



Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN INGENIERÍA DEL SOFTWARE

**DISEÑO E IMPLEMENTACIÓN DE UN
SISTEMA DE CONTROL Y
MONITORIZACIÓN DE ASISTENTES
INTELIGENTES**

Alumno: Álvaro Velasco Gil

Tutor: Jesús M. Vegas Hernández

Dedicado a mis padres y mi hermana, por confiar en mi cuando yo no lo hacía.

Agradecimientos

Quiero agradecer en primer lugar a mi tutor, por la paciencia,
a mis amigos Jorge Sanz por facilitarme los primeros cursos,
a Javier Helguera por hacerme la carrera más llevadera,
a Diego Dominguez y David Escarda por hacer más sencillas las noches estudiando en el aula,
y a mi familia, por soportar mis cambios de humor.

Resumen

Este proyecto es movido dos grandes motivaciones: la privacidad de nuestros datos, y la asistencia de la población envejecida rural.

Con ello, se narra el proceso por el cual ha sido desarrollado un proyecto de despliegue de dispositivos, los cuales se comportan como asistentes personales orientados a la población más envejecida y con menos recursos, que notablemente está alojada en poblaciones rurales.

Estos asistentes proporcionarán una ayuda diaria a este grupo social que es el que en realidad puede necesitar de un dispositivo que le ayude a realizar las tareas y responder preguntas comunes, al igual que se tratará de un dispositivo que pueda detectar un posible accidente en el hogar y proporcionar una asistencia inmediata.

Para la realización de este proyecto, por tanto, se requiere la implementación de una API contra la que se comunicará cada dispositivo, al igual que un sitio web para que el administrador del sistema pueda controlarlos. Cada dispositivo estará desplegado remotamente y podrá ser controlado, por lo que en este proyecto también se desarrollará el controlador del dispositivo.

En cuanto al desarrollo se han utilizado diferentes lenguajes, tales como Kotlin con el framework de Ktor para la creación de la API, el framework de VueJS para la creación del sitio web, donde se incluye HTML + CSS + Javascript, al igual que la utilización de Python para la creación del controlador que irá instalado en cada dispositivo.

Toda el proyecto ha sido elaborado, dada su naturaleza, siguiendo un plan basado en iteraciones.

Abstract

This project is moved by two great motivations: privacy and the assistance of the ageing rural population.

Based on that, it narrates the process by which a project has been developed to deploy devices, which behave as personal assistants aimed at the older population with fewer resources, which is notably housed in rural populations.

These assistants will provide a daily help to this social group, which is actually the one that may need a device that helps them undertake their tasks, as well as answering common questions. The device will also be able to detect a possible accident at home and provide immediate assistance.

Therefore, in order to create this project, the implementation of an API that each device will be communicated with will be required, in addition to a website that allows the system administrator to control them. Each device will be remotely deployed and can be controlled. Consequently, in this project, the device controller will also be developed.

Regarding to the development, different languages have been used, such as Kotlin with Ktor's framework for the API creation, VueJS' framework for the creation of the website where HTML + CSS + Javascript is included, as well as the use of Phyton for the creation of the controller that will be installed in each device.

The whole project has been elaborated, due to its nature, following a plan based on iterations.

Índice general

Agradecimientos	III
Resumen	V
Abstract	VII
Lista de Figuras	XIII
Lista de Tablas	XVII
1. Introducción	1
1.1. Introducción	1
1.2. Motivación	1
1.3. Objetivos	2
2. Estado del Arte	5
2.1. Introducción	5
2.2. Requisitos mínimos	9
2.3. Mercado	11
2.3.1. Google Assistant	11
2.3.2. Amazon Echo	12
2.3.3. Mycroft	12

IX

2.3.4. Snips AI	13
2.3.5. Elección Final	14
3. Plan de Desarrollo	15
3.1. Introducción	15
3.2. Metodología	15
3.3. Planificación	16
3.4. Diagramas de Gantt	17
3.5. Recursos y Herramientas	26
3.5.1. Kotlin	26
3.5.2. Ktor	26
3.5.3. Koin DI	26
3.5.4. OAuth 2.0	27
3.5.5. Exposed SQL	28
3.5.6. JSoup	28
3.5.7. PostgreSQL	29
3.5.8. VueJS	29
3.5.9. Bootstrap-Vue	29
3.5.10. Leaflet	29
3.5.11. Material icons	30
3.5.12. Astah UML	30
3.5.13. Gantt Project	30
3.5.14. MermaidJS	30
3.5.15. LaTeX	30
3.5.16. Overleaf	30
3.5.17. Axios	30
3.5.18. Python	31
3.5.19. Git	31

4. Análisis del Proyecto	33
4.1. Introducción	33
4.2. Requisitos	33
4.2.1. Requisitos Funcionales	33
4.2.2. Requisitos No Funcionales	35
4.3. Riesgos	37
4.3.1. Introducción	37
4.3.2. Riesgos Identificados	38
4.3.3. Riesgos Acontecidos	40
4.4. Costes	42
5. Diseño	45
5.1. Casos de Uso	46
5.1.1. Diagrama de Casos de Uso	46
5.1.2. Definición de Casos de Uso	47
5.2. Diagramas de Secuencia	63
5.2.1. Introducción	63
5.2.2. Relativos a interacción con el Dispositivo	63
5.2.3. Relativos a la Interacción de un Administrador	71
5.3. Interfaz de Usuario	80
6. Implementación	81
6.1. Planteamiento	81
6.2. Arquitectura	81
6.2.1. Dispositivo	81
6.2.2. BackEnd	82
6.2.3. FrontEnd	82

6.3. Implementación	83
6.3.1. Dispositivo	83
6.3.2. BackEnd	88
6.3.3. FrontEnd	107
6.4. Pruebas	128
6.4.1. Introducción	128
6.4.2. Implementación	128
7. Despliegue	131
7.1. Introducción	131
7.2. Guía	132
7.2.1. Despliegue del Back End	132
7.2.2. Despliegue del Front End	134
7.2.3. Instalación del Dispositivo	135
8. Conclusiones y Trabajo Futuro	137
8.0.1. Conclusiones	137
8.0.2. Trabajo Futuro	137
Bibliografía	139

Índice de Figuras

2.1. Secuencia de Comunicación humana	8
2.2. Secuencia de Comunicación humano-asistente	9
3.1. Diagrama de Gantt - Planteamiento	18
3.2. Diagrama de Gantt - Prototipo 0	20
3.3. Diagrama de Gantt - Prototipo 1	21
3.4. Diagrama de Gantt - Prototipo 2	22
3.5. Diagrama de Gantt - Prototipo 3	23
3.6. Diagrama de Gantt - Prototipo 4	24
3.7. Diagrama de Gantt - Realización de la memoria y entrega	25
4.1. Matriz de priorización de riesgos	37
5.1. Diagrama general de casos de uso	46
5.2. Diagrama de secuencia DS01 - Dispositivo	63
5.3. Diagrama de secuencia DS02 - Login	64
5.4. Diagrama de secuencia DS03 - Alive	65
5.5. Diagrama de secuencia DS04 - Realización de tareas pendientes	66
5.6. Diagrama de secuencia DS05 - Servidor : Marcar tarea como realizada	67
5.7. Diagrama de secuencia DS06 - Servidor : Configuración actualizada	68
5.8. Diagrama de secuencia DS07 - Servidor : Obtener configuración actual	70

5.9. Diagrama de secuencia DS08 - Creación de nuevo tipo de tarea	71
5.10. Diagrama de secuencia DS09 - Asignación de tarea a dispositivo	72
5.11. Diagrama de secuencia DS10 - Recolección de eventos	73
5.12. Diagrama de secuencia DS11 - Obtención de tareas de un dispositivo	73
5.13. Diagrama de secuencia DS12 - Cambiar configuración	74
5.14. Diagrama de secuencia DS13 - Obtención de todos los dispositivos	75
5.15. Diagrama de secuencia DS14 - Obtención de actividad	76
5.16. Diagrama de secuencia DS15 - Adición de nueva localidad	77
5.17. Diagrama de secuencia DS16 - Adición de nuevo usuario	77
5.18. Diagrama de secuencia DS17 - Asignación de dispositivo	78
5.19. Diagrama de secuencia DS18 - Desasignación de dispositivo	78
5.20. Diagrama de secuencia DS19 - Obtención de relación	79
6.1. The Clean Architecture, Robert C. Martin [28]	82
6.2. Flujo de estados del dispositivo	84
6.3. Contenido global del repositorio	87
6.4. Diagrama de diseño	91
6.5. Diagrama de Clases: General	93
6.6. Diagrama de Clases: App	94
6.7. Diagrama de Clases: Domain	95
6.8. Diagrama de Clases: Services	95
6.9. Diagrama de Clases: Model	96
6.10. Diagrama Entidad-Relación	97
6.11. Estructura del FrontEnd: Model	107
6.12. Dashboard: Planteamiento de diseño	108
6.13. Dashboard: Dispositivo en línea	109
6.14. Dashboard: Dispositivo reciente	109

6.15. Dashboard: Dispositivo ausente	109
6.16. Dashboard: Dispositivo posiblemente apagado	109
6.17. Dashboard: Dispositivo apagado por comando	110
6.18. Dashboard: Dispositivo realizando tarea enviada	110
6.19. Dashboard - Planteamiento del filtro de dispositivos	110
6.20. Dashboard - Diseño final del filtro de dispositivos	111
6.21. Dashboard - Planteamiento de diseño de acciones rápidas.	111
6.22. Dashboard - Diseño final de acciones rápidas: dispositivo asignado.	112
6.23. Dashboard - Diseño final de acciones rápidas: dispositivo sin asignar.	112
6.24. Localidades - Planteamiento de lista: agrupado por provincias.	113
6.25. Localidades - Planteamiento de lista: despliegue de provincia.	113
6.26. Localidades - Diseño final de lista: agrupado por provincias.	113
6.27. Localidades - Diseño final de lista: despliegue de provincia.	114
6.28. Localidades - Planteamiento de diseño para añadir nueva localidad.	114
6.29. Localidades - Diseño final de añadir nueva localidad.	115
6.30. Settings - Planteamiento de diseño de configuraciones.	115
6.31. Settings - Diseño final de configuración global.	116
6.32. Settings - Diseño final de configuración local.	116
6.33. Settings - Diseño final de configuración de un dispositivo.	117
6.34. Users - Planteamiento de diseño de tabla de usuarios.	117
6.35. Users - Diseño final de tabla de usuarios.	117
6.36. Users - Planteamiento de diseño de registro de usuarios	118
6.37. Users - Diseño final de registro de usuarios: código postal no registrado	118
6.38. Users - Diseño final de registro de usuarios	119
6.39. Mapa - Prototipo de iconos	119
6.40. Mapa - Marker de dispositivo asignado	120

6.41. Mapa - Marker de dispositivo no asignado	120
6.42. Mapa - Diseño final: dispositivos separados	120
6.43. Mapa - Diseño final: agrupados por localidades	121
6.44. Perfil - Planteamiento de diseño de perfil de usuario/dispositivo	121
6.45. Perfil - Planteamiento de diseño de perfil de dispositivo sin usuario	122
6.46. Perfil - Planteamiento de diseño de pagina	122
6.47. Perfil - Planteamiento de diseño estadísticas	123
6.48. Perfil - Planteamiento de diseño de filtro.	123
6.49. Perfil - Diseño final de filtro. (Posibilidades)	123
6.50. Perfil - Diseño final del perfil: Usuario con dispositivo	124
6.51. Perfil - Diseño final del perfil: Dispositivo libre	124
6.52. Perfil - Diseño final del perfil: Usuario libre	125
6.53. Perfil - Pranteamiento de diseño de asignación de dispositivo	125
6.54. Perfil - Diseño final de asignación de dispositivo	126
6.55. Perfil - Diseño final de consultas	126
6.56. Perfil - Diseño final consultas: Ampliado	127

Índice de Tablas

4.1. Riesgo 01 - Enfermedad	38
4.2. Riesgo 02 - Planificación incorrecta	38
4.3. Riesgo 03 - Pérdida del trabajo	38
4.4. Riesgo 04 - Caída de los servidores	38
4.5. Riesgo 05 - Rotura de dispositivo	39
4.6. Riesgo 06 - Desconexión de la red.	39
4.7. Riesgo 07 - Privatización de algún servicio.	39
5.1. Caso de Uso D01 - Inicio de sesión	47
5.2. Caso de Uso U01 - Refresco de token de acceso	48
5.3. Caso de Uso D02 - Notificación de dispositivo conectado.	48
5.4. Caso de Uso D03 - Realización de tarea pendiente.	49
5.5. Caso de Uso D04 - Actualización de configuración	49
5.6. Caso de Uso D05 - Envío de actividad de un dispositivo	50
5.7. Caso de Uso A01 - Inicio de sesión	50
5.8. Caso de Uso A02 - Creación de un nuevo tipo de tarea.	51
5.9. Caso de Uso A03 - Asignación de nueva tarea.	51
5.10. Caso de Uso A04 - Obtención de tipos de tarea.	52
5.11. Caso de Uso A05 - Obtención de tareas de un dispositivo.	53
5.12. Caso de Uso A06 - Cambiar configuración de dispositivos.	54

5.13. Caso de Uso A07 - Obtención de todos los dispositivos.	55
5.14. Caso de Uso A08 - Obtención de actividad de un dispositivo.	56
5.15. Caso de Uso A09 - Obtención de actividad de un usuario	57
5.16. Caso de Uso A10 - Adición de una nueva localidad.	58
5.17. Caso de Uso A11 - Adición de un nuevo usuario.	59
5.18. Caso de Uso A12 - Asignación de dispositivo a un usuario.	60
5.19. Caso de Uso A13 - Desasignación de un dispositivo a un usuario.	61
5.20. Caso de Uso A14 - Obtención de usuario de un dispositivo específico.	61
5.21. Caso de Uso A15 - Obtención de localidades.	62
5.22. Caso de Uso A16 - Obtención de la lista de usuarios registrados.	62

Capítulo 1

Introducción

1.1. Introducción

En este proyecto se abordará el proceso de control y monitorización remota de dispositivos asistentes inteligentes.

Tras la lectura de este documento se comprenderán tanto los motivos por los cuales se ha decidido tomar esta opción, como el proceso de despliegue del sistema, pasando por las fases de implementación en las que se enseñará a replicar proyectos de estructura similar, como por las fases de desarrollo en las cuales se estudian los posibles riesgos y características del proyecto, al igual que por el plan de desarrollo en el cual se programa toda la elaboración.

1.2. Motivación

La idea de este proyecto llegó impulsada gracias a dos grandes motivaciones: conseguir la privacidad en la utilización de dispositivos inteligentes, y la posibilidad de asistir a la población envejecida de entornos rurales.

La primera motivación, relativa a la privacidad viene dada ya que los asistentes inteligentes están en auge y las grandes empresas están dando acceso a estos servicios de manera gratuita, donde lo único que hay que hacer para disfrutar de uno de ellos es pagar el dispositivo físico, asumiendo únicamente los costes del hardware a un precio reducido, lo que hace cuestionar la idea del modelo de negocio que están siguiendo para que salga rentable: El desarrollo de un sistema software capaz de dar respuesta a toda cuestión que un usuario se llegue a plantear, al igual que el mantenimiento de las infraestructuras que puedan soportar todo un sistema capaz de retroalimentarse mediante la información vertida por los usuarios, tiene un coste muy elevado como para ofrecer ese servicio de manera gratuita. Un modelo de negocio en el cual se ofrece un dispositivo a un bajo precio con unas prestaciones de gran nivel sin

cobrar a mayores una cuota mensual da que pensar, por ejemplo, que el producto mediante el cual está basado su modelo de negocio no es el que el usuario cree: Para el usuario, el producto es el dispositivo que le va a servir un uso, pero para la empresa, el producto es toda la información personal que le va a proporcionar cada usuario que utilice el dispositivo. La información recopilada a través de los asistentes, al ser obtenida directamente de los hogares de cada usuario no es solamente privada, si no que también es íntima, lo que hace cuestionarse si realmente la comunidad de usuarios sabe que hay empresas aprovechándose de las conversaciones de su día a día para recolectar toda la información de un usuario que fluye en los lugares más personales de cada hogar, capturando información con gran potencial, ya sea sobre gustos, tendencias, necesidades o inclinaciones políticas, con el fin de poder no solo utilizar esos datos, sino también pudiendo incluso venderlos a terceros, a través de los cuales pueden llegar a intentar manipularnos, como ya se ha visto en casos como el escándalo de Cambridge Analytica. [1]

Esta aparente ignorancia de la población sobre el tratamiento de datos que las empresas podrían ser capaces de hacer genera la motivación suficiente para la creación de un nuevo dispositivo asistente, el cual haga más fácil la vida de un sesgo de la sociedad al cual iría orientado, sin tener la necesidad ni posibilidad de almacenar datos de carácter sensible.

Por otro lado también se observa otra gran motivación para la elaboración del proyecto, y es producida por la migración por la parte joven de la sociedad en los tiempos que acontecen actualmente, lo cual está provocando una despoblación en sus lugares de origen, dejando a los familiares de mayor edad en soledad, siendo la parte de la familia que a grosso modo necesita más atención y requiere más ayuda para pasar el día.

La implementación de un asistente inteligente el cual ayude a este conjunto de la sociedad a entretenerse proponiendo tanto eventos cercanos a ellos, como respondiendo sus dudas de una manera rápida, o sirviéndoles para buscar ayuda en caso de posible caída o solicitud de auxilio, puede ser una gran herramienta que mejore su calidad de vida.

1.3. Objetivos

Dadas y mencionadas las motivaciones que impulsan este proyecto, se puede destacar que el objetivo principal se basa en la privacidad por diseño.

El objetivo de este proyecto se logrará, por tanto, mediante el despliegue de un sistema que permita la conexión de dispositivos al servidor de manera automática y remota, a la vez que permita que sean controlados desde un sitio web de administración de una manera simple, desde la cual se podrá ver también estadísticas de uso para poder comprobar el estado de los usuarios, sin la obtención de datos que atenten contra su privacidad.

Estos dispositivos, como objetivo, estarán orientados a personas del rango de la tercera edad, estando cada dispositivo asignado a una persona, de la cual se conoce su localidad con el fin de obtener información relacionada a la hora de responder.

Los dispositivos podrán entonces ser controlados remotamente por parte de los administradores y podrán ser configurados tanto de manera individual, como en función de su

localidad, o de una manera global, pudiendo solicitar al dispositivo la actualización de manera remota a través de configuraciones, o solicitar la realización de diferentes acciones como puede ser la auto-configuración de la red Wi-Fi, u otras para mejorar la gestión de despliegue sin requerir la conexión física con un ordenador.

1.3. OBJETIVOS

Capítulo 2

Estado del Arte

2.1. Introducción

Preámbulo

Los asistentes virtuales cada vez están más presentes en nuestras vidas gracias a todos los avances de la tecnología, y puede verse en que ya aparecen en todos los teléfonos móviles para ayudar en el día a día de los usuarios, ya sea para responder preguntas, poner una canción, o buscar información de manera rápida para el usuario.

La utilización de estos asistentes virtuales hace que cada vez más personas se planteen la posibilidad de llevar más lejos la utilización de ellos, como puede ser a través de otros dispositivos inteligentes a los que se les está buscando hueco cada vez en más hogares.

Estos dispositivos nos hacen la vida más sencilla, ayudando a actualizar y controlar toda la domótica de nuestras casas, de manera que se puedan realizar acciones en nuestra vivienda que hace unos años solo se podían imaginar en películas de ciencia ficción.

La utilización de asistentes virtuales para facilitar acciones que se repiten de manera continuada a lo largo de los días está cada vez más de moda: según un estudio de la reputada Nielsen [2], durante el Q2 de 2018 ha llegado a crecer el número de dispositivos desplegados en los hogares de Estados Unidos hasta alcanzar el 24 % de la población.

A la pregunta de para qué puede querer el 24 % de las familias un dispositivo asistente en sus casas, se puede observar tras una encuesta de dicho estudio, que en la mayoría de hogares se daba un uso bastante simple, solicitando al dispositivo únicamente la reproducción de contenido multimedia, pero esto ocurre únicamente en las primeras semanas de uso. También se puede observar en el estudio que, posteriormente, se tiende a dar más uso al dispositivo llegándole a requerir información habitual que se requiere de manera repetitiva cada semana, como puede ser la consulta del tiempo, la del tráfico antes de ir a trabajar o a la consulta de algún evento cercano importante.

2.1. INTRODUCCIÓN

La simplicidad de uso y todas las posibilidades que abarca ha hecho que una vez ha sido usado este tipo de dispositivo inteligente en cualquier hogar, se le siga utilizando cada vez para más tareas, dada su usabilidad y abanico de utilidades.

En ese estudio arriba citado, también se puede observar el comportamiento de los usuarios: una vez el dispositivo está en entre las cuatro paredes de la casa, se tiende a acompañar a este dispositivo inteligente de otros dispositivos domóticos, convirtiendo poco a poco el hogar en un sistema interconectado: cuatro de cada diez hogares disponen de más de un dispositivo inteligente en el hogar, y el 45 % de los hogares se plantean acompañar al que ya tienen, comprando otro.

Esta gran aceptación a la domótica viene dada gracias a la cantidad de variantes de las cuales podemos dotar al hogar, pudiendo ser todas controladas a través del dispositivo asistente, así como puede ser la programación de los tiempos de uso de aparatos como las bombillas, el termostato, los sistemas de seguridad, e incluso tanto el frigorífico, el sistema de riego, u otras acciones relativas a la seguridad de la vivienda como puede ser bloquear las puertas y ventanas, al igual que mantener el registro de cuáles y cuándo han sido abiertas, entre otras muchas opciones.

Al irrumpir estos dispositivos en las acciones cotidianas de un día corriente, se le permite al usuario adaptarse y ver que estos aparatos le acaban facilitando el día.

Viendo el creciente uso de estos dispositivos y las grandes marcas que hacen acto de presencia facilitando su implementación, entre las cuales se encuentran Google o Amazon, se puede pensar que este mercado ya está dominado y pertenece a estos dos grandes imperios, pero en este caso no se va a tener en cuenta todo lo que ya ofrecen, sino que se va a tener en cuenta todo lo que no están ofreciendo, o qué es lo que sus dispositivos no están preparados para ofrecer:

Como bien se ha visto, el punto de mira en todo el tema de la domótica y asistentes tiene un público objetivo que es el que pertenece a un rango de edad al que le gusta la tecnología y le parecería interesante automatizar las tareas del hogar, pero el verdadero público que podría exprimir todo su potencial es el público que menos acostumbrado nos tiene a estar a la vanguardia de la tecnología, el público perteneciente a la tercera edad.

Si se realiza el acto de volver a pensar sobre lo que es un asistente personal, en cuanto a una persona que trabaja como asistente, se entiende que es una persona que ayuda en las acciones del día a día más simples y repetitivas para las que otra persona no es capaz de hacer por si misma por algún tipo de incapacidad, ya sea incapacidad temporal, o algún tipo de incapacidad física.

Una vez planteado a qué se dedica un asistente personal, es fácil darse cuenta que al hablar de este trabajador se plasme la imagen de una persona que ayuda a personas mayores, o a personas con algún tipo de discapacidad.

El rango de personas de tercera edad es un público que puede necesitar que le levanten las persianas de manera automática porque quizá, le cuesta levantarse de la cama. Es un público que puede necesitar utilizar un asistente que le recuerde realizar ciertas tareas en su día a día como puede ser ir al médico, tomarse la pastilla, hacer la compra, o recordarle

qué productos tiene que comprar, al igual que el cumpleaños de un ser querido. Todo esto acompañado de la seguridad que le puede proporcionar tareas como asegurarse que todas las puertas de su casa están bien cerradas, o saber que aún estando estas personas solas en casa, podrían solicitar auxilio en un momento de urgencia a través del asistente.

La tecnología vinculada a Internet nos hace la vida más sencilla a través del smartphone o de los ordenadores, pero las labores que nos proporciona como extra un dispositivo asistente inteligente a un usuario medio el cual ya está acostumbrado a usar un smartphone son muy pocas, ya que podría realizarlas a través de su propio teléfono, en cambio, a una persona de tercera edad a la cual le cuesta hacer un uso normal de dispositivos tecnológicos, un dispositivo asistente inteligente podría hacerle la vida mucho más sencilla, ya que lo único que necesitaría sería establecer una conversación oral con el dispositivo, evitando el intermediario que supone un ordenador o un smartphone.

El rango de personas de tercera edad, es por tanto el público que más puede necesitar un dispositivo asistente, así como el público que más puede llegar a exprimirlo.

La creación de un asistente que pueda resolver sus dudas, con el que pueda hablar, al que pueda preguntar a qué hora es la partida de cartas en el bar, o al que puedan pedir auxilio en caso de una caída, puede ser de gran utilidad para mejorar su día a día, al igual que servir de alivio para el resto de familiares que no pueden estar cerca de sus seres queridos, sabiendo que van a poder ser informados rápidamente de un posible accidente.

He aquí, por tanto, el agujero de mercado que ha sido encontrado en las grandes empresas antes nombradas, y el cual puede permitir un modelo de negocio el cual tenga como premisa buscar la satisfacción de las personas mayores, que igualmente puede servir para dar libertad a otros grupos sociales, como pueden ser las personas con algún tipo de discapacidad.

El auge de los dispositivos asistentes se puede ver como una evolución tecnológica que permita facilitar las rutinas, pero también con ello se pone en vilo la gestión de la privacidad e intimidad que hay dentro de nuestras casas. Esto es debido a que este tipo de dispositivos inteligentes pueden recolectar información a través de escuchar las conversaciones de un usuario y su día a día, siendo información que no se sabe de manera exacta a dónde va a parar, o qué se va a hacer con ella.

Esta información captada puede ser de un carácter sensible, ya que puede contener desde simples gustos, incluyendo las necesidades del usuario, como sus tendencias políticas, siendo datos que pueden estar siendo utilizados por terceros, o incluso pudiendo estar siendo vendidos.

La falta de soluciones en el mercado que cumplan todos estos puntos supone una motivación para la creación de un asistente que no esté conectado constantemente a Internet, ya que las personas de edad avanzada seguramente no tengan contratado el servicio, y que tampoco almacene datos de carácter sensible de los usuarios, sino que simplemente interaccione con cada persona y la ayude en su día a día, tanto ofreciendo actividades de la misma localidad, como respondiendo cualquier pregunta que pase por la cabeza de quien lo posea, al igual que buscando ayuda en caso de que una persona requiera auxilio.

Antes de adentrar en la elección de un asistente inteligente, se expondrá qué es, y cuál es

su funcionamiento, para entender mejor los motivos por los cuales se acaba eligiendo uno en vez de otro.

Qué es un Asistente Inteligente

Un asistente inteligente es una máquina programada de tal manera que su comportamiento se asemeje al de una persona a la que se solicita asistencia, como su propio nombre indica, pudiendo mantener una conversación que siga los protocolos de comunicación humana.

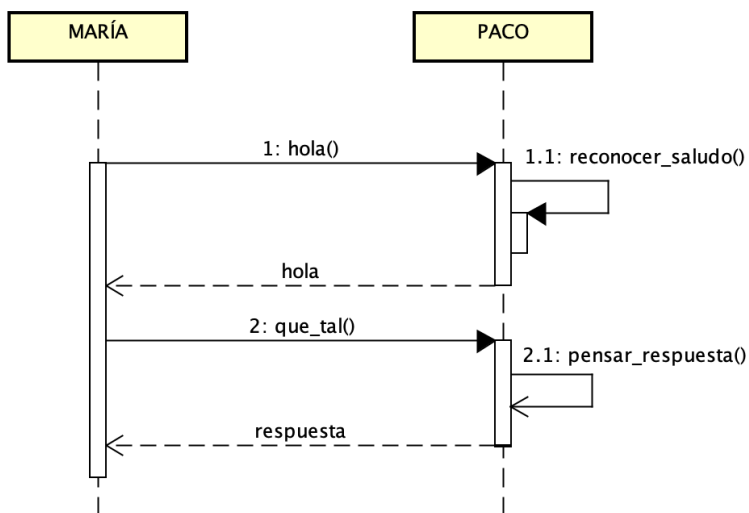


Figura 2.1: Secuencia de Comunicación humana

Como se puede ver en la figura 2.1, un protocolo de comunicación entre dos personas se basaría en un saludo para entablar conversación, para posteriormente realizar una pregunta.

En el caso de los asistentes, el proceso de conversación se basa en lo mismo: un usuario saluda al asistente mediante el uso de una palabra o conjunto de palabras, al que se llamará **hotword**, que cuando sea reconocido por el asistente inteligente, devolverá el saludo. Es entonces cuando el usuario debe realizar la pregunta o solicitar la información que requiera. Una vez hecha la pregunta, el dispositivo se pondrá a pensar la posible respuesta, entrando en el proceso al que se llamará **reconocimiento de los hechos**. Una vez identificados los hechos, devolverá la respuesta que más se acerque a lo deseado, gracias a un entrenamiento previo.

Cómo Piensa el Asistente

El proceso de pensamiento analizado de los principales asistentes del mercado, que se expondrá en el presente capítulo, tiene una estructura similar independientemente del tipo de asistente que se trate, asemejándose a la figura 2.2.

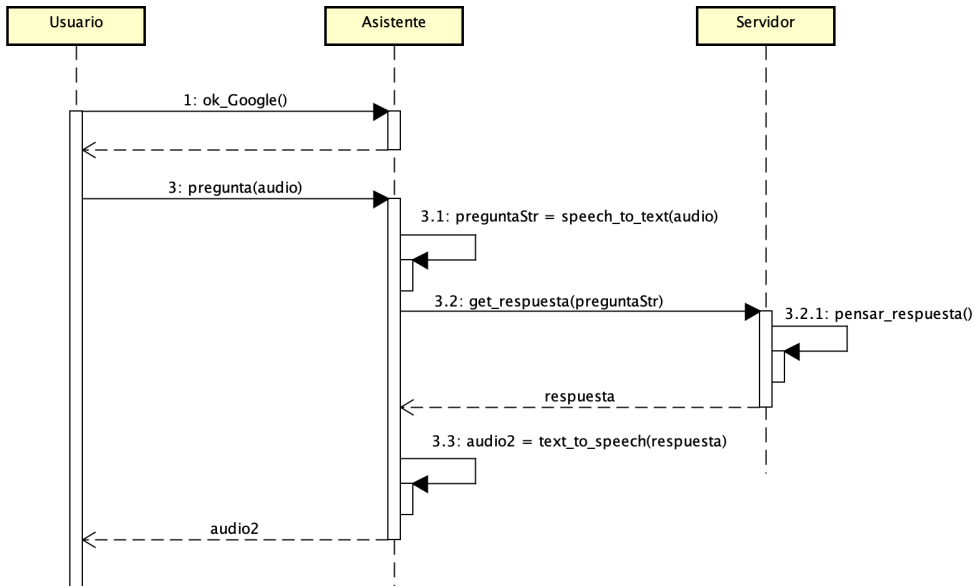


Figura 2.2: Secuencia de Comunicación humano-asistente

Lo que diferencia a un asistente de otro, es la manera en la cual piensa la respuesta, dando mayor validez a un asistente que dé una respuesta más aproximada a lo solicitado. Para lograr dar la respuesta más acertada, la mayoría de los asistentes existentes realizan el procesamiento de la respuesta en la nube debido a la gran cantidad de información que tienen previamente almacenada de otras consulta de otros usuarios para contrastar los hechos capturados, al igual que también pueden realizar en la nube el proceso de `speech-to-text` enviando los audios a sus servidores y transformándolo ahí a cadenas de texto, o el de `text-to-speech`, que sería el proceso inverso.

2.2. Requisitos mínimos

Una vez que ya se ha expuesto cual es el funcionamiento base de un asistente virtual inteligente, interesa saber cuál es de todos los existentes que más se adecúa a las necesidades establecidas para este proyecto, de manera que se le exigirá que cumpla el máximo de los siguientes puntos expuestos.

Despliegue en Dispositivos

El asistente seleccionado debe poder ser desplegado de manera gratuita en un dispositivo formado por una placa RaspberryPi, o similar.

Para ello, se requiere que el asistente disponga de una librería o de un modo de uso que se pueda implementar en un dispositivo pequeño y portátil, facilitando el cambio de una ubicación física por otra de los distintos posibles puntos que existen dentro de un hogar, manteniendo una instalación lo más simple y limpia posible, en la que solo sea requerida la localización de un enchufe activo que posea corriente eléctrica.

Tratamiento de la Información

El asistente debe seguir la Ley Orgánica de Protección de Datos [3] y el Reglamento Europeo de Protección de Datos [4], y no debe tener acceso a escuchar conversaciones ajenas violando la intimidad de los usuarios.

Cualquier vacío legal o conjunto de cláusulas de extensa longitud no podrá ser aceptado, para poder asegurar la protección de los datos de los usuarios.

En caso de almacenar algún tipo de dato, debe poder ser público para el usuario en cuestión, o haber sido aceptado expresamente por el usuario a favor de un control de su seguridad.

Conexión a Internet

La orientación de un dispositivo asistente inteligente como este a personas de una edad avanzada debe tener en cuenta que es posible que la mayoría de sus usuarios potenciales no tengan una conexión a internet en sus respectivos hogares.

Esto provoca que la conexión del dispositivo a la red tiene que ser lo mínima posible, favoreciendo los asistentes en los cuales el proceso de pensamiento sea ejecutado dentro del propio dispositivo, evitando tener que hacer el cálculo e identificación en servidores de alojados en la nube.

Es decir, el dispositivo debe ser capaz de funcionar en el lugar más remoto posible y sin ninguna conexión a internet, simplemente tras ser conectado a una red eléctrica.

Adicción de Nuevas Tareas

El software del asistente inteligente debe tener la capacidad de añadir nuevas tareas fácilmente, de modo que se puedan añadir nuevas funciones tanto propias, como desarrolladas por la comunidad de Internet.

2.3. Mercado

En los siguientes apartados se mostrarán las diferentes opciones que ofrece ya el mercado para el uso de asistentes inteligentes, con el fin de comprobar la existencia de alguno que ya tenga establecidos los requisitos base ya descritos, o que permita la opción de poder establecerlos.

2.3.1. Google Assistant

Introducción

El asistente inteligente de Google es el que está en el año 2020 a la cabeza de los asistentes virtuales [5], y esto es debido a que viene instalado en todos los dispositivos Android, ocupando estos dispositivos la mayor parte del mercado de la telefonía móvil. Esta posición privilegiada le proporciona un mayor entrenamiento, dando por resultado una elevada tasa de acierto que como consecuencia le atribuye mayor usabilidad.

Esta creación de Google tiene una gran aceptación con los diferentes aparatos existentes para la domótica del hogar, facilitando la implementación y realización de tareas:

Una tarea es un conjunto de acciones que se llevan a cabo tras accionarse un evento que hace de interruptor, pudiendo ser el evento tanto un comando de voz, la pulsación de un interruptor o la activación de un sensor, entre otras opciones.

De esta manera, Google permite el control de la domótica del hogar, o la realización de diferentes acciones en nuestro día a día, pero requiere ser configurado por cada usuario, evitando por tanto la posibilidad de un despliegue común.

Cómo Funciona

El proceso de pensamiento que hace el asistente de Google está basado en la nube, pero no solo eso sino que Google hace en sus servidores también el proceso de Speech-to-Text.

De este modo, Google almacena todos los audios[6] que se le mandan a través de su asistente, estudiándolos uno a uno y asegurando o corrigiendo sobre la respuesta que envió el asistente.

Este proceso de corregir o confirmar es el método que tiene Google de entrenamiento, de manera que en la próxima consulta sobre el mismo tema, el asistente pueda responder con mayor exactitud.

Queda claro que sobre la teoría es un buen plan de entrenamiento, y en un mundo ideal esto sería perfecto, pero esos audios también pueden contener parte de información privada, o pueden servir para espiar conversaciones privadas que un usuario puede no querer que estén almacenadas, ni que sean escuchadas por otra persona, aunque sea un propio trabajador.

En cuanto a la posibilidad de ser desplegado en otro dispositivo que no sea un smartphone o una tablet, Google nos lo pone fácil, ya que otorga una librería con la cuál facilita la instalación y uso en una gran variedad de dispositivos, cuyo requisito es que dispongan de conexión a Internet.

2.3.2. Amazon Echo

También conocido como Alexa, es la opción creada por Amazon que más fuerza está tomando en la sociedad para ser elegida como el asistente de voz inteligente que nos acompañe en nuestro día a día.

A pesar de todas las ventajas posibles similares a las que se han descrito para el asistente de Google, también comparte sus contras, como las referentes a la localización donde se procesan los audios y donde se efectúa el entrenamiento, por lo que es otra opción que va a ser rechazada.

Sin tener en cuenta la ubicación del proceso de pensamiento, Amazon asegura[7] que almacena todos los audios hasta que es el propio usuario quien decide borrarlos, pero aún con la solicitud expresa del usuario no pueden ser borrados completamente ya que han sido compartidos con terceros, que son quienes han desarrollado funciones específicas, llamadas Skills. Estos desarrolladores no solo tienen acceso a los audios sino que los poseen, pudiendo hacer una mala práctica con ellos.

Por tanto, aunque el usuario en cuestión pida a Amazon el borrado de estos ficheros, la compañía únicamente podría borrar los archivos que posee, permaneciendo todos los cuales un desarrollador de estas skills haya almacenado, sin permitir que el propio Amazon sea capaz de eliminarlos aunque tratase de hacerlo.

2.3.3. Mycroft

Ante la falta de privacidad por parte de Google y Amazon, se abre la puerta a Mycroft, proyecto Open Source que busca como pilar la seguridad de los datos y la protección de la privacidad, de manera que asegura no almacenar ningún dato o información del usuario potencial. [8]

Mycroft tiene una de las mayores comunidades OpenSource activa para el desarrollo e implementación de asistentes [9], de modo que dispone de una gran cantidad de skills que añadir a nuestro dispositivo, posicionándose como opción principal para la elaboración del proyecto.

Este asistente virtual puede ser fácilmente implementado en dispositivos tales como Raspberry, cumpliendo los propósitos y requisitos de este proyecto.

En cuanto a la conexión a Internet, el equipo de Mycroft informa acerca de estar trabajando en una herramienta que permita desplegar el asistente ya entrenado dentro del dispositivo,

evitando la conexión, pero de momento esa herramienta no está finalizada, por lo que cojea en ese aspecto y de momento, debería tener una conexión permanente.

En caso de que estén en lo cierto y estén trabajando en esa herramienta, este asistente ocuparía la primera opción como asistente a desplegar, pero en el momento actual en el cual este proyecto cobra vida, no es viable su implementación.

2.3.4. Snips AI

Snips Seed es una plataforma de inteligencia artificial para el desarrollo de dispositivos asistentes.

Esta plataforma nos permite crear nuestro propio asistente basándose en un entrenamiento previo con unos hechos predefinidos por el desarrollador.

En ese entrenamiento previo, se permite al asistente tomar lo aprendido de otros skills, que son fáciles de añadir.

Tras la implementación en el dispositivo físico, el asistente ya ha sido entrenado, por lo cual no necesitaría volver a estar conectado a internet, siendo este el mayor argumento a favor ya que es la carencia del resto de asistentes.

En cuanto a su uso, el dispositivo solamente se mantiene a la espera de poder recibir el *hotword*. Este *hotword* puede ser modificado por cualquier otra palabra que se desee, lo cual beneficia al proyecto actual con la posibilidad de asignar una palabra más acorde y familiar a las personas a quienes va orientado.

El asistente, en cuanto a su configuración interna, sigue el protocolo de comunicación descrito en la figura 2.2, con una pequeña modificación: Cuando el servidor ya ha entendido la pregunta o solicitud, genera unos hechos definidos en su entrenamiento en forma de objeto que envía a través de un puerto MQTT, y se queda a la espera de una respuesta.

Los otros skills, que han sido añadidos previamente en el entrenamiento, están levantados escuchando por el puerto, de manera que identifican los objetos que circulan por él y comprueban si ese hecho le corresponde, que de no ser así, simplemente lo ignoran. Una vez que un skill captura un hecho que sí que le corresponde, comprueba los *slots* que contiene dicho hecho para ver qué es lo que se está solicitando. Tras esto, genera una respuesta en forma de cadena de texto, y la envía por el puerto MQTT para que la recoja el *skill-server*.

Una vez capturada por este, simplemente transcribe el mensaje a audio con su skill de text-to-speech, y se lo transmite al usuario a través del altavoz, finalizando la comunicación.

Como se puede ver, el tratamiento de la información que se le otorga al dispositivo es el que nosotros deseamos, ya que el asistente no envía nada al exterior. De esta manera, nosotros controlamos qué sucede con la información, siendo el otro punto requerido en la búsqueda de un asistente ideal.

Como extra, Snips Seeed es una plataforma gratuita que se basa en una colaboración de una gran comunidad de desarrolladores. Pese a ser esta comunidad de menor tamaño que la conseguida por la opción de Mycroft, hay una gran cantidad de aplicaciones disponibles como juegos, la consulta del tiempo o la consulta de noticias, preparadas para ser asignadas directamente a nuestro dispositivo.

Como se puede ver, SnipsSeeed otorga las herramientas necesarias para cumplir todos los requisitos estipulados anteriormente, siendo la plataforma que se pone a la cabeza como posible elección.

2.3.5. Elección Final

La elección más acorde de entre todas las propuestas es el uso Snips AI, ya que se adapta a todos nuestros requisitos.

En cuanto al proyecto, va a ser necesario el desarrollo de un dispositivo que contenga este asistente, al igual que el diseño y desarrollo de un backend capaz de albergar la información relativa a cada dispositivo.

Para la gestión de estos dispositivos también será necesaria la implementación de una página web, desde la cual puedan ser los dispositivos configurados, manejados y controlados.

La implementación operativa de todo este sistema formaría un proyecto de gran envergadura, por lo que en los siguientes capítulos se documentará el diseño y desarrollo del backend y del frontend, mostrando el control y gestión real de un dispositivo el cual su rango de funciones será breve, de modo que se establezca la base y se explique como puedan ser añadidas nuevas funciones de una manera simple y sencilla.

Capítulo 3

Plan de Desarrollo

3.1. Introducción

En el siguiente capítulo se mostrará cual ha sido el método por el cual se ha desarrollado el proyecto, al igual que se definirán las fases del plan de desarrollo a seguir.

También se mostrará y explicará cuales han sido tanto las herramientas utilizadas, como las librerías necesarias para una correcta implementación y desarrollo del proyecto con el fin de acercar al lector a entender cómo se ha diseñado, y qué es lo necesario para poder replicar el contenido del mismo en otros proyectos similares.

3.2. Metodología

Para el desarrollo de este proyecto se ha escogido una implementación siguiendo la combinación de un modelo iterativo y un modelo incremental. [10]

Este método puede describirse de una manera simple y rápida, basada en la repetición de ciclos en los cuales se van mejorando las funcionalidades implementadas a partir de la experiencia del usuario, y añadiendo nuevas.

Cada ciclo puede describirse como la continuación de los siguientes pasos:

1. El desarrollador muestra un prototipo.
2. El usuario lo prueba.
3. El usuario muestra su experiencia tanto positiva como negativa, comentando posibles mejoras y proponiendo nuevas funcionalidades.

4. Se subsanan esas carencias.
5. Se obtiene una nueva versión del prototipo.
6. Vuelta al paso inicial.

La elección de este modelo se debe, por tanto, a la naturaleza del proyecto del cual se trata: al estar trabajando con unas nuevas tecnologías, y no tener muy claras las capacidades de las cuales los posibles asistentes están dotados, es preferible realizar un desarrollo basado en la entrega de pequeños prototipos que nos vayan ampliando el conocimiento de todas las posibilidades que van a poder ser implementadas, al igual que produce una retroalimentación a partir de la experiencia del usuario, siendo bastante útil tanto para mejorar, como para eliminar o para añadir nuevas funcionalidades.

En el caso del presente proyecto, el usuario el cual realiza las pruebas del prototipo se trata del propio tutor del proyecto, el cual otorga su experiencia, experiencia que está curtida a través de años de trabajo y perfeccionamiento, siendo más sabia que la del propio desarrollador, ayudando a elaborar un rediseño de las características que aumente la usabilidad.

3.3. Planificación

Una vez conocido el método de trabajo que se seguirá para el desarrollo del proyecto, se deben conocer los plazos en los cuales el proyecto debe ser finalizado y entregado.

La fecha de inicio del proyecto es, por tanto, *el 15 de Octubre de 2019*, y la fecha límite en la cual el proyecto debe ser entregado data del *29 de Febrero de 2020*.

Disponibilidad del desarrollador

El proyecto se debe ajustar a una extensión de 300 horas, como fue especificado en el cálculo de costes en el apartado 1 del presente documento.

En cuanto a la disponibilidad del proyecto, el desarrollador carece de suficiente tiempo al día como para elaborar 8 horas diarias, asemejándose a un trabajo real, debido a tener que compaginar las 8 horas diarias de un trabajo externo, con las necesarias para la completitud de dos asignaturas de la Universidad, y el presente proyecto.

Tras una planificación semanal, el desarrollador es capaz de asignar una media de 20 horas semanales, lo que supondría poder finalizar el proyecto en:

$$15\text{horas/semana} / 5 \text{ días laborables/semana} = 3\text{horas/día}$$

$$300\text{horas} / 3 \text{ horas/día} = 100 \text{ días laborales}$$

$$100 \text{ días} / 5 \text{ días/semana} = 20 \text{ semanas}$$

3.4. Diagramas de Gantt

Al datarse de un proyecto basado en prototipos y dada la amplitud posible del mismo, se va a proceder a organizar el plan de trabajo en la secuencia del siguiente ciclo:

1. *Elicitación de requisitos*

Periodo en el cual se razona cuáles son los requisitos necesarios que se van a llevar a cabo en el prototipo a desarrollar. En es este periodo en el cual se realiza el diagrama de Gantt del prototipo indicado, planeando y estipulando los tiempos necesarios para llevar a cabo cada tarea.

2. *Desarrollo*

Esta etapa es en la cual se lleva a cabo la elaboración del prototipo siguiendo el plan de desarrollo estipulado en el punto anterior.

3. *Presentación del prototipo*

Reunión con el tutor para mostrar el prototipo y sus funcionalidades, con el fin de ver si se ha alcanzado lo esperado, al igual que para realizar un brainstorm conjunto con el fin de poder elicitar los requisitos del prototipo siguiente.

El número de prototipos elaborados será tantos como permita la duración máxima del proyecto, estimada en 20 semanas, enfocando cada prototipo en la completitud de un mínimo de requisitos con los cuales se busca un acercamiento a un prototipo final lo más aproximado posible a un producto final.

Debido a la condición del desarrollador que va a llevar a cabo el proyecto, se considera que cada día marcado en los siguientes diagramas corresponde con 3 horas de trabajo.

Para llevar a cabo la organización, se han respetado dos días libres a la semana, los cuales coinciden con el fin de semana, al igual que se respetan los días festivos nacionales.

La estimación de elaboración y seguimiento, de manera general, se basa en el diagrama de Gantt mostrado en la figura 3.1.

3.4. DIAGRAMAS DE GANTT

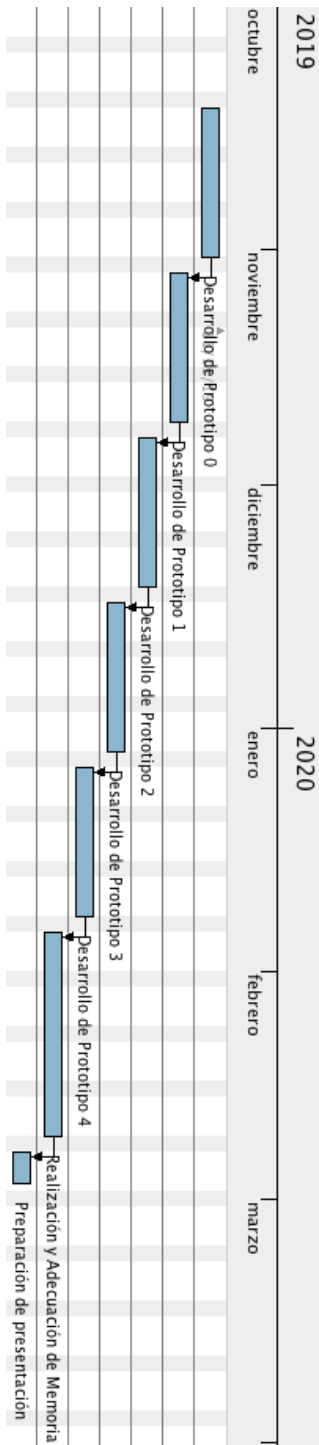


Figura 3.1: Diagrama de Gantt - Planteamiento

Una vez el proyecto se puso en marcha, se ha ido creando el diagrama de Gantt particular para cada prototipo, adecuándolo a los requisitos estipulados en cada uno de ellos.

Esta elicitación iterativa ha producido que los tiempos y fechas de desarrollo programados en cada prototipo no se asemejen a los mostrados en el diagrama de Gantt que planteaba la elaboración del proyecto.

Otros asuntos de causa mayor han producido retrasos en la elaboración del proyecto, de modo que los tiempos marcados no se asemejen a lo estipulado en un principio, estando estos motivos explicados en el capítulo 4.3.3.

A continuación, se mostrará el desglose de cada tarea nombrada en el planning inicial de la figura 3.1.

3.4. DIAGRAMAS DE GANTT

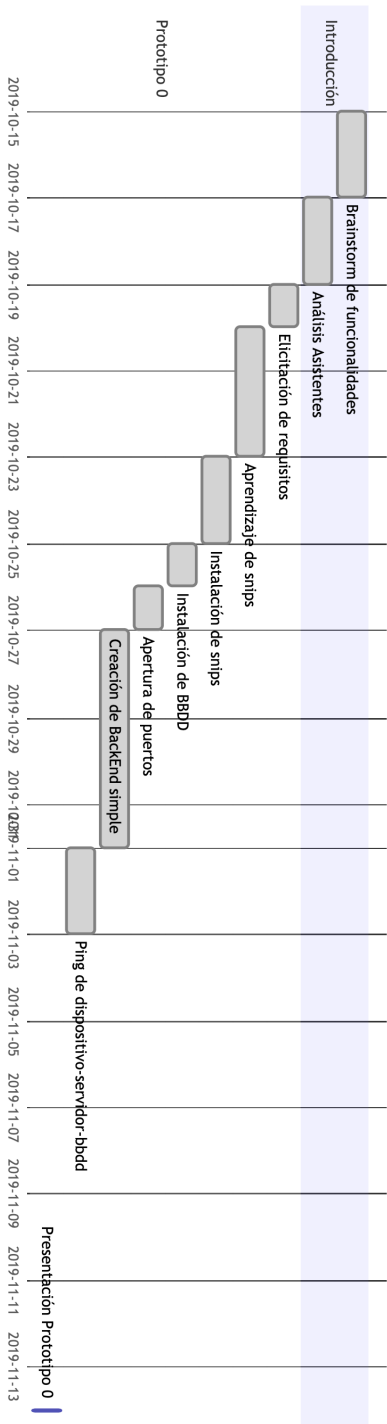


Figura 3.2: Diagrama de Gantt - Prototipo 0

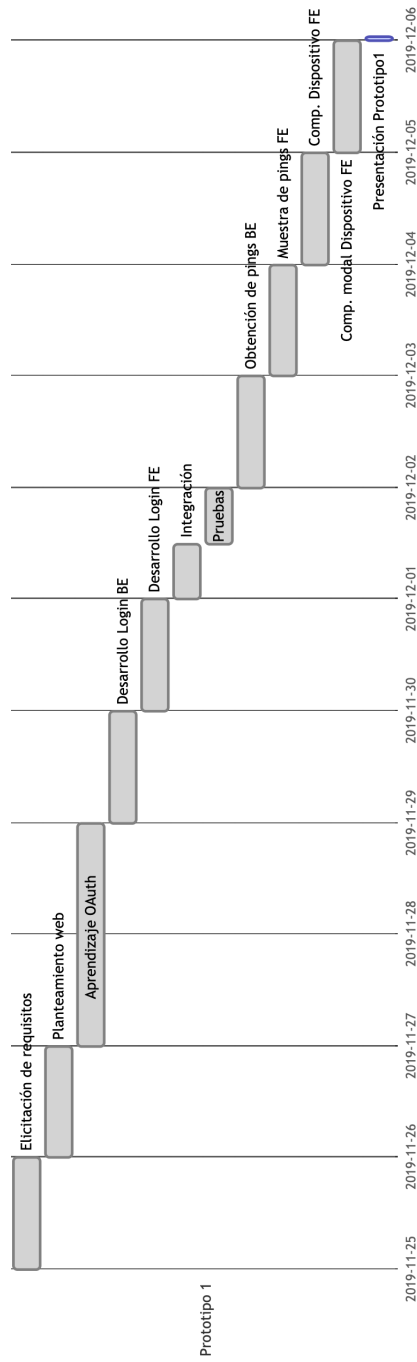


Figura 3.3: Diagrama de Gantt - Prototipo 1

3.4. DIAGRAMAS DE GANTT

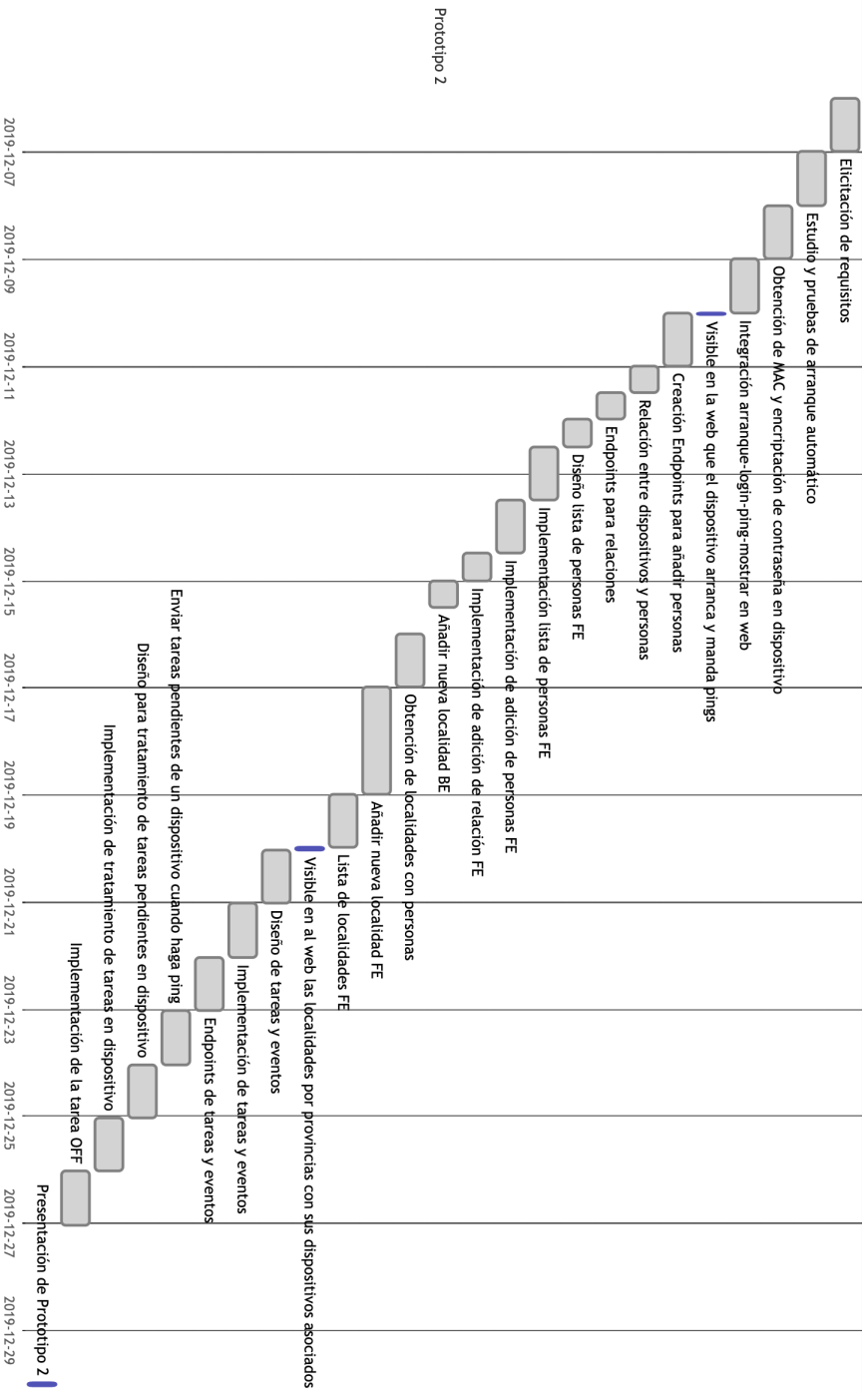


Figura 3.4: Diagrama de Gantt - Prototipo 2

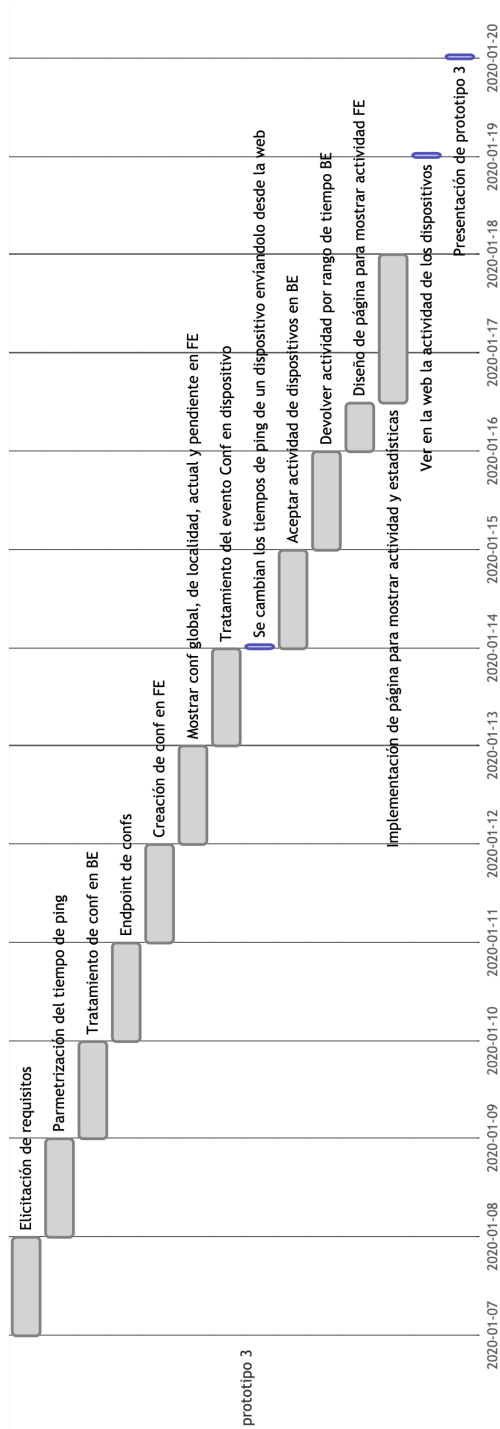


Figura 3.5: Diagrama de Gantt - Prototipo 3

3.4. DIAGRAMAS DE GANTT

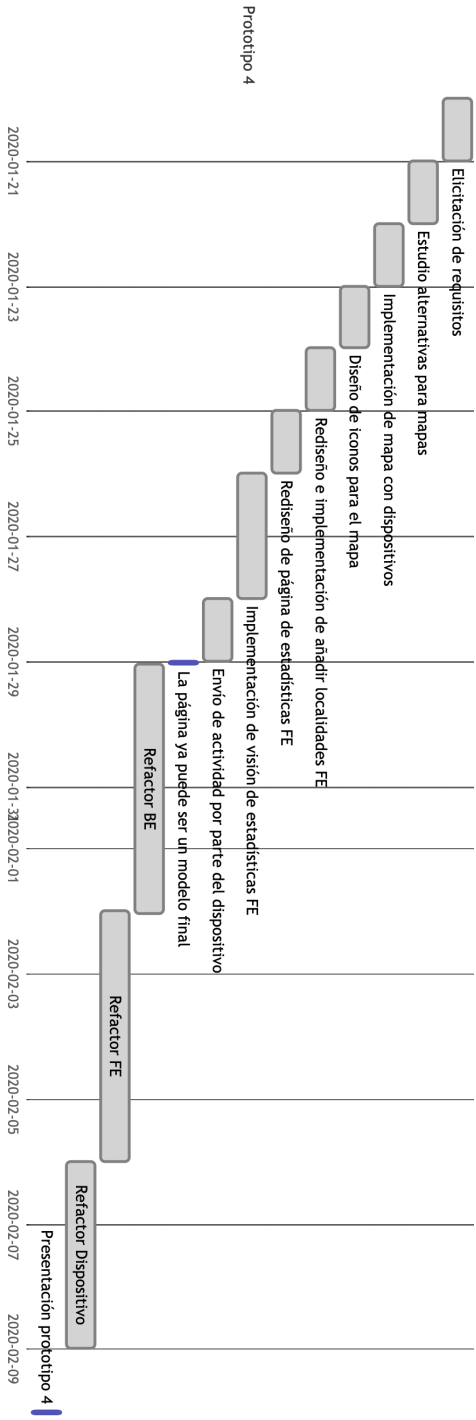


Figura 3.6: Diagrama de Gantt - Prototipo 4

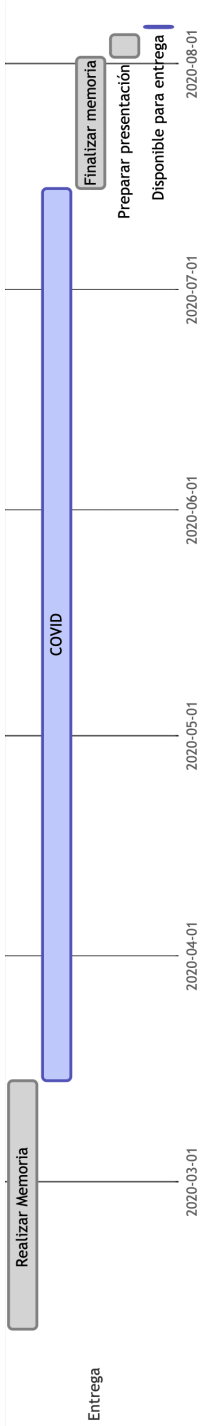


Figura 3.7: Diagrama de Gantt - Realización de la memoria y entrega

3.5. Recursos y Herramientas

Para una correcta implementación que se adecúe a los requisitos, al igual que para un futuro mantenimiento del proyecto, enumerarán a continuación las herramientas que han sido utilizadas durante su desarrollo, con el fin de poder replicar en un futuro el entorno de trabajo en un futuro, o poder entender mejor cómo ha sido implementado.

3.5.1. Kotlin

Kotlin es un lenguaje de programación funcional desarrollado por el equipo ruso de JetBrains como una evolución a Java por excelencia, permitiendo una interoperabilidad entre ambos, utilizándose bajo la JVM.

La sintaxis de Kotlin lo hace más intuitivo y simple, llegando al mismo resultado que en Java pero en un menor número de líneas, ahorrando tiempo y espacio.

Pese a que su gran popularidad es por el desarrollo Android, ahora está creciendo en su utilización en el lado del servidor gracias al framework de Ktor. [11]

3.5.2. Ktor

Es un framework que se aprovecha de la usabilidad de Kotlin para el desarrollo de clientes y servidores asíncronos en la implementación de sistemas conectados.

Este framework será añadido al proyecto a través de la instalación de sus dependencias en gradle a través del fichero *./build.gradle*.

3.5.3. Koin DI

Koin es un inyector de dependencias compatible con Kotlin, siendo bastante útil para poder aplicar el principio de inversión de dependencias en el desarrollo del backend. [12]

La configuración de este inyector de dependencias es bastante simple, teniendo una sintaxis sencilla y útil en la que únicamente hay que utilizar Kotlin, evitando así los ficheros de configuración en XML que obligan a usar otros tipos de inyectores.

La utilización de este framework será útil para poder aislar los casos de uso del sistema, de modo que no tengan dependencia de ninguna otra clase u servicio externo a los casos de uso, aumentando la escalabilidad del sistema y permitiendo una extensibilidad de los casos de uso en un futuro.

Para su utilización, se añaden las dependencias al fichero *./build.gradle*.

Se implementará a través de la instalación de su módulo en el main de la aplicación.

```
// Declare Koin
install(Koin) {
    modules(myModule)
}
```

y en el módulo de Koin se indicarán los servicios que serán inyectados.

```
// app.di.Module.kt

/**
 * This module has got the Koin configuration with the declarations of the needed
 * instances.
 * This instances could be injected now, applying the dependency inversion principle.
 */
val myModule = module {
    single { ConfRepo() as ConfService }
    single { DeviceRepo() as DeviceService }
    single { LocationRepo() as LocationService }
    single { PeopleRepo() as PeopleService }
    single { TaskRepo() as TaskService }
    single { UserRepo() as UserService }
    single { IntentRepo() as IntentsService }
    single { TokenCtrl() as AuthService }
}
```

3.5.4. OAuth 2.0

OAuth 2.0 es un protocolo que establece las pautas para una conexión segura entre un servidor y un cliente. Este protocolo se basa en la implementación de un servidor de tokens frente al cual hay que estar identificado. [13]

La secuencia correcta para la utilización de este protocolo consta de los siguientes pasos:

1. El usuario manda sus credenciales al sistema.
2. El sistema comprueba que sus credenciales son correctas.
3. El sistema solicita al servidor OAuth un nuevo inicio de sesión para un cierto usuario con las ciertas características.
4. El servidor OAuth provee al sistema un conjunto de token temporales: uno de acceso que utilizará el usuario para identificarse, y otro de refresco para que recupere el de acceso en caso de que se le caduque.
5. El sistema devuelve los tokens al usuario

Una vez que el usuario ya dispone de los tokens, simplemente tiene que realizar las peticiones normales a los end-points [14] del sistema pero introduciendo el token de acceso en la cabecera de autorización en cada llamada, de modo que el sistema lo toma, comprueba contra el servidor OAuth2.0 que sigue siendo un token válido, y en caso de serlo, deja realizar al usuario la petición.

Para la implementación de este protocolo utilizaremos un framework adaptado a la lógica de Ktor. [15]

Este framework será adaptado a las características de uso del sistema, estableciendo el método de comprobación de credenciales que se considere oportuno, pudiendo variar en caso de que las credenciales correspondan con las de un dispositivo, o con las de un administrador del sistema.

1. Añadir sus dependencia al fichero `./build.gradle`.
2. La adaptación de sus clases de identificadores y tienda de tokens con el fin de poder hacer una comprobación de credenciales y autenticar los tokens como requiera el sistema. Para ello, se han modificado dos clases: **InMemoryTokenStoreCustom** y **InMemoryIdentityCustom**, las cuales se albergan en `./app/oauth`.
3. Instalación del servidor OAuth2.0 en nuestra aplicación. Dentro de `./app/Application.kt`, instalamos el servidor:

```
// instance of tokenStore manage tokens
tokenStore = InMemoryTokenStoreCustom.get()

// Install and configure the OAuth2 server
install(OAuth2ServerFeature) {
    // set the custom store to have access -> Singleton Pattern
    tokenStore = InMemoryTokenStoreCustom.get()
}
```

3.5.5. Exposed SQL

Exposed SQL es un framework que simplifica el acceso a la base de datos evitando las consultas puras de SQL mediante una sintaxis más simple que tiene definida en su DSL. [16]

El método por el cual se ha accedido al uso de este framework es por su facilidad para parsear los resultados obtenidos en cada consulta, al igual que para limitar los posibles ataques por inyección de SQL.

3.5.6. JSoup

JSoup es una librería que permite la obtención del código fuente HTML resultante de sitios web, dando las herramientas necesarias para la captura y parseo de los diferentes valores en

función de las etiquetas, clases, y jerarquía de los distintos elementos HTML obtenidos. [17]

3.5.7. PostgreSQL

También conocido como postgres, es un sistema de gestión de bases de datos relacionales, por lo que permite una orientación a objetos en las relaciones de los elementos contenidos en sus diferentes tablas. Postgres es libre y de código abierto, lo que nos permite su uso en el proyecto.

3.5.8. VueJS

VueJS es un framework JavaScript que permite la creación y desarrollo de diferentes aplicaciones web. Entre sus ventajas está su reactividad, permitiendo cambiar la información mostrada en la web de manera dinámica, evitando tener que recargar la página.

Otro aspecto por el cual elegir este framework está en su simplicidad para crear e importar componentes, al igual que para enviar la información entre ellos.

También cabe destacar su modularidad: VueJS parte de cero, de modo que toda función de la que queramos disponer, simplemente tendrá que ser importada, pudiendo adaptar cualquier librería JavaScript fácilmente. Esto permite un mayor acceso a todas las librerías ya existentes, al igual que la creación de un proyecto con un menor peso final, al contener únicamente las librerías que son necesarias.

El aspecto más importante por el cual VueJS está siendo utilizado es su curva de aprendizaje, característica con la que todo el mundo concuerda: es más fácil de aprender a utilizar que sus principales competidores, entre los cuales se encuentran Angular y React. [18]

3.5.9. Bootstrap-Vue

Bootstrap-Vue es una adaptación de la librería de componentes Bootstrap que permite la utilización de sus componentes en VueJS de una manera más simple y rápida. [19]

3.5.10. Leaflet

Leaflet es una biblioteca JavaScript basada en OpenMaps, permitiendo la utilización e implementación de mapas en nuestras páginas web de una manera gratuita gracias a ser de código abierto. [20]

Leaflet tiene un gran desarrollo por parte de la comunidad, lo que hace a esta librería más interesante para la implementación de diferentes funciones, como puede ser la adición de cualquier tipo de marca en el mapa, la geocodificación a través de los atributos de un punto

exacto como puede ser su calle o código postal, y permitiendo también la geocodificación inversa, obteniendo esos datos de un punto exacto a partir de las coordenadas.

3.5.11. Material icons

Es la librería de iconos gratuita de Google, la cual sigue los estilos de diseño de Material Design, basada en un estilo limpio y simple.

3.5.12. Astah UML

Es una herramienta cuya función es la creación de diagramas y esquemas UML, que nos será muy útil para realizar el diseño y organizar el desarrollo del sistema.

3.5.13. Gantt Project

Herramienta útil para la implementación de diagramas de Gantt.

3.5.14. MermaidJS

Framework para la elaboración más simple de diagramas de Gantt. Dispone de editor online, lo cual es una ventaja para evitar instalar software de terceros.

3.5.15. LaTeX

Es un sistema de software para la preparación de documentos. Se ha utilizado para la elaboración de este documento.

3.5.16. Overleaf

Plataforma mediante la cual trabajar los documentos LaTeX de manera online, evitando la instalación de software de terceros.

3.5.17. Axios

Cliente HTTP basado en *Promises*. Con él se realizarán las llamadas HTTP a la API REST desde el Front End.

3.5.18. Python

Lenguaje de programación el cual proporciona facilidad para realizar scripts. Será utilizado en el desarrollo del controlador del dispositivo.

3.5.19. Git

Herramienta para llevar un sistema de control de versiones y evitar la pérdida de archivos del proyecto.

Capítulo 4

Análisis del Proyecto

4.1. Introducción

Una vez elegido el tipo de asistente virtual inteligente en el capítulo previo, en el presente capítulo se procede al análisis tanto de requisitos que debe cumplir el sistema, como de las funcionalidades que debe tener la página de administración, pasando por las operaciones que debe de ser capaz de realizar el dispositivo.

4.2. Requisitos

Para poder identificar qué es lo que se debe implementar, se debe realizar primero una identificación de las necesidades técnicas del proyecto a desarrollar, con el fin de identificar cuales son las implementaciones que se deben desarrollar.

4.2.1. Requisitos Funcionales

Los requisitos funcionales son un conjunto de funciones que debe poder servir el proyecto. Están basados en especificar qué es lo que el sistema debe ser capaz de hacer o permitir, y son los siguientes:

RF1 - Los dispositivos deben ser capaces de identificarse ante el sistema.

RF2 - El sistema debe permitir obtener un nuevo token de acceso mediante un token de refresco.

RF3 - El sistema debe ser capaz de registrar nuevos dispositivos.

4.2. REQUISITOS

- RF4 - El sistema debe devolver un par de tokens de autenticación a los dispositivos que se identifiquen.
- RF5 - El dispositivo debe informar al sistema sobre su estado cada un cierto tiempo, que pueda ser configurable, siendo guardado en un fichero de configuración.
- RF6 - El sistema debe poder actualizar remotamente el archivo de configuración tanto de dispositivos particulares, como de dispositivos de una misma localidad, como de todos los dispositivos en conjunto.
- RF7 - El sistema debe poder asignar tareas a un dispositivo, a una localidad, o a todos los dispositivos.
- RF8 - El sistema debe ser capaz de mandar tareas pendientes a realizar a un dispositivo, como puede ser actualizar su archivo de configuración, apagar el dispositivo, o reiniciarlo, cuando el dispositivo informe de que esté conectado.
- RF9 - El dispositivo debe realizar las tareas por fecha de creación.
- RF10 - El dispositivo debe informar al sistema cada vez que proceda a realizar una tarea.
- RF11 - El sistema debe poder marcar tareas de un dispositivo específico como ya realizadas, guardando la fecha de realización.
- RF12 - El sistema debe ser capaz de recibir la información básica sobre una conversación entre el asistente y el usuario, para monitorizar su uso.
- RF13 - El dispositivo debe avisar al sistema sobre conversaciones que haya tenido con el usuario, mandando la información básica que confronte con los hechos.
- RF14 - El sistema tiene que ser capaz devolver información básica sobre una localidad, como el número de teléfono, direcciones de correo, y noticias en función de la ubicación del dispositivo que llame al servicio.
- RF15 - Un administrador debe poder iniciar sesión, identificándose a continuación mediante tokens.
- RF16 - Un administrador debe ser capaz de ver todos los tipos de tareas disponibles.
- RF17 - Un administrador debe poder crear nuevos tipos de tareas.
- RF18 - El dispositivo debe tener un protocolo de adicción de nuevas tareas, de manera que al añadir una nueva tarea no haya posibilidad de afectar al resto ya implementadas.
- RF19 - Un administrador debe poder asignar tareas a dispositivos específicos, a dispositivos de una localidad específica, o a todos los dispositivos en general.
- RF20 - Un administrador debe poder ver las tareas pendientes de un dispositivo específico.
- RF21 - Un administrador debe poder cambiar la configuración a dispositivos específicos, a dispositivos de una localidad específica, o a todos los dispositivos en general.

- RF22 - Un administrador debe poder obtener una lista de todos los dispositivos, incluyendo estos sus últimos estados, últimas tareas realizadas, tareas pendientes, e información sobre el usuario asignado.
- RF23 - Un administrador debe poder ver la actividad ocurrida en un intervalo específico de tiempo de un dispositivo sin asignar.
- RF24 - Un administrador no debe poder ver actividad de un usuario que ya no tenga un dispositivo asociado.
- RF25 - Un administrador debe poder ver la actividad de un dispositivo asignado ocurrida en un intervalo específico de tiempo cuya fecha mínima sea la fecha en que se estableció la asignación.
- RF26 - Un administrador debe poder añadir nuevas localidades al sistema, incluyendo su nombre, coordenadas, y código postal.
- RF27 - Un administrador debe poder obtener la lista de localidades almacenadas en el sistema, teniendo cada localidad información sobre los dispositivos asignados a clientes.
- RF28 - Un administrador debe poder añadir nuevos clientes, que representarán a los poseedores del dispositivo.
- RF29 - Un administrador debe poder obtener la lista de clientes.
- RF30 - Un administrador debe poder asignar un dispositivo a un cliente específico.
- RF31 - Un administrador debe poder desasignar un cliente de un dispositivo.
- RF32 - Un administrador debe ser capaz de poder ver cuales son las acciones que más se le han solicitado a un dispositivo, y las estadísticas de a qué horas ha sido más veces consultado.

4.2.2. Requisitos No Funcionales

Los requisitos no funcionales, al contrario de los descritos en el apartado anterior, se basan en la descripción de cómo esta implementado el sistema y sobre cómo debe interactuar.

- NF1 - La base de datos debe ser una base de datos de tipo relacional, preferiblemente usando PostgreSQL.
- NF2 - El backend debe estar implementado con un lenguaje que pueda ser ejecutado bajo la JVM.
- NF3 - Las contraseñas que se almacenen en el sistema deben estar encriptadas mediante una encriptación de tipo SALT.
- NF4 - El administrador debe poder ver la ubicación de los dispositivos a través de un mapa implementado con Leaflet y OpenMaps.

4.2. REQUISITOS

- NF5 - De un cliente se debe almacenar su código postal, nombre, apellidos y número de documento nacional de identidad.
- NF6 - En la web administrativa, un dispositivo SIN cliente asignado debe aparecer en el mapa ubicado en la costa atlántica, con el icono en color amarillo.
- NF7 - En la web administrativa, un dispositivo CON cliente asignado debe aparecer en el mapa ubicado en la posición de su localidad, con el icono en color verde.
- NF8 - En la web administrativa, todos los dispositivos de una localidad deben aparecer en el mapa agrupados, ocultando sus iconos y mostrando el número de ellos que hay en ese grupo.
- NF9 - En la web administrativa, al hacer clic en un grupo de dispositivos del mapa, se debe mostrar el icono de cada dispositivo en color verde.
- NF10 - En la web administrativa, al hacer clic en un icono del mapa, debe salir información sobre el cliente asignado, y un enlace a para ver sus estadísticas.
- NF11 - El periodo de filtración de las estadísticas en la web administrativa debe ser de un día específico, o de un mes específico, o de un año específico.
- NF12 - Las estadísticas de acciones de un dispositivo deben ser mostradas en la web administrativa en un diagrama de tipo Doughnut, y las horas de uso en un diagrama de barras.
- NF13 - La web administrativa debe tener un panel donde se vean los dispositivos mediante tarjetas, que cambien de color en función de tiempo que lleve el dispositivo sin ser usado.
- NF14 - La web administrativa debe permitir filtrar las tarjetas en función de si los dispositivos están asignados, de su última fecha de uso, y de la última fecha de tarea realizada, mostrando cuánto tiempo hace de su último uso.
- NF15 - La web administrativa debe permitir asignar tareas a los dispositivos de una manera rápida dando clic a una tarjeta, al igual que dar acceso rápido al apartado de estadísticas y configuraciones, y mostrar información del usuario asignado.

4.3. Riesgos

4.3.1. Introducción

A continuación, se analizarán los posibles riesgos, siendo estos los posibles problemas futuros que puedan tener un efecto positivo o negativo en los objetivos del proyecto, incluyendo cada uno de ellos un conjunto de causa-efectos.

Para tratar mejor un riesgo, hay que tener en cuenta dos aspectos: cuál es la probabilidad de que ocurra, y cuál es el impacto que tendría en el proyecto en caso de que ocurriese. Una vez estimadas estas dos variables, se puede asignar una posición de ese riesgo dentro de la matriz de la figura 4.1.

Una vez asignada su posición, se deberá priorizar todo riesgo que ocupe una posición superior a la línea de tolerancia, posiciones que poseen un color mas oscuro en la Figura 4.1.

La matriz de la figura 4.1 ha sido rellena con los riesgos identificados en el Apartado 4.3.2, mostrando por tanto, que en este proyecto no se ha detectado ningún riesgo con el que haya que tomar una priorización sobre el resto.

	Alto	R3				
IMPACTO	Significante	R4	R1, R2			
	Moderado	R5, R7, R8				
	Bajo			R6		
		Bajo	Moderado	Significante	Alto	
		PROBABILIDAD				

LINEA DE TOLERANCIA

Figura 4.1: Matriz de priorización de riesgos

4.3.2. Riesgos Identificados

A continuación se clasifican los riesgos que pueden tener un impacto sobre el proyecto:

Riesgo	R-01
Descripción	El desarrollador puede enfermar debido a las condiciones externas.
Impacto	SIGNIFICANTE
Probabilidad	MODERADA
Plan de mitigación	Evitar acciones que atenten contra la salud.
Plan de contingencia	Trabajo desde casa.

Cuadro 4.1: Riesgo 01 - Enfermedad

Riesgo	R-02
Descripción	Mala planificación del proyecto.
Impacto	SIGNIFICANTE
Probabilidad	MODERADA
Plan de mitigación	Reuniones cada dos semanas de seguimiento
Plan de contingencia	Aumentar el tiempo de trabajo hasta alcanzar lo planeado.

Cuadro 4.2: Riesgo 02 - Planificación incorrecta

Riesgo	R-03
Descripción	Pérdida del trabajo elaborado.
Impacto	ALTO
Probabilidad	BAJA
Plan de mitigación	Utilización de varios servicios de control de versiones.
Plan de contingencia	Recuperar versiones más recientes.

Cuadro 4.3: Riesgo 03 - Pérdida del trabajo

Riesgo	R-04
Descripción	Caída de los servidores que alojan el servicio.
Impacto	SIGNIFICANTE
Probabilidad	BAJA
Plan de mitigación	Usar servidores que aseguren un mínimo de disponibilidad.
Plan de contingencia	Despliegue automático cuando el servidor arranque

Cuadro 4.4: Riesgo 04 - Caída de los servidores

Riesgo	R-05
Descripción	Rotura del equipo del dispositivo
Impacto	MODERADO
Probabilidad	BAJA
Plan de mitigación	Disponer de un mínimo de dos dispositivos.
Plan de contingencia	Comprar otro dispositivo.

Cuadro 4.5: Riesgo 05 - Rotura de dispositivo

Riesgo	R-06
Descripción	Desconexión de la red WIFI del hogar.
Impacto	BAJO
Probabilidad	SIGNIFICANTE
Plan de mitigación	Conexión a la red a través de tarjeta SIM.
Plan de contingencia	Recarga del saldo de la tarjeta SIM a través de Internet.

Cuadro 4.6: Riesgo 06 - Desconexión de la red.

Riesgo	R-07
Descripción	Privatización de algún servicio.
Impacto	MODERADO
Probabilidad	BAJA
Plan de mitigación	Implementación del software adaptativa a cualquier servicio, para poder ser sustituido. Conocimiento del funcionamiento de otros servicios alternativos.
Plan de contingencia	Sustitución del servicio.

Cuadro 4.7: Riesgo 07 - Privatización de algún servicio.

4.3.3. Riesgos Acontecidos

Durante el transcurso del proyecto se pudo observar la aparición de ciertos riesgos:

1. Riesgo 01 - Enfermedad

Descrito en la tabla 4.1, el desarrollador del sistema enfermó durante el último tramo del año.

Este riesgo produjo un retraso de dos semanas en la elaboración del proyecto que el desarrollador tuvo que recuperar quitándose horas de sueño, ya que compartía este proyecto con otro trabajo externo.

2. Riesgo 02 - Mala planificación del proyecto

Descrito en la tabla 4.2, al utilizarse una planificación del proyecto mediante un modelo incremental, no se tuvo en cuenta el tiempo necesario para la escritura de la documentación formal, manteniendo una documentación informal a modo de cuaderno de bitácora donde se iban narrando los hechos. Este modelaje incremental hizo que se desarrollase más funciones de las que se esperaban en un principio, por lo que el periodo de escritura de la documentación aumentó notablemente, retrasando como consecuencia la finalización total del proyecto.

3. Riesgo 07 - Privatización de servicios

Descrito en la tabla 4.7, ocurrió cuando la plataforma de nuestro asistente inteligente fue comprada por Sonos.

A fecha de 31 de Enero de 2020, Snips Seeed privatizó sus servicios, bloqueando las funciones y el acceso a entrenamientos de nuestro asistente. Pese a tener analizado como posible riesgo la privatización de algún servicio, no se esperaba que la privatización fuese del asistente en general, ya que se analizó su modelo de negocio, donde se podía observar que tenía una fuerza para posicionarse como una de las primeras potencias en el mundo de los asistentes virtuales en los años venideros, ya que era el único asistente virtual capaz de trabajar sin una conexión fija a Internet.

Sonos vio también ese potencial y quiso hacerlo suyo, comprando la compañía para poseer la mejor baza de entre todos los asistentes con el fin de convertirse en los próximos años en uno de los líderes de este sector tecnológico.

Gracias al plan de contingencia, y a la elaboración de un sistema que no dependa de ningún servicio específico, sino que todos los servicios implementen las funciones que el sistema requiera, el asistente podrá ser sustituido por otro de los nombrados anteriormente, posicionándose como mejor opción el asistente de Mycroft, descrito en el apartado 2.3.3 del documento.

4. Riesgo No Identificado - Pandemia mundial

Riesgo no identificado debido a la poca probabilidad de aparición.

Para poder ser descrito en la tabla 4.7 se debería rediseñar la matriz de riesgos, ya que este riesgo se escapa de toda planificación. Es, o era anteriormente un riesgo de ínfima probabilidad pero con el mayor impacto posible, por lo que sobrepasaría la línea de tolerancia.

A fecha de 11 de Marzo, se catalogó el virus COVID-19 como pandemia mundial [21], estableciendo el toque de queda y la prohibición de realizar la vida tal y como la conocemos. Estas medidas paralizaron la economía del mundo entero, de modo que las empresas no estaban preparadas para mantenerse en un mundo en el que la producción estaba completamente parada.

Las secuelas que este riesgo está dejando, tanto económicas por la paralización del mundo, como psicológicas debido a meses y meses de encerramiento en los hogares, ha producido un retraso en la finalización del proyecto.

Este riesgo ha afectado en el último tramo del proyecto, cuando se estaba finalizando la documentación de la memoria, de modo que el autor del proyecto tuvo que dejarlo de lado debido a su trabajo externo, ya que le fue asignado un proyecto a contrarreloj para poder controlar este virus.

El cúmulo de no poder salir a la calle debido a las restricciones y estar trabajando en ese proyecto a contrarreloj durante meses provocaron problemas de ansiedad que no le han permitido continuar.

El proyecto finalmente se ha retomado a lo largo del mes de Julio, finalizando la memoria, y entregando el proyecto.

4.4. Costes

Para poder dar un valor monetario al proyecto hay que tener en cuenta sus costes, siendo tanto costes económicos como costes de desarrollo para poder ser realizado.

1. Duración estimada del proyecto

El proyecto descrito en el presente documento es un proyecto relativo a un Trabajo de Fin de Grado (TFG), por lo tanto su estimación de horas de elaboración deberán corresponder con el coste de horas estimadas en cuanto al valor hora-crédito de la carrera.

Según la documentación proporcionada por la Universidad de Valladolid [22], cada crédito tiene una estimación de trabajo de 25 horas, y un TFG tiene asignado una cuantía de 12 créditos.

Por tanto, el trabajo estimado para la elaboración de un TFG es:

$$12 \text{ créditos} * 25 \text{ horas/crédito} = 300 \text{ horas}$$

2. Coste de personal:

El salario medio para un desarrollador junior en España que todavía no tiene el título está, en el momento en el cual se está escribiendo el presente documento, en *18966 euros/año* [23], y calculando que al año se hacen unas 1826 horas de trabajo de media, este desarrollador junior debería cobrar aproximadamente:

$$18966 \text{ euros/año} * 1 \text{ año} / 1826 \text{ horas} = 10,38 \text{ euros/hora}$$

Teniendo en cuenta que la elaboración del proyecto tiene una estimación de 300 horas, el coste de personal será de:

$$10,38 \text{ euros/hora} * 300 \text{ horas} = 3116 \text{ euros}$$

3. Coste del equipo:

Para el desarrollo del proyecto se necesita la adquisición de dos dispositivos completos. El coste del kit que proporciona la plataforma de Snips Seed es de 115 euros por cada dispositivo, por lo que sería necesaria la inversión de:

$$115 \text{ euros/dispositivo} * 2 \text{ dispositivos} = 230 \text{ euros}$$

4. Coste de suministros

La duración del proyecto a desarrollar por el desarrollador, teniendo en cuenta que realice una jornada de 8 horas al día, será:

$$\begin{aligned} 1826 \text{ horas/año} / 12 \text{ meses/año} &= 154 \text{ horas/mes} \\ 300 \text{ horas/tfg} / 154 \text{ horas/mes} &= 1,9480 \text{ meses/TFG} \end{aligned}$$

Como el gasto en suministros son cuotas mensuales, se aproxima a 2 meses de trabajo completo la resolución del TFG.

El coste medio de consumo de suministros contando el acceso a internet y el consumo de la luz suma una cuantía de 25 euros al mes. Por tanto, el coste de suministros, sería de:

$$2 \text{ meses/TFG} * 25 \text{ euros/mes} = 50 \text{ euros/TFG}$$

Teniendo en cuenta, que todo proyecto tiende a retrasarse, como indica el libro estandarte de la gestión de proyectos software [24], entre un 8% y un 10%, el coste total deberá ser recalculado, siendo la suma de todos los costes:

$$3116 + 230 + 50 = 3396 \text{ euros}$$

$$3396 \text{ euros} * 10 \% = 3735 \text{ euros}$$

El coste para la elaboración del proyecto, será por tanto la cuantía de **3608 euros**.

Capítulo 5

Diseño

5.1. Casos de Uso

5.1.1. Diagrama de Casos de Uso

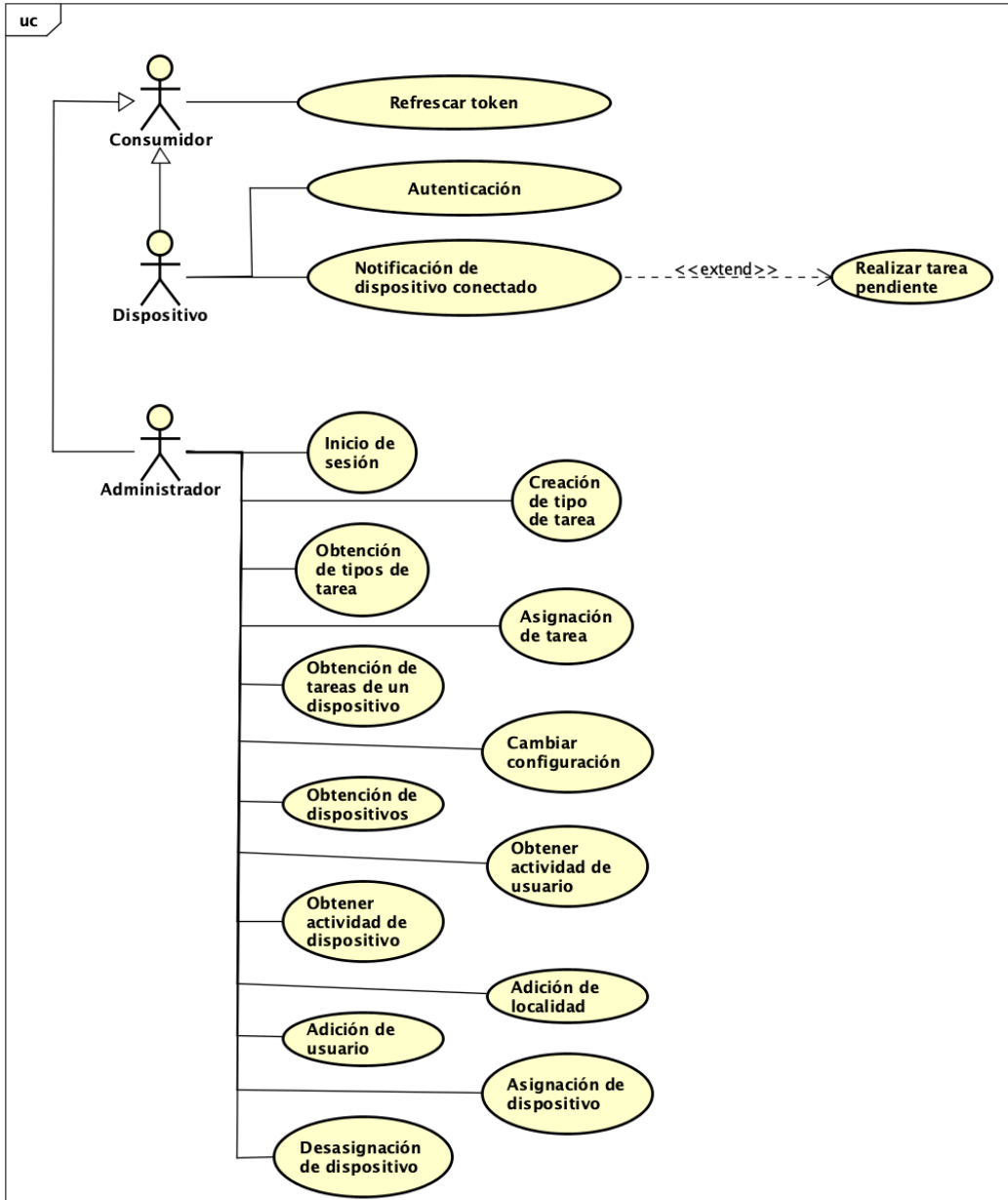


Figura 5.1: Diagrama general de casos de uso

5.1.2. Definición de Casos de Uso

Los casos de uso serán definidos de tal manera que cada caso de uso tendrá un identificador. El identificador de los casos de uso estará formado por el prefijo **CU** el cual indica que es un caso de uso, seguido de una letra que indica quién tiene asignado ese caso de uso, y un número distintivo:

- **U** - Indica que es el consumidor quien realiza ese caso de uso.
- **D** - Indica que es el dispositivo quien realiza ese caso de uso.
- **A** - Indica que es el administrador quien realiza ese caso de uso.

A continuación se definen los casos de uso mostrados en la figura 5.1

CU _D 01	Inicio de sesión
Descripción	Un dispositivo se identifica ante el sistema con el fin de obtener unos tokens de acceso, con los que autenticar posteriormente sus llamadas.
Actor	Dispositivo
Precondiciones	El dispositivo tiene conexión a Internet.
Postcondiciones	Se obtiene un par de tokens.
Flujo normal	<ol style="list-style-type: none"> 1. El dispositivo obtiene su dirección MAC. 2. El dispositivo genera la clave a partir de una encriptación de su MAC, formando sus credenciales, y se las envía al servidor. 3. El servidor comprueba la validez de las credenciales, genera un par de tokens asociados al dispositivo y los devuelve. 4. El dispositivo almacena los tokens y finaliza el caso de uso.
Flujo Alternativo	<p>Flujo alternativo 1:</p> <ol style="list-style-type: none"> 3.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error. 3.A1.2. El dispositivo recibe el error y finaliza el caso de uso. <p>Flujo alternativo 2:</p> <ol style="list-style-type: none"> 3.A2.1. El servidor detecta que las credenciales son válidas pero no hay ningún registro del dispositivo. 3.A2.2. El servidor registra el nuevo dispositivo, genera un par de tokens asociados a su identificador y los devuelve.

Cuadro 5.1: Caso de Uso D01 - Inicio de sesión

CU _U 01	Refresco de token de acceso
Descripción	Un dispositivo o usuario piden al servidor una actualización de sus tokens de acceso.
Actor	Usuario
Precondiciones	1. El usuario tiene conexión a Internet. 2. El dispositivo tiene un token de refresco válido.
Postcondiciones	Se obtiene un par de tokens.
Flujo normal	1. El usuario manda su token de refresco. 2. El sistema comprueba la validez del token de refresco, obtiene la información sobre el usuario, le asocia un par nuevo de tokens y los devuelve. 3. El usuario recibe los tokens nuevos de acceso y finaliza el caso de uso.
Flujo Alternativo	2.A1.1. El sistema comprueba que el token de refresco no es válido y responde un mensaje de error. 2.A1.2. El usuario recibe el error, descarta los tokens que tenía almacenados y finaliza el caso de uso.

Cuadro 5.2: Caso de Uso U01 - Refresco de token de acceso

CU _D 02	Notificación de dispositivo conectado
Descripción	Un dispositivo avisa de que está encendido.
Actor	Dispositivo
Precondiciones	1. El dispositivo tiene conexión a Internet. 2. El dispositivo tiene un par de tokens activos.
Postcondiciones	1. El sistema actualiza su registro.
Flujo normal	1. El dispositivo hace una petición <i>/alive</i> al servidor. 2. El sistema recibe la llamada, valida los credenciales, comprueba que no hay ninguna tarea pendiente, y responde al dispositivo. 3. El dispositivo recibe una respuesta vacía del servidor. 4. Finaliza el caso de uso.
Flujo Alternativo	2.A1.1. El sistema detecta que el dispositivo tiene tareas asignadas pendientes, de modo que las recolecta y devuelve. 2.A1.2. Se realiza el caso de uso <i>Realizar tarea pendiente</i> . 2.A1.3. Finaliza el caso de uso.

Cuadro 5.3: Caso de Uso D02 - Notificación de dispositivo conectado.

CU _D 03	Realización de tarea pendiente
Descripción	El dispositivo ejecuta una acción pendiente.
Actor	Dispositivo
Precondiciones	1. El dispositivo tiene conexión a Internet. 2. El dispositivo ha recibido una lista de tareas pendientes.
Postcondiciones	1. El sistema actualiza su registro.
Flujo normal	1. El dispositivo recibe una lista de tareas pendientes. 2. El dispositivo avisa al servidor que va a hacer la tarea. 3. El servidor marca la tarea como realizada y actualiza su registro del estado del dispositivo. 3. El dispositivo hace la tarea. 4. Finaliza el caso de uso.
Flujo Alternativo	

Cuadro 5.4: Caso de Uso D03 - Realización de tarea pendiente.

CU _D 04	Actualización de configuración
Descripción	El dispositivo actualiza su configuración.
Actor	Dispositivo
Precondiciones	1. El dispositivo tiene conexión a Internet. 2. El dispositivo tiene un token de acceso válido.
Postcondiciones	1. El dispositivo actualiza su fichero de configuración. 2. El servidor actualiza el registro de configuraciones.
Flujo normal	1. El dispositivo solicita al servidor obtener la configuración. 2. El servidor obtiene la configuración añadida recientemente que aún no haya recogido el dispositivo, obteniendo la identificación del dispositivo a partir de su token de acceso, y se la devuelve. 3. El dispositivo obtiene su configuración, actualiza el fichero donde la guarda e informa al servidor acerca de la completitud. 4. El servidor actualiza sus registros de actualizaciones e informa al dispositivo. 5. El dispositivo recibe esa respuesta y finaliza el caso de uso.
Flujo Alternativo	2.A1.1. El servidor detecta que el dispositivo tiene ya la configuración más reciente por lo que le devuelve un mensaje vacío. 2.A1.2. El dispositivo recibe el mensaje vacío, y finaliza el caso de uso.

Cuadro 5.5: Caso de Uso D04 - Actualización de configuración

CU _D 05	Envío de actividad de un dispositivo
Descripción	El dispositivo envía su actividad al servidor.
Actor	Dispositivo
Precondiciones	1. El dispositivo tiene conexión a Internet. 2. El dispositivo tiene un token de acceso válido. 3. El dispositivo ha realizado alguna actividad con el usuario.
Postcondiciones	1. El servidor actualiza su registro de actividades.
Flujo normal	1. El dispositivo procesa una nueva actividad y la envía al servidor. 2. El servidor autentica al dispositivo a partir de su token, comprueba la validez de los datos enviados, registra la actividad asignando una marca de tiempo a dicha actividad, y devuelve un mensaje de confirmación al dispositivo. 3. El dispositivo recibe esa respuesta y finaliza el caso de uso.
Flujo Alternativo	2.A1.1. El servidor detecta que la autenticación no es válida y devuelve error. 2.A1.2. El dispositivo recibe un mensaje no válido, y devuelve error.

Cuadro 5.6: Caso de Uso D05 - Envío de actividad de un dispositivo

CU _A 01	Inicio de sesión
Descripción	Un administrador se identifica ante el sistema con el fin de obtener unos tokens de acceso, con los que autenticar posteriormente sus llamadas.
Actor	Administrador
Precondiciones	El administrador tiene conexión a Internet.
Postcondiciones	El administrador obtiene un par de tokens.
Flujo normal	1. El administrador envía sus credenciales al servidor. 2. El servidor comprueba la validez de las credenciales, genera un par de tokens asociados al administrador, y los devuelve. 3. El administrador almacena los tokens y finaliza el caso de uso.
Flujo Alternativo	2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error. 2.A1.2. El administrador recibe el error y finaliza el caso de uso.

Cuadro 5.7: Caso de Uso A01 - Inicio de sesión

CU _{A02}	Creación de un nuevo tipo de tarea
Descripción	Un administrador crea un nuevo tipo de tarea que será asignable a un dispositivo.
Actor	Administrador
Precondiciones	El administrador posee un token de acceso activo.
Postcondiciones	Se registra un nuevo tipo de tarea.
Flujo normal	<ol style="list-style-type: none"> 1. El administrador envía un nuevo tipo de tarea y el contenido de esta al servidor. 2. El servidor autentica la petición, comprueba el contenido recibido, crea un nuevo tipo de tarea, e informa sobre la completitud. 3. El administrador recibe el mensaje y finaliza el caso de uso.
Flujo Alternativo	<p>Flujo Alternativo 1: 2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error.</p> <p>Flujo Alternativo 2: 2.A2.1. El servidor detecta un error en la información introducida y devuelve error.</p>

Cuadro 5.8: Caso de Uso A02 - Creación de un nuevo tipo de tarea.

CU _{A03}	Asignación de tarea
Descripción	Un administrador asigna una nueva tarea a un dispositivo específico.
Actor	Dispositivo
Precondiciones	1. El administrador está autenticado ante el sistema.
Postcondiciones	1. Se registra una nueva tarea como pendiente para un dispositivo específico.
Flujo normal	<ol style="list-style-type: none"> 1. El administrador selecciona un dispositivo específico, selecciona una tarea, y la envía. 2. El sistema recibe una nueva asignación de tarea, comprueba que es una petición autenticada, asigna la tarea como pendiente para el dispositivo y responde como petición correcta. 3. El administrador recibe la respuesta y finaliza el caso de uso.
Flujo Alternativo	

Cuadro 5.9: Caso de Uso A03 - Asignación de nueva tarea.

CU _{A04}	Recolectar tareas existentes
Descripción	Un administrador obtiene una lista con todas las tareas existentes asignables a dispositivos.
Actor	Administrador
Precondiciones	El administrador posee un token de acceso activo.
Postcondiciones	
Flujo normal	<ol style="list-style-type: none"> 1. El administrador solicita al servidor la lista de tipos de tareas existentes. 2. El servidor autentica la petición, reúne los tipos de tareas existentes, y se las devuelve al administrador finalizando la llamada. 3. El administrador recibe el mensaje con la lista de tipos de tareas y finaliza el caso de uso.
Flujo Alternativo	<p>Flujo Alternativo 1: 2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error.</p> <p>Flujo Alternativo 2: 2.A2.1. El servidor detecta un error en la información introducida y devuelve error.</p>

Cuadro 5.10: Caso de Uso A04 - Obtención de tipos de tarea.

CU _{A05}	Obtención de tareas de un dispositivo
Descripción	Un administrador obtiene una lista con todas las tareas asignadas a un dispositivo en un periodo de tiempo concreto.
Actor	Administrador
Precondiciones	El administrador posee un token de acceso activo.
Postcondiciones	
Flujo normal	<ol style="list-style-type: none"> 1. El administrador solicita al servidor la lista de tareas asignadas a un dispositivo entre dos fechas específicas. 2. El servidor autentica la petición, reúne las tareas, y se las devuelve al administrador, finalizando la llamada. 3. El administrador recibe la respuesta con la lista de tareas y finaliza el caso de uso.
Flujo Alternativo	<p>Flujo Alternativo 1: 2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error.</p> <p>Flujo Alternativo 2: 2.A2.1. El servidor detecta un error en la información introducida y devuelve error.</p>

Cuadro 5.11: Caso de Uso A05 - Obtención de tareas de un dispositivo.

CU _{A06}	Cambiar configuración
Descripción	Un administrador registra un cambio de configuración de un dispositivo, de una localidad o del sistema global.
Actor	Administrador
Precondiciones	El administrador posee un token de acceso activo.
Postcondiciones	Se registra una nueva configuración.
Flujo normal	<ol style="list-style-type: none"> 1. El administrador envía al servidor una nueva configuración, especificando si es para un dispositivo, para una localidad, o una configuración global. 2. El servidor autentica la petición, comprueba la validez de la configuración, registra esta en el sistema, y devuelve una mensaje a modo de completitud. 3. El administrador recibe el mensaje y finaliza el caso de uso.
Flujo Alternativo	<p>Flujo Alternativo 1: 2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error.</p> <p>Flujo Alternativo 2: 2.A2.1. El servidor detecta un error en la información obtenida y devuelve error.</p>

Cuadro 5.12: Caso de Uso A06 - Cambiar configuración de dispositivos.

CU _A 07	Obtención de todos los dispositivos
Descripción	Un administrador solicita la información de todos los dispositivos del sistema.
Actor	Administrador
Precondiciones	El administrador posee un token de acceso activo.
Postcondiciones	
Flujo normal	<ol style="list-style-type: none"> 1. El administrador solicita al servidor la lista global de dispositivos. 2. El servidor autentica la petición, recolecta una lista de dispositivos donde incluye las tareas pendientes, las últimas tareas realizadas e información sobre su usuario asignado, y devuelve la lista al administrador. 3. El administrador recibe los datos y finaliza el caso de uso.
Flujo Alternativo	<p>Flujo Alternativo 1: 2.A.1. El servidor comprueba que los credenciales no son válidos y devuelve error.</p> <p>Flujo Alternativo 2: 2.B.1. El servidor detecta un error en la información obtenida y devuelve error.</p>

Cuadro 5.13: Caso de Uso A07 - Obtención de todos los dispositivos.

CU _{A08}	Obtención de actividad de un dispositivo
Descripción	Un administrador solicita la información más específica de un dispositivo del sistema.
Actor	Administrador
Precondiciones	1. El administrador posee un token de acceso activo. 2. El dispositivo no tiene ningún usuario asignado.
Postcondiciones	1. El administrador recibe toda la actividad del dispositivo sin identificar el usuario que la realizó.
Flujo normal	1. El administrador solicita al servidor la información de uso de un dispositivo ocurrida en un rango de fechas específico. 2. El servidor autentica la petición, recolecta una lista de dispositivos donde incluye las tareas pendientes, las últimas tareas realizadas e información sobre su usuario asignado, y devuelve la lista al administrador. 3. El administrador recibe los datos y finaliza el caso de uso.
Flujo Alternativo	Flujo Alternativo 1: 2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error. Flujo Alternativo 2: 2.A2.1. El servidor detecta un error en la información obtenida y devuelve error.

Cuadro 5.14: Caso de Uso A08 - Obtención de actividad de un dispositivo.

CU _{A09}	Obtención de actividad de un usuario
Descripción	Un administrador solicita la información más específica de un usuario del sistema.
Actor	Administrador
Precondiciones	El administrador posee un token de acceso activo.
Postcondiciones	1. El administrador recibe los datos de usuario ocurridos entre dos fechas. 2. Los datos recibidos solo son los realizados en su dispositivo actual por el usuario en cuestión.
Flujo normal	1. El administrador solicita al servidor la información de uso de un dispositivo ocurrida en un rango de fechas específico. 2. El servidor autentica la petición, comprueba el formulario de usuario recibido, recolecta la información y responde a la llamada con ella. 3. El administrador recibe los datos y finaliza el caso de uso.
Flujo Alternativo	Flujo Alternativo 1: 2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error. Flujo Alternativo 2: 2.A2.1. El servidor detecta un error en la información obtenida y devuelve error.

Cuadro 5.15: Caso de Uso A09 - Obtención de actividad de un usuario

CU _{A10}	Adición de una nueva localidad
Descripción	Un administrador añade una nueva localidad al sistema.
Actor	Administrador
Precondiciones	1. La localidad no debe existir ya en el sistema. 2. El administrador debe poseer el nombre y código postal de la localidad.
Postcondiciones	La localidad es añadida al sistema, guardando su nombre y código postal.
Flujo normal	<ol style="list-style-type: none"> 1. El administrador envía al servidor el nombre y código postal de una localidad. 2. El servidor autentica la petición, comprueba el formulario de usuario recibido, registra la localidad y responde al administrador. 3. El administrador recibe la respuesta y finaliza el caso de uso.
Flujo Alternativo	<p>Flujo Alternativo 1: 2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error.</p> <p>Flujo Alternativo 2: 2.A2.1. El servidor detecta un error en la información obtenida y devuelve error.</p>

Cuadro 5.16: Caso de Uso A10 - Adición de una nueva localidad.

CU _A 11	Adición de nuevo usuario
Descripción	Un administrador añade un nuevo usuario en el sistema.
Actor	Administrador
Precondiciones	<ol style="list-style-type: none"> 1. El administrador posee un token de acceso activo. 2. El sistema no debe tener almacenado ningun usuario con el mismo DNI.
Postcondiciones	El usuario es añadido al sistema, teniendo una localidad asociada.
Flujo normal	<ol style="list-style-type: none"> 1. El administrador envía al servidor el dni, nombre y código postal de un nuevo usuario. 2. El servidor autentica la petición, comprueba el formulario de usuario recibido, añade el usuario al sistema, y responde la llamada. 3. El administrador recibe la respuesta y finaliza el caso de uso.
Flujo Alternativo	<p>Flujo Alternativo 1:</p> <ol style="list-style-type: none"> 2.A1.1. El servidor comprueba que los credenciales no son válidas y devuelve error. <p>Flujo Alternativo 2:</p> <ol style="list-style-type: none"> 2.A2.1. El servidor detecta un error en la información obtenida y devuelve error.

Cuadro 5.17: Caso de Uso A11 - Adición de un nuevo usuario.

CU _{A12}	Asignación de dispositivo a un usuario
Descripción	Un administrador solicita la información más específica de un usuario del sistema.
Actor	Administrador
Precondiciones	<ol style="list-style-type: none"> 1. El administrador posee un token de acceso activo. 2. El usuario no debe tener ningún dispositivo asociado en ese momento. 3. El dispositivo no debe tener ningún usuario asociado en ese momento.
Postcondiciones	El dispositivo tiene una asignación activa con el usuario.
Flujo normal	<ol style="list-style-type: none"> 1. El administrador envía al servidor el usuario y dispositivo específico. 2. El servidor autentica la petición, comprueba el formulario de usuario recibido, asigna el dispositivo al usuario, y responde al administrador. 3. El administrador recibe el mensaje y finaliza el caso de uso.
Flujo Alternativo	<p>Flujo Alternativo 1:</p> <p>2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error.</p> <p>Flujo Alternativo 2:</p> <p>2.A2.1. El servidor detecta un error en la información obtenida y devuelve error.</p>

Cuadro 5.18: Caso de Uso A12 - Asignación de dispositivo a un usuario.

CU _A 13	Desasignación del dispositivo de un usuario
Descripción	Un administrador desasigna el dispositivo a un usuario.
Actor	Administrador
Precondiciones	1. El administrador posee un token de acceso activo. 2. El usuario tiene un dispositivo asociado.
Postcondiciones	Finaliza la asignación de un dispositivo a un usuario
Flujo normal	1. El administrador solicita al servidor la finalización de una asignación entre un dispositivo y un usuario. 2. El servidor autentica la petición, comprueba los datos de usuario recibido, finaliza la asignación y responde la llamada. 3. El administrador recibe la respuesta y finaliza el caso de uso.
Flujo Alternativo	Flujo Alternativo 1: 2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error. Flujo Alternativo 2: 2.A2.1. El servidor detecta un error en la información obtenida y devuelve error.

Cuadro 5.19: Caso de Uso A13 - Desasignación de un dispositivo a un usuario.

CU _A 14	Obtención de usuario de un dispositivo específico
Descripción	Un administrador obtiene los datos de usuario asignado a un dispositivo concreto.
Actor	Administrador
Precondiciones	1. El administrador posee un token de acceso activo.
Postcondiciones	Finaliza la asignación de un dispositivo a un usuario
Flujo normal	1. El administrador solicita al servidor la información sobre el usuario asociado a un determinado dispositivo. 2. El servidor autentica la petición, recolecta la información asociada al usuario que tiene asignado ese determinado dispositivo y se la devuelve al administrador. 3. El administrador recibe la información y finaliza el caso de uso.
Flujo Alternativo	Flujo Alternativo 1: 2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error. Flujo Alternativo 2: 2.A2.1. El servidor detecta un error en la información procesada y devuelve error.

Cuadro 5.20: Caso de Uso A14 - Obtención de usuario de un dispositivo específico.

CU _{A15}	Obtención de localidades
Descripción	Un administrador obtiene la lista de las localidades junto a información relevante sobre cada una.
Actor	Administrador
Precondiciones	1. El administrador posee un token de acceso activo.
Postcondiciones	
Flujo normal	1. El administrador solicita al servidor la lista de localidades. 2. El servidor autentica la petición, recolecta la información sobre todas las localidades y se la envía al administrador. 3. El administrador recibe la información y finaliza el caso de uso.
Flujo Alternativo	2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error.

Cuadro 5.21: Caso de Uso A15 - Obtención de localidades.

CU _{A16}	Obtención de la lista de usuarios
Descripción	Un administrador obtiene una lista con todos los usuarios del sistema, indicando en cada uno información como su localidad y dispositivo asociado en caso de que lo tenga.
Actor	Administrador
Precondiciones	1. El administrador posee un token de acceso activo.
Postcondiciones	
Flujo normal	1. El administrador solicita al servidor la lista de usuarios registrados. 2. El servidor autentica la petición, recolecta la información y se la devuelve al administrador. 3. El administrador recibe la información y finaliza el caso de uso.
Flujo Alternativo	Flujo Alternativo 1: 2.A1.1. El servidor comprueba que los credenciales no son válidos y devuelve error. Flujo Alternativo 2: 2.A2.1. El servidor detecta un error en la información procesada y devuelve error.

Cuadro 5.22: Caso de Uso A16 - Obtención de la lista de usuarios registrados.

5.2. Diagramas de Secuencia

5.2.1. Introducción

En esta sección se van a plasmar los diagramas de secuencia con una breve explicación de lo que está sucediendo con el fin de que narrando los hechos se comprenda de una manera más sencilla qué es lo que se está representando y facilitando en un futuro una reimplementación.

5.2.2. Relativos a interacción con el Dispositivo

Flujo del Dispositivo

En este diagrama se puede observar como fluye el dispositivo entre sus distintos estados.

La secuencia representada muestra como el dispositivo arranca directamente este servicio, el cual inicia sesión, o permanece intentándolo, para posteriormente enviar su estado.

Se muestra el flujo tanto del caso de uso de *inicio de sesión del dispositivo*, como del caso de uso de *notificación de dispositivo conectado*, que están representados en los cuadros D01 y D02, respectivamente.

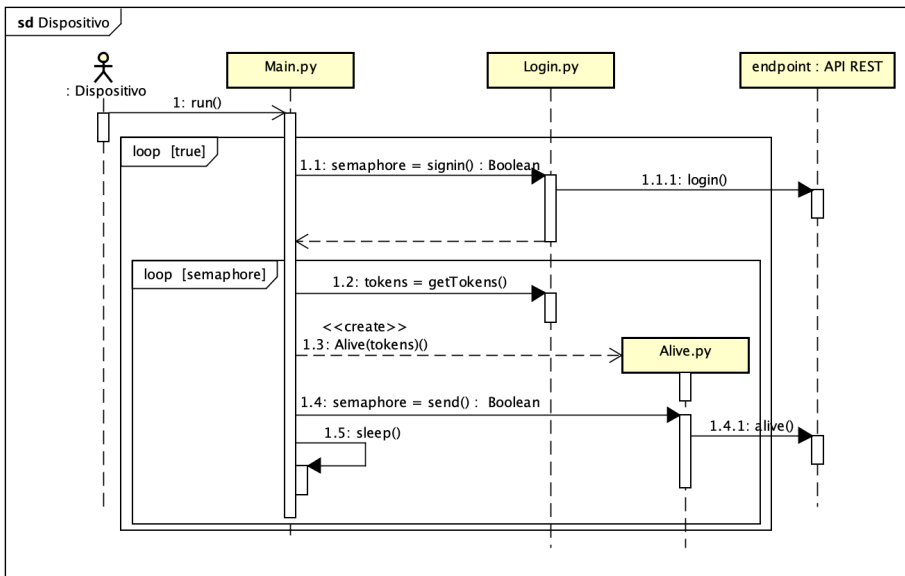


Figura 5.2: Diagrama de secuencia DS01 - Dispositivo

Petición de Inicio de sesión

En el siguiente diagrama de secuencia, el *DS02*, se muestra la continuación del diagrama de secuencia anterior, haciendo incapié a qué es lo que ocurre en el lado del servidor una vez el dispositivo intenta iniciar sesión.

Como se puede observar, a un endpoint del servicio REST desplegado le llega una petición de inicio de sesión: un flujo correcto muestra cómo dentro del servidor, es el *DeviceService* quien comprueba los credenciales, que siendo correctos sigue el flujo de secuencia y se solicita unos nuevos tokens para un dispositivo concreto al servicio de autenticación.

Si observamos la capa de dominio del servidor, que llamaremos *core*, está basada en la implementación de unas interfaces que están definidas en el propio core. De esta forma cualquier servicio que en un futuro se quiera usar deberá implementar estas interfaces. Así se asegura la integridad del sistema, permitiendo cambiar de servicio en un futuro sin necesidad de tocar el core, manteniendo la misma funcionalidad del sistema.

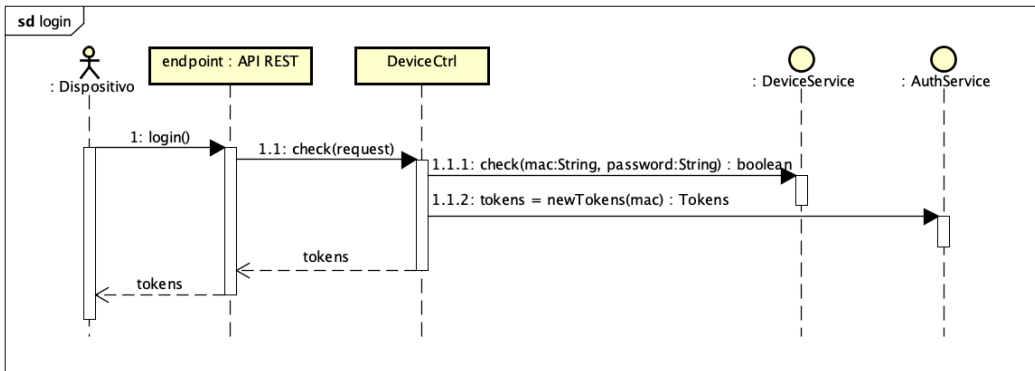


Figura 5.3: Diagrama de secuencia DS02 - Login

Petición de Dipoitvo conectado

El diagrama de secuencia *DS03*, representado en la figura 5.5, muestra también el comportamiento del servidor tras otra petición.

En este caso se trata de una petición de aviso de que el dispositivo está conectado, documentada en el cuadro *D02*, la cual se puede observar que tiene finalmente dos posibles finales correctos: la finalización, y la realización del caso de uso de *realización de tarea pendiente*, el cual está plasmado en el caso de uso *D03*.

En este diagrama se expone cómo el dispositivo realiza la petición, y tras esa petición, el servidor se encarga de comprobar primero si el token del dispositivo es correcto para posteriormente decirle al controlador de tareas, *TaskCtrl*, que el dispositivo acaba de realizar la tarea de mostrar que está conectado.

El controlador de tareas almacena el nuevo estado del dispositivo para posteriormente recuperar todas las tareas pendientes, que en caso de no haber alguna, se responde al dispositivo con un código de estado HTTP cuyo valor es 200, finalizando el caso de uso. En cambio, si el controlador sí que ha recibido tareas pendientes, se le responde al dispositivo un código 300, disparando un nuevo caso de uso en el dispositivo, el cual es la realización de tareas pendientes, como se ha nombrado anteriormente.

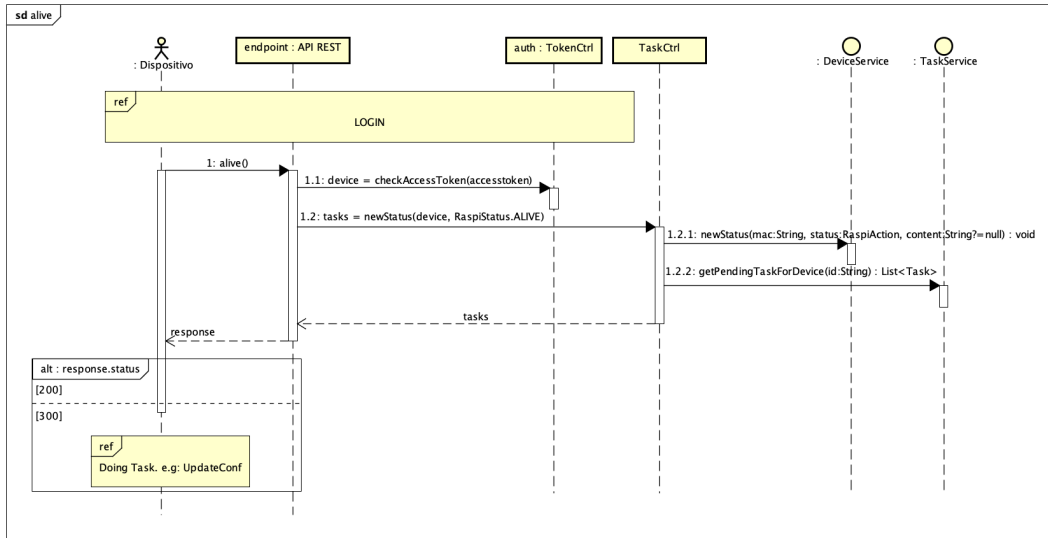


Figura 5.4: Diagrama de secuencia DS03 - Alive

Realización de Tareas pendientes

El presente diagrama de secuencia muestra el un prototipo de flujo válido a implementar en los dispositivos para implementación del caso de uso de *Realización de tarea pendiente D03*, siendo válido para cualquier tipo de tarea, que en este caso representará a la tarea configuración, la cual está asociada también al caso de uso de *actualización de la configuración D04*, ya que esta tarea es un requisito indispensable del sistema, y de este modo se puede mostrar mejor la combinación de ambas dando como resultado la posibilidad de actualizar y controlar remotamente el dispositivo.

En esta representación gráfica de la secuencia por la cual se realiza el caso de uso, se observa que previamente el dispositivo ha recibido una respuesta del servidor a una *petición de dispositivo conectado* la cual contiene un código de estado de código 300, la cual se comprueba y se obtiene del cuerpo del mensaje las tareas pendientes.

Una vez se tienen las tareas, se forma un comando, como se narra en el apartado 6.3.1, y es ejecutado contra el sistema.

Este comando hace que el sistema ejecute un script, *init.sh*, el cual arranca la realización

5.2. DIAGRAMAS DE SECUENCIA

de la tarea, en este caso la de actualizar la configuración, la cual está implementada en *Conf.py*.

La tarea primero avisa al servidor que procede a realizar la tarea pendiente, que si recibe un mensaje correcto -*código 200*- procede a realizar la tarea solicitada.

Por tanto, la tarea de configuración solicitará al servidor el archivo de configuración, que en caso de recibirlo, actualizará el fichero.

Una vez actualizado, si todo ha sido correcto, avisa al servidor sobre su finalización, de manera que el servidor marcará la tarea como realizada.

En esta secuencia se puede observar un protocolo de actuación a la hora de tratar las tareas:

1. Avisar que se va a hacer la tarea.
2. Hacer la tarea.
3. Avisar que se ha hecho la tarea.

Gracias a este protocolo un administrador puede observar si una tarea no ha sido realizada porque no le ha llegado, o porque no ha sido capaz de realizarla, dando la posibilidad a una mejor gestión de los problemas.

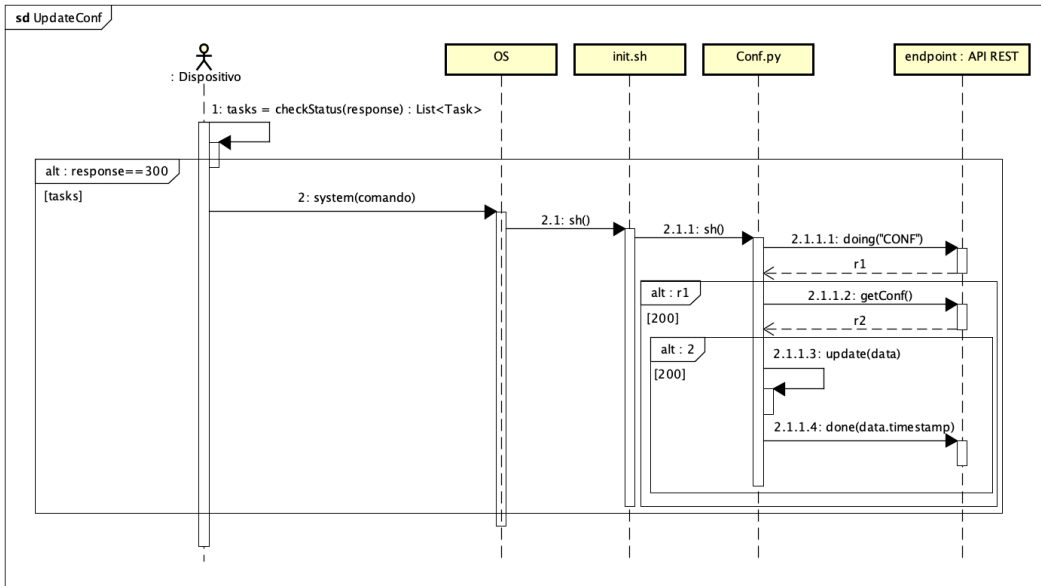


Figura 5.5: Diagrama de secuencia DS04 - Realización de tareas pendientes

Marcar Tarea como Realizada

Tras comprobar en el servidor la validez del token con el cual se hace la petición y obtener el dispositivo asociado, se almacena un nuevo estado del dispositivo, para su visualización de a qué hora se ha realizado la acción y poder observar su actividad. Posteriormente se marca esa tarea como finalizada, asignándole la hora en la cual se ha recibido en el servidor.

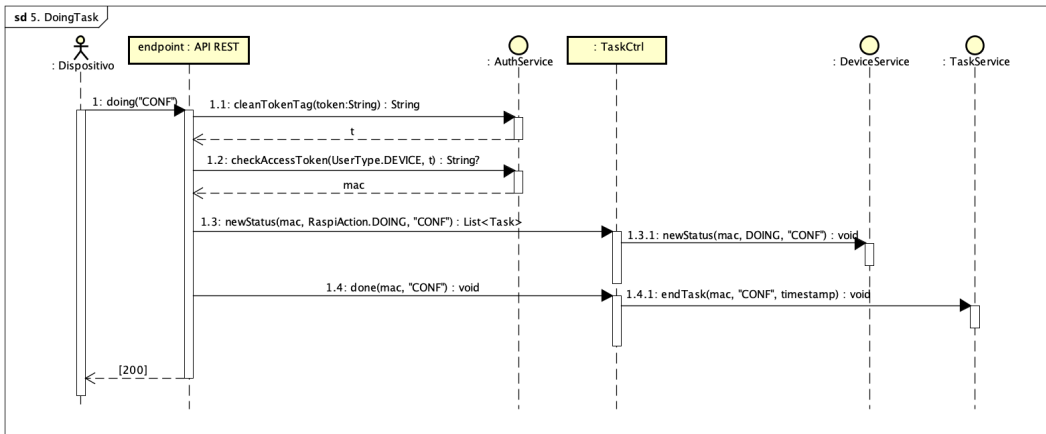


Figura 5.6: Diagrama de secuencia DS05 - Servidor : Marcar tarea como realizada

Marcar la Configuración de un dispositivo como Actualizada

La siguiente secuencia representa una de las acciones con mayor complejidad del sistema, ya que poder llevar el registro de cual es la configuración que posee cada dispositivo y a cual es a la que se ha actualizado, teniendo en cuenta que hay 3 posibles configuraciones paralelas, lleva un tedioso trabajo, que una vez clarificado no es difícil de entender.

Partimos de que el dispositivo informa que ha actualizado a la configuración que posee una marca de tiempo específica.

Entonces, el sistema obtiene las configuraciones globales, locales y propias que corresponden a cada dispositivo y comprueba:

Si el timestamp corresponde con la global, elimina todas las configuraciones globales, ya estén pendientes o no, asignadas a ese dispositivo, y añade la actualizada como no pendiente.

Si no corresponde con la global, si no que con la local, elimina todas las configuraciones locales asignadas a ese dispositivo, estén pendientes o no, y añade la actualizada como no pendiente.

Si tampoco corresponde con la local, si no que corresponde con la propia del dispositivo, elimina la configuración no pendiente asignada, y cambia el estado de la pendiente.

5.2. DIAGRAMAS DE SECUENCIA

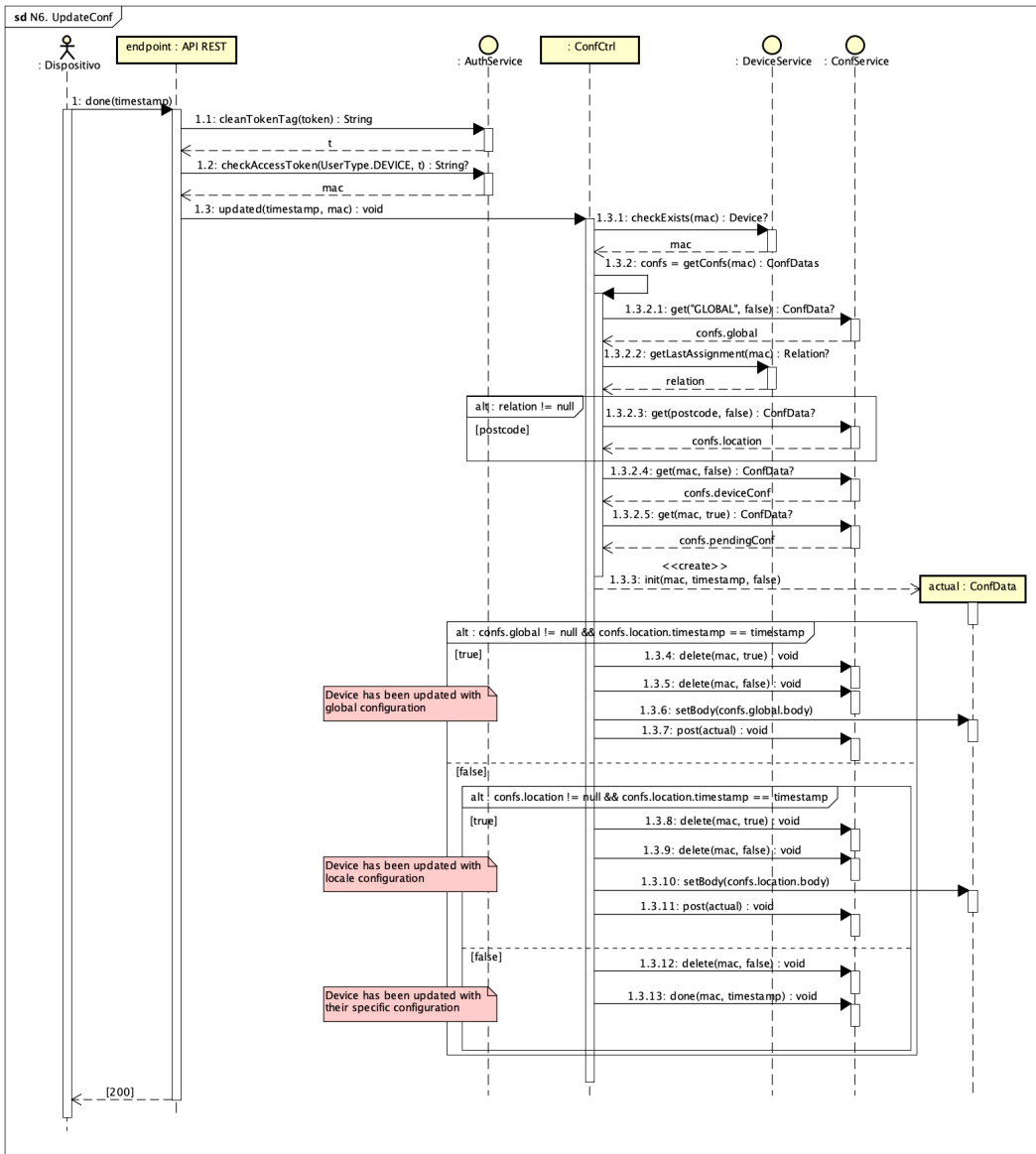


Figura 5.7: Diagrama de secuencia DS06 - Servidor : Configuración actualizada

Obtención de la Configuración Actual

Para obtener la configuración de un dispositivo, el sistema simplemente obtiene el identificador correspondiente al token de la cabecera, del cual recolecta los datos, como si tiene ese dispositivo un usuario asociado.

A partir de esos datos, recoge la configuración global, la local a partir de la asignación de usuario si posee, y la del propio dispositivo, tanto la pendiente como la instalada. De todas estas configuraciones se queda con la que tenga el timestamp más actual, de modo que será la que debe ser enviada.

Si la marca de tiempo de la que se va a devolver es más reciente que la que está guardada como la actual del dispositivo, se marca la que se devuelve como que tiene pendiente su instalación.

5.2. DIAGRAMAS DE SECUENCIA

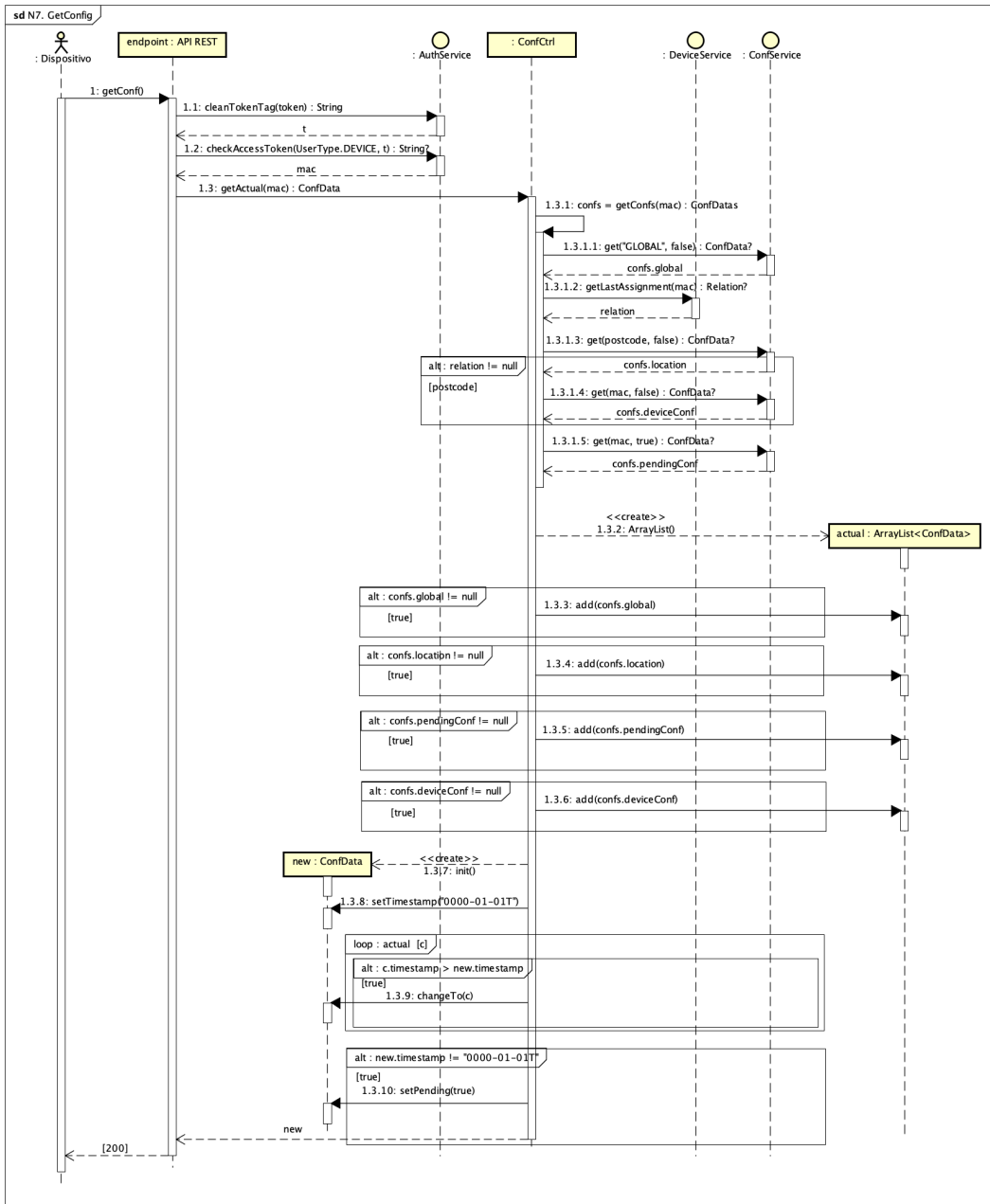


Figura 5.8: Diagrama de secuencia DS07 - Servidor : Obtener configuración actual

5.2.3. Relativos a la Interacción de un Administrador

Creación de un nuevo Tipo de Tarea

La creación de nuevas tareas en el dispositivo ha sido explicado previamente en el apartado 6.3.1, pero para poder mandar al dispositivo esa nueva tarea, hay que poder registrarla en el sistema.

En la siguiente secuencia se muestran los pasos por los cuales un nuevo tipo de tarea puede ser registrada, permitiendo posteriormente la asignación a los dispositivos.

En este caso, para diferenciar entre la creación de tipos de tareas y la creación de una asignación de tarea a los dispositivos, modificaremos el nombre de referencia, llamando a los tipos de tareas *eventos*, y a la asignación de dichos eventos a los dispositivos, *tareas*.

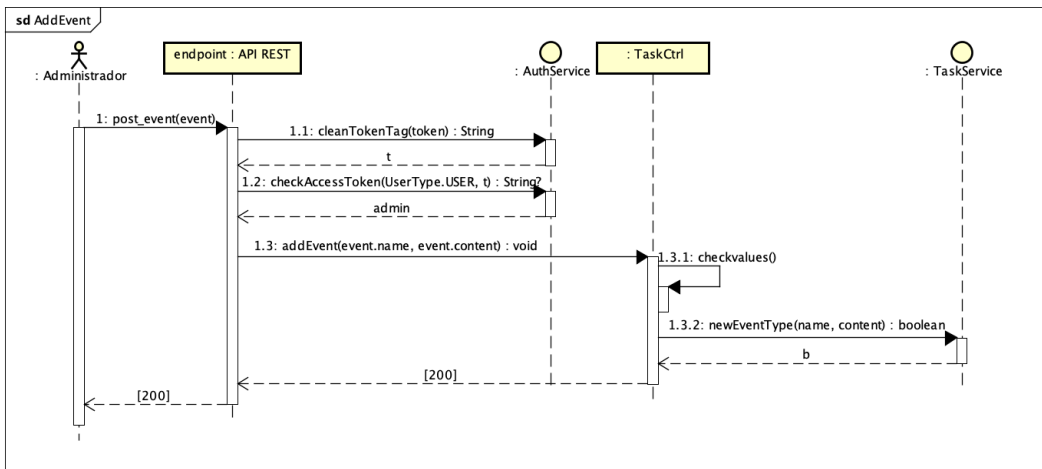


Figura 5.9: Diagrama de secuencia DS08 - Creación de nuevo tipo de tarea

Asignación de una Tarea a un Dispositivo

Una vez se ha mostrado cómo puede ser la estructura del dispositivo para realizar las tareas, y se ha dado la posibilidad de crear eventos en el servidor, solo falta mostrar como son asignables esos eventos a los dispositivos generando nuevas tareas.

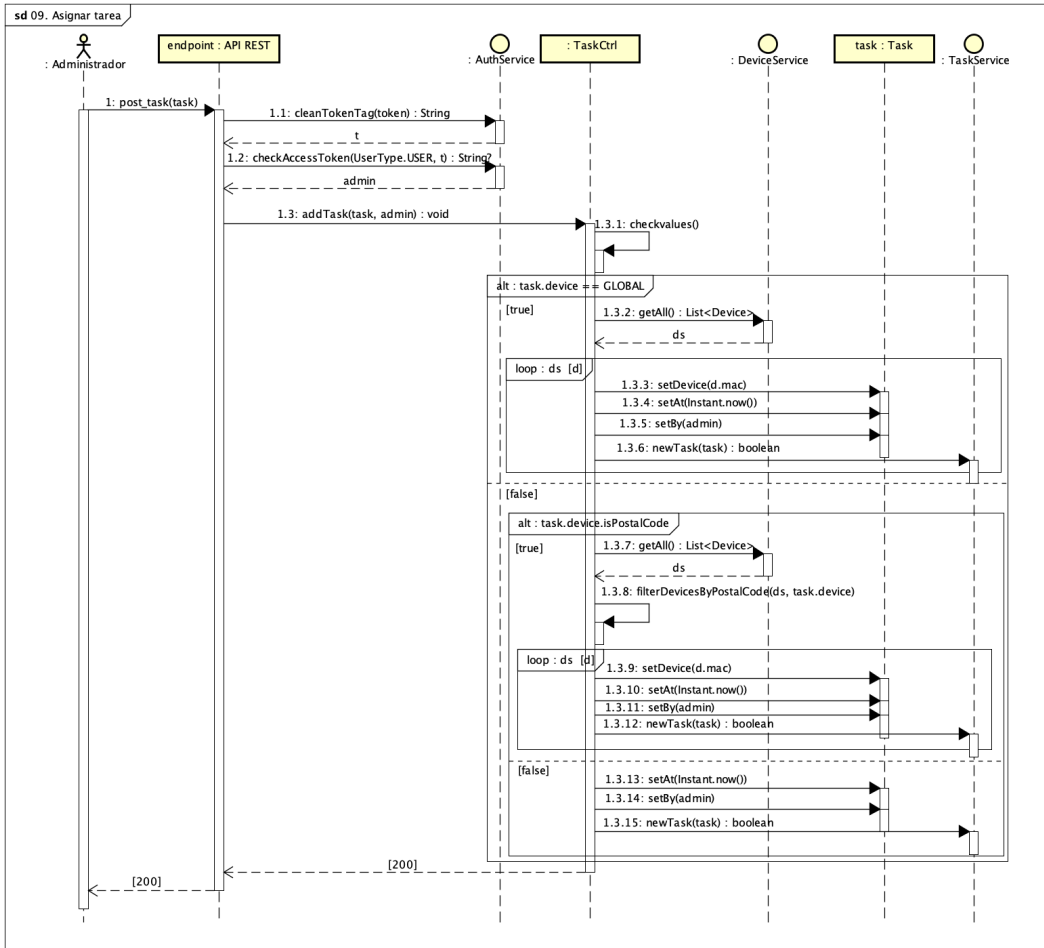


Figura 5.10: Diagrama de secuencia DS09 - Asignación de tarea a dispositivo

Recolección de Eventos

Como bien se ha definido en los requisitos y se ha estipulado en el caso de uso *A04*, un administrador debe poder obtener la lista completa de eventos del sistema, con el fin de saber cuáles están ya definidos y pueden ser asignados a los dispositivos.

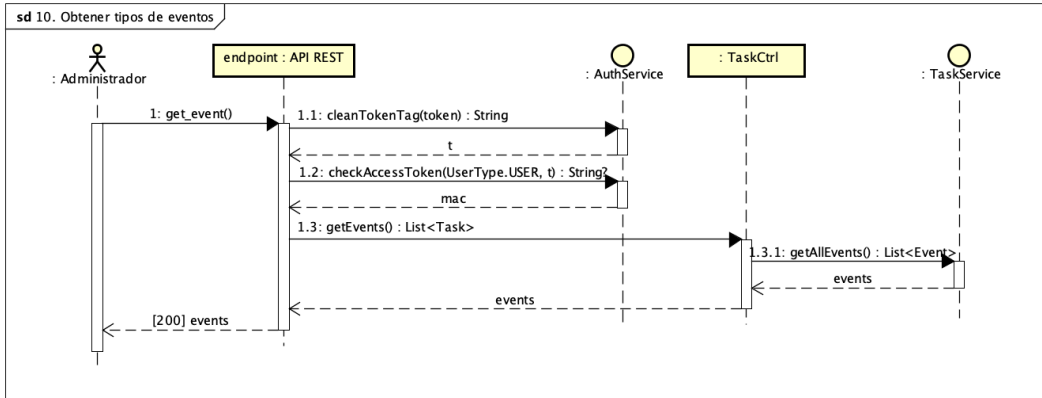


Figura 5.11: Diagrama de secuencia DS10 - Recolección de eventos

Obtención de Tareas de un Dispositivo

En los requisitos funcionales se data la necesidad de un método que permita la obtención de todas las tareas que han sido asignadas a un dispositivo en un determinado rango de tiempo, por lo que en el siguiente diagrama de secuencia se muestra la implementación.

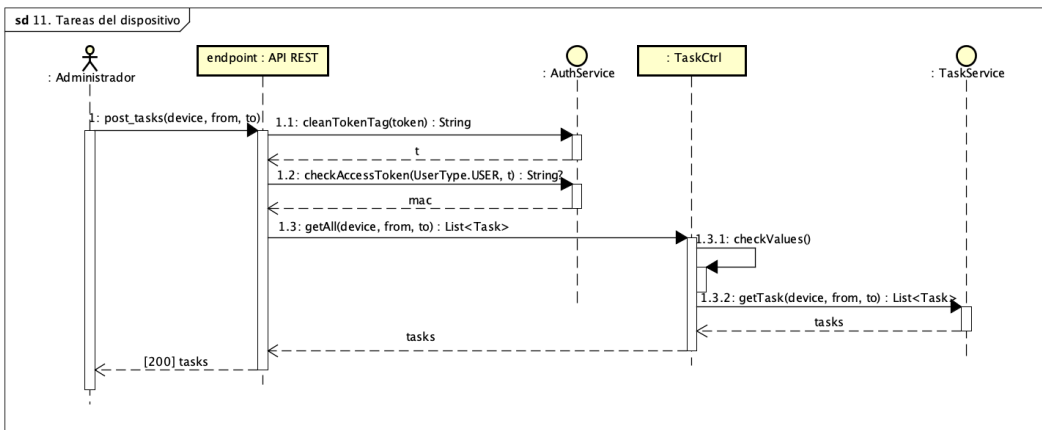


Figura 5.12: Diagrama de secuencia DS11 - Obtención de tareas de un dispositivo

Cambiar Configuración

El administrador es capaz de cambiar la configuración tanto global, como local o de un dispositivo específico a través de una llamada a un endpoint del servicio REST del sistema. Lo que ocurre en el sistema una vez se realiza la petición es documentado en el siguiente diagrama de secuencia.

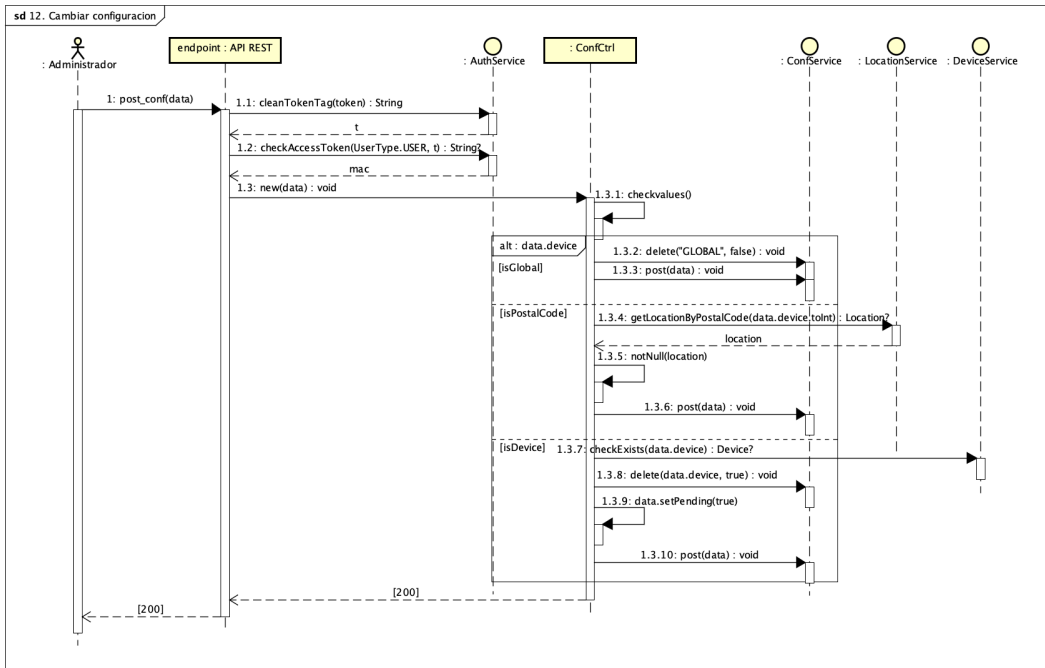


Figura 5.13: Diagrama de secuencia DS12 - Cambiar configuración

Obtención de todos los Dispositivos

Para poder llevar un registro del estado actual de los dispositivos y ver si alguno está dando algún tipo problema es necesario poder obtener todos los dispositivos. El siguiente diagrama de secuencia muestra cómo el sistema recoge primero todos los dispositivos que contiene, para posteriormente añadir a cada uno de ellos información como cuáles han sido los últimos estados que ha enviado, o cuáles son las últimas 5 tareas que ha realizado. También, cuáles han sido los últimos *intents* que ha realizado, representando un intent a la interacción entre usuario y dispositivo, al igual que obtiene las tareas pendientes que tiene cada dispositivo.

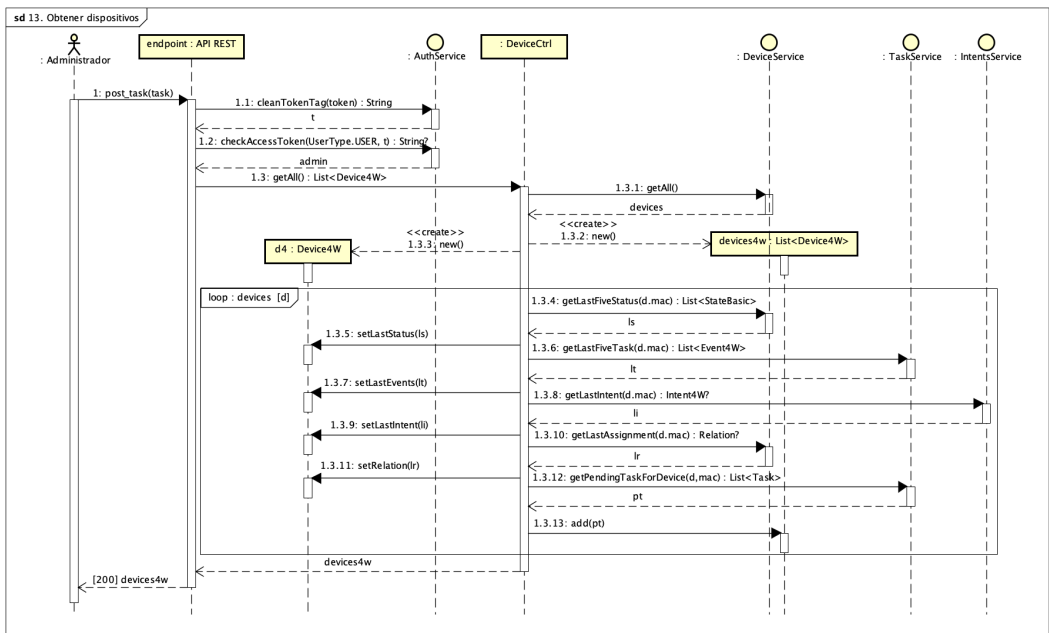


Figura 5.14: Diagrama de secuencia DS13 - Obtención de todos los dispositivos

Obtención de Actividad

Un requisito del sistema era la necesidad de un servicio que obtuviese la actividad relacionada con un dispositivo en un periodo establecido, pero también la implementación de un servicio que obtuviese la actividad de un determinado usuario.

Por motivos de seguridad, se ha implementado la obtención de la actividad de un dispositivo, de la cual se puede obtener la sucedida entre cualquier periodo de fechas siempre y cuando no tenga ningún usuario asociado.

En caso de que un dispositivo tenga un usuario asociado en el momento de la consulta, los resultados se limitarán a mostrar la actividad de ese dispositivo únicamente con el usuario actual.

De este modo, como se puede observar en el diagrama de secuencia incluido a continuación, se evita la posibilidad de que un usuario acceda a consultas de actividades hechas con el mismo dispositivo por otro usuario, al igual que no permite obtener a los administradores la actividad de un determinado usuario, si el usuario ya no tiene el dispositivo asignado, respetando por tanto la privacidad.

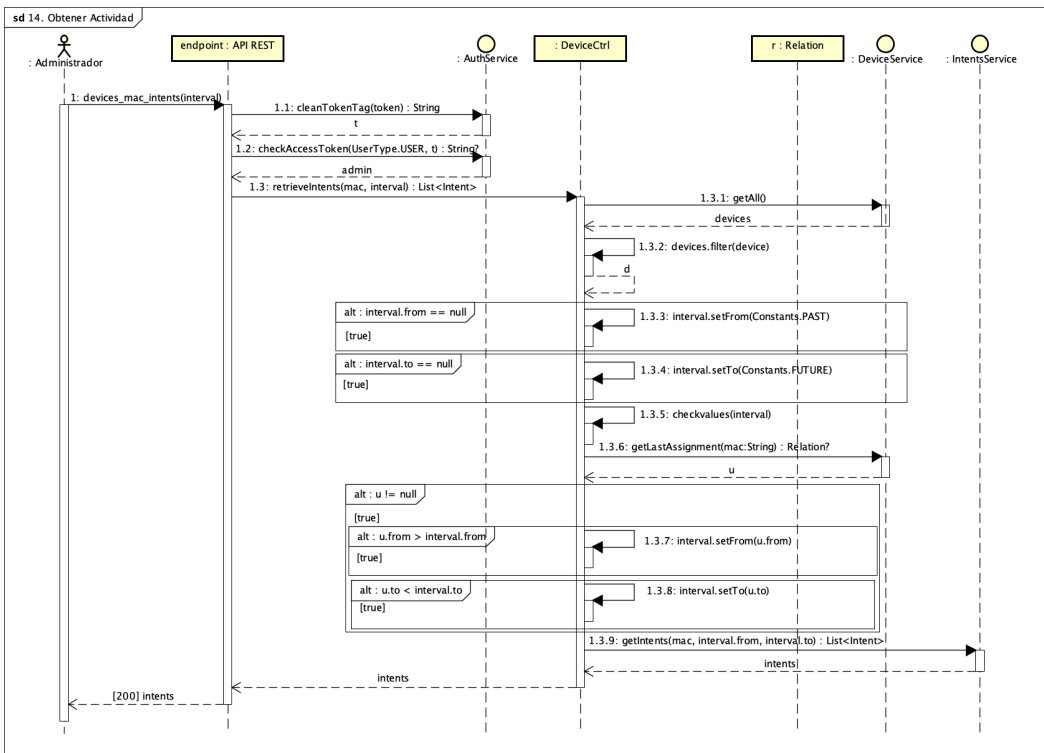


Figura 5.15: Diagrama de secuencia DS14 - Obtención de actividad

Adición de una Nueva Localidad

Con el fin de una mejor organización del sistema, se debe poder añadir la localidad en la cual se esté operando. De este modo, los dispositivos y usuarios pueden estar asociados a una localidad específica.

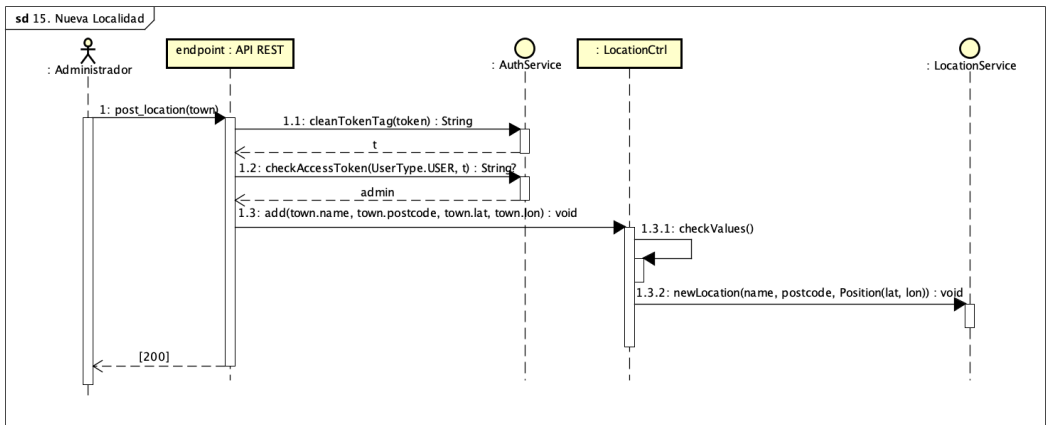


Figura 5.16: Diagrama de secuencia DS15 - Adición de nueva localidad

Adición de un Nuevo Usuario

Será necesario únicamente el nombre, documento nacional de identidad, y código postal.

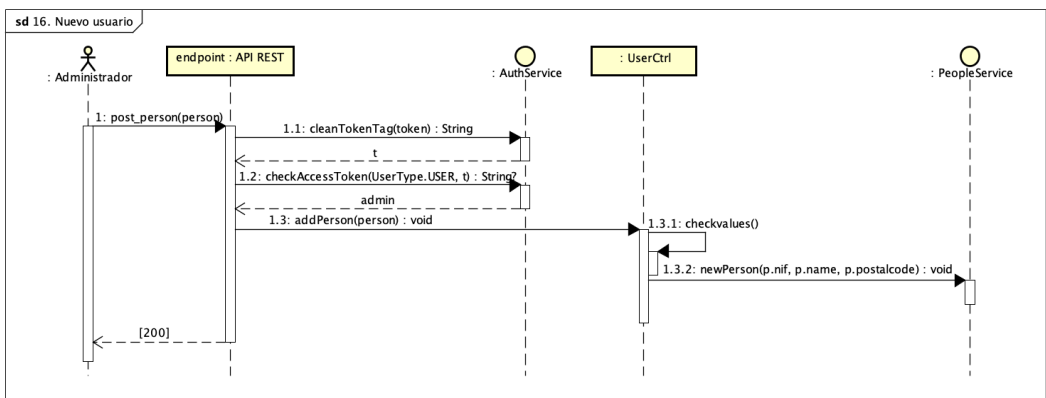


Figura 5.17: Diagrama de secuencia DS16 - Adición de nuevo usuario

Asignación de Dispositivo

Se creará únicamente una relación entre el usuario y el dispositivo, estando el dispositivo asociado a una determinada localidad a través del código postal de su poseedor.

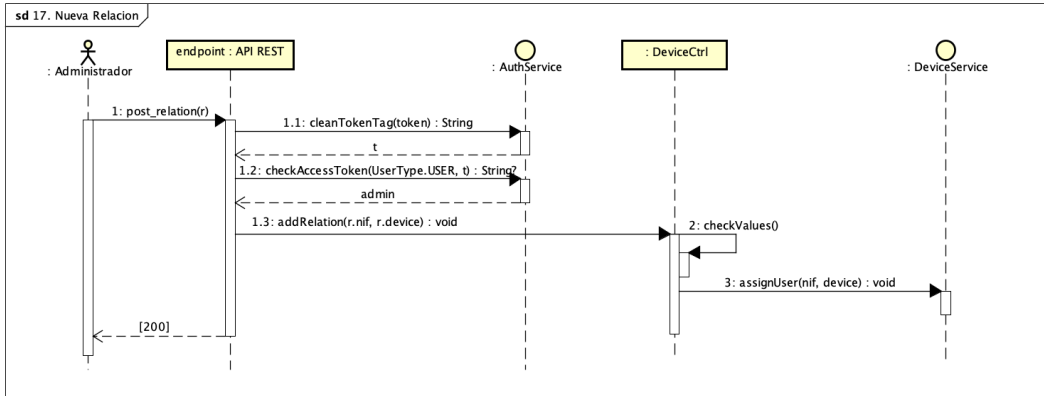


Figura 5.18: Diagrama de secuencia DS17 - Asignación de dispositivo

Desasignación de Dispositivo

Se debe permitir finalizar la relación entre un usuario y un dispositivo, de modo que el dispositivo pueda ser asociado a otro usuario en cuestión.

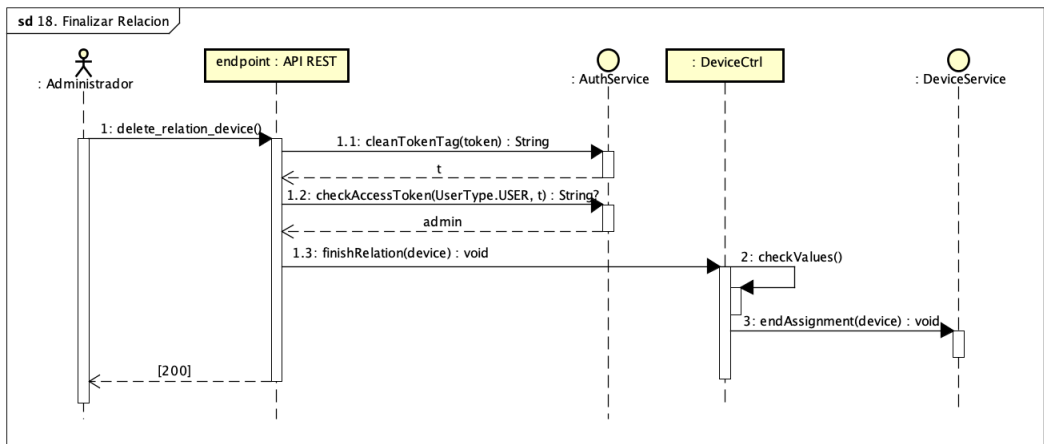


Figura 5.19: Diagrama de secuencia DS18 - Desasignación de dispositivo

Obtención de una Relación de Usuario

Por motivos de necesidad, el administrador puede requerir conocer cual es la relación de un dispositivo específico. Por ello, a continuación se muestra cómo el sistema consigue la información.

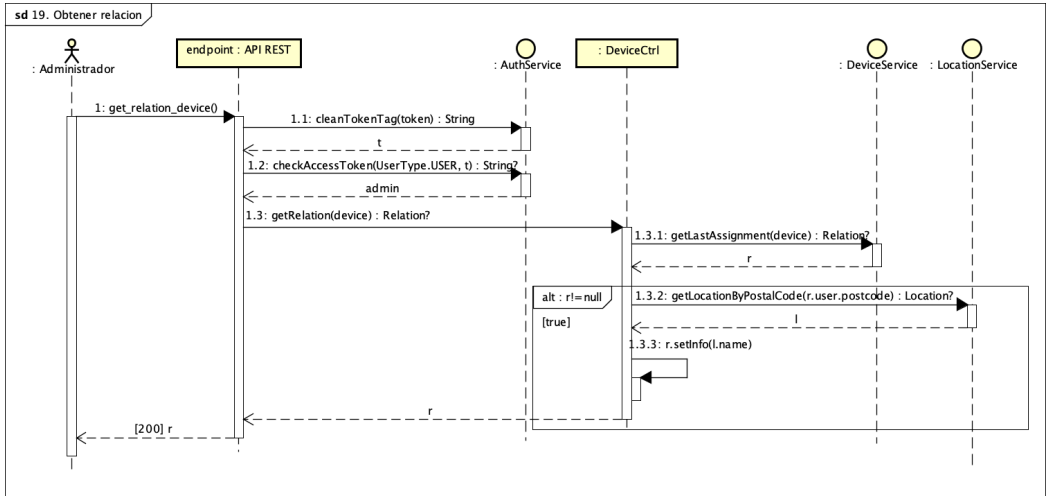


Figura 5.20: Diagrama de secuencia DS19 - Obtención de relación

5.3. Interfaz de Usuario

El diseño de la interfaz de usuario no ha sido creado con anterioridad al desarrollo del proyecto, sino que ha estado siendo diseñado durante la elaboración de cada prototipo.

Para la realización del diseño en cada prototipo se ha seguido un mismo estilo propio de LookFeel basando el proyecto en un diseño mediante tarjetas, al igual que un diseño que permita la adaptación en función de la pantalla, variando al ser mostrado tanto en ordenadores como en tabletas o smartphones.

Para la obtención de un diseño acorde y usable para el administrador se ha seguido el siguiente planteamiento.

1. Prediseño de la interfaz de usuario en papel.
2. Pruebas de usabilidad con usuarios externos al proyecto.
3. Rediseño de la interfaz de usuario.
4. Implementación del diseño.

Todo este proceso está detallado en el capítulo 6.3.3 para favorecer la comprensión del motivo por el cuál se ha ido tomando ciertas decisiones de diseño.

Capítulo 6

Implementación

6.1. Planteamiento

Una buena realización de un proyecto comienza con un buen diseño inicial en el que se establecen las bases para un desarrollo correcto.

Este proyecto puede ser dividido en 3 subproyectos, donde cada subproyecto tendrá una arquitectura diferente debido a las necesidades que requiere su implementación.

El primer subproyecto es el asignado a la implementación de un software que será desplegado en el dispositivo y que permita la conexión con el servidor, aceptando la manipulación del propio dispositivo de manera remota.

El segundo subproyecto corresponderá con el propio sistema alojado en el servidor, al que llamaremos backend, el cual tendrá acceso a una base de datos y servirá una API REST para permitir una comunicación entre los dispositivos y los administradores.

El tercer subproyecto corresponde con el desarrollo de un sitio web, al que se llamará frontend, el cual permite el acceso a los administradores para el control y gestión de los dispositivos, al igual que para consultar las estadísticas.

6.2. Arquitectura

6.2.1. Dispositivo

En cuanto a la arquitectura final del dispositivo, no va a ser especificada ya que el presente proyecto no se adentra en la implementación del asistente, sino que pone las pautas y los protocolos mediante los cuales el asistente podrá comunicarse con el servidor.

Para poder mostrar estas pautas se ha implementado un controlador de pruebas, que será explicado junto a los casos de uso del dispositivo, pudiendo permitir posteriormente el desarrollo del asistente virtual inteligente con cualquiera de los métodos existentes.

6.2.2. BackEnd

Como se ha nombrado anteriormente, el sistema desarrollado en la parte del servidor deberá servir una API REST, de modo que la arquitectura elegida para la implementación del backend es una arquitectura REST, la cual se basa en ofrecer unos end-points desde los cuales se trata la información almacenada en el sistema.

Para una mejor implementación de esta arquitectura, se estructura el sistema bajo la Clean Architecture [28], de modo que se permite el desarrollo del sistema en una organización formada por capas, donde el acceso a la siguiente capa es lineal, evitando dependencias cruzadas que perjudiquen la escalabilidad del sistema.

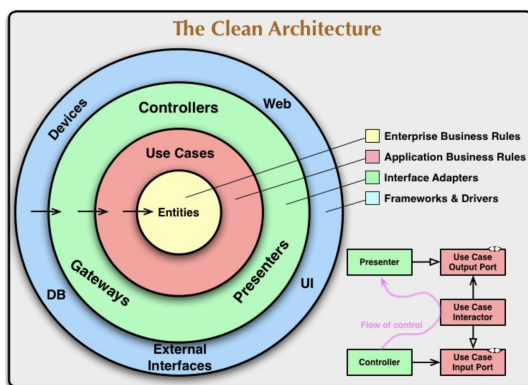


Figura 6.1: The Clean Architecture, Robert C. Martin [28]

6.2.3. FrontEnd

El sitio web a desarrollar para la administración del sistema se construye a partir del framework javascript de VueJS, como ya se ha mencionado anteriormente en el apartado 3.5.8.

VueJS propone una arquitectura MVVM [25], en la cual el diseño del sistema se basa en el desarrollo de múltiples componentes: Un componente es una unidad que dispone de su propio sistema de vista-presentador, teniendo una interfaz gráfica implementada en HTML + CSS, que efectúa un comportamiento a través de funciones JavaScript.

Esta modularidad interna basada en componentes debe seguir unas pautas [26] para sacar el máximo rendimiento de estos, al igual que para poder en un futuro reemplazar o eliminar

los componentes creados por otros que se ajusten a los nuevos requisitos sin perjudicar el resto de ellos.

Los datos que se manejan en la web aparecen en la vista de forma dinámica, de manera que es el presentador quien los puede variar.

6.3. Implementación

6.3.1. Dispositivo

El prototipo, al conectarse a una red eléctrica será el que inicie el asistente, al igual que dejará corriendo en segundo plano el subproyecto que hemos desarrollado con el fin de poder controlar y monitorizar el propio dispositivo.

Naturaleza del controlador

El monitoreo y control remoto de un dispositivo plantea el siguiente dilema: cómo invertir los roles para que un cliente haga de servidor recibiendo información, mientras el servidor hace de cliente mandando a este peticiones.

Para resolver esta adversidad se toma una visión general del proyecto:

1. El dispositivo informa cada cierto tiempo de que sigue encendido.
2. El servidor proporciona una API REST.

Como bien se sabe, las operaciones básicas de una API REST son *GET*, *POST*, *PUT* y *DELETE*, de modo que si se hace una petición *GET* para informar sobre su conexión activa a la red eléctrica, se puede aprovechar por parte del servidor esta llamada para meter un mensaje específico en el cuerpo de la respuesta: este mensaje específico será el que propicie que se realice una acción en el dispositivo.

De este modo el servidor podría manejar el dispositivo, monitorizándolo o pidiendo que haga acciones de manera remota, pudiendo tratarse por ejemplo de una tarea cuya finalidad sea que el asistente inicie una conversación con el usuario final en caso de que haya habido un accidente, de manera que se le pueda ayudar.

Este planteamiento en cuanto a la naturaleza del controlador y su modo de uso parece factible, pero tiene la pega de la usabilidad, ya que si el dispositivo tiene una configuración de mandar su estado cada 24 horas, el control del dispositivo se demoraría demasiado, y aquí entra en juego la siguiente técnica:

Se puede configurar en el propio dispositivo que las 24 horas que pasa entre aviso y aviso se obtengan a partir de un fichero de configuración, que permita la variación de estas 24 horas.

6.3. IMPLEMENTACIÓN

También, en segundo plano, se puede configurar que a ciertas horas, el dispositivo disminuya su periodo de aviso, coincidiendo con unas ciertas horas relativas a la jornada laboral del administrador del sistema, de modo que este pueda mandarle una primera tarea que sea otra reducción del periodo de aviso, por ejemplo a 10 segundos, permitiendo una comunicación síncrona más pareja a una comunicación real, y permitiendo un envío posterior de tareas al dispositivo que se realizarían en un espectro corto de tiempo.

En la figura 6.2 se puede observar el flujo de estados propuesto e implementado en este proyecto con el fin de poder controlar y monitorizar el dispositivo.

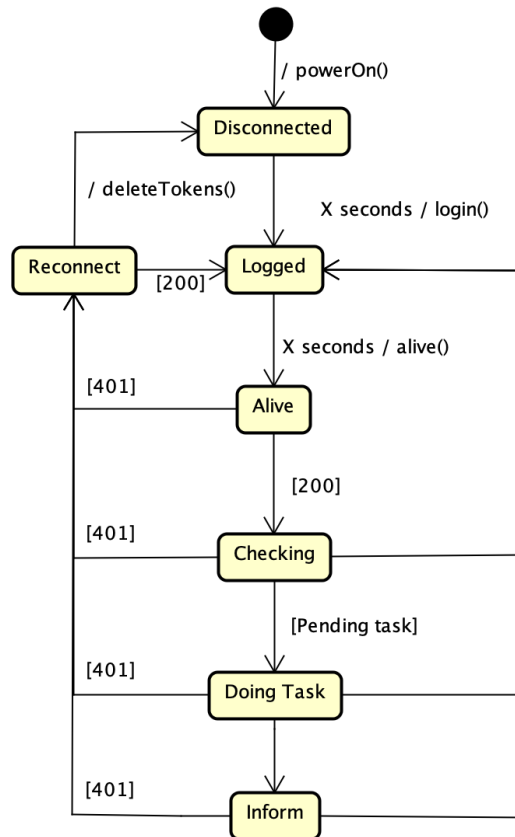


Figura 6.2: Flujo de estados del dispositivo

Desarrollo

Una vez planteada al teoría de cómo debe funcionar el asistente, toca ponerse con la práctica y demostrar que se puede formar un sistema que funcione y sea escalable.

Para ello, se establece un espacio de trabajo dentro del dispositivo, el cual se vincula con un proyecto alojado en un servicio de control de versiones. De esta manera, se permite la

posibilidad de actualizar las acciones del dispositivo con tan solo realizar una acción *pull* del repositorio de origen.

Sistema de arranque

El sistema implementado se ejecutará en el momento en el que se ejecute la clase *Main.py*, que intenta iniciar sesión con el servidor a través del método *signin()* de un objeto obtenido a través de la clase *Login.py*. En el momento de instanciación del objeto *login*, su constructor obtiene la dirección MAC del dispositivo, almacenándola en:

./cache/macfile.txt

Esta dirección MAC propia y única para cada dispositivo, será utilizada como identificador en los credenciales a la hora de identificarse contra el servidor. La contraseña, será una encriptación de la dirección MAC que se obtiene simultáneamente en el constructor.

Si se consigue iniciar sesión, se almacenan los tokens tanto en el objeto instanciado, como en el siguiente fichero externo:

./cache/tokens.json

Este almacenamiento externo permite su utilización a futuros programas que requieran los tokens para realizar peticiones contra el servidor. Continuando con el método el cual inició sesión, este devuelve un booleano, *true*, desbloqueando un semáforo del *main* que permite el envío al servidor del estado actual del dispositivo, a través de un método perteneciente a un objeto de tipo *Alive.py*, el cual devuelve también otro booleano, que será *false* en caso de que no se lleve la acción correctamente a cabo, cerrando el semáforo.

Tras la implementación de este flujo de estados se ha podido comprobar la perfecta sincronía con el servidor, permitiendo proceder al diseño de un sistema que acepte la implementación de tareas.

Sistema de tareas

En el momento en el cual se hace la petición de */alive*, existen tres posibles respuestas por parte del servidor:

1. *HTTP Status Code 200*: indica que la petición ha sido correcta.
2. *HTTP Status Code 300*: indica que la petición ha sido correcta, pero que hay múltiples respuestas, indicando al sistema del dispositivo que debe extraer la información que contiene el mensaje. En este mensaje, se incluye una lista de tareas pendientes del dispositivo:

```
[{
  "id" : "task identifier",
  "device" : "device identifier",
  "event" : "TAREA",
  "by" : "who ordered this",
  "at" : "when it was ordered"
}]
```

3. *HTTP Status Code 401*: indica que el token ha dejado de ser válido, por lo que se debería volver a iniciar sesión, o refrescar el token. Como un token de acceso tiene establecido por razones de seguridad un tiempo de validez de 30 minutos, y el dispositivo en principio realizará la petición de mostrar que sigue activo cada 24 horas, no se ve necesaria la implementación en el dispositivo de un método que refresque el token, ya que produciría un aumento del consumo de datos, al refrescar constantemente un token al que no va dar uso.

Una vez recibida una respuesta de tipo *HTTP Status Code 300*, se procede a analizar las tareas pendientes que están en el cuerpo del mensaje. Para ello, se procede a ejecutar siempre la primera de la lista gracias a que han sido devueltas por orden de asignación desde el servidor. Analizando los campos de la tarea, vemos que dispone de campo llamado **event**, el cual indica el nombre de la tarea a realizar.

El nombre de la tarea es único, y debe ser una palabra sin espacios, permitiendo utilizarlo como palabra clave. Gracias a esta palabra clave, se puede pedir al dispositivo que ejecute la tarea concreta, y para ello administramos el sistema de ficheros de modo que siga la siguiente estructura:

```
./task/TAREA/
```

y dentro de ese directorio, debe haber un script, el cual ejecute la tarea específica:

```
./task/TAREA/init.sh
```

Este script, puede llamar a otros programas o a otros scripts, de modo que se permite la realización de cualquier evento configurado.

Una vez se tiene esta estructura implementada, como se puede observar en la figura 6.3, tan solo hay configurar que al recibir como respuesta del servidor un código 300, se ejecute el script que está almacenado en la dirección obtenida al sustituir el nombre de la tarea, por ejemplo, **REBOOT**, en la ruta base:

```
sh ./task/REBOOT/init.sh
```

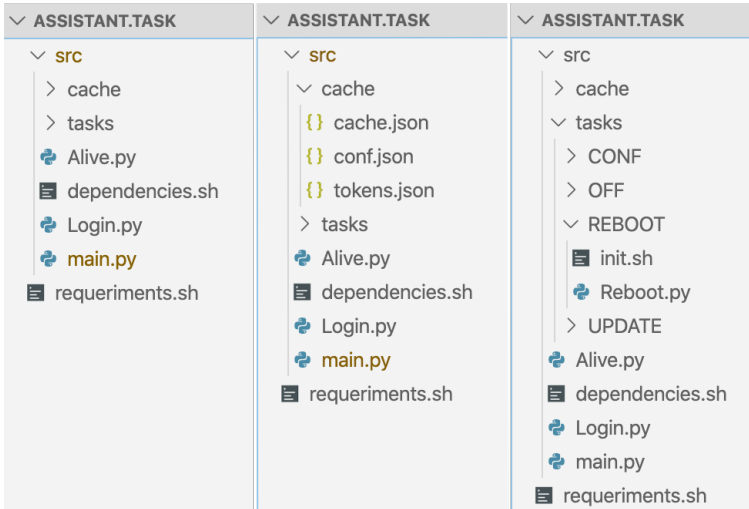


Figura 6.3: Contenido global del repositorio

Como se puede observar, este método de organización de tareas posibilita la fácil implementación de nuevas tareas sin poner en peligro las ya creadas, ya que únicamente se necesita crear un directorio nuevo por cada nueva tarea, y un script que inicie las acciones a realizar.

Inicio automático

Ya se tiene una configuración válida para el dispositivo, pero hay que poder asegurar su buen despliegue e inicio al reiniciar el dispositivo, de modo que se pueda tener como un valor seguro que, si un usuario reinicia la máquina físicamente, este dispositivo será capaz de iniciar el sistema ahora creado para poder monitorizarlo y controlarlo.

Entonces, para la configuración del espacio de trabajo se proporciona un script que permite la configuración del dispositivo en caso de ser la primera vez que se utiliza ese dispositivo, por si no se han cargado bien los servicios.

En dicho script, almacenado como *requirements.sh*, se establecen dos pasos claves del proyecto:

1. La configuración para el inicio automático del sistema de control remoto del dispositivo a través de la herramienta de *crontab* [27] que proporcionan los sistemas Unix.

```
# configure scheduled task
(crontab -l ; echo "@reboot sleep 60; cd /home/pi/assistant.task/src/ ;
python /home/pi/assistant.task/src/assistant-alive.py & >
home/pi/assistant.task/src/cache/logfile.txt") | crontab -
```

De este modo, se abre el fichero de configuración de *crontab* en el cual se plasma qué es lo que se desea que se ejecute en cada reinicio, como indica la etiqueta *@reboot*. Si

se presta atención al script, se solicita una espera de 60 segundos a partir del reinicio del dispositivo, y el motivo por el cual se establece ese tiempo es para permitir que el dispositivo pueda configurar su conexión a internet y levantar previamente todos sus servicios antes de que el sistema aquí descrito arranque, evitando posibles interferencias que no permitiese arrancar de la manera correcta, como puede ser una mala obtención de la dirección MAC de la tarjeta de red.

2. El script de los requisitos ejecuta otro script, el de la instalación de las dependencias, el cual permite la instalación de las librerías de python requeridas para el correcto funcionamiento del dispositivo.

```
# install python dependencies
sudo sh ./src/dependencies.sh
```

De esta manera, se establecen las medidas y acciones necesarias, al igual que los scripts requeridos para poder asegurar el cumplimiento y buen funcionamiento del dispositivo aplicando los protocolos de comunicación descritos.

6.3.2. BackEnd

Introducción

Para el desarrollo del BackEnd se va a seguir la arquitectura descrita en el apartado 6.2.2, de modo que a continuación se describirá cómo esa arquitectura ha sido implementada, con qué herramientas, y cuáles son las dependencias o servicios que posee.

Principios de diseño

Para un buen desarrollo de un sistema software se debe seguir los patrones de diseño, al igual que se deben aplicar los diferentes principios de diseño.

A la hora del desarrollo, se ha considerado respetar los principios de diseño **SOLID** [28]:

1. Open-Closed Principle:

Partimos de un lenguaje como Kotlin el cual toda clase de datos es cerrada en cuanto a extensión. De este modo, se ha asegurado que toda clase que se requiera como abierta lleva el operador *open* con ella.

2. Single responsibility Principle:

Se ha respetado que cada clase tenga únicamente una única función, dividiendo en varias toda clase que requiera más de una responsabilidad. Esto se ha podido asegurar a la hora de la implementación y segregación del sistema en servicios a la hora de acceder a la base de datos, al igual que en los múltiples controladores que posee el sistema con tal de acceder y realizar funciones diferentes.

3. Liskov Substitution Principle:

El principio de sustitución de Liskov nos dice que debemos preservar el funcionamiento de una interfaz o clase en el momento que la heredamos, de modo que una redefinición de un método debe poder cumplir los contratos. Este principio se ha cumplido a la hora de implementar los servicios, ya que se ha requerido que cumplan todos y cada uno de los requisitos impuestos en las interfaces de la capa de la lógica del sistema.

4. Dependency Inversor Principle:

El sistema ha sido implementado siguiendo el esquema de The Clean Architecture, como ya se ha nombrado anteriormente, de modo que los endpoint de la API Rest son quienes solicitan a los controladores la realización de cierto caso de uso. Los controladores, para la realización de estos casos de uso pueden requerir el acceso a servicios o repositorios externos. Para ello, cada controlador indica cuales son los servicios que va a necesitar, y si un servicio quiere servir al caso de uso, debe cumplir con los contratos establecidos en la interfaz pública del caso de uso. De esta manera, siguiendo todos los principios anteriores, se cumple este principio ya que si el servicio integra la interfaz, el servicio puede ser inyectado en el caso de uso, permitiendo una estanquidad de los controladores a los servicios externos, y con ello facilitando el cambio de un servicio por otro sin que afecte al sistema, y cumpliendo así el principio de inversión de dependencia. Para la inyección de estas dependencias ha sido utilizado el framework de Koin, el cual ha sido explicado anteriormente en el apartado 3.5.3.

5. Interface Segregation Principle:

El principio de segregación de interfaces nos indica que una buena práctica se basa en contar con varias interfaces específicas de modo que se pueda utilizar una u otra en función del tipo de cliente. Esto se ha llevado a cabo a la hora de comprobación de credenciales, ya que se proporciona una interfaz u otra en función de si el token recibido viene por parte de un dispositivo, o de un administrador, pudiendo ambos iniciar sesión, pero ejecutándolo diferentes controladores.

En cuanto a patrones de diseño, se han seguido en mayor medida los descritos a continuación gracias a haber aplicado los principios SOLID descritos anteriormente:

La elaboración de un sistema por capas con una comunicación de dependencias lineal nos permite la utilización del **patrón fachada**, ya que para la conexión entre servicios y controladores hay una interfaz pública la cual limita el número y tipo de objetos que se mandan entre ambos, evitando una sobrecomunicación, al igual que una relación de dependencias circulares.

Gracias a la aplicación de este patrón, se da cabida a la posibilidad de aplicación del **patrón adaptador**, permitiendo la creación de unas nuevas interfaces intermedias que adapten las interfaces públicas de los controladores a posibles futuros servicios que sustituyan a los actuales, sin necesidad de modificar el comportamiento de los controladores los cuales ya realizan los casos de uso, y asegurando entonces un aislamiento escalable.

También, los servicios han servido de punto de acceso a la información de la base de datos gracias al uso del framework *ExposedSQL*, de modo que ha intercambiado la información con

los controladores a través de objetos, separando por tanto el acceso a la BBDD de la capa de negocio, aplicando por tanto el **patrón DAO** y posibilitando cambiar en un futuro el sistema de almacenaje de información sin que requiera un solo cambio en la capa de negocio del sistema.

A la hora de mantener un registro de cómo está funcionando el sistema, se ha utilizado el **patrón Singleton** con el fin de obtener una instancia del objeto *Logger*, y con ello poder retransmitir logs desde cualquier punto del sistema sin requerir aumentar los recursos y evitando errores de interferencias por re-declaración. También, se ha aplicado este patrón para el acceso al *TokenStore*, el cual almacena los tokens activos, y de esta manera permitir la comunicación entre el controlador de sesión con el servidor OAuth a través de la misma instancia, ya que al instanciar un nuevo objeto se perderían los tokens activos ya creados.

Servicios

Los servicios forman el acceso a la capa más externa según el esquema de *The Clean Architecture*, y han sido implementados aceptando y cumpliendo las condiciones impuestas en las interfaces públicas de la capa de negocio.

De esta manera, los servicios que requiere el sistema son los siguientes:

1. *ConfService*: A través de este servicio se puede obtener cualquier configuración existente, ya sea de los dispositivos, de las localidades, o globales. También tiene la opción de marcar ciertas configuraciones como ya instaladas, crear nuevas, o borrarlas en función del identificador o de su estado.
2. *DeviceService*: Este servicio proporciona la información relativa a un dispositivo, siendo posible comprobar si un dispositivo existe en el sistema, comprobar las credenciales, crear uno nuevo, obtener todos, obtener el dispositivo asociado a un determinado usuario, asignar un dispositivo a un usuario concreto, o almacenar y recolectar los últimos estados del dispositivo.

En este servicio se puede apreciar la utilidad de segregar el sistema en servicios, al igual que la inversión de dependencias, ya que se podría sustituir fácilmente el servicio cambiando por tanto la manera en la que se registran los dispositivos, o cambiando la manera en la que se comprueban las credenciales, que en el caso de este sistema es mediante la encriptación de la dirección MAC, por otro tipo de comprobación sin alterar el funcionamiento de la capa de negocio, que seguiría aislada.

3. *IntentsService*: Permite almacenar las actividades realizadas en los dispositivos, es decir, la información relevante obtenida tras una comunicación *máquina-usuario*. También permite obtenerlos, ya sea el último de una máquina, o una lista de los realizados por un dispositivo específico entre dos fechas.
4. *LocationService*: Ofrece la posibilidad de almacenar nueva localidades y luego recopilarlas ya sean todas, por código postal, o por provincias.

5. *PeopleService* Relativo a los clientes que tendrán en su poder un dispositivo. Es útil para añadir personas y para obtenerlas después ya sean todas, o por su nif, o por código postal, al igual que poder obtener los datos sobre el dispositivo asignado para una persona en concreto.
6. *TaskService*: Controla la información almacenada sobre los eventos y tareas asignadas a dispositivos, de manera que permite la adición y obtención de eventos, al igual que de dispositivos. También permite la obtención de tareas pendientes de un dispositivo concreto, o la recolección de todas las tareas de un dispositivo concreto ocurridas en un rango de tiempo concreto.
7. *UserService*: Este servicio lleva la gestión de usuarios del sistema, es decir, de administradores del sistema, que son diferentes que los clientes de los dispositivos. Estos usuarios poseen una contraseña y el servicio ofrece la manera de verificar que sus credenciales son correctos.

Más a fondo

Debido a la complejidad del sistema, se ilustrará el diseño del sistema con varias figuras y sus explicaciones correspondientes con el fin de que el lector pueda comprender la implementación y replicarla en un futuro, al igual que pueda modificar lo que requiera, si en un futuro quiere tomar este documento como manual para una reimplementación del sistema.

El backend estará estructurado en 3 capas:

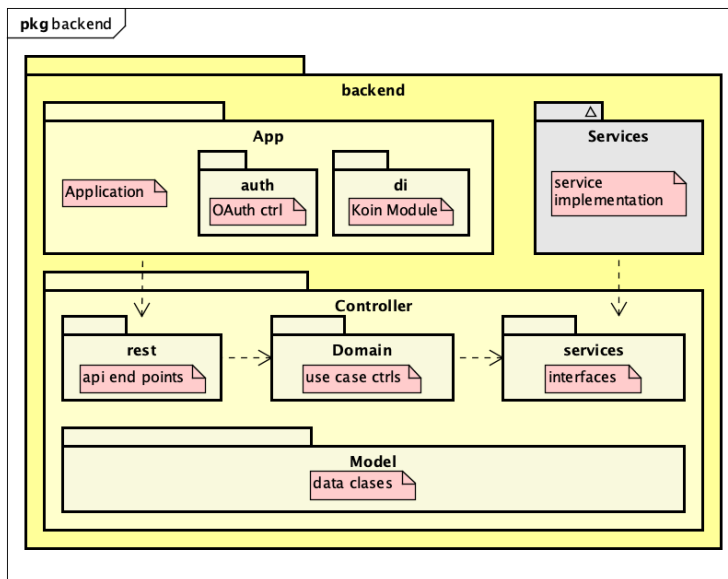


Figura 6.4: Diagrama de diseño

1. **App:** Albergará la aplicación que iniciará el sistema, al igual que tendrá dos subdirectorios. El directorio *auth* manejará la configuración sobre los tokens de acceso, y el directorio *di* será el que contenga la configuración del inyector de dependencias, el cual se configura mediante la implementación de un módulo.
2. **Controller:** Se asocia a la *capa de negocio* del sistema, de modo que se puede considerar el componente aislado principal del sistema, ya que no depende de ninguna clase externa y proporciona modos mediante los cuales se puede acceder a su uso. Internamente, se puede observar el directorio *rest*, el cual contiene los end-points del sistema, los cuales se explicarán más adelante. Otro de los directorios es el subdirectorio de *Domain*, el cual contiene los controladores mediante los cuales se realizan los casos de uso. Estos controladores tienen la lógica del sistema, y para acceder a recursos externos como puede ser la base de datos u otros servicios externos, utilizan servicios externos inyectados a través del framework, siempre y cuando esos servicios externos utilicen las interfaces que proporcionan los controladores. El tercer directorio es el de *services*, el cual contiene las interfaces requeridas por los controladores y consiguiendo con esto una inversión de dependencias, ya que de requerir un servicio por parte del controlador, se ha pasado a que el servicio requiera, para poder ser utilizado, la implementación de estas interfaces.

Se puede observar un cuarto directorio, *Model*, el cual contiene clases que serán utilizadas para el envío de datos entre las diferentes capas, evitando la utilización en esta capa de negocio de clases de datos externas que produzcan una dependencia.

3. **Services:** Son los servicios externos que proporcionan información al sistema. Estos servicios deben implementar las interfaces de servicios que proporciona la capa de negocio. De este modo, si las implementan, pueden ser inyectados a través del inyector de dependencias.

La aplicación que arranca el sistema y maneja toda la configuración es `./src/app/Application.kt`.

En cuanto a los endpoints del sistema, se pueden declarar en la propia aplicación, pero se considera buena-práctica el disgregarlos en diferentes módulos, agrupados por rutas o servicios similares, obteniendo así una mejor visibilidad y permitiendo un acceso intuitivo a la hora del desarrollo.

A cada módulo de ruta se le otorga el acceso a los servicios necesarios que va a requerir los controladores. De este modo, al declarar los end-points se le proporcionan los servicios en su constructor, y se evita una dependencia directa de la capa de negocio con el framework elegido para inyectar esas dependencias, ya que las dependencias se inyectan en el módulo iniciador de la aplicación, el cual es externo a la capa de negocio, siguiendo manteniéndola aislada del exterior.

En el siguiente diagrama se puede observar los 3 subpaquetes ya descritos anteriormente que conforman el sistema, al igual que las dependencias entre ellos: tanto el paquete *App* como el paquete *Services* dependen del paquete *Controller*, el cual no depende de ningún otro y por tanto, está aislado de los cambios del exterior.

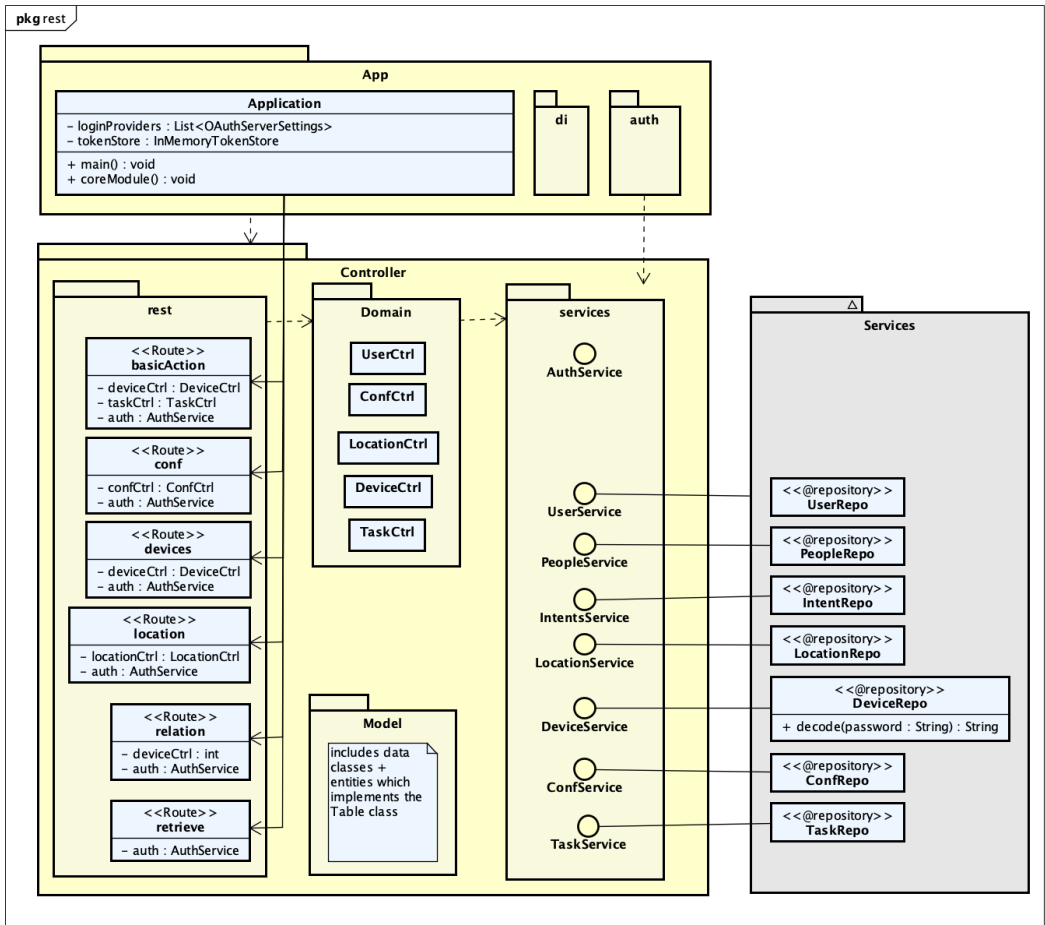


Figura 6.5: Diagrama de Clases: General

En la figura anterior se ha podido ver la estructura y dependencias de cada subpaquete de manera general. Ahora, se particularizará mostrando la estructura interna de cada uno de ellos, explicando qué es y para qué se utiliza cada parte.

Teniendo en cuenta la figura 6.6, podemos observar diferentes partes:

Por un lado, tanto *InMemoryIdentityCustom* como *InMemoryTokenStore*, que son modificaciones del framework OAuth que permiten la utilización del store de tokens fuera del framework, de modo que los tokens se puedan gestionar a través del sistema. Su instalación ya ha sido explicada en el apartado 3.5.4, de modo que no se va a entrar ahora de nuevo.

Una vez ya obtenido un acceso público de *TokenStore*, este será gestionado únicamente desde el servicio creado para el control de tokens, *TokenCtrl*, el cual implementa la interfaz *AuthService*, permitiendo que este servicio sea inyectado en el sistema.

6.3. IMPLEMENTACIÓN

Esta inyección se produce tras la implementación en *Application* del módulo de dependencias creado con Koin, *app.di.Module.myModule*, el cual también ha sido explicado cómo debe ser instalado en la aplicación en el apartado 3.5.3, al igual que el resto de servicios.

La aplicación, por tanto, obtendrá los servicios inyectados y se los proporcionará a cada ruta, como se pueden observar en la figura 6.6, de modo que la capa de negocio ya dispondrá de los servicios para su uso, a través de las interfaces que había predefinido.

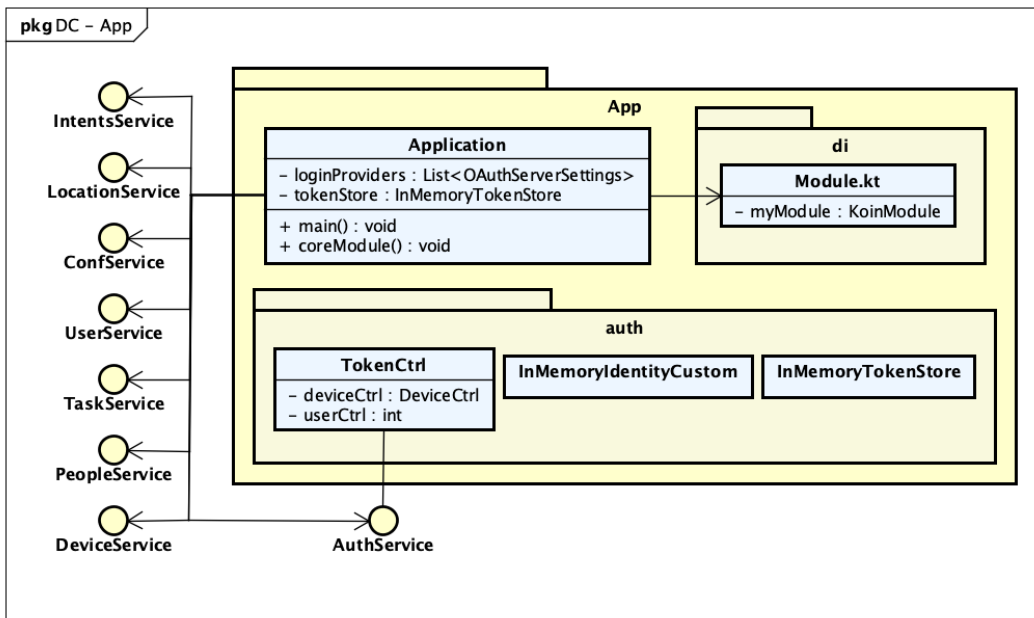


Figura 6.6: Diagrama de Clases: App

En cuanto a la capa de negocio expuesta en la 6.5 como *Controller*, se puede observar que los paquetes *Domain*, *services*, y *Model* no han sido explicados, por lo que se mostrará en las siguientes figuras su contenido.

Se puede observar en la figura 6.7 que *Domain* contiene el dominio de la aplicación, gestionando los controladores que hacen que se cumplan todos los casos de uso descritos anteriormente, almacenando toda la lógica del sistema.

Cada controlador tiene atribuidas unas las dependencias, que están relacionadas con las interfaces descritas por esta capa de negocio, asegurando por tanto que no se va a tener que implementar ninguna operación externa.

Las interfaces *-descritas en la figura 6.8-* muestran las operaciones requeridas por los controladores, sirviendo de contrato para los servicios externos que quieran dar un uso futuro al sistema.

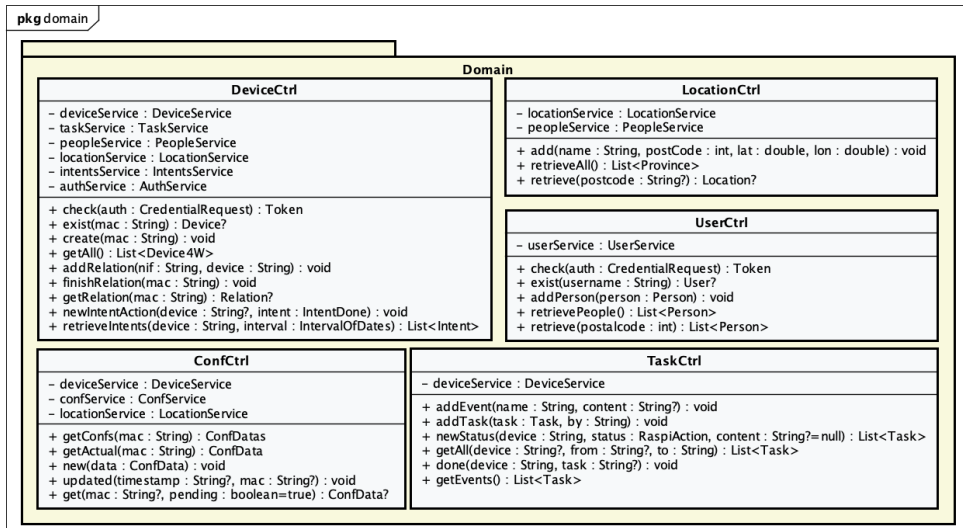


Figura 6.7: Diagrama de Clases: Domain

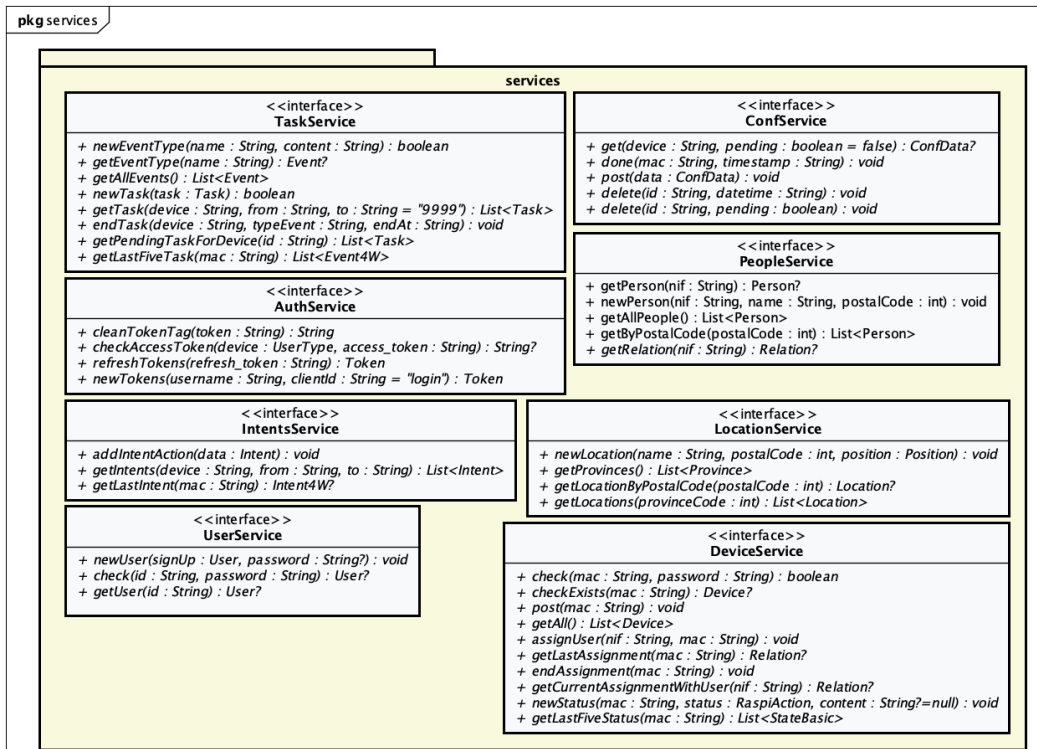


Figura 6.8: Diagrama de Clases: Services

6.3. IMPLEMENTACIÓN

Para la comunicación entre la capa de negocio con las otras capas, al igual que para la gestión de la información en comunicaciones internas se necesitan unas clases que sirvan para el envío de datos, favoreciendo y simplificando la comunicación.

Estas clases, llamadas *clases de datos*, son las mostradas a continuación en la figura 6.9.

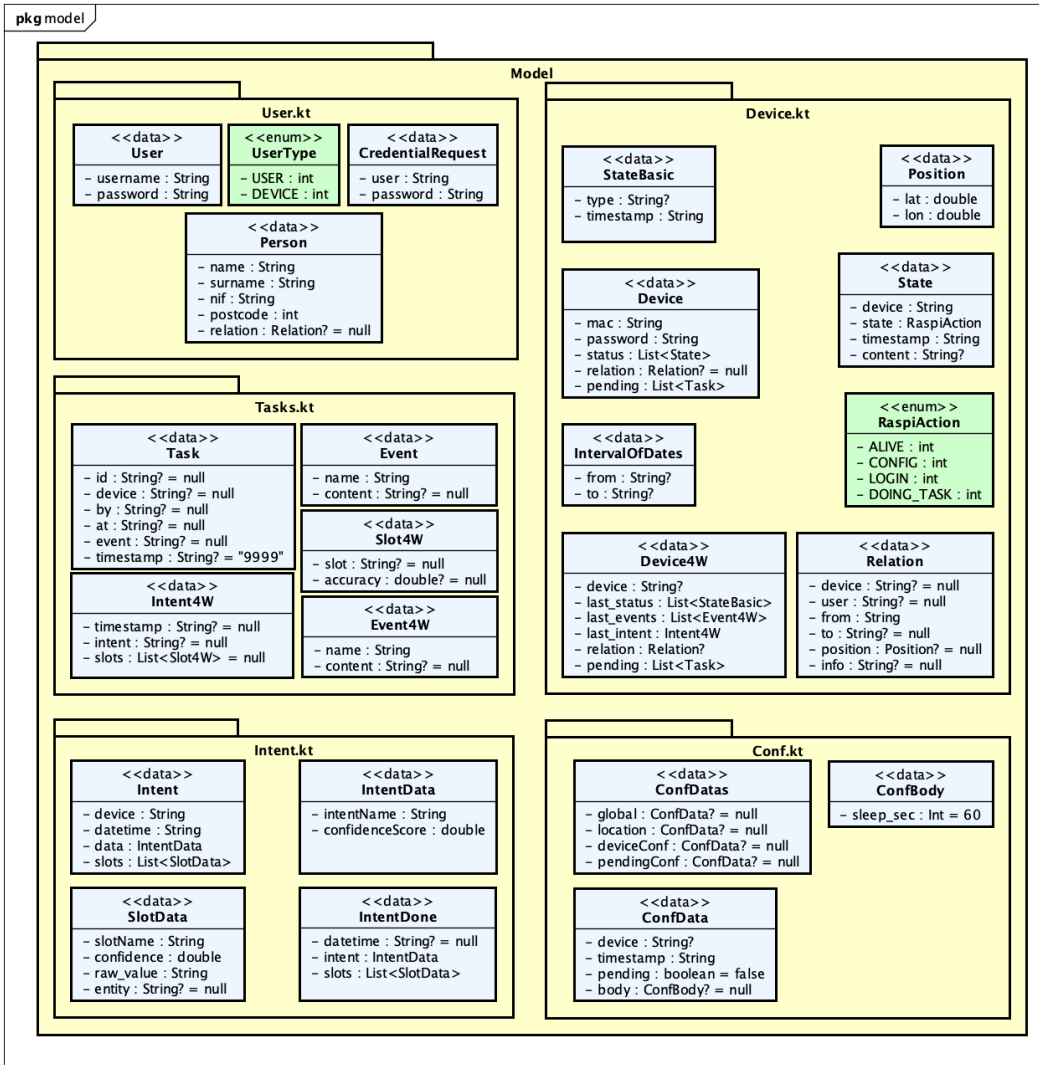


Figura 6.9: Diagrama de Clases: Model

Diagrama Entidad-Relacion

En el siguiente diagrama, representado con la figura 6.10, se puede observar cuales son las relaciones entre las distintas tablas que componen la base de datos, de modo que se aprecian las dependencias de claves que comparten entre si.

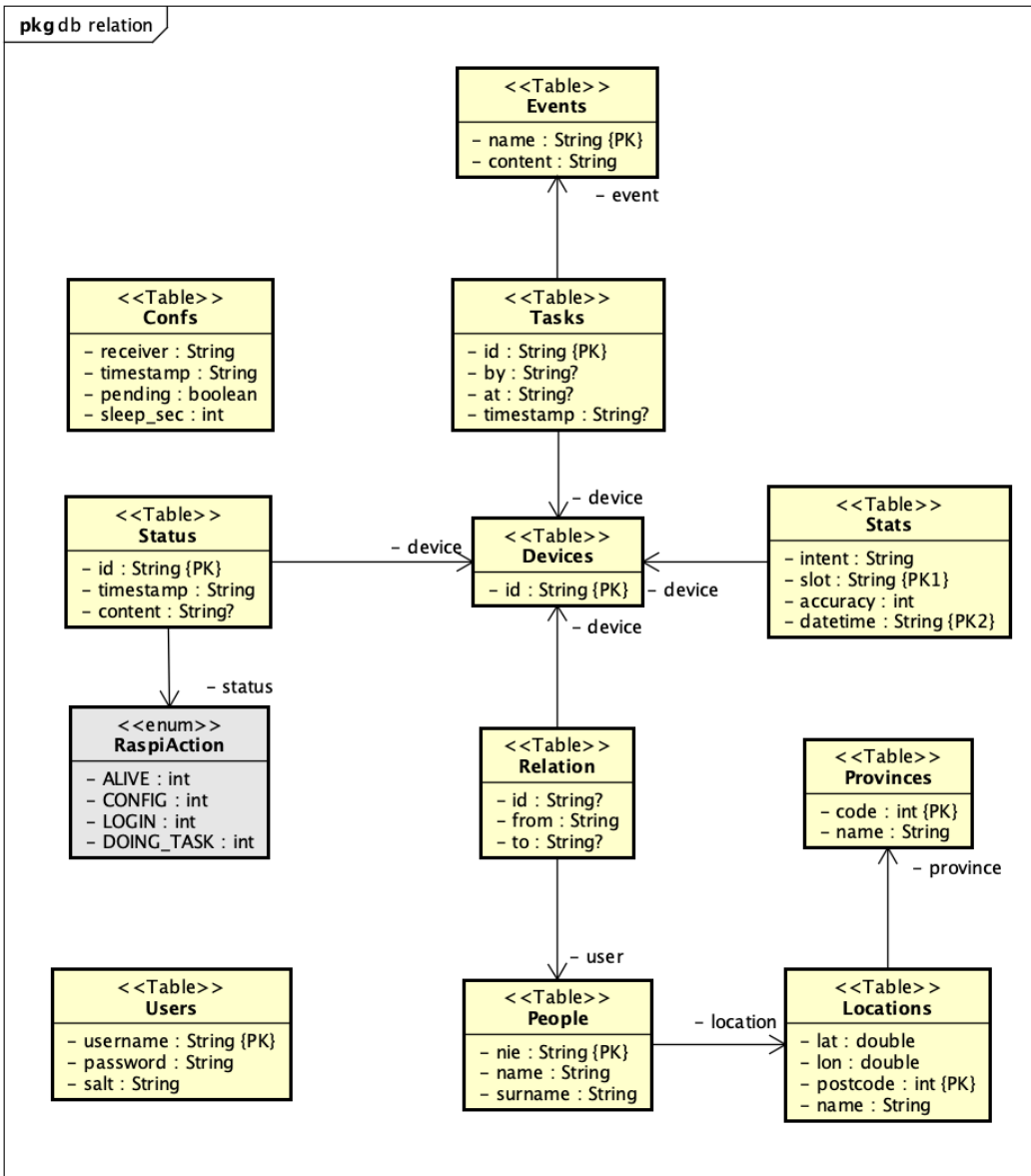


Figura 6.10: Diagrama Entidad-Relación

End-Points

Los end-points del sistema se van a diferenciar en dos tipos: los habilitados para el dispositivo, que comienzan por */device*, y los habilitados para el administrador, que comienzan por */worker*.

De esta manera, el sistema validará el token recibido en la petición, obteniendo qué tipo de usuario es, y sólo dejará acceder al recurso si el usuario es del tipo correcto para esa llamada.

Todas las llamadas, excepto las de inicio de sesión y de refresco de credenciales, deberán contener el token de acceso como parámetro del header.

En cuanto a las posibles respuestas para las llamadas, hay 3 respuestas preestablecidas:

1. HttpStatusCode 401 : Unauthorized.
Se da cuando los credenciales introducidos en el inicio de sesión no son correctos, o cuando el token de acceso no sea válido, ya sea por haberse expirado, por ser falso o por ser de un tipo de usuario no permitido para esa llamada.
2. Http Status Code 400 : Bad Request.
Se va a obtener cuando los datos introducidos en la llamada no son válidos, ya falte algún parámetro en el formulario, o no cumplan con las especificaciones.
3. Http Status Code 500 : Internal Server Error.
Será devuelto cuando la llamada haya sido completamente correcta, pero el procesamiento de ella haya dado con algún tipo de error en el lado del servidor. En este caso, se deberá contactar con el administrador del BackEnd.

[POST] /device/login *Inicio de sesión con un dispositivo.*

Parámetros de la llamada:

```
1      "user" : "mac"
2      "password" : "encrypted mac"
```

RESPONSE:

Http Status Code 200 : Ok

```
1      {
2          "access_token" : "Bearer lo-que-sea-de-acceso",
3          "refresh_token" : "Bearer lo-que-sea-de-refresco"
4      }
```

[GET] */device/alive* Envío de ping de dispositivo.

RESPONSE:

Http Status Code 200 : Ok

```
1 {"status": 200,"action": "ALIVE"}
```

Http Status Code 300 : Multiple Choices

```
1 JSONArray<Task>
```

[GET] */device/tasks/:id/doing* Aviso de ir a realizar una tarea, especificando su identificador.

RESPONSE:

Http Status Code 200 : Ok

```
1 {"status": 200,"action": "ALIVE"}
```

[GET] */device/confjs* Devuelve la configuración más actual del dispositivo.

RESPONSE:

Http Status Code 200 : Ok

```
1 JSON<ConfData>
```

[GET] */device/confjs/:timestamp* Aviso del dispositivo de haber instalado la configuración que tiene un timestamp concreto.

RESPONSE:

Http Status Code 200 : Ok

```
1 Configuration has been updated on device.
```

[POST] */device/intents* Aviso del dispositivo de haber realizado una conversación con el usuario, enviando los datos básicos.

BODY:

```
1 JSON<IntentDone>
```

RESPONSE:

Http Status Code 200 : Ok

6.3. IMPLEMENTACIÓN

Entre todas las llamadas, hay dos que no corresponden ni están limitadas al tipo de dispositivo, permitiendo que todo token activo sea valido para acceder al recurso, y son las siguientes:

[GET] /auth/refresh_token *End-point para actualizar los tokens.*

BODY:

```
1  {
2      "refresh_token" : "Bearer lo-que-sea-de-refresco"
3  }
```

RESPONSE:

Http Status Code 200 : Ok

```
1  {
2      "access_token" : "Bearer nuevo-de-acceso",
3      "refresh_token" : "Bearer nuevo-de-refresco"
4  }
```

[GET] /towns/SANVICENTEDELPALACIO *Recibe información acerca de una localidad en concreto. De momento solo está configurado para obtener la información relativa a la localidad de San Vicente del Palacio.*

RESPONSE:

Http Status Code 200 : Ok

```
1  {
2      "status" : 200,
3      "action" : "Alive",
4      "data" : {
5          "data" : JSONArray<Place>
6      }
7  }
```

[POST] */worker/events* Crea un nuevo tipo de evento asignable a dispositivos.

BODY:

```
1  {
2    "name" : "REBOOT",
3    "content" : "Campo disponible"
4  }
```

RESPONSE:

Http Status Code 200 : Ok

```
1  Added
```

[GET] */worker/events* Obtiene todos los tipos de eventos asignables a dispositivos.

RESPONSE:

Http Status Code 200 : Ok

```
1  JSONNarray<Event>
```

[POST] */worker/task* Asigna una tarea a un dispositivo concreto.

BODY:

```
1  {
2    "device" : "Direccion MAC" || "codigo postal" || "GLOBAL",
3    "event" : "REBOOT"
4  }
```

RESPONSE:

Http Status Code 200 : Ok

```
1  Added.
```

6.3. IMPLEMENTACIÓN

[POST] /worker/tasks Recibe las tareas asignadas a un dispositivo concreto en un periodo de tiempo.

BODY:

```
1  {
2    "device" : "Direccion MAC" || "codigo postal" || "GLOBAL",
3    "from" : "YYYY-MM-DDTHH:MM:SSZ" || null,
4    "to" : "YYYY-MM-DDTHH:MM:SSZ" || null
5  }
```

RESPONSE:

Http Status Code 200 : Ok

```
1  JSONNarray<Event>
```

[POST] /worker/confjs Recibe las tareas asignadas a un dispositivo concreto en un periodo de tiempo.

BODY:

```
1  {  "device" : "Direccion MAC" || "codigo postal" || "GLOBAL",
2    "body" : JSON<ConfBody> }
```

RESPONSE:

Http Status Code 200 : Ok

```
1  Added new configuration
```

[GET] /worker/confjs/:id Recibe todas las configuraciones posibles. Si el identificador es de un dispositivo, trata de recuperar la asignada al dispositivo, la pendiente de ser instalada, la de su localidad, y la global. Si el identificador es el código postal de una localidad, obtiene la de la localidad y la global, y si el identificador indica "GLOBAL", pues solo recupera la global.

RESPONSE:

Http Status Code 200 : Ok

```
1  {
2    "global" : ConfData || null,
3    "location" : ConfData || null,
4    "deviceConf" : ConfData || null,
5    "pendingConf" : ConfData || null,
6  }
```

[GET] /worker/devices *Obtiene la lista con todos los dispositivos con su información de últimos estados, actividades, y demás.*

RESPONSE:

Http Status Code 200 : Ok

```
1  [{
2    "device" : "mac",
3    "last_status" : [{
4      "type" : "REBOOT",
5      "timestamp" : "YYYY-MM-DDTHH:MM:SSZ"
6    }] || null,
7    "last_events" : JSONarray<Event4W> || null,
8    "last_intent" : JSON<Intent4W> || null,
9    "relation" : JSON<Relation> || null,
10   "pending" : JSONarray<Task> || null
11  }]
```

[POST] /worker/devices/:id/intents *Recibe la actividad de un dispositivo en un periodo de tiempo.*

BODY:

```
1  { "from" : "YYYY-MM-DDTHH:MM:SSZ" || null,
2    "to" : "YYYY-MM-DDTHH:MM:SSZ" || null }
```

RESPONSE:

Http Status Code 200 : Ok

```
1  [{
2    "device" : "mac",
3    "datetime" : "YYYY-MM-DDTHH:MM:SSZ",
4    "data" : JSON<IntentData>,
5    "slots" : JSONarray<SlotData>
6  }]
```

[POST] /worker/towns *Añade una nueva localidad al sistema.*

BODY:

```
1  { "name" : "Laguna de Duero",
2    "postcode" : 47140,
3    "latitude" : 41.585467,
4    "longitude" : -4.725943
5  }
```

6.3. IMPLEMENTACIÓN

RESPONSE:

Http Status Code 200 : Ok

```
1 Town Laguna de Duero successfully added.
```

[GET] */worker/towns* *Obtiene todas las localidades del sistema.*

RESPONSE:

Http Status Code 200 : Ok

```
1  [{
2    "code" :47,
3    "name" :Valladolid,
4    "locations" :[{
5      "name" :"Laguna de Duero",
6      "postcode" :47140,
7      "latitude" :41.585467,
8      "longitude" :-4.725943
9    }]
10  }]
```

[GET] */worker/towns/:postalcode* *Obtiene una localidad específica a través de su código postal.*

RESPONSE:

Http Status Code 200 : Ok

```
1  {
2    "name" :"Laguna de Duero",
3    "postcode" :47140,
4    "latitude" :41.585467,
5    "longitude" :-4.725943
6  }
```

[POST] */worker/relations* *Asigna un dispositivo a un usuario.*

BODY:

```
1  { "nif" : "124232xxG",
2    "device" : "mac"
3  }
```

RESPONSE:

Http Status Code 200 : Ok

```
1 Added relation.
```

[DELETE] */worker/relations/:device* Elimina la relación que tiene actualmente el dispositivo especificado.

RESPONSE:

Http Status Code 200 : Ok

```
1 Finished relation.
```

[GET] */worker/relations/:device* Obtiene la última relación de un dispositivo específico.

RESPONSE:

Http Status Code 200 : Ok

```
1 {
2   "device" : "mac",
3   "user"  : JSON<Person>,
4   "from"  : "YYYY-MM-DDTHH:MM:SSZ",
5   "to"    : "YYYY-MM-DDTHH:MM:SSZ" || null,
6   "position" : {"lat" : 41.123, "lon": -4.56} || null,
7   "info"  : "information" || null
8 }
```

[GET] */worker/relations/:device* Obtiene la última relación de un dispositivo específico.

RESPONSE:

Http Status Code 200 : Ok

```
1 {
2   "device" : "mac",
3   "user"  : JSON<Person>,
4   "from"  : "YYYY-MM-DDTHH:MM:SSZ",
5   "to"    : "YYYY-MM-DDTHH:MM:SSZ" || null,
6   "position" : {"lat" : 41.123, "lon": -4.56} || null,
7   "info"  : "information" || null
8 }
```

6.3. IMPLEMENTACIÓN

[POST] */worker/login* Inicio de sesión de un administrador.

BODY:

```
1  {
2    "user" : "admin",
3    "password" : "La-Que-Sea-2020"
4  }
```

RESPONSE:

Http Status Code 200 : Ok

```
1  {
2    "access_token" : "Bearer lo-que-sea-de-acceso",
3    "refresh_token" : "Bearer lo-que-sea-de-refresco"
4  }
```

[POST] */worker/person* Añade nuevo cliente al sistema.

BODY:

```
1  {
2    "name" : "Alvaro",
3    "surname" : "Velasco",
4    "nif" : "124232xxG",
5    "postcode" : "47140"
6  }
```

RESPONSE:

Http Status Code 200 : Ok

[GET] */worker/people* Obtener todos los clientes del sistema.

RESPONSE:

Http Status Code 200 : Ok

```
1  [{
2    "name" : "Alvaro",
3    "surname" : "Velasco",
4    "nif" : "124232xxG",
5    "postcode" : "47140"
6  }]
```

6.3.3. FrontEnd

Estructura de directorios

La parte del sitio web del sistema se ha desarrollado con VueJS a través del instalador de gradle, lo que proporciona un sistema de directorios similar a cualquier otro proyecto de VueJS, de modo que es más simple entender el funcionamiento.

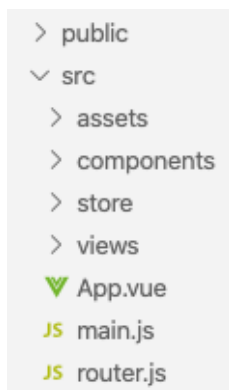


Figura 6.11: Estructura del FrontEnd: Model

En cuanto a una explicación rápida, el proyecto tendrá dos directorios sobre los que se trabaja, y que pueden observarse en la figura 6.27:

1. **/public** el cual almacena el fichero *index.html* del sistema, donde se pueden importar todos los script, al igual que almacena los posibles iconos del sistema, siendo de acceso público. En cuanto a la práctica, no se recomienda almacenar ahí ningún fichero de carácter sensible, ya que es de fácil acceso al público.
2. **/src** contiene la lógica del sitio web, y funciona del siguiente modo: */src/main.js* carga la aplicación Vue, y al ser el módulo principal llevará el control de todas las peticiones que se realizan a la aplicación. Por ello, se ha instalado dentro un interceptor de peticiones, el cual añade el token de acceso al sistema en todas y cada una de ellas, al igual que lo refresca en caso de que se caduque.

Este módulo por tanto carga la aplicación, *App.vue*, la cual sirve de contenedor para mostrar todas las vistas posibles. Para mostrar estas vistas, se seleccionan a través del enrutador el cual está definido en */src/router.js*. Las vistas que pueden ser cargadas deben almacenarse en */src/views*, y son las que acceden a los datos de la aplicación. Los datos de la aplicación se pueden almacenar tanto en las propias vistas si solo son necesarios en esa vista, o en el controlador si se va a requerir compartir la información con otras vistas.

Los controladores están en */src/store* y almacenan tanto la información generada por la aplicación, como la que obtienen de servicios externos. Para un desarrollo mejor

estructurado de cada vista, el contenido puede ser dividido en componentes, de modo que cada vista sea un conjunto de cada uno de ellos y permitiendo exportar esos componentes una vez estén implementados a otras aplicaciones, al igual que pudiendo traer componentes externos. Estos componentes estarán en `/src/components` y toda la información de la que puedan disponer será la generada por ellos mismos, o la información que se les ha introducido a través de atributos, pero nunca accederán al controlador del sistema.

Se considera mala práctica que estos componentes accedan a métodos de sus padres, *siendo el padre la vista que lo carga, o otro componente que lo cargue*, ya que crearía una dependencia que lo convertiría en componente no aislado, produciendo errores si se incluye en un nuevo padre que no implemente esos métodos, por lo que si se requiere llamar a un antecesor, lo recomendable sería la emisión de eventos, los cuales podrían ser recogidos y tratados por el padre.

En cuanto a la inclusión de scripts externos para la mejora del sistema, pueden ser añadidos via instalación por comandos con `npm`, o pueden ser descargados e incluidos en el directorio `/src/assets`.

Diseño de interfaz

El diseño de la interfaz ha sido basado en la implementación de los requisitos del sistema que han sido definidos en el apartado 4.2.1.

En cuanto a la interfaz de usuario, se ha buscado seguir los patrones de *responsive web design*, permitiendo la adaptación del contenido de la página en función del dispositivo el cual la esté cargando, pudiendo ser tanto monitores de ordenadores, como tablets o smartphones.

1. Dashboard

Se prediseñó un panel de dashboard en el cual los dispositivos apareciesen como recuadros de colores, que cambiasen su color en función del estado. Al pulsar un color, se abriría una tarjeta en la cual aparecería la información relativa al dispositivo.

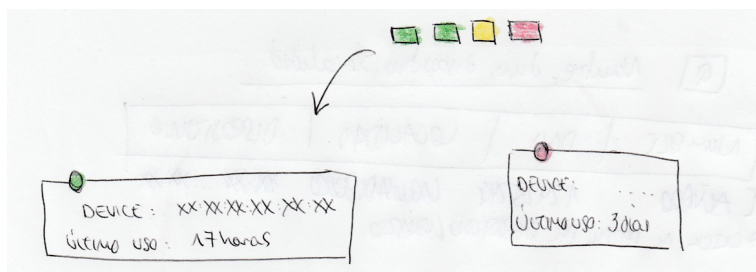


Figura 6.12: Dashboard: Planteamiento de diseño

Finalmente, pese a que esta opción permitía la visualización de cientos de dispositivos a la vez, no proporcionaba una buena identificación de qué recuadro era un dispositivo

concreto, o cuánto tiempo exacto había pasado. Por ello, se optó por mostrar directamente las tarjetas de los dispositivos, cambiando la forma en que aparecen los colores y haciéndola más fácil de comprender, localizar e identificar un dispositivo concreto.

En las siguientes figuras aparecen las posibles variaciones de las tarjetas.



Figura 6.13: Dashboard: Dispositivo en línea

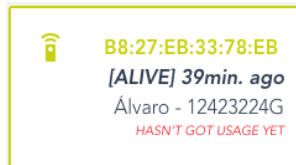


Figura 6.14: Dashboard: Dispositivo reciente



Figura 6.15: Dashboard: Dispositivo ausente



Figura 6.16: Dashboard: Dispositivo posiblemente apagado

6.3. IMPLEMENTACIÓN

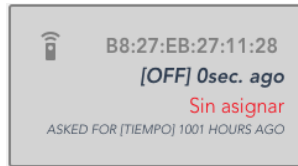


Figura 6.17: Dashboard: Dispositivo apagado por comando

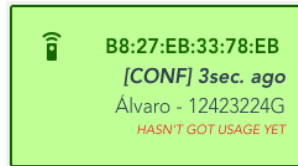


Figura 6.18: Dashboard: Dispositivo realizando tarea enviada

En el dashboard, con el fin de poder localizar más rápidos los dispositivos que cumplan una serie de condiciones, se debe permitir un filtrado de ellos:

- a) El dispositivo tiene un usuario asignado o no.
- b) Se ordenan por fecha de su última actividad, o por fecha de su última tarea.
- c) Ese orden es realizado de manera ascendente, o descendente.

El panel que permitiese este filtrado fue prediseñado de manera simple en una ventana emergente que recibe el nombre de *modal* en el mundo del desarrollo web, facilitando de manera simple y efectiva la realización, como se puede ver a continuación.

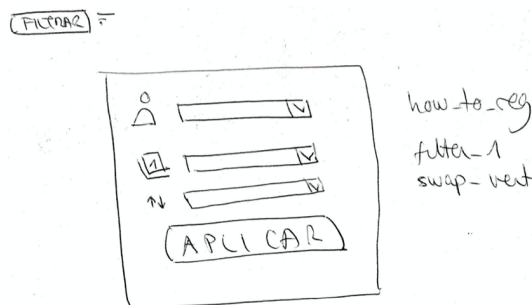


Figura 6.19: Dashboard - Planteamiento del filtro de dispositivos

El diseño final implementado no tuvo ningún cambio ya que el planteamiento del diseño cumplía con las expectativas, ofreciendo una satisfactoria experiencia de usuario.

FILTRAR DISPOSITIVOS

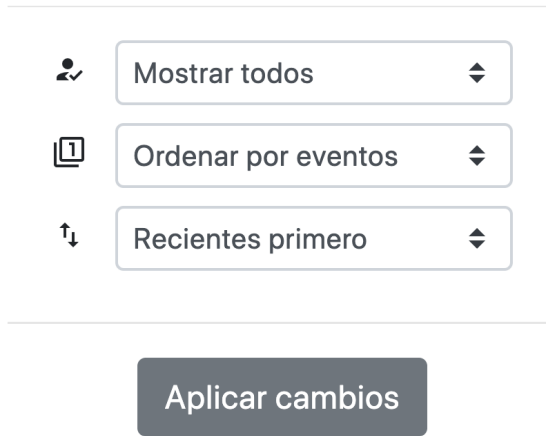


Figura 6.20: Dashboard - Diseño final del filtro de dispositivos

En cuanto a las tarjetas en las cuales aparece la información del dispositivo, deben permitir un acceso a la visualización de estadísticas, o a la selección rápida de tareas. También, debe permitir servir de punto base para acceder a otros servicios del sistema como puede ser la configuración de los parámetros del dispositivo.

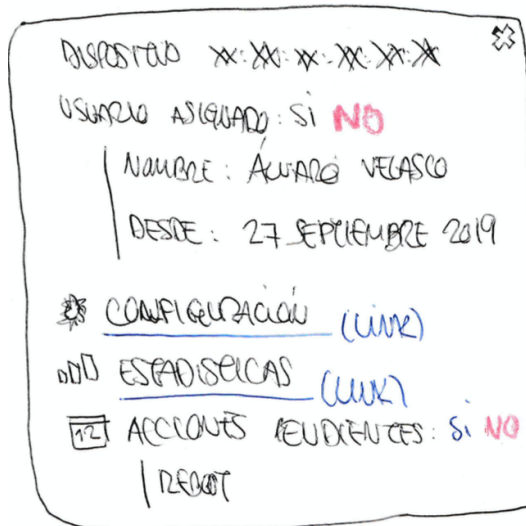


Figura 6.21: Dashboard - Planteamiento de diseño de acciones rápidas.

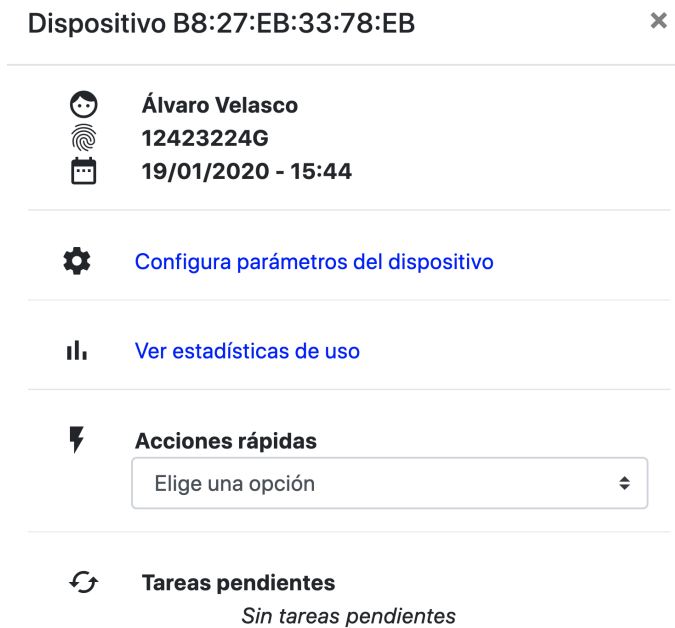


Figura 6.22: Dashboard - Diseño final de acciones rápidas: dispositivo asignado.

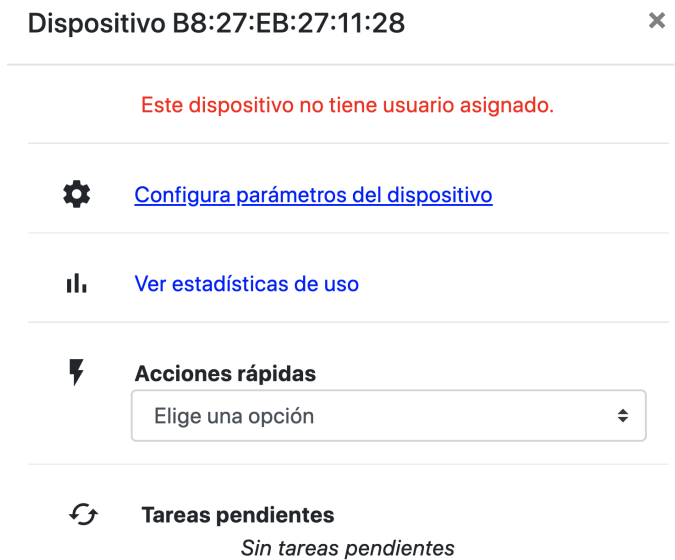


Figura 6.23: Dashboard - Diseño final de acciones rápidas: dispositivo sin asignar.

2. Localidades

En esta sección se buscaba la posibilidad de ver una lista detallada de qué localidades estaban registradas en el sistema, al igual que cuántos dispositivos o personas estaban asociados a cada localidad.

Se contempló como buena idea en los diseños previos la posibilidad de una lista de provincias que contengan alguna localidad registrada mostrando las estadísticas generales, y que pulsando en ellas, se abriese la lista de localidades asociadas, con sus datos particulares.

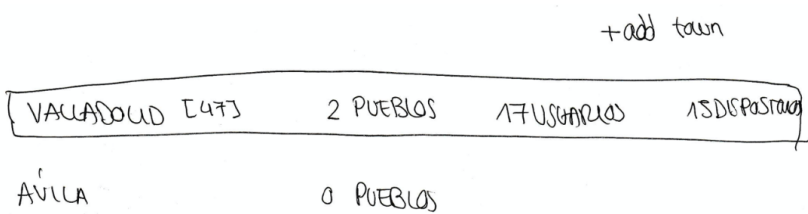


Figura 6.24: Localidades - Planteamiento de lista: agrupado por provincias.

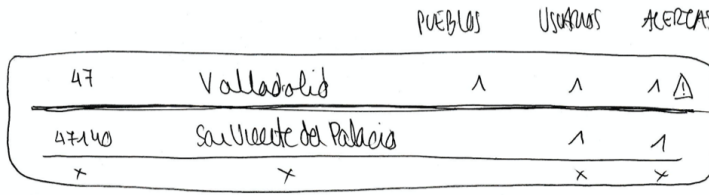


Figura 6.25: Localidades - Planteamiento de lista: despliegue de provincia.

ID	Provincia	Localidad	Personas	Dispositivos
26	La Rioja	1	0	0
40	Segovia	1	0	0
47	Valladolid	3	2	1
[1]	Álava			No hay datos registrados
[2]	Albacete			No hay datos registrados
[3]	Alicante			No hay datos registrados

Figura 6.26: Localidades - Diseño final de lista: agrupado por provincias.

6.3. IMPLEMENTACIÓN

ID	Provincia	Localidad	Personas	Dispositivos
26	La Rioja	1	0	0
40	Segovia	1	0	0
47	Valladolid	3	2	1
47140	Laguna de Duero		2#	1#
47493	San Vicente del Palacio		0#	0#
47151	Boecillo		0#	0#

[1] Álava No hay datos registrados

Figura 6.27: Localidades - Diseño final de lista: despliegue de provincia.

En cuanto al diseño final se puede apreciar la posibilidad de añadir nuevas localidades. Para ello se planteó la posibilidad de añadirlo vía formulario: nombre de la localidad, código postal, provincia, y posición geográfica.

NOMBRE: _____
CÓDIGO POSTAL: _____
PROVINCIA: _____
LATITUD: _____
LONGITUD: _____
AÑADIR

Figura 6.28: Localidades - Planteamiento de diseño para añadir nueva localidad.

Este prediseño, se ve que no es muy útil ya que la posición geográfica tendríamos que buscarla en otros servicios de mapas, o por que se debería introducir demasiados datos.

Finalmente se rediseñó, cambiando este prediseño por la implementación de un mapa que permitiese la búsqueda de la localidad, y añadirla seleccionándola en el mapa, de tal manera que se guarden todos los datos del formulario automáticamente sin necesidad de rellenarlos a mano, mejorando la experiencia de usuario.

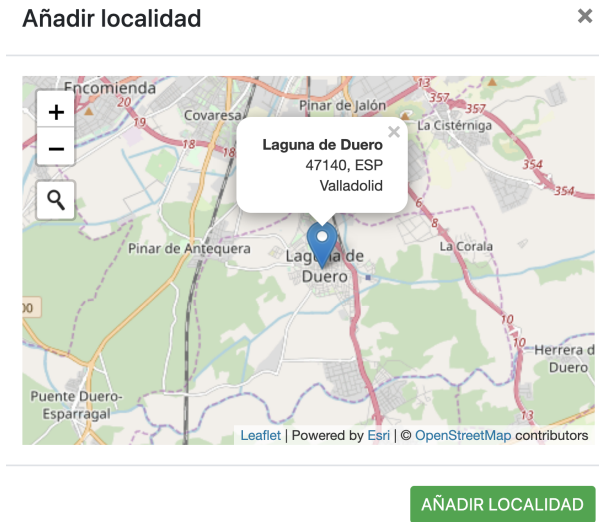


Figura 6.29: Localidades - Diseño final de añadir nueva localidad.

En cuanto a la lista de localidades desplegada, un click sobre una localidad específica nos debería llevar al panel de configuraciones, permitiendo cambiar también la configuración de esa localidad.

3. Settings

En esta sección se permite cambiar las configuración preestablecida global. En caso de que se acceda a través de un enlace de una localidad, se permitirá cambiar la configuración de esa localidad, y en caso de acceder desde un dispositivo, se permitirá cambiar tango la global, como la del dispositivo, como la del pueblo del usuario asignado, en caso de que lo tenga.

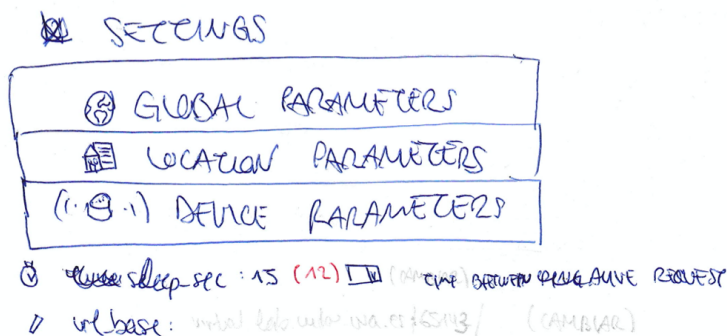


Figura 6.30: Settings - Planteamiento de diseño de configuraciones.

De este planteamiento de diseño se han podido sacar aspectos interesantes, como la

6.3. IMPLEMENTACIÓN

posibilidad de mostrar la configuración específica de un dispositivo, o si tiene aún alguna configuración pendiente específica por instalar mostrando esos valores pendientes en rojo al lado del valor que tiene instalado.

En cuanto a la implementación real, se muestra cómo solo se ha definido al configuración de tiempo entre avisos, pudiendo ser esta parametrización fácilmente ampliable añadiendo simplemente nuevos campos, ya que la base y la lógica ya está montada.



Figura 6.31: Settings - Diseño final de configuración global.

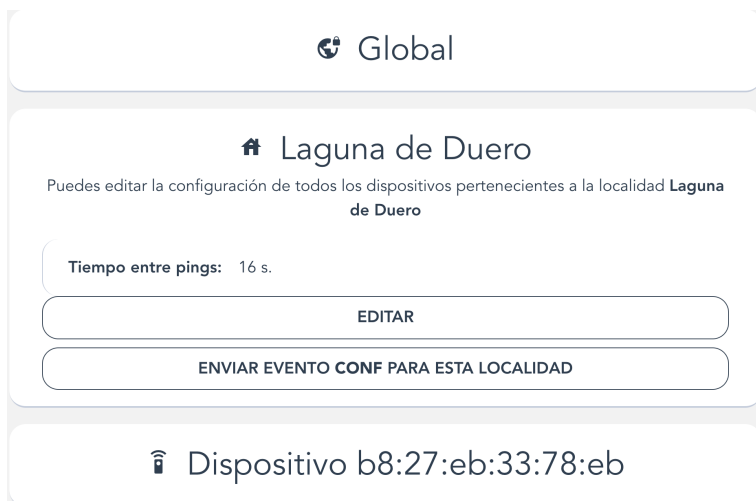


Figura 6.32: Settings - Diseño final de configuración local.



Figura 6.33: Settings - Diseño final de configuración de un dispositivo.

4. Usuarios

Se mostrará una lista simple de todos los usuarios almacenados, de tal manera que aparezcan sus datos básicos, al igual que aparecerá su dispositivo asociado. La lista se debería poder filtrar en función del nombre, dni o dispositivo asociado con el fin de poder encontrar más fácilmente el usuario que se requiera.

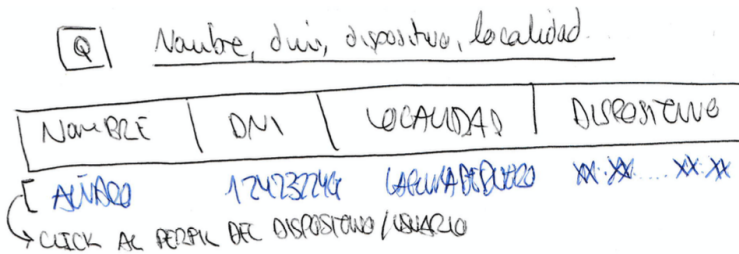


Figura 6.34: Users - Planteamiento de diseño de tabla de usuarios.

Search by name, nif, town, or device...				Añadir usuario
NOMBRE	NIF	CIUDAD	DISPOSITIVO	
Álvaro Velasco	12423224G	Laguna de Duero	b8:27:eb:33:78:eb	
Pedro Pedrin	12XXXXXXG	Laguna de Duero		

Figura 6.35: Users - Diseño final de tabla de usuarios.

6.3. IMPLEMENTACIÓN

También se debe dar la posibilidad de añadir usuarios nuevos al sistema, por lo que se realiza el diseño de un panel para añadir a los usuarios.

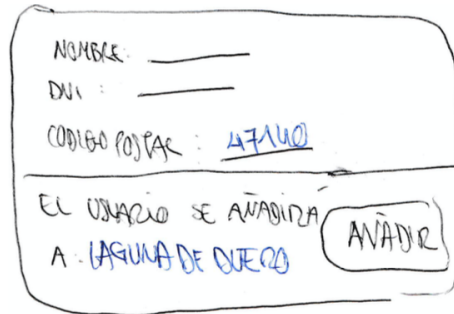



Figura 6.36: Users - Planteamiento de diseño de registro de usuarios

A la hora de añadir usuarios, el código postal debe corresponder con el código postal de alguna ciudad ya registrada en el sistema, por lo que con tan solo ponerlo, ya te dice a qué localidad corresponde, si no, la aplicación web debería avisar sobre la necesidad de registrar anteriormente la localidad.

A screenshot of the final user registration form. The form is titled 'Nuevo usuario' with a close button (X) in the top right corner. It features three input fields, each with a red border and a red 'X' in the top right corner, indicating that the input is invalid. The first field is labeled 'Nombre y apellidos' with a person icon. The second field is labeled 'Documento acreditativo' with a fingerprint icon. The third field is labeled 'Código postal' with a house icon. At the bottom right of the form is a blue button labeled 'AÑADIR USUARIO'.


Figura 6.37: Users - Diseño final de registro de usuarios: código postal no registrado

Nuevo usuario
✕




Benito Antonio Martinez Ocasio

✓



00000000X

✓



47140

✓

Se añadirá a Laguna de Duero

AÑADIR USUARIO

Figura 6.38: Users - Diseño final de registro de usuarios

5. Mapa

La visualización a nivel estatal de la localización puede ser un buen aspecto para el control de los dispositivos, de modo que se pueda observar geográficamente cuántos dispositivos hay, y cómo están de dispersos por toda la geografía.

Para ello se quiere poder mostrar un mapa donde aparezca un icono ubicado en la localización del usuario que lo tiene asignado, por lo que se necesita el diseño de ese icono:

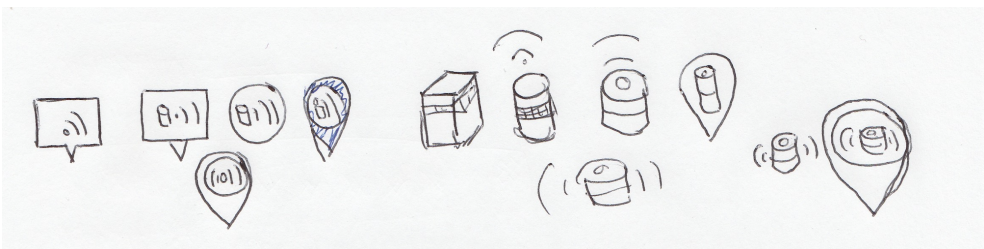


Figura 6.39: Mapa - Prototipo de iconos

El icono deberá aparecer en el mapa de color verde y ubicado en su lugar en caso de que el dispositivo esté asignado a algún usuario. En caso de que no tenga ninguna asignación, aparecerá en color amarillo, ubicado en la costa atlántica.



Figura 6.40: Mapa - Marker de dispositivo asignado



Figura 6.41: Mapa - Marker de dispositivo no asignado

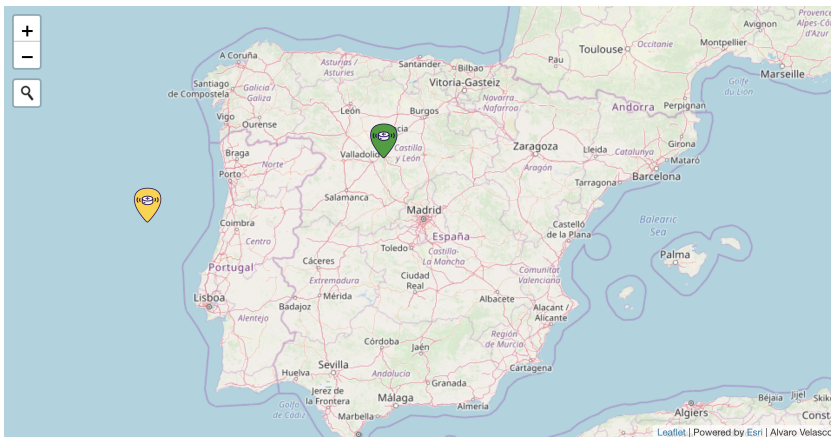


Figura 6.42: Mapa - Diseño final: dispositivos separados

En cuanto a los dispositivos mostrados en el mapa, se requiere que sean agrupados por localidades, mostrando el número en caso de que haya más de uno facilitando la representación y el número de ellos, como se ve en la figura 6.43, ya que mostrar todas las etiquetas juntas sería un problema de visualización. En caso de que se quiera conocer cuales son los dispositivos de la localidad, con tan solo dar click al número se

acercaría el mapa y mostraría todos los markers sin solaparse, colocados en espiral y permitiendo, por tanto, la selección individual de cada uno de ellos.

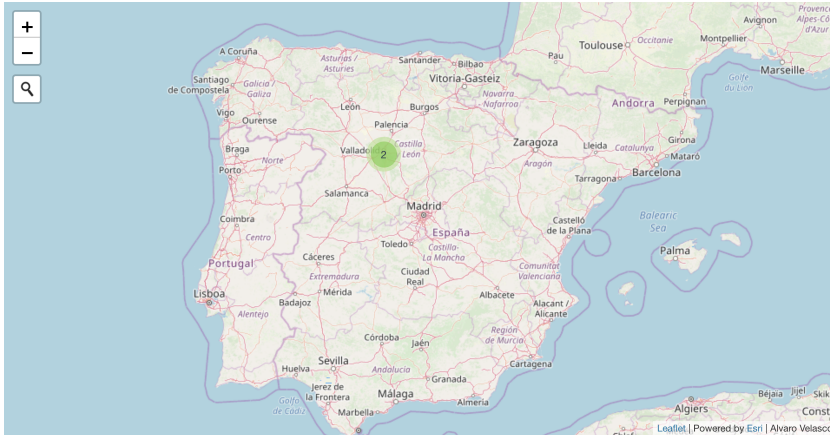


Figura 6.43: Mapa - Diseño final: agrupados por localidades

6. Perfil

Las estadísticas a mostrar en el perfil pueden ser tanto de un dispositivo sin asignar, como de un dispositivo asignado, como de un usuario sin dispositivo asignado.

Debido a estas tres variantes se debe configurar la página de modo que permita ver ciertas funciones, ocultando otras, en función de qué es lo que se está observando.

- a) Usuario sin dispositivo: Debe mostrar únicamente la información básica personal del usuario, y dar la posibilidad de asignar un dispositivo.

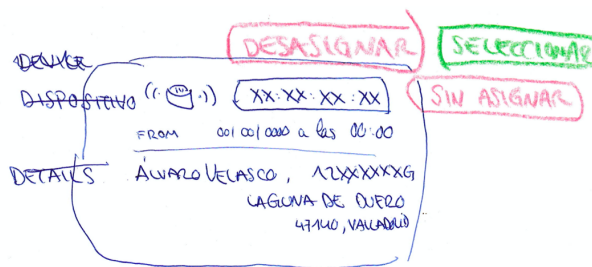


Figura 6.44: Perfil - Planteamiento de diseño de perfil de usuario/dispositivo

El diseño final de esta variante puede verse en la figura 6.52.

- b) Dispositivo sin asignar: Debe permitir ver tanto las estadísticas globales de actividad, como las tareas realizadas, pendientes, y asignar nuevas tareas. Se hace un diseño previo sobre como debería mostrarse la tarjeta con la información del usuario, con la posibilidad de asignar o desasignar un dispositivo.

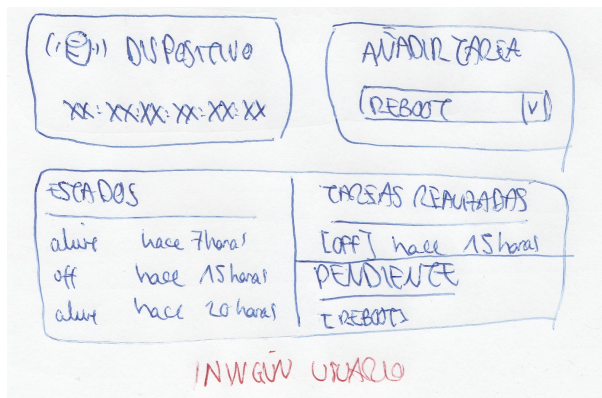


Figura 6.45: Perfil - Planteamiento de diseño de perfil de dispositivo sin usuario

- c) Dispositivo asignado a usuario: Debe permitir ver tanto las opciones anteriores, limitadas por la fecha en que empezó la asignación, al igual que la información del usuario asociado, permitiendo desasignar al usuario el dispositivo, y asociar uno nuevo.

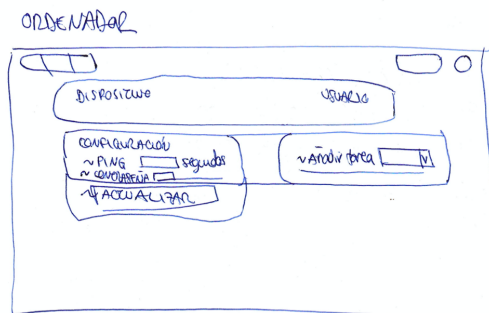


Figura 6.46: Perfil - Planteamiento de diseño de página

En la figura 6.46 se puede observar ya un diseño general de cómo será la vista de la página para todas las variantes, teniendo en cuenta las posibilidades que hemos nombrado anteriormente. En función del estado que se encuentre, se ocultaría el panel de usuario, o saldría la opción de asignar un nuevo dispositivo. También, este prediseño deja ver la colocación de todas las tarjetas, apareciendo la posibilidad de un diseño en el cual se pueda modificar la configuración o añadir tareas a realizar por el dispositivo, que solo estaría visible en caso de haberlo.

Una vez tratada la posibilidad de añadir tareas, la página debería facilitar la tarea de ver qué tareas se han realizado, o qué tareas hay pendientes, al igual que poder ver cómo ha interactuado el usuario con el dispositivo, o ver las estadísticas sobre cuál es la mayor cuestión con la que se interactúa con el dispositivo. Por ello, se plasta otra variante de diseño que permita todas estas funciones, como es representado en la figura 6.47

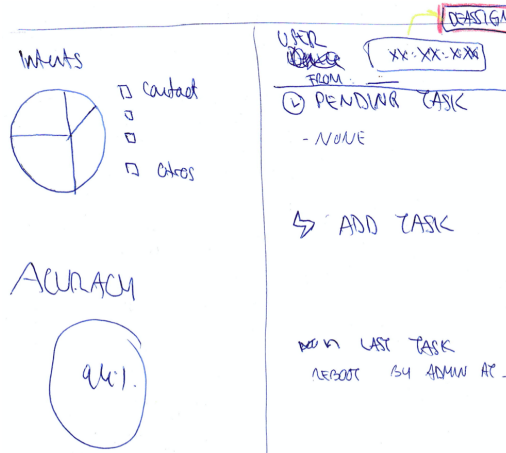


Figura 6.47: Perfil - Planteamiento de diseño estadísticas

Con este prediseño que da pie al muestreo de las estadísticas, una función útil sería el filtrado de ellas por fechas, pudiendo ser ese filtrado por días específicos, por meses, o por años. En cuanto a la posición de los filtros de las estadísticas, se establecerá en la parte superior de la página, favoreciendo la intuición del administrador, mostrando su prototipo en la figura 6.48, que finalmente se ha implementado tal cual se diseñó, como muestra la figura 6.49.



Figura 6.48: Perfil - Planteamiento de diseño de filtro.

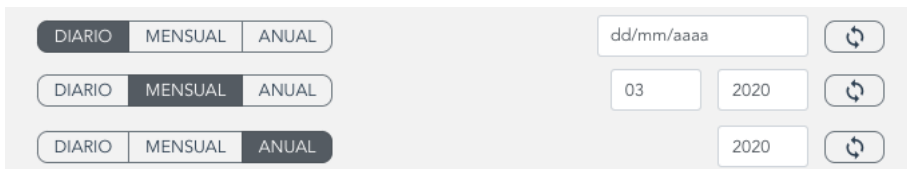


Figura 6.49: Perfil - Diseño final de filtro. (Posibilidades)

Finalmente, debido a la gran variante de posibilidades y prediseños de la página, se ha optado por una conjunción de todos los prediseños, dando prioridad a mostrar las opciones por tarjetas que serán colocadas en orden de mayor a menor uso del administrador, facilitando el acceso a todas las funcionalidades posibles.

6.3. IMPLEMENTACIÓN

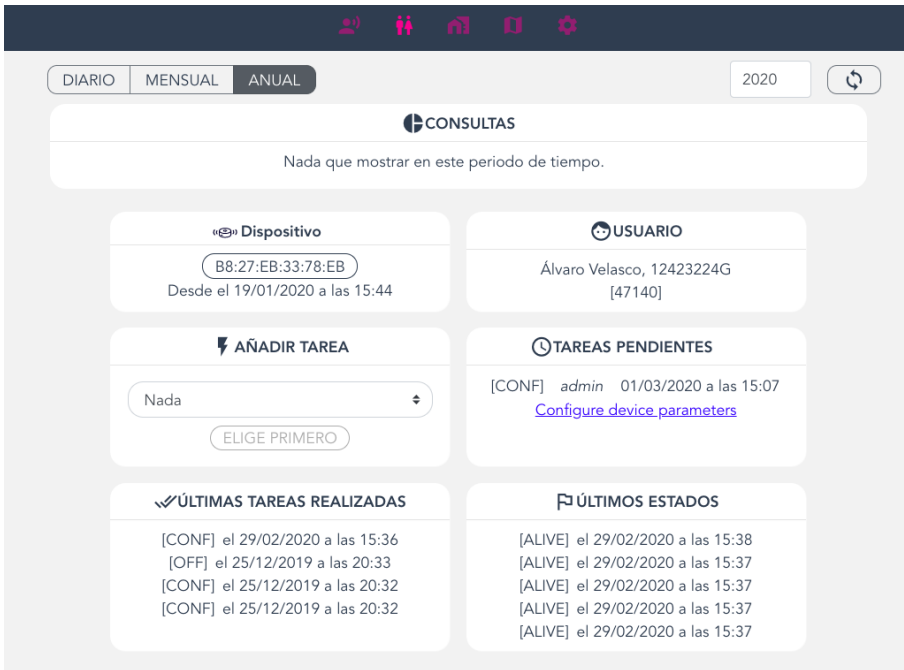


Figura 6.50: Perfil - Diseño final del perfil: Usuario con dispositivo



Figura 6.51: Perfil - Diseño final del perfil: Dispositivo libre

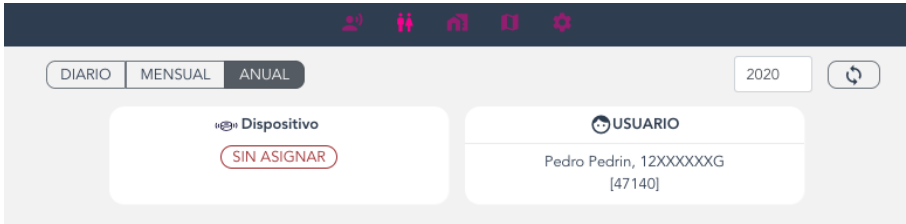


Figura 6.52: Perfil - Diseño final del perfil: Usuario libre

Visto ya todo el sistema web, vemos que existe la posibilidad de asignar un dispositivo a un usuario específico, pero no se muestra como es esa asignación. Para ello, y pensando priorizar la usabilidad, se potencia mostrar una lista de los dispositivos que están disponibles, es decir, que no tienen ningún usuario asociado todavía. Esos dispositivos deberían estar apagados y amontonados en cajas en el despacho del administrador del proyecto. Por ello, y planteando lo que el administrador haría, sería conectar un dispositivo sin asignar, para ver que funciona. Por tanto, en la página, al mostrar los dispositivos sin asignar se ordenan poniendo los primeros los que han realizado un *ping* de manera más reciente, ya que sería el que el administrador acabase de encender, como se muestra el prediseño en la figura 6.53.

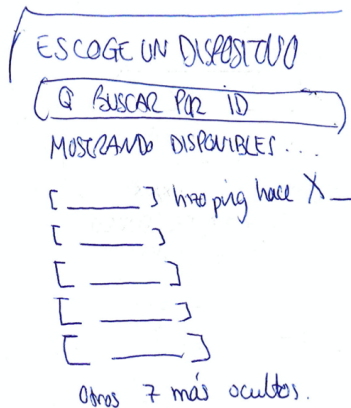


Figura 6.53: Perfil - Pranteamiento de diseño de asignación de dispositivo

En el diseño final, mostrado en la figura 6.54 no se realiza ningún cambio de diseño, dando por bueno y correcto el diseño previo.

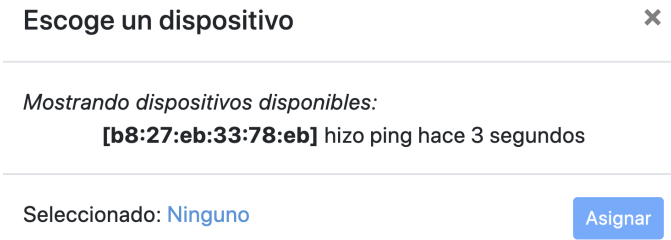


Figura 6.54: Perfil - Diseño final de asignación de dispositivo

En cuanto a las estadísticas sobre la actividad del usuario con el dispositivo, se ha optado por la opción de mostrar de base dos gráficas para las estadísticas:

- a) Doughnut Chart: Para visualizar la relación entre la utilización de las distintas consultas.
- b) Gráfico de barras: Para visualizar a qué horas el dispositivo ha sido utilizado más veces. Esta gráfica puede ayudarnos en un futuro para ver en qué hábitos del día a día se puede mejorar la experiencia del usuario.

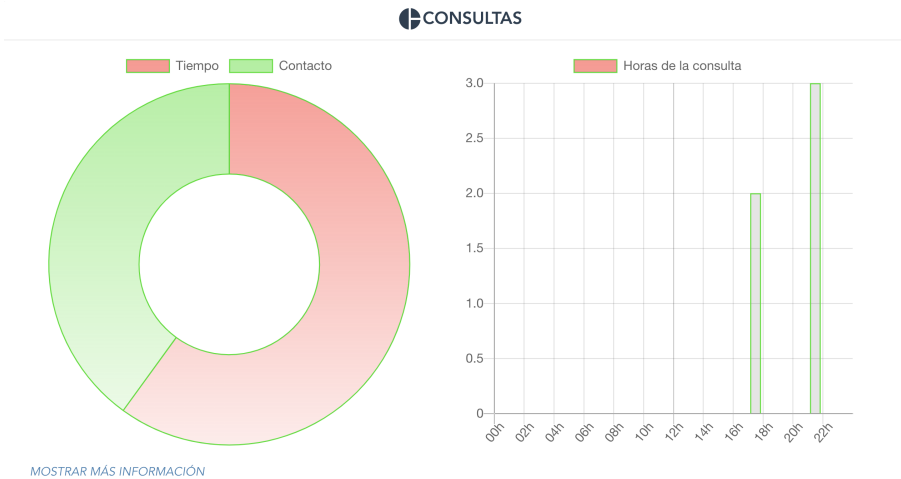


Figura 6.55: Perfil - Diseño final de consultas

En caso de que se sospeche sobre un posible peligro acontecido en el hogar de nuestro usuario, se puede seleccionar la opción de *Mostrar más información*, desplegando una tabla de color azul en la cuál aparecen las últimas 5 consultas que se han hecho al dispositivo, pudiendo identificar una consulta de auxilio, por ejemplo.

Si se quiere más información sobre la información intercambiada en una consulta concreta se puede pulsar en la fila correspondiente, haciendo aparecer una segunda tabla en la cual se muestra la información relativa a esa consulta.

OCULTAR INFORMACIÓN

Fecha	Consulta	Acierto
18-01-2020 a las 21:35	Tiempo	91%
18-01-2020 a las 21:37	Tiempo	91%
18-01-2020 a las 21:38	Tiempo	91%
16-01-2020 a las 17:21	Contacto	91%
16-01-2020 a las 17:32	Contacto	91%

Mostrando slots de la consulta **Tiempo**

Slot	Valor	Acierto
Temperatura	¿hace frío?	76%
Humedad	¿hace frío?	63%
Sensación Térmica	¿hace frío?	83%

Figura 6.56: Perfil - Diseño final consultas: Ampliado

Como se puede observar en la figura 6.50, la vista, en caso de tener un dispositivo asignado, o ser una vista referida a la configuración de un dispositivo, que también se puede ver en la figura 6.51, aparecen diferentes tarjetas, como de las últimas tareas realizadas que corresponde a las ordenadas por un administrador, o los últimos estados del dispositivo, donde se podrá ver si el dispositivo ha estado en uso, o simplemente está conectado (*ALIVE*), lo cual puede ayudar a la identificación de su uso de manera más rápida para el administrador.

Otras tarjetas que se pueden ver son la de añadir una nueva tarea al dispositivo, donde se podría por ejemplo solicitar su actualización o apagado de manera remota. Una vez que se seleccionase una de estas tareas para realizarse, aparecerían en la tarjeta vecina, en la lista de tareas pendientes.

Esta otra vista permite ver tanto las pendientes, como acceder a la configuración del dispositivo, que sería la figura 6.33.

6.4. Pruebas

6.4.1. Introducción

Las pruebas del software es una de las partes más importantes en el desarrollo del software, ya que ayudan a que se cumplan los requisitos establecidos, al igual que aseguran que al realizar un cambio, todo va a seguir funcionando como debería.

Una manera de realizar pruebas es basar el desarrollo en TDD, también conocida como *Test Development Driven*. Este modo de desarrollo se basa en la elaboración de test que describen cómo debe comportarse lo que se va a programar antes de la escritura del código.

6.4.2. Implementación

En este proyecto se ha realizado TDD a la hora de la implementación de los controladores del backend, siguiendo el Red-Green-Refactoring, siendo los controladores la parte de mayor relevancia en cuanto a funcionamiento, ya que con ellos se trata la información recibida por los dispositivos remotos, al igual que las acciones que se van a efectuar sobre ellos.

Para la correcta implementación de TDD, se han creado las siguientes categorías de test:

- BlackBox - Test de caja negra, son los escritos antes de la implementación del código y se basan en las entradas y salidas que los métodos deben cumplir.
- WhiteBox - Test de caja blanca, son los test que comprueban el correcto funcionamiento interno de cada función o método que se quiera implementar.
- Isolation - Son test en aislamiento, utilizando mocks para recibir un determinado comportamiento de las clases externas.
- Sequence - Son test de secuencia, mostrando el comportamiento en determinados escenarios.

A la hora de realizar los test, se ha seguido el mantra de *Red-Green-Refactoring*.

- Red - Fase en la que se escriben los test antes de la implementación del código, de modo que todos fallan.
- Green - Implementación del código de modo que todos los test se vuelvan verdes.
- Refactoring - Proceso de mantener los test verdes pero simplificando la implementación.

De esta manera, se puede confirmar que las partes críticas de este software están probadas, asegurando su correcto funcionamiento.

Para la comprobación de estos test, se puede ejecutar en el repositorio del backend el siguiente comando:

```
gradle clean test
```

Tras la ejecución, que pasa todos los test de manera satisfactoria, se genera el siguiente archivo, en el cual se puede comprobar el estado de cada uno de los test:

```
./build/reports/tests/test/index.html
```


Capítulo 7

Despliegue

En el presente capítulo se explicará los métodos por los cuales se debería poder desplegar todo el sistema de manera que podría dejarse una copia operativa que permitiese el control remoto de dispositivos asistentes.

7.1. Introducción

Para que el sistema no presente fallos, una recomendación es el despliegue por orden de las pautas aquí mostradas, aun que no debería ser problema el despliegue en cualquier otro orden.

La recomendación del orden a seguir en la guía es la siguiente:

1. Despliegue del backend.
2. Despliegue del frontend.
3. Instalación del dispositivo.

7.2. Guía

7.2.1. Despliegue del Back End

Para el despliegue correcto del Back End es necesario seguir las siguientes pautas por el orden que marca la guía:

Instalación de la base de datos

Para la base de datos se utilizará PostgreSQL, como se ha nombrado en la sección 3.5.7. Esta base de datos puede ser sustituida por cualquier otra, como puede ser MySQL, pero deberá ser una base de datos relacional.

Para ello, nos conectamos desde la máquina la cual hará de servidor, y seguimos los pasos:

1. Se obtienen los certificados:

```
$ sudo apt-get install wget ca-certificates
```

2. Se añade la clave de postgresql:

```
$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc  
| sudo apt-key add -
```

3. Se configura:

```
$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/  
lsb_release -cs'-pgdg main" >> /etc/apt/sources.list.d/pgdg.list'
```

4. Se actualiza:

```
$ sudo apt-get update
```

5. Se instala postgresql:

```
$ sudo apt-get install postgresql postgresql-contrib
```

6. Se crea la base de datos:

Para ello, nos conectamos al usuario de postgres

```
$ sudo su - postgres
```

Y accedemos a postgreSQL:

```
$ psql
```

Una vez dentro, se crea la base de datos, que en este caso se va a llamar *assistant*, a la que podemos acceder tecleando simplemente su nombre. La creación de las tablas se va a dejar a Kotlin y a su framework Exposed, para evitar fallos en algún tipo de dato o una mala relación de claves.

```
postgres@psql> CREATE DATABASE assistant;
```

Con esto sería suficiente, pero para proporcionar seguridad se procede al cambio de contraseña del usuario de postgres:

```
postgres@psql> ALTER USER postgres PASSWORD 'nueva_contrasea';
```

Es muy importante recordar la contraseña, ya que es necesario utilizarla posteriormente en el despliegue del backend para permitir su acceso. Para el despliegue temporal, la contraseña utilizada será: **cu4lquie.Rar**

En este punto, la base de datos ya estaría disponible para la conexión desde dentro del servidor, localizándola en el **puerto 5432**.

Configuración y despliegue del backend

Para su despliegue de prueba, únicamente es necesaria la localización del archivo **JAR** y su colocación dentro del servidor en el directorio */home/assistant/core*. Este paso no es relevante, pero es recomendable para una correcta localización del archivo, ya que los logs del sistema se almacenarán y archivarán en una carpeta contenedora de esa ruta, de modo que todo se tendría mejor ordenado.

Una vez con el fichero en esa ruta, tan solo se tiene que acceder a ese directorio y ejecutar el archivo:

```
$ cd /home/assistant/core
$ java -jar assistant.core.jar
```

Desplegado de esta manera se podría visualizar los logs del sistema en tiempo real, pero con el impedimento de que no se podría cerrar la consola. Para evitar este problema, se puede ejecutar de la siguiente manera, ocultando las salidas con el comando **nohup** y estableciéndolo en segundo plano con el símbolo **&**

```
$ cd /home/assistant/core
$ nohup java -jar assistant.core.jar &
```

7.2. GUÍA

En caso de querer finalizar la ejecución, tan solo habría que finalizar el proceso. Para ello obtenemos la lista de procesos:

```
$ ps -ef | grep java
```

Lo que nos mostraría la lista de procesos java ejecutados en la máquina, de donde se puede ver el número de proceso que corresponde a nuestro backend, y se eliminaría de la siguiente forma:

```
$ kill -9 numerodeproceso
```

El acceso al sistema desplegado de prueba podría darse a través de las credenciales de un administrador, las cuales tendrían un usuario llamado **admin**, al que le corresponde la contraseña **seren0314**.

En caso de querer cambiar la configuración por defecto, o los datos de acceso a la base de datos como el usuario y contraseña que se ha mencionado al comienzo de la guía, se deberá cambiar los valores establecidos del proyecto, que estan configurados en el siguiente fichero:

```
/src/controller/model/util/Credentials.kt
```

Una vez estos datos hayan sido actualizados, se deberá volver a formar el archivo JAR, por lo que nos ayudaremos de la herramienta de gradle, que será ejecutada desde dentro de la carpeta del repositorio:

```
$ gradlew shadowJar
```

Tras este comando, se generará **assistant.core.jar**, el cual está localizado en *./build/lib/*.

Una vez el archivo JAR está generado, se repetirían los pasos descritos al principio de la sección donde se ha descrito la posibilidad de un despliegue de prueba.

Una vez desplegado, será accesible a través del puerto 8082.

7.2.2. Despliegue del Front End

Como ya se ha mencionado en numerosas ocasiones, el front end está elaborado con VueJS, el cual está integrado con el módulo de paquetes de Node, también conocido como **npm**. Esto nos permite la compresión de la totalidad del proyecto en tan solo clases HTML, CSS y Javascript, además de los archivos de recursos como pueden ser las imágenes.

Para la realización de esta compresión tan solo hay que acceder a la carpeta del repositorio y ejecutar:

```
$ npm run build
```

Lo cual generará una carpeta en el mismo repositorio llamada **dist**.

El despliegue del front end se realizará por tanto copiando el contenido de esta carpeta y pegándolo dentro del servidor apache, que en el caso habitual se encuentra en la siguiente ruta:

```
/var/www/html
```

y siendo accesible por tanto a través del puerto 80 de la ubicación de nuestro servidor.

7.2.3. Instalación del Dispositivo

La instalación del software en el dispositivo podrá ser flasheando una imagen con el sistema operativo y el asistente ya cargado.

La guía por tanto, no entrará en ese apartado ino en la posibilidad de instalar el asistente en cualquier dispositivo que disponga de una arquitectura Unix.

Para ello, se introducirá el repositorio **assistant.task** en el dispositivo, dentro del directorio y aprovechando los scripts contenidos en él, se instalará todo lo necesario.

1. Acceso al repositorio:

```
$ cd /home/pi/assistant.task
```

2. Instalación de dependencias

```
$ sh requirements.sh
```

Tras la ejecución del script ya se instalan las dependencias requeridas y se configura el inicio automático del controlador remoto del asistente en cada reinicio de la máquina.

Capítulo 8

Conclusiones y Trabajo Futuro

8.0.1. Conclusiones

En la realización de este proyecto se ha puesto a prueba la mayor parte de los conocimientos adquiridos a lo largo del grado de Ingeniería Informática.

Las características de este proyecto, al igual que su dinamismo, han permitido demostrar los conocimientos ya que este proyecto no se ha centrado únicamente en una interfaz de usuario, un controlador de un dispositivo, o la creación de una API, lo que permitiría la práctica de un conocimiento conocido como vertical, sino que han sido desarrollados todos ellos, permitiendo un crecimiento horizontal.

8.0.2. Trabajo Futuro

La elaboración de este proyecto deja abiertas las puertas a la elección e implementación final del asistente, ya que el presente documento narra la manera en la cual se ha implementado el controlador del dispositivo y cómo se puede manejar de manera remota un dispositivo, dejando sin implementar el asistente inteligente pero permitiendo la inclusión de cualquiera de los nombrados en el apartado 2.3, o cualquiera de los que serán creados en el futuro.

También, la manera de trabajar del controlador del dispositivo permite la inclusión sencilla de nuevas tareas, sin requerir la modificación de los archivos contenedores de las ya existentes.

En cuanto a trabajo futuro, se contempla la posibilidad de añadir nuevas fases en las que el backend explore los resultados obtenidos al buscar las localidades que tiene registradas, con el fin de recabar archivos que formarán parte de la caché de los dispositivos en los cuales aparecerá información como establecimientos y horarios de ese municipio, con el fin de que el usuario pueda preguntar al asistente sobre ello.



Bibliografía

- [1] JAVIER P. (2018, Abril 10). *Xataka: El escándalo de Cambridge Analytica resume todo lo que está terriblemente mal con Facebook*. Recuperado a 16 de Enero de 2020, de <https://www.xataka.com/privacidad/el-escandalo-de-cambridge-analytica-resume-todo-lo-que-esta-terriblemente-mal-con-facebook>
- [2] THE NIELSEN COMPANY (2018). *Nielsen: Despite their vast capabilities, smart speakers are all about the music*. Recuperado a 17 de Enero de 2020, de <https://www.nielsen.com/us/en/insights/article/2018/smart-speaking-my-language-despite-their-vast-capabilities-smart-speakers-all-about-the-music/>
- [3] BOE.ES (2019, Junio 25). *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*. Recuperado a 17 de Enero de 2020, de <https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673>
- [4] PARLAMENTO EUROPEO (2016, Mayo 25). *RGPD - Reglamento General de Protección de Datos*. Recuperado a 17 de Enero de 2020, de <https://rgpd.es/>
- [5] LABORATORIO DE PERIODISMO LUCA DE TENA (2019, Octubre 17). *El 42% de los usuarios de altavoces inteligentes en España escuchan noticias a través de ellos*. Recuperado a 19 de Enero de 2020, de <https://laboratoriodeperiodismo.org/altavoces-inteligentes-espana-noticias/>
- [6] 20MINUTOS (2016, Junio 5). *Google graba y almacena tu voz: dónde puedes escucharla (y borrarla)*. Recuperado a 17 de Enero de 2020, de <https://www.20minutos.es/noticia/2762528/0/google-graba-almacena-tu-voz-donde-como-borrarlo/>
- [7] PÍXEL, EL MUNDO. (2019, Julio 4) *Escándalo en Amazon: la empresa reconoce que guarda tus conversaciones para siempre*. Recuperado a 19 de Enero de 2020, de <https://www.elmundo.es/tecnologia/2019/07/04/5d1ccf42fc6c833f3f8b460d.html>

- [8] MYCROFT AI (n.d.). *Documentation*. Recuperado a 20 de Enero de 2020, de <https://mycroft-ai.gitbook.io/docs/>
- [9] MYCROFT AI (n.d.). *Community*. Recuperado a 20 de Enero de 2020, de <https://community.mycroft.ai/>
- [10] PROYECTOSAGILES.COM (n.d.). *Desarrollo iterativo e incremental*. Recuperado a 22 de Enero de 2020, de <https://proyectosagiles.org/desarrollo-iterativo-incremental/>
- [11] JETBRAINS, (n.d.). *Ktor Servers*. Recuperado a 18 de Enero de 2020, de <https://ktor.io/servers/index.html>
- [12] INSERTKOINIO (2019, Noviembre 5). *Koin*. Recuperado a 18 de Enero de 2020, de <https://github.com/InsertKoinIO/koinreadme>
- [13] OAUTH.NET (n.d.). *OAuth 2.0*. Recuperado a 18 de Enero de 2020, de <https://oauth.net/2/>
- [14] SMARTBEAR.COM (n.d.). *What is an API Endpoint?*. Recuperado a 18 de Enero de 2020, de <https://smartbear.com/learn/performance-monitoring/api-endpoints/>
- [15] MYNDOCS (2019, Septiembre 7). *Kotlin OAuth2 server*. Recuperado a 18 de Enero de 2020, de <https://github.com/myndocs/kotlin-oauth2-server>
- [16] JETBRAINS (2019, Junio 3). *Exposed SQL DSL*. Recuperado a 18 de Enero de 2020, de <https://github.com/JetBrains/Exposed>
- [17] JONATHAN H. (n.d.). *Download and install JSoup*. Recuperado a 19 de Enero de 2020, de <https://jsoup.org/download>
- [18] BEN R. (2019, Diciembre 5). *Vue vs React: Which is the Better Framework?*. Recuperado a 19 de Enero de 2020, de <https://buttercms.com/blog/vue-vs-react-which-is-the-better-framework>
- [19] BOOTSTRAP-VUE (n.d.). *Components*. Recuperado a 19 de Enero de 2020, de <https://bootstrap-vue.org/docs/components>
- [20] LEAFLET(n.d.). *An open-source JavaScript library for mobile-friendly interactive maps*. Recuperado a 19 de Enero de 2020, de <https://leafletjs.com/reference-1.6.0.html>

- [21] ORGANIZACIÓN MUNDIAL DE LA SALUD (2020, Marzo 11). *WHO Director-General's opening remarks at the media briefing on COVID-19*. Recuperado a 31 de Marzo de 2020, de <https://www.who.int/dg/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19—11-march-2020>
- [22] UNIVERSIDAD DE VALLADOLID (n.d.). *Preguntas Frecuentes*. Recuperado a 21 de Enero de 2020, de <http://www.uva.es/export/sites/uva/2.docencia/2.02.mastersoficiales/2.02.13.preguntasfrecuentes/index.html>
- [23] INDEED.COM (2019, Diciembre). *Salarios para empleos de Programador/a junior en España*. Recuperado a 21 de Enero de 2020, de <https://es.indeed.com/salaries/programador-junior-Salaries>
- [24] BOB HUGHES AND MIKE COTTERELL (2009). *Software Project Management*. McGraw-Hill Education.
- [25] JEREMY L. (2014, Abril 24). *Model-View-ViewModel (MVVM) Explained*. Recuperado a 23 de Enero de 2020, de <https://www.wintellect.com/model-view-viewmodel-mvvm-explained/>
- [26] EDER N. (2017, Octubre 9). *The Vue architecture that worked for me. (in small and large apps)*. Recuperado a 23 de Enero de 2020, de <https://medium.com/@ederng/the-vue-architecture-that-worked-for-me-in-small-and-large-apps-9b253cf92951>
- [27] VIVEK G. (2019, Julio 19). *Linux Execute Cron Job After System Reboot*. Recuperado a 24 de Enero de 2020, de <https://www.cyberciti.biz/faq/linux-execute-cron-job-after-system-reboot/>
- [28] ROBERT C. MARTIN (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson.