del dispositivo junto al empaquetado de los componentes del mismo, en la carcasa descrita en el capítulo anterior.

El componente de partida es la bandeja interior de la carcasa, esta ha de ser separada de la parte superior para facilitar la instalación de los componentes internos, este componente se puede observar en la siguiente figura (ver figura 24).



Figura 24: Bandeja interior de la carcasa

Una vez separada se ha de instalar el micrófono, para ello se introduce el cable del mismo por el agujero correspondiente de la carcasa, tras ello el cable pasa por el agujero trasero de la bandeja interior y posteriormente una dobléz por el agujero trasero de la carcasa para poder liberar el cable sobrante (ver figura 25).

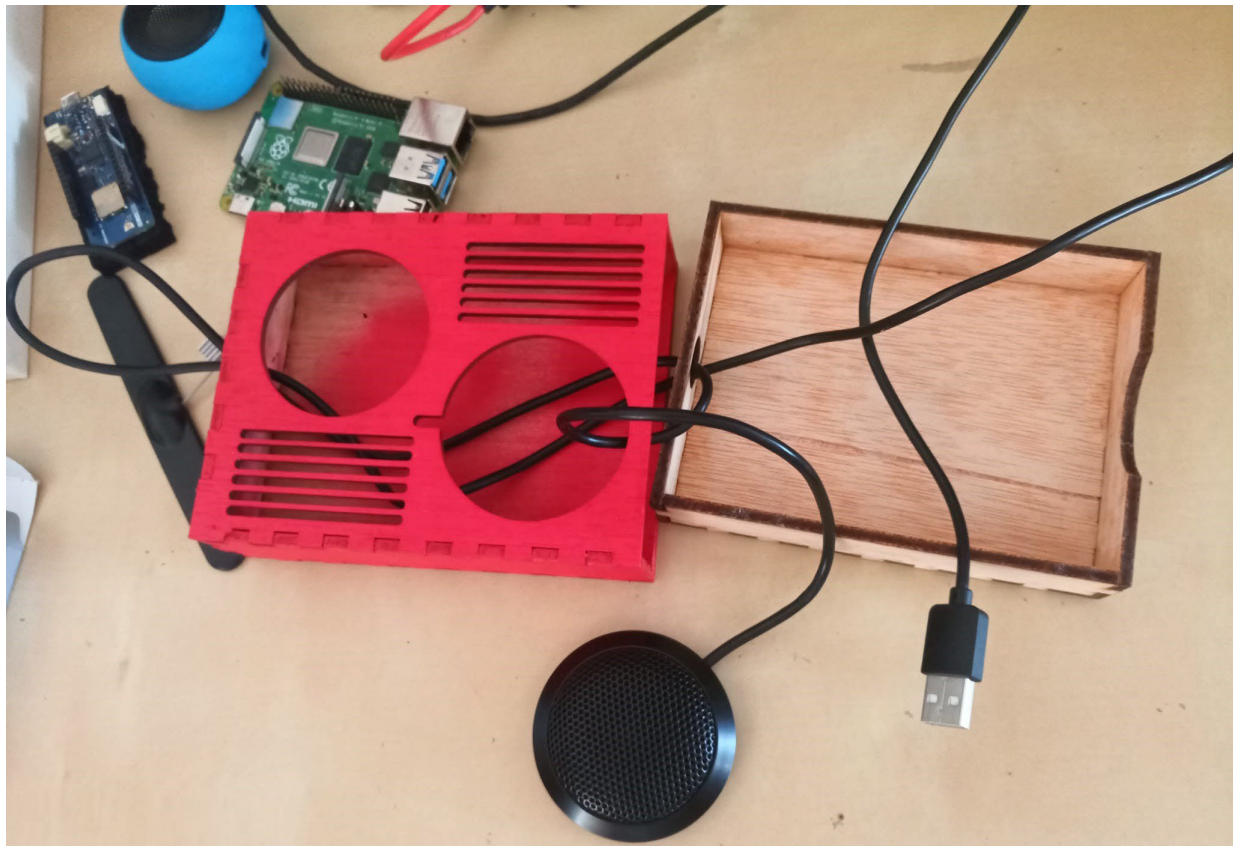


Figura 25: Etapa inicial de la instalación en la carcasa

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Posteriormente, se han de instalar tanto la Raspberry Pi como el Arduino MKR 1310 WAN atendiendo a la siguiente distribución (ver figura 26). El objetivo de esto es aprovechar de manera correcta el espacio y que los componentes se coloquen basándose en el diseño previo realizado, así como los cables que los conectan.



Figura 26: Etapa intermedia de la instalación en la carcasa

A continuación se instalan el altavoz y el micrófono acoplándolos en la parte superior empleando la ranura destinada para ello, finalmente se conecta la antena al Arduino MKR 1310 WAN y se coloca en el lateral resultando en la siguiente figura (ver figura 27).



Figura 27: Etapa final de la instalación en la carcasa

Nota: tanto la Raspberry Pi como el Arduino MKR 1310 WAN han de tener implementados sus respectivos programas antes del ensamblado.

5.3. Implementación *software*

El *software* es la parte del trabajo fin de grado que no es tangible, es decir, aquella parte que controla que los componentes *hardware* del trabajo fin de grado funcionen de manera adecuada.

Por ello es necesario diferenciar los dos grandes elementos hardware que dan lugar a este TFG, por un lado, encontramos el **dispositivo físico** que se encargará de recibir las peticiones, procesarlas y enviarlas al servicio receptor y, por otro lado, el servidor de la aplicación web, el cual si bien es cierto que no se define en profundidad, puesto que está asignado a un servicio externo, sí que mantiene la **aplicación web** cuyo componente *software* se define a continuación.

5.3.1. Implementación de la base de datos de la aplicación web

La base de datos es un elemento imprescindible para la gestión de la información que emplea la aplicación web para funcionar esta, tal y como se mencionó en el capítulo anterior, es del tipo SQL, permitiendo una conexión total con la aplicación web, el diseño de la base de datos que se empleó para esta aplicación concreta del punto de asistencia es el siguiente (ver listing 1):

Listing 1: Script de creación de la base de datos

```
create database IF NOT EXISTS nombre_de_tu_base_de_datos;
use nombre_de_tu_base_de_datos;

create user 'username_de_la_base_de_datos'@localhost identified by
'password_de_la_base_de_datos';
GRANT ALL PRIVILEGES on nombre_de_tu_base_de_datos.* TO
'username_de_la_base_de_datos'@localhost;
flush privileges;

CREATE TABLE IF NOT EXISTS centro_estudio (
  id_centro_estudio int(11) NOT NULL AUTO_INCREMENT,
  nombre varchar(50) NOT NULL,
  localizacion varchar(100) NOT NULL,
  PRIMARY KEY (id_centro_estudio)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;

CREATE TABLE IF NOT EXISTS dispositivo (
  id_dispositivo int(11) NOT NULL AUTO_INCREMENT,
```



```
id_centro_estudio int(11) NOT NULL,  
localizacion varchar(50) NOT NULL,  
nombre_dispositivo varchar(50) NOT NULL,  
PRIMARY KEY (id_dispositivo),  
FOREIGN KEY (id_centro_estudio)  
REFERENCES centro_estudio(id_centro_estudio)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1;  
  
CREATE TABLE IF NOT EXISTS gestor (  
id_gestor int(11) NOT NULL AUTO_INCREMENT,  
nombre varchar(30) NOT NULL,  
username varchar(30) NOT NULL,  
password varchar(50) NOT NULL,  
email varchar(50) NOT NULL,  
id_centro_estudio int(11) NOT NULL,  
administrador bool NOT NULL,  
PRIMARY KEY (id_gestor),  
FOREIGN KEY (id_centro_estudio)  
REFERENCES centro_estudio(id_centro_estudio)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;  
  
CREATE TABLE IF NOT EXISTS peticion (  
id_peticion int(11) NOT NULL AUTO_INCREMENT,  
fecha_peticion Date NOT NULL,  
tipo_peticion varchar(20) NOT NULL,  
id_dispositivo int(11) NOT NULL,  
estado int(1) NOT NULL,  
prioridad int(1) NOT NULL,  
id_gestor int(11),  
PRIMARY KEY (id_peticion),  
FOREIGN KEY (id_dispositivo) REFERENCES dispositivo(id_dispositivo),  
FOREIGN KEY (id_gestor) REFERENCES gestor(id_gestor)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

5.3.2. Implementación de la aplicación web

La aplicación web, tal y como se detalló en el diseño del capítulo 4 hará uso de Java como su lenguaje de programación y concretamente de Java Spring Boot, existen algunas herramientas como "Spring Initializr" [55] que permiten seleccionar las dependencias que empleará Spring para satisfacer las necesidades de la aplicación web.

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Las dependencias que requiere la aplicación web son:

- **spring-boot-starter-data-jpa**
- **spring-boot-starter-thymeleaf**
- **spring-boot-starter-web**
- **mysql-connector-j**
- **spring-boot-starter-test**
- **spring-boot-starter-validation**
- **spring-boot-maven-plugin**

Una vez obtenido el trabajo fin de grado inicial de Maven se han de implementar las diferentes clases que darán funcionalidad a la aplicación web, comenzando por el modelo se observan las siguientes clases (ver figura 28).

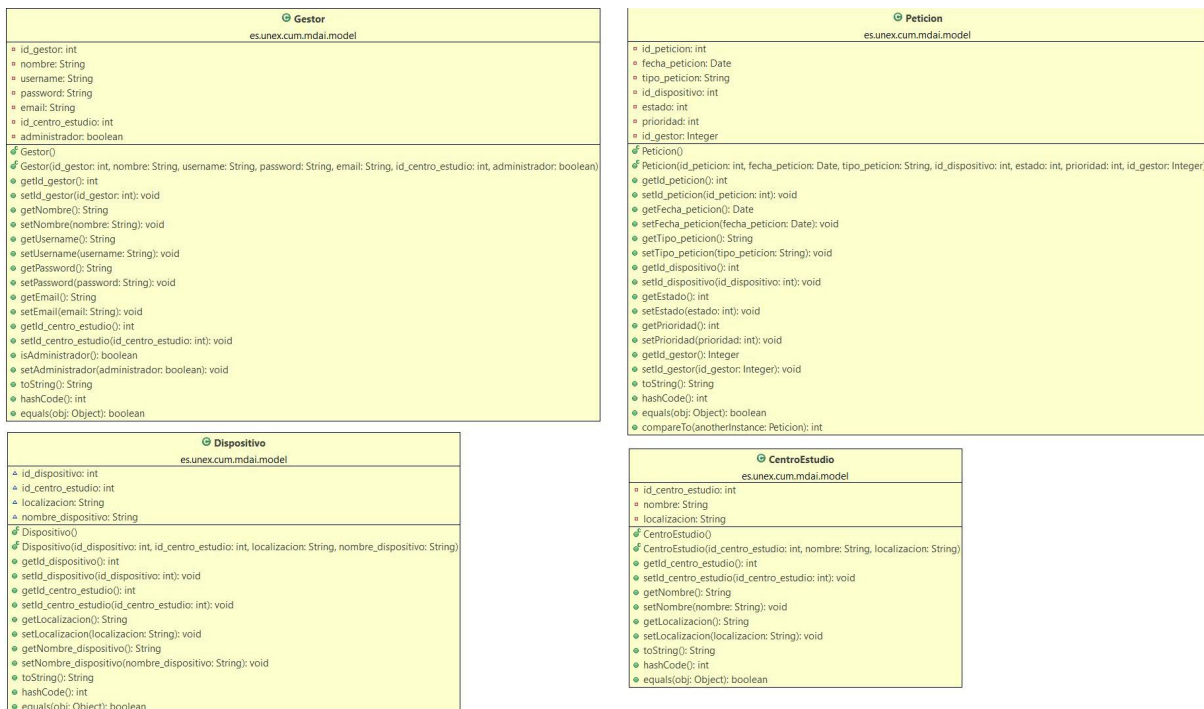


Figura 28: Clases que conforman el modelo de la aplicación web

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

El modelo tiene por objetivo asociar un elemento objeto a cada una de las diferentes clases de datos, de manera que los datos tomen unidad en función de la clase que constituyen.

A continuación se ha de implementar la clase controlador, pero previamente se requieren las clases intermedias que empleará el controlador para el tratamiento de la información, siendo estas las interfaces de servicio (ver figura 29), los servicios (ver figura 30) y el repositorio (ver figura 31).

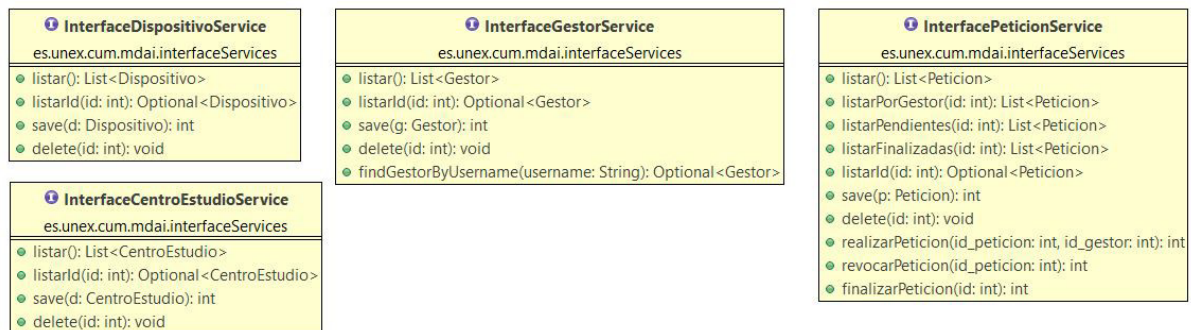


Figura 29: Clases que conforman las interfaces de servicio de la aplicación web

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

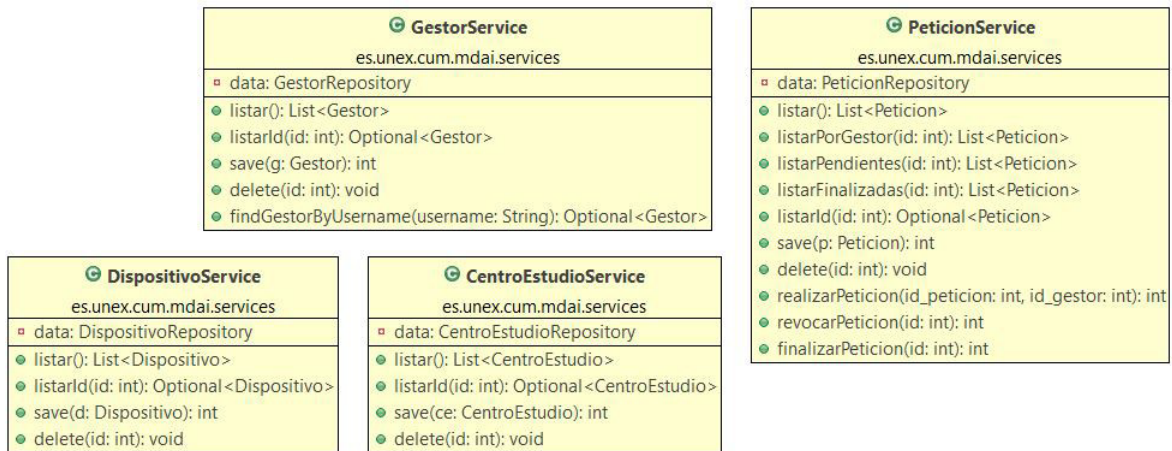


Figura 30: Clases que conforman los servicios de la aplicación web



Figura 31: Clases que conforman el repositorio de la aplicación web

Una vez definidas las clases que emplea el controlador, este se define en función del diagrama de la página siguiente (ver figura 32). La utilidad del controlador consiste en orquestar como las diferentes vistas harán uso de los datos y dar funcionalidad a las mismas, de manera que si ocurre una actualización, el controlador se encarga tanto de actualizar los datos de la vista como los de la base de datos subyacente.


 Controlador es.unex.cum.mdai.controller
<ul style="list-style-type: none"> ▫ serviceCentroEstudio: InterfaceCentroEstudioService ▫ serviceDispositivo: InterfaceDispositivoService ▫ serviceGestor: InterfaceGestorService ▫ servicePeticion: InterfacePeticionService ▫ usuario_Logueado: Gestor
<ul style="list-style-type: none"> ● dispositivosSelect(): List<Dispositivo> ● centrosestudiosSelect(): List<CentroEstudio> ● gestoresSelect(): List<Gestor> ● procesarLogin(gestor: Gestor, result: BindingResult, model: Model): String ● login(model: Model): String ● logout(model: Model): String ● adminIndex(model: Model): String ● adminCentrosEstudio(model: Model): String ● adminDispositivos(model: Model): String ● adminGestores(model: Model): String ● adminPeticones(model: Model): String ● agregarCentroEstudio(model: Model): String ● agregarDispositivo(model: Model): String ● agregarGestor(model: Model): String ● agregarPeticion(model: Model): String ● saveCentroEstudio(centroestudio: CentroEstudio, result: BindingResult, model: Model): String ● saveDispositivo(dispositivo: Dispositivo, result: BindingResult, model: Model): String ● saveGestor(gestor: Gestor, result: BindingResult, model: Model): String ● savePeticion(peticion: Peticion, result: BindingResult, model: Model): String ● editarCentroEstudio(id_centro_estudio: int, model: Model): String ● editarDispositivo(id_dispositivo: int, model: Model): String ● editarGestor(id_gestor: int, model: Model): String ● editarPeticion(id_peticion: int, model: Model): String ● deleteCentroEstudio(id_centro_estudio: int, model: Model): ResponseEntity<String> ● deleteDispositivo(id_dispositivo: int, model: Model): ResponseEntity<String> ● delete(id_gestor: int, model: Model): ResponseEntity<String> ● deletePeticion(id_peticion: int, model: Model): ResponseEntity<String> ● gestorPeticones(model: Model): String ● gestorRealizandose(model: Model): String ● gestorFinalizadas(model: Model): String ● realizandosePeticion(id_peticion: int, model: Model): ResponseEntity<String> ● revocarPeticion(id_peticion: int, model: Model): ResponseEntity<String> ● finalizarPeticion(id_peticion: int, model: Model): ResponseEntity<String>

Figura 32: Clase controladora de la aplicación web

Tras haber definido el modelo y el controlador se ha de definir una clase auxiliar (ver figura 33) previa al apartado de la vista, la finalidad de esta clase es la de establecer la conexión con la base de datos.

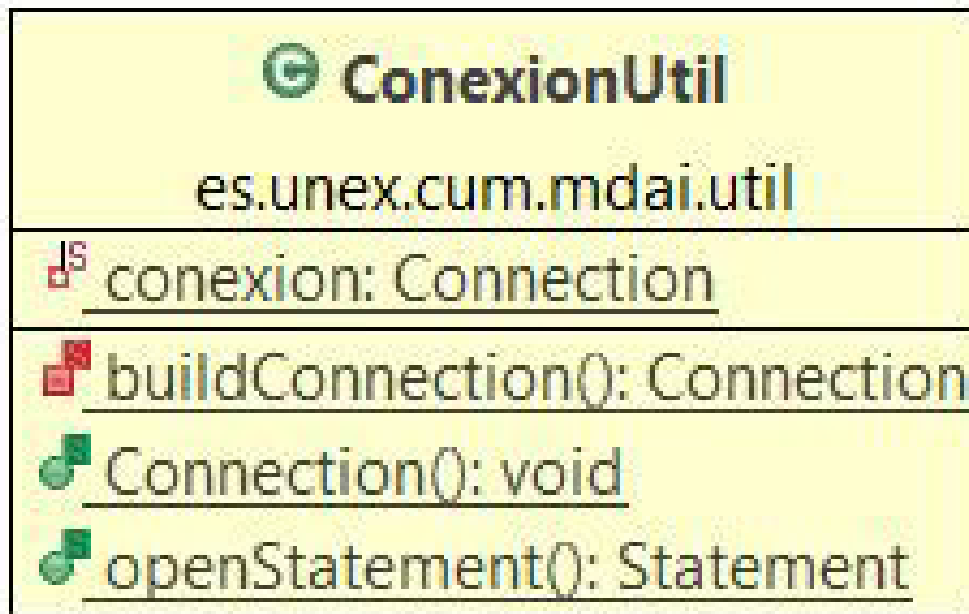


Figura 33: Clase auxiliar para las conexiones internas de la aplicación web

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Finalmente, la implementación del diagrama completo que incluye los apartados del modelo y el controlador se ve reflejada en la siguiente figura (ver figura 34).

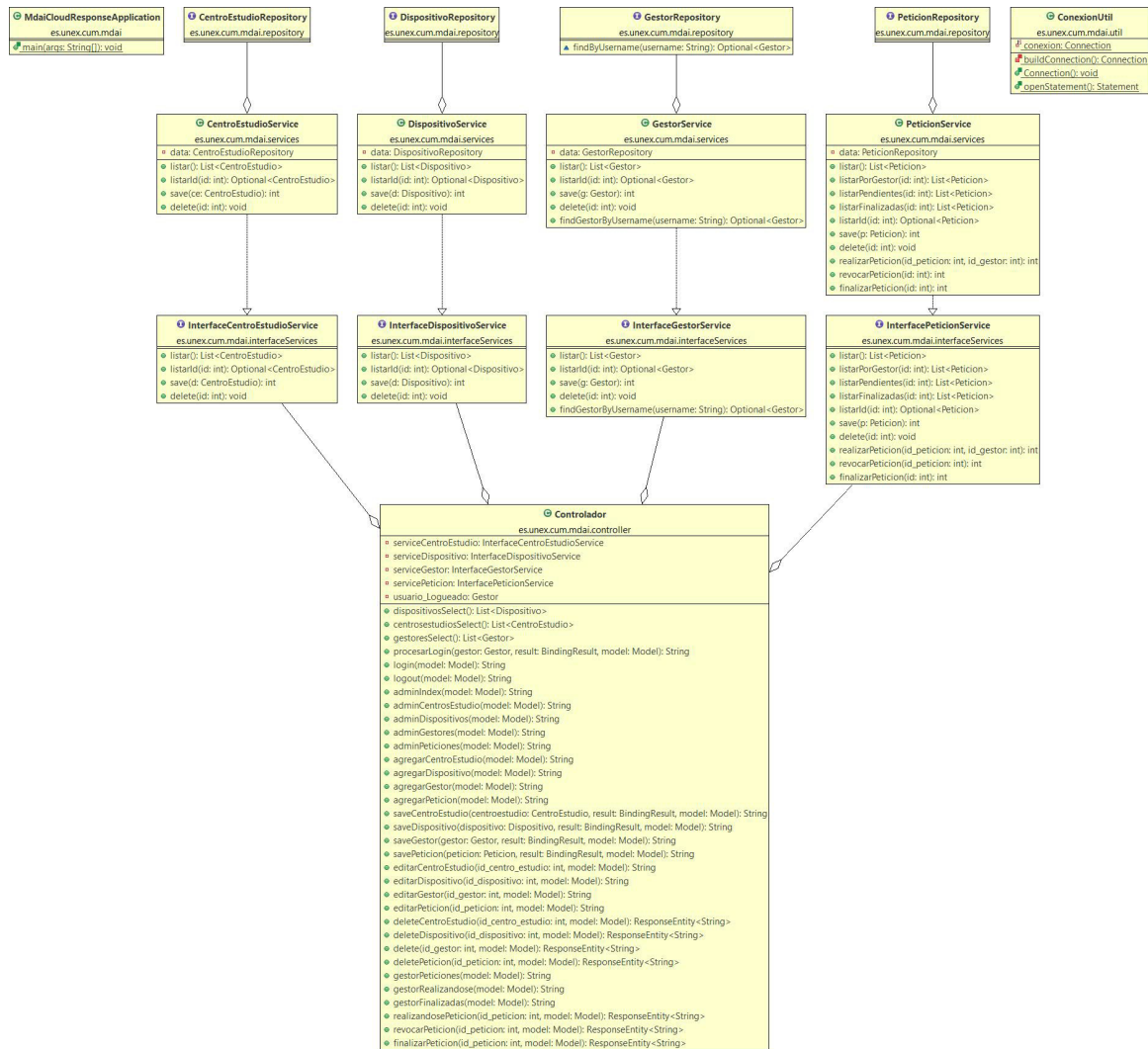


Figura 34: Diagrama de la implementación Modelo - Controlador de la aplicación web

Una vez definidas las clases previas necesarias se procede a explicar el apartado de la vista, este será visualizado por el usuario gestor y le permitirá realizar las modificaciones oportunas en el sistema, de manera que pueda atender las peticiones de manera satisfactoria.

Es en este apartado donde se hace uso de Thymeleaf, su utilidad radica en las capacidades de modificación e interconexión vista-controlador que proporciona. Permitiendo el uso de estructuras condicionales como *th:if*, relacionadas con objetos como *th:field*, de asignación como *th:text* y de control de errores.

Al igual que ocurrió en el capítulo anterior, la implementación de la aplicación web posee dos vertientes, la relacionada con el usuario gestor y la relacionada con el usuario administrador del sistema, siendo estas descritas a continuación. Cabe destacar que la distinción entre ambos tipos de usuario ocurre en el inicio de sesión, cuya vista es compartida por ambos (ver figura 35).

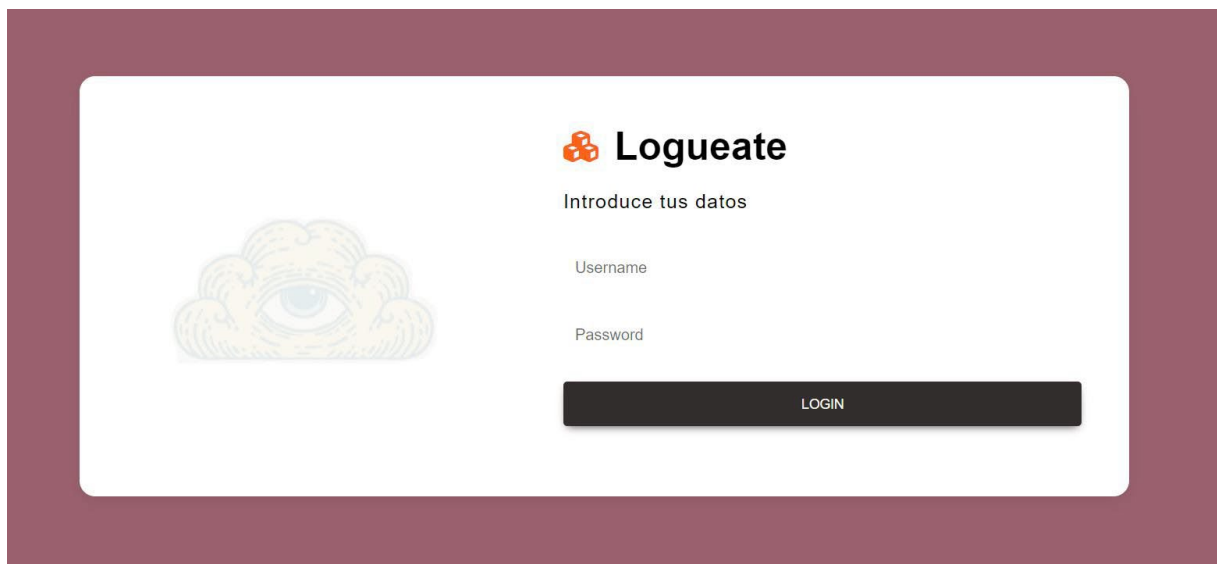


Figura 35: Vista común del inicio de sesión

Implementación de la aplicación web para el usuario gestor

El usuario gestor, según la información detallada en capítulos anteriores, tiene una labor de monitorización de las peticiones y atención de estas, por ello una vez que ha recibido la notificación de una nueva petición, y tras el anterior inicio de sesión se le mostrará la siguiente vista (ver figura 36).

Fecha de realizacion	Tipo	Lugar	Acciones
2023-05-16	asistencia del bedel	Centro Universitario de Merida	<button>ASUMIR</button>

Figura 36: Vista de visualización de peticiones no atendidas

Una vez seleccionada una petición para su atención, la petición pasará a estar "en proceso", y en consecuencia dejará de aparecer en la anterior vista para el resto de gestores, al ser una petición en proceso aparecerá en la vista de peticiones que están siendo atendidas (ver figura 37).

Fecha de realizacion	Tipo	Lugar	ID Gestor	Acciones
2023-05-16	asistencia del bedel	Centro Universitario de Merida	2	<button>REVOCAR</button> <button>TERMINAR</button>

Figura 37: Vista de atención de peticiones

Finalmente, tras haber atendido la petición, el usuario gestor la marcará como "finalizada" quedando archivada en la lista de peticiones finalizadas (ver figura 38).

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Fecha de realizacion	Tipo	Lugar	ID Gestor
2023-05-11	bedel mando	Centro Universitario de Merida	2

Figura 38: Vista de visualización de peticiones finalizadas

En lo respectivo al usuario gestor no queda ninguna funcionalidad más por implementar, veamos las del usuario administrador.

Implementación de la aplicación web para el usuario administrador

El usuario administrador según la información detallada en capítulos anteriores, tiene una labor de gestión completa del sistema, por ello posee libertad absoluta de visualización, creación, modificación y eliminación de cualquier elemento del sistema, una vez inicie sesión verá la siguiente vista (ver figura 39).

Administrar Dispositivos	Administrar Gestores	Administrar Peticiones	Administrar Centros de Estudio	Cerrar Sesion
--------------------------	----------------------	------------------------	--------------------------------	---------------



Figura 39: Vista de inicio de sesión del administrador

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

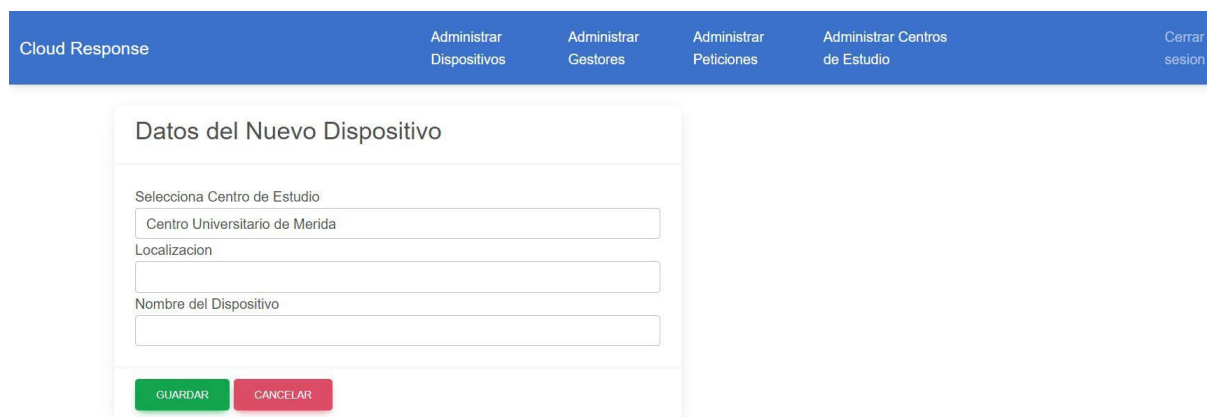
En lo relativo a la administración de dispositivos, el usuario administrador podrá visualizar, modificar y eliminarlos (ver figura 40) y crear uno nuevo (ver figura 41).



The screenshot shows a web interface for device management. At the top, there is a blue navigation bar with the text 'Cloud Response' on the left and several menu items: 'Administrar Dispositivos', 'Administrar Gestores', 'Administrar Peticiones', 'Administrar Centros de Estudio', and 'Cerrar sesion'. Below the navigation bar, there is a 'NUEVO' button. The main content area contains a table with the following data:

ID Dispositivo	ID Centro de Estudio	Localizacion	Nombre	Acciones
1	1	Centro Universitario de Merida	A8610A3032337611	EDITAR ELIMINAR

Figura 40: Vista de administración de los dispositivos



The screenshot shows a form titled 'Datos del Nuevo Dispositivo'. The form has the following fields and buttons:

- Selección de 'Centro de Estudio' (Centro Universitario de Merida)
- Localización (input field)
- Nombre del Dispositivo (input field)
- Buttons: 'GUARDAR' (green) and 'CANCELAR' (red)

Figura 41: Vista para añadir y modificar dispositivos

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

En el apartado de administración de gestores, el usuario administrador podrá visualizar, modificar y eliminarlos (ver figura 42) y crear nuevos (ver figura 43).

ID Gestor	Nombre	Username	Password	Email	ID Centro de Estudio	Administrador	Acciones
1	admin	admin	1234	admin@gmail.com	1	true	EDITAR ELIMINAR
2	Javier	javier	1234	javy@gmail.com	1	false	EDITAR ELIMINAR

Figura 42: Vista de administración de los gestores

Cloud Response	Administrar Dispositivos	Administrar Gestores	Administrar Peticiones	Administrar Centros de Estudio	Cerrar sesion
----------------	--------------------------	----------------------	------------------------	--------------------------------	---------------

Datos del Nuevo Gestor

Nombre

Username

Password

Email

Selecciona Centro de Estudio

Es administrador?

[GUARDAR](#) [CANCELAR](#)

Figura 43: Vista para añadir y modificar gestores

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

En lo que respecta a la administración de peticiones, el usuario administrador podrá visualizar, modificar y eliminarlas (ver figura 44) y crear nuevas (ver figura 45).

ID Petición	Fecha de realización	Tipo	ID del Dispositivo	Estado	Prioridad	ID Gestor	Acciones
3	2023-05-16	Asistencia del Bedel	1	2	1	2	EDITAR ELIMINAR

Figura 44: Vista de administración de las peticiones

Datos de la Nueva Peticion

Tipo de Peticion
Fuego

Selección Dispositivo
A8610A3032337611

Estado de la Peticion
Creada

Prioridad de la Peticion
Baja

[GUARDAR](#) [CANCELAR](#)

Figura 45: Vista para añadir y modificar peticiones

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Finalmente, en lo relativo a la administración de centros de estudio, el usuario administrador podrá visualizar, modificar y eliminarlos (ver figura 46) y crear nuevos (ver figura 47).

ID	Nombre	Localizacion	Acciones
1	Centro Universitario de Merida	Calle Sta. Teresa Jornet, 38, 06800 Merida, Badajoz	EDITAR ELIMINAR

Figura 46: Vista de administración de los centros de estudio

Datos del Nuevo Centro Estudio

Nombre

Localizacion

[GUARDAR](#) [CANCELAR](#)

Figura 47: Vista para añadir y modificar centros de estudio

Nota I: En el directorio "resources" se encuentra el fichero "application.properties" el cual se emplea para establecer los datos de conexión con la base de datos, además de otros parámetros útiles, en este trabajo fin de grado se emplearon los siguientes (ver listing 2):

Listing 2: Código de configuración del "application.properties"

```
spring.datasource.url=jdbc:mysql://localhost:3306/Nombre_de_tu_base_de_datos
spring.datasource.username=Tu_username_de_la_base_de_datos
spring.datasource.password=Tu_password_de_la_base_de_datos
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#si usamos create se crearian de 0 cada vez que se lanza
spring.jpa.hibernate.ddl-auto=update
#spring.web.resources.static-locations=classpath:/images/
```

Nota II: si bien el administrador tiene control total de la aplicación, su función principal será la de dar de alta nuevos gestores y dispositivos en el sistema.

5.3.3. Implementación del *software* del dispositivo físico

El dispositivo físico tal y como se detalló en el **Diseño del punto de asistencia** estará compuesto por dos dispositivos *hardware* siendo estos las Raspberry Pi 4 modelo B y el Arduino MKR 1310 WAN, la implementación *software* de estos elementos se describe a continuación:

Implementación *software* de la Raspberry Pi 4

La Raspberry Pi 4 modelo B hará uso de Python como su lenguaje de programación y actuará bajo el sistema operativo "Ubuntu Server 22.04.02 LTS de 64bits".

Para comenzar a trabajar es recomendable el uso de Conda, un software de gestión de paquetes que permite trabajar con entornos virtuales [56] y distintas versiones de Python.

Una vez se ha instalado Conda, o en este caso Miniforge (una adaptación de Conda compatible con Raspberry Pi), es conveniente crear un entorno en la versión de Python 3.8, puesto que es compatible con todas las librerías y paquete que emplearemos para dar funcionalidad al asistente conversacional, además de un directorio de trabajo que recogerá todo el trabajo fin de grado de implementación *software* del dispositivo físico.

Acto seguido se ha de instalar Vosk, el componente de interpretación de la entrada por voz del usuario que transformará esta a una entrada válida para el asistente conversacional. Para adaptar Vosk al lenguaje objetivo de este trabajo fin de grado, el español, se hace uso de modelos pre-entrenados de lenguaje, estos modelos se obtienen de Alphacephei [57]. Una vez descargado es necesario extraer el contenido en el directorio de trabajo, aunque es recomendable crear un subdirectorio en este para facilitar la organización.

A continuación es necesario instalar Rasa, esta tarea es compleja debido a que algunas dependencias de Rasa como Tensorflow no están adaptadas en su totalidad para la arquitectura ARM de 64 bits (aarch64) [58], por ello este proceso se explica en detalle en el **Anexo IV. Instalación de Rasa NLU en Raspberry Pi 4 Modelo B.**

Una vez instalado Rasa, es necesario inicializarlo, se recomienda emplear el anteriormente citado directorio de trabajo, tras ello se debe introducir el siguiente comando (ver listing 3):

Listing 3: Código para comenzar a configurar Rasa

```
|| $ rasa init
```

En ese momento comenzará una interacción con Rasa que desembocará en la solicitud de Rasa de entreno de un modelo inicial, esto es recomendable para posteriormente hacer el reentreno de un modelo basado en el aprendizaje que le demos a Rasa. Tras haber realizado los pasos previos en el directorio de trabajo, se encontrarán nuevos elementos generados por Rasa, tal y como se puede apreciar en la figura de la página siguiente (ver figura 48).

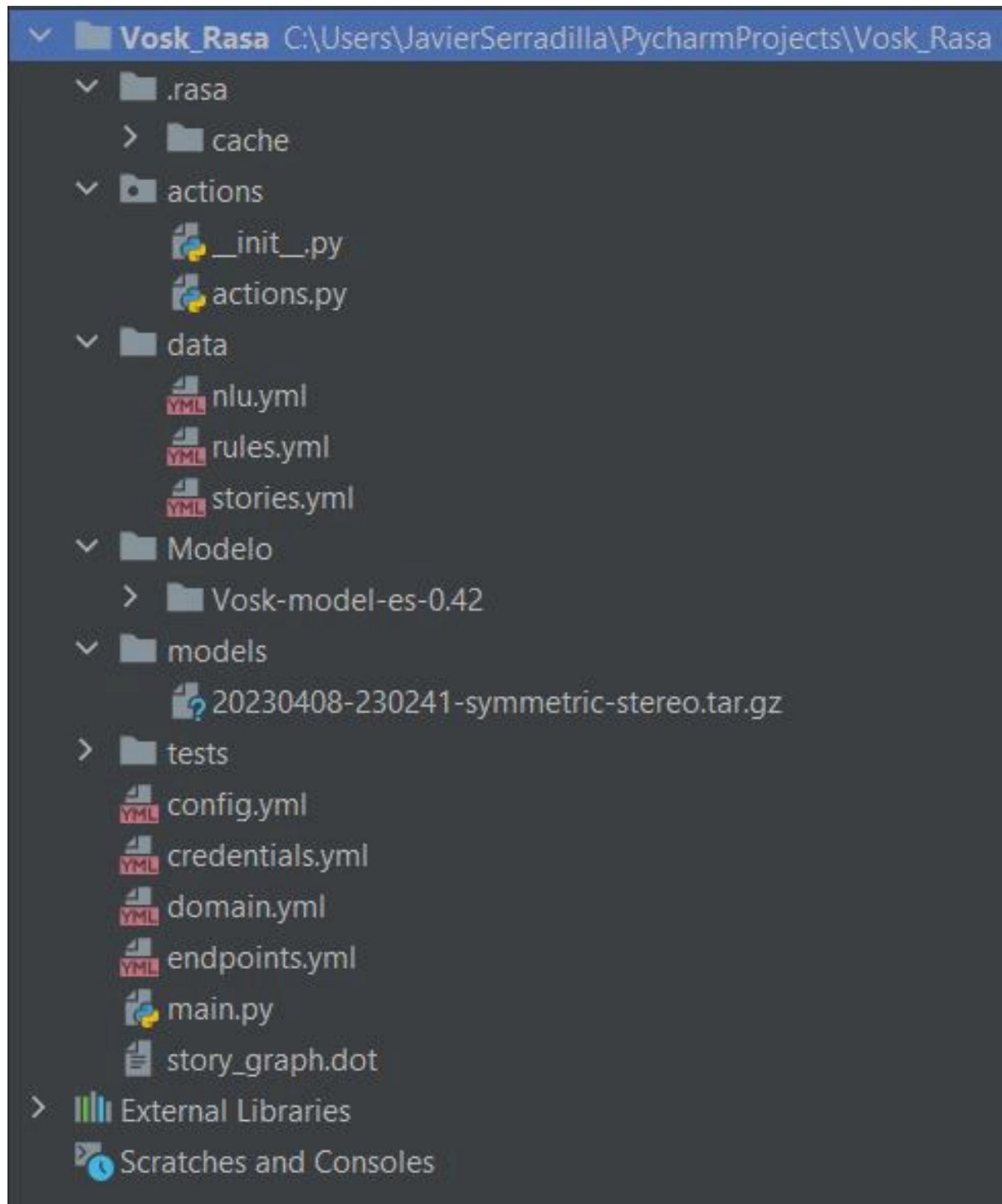


Figura 48: Directorio de trabajo de la implementación *software* de la Raspberry Pi del dispositivo físico

El siguiente paso consiste en añadir datos de entrada al núcleo de Rasa para que este asocie las palabras clave con distintas *Entities*, que posteriormente serán empleadas para la clasificación de peticiones y procesadas para su envío al servicio receptor usando como intermediario al Arduino MKR 1310 WAN. Para ello, dentro del subdirectorio "data" encontramos varios ficheros importantes, en el fichero "nlu.yml" se definen las *Intents* junto a las *Entities* con las que trabajará Rasa de la siguiente manera (ver listing 4):

Listing 4: Código de configuración del fichero "nlu.yml" I

```
nlu:  
- intent: nombre de la petición  
  examples: |  
    - [Entity a reconocer](Categoria de la entity)  
    - ...
```

Además, al definir una categoría hay que especificarle que tipo de elementos entran en esa categoría, para ello hay que añadir los elementos de la siguiente manera en el mismo fichero (ver listing 5):

Listing 5: Código de configuración del fichero "nlu.yml" II

```
- lookup: Categoria de la entity  
  examples: |  
    - palabra que entra en esa categoria  
    - ...
```

En el fichero "rules.yml" se definen las normas de conducta, es decir, acciones que se realizan de manera directa una vez el usuario realiza un patrón, como preguntar la hora, despedirse cuando el usuario se despida, etc. Esto se añade en el fichero de la siguiente manera (ver listing 6):

Listing 6: Código de configuración del fichero "rules.yml"

```
rules:  
- rule: Ejemplo de norma  
  steps:  
    - intent: nombre de la intent a ejecutar  
    - action: intent o accion que activa la norma
```

En el fichero "stories.yml" se definen orientaciones que suelen tomar las conversaciones y que ayudan a Rasa a entender el contexto de la conversación con el usuario, para ello se añaden al fichero las siguientes líneas de código (ver listing 7):

Listing 7: Código de configuración del fichero "stories.yml"

```
stories:  
- story: nombre de la historia  
  steps:  
  - intent: intent inicial  
  - action: intent o accion que activa la historia  
  - intent: intent siguiente  
  - action: intent o accion que activa la intent siguiente  
  - ...
```

Volviendo al directorio original de trabajo encontramos varios ficheros importantes como "config.yml" que se utiliza para especificar elementos de funcionamiento durante el entreno de la red neuronal y la identificación de *Intents* y *Entities*, el fichero "credentials.yml" es necesario para el funcionamiento del trabajo fin de grado, para ello hay que añadir las siguientes líneas de código (ver listing 8):

Listing 8: Código de configuración del fichero "config.yml"

```
rest:  
  
rasa:  
  url: "http://localhost:5002/api"
```

De igual manera en el fichero "endpoints.yml" hay que añadir las siguientes líneas de código (ver listing 9):

Listing 9: Código de configuración del fichero "endpoints.yml"

```
action_endpoint:  
  url: "http://localhost:5055/webhook"
```

El fichero "domain.yml" es uno de los ficheros más importantes de Rasa, pues actúa como fichero de configuración del núcleo de Rasa, hay varias líneas importantes que se mostrarán en la página siguiente (ver listing 10):

Listing 10: Código de configuración del fichero "domain.yml"

```
# Permite almacenar las Entities en unidades de almacenamiento
# dinamicas llamadas slots
config:
  store_entities_as_slots: true

# Permite parametrizar la configuracion de la sesion
session_config:
  session_expiration_time: 60
  carry_over_slots_to_new_session: true

# Definicion de las intents que se definieron
# en el fichero "nlu.yml" el parametro 'use_entities: true'
# se usa para que reconozca entities en esa intent
intents:
- nombre de la intent:
  use_entities: true
- ... :
  ...

# Definicion de las entities que se definieron
# en el fichero "nlu.yml"
entities:
- nombre de la entity

# Definicion de las respuestas
# deben empezar por 'utter_'
responses:
  utter_nombre de la intent a la que responde:
    - text: La respuesta que da Rasa
    - text: ...

# Definicion de las acciones, que se definiran a continuacion
# deben empezar por 'accion_'
actions:
- accion_nombre de la accion
- accion_...
```

En el subdirectorio "actions" encontramos el fichero "actions.py" este fichero escrito en python contiene las clases y funciones que empleará el asistente, totalmente personalizable, pero sigue unos estándares, tal y como se observa en el código explicativo de la página siguiente (ver listing 11):

Listing 11: Código de configuración del fichero "actions.py"

```
class Action_Nombre_de_la_Action(Action):

    def name(self) -> Text:
        return "accion_a_la_que_responde"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        parametro_a_recoger = next(
            tracker.get_latest_entity_values(
                "nombre_de_la_entity"), None)

        msg = f"Mensaje a responder"
        dispatcher.utter_message(text=msg)

    return []
```

Tras haber pre-configurado Rasa solo es necesario ejecutar en el terminal el siguiente comando para que de comienzo el entreno del modelo de inteligencia artificial (ver listing 12):

Listing 12: Código para comenzar a entrenar el modelo de inteligencia artificial en Rasa

```
|| $ rasa train
```

Cabe destacar que para este trabajo fin de grado se emplea un fichero auxiliar descrito en el directorio raíz de nombre "main.py" que define el patrón de conducta del asistente y qué hacer con las entradas que recibe Rasa. Este fichero tiene las siguientes líneas de código que deberían ser comunes a trabajos fin de grado similares al punto de asistencia inteligente (ver listing 13):

Listing 13: Código del main.py

```
# Incorporamos json para el Speech to Text
import json

# Para incorporar el modelo
import os
ROOT_DIR = os.path.abspath(os.curdir)
ROOT_DIR += "/Modelo/Vosk-model-es-0.42"

# Incorporamos serial para la comunicacion con la Raspberry
```

```
import serial
import time

# Esto es para la conexion al Arduino MKR 1310 WAN
# Obtener nombre del dispositivo serial:
dmesg | grep -v disconnect | grep -Eo "tty(ACM|USB)." | tail -1
ser = serial.Serial('/dev/nombre_dispositivo_serial', 9600)
ser.flushInput()

from vosk import Model, KaldiRecognizer
import sounddevice as sd

# Obtenemos el samplerate para el Kaldi recognizer
device_info = sd.query_devices(sd.default.device[0], 'input')
samplerate = int(device_info['default_samplerate'])

# Para la voz del asistente
import pyttsx3

# Para que trabaje junto a Rasa
import requests

# Se le pasa el path completo con una 'r' al inicio
model = Model(ROOT_DIR)

# Establecemos las funciones de queue y callback
# para sounddevice
import queue
q = queue.Queue()

def recordCallback(indata, frames, time, status):
    if status:
        print(status, file=sys.stderr)
        q.put(bytes(indata))

# samplerate es la frecuencia
recognizer = KaldiRecognizer(model, samplerate)
# esto capta todo el chunk y no solo palabras sueltas
recognizer.SetWords(False)

#Inicializamos el Speech to Text
escuchando = False
```

```
#Inicializamos la voz
engine = pyttsx3.init()
# esa propiedad es la velocidad de la voz (180),
# si ponemos mas va mas rapido
engine.setProperty("rate", 180)
# Para cambiar la voz y establecer una femenina
voices = engine.getProperty('voices')
engine.setProperty('voice', 'spanish+f1')
# Para cambiar el volumen y aumentarlo un 50%
# volume = engine.getProperty('volume')
# engine.setProperty('volume', volume+0.50)

# Funcion de temporizacion para enviar un mensaje cada 2
# minutos tal y como pide el servicio receptor
def temporizacion(t):
    while t:
        mins, secs = divmod(t, 60)
        timer = '{:02d}:{:02d}'.format(mins, secs)
        print(timer, end="\r")
        time.sleep(1)
        t -= 1
    engine.say("Ya estoy disponible para enviar otra peticion")
    engine.runAndWait()

def get_command():
    escuchando = True
    try:
        with sd.RawInputStream(dtype='int16',
                               channels=1, callback=recordCallback):
            while escuchando:
                data = q.get()
                if recognizer.AcceptWaveform(data):
                    recognizerResult = recognizer.Result()
                    # convierte el la informacion del recognizer
                    # en un diccionario
                    resultDict = json.loads(recognizerResult)
                    if not resultDict.get("text", "") == "":
                        response = recognizerResult[14:-3]
                        escuchando = False
                        sd.stop()
                        return response
                    else:
                        print("No hay sonido entrante")
```

```
except Exception as e:
    engine.say("Ha ocurrido un error al escucharte")
    engine.runAndWait()

# Comienza el programa
if __name__ == '__main__':
    texto = "Default"

    # Aqui va el sonido para indicar que
    # el Speech to Text ya esta cargado
    engine.say("Ya estoy disponible para atender
    tus peticiones")
    engine.runAndWait()

    engine.say("Cuando termine solamente diga
    TERMINAR para que finalice mi servicio")
    engine.runAndWait()

    # Configurar en funcion de donde se encuentre
    sender = "AulaDefault"

    respuestaRasa = "Default"

    while texto != "terminar":
        texto = get_command()

        if texto != "":
            engine.say("Entendido, voy a procesar tu peticion,
            un momento")
            engine.runAndWait()
            hayMensaje = False

            # Se envia la peticion a Rasa
            try:
                respuestaRasa = requests.post(
                    'http://localhost:5002/webhooks/rest/webhook',
                    json={"sender": sender, "message": texto})
                hayMensaje = True
            except requests.exceptions.ConnectionError:
                engine.say("Algo ha fallado con el servidor")
                engine.runAndWait()
```



```
respuestaTextual = ""
if hayMensaje:
    for i in respuestaRasa.json():
        respuestaTextual = (f' {i['text']}')

if respuestaTextual.isnumeric() and not 0:
    # Se procesa el mensaje y se envia
    mensajeBytes = respuestaTextual.encode('utf-8')
    ser.write(mensajeBytes)

    engine.say("Se ha enviado la peticion,
    ha de esperar 2 minutos
    para volver a realizar otra")
    engine.runAndWait()
    temporizacion(120)
elif respuestaTextual == 0:
    engine.say("Ha ocurrido un error al enviar
    la peticion, intentelo de nuevo")
    engine.runAndWait()
else:
    engine.say(f' {respuestaTextual}')
    engine.runAndWait()
else:
    engine.say("Por favor, realice una peticion
    o diga TERMINAR para que finalice mi servicio")
    engine.runAndWait()

# Indicamos al usuario que el programa finalizara
engine.say("Hasta luego, que tengas un buen dia")
engine.runAndWait()

# Finalizamos el programa
exit(0)
```

Uno de los últimos pasos es generar los siguientes *scripts* de bash [59] para poder automatizar el funcionamiento del asistente conversacional cuando se inicie la Raspberry Pi (este proceso tarda en torno a 7 minutos debido a las limitaciones de la Raspberry), los *scripts* son los siguientes:

Para activar las *actions* definidas en Rasa (ver listing 14):

Listing 14: Código del script "startRasaActions.sh"

```
#!/bin/bash
source $HOME/miniforge3/etc/profile.d/conda.sh
conda activate VoskRasa # Cambiar por el nombre de tu entorno miniforge3
cd /home/pi-cume/Vosk_Rasa/
rasa run actions
```

Para activar el intercambio de mensajes con Rasa (ver listing 15):

Listing 15: Código del script "startRasaServer.sh"

```
#!/bin/bash
source $HOME/miniforge3/etc/profile.d/conda.sh
conda activate VoskRasa # Cambiar por el nombre de tu entorno miniforge3
cd /home/pi-cume/Vosk_Rasa/
rasa run -m models --endpoints endpoints.yml --port 5002 --credentials
credentials.yml
```

Para activar el asistente conversacional con todas sus dependencias (ver listing 16):

Listing 16: Código del script "startRasaMain.sh"

```
#!/bin/bash
source $HOME/miniforge3/etc/profile.d/conda.sh
conda activate VoskRasa # Cambiar por el nombre de tu entorno miniforge3
cd /home/pi-cume/Vosk_Rasa/
bash startRasaServer.sh &
bash startRasaActions.sh &
sleep 180
python main.py
```

El último paso consiste en crear un servicio de Linux que se ejecute durante el arranque de la Raspberry Pi y ejecute el asistente de forma automática, para ello en el directorio "/etc/systemd/system/" hemos de crear un servicio, en esta aplicación el servicio se llamará "iniciarNou.service" y su código de configuración se muestra en la página siguiente (ver listing 17):

Listing 17: Código del servicio "iniciarNou.service"

```
[Unit]
Description=Servicio para iniciar Rasa y Vosk - Nou
StartLimitIntervalSec=0

[Service]
Type=simple
User=pi-cume
WorkingDirectory=/home/pi-cume/Vosk_Rasa/
ExecStart=/home/pi-cume/Vosk_Rasa/startRasaMain.sh

[Install]
WantedBy=multi-user.target
```

Finalmente, lo sincronizamos con el sistema con los siguientes comandos (ver listing 18):

Listing 18: Códigos para sincronizar el servicio con el sistema

```
sudo systemctl enable iniciarNou.service
sudo systemctl daemon-reload
sudo systemctl start iniciarNou.service
```

Una vez definida la implementación *software* de la Raspberry Pi es necesario explicar la implementación *software* del Arduino MKR 1310 WAN

Implementación *software* del Arduino MKR 1310 WAN

El Arduino MKR 1310 WAN es compatible con la tecnología LoRaWAN y consecuentemente con los servicios que ofrece The Things Network, estos servicios se emplearán en este trabajo fin de grado a modo de servicio receptor de peticiones, con la condición de que nuestro dispositivo físico solamente podrá enviar una petición cada 2 minutos según establece The Things Network.

Esta condición se contempla y cumple al establecer un limitador en la Raspberry Pi encargada del procesamiento de las peticiones, una vez procesada se envía al Arduino para que la traslade a The Things Network, el código de implementación *software* del Arduino se encuentra en la página siguiente (ver listing 19):

Nota: Los elementos "appKey" y "appEUI" se obtienen de The Things Network en el apartado Conexión con The Things Network

Listing 19: Código de configuración del Arduino MKR 1310 WAN

```
#include <MKRWAN.h>

LoRaModem modem;

// Cambiar en caso de cambio de aplicacion
String appEui = "Se obtiene en The Things Network";
String appKey = "Se obtiene en The Things Network";

void setup() {
  // Establece la conexion Serial:
  Serial.begin(9600);
  while (!Serial);
  // Cambiar en funcion de la region (eg. US915, AS923, ...)
  if (!modem.begin(EU868)) {
    while (1) {}
  };

  int connected = modem.joinOTAA(appEui, appKey);
  if (!connected) {
    while (1) {}
  }

  // Establece el intervalo de envio cada 60 segundos.
  modem.minPollInterval(60);
}

void loop() {

  int err;
  if (Serial.available() > 0)
  {
    String msg = Serial.readStringUntil('\n');
    modem.beginPacket();
    modem.print(msg);
    err = modem.endPacket(true);
  }

  // Si > 0 Se ha enviado correctamente
  delay(1000);
}
```

5.4. Implementación de las conexiones

La implementación de las conexiones es la parte final de la implementación, una vez se ha realizado la implementación tanto *hardware* como *software* de los distintos elementos que conforman el punto de asistencia inteligente es necesario garantizar que la información se transmite entre ellos de manera adecuada.

En lo referente a las conexiones hay tres que son de vital importancia, estas son:

- **Conexión con The Things Network**
- **Conexión con el bot de Telegram**
- **Conexión con la base de datos**

5.4.1. Conexión con The Things Network

The Things Network ofrece un servicio en línea gratuito compatible con LoRaWAN de almacenamiento y monitorización de mensajes, además de ser compatible con múltiples conexiones entre diferentes servicios de Internet de las cosas.

En este trabajo fin de grado cumplirá la función de servicio receptor, pues será el punto del que Node-red obtendrá las peticiones procesadas y las transformará en peticiones útiles para los usuarios gestores de estas.

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Para comenzar a utilizar The Things Network como servicio receptor solo hay que iniciar sesión en su web y se mostrará una vista similar a la siguiente (ver figura 49).

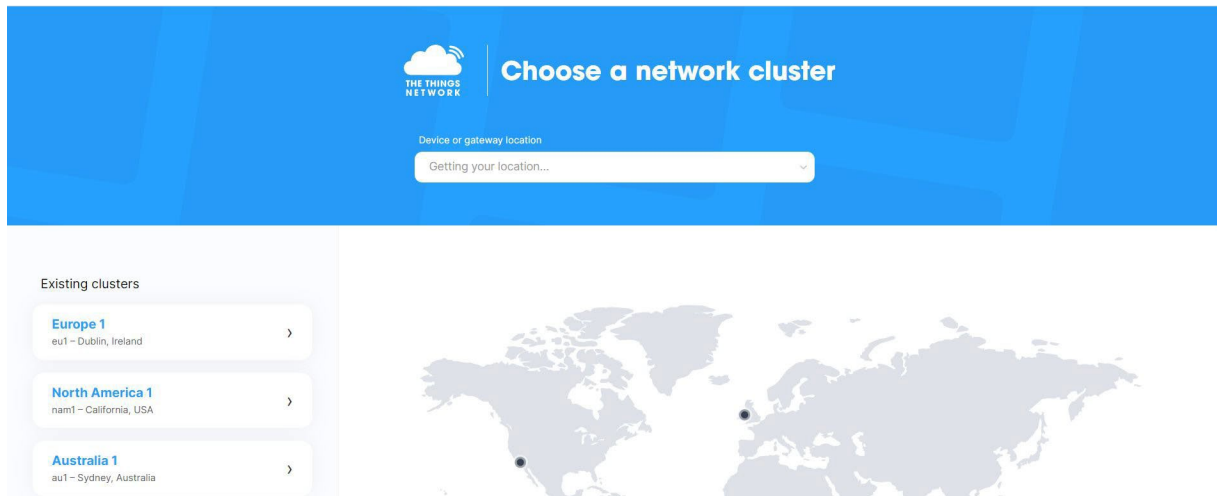


Figura 49: Vista inicial de The Things Network

En primer lugar, se selecciona la región de uso, en el caso de este trabajo fin de grado "Europe 1 eu1 - Dublin, Ireland", al hacerlo se mostrará la siguiente vista (ver figura 50).

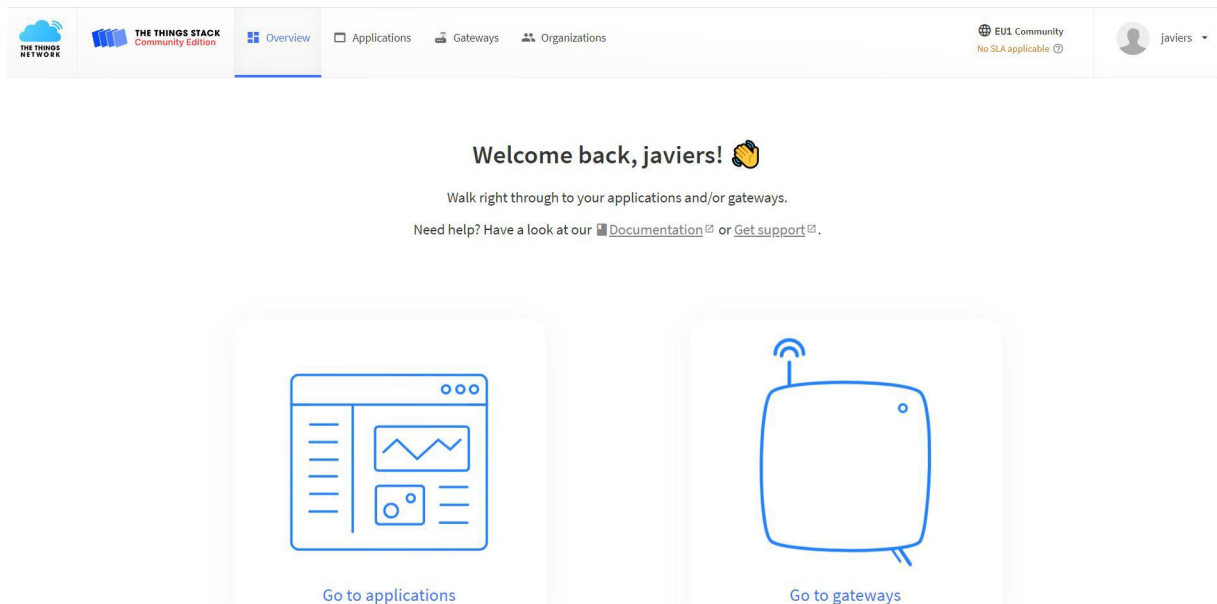
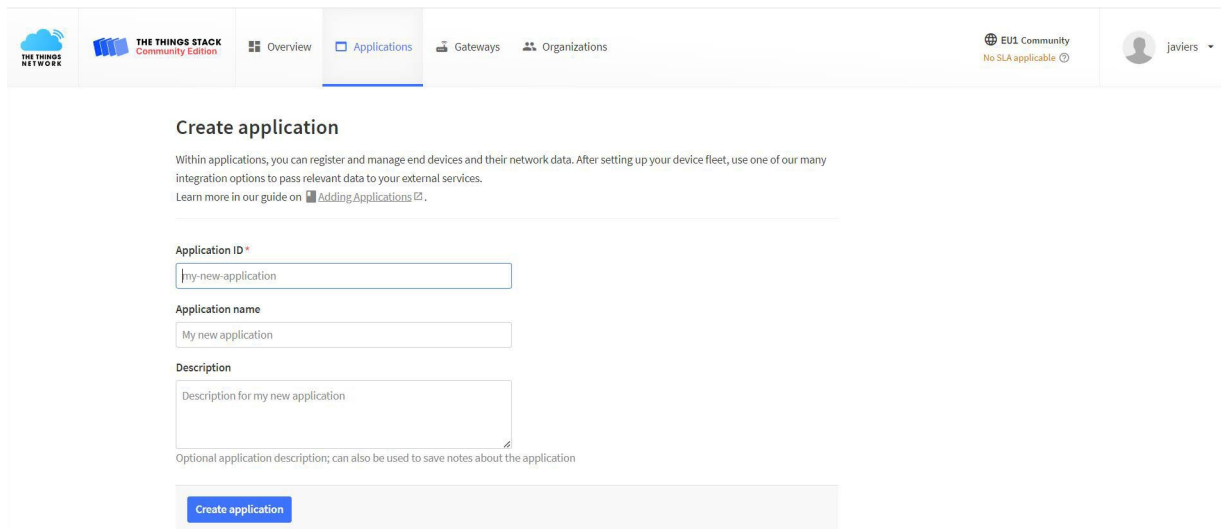


Figura 50: Vista de servicios de The Things Network

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

En segundo lugar, se selecciona "Go to applications" y en la vista siguiente "Create application", dando como resultado la siguiente vista (ver figura 51).



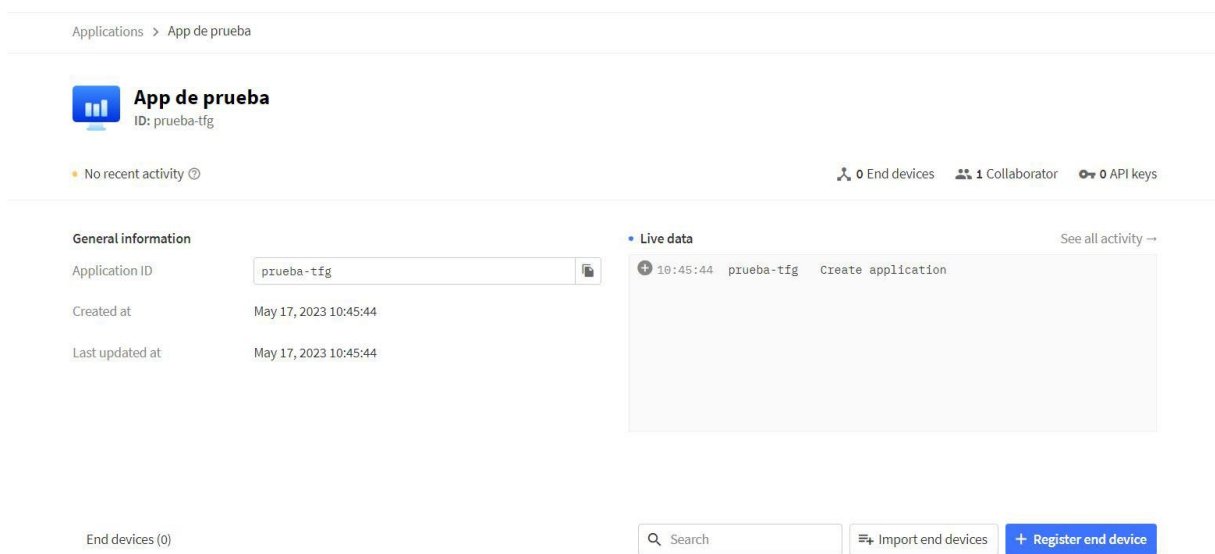
The screenshot shows the 'Create application' form in the The Things Network interface. The navigation bar at the top includes 'THE THINGS NETWORK', 'THE THINGS STACK Community Edition', 'Overview', 'Applications' (selected), 'Gateways', and 'Organizations'. On the right, it shows 'EU1 Community No SLA applicable' and a user profile for 'javier'. The main content area is titled 'Create application' and contains the following fields:

- Application ID ***: A text input field containing 'my-new-application'.
- Application name**: A text input field containing 'My new application'.
- Description**: A text area containing 'Description for my new application'. Below it, a note reads: 'Optional application description; can also be used to save notes about the application'.

At the bottom of the form is a blue button labeled 'Create application'.

Figura 51: Creación de una aplicación de The Things Network

Tras introducir los datos correspondientes se ha de seleccionar "Create application" y el resultado es una vista similar a esta (ver figura 52).



The screenshot shows the 'App de prueba' application view in the The Things Network interface. The breadcrumb navigation is 'Applications > App de prueba'. The application details are as follows:

- App de prueba** (ID: prueba-tfg)
- No recent activity**
- 0 End devices**, **1 Collaborator**, **0 API keys**

The 'General information' section displays:

- Application ID**: prueba-tfg
- Created at**: May 17, 2023 10:45:44
- Last updated at**: May 17, 2023 10:45:44

The 'Live data' section shows a single event: '10:45:44 prueba-tfg Create application'. Below this, there is a search bar and two buttons: 'Import end devices' and 'Register end device'.

Figura 52: Vista de una aplicación personal en The Things Network

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

El siguiente paso es el de vincular la aplicación con el dispositivo físico, para ello hay que seleccionar "+ Register end device" e introducir los datos correspondientes, en el caso de esta aplicación puede verse en la figura siguiente (ver figura 53).

End device type

Input method ⓘ

Select the end device in the LoRaWAN Device Repository


Enter end device specifics manually

End device brand ⓘ* Model ⓘ* Hardware Ver. ⓘ* Firmware Ver. ⓘ* Profile (Region)*

Arduino SA | v Arduino MKR WAN 1... | v 1.0 | v 1.2.3 | v EU_863_870 | v

Arduino MKR WAN 1310

LoRaWAN Specification 1.0.2, RP001 Regional Parameters 1.0.2, Over the air activation (OTAA), Class A



The Arduino MKR WAN 1310 is a development board that provides a practical and cost-effective solution to add LoRaWAN® connectivity for projects requiring long-range, low-power wireless communication. Sensors and actuators can be connected to the board through the analog, digital, UART, SPI, and I2C pins. The MKR WAN 1310 comes complete with an ATECC508 secure element, a battery charger, 2MByte SPI Flash, and power consumption as low as 104 uA.

[Product website](#) ↗

Frequency plan ⓘ*

Europe 863-870 MHz (SF9 for RX2 - recommended) | v

Provisioning information

JoinEUI ⓘ*

..... Confirm

Figura 53: Vista de configuración de un dispositivo en The Things Network

Nota: la información siguiente recogida en "Provisioning information" es confidencial y única de cada aplicación, los campos a rellenar son los siguientes:

- **JoinEUI:** es un valor identificativo de la aplicación para TTN. Puede ser introducido de manera aleatoria.
- **DevEUI** es el valor identificativo del dispositivo Arduino MKR 1310 WAN, en el **Anexo I. Obtención del DevEUI** se explica como obtenerlo.
- **AppKey** es la clave de encriptación del dispositivo Arduino MKR 1310 WAN que empleará The Things Network para el envío de mensajes y la recepción. **Es importante almacenarla y tenerla a mano.**

Tras haber completado la información nuestro dispositivo aparecerá configurado en la vista anterior, el último paso es acceder al apartado "Payload formatters" e introducir el siguiente código (ver listing 20):

Listing 20: Código del decoder de The Things Network

```
function Decoder(bytes, port) {  
  var result = "";  
  for (var i = 0; i < bytes.length; i++) {  
    result += String.fromCharCode(parseInt(bytes[i]));  
  }  
  return { payload: result, };  
}
```

Esto dará como resultado la siguiente vista (ver figura 54):

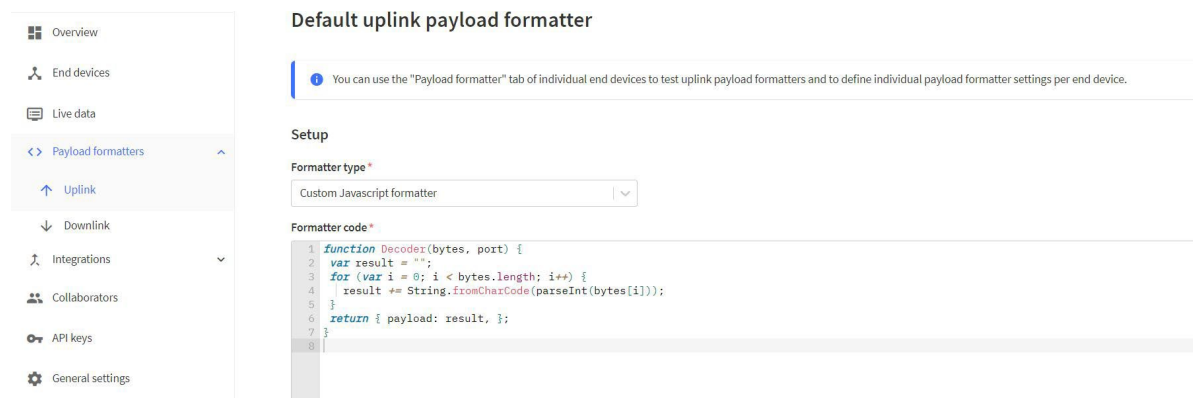


Figura 54: Vista de configuración del *payload* en The Things Network

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Finalmente, en el apartado "Integrations" hay que seleccionar "MQTT", y se mostrará una vista similar a la siguiente (ver figura 55), esto se empleará en el apartado de Node-red para la interconexión entre ambos servicios.

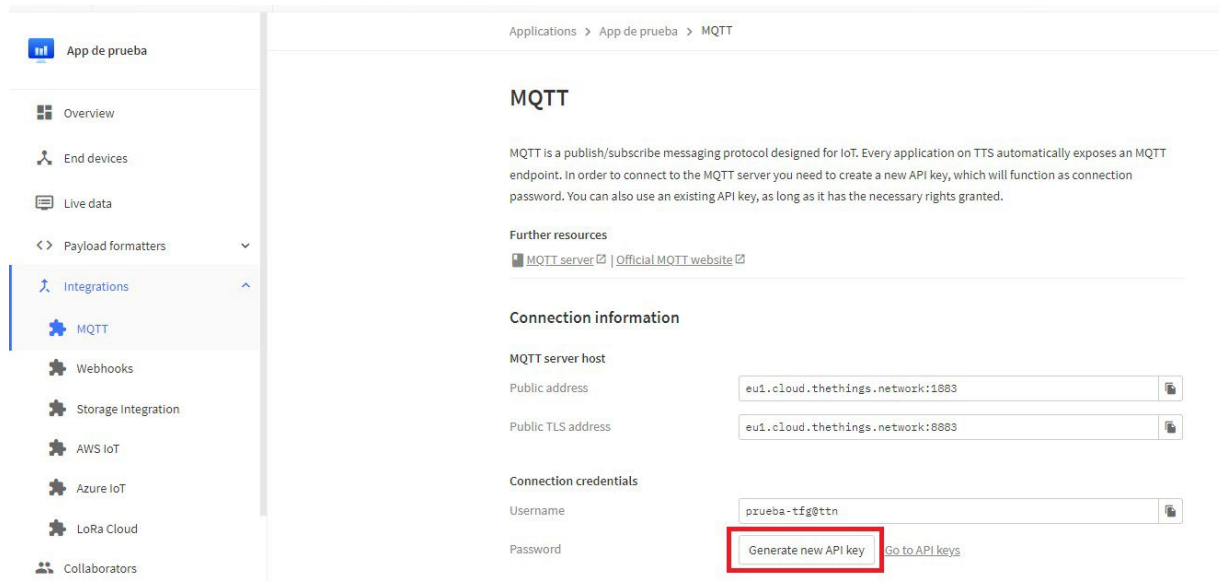


Figura 55: Vista de configuración de la API en The Things Network

Al presionar el botón "Generate new API key" se mostrará el valor de la API key oculto con asteriscos, tras seleccionar el botón con un ojo se mostrará, **este valor es importante tenerlo recogido para su posterior uso.**

5.4.2. Conexión con Node-red

Node-red permite la interconexión de los componentes del servidor con el servicio receptor, además es una herramienta del ecosistema Node.js [60], por lo tanto, requiere instalar Node y una vez instalado instalar el paquete Node-red, para ello en función del sistema operativo se instala con los siguientes comandos en los respectivos terminales (ver listing 21):

Listing 21: Código para instalar Node-red

```
# En Linux
$ sudo npm install -g --unsafe-perm node-red

# En Windows
> npm install -g --unsafe-perm node-red
```

Y de nuevo en el terminal, hay que ejecutar el siguiente código para iniciar Node-red (ver listing 22):

Listing 22: Código para iniciar Node-red

```
# En Linux
$ node-red

# En Windows
> node-red
```

Una vez inicializado se mostrará una vista similar a la siguiente (ver figura 56), acto seguido para comenzar a trabajar hay que acceder a la dirección IP que indique.

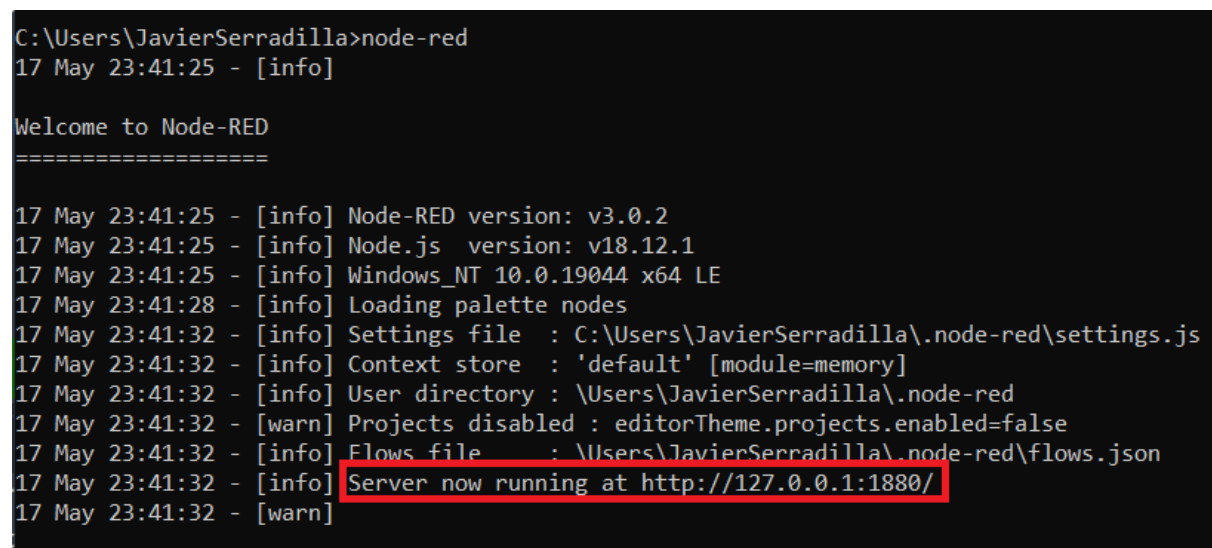


Figura 56: Vista de inicialización de Node-red

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Una vez accedida a la dirección anteriormente mencionada, en el apartado "network" se encuentra el primer bloque a trabajar "mqtt in", que debe configurarse como se muestra en la siguiente figura (ver figura 57). MQTT es un protocolo de comunicación entre dispositivos [61], en este trabajo fin de grado se emplea para la comunicación entre Node-red y The Things Network.

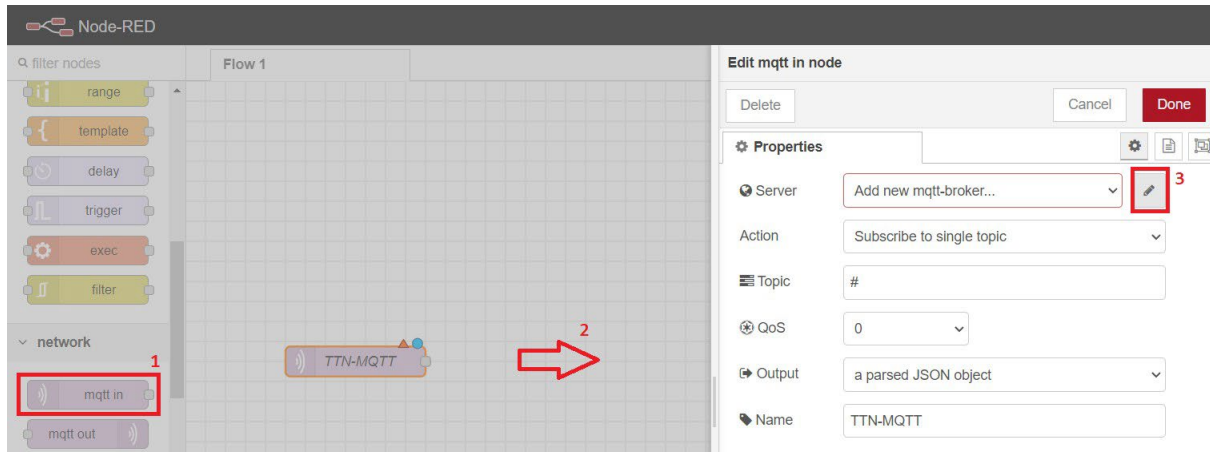


Figura 57: Configuración de MQTT en Node-red I

En el apartado de "Add new mqtt-broker" hay que introducir la información presente la siguiente figura (ver figura 58).

The image shows the configuration interface for an MQTT broker in Node-RED. The title bar reads "Edit mqtt in node > Edit mqtt-broker node". At the top, there are three buttons: "Delete", "Cancel", and "Update". Below this is a "Properties" section with a gear icon and a document icon. The "Name" field is set to "TTN-mqtt". There are three tabs: "Connection", "Security", and "Messages". Under the "Connection" tab, the "Server" field is "eu1.cloud.thethings.network" and the "Port" is "1883". There are two checkboxes: "Connect automatically" (checked) and "Use TLS" (unchecked). The "Protocol" dropdown is set to "MQTT V3.1.1". The "Client ID" field is "Leave blank for auto generated". The "Keep Alive" field is "60". Under the "Session" section, there is a checkbox "Use clean session" which is checked.

Figura 58: Configuración de MQTT en Node-red II

Finalmente, en el apartado "Security" hay que introducir la información correspondiente que se indica en la figura siguiente (ver figura 59).

The image shows a screenshot of the Node-RED interface for editing an MQTT broker node. The title bar reads "Edit mqtt in node > Edit mqtt-broker node". At the top, there are three buttons: "Delete", "Cancel", and "Update". Below this is a "Properties" section with a gear icon and a document icon. The "Name" field contains "TTN-mqtt". There are three tabs: "Connection", "Security", and "Messages". The "Security" tab is active. Under "Security", there are two fields: "Username" with the value "Tu Application ID" and "Password" with a masked value ".....". A red rectangular box highlights the "Password" field, and a red arrow points upwards from the text "Tu API Key" below it to the password field.

Figura 59: Configuración de MQTT en Node-red III

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Tras haber completado este paso, Node-red estaría vinculado a la aplicación de The Things Network y la conexión se habría realizado de forma satisfactoria, indicada en el elemento "mqtt in" con un *connected*, para poder observar la información recibida desde The Things Network habría que añadir 2 elementos de tipo "debug" situado en el apartado "common" y configurarlos como se muestra en la figura siguiente (ver figura 60).

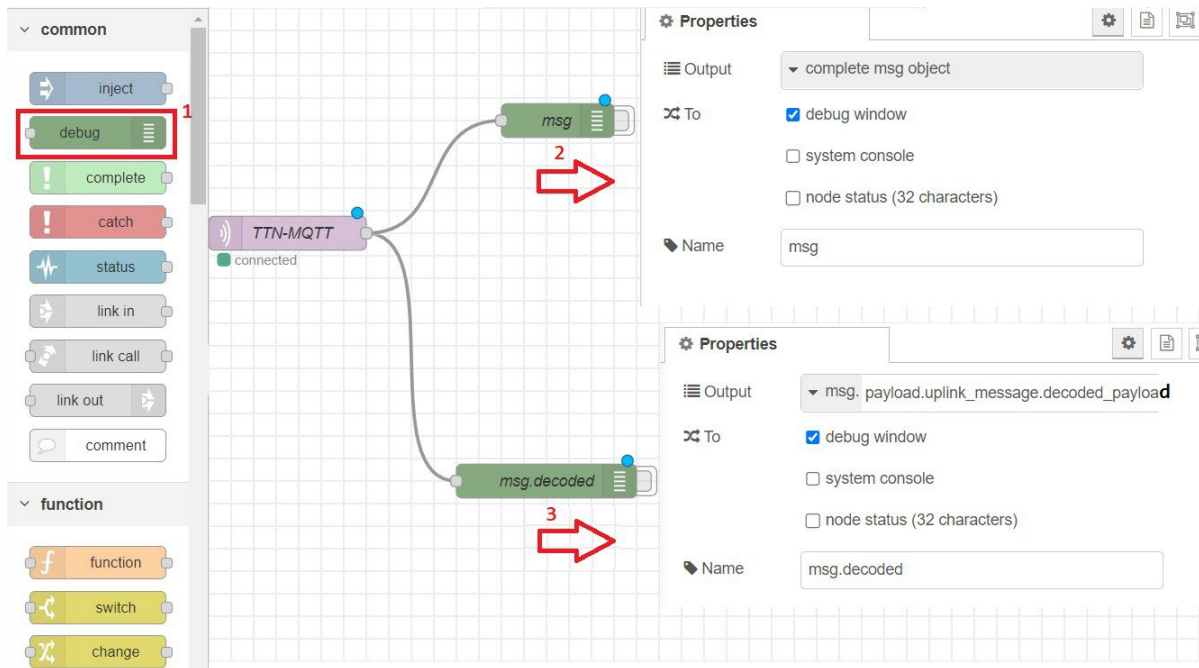


Figura 60: Recepción de mensajes en Node-red

5.4.3. Conexión con la base de datos

Tras haber creado la base de datos en apartados anteriores, Node-red permite la conexión con esta mediante los elementos que se encuentran en el paquete "node-red-node-mysql", para instalar este paquete hay que introducir en la consola el siguiente comando (ver listing 23):

Listing 23: Código para instalar el paquete MySQL en Node-red

```
# En Linux
$ npm i node-red-node-mysql

# En Windows
> npm i node-red-node-mysql
```

Tras instalarlo y reiniciar Node-red aparecerá el apartado "storage" que contendrá el elemento "mysql", este ha de ser configurado como se indica en las figuras siguientes (ver figura 61) y (ver figura 62).

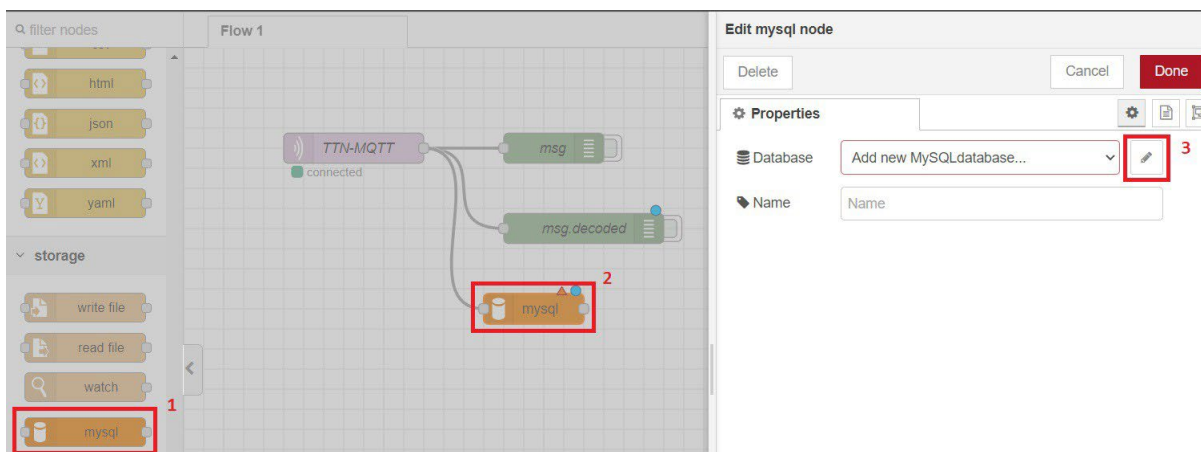


Figura 61: Configuración del elemento Mysql en Node-red I

Edit mysql node > **Add new MySQLdatabase config node** 7

Properties ⚙️ 📄

🌐 Host	<input type="text" value="La IP de la base de datos"/>	1
🔄 Port	<input type="text" value="el puerto en el que esté la base de datos"/>	2
👤 User	<input type="text" value="El username de tu base de datos"/>	3
🔒 Password	<input type="text" value="La password de la base de datos"/>	4
🗄 Database	<input type="text" value="El nombre de tu base de datos"/>	5
🕒 Timezone	<input type="text" value="±hh:mm"/>	
🗉 Charset	<input type="text" value="UTF8"/>	
📁 Name	<input type="text" value="El nombre que le das al nodo"/>	6

Tip: The timezone should be specified as ±hh:mm or leave blank for 'local'.

Figura 62: Configuración del elemento Mysql en Node-red II

Tras haber realizado los pasos anteriores, la conexión con la base de datos de MySQL quedaría completada con éxito, pero las bases de datos se emplean para almacenar y recolectar información, entre otras muchas funciones, por lo tanto, es necesario definir dos elementos de tipo "function" que se encuentran en el apartado con el mismo nombre, una vez generados los dos elementos es necesario codificarlos basándonos en los siguientes códigos (ver listing 24) y (ver listing 25):

Listing 24: Código del elemento Insercion en Node-red

```
var tzoffset = (new Date()).getTimezoneOffset() * 60000; //offset in milis

var fechaPeticion = (new Date(Date.now() - tzoffset)).toISOString().slice(0,
-1);

//Lo devuelve en JSON
var tipoPeticion = msg.payload.uplink_message.decoded_payload["payload"];

var dispositivoPeticion = msg.payload.end_device_ids["dev_eui"];

var prioridadPeticion;

var tipoPeticionMensaje;

switch (tipoPeticion) {
  case "1":
    tipoPeticionMensaje = "fuego";
    prioridadPeticion = 3;
    break;
  case "2":
    tipoPeticionMensaje = "socorro";
    prioridadPeticion = 3;
    break;
  case "3":
    tipoPeticionMensaje = "bedel mando";
    prioridadPeticion = 1;
    break;
  case "4":
    tipoPeticionMensaje = "bedel llave persiana";
    prioridadPeticion = 1;
    break;
  case "5":
    tipoPeticionMensaje = "asistencia del bedel";
    prioridadPeticion = 2;
    break;
  case "6":
    tipoPeticionMensaje = "emergencia";
```

```
        prioridadPeticon = 3;
        break;
    default:
        tipoPeticonMensaje = "error en dispositivo";
        prioridadPeticon = 1;
    }

    //msg.topic = "Fecha = " + fechaPeticon + ", TipoPeticon = " +
    tipoPeticon + ", DispositivoPeticon = " + dispositivoPeticon + ",
    PrioridadPeticon = " + prioridadPeticon;

    msg.topic = "INSERT INTO peticon (id_dispositivo, fecha_peticon,
    tipo_peticon, estado, prioridad, id_gestor) SELECT
    dispositivo.id_dispositivo, CURDATE(), '" + tipoPeticonMensaje + "', 0,
    " + prioridadPeticon + ", null FROM dispositivo WHERE
    dispositivo.nombre_dispositivo = '" + dispositivoPeticon + "';"

    return msg;
```

Listing 25: Código del elemento Consultar Aula en Node-red

```
var dispositivoPeticon = msg.payload.end_device_ids["dev_eui"];
dispositivoPeticon = "" + dispositivoPeticon + "";
msg.topic = "select dispositivo.localizacion from dispositivo where
dispositivo.nombre_dispositivo = "+ dispositivoPeticon + ";";
msg.leido = true;
return msg;
```

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Tras haber implementado estos elementos, la vista de Node-red ha de ser similar a la siguiente (ver figura 63).

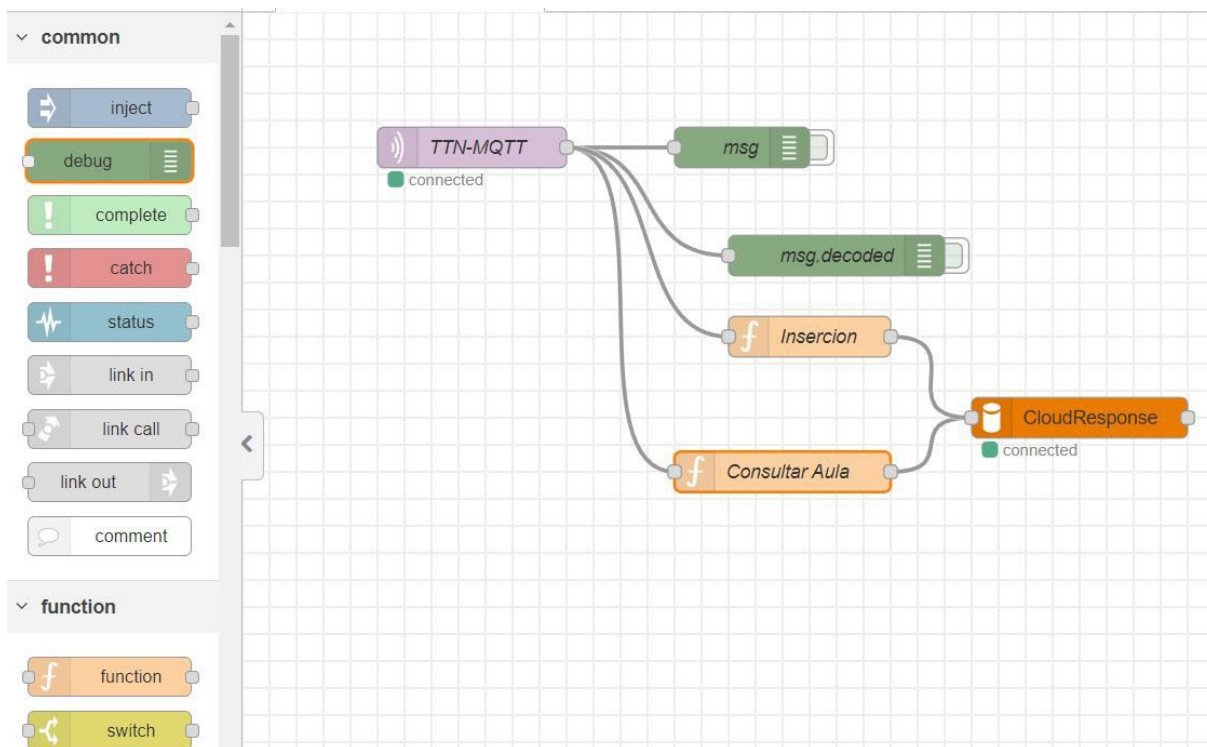


Figura 63: Vista de Node-red con los elementos de MySQL

Con el objetivo de facilitar la conexión con el bot de Telegram que se explicará en el siguiente subapartado, se han de añadir dos elementos adicionales a la anterior vista de Node-red, siendo estos de tipo "function" y "switch" y encontrados en la sección "function", el elemento "switch" ha de ser configurado como en la siguiente figura (ver figura 64).

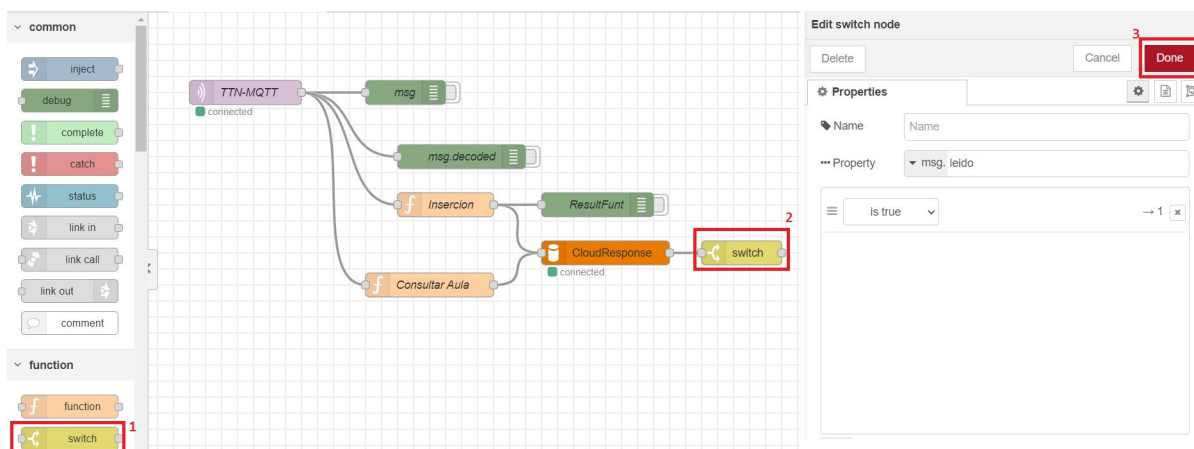


Figura 64: Configuración del elemento switch en Node-red

Y el elemento "function" con el código siguiente (ver listing 26):

Listing 26: Código del elemento Volcar Aula en Node-red

```
|| flow.set("localizacionDispositivo",msg.payload[0].localizacion);  
|| return msg;
```

Nota: el elemento de tipo "debug" con nombre "ResultFunt" es un elemento no trascendental que se emplea para la comprobación de mensajes, similar a "msg" y "msg.decoded", el tipo de *Output* es "msg.topic".

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

Tras haber realizado los pasos anteriores, la vista resultante ha de ser similar a la siguiente (ver figura 65).

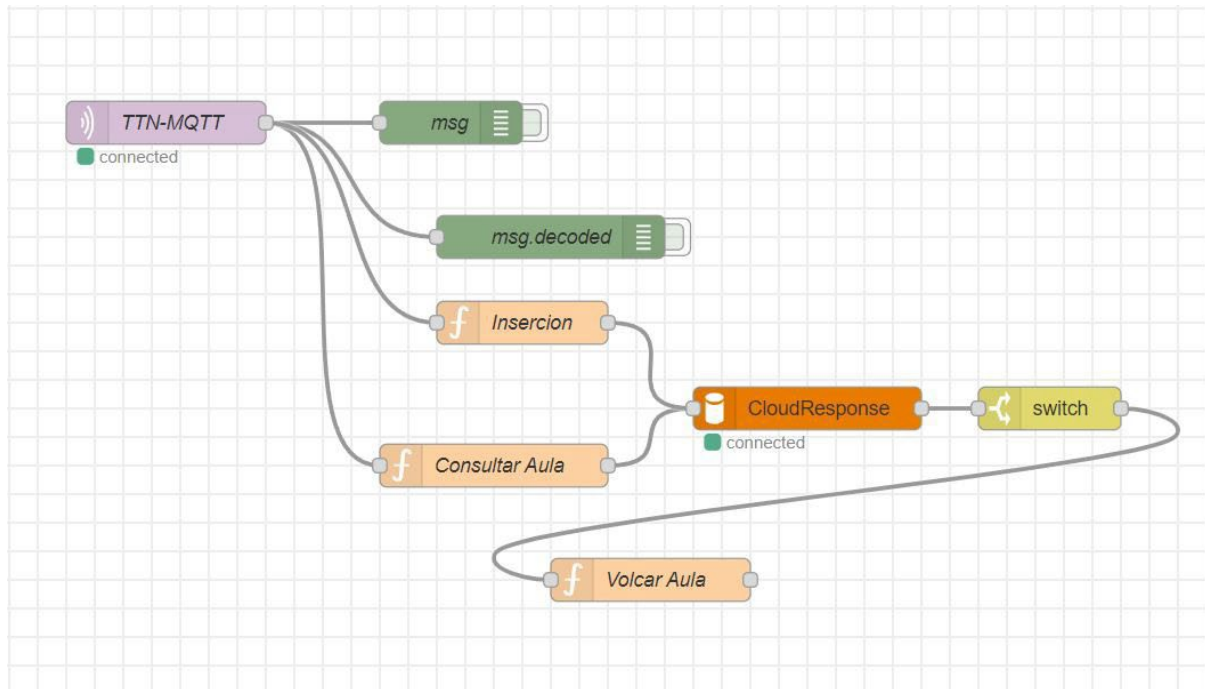


Figura 65: Vista final del elemento MySQL con Node-red

5.4.4. Conexión con el bot de Telegram

La conexión con el bot de Telegram es la última de las conexiones que se realizan a través de Node-red, esta conexión es la encargada de transmitir al bot de Telegram la información que debe comunicar al usuario gestor.

Partiendo de la anterior implementación, para añadir a Node-red los elementos necesarios de Telegram se requiere instalar dos paquetes de elementos, siendo estos "node-red-contrib-telegrambot" y "node-red-contrib-telegrambot-home" para ello es necesario ejecutar los siguientes códigos en terminal (ver listing 27):

Listing 27: Código para instalar los paquetes del bot de Telegram en Node-red

```
# En Linux
$ npm install node-red-contrib-telegrambot
$ npm install node-red-contrib-telegrambot-home
$ sudo service nodered restart

# En Windows
> npm install node-red-contrib-telegrambot
> npm install node-red-contrib-telegrambot-home
```

Tras haber instalado los nuevos paquetes aparecerá un apartado nuevo llamado "telegram" hay que seleccionar el elemento "sender" y configurarlo basándonos en las siguientes figuras (ver figura 66) y (ver figura 67).

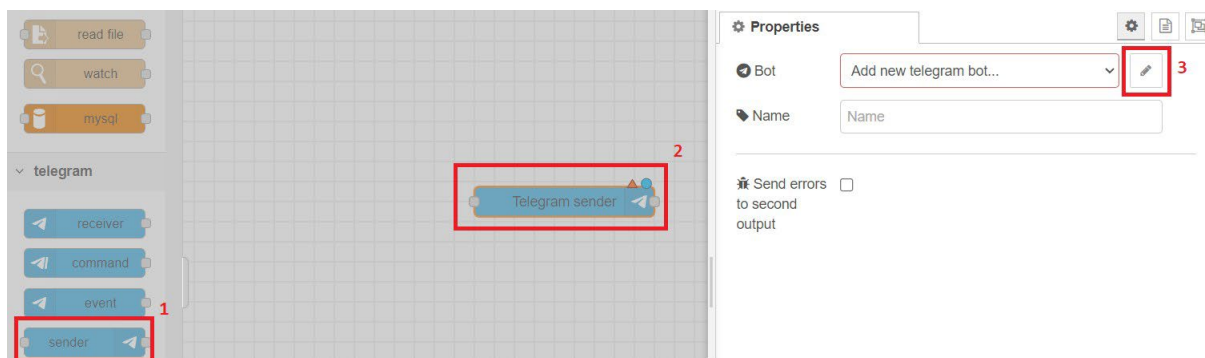


Figura 66: Configuración del elemento Telegram Sender con Node-red I

The screenshot displays the configuration window for a Telegram Sender node in Node-red II. At the top right, there are 'Cancel' and 'Add' buttons, with a red '3' next to the 'Add' button. The 'Properties' section contains the following fields:

- Bot-Name:** A text input field containing 'Nombre del bot al que se conectará', highlighted with a red box and labeled '1'.
- Token:** A text input field containing 'Clave del bot al que se conectará', highlighted with a red box and labeled '2'.

A yellow tip box below the Token field reads: 'Tip: If you don't have a token yet, you can create a new one here: @BotFather.' Below the properties section, there are four more fields:

- Users:** A text input field with placeholder text '(Optional list of authorized user names e.g.: hugo,sepp,egon)'.
- ChatIds:** A text input field with placeholder text '(Optional list of authorized chat-ids e.g.: -1234567,2345678,-3456789)'.
- Server URL:** A text input field with placeholder text '(Optional URL for proxying and testing e.g.: https://api.telegram.org)'.
- Update Mode:** A dropdown menu currently set to 'Polling'.

Figura 67: Configuración del elemento Telegram Sender con Node-red II

Nota: los valores para los campos "Bot-Name" y "Token" se explican en el **Anexo II. Creación y configuración de un bot de Telegram**

Una vez terminado este proceso la conexión con el bot de Telegram estará activa, pero faltará un elemento que le transmita la información a enviar al bot, para ello es necesario generar un elemento "function" con la codificación de la página siguiente (ver listing 28):

Listing 28: Código del elemento Notificar en Node-red

```
var tipoPeticon = msg.payload.uplink_message.decoded_payload["payload"];
var tipoPeticonMensaje = "default"
switch (tipoPeticon) {
  case "1":
    tipoPeticonMensaje = "Incendio";
    break;
  case "2":
    tipoPeticonMensaje = "Auxilio";
    break;
  case "3":
    tipoPeticonMensaje = "Mando del proyector";
    break;
  case "4":
    tipoPeticonMensaje = "Llave de persiana";
    break;
  case "5":
    tipoPeticonMensaje = "Asistencia del conserje";
    break;
  case "6":
    tipoPeticonMensaje = "Emergencia Sanitaria";
    break;
  default:
    tipoPeticonMensaje = "Error en dispositivo";
}

var localizacionPeticon = flow.get('localizacionDispositivo');

var message = 'Ha llegado una nueva Peticon:\r\n' +
  'Peticon de ' + tipoPeticonMensaje + "\r\n En el Aula: " +
  localizacionPeticon;

//Configuracion del Payload
msg.payload = {chatId: -Tu_id_de_chat , type : 'message' , content :
  message};

//Configuracion de las Opciones
msg.payload.options = {disable_web_page_preview: true, parse_mode :
  "Markdown"};

return msg;
```

Nota: el valor de "chatId" se explica en el **Anexo II. Creación y configuración de un bot de Telegram.**

Finalmente, ha de quedar una vista similar a la siguiente figura que reflejará la implementación completa con Node-red (ver figura 68).

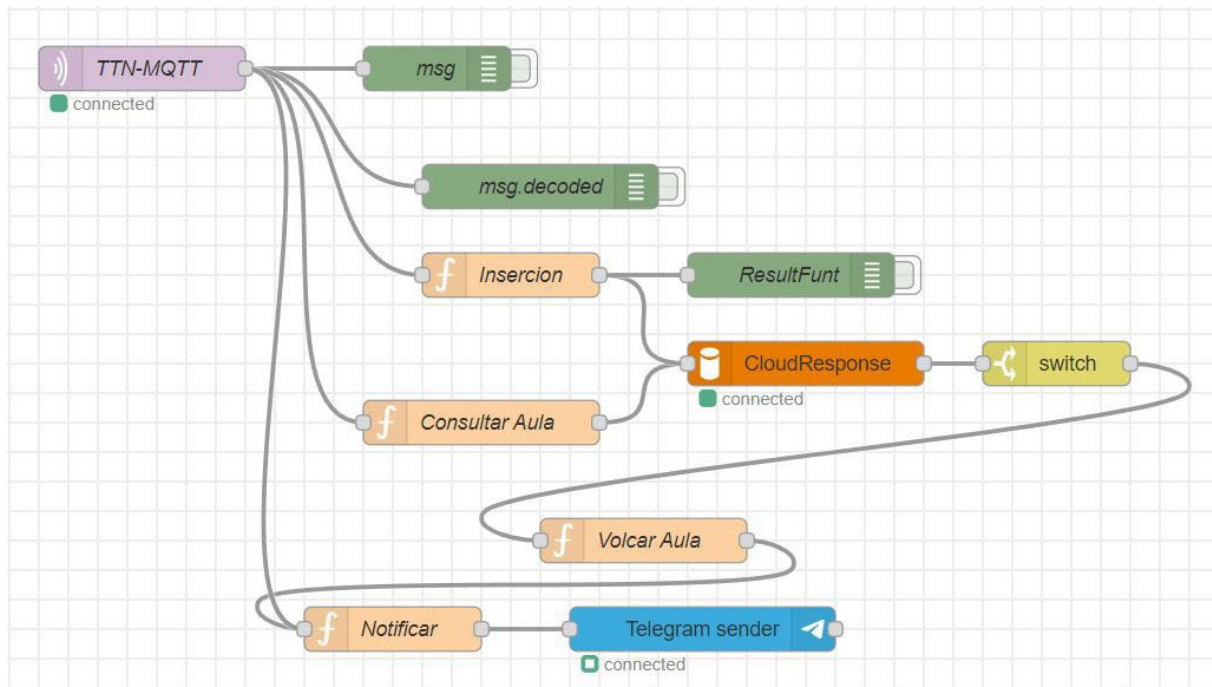


Figura 68: Vista completa de la implementación de Node-red

Tras haber implementado las conexiones, el trabajo fin de grado estaría plenamente operativo y funcional, cumpliendo con las especificaciones recogidas en las distintas etapas anteriores y con una implementación revisada y segura.

6. Conclusiones y trabajos futuros

Este último capítulo tiene por objetivo realizar una comparativa entre los objetivos propuestos (ver Objetivos del trabajo fin de grado) y los resultados obtenidos tras haber completado las distintas etapas del trabajo fin de grado.

Finalmente, se comentarán posibles mejoras que pueden incluirse en el trabajo fin de grado, así como diferentes rumbos que podrían tomarse partiendo de la base realizada en este trabajo fin de grado.

6.1. Conclusiones del trabajo fin de grado

El objetivo principal que se proponía al comienzo de este trabajo fin de grado era el de construir un dispositivo de asistencia inteligente, capaz de reconocer patrones de voz concretos del usuario y transformar esos patrones, en peticiones útiles para su posterior atención, haciendo uso de LoRaWAN como alternativa a Internet para el envío de estas. Además, se ha planteado un análisis de viabilidad energética de este producto, con el objetivo de mostrar la viabilidad de este producto y los servicios que ofrece en distintas situaciones y usos.

El producto resultante, además de cumplir con todo lo anteriormente mencionado, ha cumplido con todos los objetivos específicos, siendo estos el uso de redes neuronales para mejorar la interacción persona-dispositivo y la posibilidad de reentrenamiento de la propia red neuronal para adaptarse a nuevas situaciones y mejorar constantemente la interacción con el usuario, cabe destacar que la implementación de las tecnologías ideadas en las primeras fases ha podido llevarse a cabo de manera correcta y útil.

Rasa ha resultado ser una herramienta perfectamente funcional con el trabajo fin de grado y ha brindado la posibilidad tanto de creación como de reentrenamiento de los modelos basados en redes neuronales, ofreciendo una capa intermedia de servicio y permitiendo su modificación a un nivel más alto que si se hubiera tenido que programar y reentrenar manualmente dicha red.

En lo referente a lo anteriormente citado, Spring Boot ha facilitado en gran medida la creación y modificación de la aplicación web, así como aplicaciones compatibles a Spring Boot como Thymeleaf, estas aplicaciones han resultado en un conjunto de herramientas útiles para el trabajo en la aplicación web, logrando que se pudiera realizar en el plazo de 45 horas propuesto durante la planificación.

Por un lado, Node-red ha cumplido con las especificaciones iniciales, esta aplicación *software* ha sido capaz de mantener el entramado de conexiones necesario para que una vez las peticiones procesadas fueran almacenadas por el servicio receptor. Además, gracias a la gran

variedad de paquetes creados por la comunidad, las posibilidades de integración con distintas tecnologías como han sido las base de datos MySQL o los bots de Telegram han sido muy elevadas y si sigue el mismo rumbo que lleva actualmente puede ser una alternativa muy interesante a la programación manual de todos los elementos de una red de Internet de las cosas.

Por otro lado, también es necesario destacar las posibilidades que han brindado tanto la Raspberry Pi como el Arduino MKR 1310 WAN, si bien es cierto que la Raspberry Pi no ofrece un hardware específicamente creado para el uso de redes neuronales, ha quedado demostrado en este trabajo fin de grado que es posible su tratamiento y funcionalidad en este tipo de dispositivos *hardware*. Aunque es notorio que sus tiempos de ejecución no han sido tan cortos como se podría esperar en otro tipo de dispositivos de asistencia. En lo respectivo al Arduino es importante destacar su gran flexibilidad, puesto que permite el paso de información entre dispositivos claramente diferenciados como son este y la Raspberry Pi, y su capacidad de envío de información a grandes distancias mediante el uso de la antena.

Finalmente, y atendiendo a los puntos definidos durante el análisis, la planificación y diseño, la implementación obtenida cumple con las especificaciones que habían sido fijadas en estos puntos anteriores, es por ello, que además de haber cumplido con dichas especificaciones y atendiendo al espacio temporal designado a cada una de esas etapas, el producto resultante ha sido desarrollado e implementado de manera satisfactoria.

6.2. Trabajos futuros

Tras haber sido desarrollado el producto final, es necesario tener en cuenta que el producto desarrollado en la etapa de implementación de este trabajo fin de grado no es un producto final para todas las demás posibilidades que ofrece, o dicho de otro modo, este producto no es una solución que se pueda estandarizar al resto de situaciones de uso.

Las posibilidades que brinda este trabajo fin de grado van más allá de la propuesta en la etapa de implementación, y además el *hardware* propuesto puede ser reemplazado basándose en distintas necesidades de implementación, como puede ser el coste o la disponibilidad de componentes. El mercado actual de componentes *hardware* está en constante cambio y evolución [62], a diario surgen nuevos productos y soluciones que permiten el progreso y la mejora de aplicaciones anteriormente descritas. Este trabajo fin de grado no es una excepción y está expuesto al constante cambio que vive la tecnología actual, es por ello que de cara a futuras modificaciones el *hardware* es uno de los puntos a tener en cuenta.

Otro punto a tener en cuenta sería la posibilidad de contar con servidores dedicados a la recolección de peticiones mediante LoRaWAN, esto aportaría independencia de servicios de terceros como podría ser The Things Network, así como mayor libertad y manejo de los datos recibidos.

Finalmente, la aplicación web es otro punto a tener en cuenta de cara a trabajos futuros y mejoras en el producto, la aplicación realizada en este trabajo fin de grado es muy específica y no muestra las posibilidades totales que ofrece la implementación de aplicaciones web en general, siendo posible la generación de aplicaciones telefónicas nativas a partir de la aplicación web desarrollada, como por ejemplo con tecnología React [63].

7. Bibliografía

Referencias:

- [1] Agencia EFE. *La criminalidad aumenta un 18 % en 2022 y las violaciones un 34 %*. url: <https://efe.com/espana/2023-03-17/balance-de-criminalidad-2022-aumentan-violaciones-un-34-por-ciento-y-delitos-un-18-por-ciento/> (visitado 26-04-2023).
- [2] Sabermás. *SMARTPHONE*. url: <https://www.sabermas.umich.mx/archivo/tecnologia/57-numero-745/116-smartphone.html> (visitado 27-04-2023).
- [3] Ramón López de Mántaras. *El futuro de la IA: hacia inteligencias artificiales realmente inteligentes | OpenMind*. url: <https://www.bbvaopenmind.com/articulos/el-futuro-de-la-ia-hacia-inteligencias-artificiales-realmente-inteligentes/> (visitado 27-04-2023).
- [4] Amazon Web Services. *¿Qué es LoRa WAN? - AWS IoT Core*. url: https://docs.aws.amazon.com/es_es/iot/latest/developerguide/connect-iot-lorawan-what-is-lorawan.html (visitado 27-04-2023).
- [5] Oratile Khutsoane, Bassey Isong y Adnan M. Abu-Mahfouz. *IoT devices and applications based on LoRa/LoRaWAN*. 2017. doi: 10.1109/IECON.2017.8217061.
- [6] Android Developers. *Intents y filtros de intents | Desarrolladores de Android | Android Developers*. url: <https://developer.android.com/guide/components/intents-filters?hl=es-419> (visitado 27-04-2023).
- [7] Ionos. *¿Qué es una neural network? - IONOS*. url: <https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-busqueda/que-es-una-neural-network/> (visitado 27-04-2023).
- [8] Becolve Digital. *Conceptos básicos que te ayudarán a entender LoRa y LoRaWAN en minutos*. url: <https://becolve.com/blog/conceptos-tecnicos-basicos-que-te-ayudaran-a-entender-lora-y-lorawan-low-power-wide-area-network-en-pocos-minutos/> (visitado 01-05-2023).
- [9] Amazon Developer. *Amazon Alexa Official Site: What is Alexa?* url: <https://developer.amazon.com/es-ES/alexa> (visitado 01-05-2023).
- [10] Google Home. *Dispositivos domésticos inteligentes que funcionan con Google | Google Home*. url: https://home.google.com/intl/es_es/what-is-google-home/ (visitado 01-05-2023).
- [11] Apple. *HomePod (2.ª generación) - Apple (ES)*. url: <https://www.apple.com/es/homepod-2nd-generation/> (visitado 02-06-2023).

- [12] hospitecnia - Jaime Punter y Eugeni Sedano. *Dispositivos de monitorización remota en atención sanitaria*. url: <https://hospitecnia.com/tecnologia/equipamiento-medico/dispositivos-monitorizacion-remota-atencion-sanitaria/> (visitado 15-05-2023).
- [13] Salvador Martínez-Cruz et al. «An Outdoor Navigation Assistance System for Visually Impaired People in Public Transportation». En: *IEEE Access* 9 (2021), pp. 130767-130777. doi: 10.1109/ACCESS.2021.3111544.
- [14] blogthinkbig - Moncho Terol. *LoRaWAN: ¿qué es, para qué sirve y dónde se utiliza? | Telefónica*. url: <https://blogthinkbig.com/lorawan-ventajas-usos-telefonica> (visitado 02-06-2023).
- [15] Lpwan. *Licencias LoRa*. url: <https://lpwan.es/tag/licencias/> (visitado 02-06-2023).
- [16] The Things Network - Eduardo Medina. *The Things Network? - Post - The Things Network*. url: <https://www.thethingsnetwork.org/community/barranquilla/post/the-things-network> (visitado 01-05-2023).
- [17] Capgemini. *Los consumidores prefieren interactuar con asistentes virtuales de voz y texto antes que con humanos, lo que crea oportunidades de negocio - Capgemini Spain*. url: <https://www.capgemini.com/es-es/noticias/notas-de-prensa/los-consumidores-prefieren-interactuar-con-asistentes-virtuales-de-voz-y-texto-antes-que-con-humanos-lo-que-crea-oportunidades-de-negocio/> (visitado 03-05-2023).
- [18] KANLLI - Jonathan Liege y KANLLI - Elena Lostalé. «LA ERA DE LA VOZ: ASISTENTES VIRTUALES Y VOICE MARKETING». En: (2018).
- [19] Profesionalreview - José Antonio Castillo. *Puerto serie – Qué es, para qué sirve y tipos*. url: <https://www.profesionalreview.com/2020/03/07/puerto-serie-que-es-para-que-sirve-y-tipos/> (visitado 03-05-2023).
- [20] Rasa. *NLU Training Data*. url: <https://rasa.com/docs/rasa/nlu-training-data/> (visitado 03-05-2023).
- [21] CNX Software. *La escasez de Raspberry Pi explicada - CNX Software - Noticias de Sistemas Embebidos*. url: <https://www.cnx-software.es/2022/04/05/la-escasez-de-raspberry-pi-explicada/> (visitado 03-05-2023).
- [22] adslzone.net - Rocío GR. *Placas potentes alternativas a la Raspberry Pi 4: mejores opciones*. url: <https://www.adslzone.net/listas/gadgets/alternativas-raspberry-pi/> (visitado 02-06-2023).
- [23] Sinelec. *¿Qué es Node-RED y para qué sirve? | Grupo Sinelec*. url: <https://blog.gruposinelec.com/actualidad/que-es-node-red-y-para-que-sirve/> (visitado 01-05-2023).
- [24] MySQL Developer. *MySQL :: MySQL 8.0 Reference Manual*. url: <https://dev.mysql.com/doc/refman/8.0/en/> (visitado 01-05-2023).

- [25] Xataka - Yúbal Fernández. *Bots de Telegram: qué son, cómo funcionan y 17 recomendados para empezar*. url: <https://www.xataka.com/basics/bots-telegram-que-como-funcionan-recomendados-para-empezar> (visitado 01-05-2023).
- [26] IBM. *¿Qué es Java Spring Boot? | IBM*. url: <https://www.ibm.com/mx-es/topics/java-spring-boot> (visitado 09-05-2023).
- [27] Telegram. *Preguntas frecuentes*. url: <https://telegram.org/faq/es#p-que-es-telegram-que-puedo-hacer-aqui> (visitado 01-05-2023).
- [28] Amazon Web Services. *¿Qué es una aplicación web? - Explicación de las aplicaciones web - AWS*. url: <https://aws.amazon.com/es/what-is/web-application/> (visitado 01-05-2023).
- [29] softwarezulemakeydi.blogspot. *Ingeniería de software: viabilidad y factibilidad*. url: <http://softwarezulemakeydi.blogspot.com/2013/05/viabilidad-y-factibilidad.html> (visitado 03-05-2023).
- [30] Emprendedores - redacción. *¿Es viable mi proyecto de negocio? - Emprendedores*. url: <http://www.emprendedores.es/crear-una-empresa/viabilidad-negocio/> (visitado 03-05-2023).
- [31] Bugeados. *Lo que debes saber sobre la Raspberry Pi 4 antes de lanzarte a comprar una | Computer Hoy*. url: <https://bugeados.com/raspberry/cual-es-el-consumo-de-una-raspberry-pi-3-4/> (visitado 30-05-2023).
- [32] Arduino.cl. *Arduino MKR WAN 1310 | Arduino.cl - Compra tu Arduino en Línea*. url: <https://arduino.cl/producto/arduino-mkr-wan-1310/> (visitado 30-05-2023).
- [33] Selectra. *Consulta el precio de la luz hoy: coste de la electricidad por horas*. url: <https://selectra.es/energia/info/que-es/precio-kwh> (visitado 30-05-2023).
- [34] Ionos. *Estructura de desglose de trabajo (EDT) | Ejemplo y plantilla - IONOS*. url: <https://www.ionos.es/startupguide/productividad/estructura-de-desglose-de-trabajo/> (visitado 04-05-2023).
- [35] Mundoerp - Sergio Martínez. *Metodología iterativa o incremental en la gestión de proyectos*. url: <https://www.mundoerp.com/blog/metodologia-iterativa-o-incremental-gestion-proyectos/> (visitado 04-05-2023).
- [36] Agencia Tributaria. *Agencia Tributaria: Asignaciones para gastos de locomoción*. url: https://sede.agenciatributaria.gob.es/Sede/en_gb/ayuda/manuales-videos-folletos/manuales-ayuda-presentacion/irpf-2020/7-cumplimentacion-irpf/7_1-rendimientos-trabajo-personal/7_1_1-rendimientos-integros/7_1_1_2-dietas-gastos-viaje/asignaciones-gastos-locomocion.html (visitado 31-05-2023).
- [37] INE. *INEbase / Servicios / Hostelería y turismo / Hoteles: encuesta de ocupación, índice de precios e indicadores de rentabilidad / Últimos datos*. url: https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736177015&menu=ultiDatos&idp=1254735576863 (visitado 31-05-2023).

- [38] Glassdoor. *Sueldo: Ingeniero Informático en España en 2023* | Glassdoor. url: https://www.glassdoor.es/Sueldos/ingeniero-inform%C3%A1tico-sueldo-SRCH_KO0,21.htm (visitado 31-05-2023).
- [39] Lucidchart. *¿Qué es el lenguaje unificado de modelado (UML)?* | Lucidchart. url: <https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml> (visitado 08-05-2023).
- [40] Lucidchart. *Tutorial de diagrama de secuencia UML* | Lucidchart. url: <https://www.lucidchart.com/pages/es/diagrama-de-secuencia> (visitado 08-05-2023).
- [41] IBM. *Programación orientada a objetos - Documentación de IBM*. url: <https://www.ibm.com/docs/es/spss-modeler/saas?topic=language-object-oriented-programming> (visitado 09-05-2023).
- [42] MySQL. *MySQL :: MySQL Connectors*. url: <https://www.mysql.com/products/connector/> (visitado 09-05-2023).
- [43] estrada web group - Administrador. *Mensajes de notificación profesionales al usuario con jQuery y SweetAlert* | Estrada Web Group. url: <https://estradawebgroup.com/Post/Mensajes-de-notificacion-profesionales-al-usuario-con-jQuery-y-SweetAlert/4252> (visitado 09-05-2023).
- [44] Boxes.py. *Gallery - Boxes.py*. url: <https://www.festi.info/boxes.py/> (visitado 10-05-2023).
- [45] Autodesk - AutoCAD. *Autodesk AutoCAD 2024 | Obtener precios y suscribirse al software AutoCAD*. url: <https://www.autodesk.es/products/autocad/overview?term=1-YEAR&tab=subscription> (visitado 10-05-2023).
- [46] Apen. *Hardware - Apen Informática*. url: <https://apen.es/glosario-de-informatica/hardware/> (visitado 10-05-2023).
- [47] Santander Universidades. *¿Qué es Python?* | Blog Becas Santander. url: <https://www.becas-santander.com/es/blog/python-que-es.html> (visitado 10-05-2023).
- [48] ipglobal. *¿Cómo entiende un Bot? NLU en Rasa - Ipglobal : Tech Hub*. url: <https://www.ipglobal.es/como-entiende-un-bot-nlu-en-rasa/> (visitado 10-05-2023).
- [49] sinologic. *Probamos Vosk: un ASR gratuito, libre y que no necesita Internet* | Sinologic. url: <https://www.sinologic.net/2021-03/probamos-vosk-asr-gratuito-libre-y-offline.html> (visitado 10-05-2023).
- [50] Amazon Web Services. *¿Qué es un IDE? - Explicación de los entornos de desarrollo integrado - AWS*. url: <https://aws.amazon.com/es/what-is/ide/> (visitado 10-05-2023).
- [51] Java. *¿Qué es Java y por qué lo necesito?* url: <https://aws.amazon.com/es/what-is/java/> (visitado 15-05-2023).
- [52] campusmvp - José Manuel Alarcón. *Java: ¿Qué es Maven? ¿Qué es el archivo pom.xml?* | campusMVP.es. url: <https://www.campusmvp.es/recursos/post/java-que-es-maven-que-es-el-archivo-pom-xml.aspx> (visitado 15-05-2023).

- [53] openwebinars - Luis Miguel López Magaña. *Qué es Thymeleaf* | *OpenWebinars*. url: <https://openwebinars.net/blog/que-es-thymeleaf/> (visitado 15-05-2023).
- [54] Developer mozilla. *MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web* | *MDN*. url: <https://developer.mozilla.org/es/docs/Glossary/MVC> (visitado 15-05-2023).
- [55] Spring. *Spring Initializr*. url: <https://start.spring.io/> (visitado 15-05-2023).
- [56] Python. *venv — Creación de entornos virtuales — documentación de Python - 3.8.16*. url: <https://docs.python.org/es/3.8/library/venv.html> (visitado 16-05-2023).
- [57] Alphacephei. *VOSK Models*. url: <https://alphacephei.com/vosk/models> (visitado 16-05-2023).
- [58] Hmong - basada en Wikipedia. *AArch64 ARMv8-AyARMv8-R (arquitectura en tiempo real)*. url: <https://hmong.es/wiki/ARM64> (visitado 16-05-2023).
- [59] Bioinf. *Scripts de bash — Bioinformatics at COMAV 0.1 documentation*. url: https://bioinf.comav.upv.es/courses/unix/scripts_bash.html (visitado 17-05-2023).
- [60] Node.js. *Acerca* | *Node.js*. url: <https://nodejs.org/es/about> (visitado 17-05-2023).
- [61] Amazon Web Services. *¿Qué es el MQTT? - Explicación del protocolo MQTT - AWS*. url: <https://aws.amazon.com/es/what-is/mqtt/> (visitado 17-05-2023).
- [62] itsitio - Alejandro Alonso. *Qué se puede esperar del mercado de componentes en 2023 - ITSitio*. url: <https://www.itsitio.com/ar/que-se-puede-esperar-del-mercado-de-componentes-en-2023/> (visitado 18-05-2023).
- [63] React dev. *React*. url: <https://es.react.dev/> (visitado 18-05-2023).
- [64] Developer mozilla. *¿Cómo se utiliza Github pages? - Aprende desarrollo web* | *MDN*. url: https://developer.mozilla.org/es/docs/Learn/Common_questions/Tools_and_setup/Using_Github_pages (visitado 18-05-2023).

8. Anexo I. Obtención del *DevEUI*

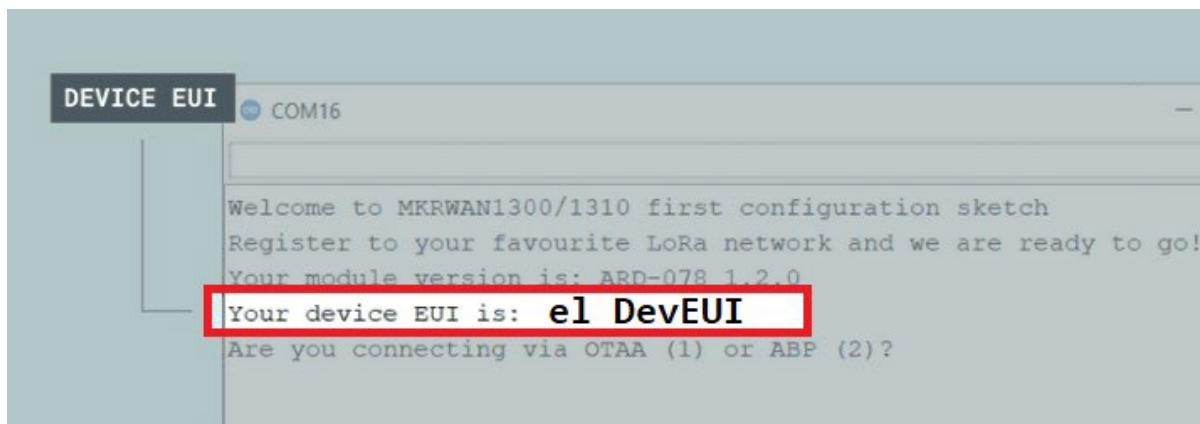
El objetivo de este anexo es el de obtener el DevEUI, necesario en el ecosistema de The Things Network para la vinculación entre un dispositivo *hardware* concreto y la propia red de The Things Network.

Para ello es necesario hacer uso del Arduino IDE, una vez descargado y ejecutado se requieren dos componentes que han de ser instalados a través del Arduino IDE, estos son:

1. **Información de la placa MKR 1310 WAN** Se instala en el apartado "Tools" luego acceder a "Board" y finalmente a "Board Manager" una vez ahí buscar "Arduino SAMD boards (32-bits ARM Cortex M0+)" e instalarlo.
2. **La librería MKRWAN** Se instala en el apartado "Tools" y acceder a "Manage libraries..." Una vez ahí buscar "MKRWAN" e instalarla.

Tras haber realizado estos pasos debe de aparecer en el apartado de la librería "MKRWAN" un archivo llamado "FirstConfiguration", esto se encuentra en el apartado "File" posteriormente acceder a "Examples" de ahí acceder a "MKRWAN" y finalmente encontraremos "FirstConfiguration".

Este archivo debe ser cargado en el Arduino, que previamente ha de estar conectado para que lo reconozca el IDE, para ello se utiliza el botón de "Upload", tras haber sido lanzado el archivo en la terminal del IDE aparecerá una vista similar a la siguiente figura (ver figura 69) con el DevEUI.



```
DEVICE EUI COM16
Welcome to MKRWAN1300/1310 first configuration sketch
Register to your favourite LoRa network and we are ready to go!
Your module version is: ARD-078 1.2.0
Your device EUI is: e1 DevEUI
Are you connecting via OTAA (1) or ABP (2)?
```

Figura 69: Vista de la ejecución del archivo FirstConfiguration con el DevEUI

9. Anexo II. Creación y configuración de un bot de Telegram

El objetivo de este anexo es el de crear y configurar un bot de Telegram para que notifique a los usuarios gestores que se encuentran en el mismo canal de Telegram que el bot, de la llegada de nuevas peticiones.

Para ello es necesario una cuenta de Telegram e iniciar sesión en esta, una vez iniciada la sesión en el buscador de la aplicación hay que buscar “@botfather”, el resultado ha de ser igual a la siguiente figura (ver figura 70):



Figura 70: Vista del Bot Father en Telegram I

Nota: se recomienda verificar que tiene un “check” azul.

Lo siguiente es comenzar un "chat" con él, una vez dentro, hay que seleccionar "iniciar" otra opción es escribir "/start", lo siguiente es escribir "/newbot", las preguntas que realizará son las siguientes:

- **Nombre:** el *nombre* que le demos al bot, es necesario tenerlo almacenado y a mano para la configuración posterior de Node-red.
- **Username:** el username identificativo del bot

Tras introducir dicha información el "BotFather" generará el bot que hemos pre-configurado y nos enviará su **token**, este valor es necesario almacenarlo y tenerlo a mano para su posterior uso con Node-red, además nos permitirá acceder al bot en el enlace que nos envía, tal y como se aprecia en la siguiente figura (ver figura 71).

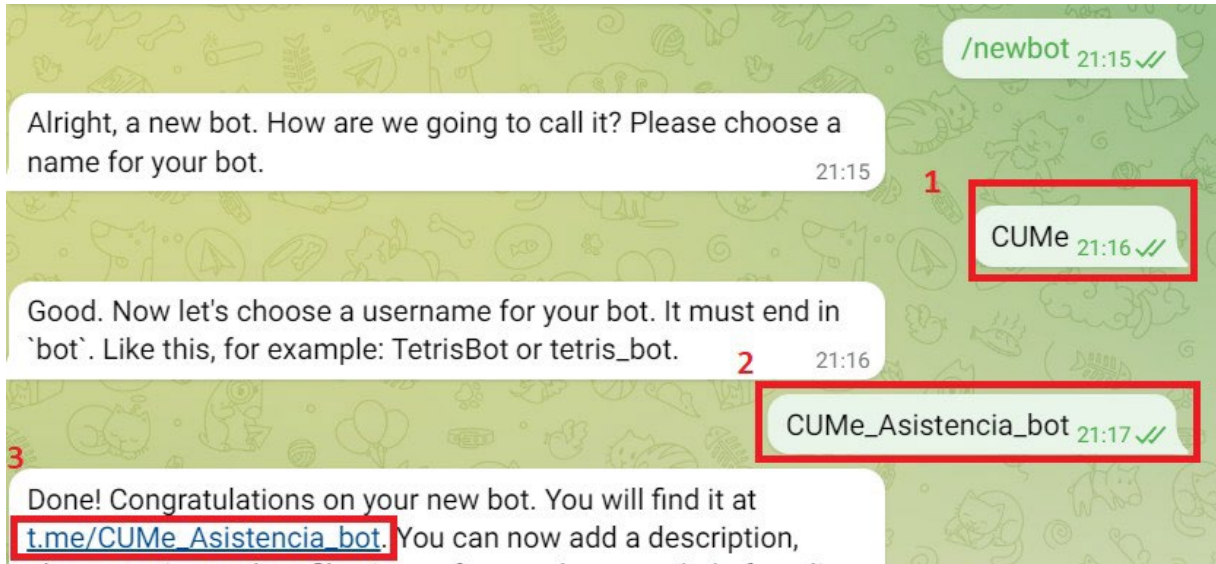


Figura 71: Vista del Bot Father en Telegram II

Sin embargo, aún hay que configurar algunos parámetros adicionales del bot, para ello seleccionaremos el enlace anteriormente citado (ver el elemento 3 en la figura 71) y una vez dentro hay que seleccionar "iniciar" o escribir "/start". A continuación hay que regresar al chat de "@Bot Father" y escribir "/setuserpic", este comando se emplea para asignar un icono al bot, tras esto hay que introducir "@" y seguido el nombre del bot, tras esto pedirá la imagen que usará el bot como icono, se le envía al chat y estaría configurado el bot.

El último paso para poder trabajar con Node-red es crear un grupo de Telegram y añadir al bot a dicho grupo, de este grupo es necesario obtener el **chatId** o identificador del grupo, este valor se encuentra en la barra de navegación si utilizamos "Telegram Web" y nos situamos en el grupo, es el valor que va precedido por un "-" y que incluye al propio guion, es necesario almacenarlo y tenerlo a mano para la configuración de Node-red.

En resumen, tras haber seguido los pasos anteriores se habrán obtenido tres elementos fundamentales para la realización de la conexión con Node-red, siendo estos:

- **Nombre:** el nombre que empleará el bot. Necesario para el elemento "Telegram Sender".
- **Token:** la clave identificativa del bot. Necesario para el elemento "Telegram Sender".
- **ChatId:** el código identificativo del chat donde el bot enviará los mensajes. Necesario para el elemento "function" encargado de "Notificar".

10. Anexo III. Manuales de uso de la aplicación web

La aplicación web posee dos tipos de funcionalidad, las cuales variarán en función del tipo de usuario, es decir, si es un gestor o un administrador, es por esto que el manual se divide en dos, dedicando un apartado para cada tipo de usuario.

Lo primero que han de hacer independientemente de si se es usuario, administrador o gestor es acceder al sitio web e iniciar sesión en la vista que se refleja en la figura 35 (ver figura 35).

10.1. Manual de uso del usuario gestor

El usuario gestor puede realizar las siguientes operaciones:

- **Visualizar las peticiones disponibles:** consiste en poder observar la lista de peticiones que aún no han sido asignadas a otro usuario gestor.
- **Encargarse de una petición:** consiste en asumir una petición de la lista de peticiones disponibles, de manera que deje de aparecer como "petición disponible" para el resto de usuarios gestores.
- **Dar por finalizada una petición:** consiste en marcar una petición como atendida, esta pasará a formar parte de las "peticiones finalizadas".
- **Revocar una petición de la que se encargaba:** consiste en devolver una petición que previamente se había asumido a la lista de peticiones disponibles, de manera que otro usuario gestor pueda realizarla.
- **Visualizar las peticiones finalizadas:** consiste en poder observar la lista de peticiones que otro gestor ha marcado como finalizada, es decir, ya ha sido atendida.
- **Cerrar sesión:** consiste en cancelar la sesión activa, volviendo en el proceso a la vista de inicio de sesión.

Para **visualizar las peticiones disponibles** el usuario gestor ha de seleccionar "Gestionar Peticiones", tal y como se observa en la figura 36 (ver figura 36).

Para **encargarse de una petición** el usuario gestor ha de seleccionar "Asumir", tal y como se observa en la figura 36 (ver figura 36).

Para **dar por finalizada una petición** el usuario gestor ha de seleccionar "Terminar", tal y como se observa en la figura 37 (ver figura 37).

Para **revocar una petición de la que se encargaba** el usuario gestor ha de seleccionar "Revocar", tal y como se observa en la figura 37 (ver figura 37).

Para **visualizar las peticiones finalizadas** el usuario gestor ha de seleccionar "Peticiones Finalizadas", tal y como se observa en las figuras 36 y 37 (ver figura 36), (ver figura 37).

Para **cerrar sesión** el usuario gestor ha de seleccionar "Cerrar sesión" en cualquiera de las vistas presentes en las figuras 36, 37 y 38, (ver figura 36), (ver figura 37), (ver figura 38) de esta manera volverá a la vista original de inicio de sesión (ver figura 35).

10.2. Manual de uso del usuario administrador

El usuario administrador puede realizar las siguientes operaciones:

- **Visualizar, añadir, modificar y eliminar dispositivos:** de manera que posee control total sobre los dispositivos del sistema. Es uno de los principales cometidos del usuario administrador el dar de alta los dispositivos, para que los usuarios gestores puedan registrar el origen de la petición.
- **Visualizar, añadir, modificar y eliminar gestores:** de manera que posee control total sobre los gestores del sistema. Es uno de los principales cometidos del usuario administrador, el dar de alta a los usuarios gestores, para que estos puedan acceder al sistema y atender las peticiones.
- **Visualizar, añadir, modificar y eliminar peticiones:** de manera que posee control total sobre las peticiones del sistema. Estas funcionalidades se utilizan principalmente durante la etapa de pruebas con los usuarios gestores.
- **Visualizar, añadir, modificar y eliminar centros de estudio:** de manera que posee control total sobre los centros de estudio del sistema. Se emplea para asignar a los distintos gestores los centros sobre los que han de trabajar.

- **Cerrar sesión:** consiste en cancelar la sesión activa, volviendo en el proceso a la vista de inicio de sesión.

Para **visualizar los dispositivos** el usuario administrador ha de seleccionar "Administrar Dispositivos", tal y como se observa en la figura 39 (ver figura 39). En la siguiente vista (ver figura 40) podrá **añadir dispositivos** al seleccionar "Nuevo", **modificar dispositivos** al seleccionar "Editar" ambas compartiendo la misma vista (ver figura 41) y eliminar dispositivos seleccionando "Eliminar" (ver figura 40).

Para **visualizar los gestores** el usuario administrador ha de seleccionar "Administrar Gestores", tal y como se observa en la figura 39 (ver figura 39). En la siguiente vista (ver figura 42) podrá **añadir gestores** al seleccionar "Nuevo", **modificar gestores** al seleccionar "Editar" ambas compartiendo la misma vista (ver figura 43) y eliminar gestores seleccionando "Eliminar" (ver figura 42).

Para **visualizar las peticiones** el usuario administrador ha de seleccionar "Administrar Peticiones", tal y como se observa en la figura 39 (ver figura 39). En la siguiente vista (ver figura 44) podrá **añadir peticiones** al seleccionar "Nuevo", **modificar peticiones** al seleccionar "Editar" ambas compartiendo la misma vista (ver figura 45) y eliminar peticiones seleccionando "Eliminar" (ver figura 44).

Para **visualizar los centros de estudio** el usuario administrador ha de seleccionar "Administrar Centros de Estudio", tal y como se observa en la figura 39 (ver figura 39). En la siguiente vista (ver figura 46) podrá **añadir centros de estudio** al seleccionar "Nuevo", **modificar centros de estudio** al seleccionar "Editar" ambas compartiendo la misma vista (ver figura 47) y eliminar peticiones seleccionando "Eliminar" (ver figura 46).

Para **cerrar sesión** el usuario administrador ha de seleccionar "Cerrar sesión" en cualquiera de las vistas presentes entre las figuras 39 y 47, (ver figura 39), (ver figura 47), de esta manera volverá a la vista original de inicio de sesión (ver figura 35).

11. Anexo IV. Instalación de Rasa NLU en Raspberry Pi 4 Modelo B

El objetivo de este anexo es el de detallar el proceso de instalación de Rasa NLU en la Raspberry Pi 4 Modelo B así como las dependencias necesarias para que funcione correctamente, esta implementación se basa en la que realizó Khaled Sakallah (khalo-sa/rasa-apple-silicon) para Apple Silicon basados en ARM de 64 bits (aarch64) en la web de *GitHub* [64]. Esta solución es válida para la Raspberry Pi 4 Modelo B pues ambas comparten la arquitectura ARM de 64 bits.

La dificultad de la instalación de Rasa radica en la incompatibilidad de algunas de las dependencias que requiere para funcionar con la arquitectura aarch64, siendo algunas de estas tensorflow y tensorflow-addons.

Para comenzar es recomendable crear un entorno virtual de python en la versión 3.8 para ello se recomienda el uso de conda; sin embargo, conda no es compatible con Raspberry Pi 4, esto sería un problema de no ser por la existencia de Miniforge3, una adaptación de conda para este tipo de dispositivos, una vez instalado Miniforge desde su web oficial, es necesario crear el entorno, para ello hay que introducir los siguientes comandos (ver listing 29):

Listing 29: Código de creación del entorno virtual usando Conda

```
$ sudo wget https://github.com/khalo-sa/rasa-apple-silicon/blob/docker/output/native/rasa_3.0.4_env.yml
$ conda env create -f rasa_3.0.4_env.yml
```

Nota I: el fichero con extensión “.yaml” citado anteriormente contiene las siguientes dependencias (ver listing 30):

Listing 30: Código que muestra las dependencias de Rasa para Conda

```
- aiohttp>=3.6,<3.7.4
- dask==2021.11.2
- h5py==3.1.0
- numpy==1.19.5
- python==3.8.12
- ruamel.yaml>=0.16.5,<0.17.0
- scikit-learn==0.24.2
- uvloop==0.14
- pip
- pip:
  - CacheControl>=0.12.9,<0.13.0
  - PyJWT>=2.0.0,<3.0.0
  - SQLAlchemy>=1.4.0,<1.5.0
  - absl-py>=0.9,<0.14
  - aio-pika>=6.7.1,<7.0.0
  - apscheduler>=3.6,<3.8
  - async_generator>=1.10,<1.11
  - attrs>=19.3,<21.3
  - boto3>=1.12,<2.0
  - cloudpickle>=1.2,<1.7
```

Desarrollo de un punto de asistencia inteligente basado en reconocimiento de patrones de voz con tecnología LoRaWAN

```
- colorama>=0.4.4,<0.5.0
- colorclass>=2.2,<2.3
- coloredlogs>=10,<16
- colorhash>=1.0.2,<1.1.0
- fbmessenger>=6.0.0,<6.1.0
- google-auth>=2.3.3,<3.0.0
- https://github.com/KumaTea/tensorflow-aarch64/releases/download/v2.6/
tensorflow-2.6.0-cp38-cp38-linux_aarch64.whl
- https://github.com/Qengineering/TensorFlow-Addons-
Raspberry-Pi_64-bit/raw/main/tensorflow_addons-
0.14.0.dev0-cp38-cp38-linux_aarch64.whl
- joblib>=0.15.1,<1.1.0
- jsonpickle>=1.3,<2.1
- jsonschema>=3.2,<3.3
- kafka-python>=1.4,<3.0
- keras<2.7.0
- matplotlib>=3.1,<3.4
- mattermostwrapper>=2.2,<2.3
- networkx>=2.4,<2.7
- packaging>=20.0,<21.0
- prompt-toolkit>=2.0,<3.0
- psycopg2-binary>=2.8.2,<2.10.0
- pyTelegramBotAPI>=3.7.3,<4.0.0
- pydot>=1.4,<1.5
- pykwalify>=1.7,<1.9
- pymongo>=3.8,<3.11
- python-dateutil>=2.8,<2.9
- python-engineio # >=4,<5.0.0 || >5.0.0,<6
- python-socketio>=4.4,<6
- pytz>=2019.1,<2022.0
- questionnaire>=1.5.1,<1.11.0
- randomname>=0.1.5,<0.2.0
- rasa-sdk>=3.0.0,<4.0.0
- redis>=3.4,<4.0
- regex>=2020.6,<2021.9
- requests>=2.23,<3.0
- rocketchat_API>=0.6.31,<1.17.0
- sanic>=21.6.0,<22.0.0
- sanic-cors>=1.0.0,<2.0.0
- sanic-jwt>=1.6.0,<2.0.0
- sanic-routing>=0.7.2,<0.8.0
- scipy>=1.4.1,<2.0.0
- sentry-sdk>=0.17.0,<1.4.0
- setuptools>=41.0.0
- sklearn-crfsuite>=0.3,<0.4
- slackclient>=2.0.0,<3.0.0
- tarsafe>=0.0.3,<0.0.4
# Solventada por enlace superior - tensorflow>=2.6,<2.6.2
# Solventada por enlace superior - tensorflow-addons>=0.14,<0.15
- tensorflow-estimator>=2.6,<2.7
- tensorflow-probability>=0.11,<0.14
# No disponible para aarch64 - tensorflow-text>=2.6,<2.7
- tensorflow_hub>=0.10,<0.13
- terminaltables>=3.1.0,<3.2.0
- tqdm>=4.31,<5.0
- twilio>=6.26,<6.51
- typing-extensions>=3.7.4,<4.0.0
- typing-utils>=0.1.0,<0.2.0
- ujson>=1.35,<5.0
- webexteamssdk>=1.1.1,<1.7.0
```

Nota II: si apareciera "conda command not found" en el terminal es necesario modificar el archivo ".bashrc" para ello habría que introducir los siguientes comandos (ver listing 31):

Listing 31: Código para solucionar el error de Conda

```
|| $ echo 'export PATH=/home/tu_username/miniconda3/bin:$PATH' >> ~/.bashrc
|| $ source .bashrc
```

Y tras esto es necesario activarlo mediante el siguiente comando (ver listing 32):

Listing 32: Código de activación del entorno virtual usando Conda

```
|| $ conda activate rasa304
```

Nota III: Si al ejecutar algún comando de Rasa la ejecución fuera errónea es posible que este ocurriendo un error de compatibilidad con la dependencia protobuff, para solucionarlo hay que cambiar la versión, para ello se introduce el siguiente comando (ver listing 33):

Listing 33: Código para solventar el error de Protobuff

```
|| $ pip install protobuf==3.20.*
```

Nota IV: Si al ejecutar el comando "rasa run actions" ocurre un error con packaging version, se soluciona mediante el siguiente comando (ver listing 34):

Listing 34: Código para solventar el error de Packaging version

```
|| $ pip install 'packaging>=20.0,<21.0' --force-reinstall
```

Tras esto, Rasa debería estar plenamente funcional en la Raspberry Pi 4 Modelo B, además de poder ejecutar sus comandos clásicos como son "rasa init", "rasa train", "rasa run actions", etc.