Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Ecosistema Android multi-aplicación con servicio REST para la detección de caídas y la gestión de incidencias, alarmas y eventos

Autor: Marina Manzano Gamito
Tutor: María Teresa Ariza Gómez

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2025

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Ecosistema Android multi-aplicación con servicio REST para la detección de caídas y la gestión de incidencias, alarmas y eventos

Autor:
Marina Manzano Gamito

Tutora:
María Teresa Ariza Gómez
Profesora Titular

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2025

Trabajo Fin de Grado: Ecosistema Android multi-aplicación con servicio REST para la detección de caídas y la gestión de incidencias, alarmas y eventos

Autor: Marina Manzano Gamito

Tutor: María Teresa Ariza Gómez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2025

El Secretario del Tribunal

A mi familia

A mis amigos

Agradecimientos

Quiero aprovechar esta pequeña sección para agradecer a todas las personas que han hecho posible que pueda escribir esto.

En primer lugar, agradecer a mi familia, mis padres, Joaquín y Mª Asun, por su apoyo incondicional, por escucharme y comprenderme durante toda la carrera, y por ser siempre un lugar seguro al que acudir. A mi hermano Joaquín, por ser una inspiración y un ejemplo a seguir. A mis abuelos, Adolfo, Asunción, Mari y Pepe, por alegrarse, incluso más que yo, con cada pequeño logro conseguido en el camino, por cada 2,5% que sumaba a mi porcentaje de ingeniera con cada asignatura aprobada. Sin duda, este trabajo, que supone completar el 100%, va para ellos, en especial para Pepe.

En segundo lugar, quiero agradecer también a mis amigas, por ser un lugar al que acudir siempre cuando necesito desconectar, desahogarme o simplemente reír. Gracias por estar siempre para reír y llorar juntas.

También quiero agradecer a mi pareja, Antonio Jesús, por ser el mejor compañero que podría tener en este camino, por ayudarme, darme ánimos y creer siempre en mí. Te quiero.

No puedo dejar de mencionar a la que fue mi profesora de informática en el instituto, María Dolores. Ser tu alumna ha sido probablemente lo más determinante para decidirme por esta carrera. Gracias por transmitir tu pasión.

Por último, gracias a todos los profesores que me han acompañado estos años, en especial a mi tutora, María Teresa Ariza Gómez, por haberme dado la oportunidad de realizar este trabajo, por sus consejos y por su orientación a lo largo del desarrollo de este.

Sin cada uno de vosotros, el camino hubiera sido, sin duda, mucho más difícil. Gracias de corazón.

Marina Manzano Gamito

Sevilla, 2025

Resumen

En la actualidad, la tecnología juega un papel crucial en la vida de las personas y en su calidad de vida. Sin embargo, su uso no resulta accesible para algunos colectivos, como personas mayores o en situación de dependencia que viven solas o que requieren ayuda en su día a día. En este contexto se ha diseñado y desarrollado un sistema compuesto por tres aplicaciones Android que hacen uso de un servicio REST y una base de datos, con el fin de ofrecer una solución accesible para la gestión de alarmas, eventos e incidencias.

La primera de las aplicaciones Android desarrolladas, SensApp, está dirigida a los usuarios finales y les permite visualizar sus eventos y alarmas diarias, detectar caídas y crear incidencias. La segunda aplicación, SensApp Control, está enfocada a los responsables de los usuarios o sus familiares, estos pueden acceder a los datos del usuario y crear o editar alarmas y eventos para este. Por último, SensApp Plus está dividida en dos módulos, uno para operadores, encargados de atender las incidencias existentes, y otro para administrativos, donde pueden gestionar los usuarios, responsables y operadores y además, acceder a todos los datos necesarios. Las tres aplicaciones hacen uso del servicio REST desarrollado para la persistencia de datos.

El objetivo principal de este Trabajo de Fin de Grado es, por tanto, ofrecer un sistema tecnológico que mejore la calidad de vida de las personas mayores o aquellas que requieran ayuda, facilitándoles el uso de la tecnología, permitiendo así que puedan incorporarla a su día a día con el objetivo de fomentar su autonomía.

Abstract

Nowadays, technology plays a crucial role in people's lives and their quality of life. However, its use is not accessible to some groups, such as elderly or dependent people who live alone or require help in their daily lives. In this context, a system composed of three Android applications has been designed and developed, using a REST service and a database to provide an accessible solution for management of alarms, events, and incidents.

The first of the Android applications developed, SensApp, is addressed to end users and allows them to visualize their daily events and alarms, detect falls, and create incidents. The second application, SensApp Control, is designed for caregivers or family members, allowing them to access the user data and create or edit alarms and events for the user. Finally, SensApp Plus is divided into two modules, one for operators, who are responsible for handling existing incidents, and another one for administratives, who can manage users, responsibles for the users, and operators, as well as access all necessary data. All three applications utilize the REST service developed for data persistence.

The main objective of this project is, therefore, to offer a technological system that improves the quality of life of elderly people or those who require assistance, facilitating the use of technology, allowing them to incorporate it into their daily lives, promoting their autonomy.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

API	Interfaz de Programación de Aplicaciones
AWS	Amazon Web Services
BBDD	Base de Datos
EC2	Elastic Compute Cloud
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
REST	Representational State Transfer
SCP	Secure Copy Protocol
SQL	Structured Query Language
SSH	Secure Shell
URL	Uniform Resource Locator

1 INTRODUCCIÓN

Un viaje de mil millas empieza con un simple paso.

Lao Tse

1.1 Motivación

En la actualidad, estamos viviendo un momento en el que el auge de la tecnología ha transformado por completo nuestro estilo de vida, mejorando enormemente la calidad de vida de las personas. Sin embargo, a pesar de que el mundo está cada vez más digitalizado, algunos grupos de personas no se han visto beneficiados por estas mejoras debido a las barreras tecnológicas que encuentran en su uso.

En este contexto surge este Trabajo de Fin de Grado, cuyo objetivo principal es ofrecer una solución accesible a personas mayores o que se encuentren en una situación de dependencia, para permitirles llevar un mejor control de sus rutinas y eventos en su día a día, fomentando así su autonomía y reduciendo la necesidad de una supervisión constante por parte de sus responsables.

El fin de este proyecto es ofrecer a estos colectivos una herramienta tecnológica pero intuitiva que permita a los usuarios ver sus alarmas diarias y eventos además de proporcionarles un servicio de atención en caso de que se detecte una caída o de que el usuario reporte una incidencia de forma manual. Este sistema supone una mejora frente a los servicios disponibles en la actualidad, como por ejemplo la Teleasistencia de la Junta de Andalucía [1], que solo ofrece la posibilidad de reportar incidencias de manera manual por parte del usuario.

Desde un punto de vista técnico, el desarrollo de este proyecto ha supuesto una gran oportunidad para aplicar algunos de los conocimientos adquiridos durante el grado, así como para aprender y aplicar nuevas tecnologías. En el desarrollo del proyecto se han integrado diferentes tecnologías como servicios REST, bases de datos y aplicaciones móviles.

Este proyecto no solo busca aportar valor académico y tecnológico, sino también abordar un reto social tan importante como la inclusión digital, contribuyendo a la integración del uso de la tecnología en el día a día de las personas que más lo necesitan. En un mundo cada vez más digitalizado, la inclusión de todas las personas debe ser una prioridad.

1.2 Objetivos y alcance

En este apartado se detallan los objetivos del proyecto:

- Desarrollar una aplicación para dispositivos móviles Android orientada a los usuarios finales, garantizando un uso accesible que les permita consultar y gestionar sus alarmas y eventos, así como reportar incidencias. Esta aplicación se ha llamado “SensApp”.
- Desarrollar una aplicación para dispositivos móviles Android dirigida a los responsables de los usuarios que les ofrezca la posibilidad de consultar y modificar los datos de su usuario asociado, editar o crear nuevas alarmas y eventos para este, y consultar la información acerca de las incidencias que se han reportado. Esta aplicación se ha llamado “SensApp Control”.

- Desarrollar una aplicación para dispositivos móviles Android enfocada a los administrativos y operadores del servicio. A través de esta aplicación se les ofrece a los administrativos la posibilidad de gestionar todos los datos existentes, así como dar de alta a nuevos usuarios, responsables u operadores, mientras que a los operadores se les ofrece la posibilidad de consultar algunos datos y gestionar incidencias en tiempo real. Esta aplicación se ha llamado "SensApp Plus".
- Implementar un servicio REST que permita gestionar todos los aspectos relacionados con las aplicaciones Android, como la autentificación de usuarios, la creación de nuevos datos o la consulta de estos en las aplicaciones móviles desarrolladas.

1.2.1 Objetivos específicos

Los objetivos específicos del proyecto son los siguientes:

- Creación de las aplicaciones Android, "SensApp" para usuarios, "SensApp Control" para responsables y "SensApp Plus" para administrativos y operadores.
- Implementación de una funcionalidad de notificaciones que permita a los usuarios recibir avisos en los horarios correspondientes a sus alarmas.
- Implementación de un sistema de detección de caídas utilizando el acelerómetro de los dispositivos móviles. Se deben analizar los datos de este sensor para detectar patrones específicos de movimiento que indiquen una posible caída.
- Diseño e implementación de una base de datos local SQLite para el almacenamiento de credenciales, facilitando el inicio de sesión.
- Implementación de un servicio REST para registrar y consultar los datos necesarios para el funcionamiento del sistema.
- Diseño y despliegue de una base de datos PostgreSQL optimizada y alojada en la nube de AWS, que almacene la información del sistema.
- Despliegue del servicio REST en una instancia EC2 de AWS, garantizando una alta disponibilidad.
- Configuración del firewall en la instancia EC2 de AWS para permitir únicamente las conexiones necesarias, garantizando así la accesibilidad al servicio y la máxima seguridad posible.
- Pruebas de funcionalidad, usabilidad y rendimiento del sistema que aseguren la estabilidad y el correcto funcionamiento del mismo.

1.2.2 Funcionalidad ofrecida al usuario final

Las funcionalidades del sistema dependen del rol que tome el usuario, asegurando así que cada rol tenga acceso a las opciones necesarias para un buen funcionamiento de la aplicación. Se detallan las funcionalidades ofrecidas a cada tipo de usuario:

- Usuario: Es el usuario final de la aplicación "SensApp" y tiene acceso a las siguientes funcionalidades:
    - Inicio de sesión accesible: Para garantizar que el uso de esta aplicación resulte sencillo, la aplicación almacena las credenciales del usuario en una base de datos local SQLite, tras un primer inicio de sesión. Con esto se elimina la necesidad de iniciar sesión manualmente cada vez que se use la aplicación, garantizando un acceso más cómodo al usuario final.
    - Visualización de eventos: Se muestra una lista con los eventos para el día actual. Cada evento puede marcarse como "completado" pulsándolo durante unos segundos.
    - Visualización de alarmas: Se permite al usuario visualizar una lista de sus alarmas. Del mismo modo que los eventos, las alarmas pueden ser marcadas "completada" pulsándolas durante unos segundos. Además, cuando se active una alarma, el usuario recibe una notificación sonora como recordatorio.
    - Creación de incidencias automática: Si se detecta una caída se genera de forma automática una incidencia.

    - Creación de incidencias manual: El usuario puede generar una incidencia de manera manual pulsando durante unos segundos un botón específico para esta funcionalidad.

- Responsable: Es el usuario final de la aplicación "SensApp Control", dispone de diferentes funcionalidades para garantizar una experiencia óptima al usuario.
    - Consulta y edición de información: El responsable tiene la opción de consultar y modificar tanto su información como la de su usuario asociado.
    - Gestión de eventos: Permite consultar los eventos del usuario, editarlos y crear nuevos.
    - Gestión de alarmas: Permite consultar las alarmas del usuario, editarlas y crear nuevas.
    - Consulta de incidencias: Se proporciona acceso a una lista con todas las incidencias registradas del usuario y se permite consultar los detalles de las mismas.
- Administrativo: Es uno de los usuarios finales de la aplicación "SensApp Plus", tiene acceso a las herramientas de gestión del sistema.
    - Gestión de usuarios, responsables y operadores: Puede crear y editar datos asociados a usuarios, responsables y operadores.
    - Consulta de incidencias: Puede acceder a una lista con todas las incidencias registradas.
    - Consulta y edición de datos: Se permite consultar y modificar todos los datos necesarios.
- Operador: Es el otro tipo de usuario final de la aplicación "SensApp Plus", el encargado de gestionar las incidencias registradas.
    - Consulta de datos: Se permite acceder a ciertos datos correspondientes a los usuarios y modificarlos.
    - Gestión de incidencias en tiempo real: Dispone de una lista con las incidencias sin resolver, para ofrecer una atención en tiempo real.

1.2.3 Arquitectura del sistema

En la figura 1-1 se representa el esquema completo de la arquitectura del sistema. En este diagrama se puede ver que las tres aplicaciones Android ("SensApp", "SensApp Control" y "SensApp Plus") acceden al servicio REST alojado en una instancia EC2 de la nube de AWS. Este servicio se encarga de gestionar las solicitudes HTTP de las aplicaciones, realizando las correspondientes consultas a la base de datos PostgreSQL, la cual también se encuentra alojada en AWS.

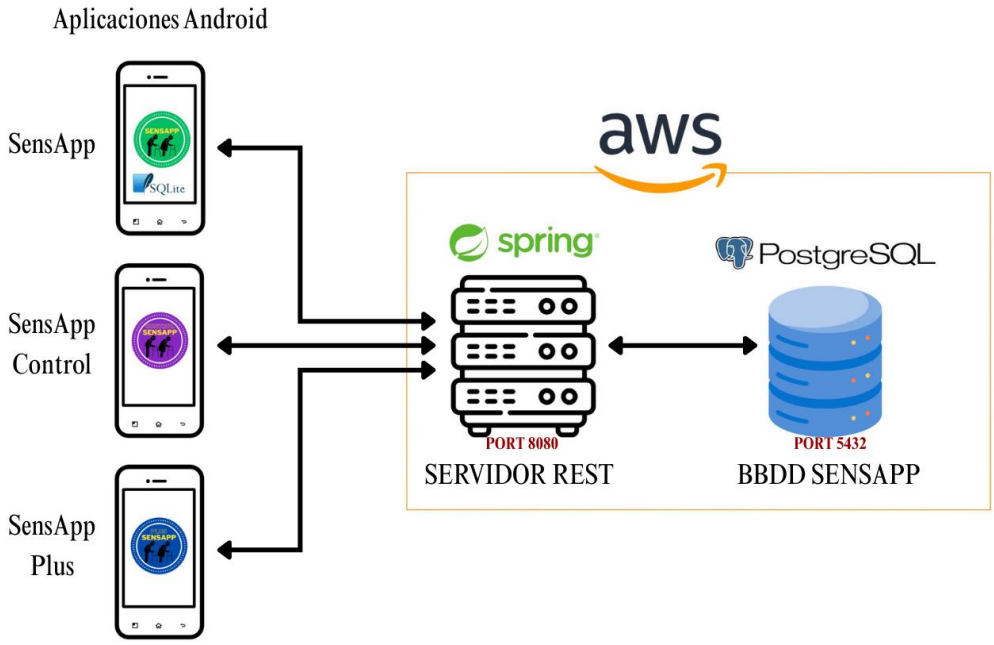

Figura 1-1. Esquema de la arquitectura del Trabajo Fin de Grado

1.3 Estructura de la memoria

A continuación, se describe la organización de los capítulos que conforman el presente documento:

1. Introducción: En este primer capítulo se detallan los objetivos del proyecto, su contexto y motivación. Se incluye además una visión general de la estructura del sistema desarrollado.
2. Recursos utilizados: Se presentan todos los recursos utilizados durante el desarrollo del proyecto, tanto hardware como software, junto con una breve descripción de cada uno de ellos.
3. Servicio REST: Se explican las principales funcionalidades del servicio REST implementado, así como su comunicación con la base de datos PostgreSQL y su despliegue en la nube de AWS.
4. Aplicación Android "SensApp": En este capítulo se detalla la estructura y las principales características de la aplicación destinada a los usuarios, detallando todas las funcionalidades existentes para el usuario final de la misma.
5. Aplicación Android "SensApp Control": Se explica la estructura y las funcionalidades de la aplicación orientada a los responsables de los usuarios, detallando todas las opciones disponibles en esta.
6. Aplicación Android "SensApp Plus": En este capítulo se detalla la estructura y funcionalidad de la aplicación dirigida a los administrativos y operadores. Se detallan todas las opciones para cada tipo de usuario final.
7. Conclusiones y líneas futuras: En el último capítulo de esta memoria se exponen las conclusiones obtenidas tras la realización del proyecto, así como las posibles mejoras o expansiones futuras del mismo.
8. Anexo - Código de las implementaciones: En este anexo se incluye el enlace al repositorio de GitHub donde se encuentra todo el código desarrollado. Además, se detallan los pasos necesarios para desplegar las aplicaciones, el servicio REST y la base de datos PostgreSQL.

2 RECURSOS UTILIZADOS

El saber no es suficiente, debemos aplicarlo. El querer no es suficiente, debemos hacer.

Bruce Lee

Para realizar este trabajo se ha hecho uso de diversos recursos tanto hardware como software, en este capítulo se van a detallar todas las herramientas y tecnologías usadas, explicando sus principales características y su papel en el desarrollo del trabajo.

2.1 Recursos Hardware

2.1.1 Ordenador portátil

Tanto para realizar el diseño y desarrollo del proyecto, como para redactar el presente documento, se ha utilizado el portátil MSI GF63 Thin 10SCSR-205ES, cuyas principales características [2] son las siguientes:

- Procesador Intel Core i7 de décima generación.
- Memoria RAM de 16GB, distribuida en dos módulos de 8GB cada uno.
- Almacenamiento SSD 1TB.
- Gráfica NVIDIA GeForce GTX 1650 Ti Max-Q.

Figura 2-1. Portátil MSI GF63 Thin 10SCSR-205ES

2.1.2 Teléfono móvil Android

El dispositivo móvil utilizado para realizar las pruebas de las aplicaciones Android desarrolladas ha sido el Huawei Mate 10 lite [3], que cuenta con el sistema operativo Android 8.0 Oreo.

Figura 2-2. Huawei Mate 10 lite

2.2 Recursos Software

2.3.1 Android Studio

El entorno de desarrollo integrado (IDE) Android Studio ha sido el elegido para implementar las aplicaciones móviles Android, ya que es el IDE oficial para esta plataforma. Este recurso ha optimizado el desarrollo de código, debido a que proporciona herramientas para desarrolladores de IntelliJ IDEA, estructura el proyecto en módulos que facilitan el diseño, e incorpora Gradle como sistema de compilación flexible y eficiente. [4]

Figura 2-3. Android Studio

2.3.2 Emulador del Google Pixel 4

Android Studio proporciona una amplia variedad de emuladores. Para este proyecto se ha seleccionado el emulador del Google Pixel 4 API 33. Este emulador reproduce las características hardware y software del dispositivo móvil, permitiendo realizar pruebas y depuración de las aplicaciones desarrolladas en el IDE de forma eficiente. Además, al estar integrado en el propio Android Studio, se optimiza el proceso de pruebas y depuración.

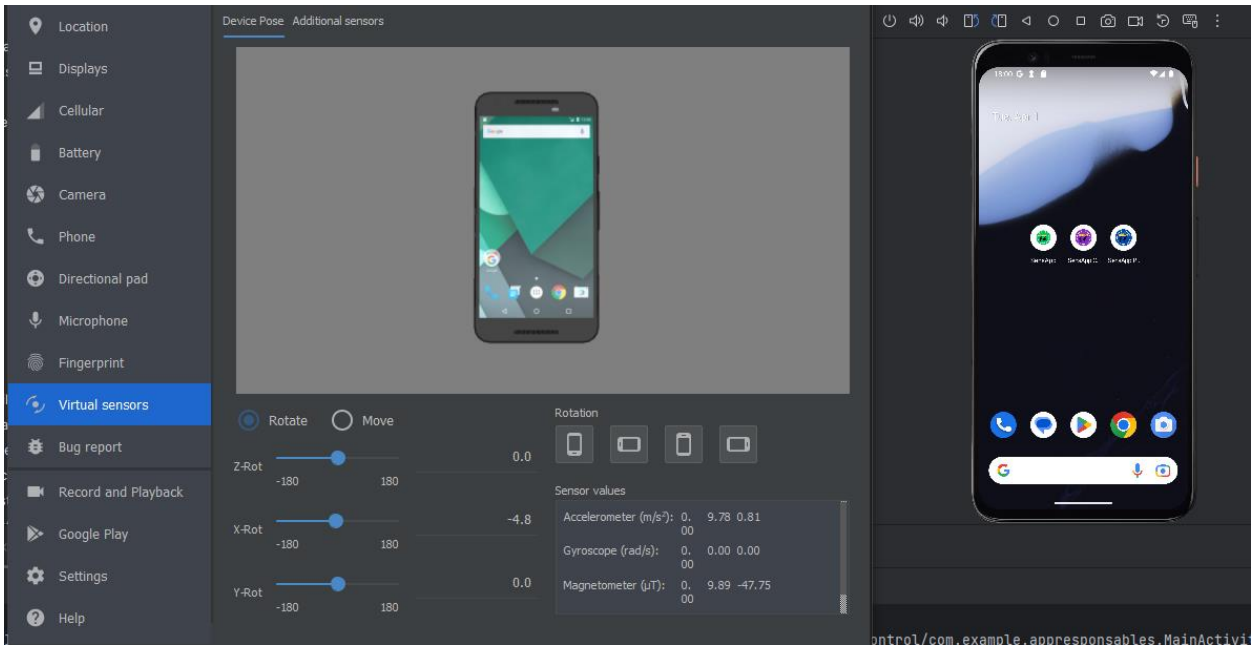

Figura 2-4. Emulador del Google Pixel 4 API 33

2.3.3 Spring

Spring es un framework de desarrollo para aplicaciones basado en Java. Se ha utilizado debido a que ofrece herramientas y bibliotecas que permiten desarrollar servicios web RESTful de manera muy eficiente. [5]

Figura 2-5. Spring

2.3.4 Maven

Maven es una herramienta usada para crear y gestionar proyectos Java. Permite gestionar las dependencias y definir la estructura de un proyecto mediante un archivo de configuración (pom.xml). [6]

Figura 2-6. Maven

2.3.5 RESTClient

RestClient es una herramienta que permite realizar pruebas en un servicio web RESTful desde un navegador. Permite ejecutar peticiones HTTP de tipo GET, POST, PUT y DELETE con la capacidad de incluir variables

en el path, parámetros de consulta u objetos en el cuerpo (body) de las peticiones cuando sea necesario. Se ha utilizado para depurar el servicio RESTful desarrollado. [7]

Figura 2-7. RESTClient

2.3.6 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto. Se ha usado para desarrollar la base de datos donde se van a almacenar todos los datos del proyecto y a la que accede el servicio web RESTful. [8]

Figura 2-8. PostgreSQL

2.3.7 SQLite

SQLite es un sistema de gestión de bases de datos relacional de código abierto y ligero. Está especialmente indicado para teléfonos móviles, ya que prioriza la eficiencia y simplicidad. Por este motivo, se ha utilizado en el proyecto para desarrollar la base de datos local de una las aplicaciones móviles, SensApp. [9]

Figura 2-9. SQLite

2.3.8 Swagger

Swagger es una herramienta de código abierto que permite documentar APIs RESTful de manera interactiva. Genera de manera automática la lista de endpoints disponibles para cada recurso del servicio, e incorpora la capacidad de probar cada uno de ellos desde la documentación HTML generada. [10]

Figura 2-10. Swagger

2.3.9 Visual Studio Code

Visual Studio Code es un editor de código para desarrolladores que cuenta con un gran ecosistema de extensiones, lo que lo hace especialmente útil. Se ha usado en el proyecto para manejar el desarrollo de código. [11]

Figura 2-11. Visual Studio Code

2.3.10 Amazon Web Services

Amazon Web Services (AWS) es una plataforma de servicios en la nube que ofrece una amplia variedad de servicios. En el proyecto se ha usado una instancia de base de datos PostgreSQL y una instancia EC2 para alojar el servicio RESTful. [12]

Figura 2-12. Amazon Web Services

2.3.11 SSH

El protocolo Secure Shell (SSH) permite acceder de manera segura a la instancia EC2 de AWS mediante el uso

de un par de claves pública/privada.

Figura 2-13. SSH

2.3.12 Tmux

Tmux es una herramienta para multiplexar terminales. En este proyecto se utiliza para crear una sesión donde ejecutar el servicio RESTful para poder mantenerlo activo aun cuando se desconecta la conexión SSH. [13]

Figura 2-14. Tmux

3 SERVICIO REST

Lo importante es no dejar de hacerse preguntas.

Albert Einstein

En este capítulo se van a explicar las características del servicio REST desarrollado para el funcionamiento de las tres aplicaciones. Se van a detallar los endpoints disponibles, las clases que componen el servicio y las dependencias de este. También se explicará la estructura de la base de datos PostgreSQL utilizada, detallando cada una de las tablas creadas. Para finalizar el capítulo, se va a aclarar el despliegue en la nube del servicio REST y de la base de datos.

3.1 Servicio REST

3.1.1 Definición de endpoints

Los endpoints permiten a las aplicaciones Android interactuar con el servicio y realizar operaciones CRUD sobre la base de datos. Para cada uno de los endpoints definidos en la API REST, se va a detallar su propósito, el recurso sobre el que opera, el método HTTP empleado y los parámetros de entrada requeridos. Para una mayor claridad, los endpoints se van a dividir según el recurso al que acceden.

- **Usuarios.**

Tabla 3-1. Endpoints del recurso Usuarios

Path	Método HTTP	Parámetros de entrada	Descripción
/usuario/lista	GET	No requiere parámetros	Permite obtener una lista paginada de todos los usuarios existentes.
/usuario	GET	DNI, contraseña (Request Param)	Permite obtener un usuario con el DNI y la contraseña recibidos como parámetros.
/usuario/{dni}	GET	DNI (Path variable)	Obtiene el usuario cuyo DNI se corresponde con el especificado en el path.
/usuario	POST	Usuario (Request Body)	Crea un nuevo usuario a partir del objeto Usuario recibido en el cuerpo de la petición.
/usuario/{dni}	PUT	DNI (Path variable), Usuario (Request Body)	Actualiza el usuario cuyo DNI se corresponde con el especificado en el path, utilizando para ello el objeto Usuario recibido en el cuerpo de

			la petición.
/usuario	DELETE	DNI (Request Param)	Elimina el usuario cuyo DNI se corresponda con el recibido como parámetro.

- **Responsables.**

Tabla 3-2. Endpoints del recurso Responsables

Path	Método HTTP	Parámetros de entrada	Descripción
/responsable/lista	GET	No requiere parámetros	Permite obtener una lista de todos los responsables existentes.
/responsable/lista/{dni_usuario}	GET	DNI_usuario (Path variable)	Permite obtener una lista paginada de todos los responsables asociados a un usuario concreto.
/responsable	GET	usuario, contraseña (Request Param)	Permite obtener un responsable con el usuario y la contraseña recibidos por parámetros.
/responsable/{usuario}	GET	usuario (Path variable)	Permite obtener un responsable con el nombre de usuario especificado en el path.
/responsable	POST	Responsable (Request Body)	Crea un nuevo responsable a partir del objeto Responsable recibido en el cuerpo de la petición.
/responsable/{usuario}	PUT	usuario (Path variable), Responsable (Request Body)	Actualiza el responsable cuyo nombre de usuario se corresponde con el especificado en el path, utilizando para ello el objeto Responsable recibido en el cuerpo de la petición.
/responsable/lista/{dni_usuario}	DELETE	DNI_usuario (Path variable)	Elimina todos los responsables asociados al DNI_usuario especificado en el path.
/responsable/{usuario}	DELETE	usuario (Path variable)	Elimina el responsable cuyo nombre de usuario se corresponda con el especificado en el path.

- **Operadores.**

Tabla 3-3. Endpoints del recurso Operadores

Path	Método HTTP	Parámetros de entrada	Descripción
/operador/lista	GET	No requiere parámetros	Permite obtener una lista paginada de todos los operadores existentes.

/operador	GET	usuario, contraseña (Request Param)	Permite obtener un operador con el nombre de usuario y la contraseña recibidos como parámetros.
/operador/{usuario}	GET	usuario (Path variable)	Obtiene el operador cuyo nombre de usuario se corresponde con el especificado en el path.
/operador	POST	Operador (Request Body)	Crea un nuevo operador a partir del objeto Operador recibido en el cuerpo de la petición.
/operador/{usuario}	PUT	usuario (Path variable), Operador (Request Body)	Actualiza el operador cuyo nombre de usuario se corresponde con el especificado en el path, utilizando para ello el objeto Operador recibido en el cuerpo de la petición.
/operador/{usuario}	DELETE	usuario (Path variable)	Elimina el operador cuyo nombre de usuario se corresponda con el especificado en el path.

- **Administrativos.**

Tabla 3-4. Endpoints del recurso Administrativos

Path	Método HTTP	Parámetros de entrada	Descripción
/administrativo/lista	GET	No requiere parámetros	Permite obtener una lista de todos los administrativos existentes.
/administrativo	GET	usuario, contraseña (Request Param)	Permite obtener un administrativo con el nombre de usuario y la contraseña recibidos como parámetros.
/administrativo/{usuario}	GET	usuario (Path variable)	Obtiene el administrativo cuyo nombre de usuario se corresponde con el especificado en el path.
/administrativo	POST	Administrativo (Request Body)	Crea un nuevo administrativo a partir del objeto Administrativo recibido en el cuerpo de la petición.
/administrativo/{usuario}	PUT	usuario (Path variable), Administrativo (Request Body)	Actualiza el administrativo cuyo nombre de usuario se corresponde con el especificado en el path, utilizando para ello el objeto Administrativo recibido en el cuerpo de la petición.
/administrativo/{usuario}	DELETE	usuario (Path variable)	Elimina el administrativo cuyo nombre de usuario se corresponda con el especificado en el path.

- **Alarmas.**

Tabla 3-5. Endpoints del recurso Alarmas

Path	Método HTTP	Parámetros de entrada	Descripción
/alarma/lista	GET	DNI_usuario (Request Param)	Permite obtener una lista paginada de todas las alarmas asociadas a un DNI_usuario recibido como parámetro.
/alarma	GET	DNI_usuario, nombre (Request Param)	Permite obtener una alarma con DNI_usuario y nombre recibidos como parámetros.
/alarma	POST	Alarma (Request Body)	Crea una nueva alarma a partir del objeto Alarma recibido en el cuerpo de la petición.
/alarma	PUT	DNI_usuario, nombre (Request Param), Alarma (Request Body)	Actualiza la alarma cuyo DNI_usuario y nombre se corresponde con los recibidos como parámetros, utilizando para ello el objeto Alarma recibido en el cuerpo de la petición.
/alarma	DELETE	DNI_usuario, nombre (Request Param)	Elimina la alarma cuyo DNI_usuario y nombre se correspondan con los recibidos como parámetros.

- **Eventos.**

Tabla 3-6. Endpoints del recurso Eventos

Path	Método HTTP	Parámetros de entrada	Descripción
/evento/lista	GET	DNI_usuario (Request Param)	Permite obtener una lista paginada de todos los eventos asociados a un DNI_usuario recibido como parámetro.
/evento	GET	DNI_usuario, nombre (Request Param)	Permite obtener un evento con DNI_usuario y nombre recibidos como parámetros.
/evento	POST	Evento (Request Body)	Crea un nuevo evento a partir del objeto Evento recibido en el cuerpo de la petición.
/evento	PUT	DNI_usuario, nombre (Request Param), Evento (Request Body)	Actualiza el evento cuyo DNI_usuario y nombre se corresponde con los recibidos como parámetros, utilizando para ello el objeto Evento recibido en el cuerpo de la petición.
/evento	DELETE	DNI_usuario, nombre (Request Param)	Elimina el evento cuyo DNI_usuario y nombre se correspondan con los recibidos como parámetros.

- **Incidencias.**

Tabla 3-7. Endpoints del recurso Incidencias

Path	Método HTTP	Parámetros de entrada	Descripción
/incidencia/lista	GET	No requiere parámetros	Permite obtener una lista paginada de todas las incidencias existentes.
/incidencia/ {dni_usuario}	GET	DNI_usuario (Path variable)	Permite obtener una lista paginada de todas las incidencias asociados a un DNI_usuario especificado en el path.
/incidencia/lista/ {operador}	GET	operador (Path variable)	Permite obtener una lista paginada de todas las incidencias cuyo operador asociado sea el especificado en el path.
/incidencia/ noresuelta	GET	No requiere parámetros	Permite obtener una lista paginada de todas las incidencias sin resolver y sin un operador asignado.
/incidencia	GET	id (Request Param)	Permite obtener la incidencia cuyo ID es el recibido por parámetros.
/incidencia	POST	Incidencia (Request Body)	Crea una nueva incidencia a partir del objeto Incidencia recibido en el cuerpo de la petición.
/incidencia	PUT	id, operador (Request Param), Incidencia (Request Body)	Actualiza la incidencia cuyo ID y operador se corresponden con los recibidos por parámetros, utilizando para ello el objeto Incidencia recibido en el cuerpo de la petición.
/incidencia/{id}	PUT	id (Path variable), Incidencia (Request Body)	Actualiza la incidencia cuyo ID se corresponde con el especificado en el path, utilizando para ello el objeto Incidencia recibido en el cuerpo de la petición.
/incidencia/{id}	DELETE	id (Path variable)	Elimina la incidencia cuyo ID se corresponde con el especificado en el path.

3.1.2 Clases

En este apartado se explicarán todas las clases que componen la API REST. Se pueden distinguir tres tipos diferentes de clases: aplicación, entidades y controladores. Se procede a explicar cada una de ellas y su funcionalidad en el sistema.

3.1.2.1 Aplicación

La clase principal de la aplicación, Application.java, es la responsable de iniciar el servicio REST. En esta clase se define el método main, este se encarga de iniciar la aplicación, iniciando el contexto de Spring Boot, y permitir que la aplicación comience a escuchar y a manejar las solicitudes HTTP.

Se ha hecho uso de la notación @SpringBootApplication para que Spring Boot configure automáticamente la ejecución de la aplicación.

3.1.2.2 Entidades

Las entidades son clases que representan los recursos del sistema, se corresponden con tablas de la base de datos que se usan en el servicio, por lo que, en cada clase de entidad, se definen los atributos correspondientes a cada uno de los campos existentes en la base de datos.

Algo común a todas las clases entidad es que cuentan con dos constructores, uno que permite inicializar un objeto del tipo definido con todos sus atributos, y otro sin parámetros, para cuando sea necesario crear una instancia sin inicializar los atributos. Cada uno de los atributos cuenta con su correspondiente getter y setter, estos métodos permiten acceder y modificar sus valores respectivamente.

A continuación, se va a detallar cada una de las clases entidad creadas.

3.1.2.2.1 Usuario.java

La clase Usuario es una entidad que representa a los usuarios del sistema, los que van a hacer uso de la aplicación SensApp. Los atributos de esta clase se corresponden con las columnas de la tabla Usuarios de la base de datos y son los siguientes:

- DNI: Representa el DNI del usuario y actúa como clave primaria, es decir, como identificador único.
- Nombre: Contiene el nombre del usuario.
- Apellidos: Contiene los apellidos del usuario.
- Teléfono: Contiene el número de teléfono del usuario.
- Fecha de nacimiento: Contiene la fecha de nacimiento del usuario, utilizada para calcular su edad en el sistema.
- Domicilio: Contiene el domicilio del usuario.
- Enfermedades previas: Contiene una cadena de texto utilizada para describir las enfermedades previas o datos de interés del usuario.
- Alergias: Contiene una cadena de texto utilizada para describir las alergias del usuario.
- Contraseña: Contiene la contraseña de acceso del usuario a la aplicación.

3.1.2.2.2 Responsable.java

La clase Responsable es una entidad que representa a los responsables del sistema, los usuarios de la aplicación SensApp Control. Los atributos de esta clase se corresponden con las columnas de la tabla Responsables de la base de datos y son los siguientes:

- Usuario: Representa el nombre de usuario del responsable y actúa como clave primaria.
- Nombre: Contiene el nombre del responsable.
- Apellidos: Contiene los apellidos del responsable.
- Teléfono: Contiene el número de teléfono del responsable.
- DNI: Contiene el DNI del responsable.
- DNI_usuario: Contiene el DNI del usuario al que está asociado este responsable. Es una clave externa que referencia al atributo DNI de Usuarios.
- Contraseña: Contiene la contraseña de acceso del responsable a la aplicación.

3.1.2.2.3 Operador.java

La clase Operador es una entidad que representa a los operadores del sistema, uno de los tipos de usuarios de la aplicación SensApp Plus. Los atributos de esta clase se corresponden con las columnas de la tabla Operadores de la base de datos y son los siguientes:

- Usuario: Representa el nombre de usuario del operador y actúa como clave primaria.

- Nombre: Contiene el nombre del operador.
- Apellidos: Contiene los apellidos del operador.
- Activo: Indicador de actividad del operador, si está trabajando toma el valor de TRUE, si no, toma el valor de FALSE.
- Ocupado: Indicador de actividad del operador respecto a las incidencias, si está atendiendo una toma el valor de TRUE, si no, toma el valor de FALSE.
- Contraseña: Contiene la contraseña de acceso del operador a la aplicación.

3.1.2.2.4 Administrativo.java

La clase Administrativo es una entidad que representa a los administrativos del sistema, el otro tipo de usuario de la aplicación SensApp Plus. Los atributos de esta clase se corresponden con las columnas de la tabla Administrativos de la base de datos y son los siguientes:

- Usuario: Representa el nombre de usuario del administrativo y actúa como clave primaria.
- Nombre: Contiene el nombre del administrativo.
- Apellidos: Contiene los apellidos del administrativo.
- Contraseña: Contiene la contraseña de acceso del administrativo a la aplicación.

3.1.2.2.5 Alarma.java

La clase Alarma es una entidad que representa las alarmas del sistema, eventos diarios asociados a un usuario. Los atributos de esta clase se corresponden con las columnas de la tabla Alarmas de la base de datos y son los siguientes:

- DNI_usuario: Representa el DNI del usuario al que está asociada la alarma. Es una clave externa que referencia al atributo DNI de Usuarios y junto con el atributo Nombre, actúa como clave primaria compuesta.
- Nombre: Contiene el nombre de la alarma y junto con el atributo DNI_usuario, actúa como clave primaria compuesta.
- Tipo: Contiene el tipo de alarma, atributo usado para clasificar las alarmas en un conjunto de tipos concreto.
- Hora: Representa la hora a la que está programada la alarma.
- Completado: Representa la fecha en la que se completó por última vez la alarma. Este atributo se actualiza cada vez que la alarma se marca como completada y permite controlar si en el día actual se ha completado o no.

3.1.2.2.6 Evento.java

La clase Evento es una entidad que representa los eventos del sistema, eventos puntuales asociados a un usuario. Los atributos de esta clase se corresponden con las columnas de la tabla Eventos de la base de datos y son los siguientes:

- DNI_usuario: Representa el DNI del usuario al que está asociado el evento. Es una clave externa que referencia al atributo DNI de Usuarios y junto con el atributo Nombre, actúa como clave primaria compuesta.
- Nombre: Contiene el nombre del evento y junto con el atributo DNI_usuario, actúa como clave primaria compuesta.
- Tipo: Contiene el tipo de evento, atributo usado para clasificar los eventos en un conjunto de tipos concreto.
- Dia: Representa la fecha del evento.
- Completado: Representa si se ha completado o no el evento.

3.1.2.2.7 Incidencia.java

La clase Incidencia es una entidad que representa las incidencias registradas en el sistema, asociadas a un usuario. Los atributos de esta clase se corresponden con las columnas de la tabla Incidencias de la base de datos y son los siguientes:

- ID: Contiene un identificador único que actúa como clave primaria.
- DNI_usuario: Representa el DNI del usuario al que está asociada la incidencia. Es una clave externa que referencia al atributo DNI de Usuarios.
- Operador: Contiene el nombre de usuario del operador que atiende esta incidencia. Es una clave externa que referencia al atributo Usuario de Operadores.
- Descripción: Contiene una cadena de texto utilizada para describir la incidencia.
- Procedimiento: Contiene una cadena de texto utilizada para describir el procedimiento llevado a cabo para resolver la incidencia.
- Resuelta: Representa si ha resuelto o no la incidencia.
- Hora: Contiene la hora a la que se registró la incidencia.
- Fecha: Contiene la fecha en la que se registró la incidencia.

3.1.2.3 Controladores

Las clases controladoras de la API REST son responsables de implementar las funcionalidades que ofrece el servicio de manera organizada. Actúan como intermediarias entre la API y las entidades, gestionando la lógica necesaria para llevar a cabo las operaciones CRUD a realizar.

En el servicio REST los controladores se encargan de recibir y procesar las diferentes solicitudes HTTP, realizar las operaciones correspondientes según la solicitud y los datos recibidos, y devolver una respuesta.

Para este servicio se ha definido un controlador para cada una de las entidades existentes, esto permite mantener una estructura clara y organizada.

3.1.2.3.1 UsuarioController.java

La clase UsuarioController se encarga de gestionar todas las solicitudes HTTP recibidas relacionadas con los usuarios del sistema. Esta clase implementa la lógica para realizar las operaciones necesarias asociadas a este recurso. En la tabla 3-8 se pueden observar los métodos de esta clase y los endpoints que definen.

Tabla 3-8. Métodos Usuario Controller

Método	Endpoint	Método HTTP	Descripción
obtenerUsuarios	/usuario/lista	GET	Obtiene una lista paginada de todos los usuarios existentes.
getUsuario	/usuario	GET	Obtiene el usuario cuyo DNI y contraseña se reciben como parámetros del método.
getUsuario	/usuario/{dni}	GET	Obtiene el usuario cuyo DNI se recibe como parámetro del método.
creaUsuario	/usuario	POST	Crea un nuevo usuario, a partir de un objeto Usuario recibido.
updateUsuario	/usuario/{dni}	PUT	Actualiza el usuario cuyo DNI se recibe

			como parámetro del método, con el objeto Usuario recibido.
deleteUsuario	/usuario	DELETE	Elimina el usuario cuyo DNI se recibe como parámetro del método.

3.1.2.3.2 ResponsableController.java

La clase ResponsableController se encarga de gestionar todas las solicitudes HTTP recibidas relacionadas con los responsables del sistema. Esta clase implementa la lógica para realizar las operaciones necesarias asociadas a este recurso. En la tabla 3-9 se pueden observar los métodos de esta clase y los endpoints que definen.

Tabla 3-9. Métodos Responsable Controller

Método	Endpoint	Método HTTP	Descripción
obtenerAllResponsables	/responsable/lista	GET	Obtiene una lista de todos los responsables existentes.
obtenerResponsables	/responsable/lista/ {dni_usuario}	GET	Obtiene una lista de los responsables asociados al usuario cuyo DNI se recibe como parámetro del método.
getResponsable	/responsable	GET	Obtiene el responsable cuyo nombre de usuario y contraseña se reciben como parámetros del método.
getResponsable	/responsable/ {usuario}	GET	Obtiene el responsable cuyo nombre de usuario se recibe como parámetro del método.
creaResponsable	/responsable	POST	Crea un nuevo responsable, a partir de un objeto Responsable recibido.
updateResponsable	/responsable/ {usuario}	PUT	Actualiza el responsable cuyo nombre de usuario se recibe como parámetro del método, con el objeto Responsable recibido.
deleteResponsables	/responsable/lista/ {dni_usuario}	DELETE	Elimina todos los responsables asociados al usuario cuyo DNI se recibe como parámetro del método.
deleteResponsable	/responsable/ {usuario}	DELETE	Elimina el responsable cuyo nombre de usuario se recibe como parámetro del método.

3.1.2.3.3 OperadorController.java

La clase OperadorController se encarga de gestionar todas las solicitudes HTTP recibidas relacionadas con los operadores del sistema. Esta clase implementa la lógica para realizar las operaciones necesarias asociadas a este recurso. En la tabla 3-10 se pueden observar los métodos de esta clase y los endpoints que definen.

Tabla 3-10. Métodos Operador Controller

Método	Endpoint	Método HTTP	Descripción
obtenerOperadores	/operador/lista	GET	Obtiene una lista paginada de todos los operadores existentes.
getOperador	/operador	GET	Obtiene el operador cuyo nombre de usuario y contraseña se reciben como parámetros del método.
getOperador	/operador/{usuario}	GET	Obtiene el operador cuyo nombre de usuario se recibe como parámetro del método.
creaOperador	/operador	POST	Crea un nuevo operador, a partir de un objeto Operador recibido.
updateOperador	/operador/{usuario}	PUT	Actualiza el operador cuyo nombre de usuario se recibe como parámetro del método, con el objeto Operador recibido.
deleteOperador	/operador/{usuario}	DELETE	Elimina el operador cuyo nombre de usuario se recibe como parámetro del método.

3.1.2.3.4 AdministrativoController.java

La clase AdministrativoController se encarga de gestionar todas las solicitudes HTTP recibidas relacionadas con los administrativos del sistema. Esta clase implementa la lógica para realizar las operaciones necesarias asociadas a este recurso. En la tabla 3-11 se pueden observar los métodos de esta clase y los endpoints que definen.

Tabla 3-11. Métodos Administrativo Controller

Método	Endpoint	Método HTTP	Descripción
obtenerAdministrativos	/administrativo/lista	GET	Obtiene una lista paginada de todos los administrativos existentes.
getAdministrativo	/administrativo	GET	Obtiene el administrativo cuyo nombre de usuario y contraseña se reciben como parámetros del método.
getAdministrativo	/administrativo/ {usuario}	GET	Obtiene el administrativo cuyo nombre de usuario se recibe como parámetro del método.
creaAdministrativo	/administrativo	POST	Crea un nuevo administrativo, a partir de un objeto Administrativo recibido.

updateAdministrativo	/administrativo/ {usuario}	PUT	Actualiza el administrativo cuyo nombre de usuario se recibe como parámetro del método, con el objeto Administrativo recibido.
deleteAdministrativo	/administrativo/ {usuario}	DELETE	Elimina el administrativo cuyo nombre de usuario se recibe como parámetro del método.

3.1.2.3.5 AlarmaController.java

La clase AlarmaController se encarga de gestionar todas las solicitudes HTTP recibidas relacionadas con las alarmas del sistema. Esta clase implementa la lógica para realizar las operaciones necesarias asociadas a este recurso. En la tabla 3-12 se pueden observar los métodos de esta clase y los endpoints que definen.

Tabla 3-12. Métodos Alarma Controller

Método	Endpoint	Método HTTP	Descripción
obtenerAlarmas	/alarma/lista	GET	Obtiene una lista paginada de todas las alarmas asociadas al usuario cuyo DNI se recibe como parámetro del método.
getAlarma	/alarma	GET	Obtiene la alarma cuyo nombre y DNI_usuario se reciben como parámetros del método.
creaAlarma	/alarma	POST	Crea una nueva alarma, a partir de un objeto Alarma recibido.
updateAlarma	/alarma	PUT	Actualiza la alarma cuyo nombre y DNI_usuario se reciben como parámetros del método, con el objeto Alarma recibido.
deleteAlarma	/alarma	DELETE	Elimina la alarma cuyo nombre y DNI_usuario se reciben como parámetros del método.

3.1.2.3.6 EventoController.java

La clase EventoController se encarga de gestionar todas las solicitudes HTTP recibidas relacionadas con los eventos del sistema. Esta clase implementa la lógica para realizar las operaciones necesarias asociadas a este recurso. En la tabla 3-13 se pueden podemos observar los métodos de esta clase y los endpoints que definen.

Tabla 3-13. Métodos Evento Controller

Método	Endpoint	Método HTTP	Descripción
obtenerEventos	/evento/lista	GET	Obtiene una lista paginada de todos los eventos asociados al usuario cuyo

			DNI se recibe como parámetro del método.
getEvento	/evento	GET	Obtiene el evento cuyo nombre y DNI_usuario se reciben como parámetros del método.
creaEvento	/evento	POST	Crea un nuevo evento, a partir de un objeto Evento recibido.
updateEvento	/evento	PUT	Actualiza el evento cuyo nombre y DNI_usuario se reciben como parámetros del método, con el objeto Evento recibido.
deleteEvento	/evento	DELETE	Elimina el evento cuyo nombre y DNI_usuario se reciben como parámetros del método.

3.1.2.3.7 IncidenciaController.java

La clase IncidenciasController se encarga de gestionar todas las solicitudes HTTP recibidas relacionadas con las incidencias del sistema. Esta clase implementa la lógica para realizar las operaciones necesarias asociadas a este recurso. En la tabla 3-14 se pueden observar los métodos de esta clase y los endpoints que definen.

Tabla 3-14. Métodos Incidencia Controller

Método	Endpoint	Método HTTP	Descripción
obtenerIncidencias	/incidencia/lista	GET	Obtiene una lista paginada de todas las incidencias existentes.
obtenerIncidenciasDni	/incidencia/ {dni_usuario}	GET	Obtiene una lista paginada de las incidencias asociadas al usuario cuyo DNI se recibe como parámetro del método.
obtenerIncidenciasOperador	/incidencia/lista/ {operador}	GET	Obtiene una lista paginada de las incidencias asociadas al operador cuyo nombre de usuario se recibe como parámetro del método.
obtenerIncidenciasNoResueltas	/incidencia/ noresuelta	GET	Obtiene una lista paginada de las incidencias no resueltas.
getIncidencia	/incidencia	GET	Obtiene la incidencia cuyo ID se recibe como parámetro del método.
creaIncidencia	/incidencia	POST	Crea una nueva incidencia, a partir de un objeto Incidencia recibido.

updateIncidencia	/incidencia	PUT	Actualiza la incidencia cuyo ID y operador se reciben como parámetros del método, con el objeto Incidencia recibido.
updateIncidencia	/incidencia/{id}	PUT	Actualiza la incidencia cuyo ID se recibe como parámetro del método, con el objeto Incidencia recibido.
deleteIncidencia	/incidencia/ {id}	DELETE	Elimina la incidencia cuyo ID se recibe como parámetro del método.

3.1.3 Dependencias

El servicio REST desarrollado incluye varias dependencias esenciales para el correcto funcionamiento del mismo. Estas se definen en el fichero de configuración pom.xml, en este caso las principales dependencias añadidas son:

- PostgreSQL: Añadir esta dependencia es crucial, ya que es la que permite establecer una conexión con la base de datos. Es gracias a esta dependencia que el servicio puede realizar operaciones sobre las tablas existentes.
- Swagger: Swagger es una herramienta que permite documentar la API de manera interactiva. Gracias a esta dependencia se genera automáticamente la documentación HTML, que incluye la lista de endpoints disponibles e incluso incorpora la capacidad de probarlos.

Manejar las dependencias en este fichero de configuración facilita la misma ya que permite gestionarlas de manera única y centralizada.

3.1.3.1 Documentación Swagger

Swagger ha sido la herramienta elegida para generar la documentación, ya que permite crear una interfaz visual e interactiva. Para que Swagger pueda interpretar correctamente los controladores creados y sus métodos, y generar la documentación adecuada, se hace uso de anotaciones en el código de las clases controladoras, algunas de estas anotaciones son:

- @RestController: Utilizada para definir que la clase es un controlador REST.
- @ApiOperation: Utilizada para describir un método específico, añadiendo una descripción del funcionamiento del mismo.
- @PostMapping: Utilizada para mapear solicitudes HTTP de tipo POST, define los endpoints encargados de crear nuevos recursos.
- @GetMapping: Utilizada para mapear solicitudes HTTP de tipo GET, define los endpoints encargados de obtener recursos.
- @PutMapping: Utilizada para mapear solicitudes HTTP de tipo PUT, define los endpoints encargados de actualizar recursos.
- @DeleteMapping: Utilizada para mapear solicitudes HTTP de tipo DELETE, define los endpoints encargados de eliminar recursos.
- @RequestBody: Utilizada para indicar que el valor de un parámetro debe ser obtenido del cuerpo de la solicitud.
- @RequestParam: Utilizada para indicar que el valor de un parámetro debe ser obtenido de los

parámetros de la URL de la solicitud.

- @PathVariable: Utilizada para indicar que el valor de un parámetro debe ser obtenido de la URL de la solicitud.

En las siguientes figuras se muestran imágenes de la documentación generada por Swagger, en la primera de ellas se muestra una vista general de todos los controladores y en la segunda se muestran los endpoints de uno de ellos, Administrativo Controller.

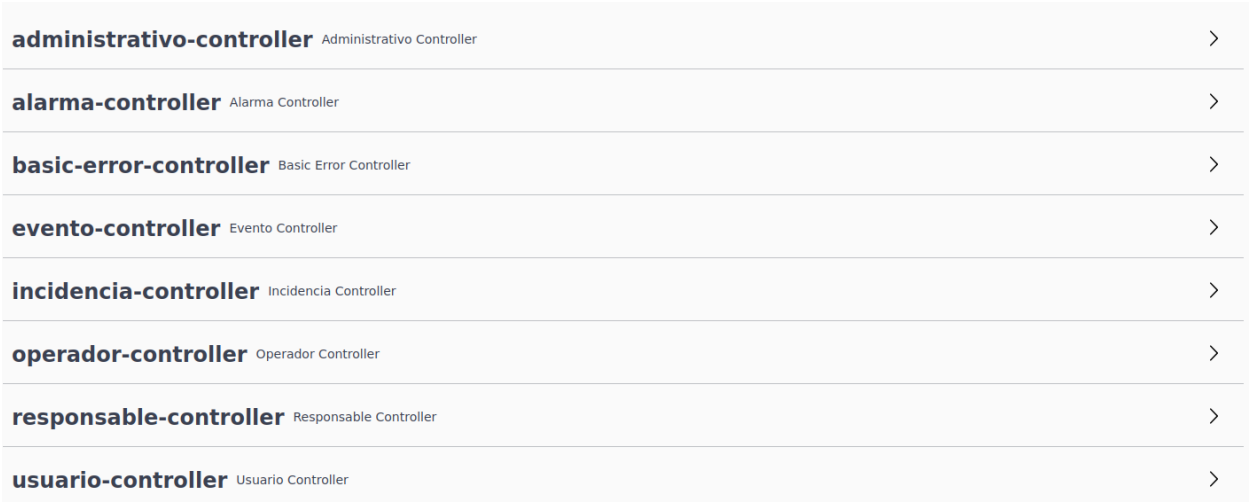

Figura 3-1. Vista general de Swagger

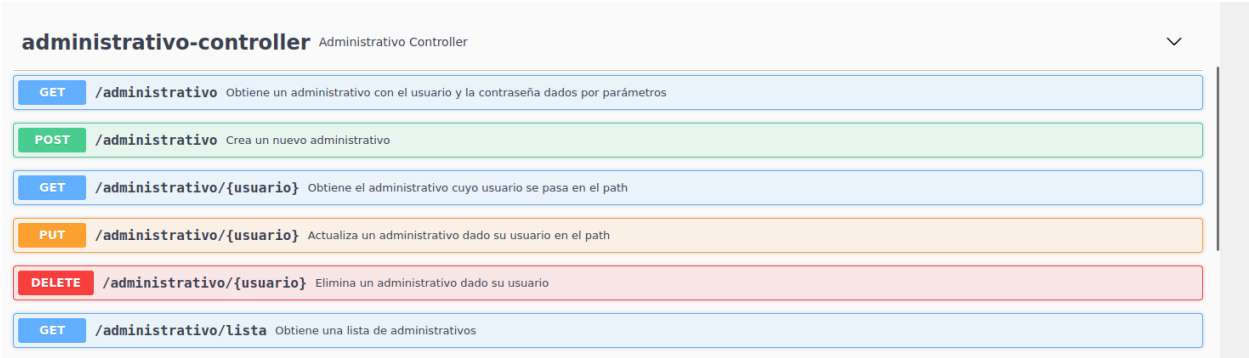

Figura 3-2. Endpoints de Administrativo Controller

Como se adelantaba antes, desde esta misma interfaz es posible probar los endpoints definidos, se muestra un ejemplo de ello en la figura 3-3.

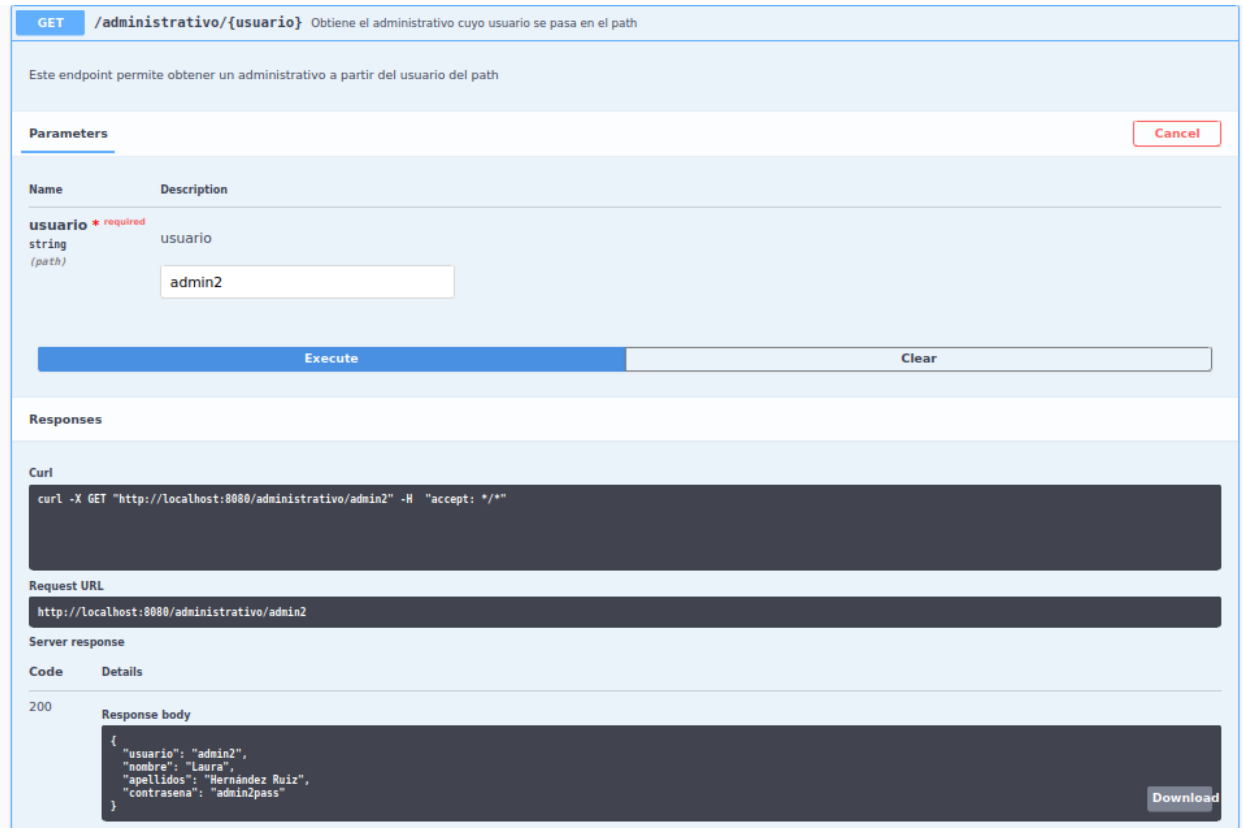

Figura 3-3. Ejemplo de uso del endpoint /administrativo/{usuario}

3.2 Base de datos

La base de datos creada para este proyecto está compuesta por siete tablas, como se puede ver en la figura 3-4, que permiten almacenar toda la información necesaria sobre las entidades de la aplicación, existiendo una correspondencia entre estas y las tablas. Las tablas existentes están relacionadas entre sí mediante el uso de claves externas, para asegurar que exista integridad y coherencia en los datos almacenados.

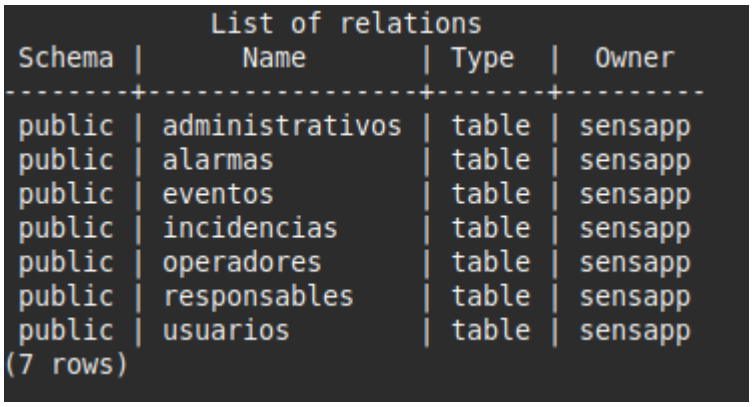

Figura 3-4. Tablas de la BBDD

En el siguiente esquema se puede observar la estructura de las tablas y sus relaciones.

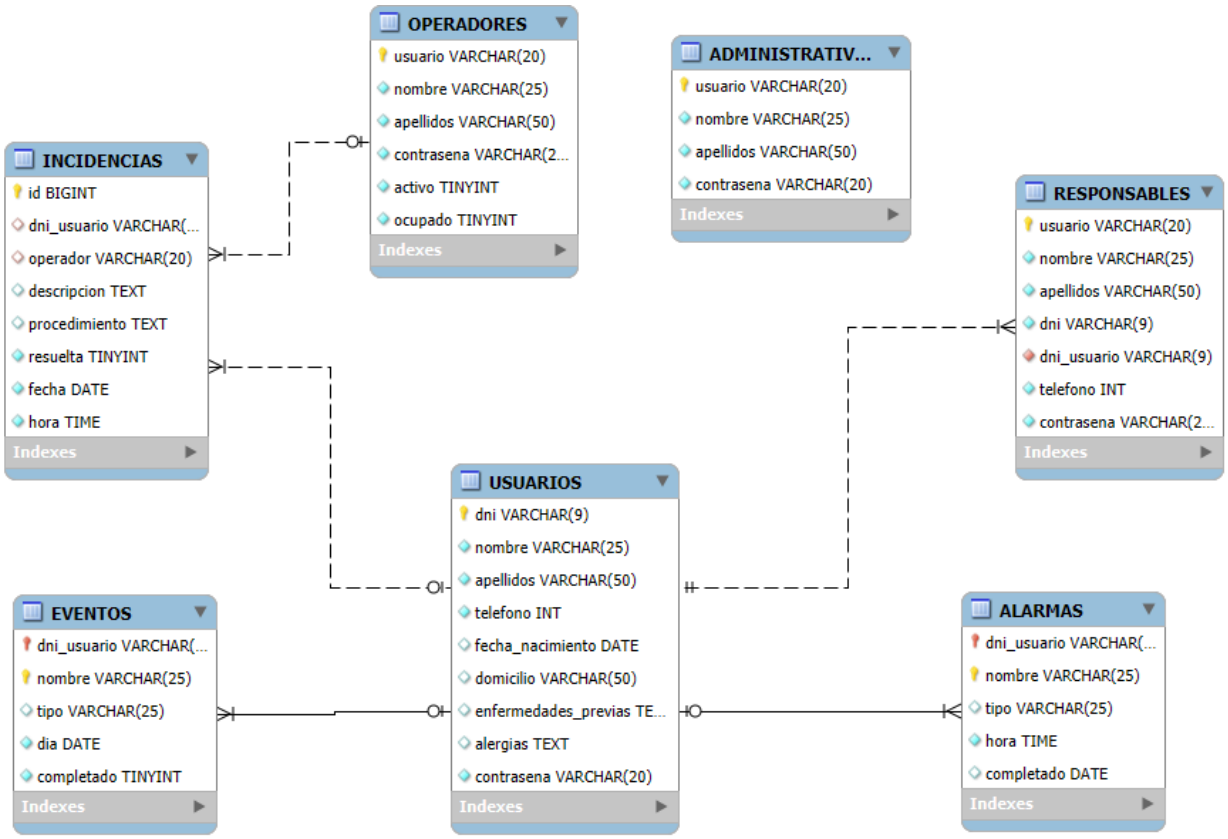

Figura 3-5. Esquema de la BBDD

3.2.1 Usuarios

La tabla Usuarios almacena la información personal de los usuarios de la aplicación SensApp. Cada entrada en esta tabla cuenta de un identificador único, que se corresponde con la columna DNI.

En la figura 3-6 se puede apreciar su estructura.

```
                                Table "public.usuarios"
       Column        |         Type          | Collation | Nullable | Default
---------------------+-----------------------+-----------+----------+---------
 dni                 | character varying(9)  |           | not null |
 nombre              | character varying(25) |           | not null |
 apellidos           | character varying(50) |           | not null |
 telefono            | integer               |           | not null |
 fecha_nacimiento    | date                  |           |          |
 domicilio           | character varying(50) |           |          |
 enfermedades_previas | text                  |           |          |
 alergias            | text                  |           |          |
 contrasena          | character varying(20) |           | not null |
Indexes:
    "usuarios_pkey" PRIMARY KEY, btree (dni)
Referenced by:
    TABLE "alarmas" CONSTRAINT "alarmas_dni_usuario_fkey" FOREIGN KEY (dni_usuario)
REFERENCES usuarios(dni) ON UPDATE RESTRICT ON DELETE RESTRICT
    TABLE "eventos" CONSTRAINT "eventos_dni_usuario_fkey" FOREIGN KEY (dni_usuario)
REFERENCES usuarios(dni) ON UPDATE RESTRICT ON DELETE RESTRICT
    TABLE "incidencias" CONSTRAINT "incidencias_dni_usuario_fkey" FOREIGN KEY (dni_u
suario) REFERENCES usuarios(dni) ON UPDATE RESTRICT ON DELETE RESTRICT
    TABLE "responsables" CONSTRAINT "responsables_dni_usuario_fkey" FOREIGN KEY (dni
_usuario) REFERENCES usuarios(dni) ON UPDATE RESTRICT ON DELETE RESTRICT
```

Figura 3-6. Tabla Usuarios

3.2.2 Responsables

La tabla Responsables almacena la información personal de los usuarios de la aplicación SensApp Control. Cada entrada en esta tabla cuenta de un identificador único, que se corresponde con la columna usuario. Además, la columna DNI_usuario es una clave externa que referencia a la columna DNI de la tabla Usuarios.

En la siguiente figura se puede apreciar su estructura.

```
                Table "public.responsables"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 usuario     | character varying(20) |           | not null |
 nombre      | character varying(25) |           | not null |
 apellidos   | character varying(50) |           | not null |
 dni         | character varying(9)  |           | not null |
 dni_usuario | character varying(9)  |           | not null |
 telefono    | integer               |           | not null |
 contrasena  | character varying(20) |           | not null |
Indexes:
    "responsables_pkey" PRIMARY KEY, btree (usuario)
Foreign-key constraints:
    "responsables_dni_usuario_fkey" FOREIGN KEY (dni_usuario) REFERENCES usuarios(dn
i) ON UPDATE RESTRICT ON DELETE RESTRICT
```

Figura 3-7. Tabla Responsables

3.2.3 Operadores

La tabla Operadores almacena la información personal de uno de los tipos de usuarios de la aplicación SensApp Plus. Cada entrada en esta tabla cuenta de un identificador único, que se corresponde con la columna usuario.

En la siguiente figura se puede apreciar su estructura.

```
                    Table "public.operadores"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 usuario    | character varying(20) |           | not null |
 nombre     | character varying(25) |           | not null |
 apellidos  | character varying(50) |           | not null |
 contrasena | character varying(20) |           | not null |
 activo     | boolean               |           | not null |
 ocupado    | boolean               |           | not null |
Indexes:
    "operadores_pkey" PRIMARY KEY, btree (usuario)
Referenced by:
    TABLE "incidencias" CONSTRAINT "incidencias_operador_fkey" FOREIGN KEY (operador)
 REFERENCES operadores(usuario) ON UPDATE RESTRICT ON DELETE SET NULL
```

Figura 3-8. Tabla Operadores

3.2.4 Administrativos

La tabla Administrativos almacena la información personal del otro tipo de usuarios de la aplicación SensApp Plus. Cada entrada en esta tabla cuenta de un identificador único, que se corresponde con la columna usuario.

En la siguiente figura se puede apreciar su estructura.

```
                  Table "public.administrativos"
   Column   |         Type          | Collation | Nullable | Default
------------+-----------------------+-----------+----------+---------
 usuario    | character varying(20) |           | not null |
 nombre     | character varying(25) |           | not null |
 apellidos  | character varying(50) |           | not null |
 contrasena | character varying(20) |           | not null |
Indexes:
    "administrativos_pkey" PRIMARY KEY, btree (usuario)
```

Figura 3-9. Tabla Administrativos

3.2.5 Alarmas

La tabla Alarma almacena la información de las alarmas registradas. Cada entrada en esta tabla cuenta de un identificador único compuesto, que se corresponde con la columna DNI_usuario y nombre.

En la siguiente figura se puede apreciar su estructura.

```
                    Table "public.alarmas"
   Column    |          Type          | Collation | Nullable | Default
-------------+------------------------+-----------+----------+---------
 dni_usuario | character varying(9)   |           | not null |
 nombre      | character varying(25)  |           | not null |
 tipo        | character varying(25)  |           |          |
 hora        | time without time zone |           | not null |
 completado  | date                   |           |          |
Indexes:
    "alarmas_pkey" PRIMARY KEY, btree (dni_usuario, nombre)
Foreign-key constraints:
    "alarmas_dni_usuario_fkey" FOREIGN KEY (dni_usuario) REFERENCES usuarios(dni) ON
UPDATE RESTRICT ON DELETE RESTRICT
```

Figura 3-10. Tabla Alarmas

3.2.6 Eventos

La tabla Evento almacena la información de los eventos registrados. Cada entrada en esta tabla cuenta de un identificador único compuesto, que se corresponde con la columna DNI_usuario y nombre.

En la siguiente figura se puede apreciar su estructura.

```
                    Table "public.eventos"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 dni_usuario | character varying(9)  |           | not null |
 nombre      | character varying(25) |           | not null |
 tipo        | character varying(25) |           |          |
 dia         | date                  |           | not null |
 completado  | boolean               |           | not null |
Indexes:
    "eventos_pkey" PRIMARY KEY, btree (dni_usuario, nombre)
Foreign-key constraints:
    "eventos_dni_usuario_fkey" FOREIGN KEY (dni_usuario) REFERENCES usuarios(dni) ON
UPDATE RESTRICT ON DELETE RESTRICT
```

Figura 3-11. Tabla Eventos

3.2.7 Incidencias

La tabla Incidencias almacena la información de las incidencias registradas. Cada entrada en esta tabla cuenta de un identificador único, que se corresponde con la columna ID. Además, la columna DNI_usuario es una clave externa que referencia a la columna DNI de la tabla Usuarios y la columna operador es una clave externa que referencia a la columna usuario de la tabla Operadores.

En la siguiente figura se puede apreciar su estructura.

```
                                  Table "public.incidencias"
   Column     |          Type          | Collation | Nullable |                Default
--------------+------------------------+-----------+----------+---------------------------------------
 id           | integer                |           | not null | nextval('incidencias_id_seq'::regclass)
 dni_usuario  | character varying(9)   |           |          |
 operador     | character varying(20)  |           |          |
 descripcion  | text                   |           |          |
 procedimiento | text                  |           |          |
 resuelta     | boolean                |           | not null |
 fecha        | date                   |           | not null |
 hora         | time without time zone |           | not null |
Indexes:
    "incidencias_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "incidencias_dni_usuario_fkey" FOREIGN KEY (dni_usuario) REFERENCES usuarios(dni) ON UPDATE RESTRICT ON
DELETE RESTRICT
    "incidencias_operador_fkey" FOREIGN KEY (operador) REFERENCES operadores(usuario) ON UPDATE RESTRICT ON
DELETE SET NULL
```

Figura 3-12. Tabla Incidencias

3.3 Despliegue en la nube

Para dar por finalizado este capítulo, se va a explicar cómo se ha desplegado el servicio y la base de datos en la nube. Este despliegue permite tener accesibles las aplicaciones desarrolladas desde cualquier lugar, garantizando una alta disponibilidad a los usuarios de las mismas.

3.3.1 Servicio REST

Para el Servicio REST se ha usado una instancia EC2 de AWS. Esta instancia va a actuar como servidor, ejecutando la aplicación desarrollada. Se muestra la instancia en ejecución en la figura 3-13.

Figura 3-13. Instancia EC2 en ejecución

Para conseguir que se ejecute la aplicación, una vez que la instancia está en ejecución en AWS, transferimos los ficheros necesarios haciendo uso de SCP y de un par de claves pública/privada, como podemos ver en la figura 3-14.

```
dit@localhost:~$ scp -i /home/dit/TFG/tfgBD/sensapp.pem -r /home/dit/TFG/tfgBD/tfgaws/ ec2-user@54.83.156.30:/home/ec2-user/TFG
Evento.java                                   100% 1569     6.6KB/s   00:00
Responsable.java                              100% 2177     9.4KB/s   00:00
IncidenciaController.java                     100%   21KB 113.0KB/s   00:00
OperadorController.java                       100%   14KB  74.4KB/s   00:00
Application.java                              100%  568     6.0KB/s   00:00
UsuarioController.java                        100%   14KB  76.6KB/s   00:00
Incidencia.java                               100% 2442    10.2KB/s   00:00
AdministrativoController.java                 100%   14KB  75.5KB/s   00:00
Operador.java                                 100% 1805     7.9KB/s   00:00
Alarma.java                                   100% 1589     6.8KB/s   00:00
AlarmaController.java                         100%   12KB  67.5KB/s   00:00
EventoController.java                         100%   12KB  65.5KB/s   00:00
Usuario.java                                  100% 2853    12.1KB/s   00:00
ResponsableController.java                    100%   18KB 101.4KB/s   00:00
Administrativo.java                           100% 1382    14.3KB/s   00:00
pom.xml                                       100% 1778     7.5KB/s   00:00
```

Figura 3-14. Transferencia de ficheros a la instancia EC2

Cuando los ficheros estén disponibles en la instancia EC2, podremos conectarnos a esta haciendo uso de SSH y del par de claves pública/privada de nuevo. Una vez hayamos accedido al EC2, ejecutamos la aplicación en una sesión tmux para poder mantenerlo activo aun cuando se desconecta la conexión SSH. Todos estos pasos se pueden ver en las figuras 3-15 y 3-16.

```
dit@localhost:~/TFG/tfgBD$ ssh -i sensapp.pem ec2-user@54.83.156.30

A newer release of "Amazon Linux" is available.
  Version 2023.6.20250317:
  Version 2023.7.20250331:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Apr  5 13:39:29 2025 from 37.132.132.11
[ec2-user@ip-172-31-86-98 ~]$ cd tfg/tfgBD/tfgaws/
[ec2-user@ip-172-31-86-98 tfgaws]$ tmux new-session -s sensapp
[detached (from session sensapp)]
[ec2-user@ip-172-31-86-98 tfgaws]$
```

Figura 3-15. Conexión a la instancia EC2 mediante SSH

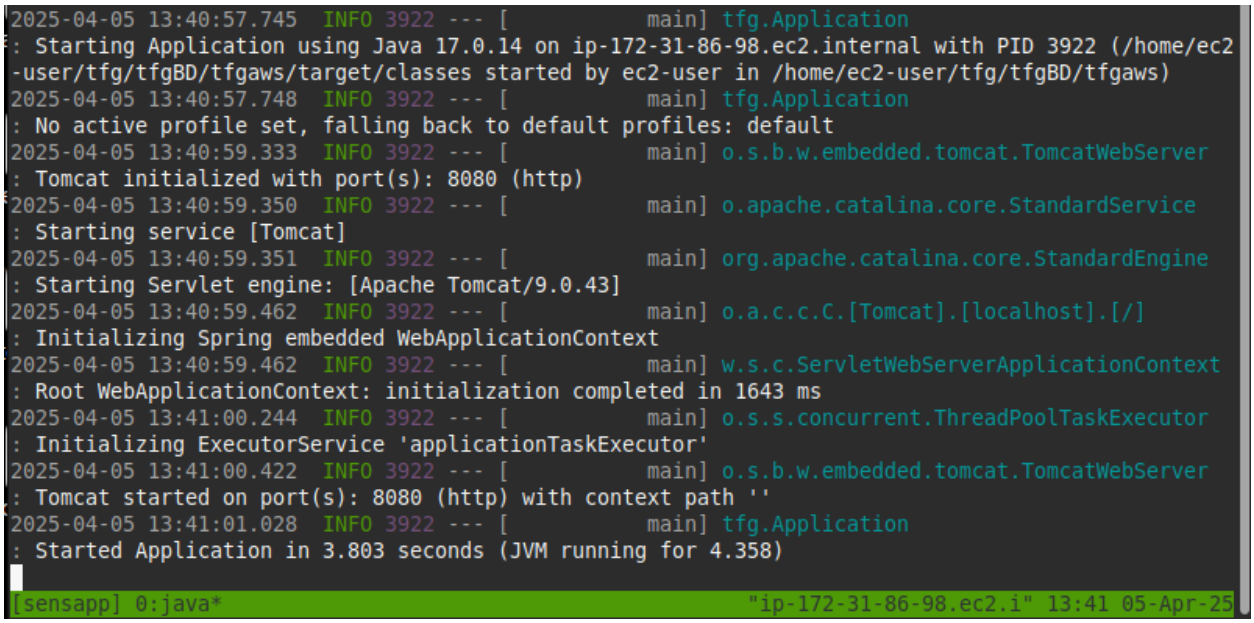

Figura 3-16. Aplicación activa en sesión de tmux

3.3.2 Base de datos

Para la base de datos se ha desplegado una instancia de Amazon RDS con PostgreSQL como sistema gestor, definiendo las credenciales necesarias para permitir la conexión. Se muestra la instancia disponible en la figura 3-17.

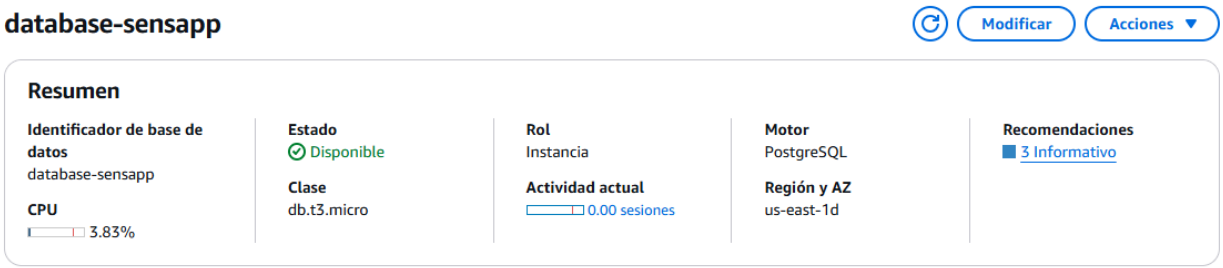

Figura 3-17. Instancia de RDS disponible

Cuando la instancia está disponible, se accede a ella mediante psql y se ejecutan los comandos \i sentencias.sql y \i secuenciasInsert.sql.

El fichero sentencias.sql, no es más que un conjunto de sentencias SQL para crear las tablas mostradas en el apartado anterior, mientras que el fichero sentenciasInsert.sql es otro conjunto de sentencias para insertar unos datos iniciales en las tablas creadas.

```
dit@localhost:~/TFG/tfgBD/tfgBD$ psql --host=database-sensapp.cz4i4mewyc6r.us-east-1.rds.amazonaws.com --port=5432
--username=sensapp --password --dbname=databasesensapp
Password:
psql (14.17 (Ubuntu 14.17-0ubuntu0.22.04.1), server 17.2)
WARNING: psql major version 14, server major version 17.
         Some psql features might not work.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

databasesensapp=> \i sentencias.sql
psql:sentencias.sql:5: NOTICE:  table "usuarios" does not exist, skipping
DROP TABLE
psql:sentencias.sql:6: NOTICE:  table "responsables" does not exist, skipping
DROP TABLE
psql:sentencias.sql:7: NOTICE:  table "alarmas" does not exist, skipping
DROP TABLE
psql:sentencias.sql:8: NOTICE:  table "eventos" does not exist, skipping
DROP TABLE
psql:sentencias.sql:9: NOTICE:  table "administrativos" does not exist, skipping
DROP TABLE
psql:sentencias.sql:10: NOTICE:  table "operadores" does not exist, skipping
DROP TABLE
psql:sentencias.sql:11: NOTICE:  table "incidencias" does not exist, skipping
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
databasesensapp=> \i sentenciasInsert.sql
INSERT 0 3
INSERT 0 3
INSERT 0 3
INSERT 0 3
INSERT 0 3
INSERT 0 3
INSERT 0 3
```

Figura 3-18. Acceso a la BBDD y creación de tablas

3.3.3 Seguridad

Debido a que las instancias creadas cuentan con IPs públicas se ha querido garantizar la seguridad del despliegue de esta aplicación, para ello se han configurado los grupos de seguridad, tanto de la instancia EC2 como de la RDS.

- **Instancia EC2.**

Para el caso de la instancia EC2 se han configurado dos reglas de entrada, una para permitir la conexión SSH para acceder a la misma, y otra para permitir las peticiones HTTP al puerto 8080 del servidor.

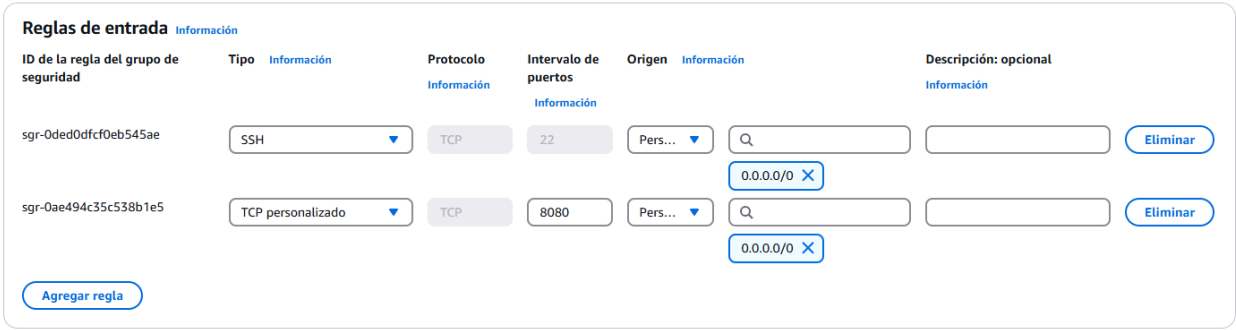

Figura 3-19. Reglas de entrada de la instancia EC2

- **Instancia RDS.**

Por otro lado, para la instancia RDS de la base de datos, ha sido necesario configurar una regla de entrada que permita la conexión a la base de datos en el puerto 5432.

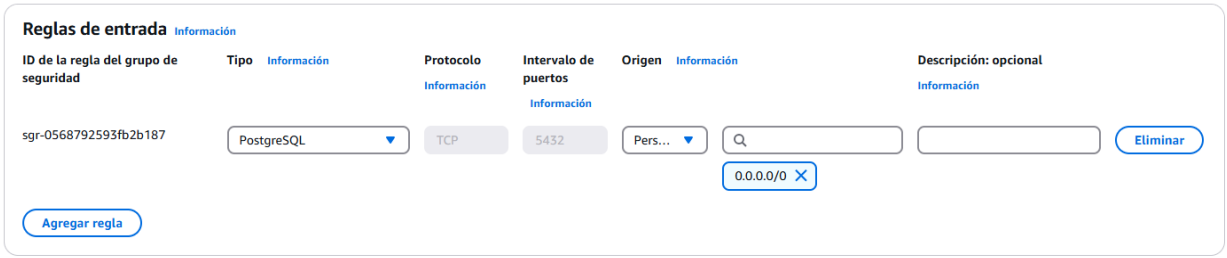

Figura 3-20. Reglas de entrada de la instancia RDS

Con esta configuración nos aseguramos de que solo conexiones autorizadas podrán acceder al servicio y a la base de datos, garantizando así su protección.

4 APLICACIÓN SENSAPP

The only limit to out realization of tomorrow is our doubts of today.

Franklin D. Roosevelt

La aplicación que se va a describir en este capítulo es la parte fundamental del sistema, ya que está dirigida a los usuarios que se encuentran en situación de dependencia o en alguna otra similar. Esta aplicación les permite poder ver una lista de sus alarmas y sus eventos diarios, además de reportar incidencias de forma manual o automática (cuando se detecte una caída). Se ha diseñado para ofrecer una interfaz de usuario accesible y fácil de usar.

En este capítulo se van a explicar las funcionalidades que integra, la estructura de ficheros de la aplicación y la configuración necesaria para garantizar un buen funcionamiento.

Para facilitar una visión general de la aplicación, se representa en la figura 4-1 el diagrama de casos de uso de la aplicación.

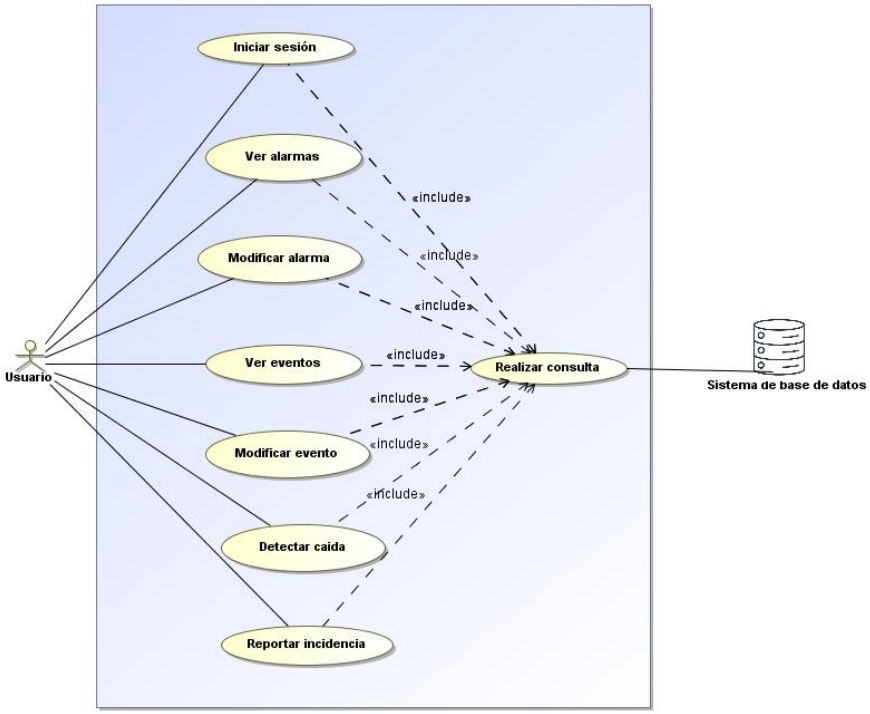

Figura 4-1. Casos de uso aplicación SensApp[1]

[1] En el diagrama representado se han omitido las relaciones de extensión, por simplificación. Estas quedan reflejadas en las tablas de cada uno de los casos de usos.

4.1 Funcionalidades

En este apartado se van a explicar todas las funcionalidades ofrecidas por la aplicación, detallando para cada una sus casos de uso. Para la mayoría de estos, el actor principal es el usuario.

4.1.1 Inicio de sesión

Para poder acceder a las funciones que ofrece la aplicación el usuario debe estar registrado. El proceso de inicio de sesión garantiza seguridad para el usuario, sin embargo, introducir las credenciales cada vez que se accede a la aplicación puede ser una tarea incómoda e incluso puede representar una dificultad para los usuarios de esta aplicación. Es por este motivo que se ha creado una base de datos SQLite local, donde se almacenan las credenciales tras un primer intento de inicio de sesión exitoso. Esto permite a los usuarios un acceso más cómodo, ya que no será necesario introducir de nuevo la contraseña hasta que se desinstale y se vuelva a instalar la aplicación.

La interfaz de usuario se representa en la figura 4-2.

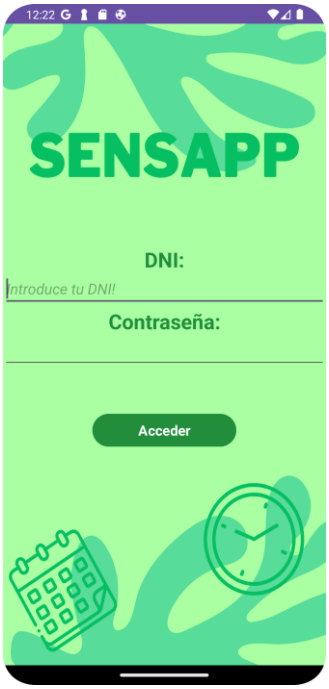

Figura 4-2. Pantalla de inicio de sesión

4.1.1.1 CDU – Iniciar sesión

En este caso de uso se pretende dar acceso a un usuario registrado al resto de funcionalidades de la aplicación.

Tabla 4-1. CDU-1.01 – Iniciar sesión

CDU-1.01	**Iniciar sesión**	
Descripción	El usuario introduce sus credenciales para acceder a la aplicación. Si estas son correctas, el usuario podrá acceder a la pantalla principal de la aplicación, si no, se mostrará un mensaje de error.	
Precondición	El usuario debe estar registrado en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario introduce un DNI y una contraseña en la pantalla de inicio

		de sesión y pulsa el botón “Iniciar sesión”.
	2	Se llama al CDU – 1.08 para realizar una petición al servicio REST, para comprobar si existe el usuario mediante una consulta a la BBDD.
	3	Si los datos introducidos se corresponden con un usuario registrado en el sistema, se da acceso al usuario a la pantalla principal de la aplicación.
	3.1	Tras un inicio de sesión exitoso, si no había ningún usuario registrado en la BBDD local SQLite, se almacenan en esta los datos introducidos, para facilitar futuros inicios de sesión.
Postcondición	El usuario puede acceder a todas las funcionalidades de la aplicación.	
Excepciones	Paso	Acción
	1	Si hay algún usuario almacenado en la BBDD local SQLite se pasa directamente al paso 2.
	3	Si los datos no son correctos, se muestra un mensaje de error al usuario.

4.1.2 Pantalla principal

Cuando el usuario inicia sesión correctamente en la aplicación accede a la pantalla principal. Desde esta pantalla, el usuario puede ver su lista de eventos para el día de hoy, su lista de alarmas diarias, y reportar incidencias manual o automáticamente.

El motivo de usar una única pantalla es facilitar el uso de la aplicación, logrando que el usuario pueda acceder a todas las funcionalidades desde una misma interfaz simple y clara.

Figura 4-3. Pantalla principal

4.1.2.1 CDU – Ver alarmas

El objetivo de este caso es permitir al usuario ver su lista de alarmas. El estado de estas es fácilmente reconocible debido a que los colores de los elementos de la lista dependen de este:

- Rojo: Alarmas cuya hora ya ha pasado y no han sido completadas.
- Azul: Alarmas cuya hora aún no ha llegado.
- Verde: Alarmas completadas.

Además, a la hora de cada alarma, el usuario puede ver una notificación en su pantalla (que además es sonora), para facilitar que se acuerde de la alarma. Esto se puede ver en la figura 4-4.

Figura 4-4. Notificación alarma

A continuación se detalla el caso de uso en la tabla 4-2.

Tabla 4-2. CDU-1.02 – Ver alarmas

CDU-1.02	**Ver alarmas**	
Descripción	Permite al usuario ver la lista de sus alarmas, además de recibir una notificación sonora cuando llegue la hora de cada una de ellas.	
Precondición	El usuario debe haber iniciado sesión con éxito (CDU – 1.01).	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accede a la pantalla principal.
	2	Se llama al CDU – 1.08 para realizar una petición al servicio REST, para obtener la lista de las alarmas del usuario.
	3	Todas las alarmas del usuario se muestran en pantalla.
	4	Cuando llega la hora de una alarma, llega una notificación al usuario.

	5	El usuario pulsa “OK” para detener la notificación.	
Postcondición	El usuario ve su lista de alarmas actualizada, la alarma que antes se representaba en color azul, se representa ahora en color rojo.		

4.1.2.2 CDU – Modificar alarma

En este caso de uso se pretende que el usuario cambie el estado de sus alarmas. Cuando el usuario haga un long click sobre una de sus alarmas sin completar se marcarán como completadas y se actualizará la lista de alarmas. Se ha optado por usar un long click en lugar de un click simple para este comportamiento para evitar posibles pulsaciones por accidente.

Este caso de uso se detalla en la tabla 4-3.

Tabla 4-3. CDU-1.03 – Modificar alarma

CDU-1.03	**Modificar alarma**	
Descripción	Permite al usuario modificar el estado de sus alarmas.	
Precondición	El usuario debe haber iniciado sesión con éxito (CDU – 1.01).	
Secuencia normal	**Paso**	**Acción**
	1	El usuario visualiza la lista de sus alarmas diarias (CDU – 1.02).
	2	El usuario realiza un long click sobre una alarma.
	3	Se llama al CDU – 1.08 para realizar una petición al servicio REST, para actualizar la alarma.
	3.1	Se llama al CDU – 1.08 para realizar una petición al servicio REST, para actualizar la lista de alarmas.
	3.2	Se actualiza la lista representada en pantalla.
Postcondición	El usuario ve su lista de alarmas actualizada, la alarma que ha pulsado se muestra ahora como completada, es decir, en color verde.	

4.1.2.3 CDU – Ver eventos

El objetivo de este caso es permitir al usuario ver su lista de eventos para el día actual. El estado de los eventos se representa gráficamente en los colores de estos, para conseguir una identificación rápida y sencilla:

- Rojo: Eventos no completados.
- Verde: Eventos completados.

En la tabla 4-4 se detalla el caso de uso.

Tabla 4-4. CDU-1.04 – Ver eventos

CDU-1.04	**Ver eventos**
Descripción	Permite al usuario ver la lista de los eventos para el día actual.

Precondición	El usuario debe haber iniciado sesión con éxito (CDU – 1.01).
Secuencia normal	**Paso** / **Acción** 1 / El usuario accede a la pantalla principal. 2 / Se llama al CDU – 1.08 para realizar una petición al servicio REST, para obtener la lista de eventos. 3 / Todos los eventos del usuario se muestran en pantalla. 4 / El usuario visualiza una lista de sus eventos, donde puede ver fácilmente el estado de estos.
Postcondición	El usuario ve su lista de eventos.

4.1.2.4 CDU – Modificar evento

Este caso de uso es muy similar al de modificar alarmas. En él se pretende que el usuario cambie el estado de sus eventos. Para esto, se ha implementado una funcionalidad para que cuando el usuario haga long click sobre uno de sus eventos, este se marque como completado. Del mismo modo que en el caso de las alarmas, se ha optado por usar un long click para evitar pulsaciones por accidente y facilitar su uso. Se detalla en la tabla 4-5.

Tabla 4-5. CDU-1.05 – Modificar evento

CDU-1.05	**Modificar evento**
Descripción	Permite al usuario modificar el estado de sus eventos.
Precondición	El usuario debe haber iniciado sesión con éxito (CDU – 1.01).
Secuencia normal	**Paso** / **Acción** 1 / El usuario visualiza la lista de sus eventos (CDU – 1.04). 2 / El usuario realiza un long click sobre un evento. 3 / Se llama al CDU – 1.08 para realizar una petición al servicio REST, para actualizar el evento. 3.1 / Se llama al CDU – 1.08 para realizar una petición al servicio REST, para actualizar la lista de eventos. 3.2 / Se actualiza la lista representada en pantalla.
Postcondición	El usuario ve su lista de eventos actualizada, el evento que ha pulsado se muestra ahora como completado, es decir, en color verde.

4.1.2.5 CDU – Detectar caída

Este caso de uso pretende reportar una incidencia cuando se detecte que el usuario ha sufrido una caída. Para determinar esto se hace uso del acelerómetro, detectando un movimiento brusco seguido de uno normal. Esta detección se realiza midiendo la gravedad en los ejes X, Y y Z, y calculando la aceleración total según la siguiente fórmula:

$$a_{TOTAL} = \sqrt{X^2 + Y^2 + Z^2}$$

Cuando se produce una caída, el acelerómetro detecta una fuerte aceleración con respecto a la de la gravedad de la tierra, que es, aproximadamente 9.8 m/s^2, seguida de una desaceleración. Se ha definido un umbral de referencia de tal forma que, cuando se detecte este patrón de movimiento, aceleración brusca seguida de una desaceleración, y se supere el umbral (establecido en 1.7) se considere que se ha detectado una caída.

Para evitar posibles errores, una vez detectada la incidencia el usuario recibe una notificación sonora que indica que se ha detectado una incidencia, el usuario puede pulsar en "Crear " o en "No crear". Si tras 15 segundos no se ha obtenido respuesta del usuario, se crea la incidencia. La notificación se puede ver en la figura 4-5.

Figura 4-5. Confirmación de incidencia detectada

Este caso de uso se detalla a continuación.

Tabla 4-6. CDU-1.06 – Detectar caída

CDU-1.06	**Detectar caída**	
Descripción	Permite detectar una caída analizando los datos del acelerómetro del dispositivo. Tras detectarla se crea una incidencia.	
Precondición	El usuario debe haber iniciado sesión con éxito (CDU – 1.01) y el dispositivo debe contar con un sensor de tipo acelerómetro.	
Secuencia normal	**Paso**	**Acción**
	1	Se monitorizan los datos del acelerómetro.
	2	Se detecta un patrón de movimiento que puede corresponder a una caída.
	3	Se notifica al usuario de que se ha detectado una caída, mediante un diálogo.
	3.1	El usuario pulsa "Crear" o no se obtiene respuesta tras 15 segundos.

	4	Se llama al CDU – 1.08 para realizar una petición al servicio REST, para registrar la incidencia.
	4.1	El usuario recibe un mensaje por pantalla indicando que se ha registrado una incidencia.
Postcondición	Se ha registrado una incidencia sin resolver en el sistema, quedando pendiente de resolver por un operador.	
Excepciones	Paso	Acción
	3	Si el usuario pulsa "No crear" no se registra ninguna incidencia.

4.1.2.6 CDU - Reportar incidencia

Este caso de uso complementa al anterior, permite crear una incidencia cuando lo desee el usuario, por si le ha ocurrido otro tipo de incidencia que no sea necesariamente una caída. Haciendo un long click en el icono correspondiente en pantalla se crea una incidencia del mismo modo que antes, incluyendo la notificación de confirmación para evitar errores. Se detalla en la tabla 4-7.

Tabla 4-7. CDU-1.07 – Reportar incidencia

CDU-1.07	**Reportar incidencia**	
Descripción	Permite al usuario reportar una incidencia de manera manual.	
Precondición	El usuario debe haber iniciado sesión con éxito (CDU – 1.01).	
Secuencia normal	**Paso**	**Acción**
	1	El usuario realiza un long click en el icono de incidencias.
	2	Se notifica al usuario de que se ha detectado una incidencia, mediante un diálogo.
	2.1	El usuario pulsa "Crear" o no se obtiene respuesta tras 15 segundos.
	3	Se llama al CDU – 1.08 para realizar una petición al servicio REST, para registrar la incidencia.
	3.1	El usuario recibe un mensaje por pantalla indicando que se ha registrado una incidencia.
Postcondición	Se ha registrado una incidencia sin resolver en el sistema, quedando pendiente de resolver por un operador.	
Excepciones	Paso	Acción
	2	Si el usuario pulsa "No crear" no se registra ninguna incidencia.

4.1.3 Petición al Servicio REST y consulta a la BBDD

4.1.3.1 CDU - Realizar consulta

Este caso de uso se incluye en todos los anteriores, y permite realizar una consulta a la base de datos utilizada

por el sistema para consultar, modificar o crear algún dato. Se detalla en la tabla 4-8.

Tabla 4-8. CDU-1.08 – Realizar consulta

<table>
<tr><td>CDU-1.08</td><td colspan="2">Realizar consulta</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite realizar una consulta a la base de datos.</td></tr>
<tr><td>Actor principal</td><td colspan="2">Sistema de base de datos</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table><tr><th>Paso</th><th>Acción</th></tr><tr><td>1</td><td>Se realiza una petición al servicio REST para obtener, modificar o crear algún dato de la BBDD.</td></tr><tr><td>2</td><td>Se realiza la consulta SQL.</td></tr><tr><td>3</td><td>Se devuelve el resultado del método REST invocado.</td></tr></table></td></tr>
<tr><td>Postcondición</td><td colspan="2">Se ha realizado una consulta a la BBDD.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Paso</td><td>Acción</td></tr><tr><td>1</td><td>Si no hay conexión a internet no se podrá realizar la petición correctamente.</td></tr><tr><td>2</td><td>Si no hay acceso a la BBDD no se podrá realizar la consulta.</td></tr></table></td></tr>
</table>

4.2 Estructura de ficheros de la aplicación

La estructura de este proyecto está organizada de manera simple y clara, lo que facilita el mantenimiento de la misma y el uso de reutilización. Los principales ficheros de código y de configuración de la aplicación se encuentran en el directorio src/main/. En esta carpeta podemos encontrar el fichero de configuración AndroidManifest.xml, que contiene la configuración necesaria para el correcto funcionamiento de la aplicación.

También podemos encontrar dos grandes directorios, java/com/example/appusuarios/, donde se almacena todo el código fuente, y res/, donde se almacenan los diferentes recursos. Estos se explican en los siguientes subapartados.

4.2.1 Directorio del código fuente de la aplicación

Como se adelantaba antes, este directorio contiene el código fuente de la aplicación. En sus ficheros se define toda la lógica de las funcionalidades descritas en apartados anteriores y de toda la aplicación. Se incluyen, por tanto, ficheros que gestionan el sistema de alarmas, la comunicación con el servicio REST, la interacción con la base de datos local, la representación de datos en pantalla, etc. En la figura 4-6 se puede observar su estructura.

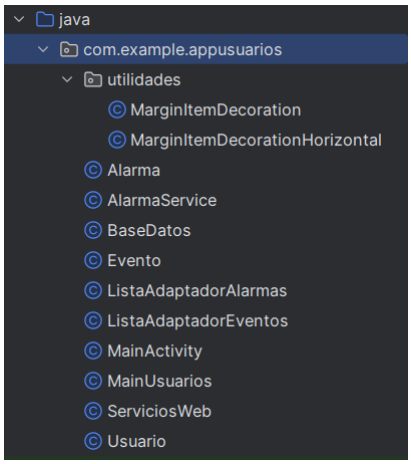

Figura 4-6. Estructura del directorio java/com/example/appusuarios/

Los principales archivos de este directorio son los siguientes:

- Alarma.java, Evento.java y Usuario.java: JavaBeans utilizados para representar las entidades Alarma, Evento y Usuario, respectivamente, en la aplicación. Todas contienen los atributos y métodos getter y setter correspondientes según lo visto en el capítulo 3 de este documento.
- ListaAdaptadorAlarmas.java y ListaAdaptadorEventos.java: Clases encargadas de adaptar la visualización de una lista de alarmas o eventos, respectivamente, para poder ofrecer al usuario una interfaz gráfica más atractiva y simple.
- BaseDatos.java: Clase encargada de gestionar la base de datos local SQLite de la aplicación. Crea la base de datos y define los métodos necesarios para manejar los datos desde las actividades.
- AlarmaService.java: Esta clase es un servicio que se encarga de gestionar las notificaciones que recibe el usuario cuando llega la hora de alguna de sus alarmas.
- ServiciosWeb.java: Clase encargada de gestionar una única instancia de la cola de peticiones al servidor REST mediante el patrón Singleton[2]. Esta clase permite realizar las peticiones de manera centralizada, lo que garantiza que solo existirá una cola de solicitudes.
- MainActivity.java: Actividad encargada de gestionar el inicio de sesión del usuario y de redirigirlo, en caso de éxito, a la pantalla principal.
- MainUsuarios.java: Actividad de la pantalla principal de la aplicación. Se encarga de toda la funcionalidad descrita en este capítulo, una vez que el usuario ha iniciado sesión.
- Utilidades/MarginItemDecoration.java y utilidades/MarginItemDecorationHorizontal.java: Clases encargadas de agregar márgenes personalizados entre los elementos de un RecyclerView[3], mejorando la apariencia de las listas mostradas en pantalla.

4.2.2 Directorio de recursos

Este directorio contiene los recursos de los que hace uso la aplicación, es decir, imágenes, audios, valores de configuración, etc. A su vez, este directorio contiene subdirectorios específicos según el tipo de recurso que se almacena dentro, como se puede ver en la figura 4-7.

[2] Patrón de diseño software que asegura que solo se tenga una única instancia de una determinada clase.
[3] Componente de Android Studio que permite mostrar listas o colecciones de datos de manera eficiente.

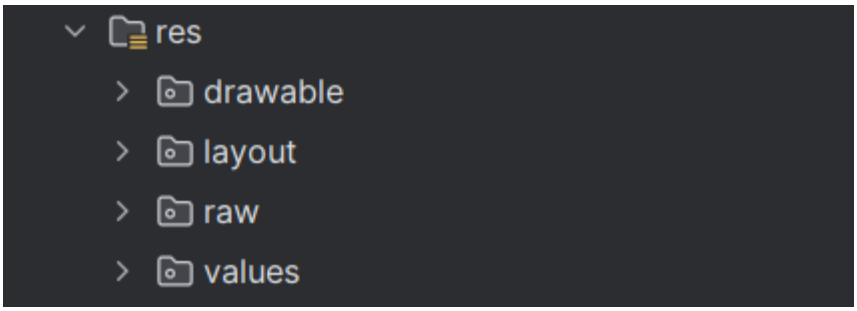

Figura 4-7. Estructura del directorio res/

Cada una de estas carpetas almacena recursos de un tipo concreto:

- Drawable: Contiene los recursos gráficos de los que se hacen uso en el proyecto. Principalmente se almacenan en formato .png o .xml.
- Layout: Contiene todos los ficheros XML que definen las interfaces de usuario, ya sean pantallas de la aplicación o recursos que se usan en las pantallas.
- Raw: Contiene archivos de sonido, para los diálogos sonoros implementados.
- Values: Contiene ficheros XML que definen valores globales, como cadenas de texto (strings.xml) o colores (colors.xml), que son usados en toda la aplicación.

4.3 Configuración del proyecto

La configuración necesaria para que el proyecto funcione correctamente se encuentra en los ficheros build.gradle y AndroidManifest.xml, en ellos se definen las dependencias del proyecto, las configuraciones, los permisos necesarios y todo lo necesario para el uso correcto de la aplicación.

4.3.1 Dependencias del build.gradle

Este archivo es el encargado de definir las dependencias de la aplicación. Se especifican todas las librerías y versiones que se requieren, además de la configuración necesaria para construir la aplicación.

La dependencia más destacable en esta aplicación es:

- com.android.volley:volley:1.2.1: Es una biblioteca utilizada para realizar peticiones HTTP, en esta aplicación se usa para poder enviar las peticiones al servicio REST. Se ha elegido esta biblioteca debido a que permite realizar las peticiones de forma muy eficiente.

4.3.2 Permisos del AndroidManifest.xml

En el archivo AndroidManifest.xml se definen los permisos que requiere la aplicación para poder interactuar con el sistema operativo del dispositivo móvil. Los principales permisos necesarios para esta aplicación son los siguientes:

- Internet: Este permiso permite a la aplicación hacer uso de conexiones a internet. Es necesario para realizar las peticiones HTTP al servicio REST.
- Receive_boot_completed: Permite a la Aplicación recibir una notificación cuando el dispositivo se reinicia. Es importante en la aplicación para gestionar las alarmas correctamente, ya que se deben activar aún tras un reinicio.
- Schedule_exact_alarm: Es el encargado de permitir a la aplicación programar las alarmas a una hora concreta. Es de vital importancia en la aplicación, ya que es el que permite que las notificaciones sean emitidas a la hora correspondiente.

En la figura 4-8 podemos ver un fragmento de este fichero donde se definen los permisos.

AndroidManifest.xml

```
4    <uses-permission android:name="android.permission.INTERNET" />
5    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
6    <uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM"/>
```

Figura 4-8. Fragmento AndroidManifest.xml de SensApp

5 APLICACIÓN SENSAPP CONTROL

La perseverancia no es una carrera larga. Son muchas carreras cortas, una tras otra

Walter Elliot

En este quinto capítulo se describe la aplicación dirigida a los responsables de los usuarios, SensApp Control. Esta les permite programar alarmas y eventos para su usuario asociado, además de poder consultar y modificar tanto sus datos como los del usuario. También permite consultar las incidencias reportadas por el usuario.

El diseño de la aplicación se ha basado en un menú lateral desplegable desde donde se accede a las distintas funcionalidades ofrecidas, para lograr una navegación intuitiva. La estructura, por tanto, se ha simplificado, de forma que solo existen dos actividades principales, una primera para el inicio de sesión y, otra general para el resto de la aplicación. Esta última actividad actúa como contenedor de diferentes fragmentos que representan cada una de las funcionalidades.

A lo largo de este capítulo se van a detallar las funcionalidades, así como la estructura de ficheros de la aplicación y los aspectos clave de la configuración necesaria.

Para facilitar una comprensión global de la aplicación, en la figura 5-1 se presenta el diagrama de casos de uso.

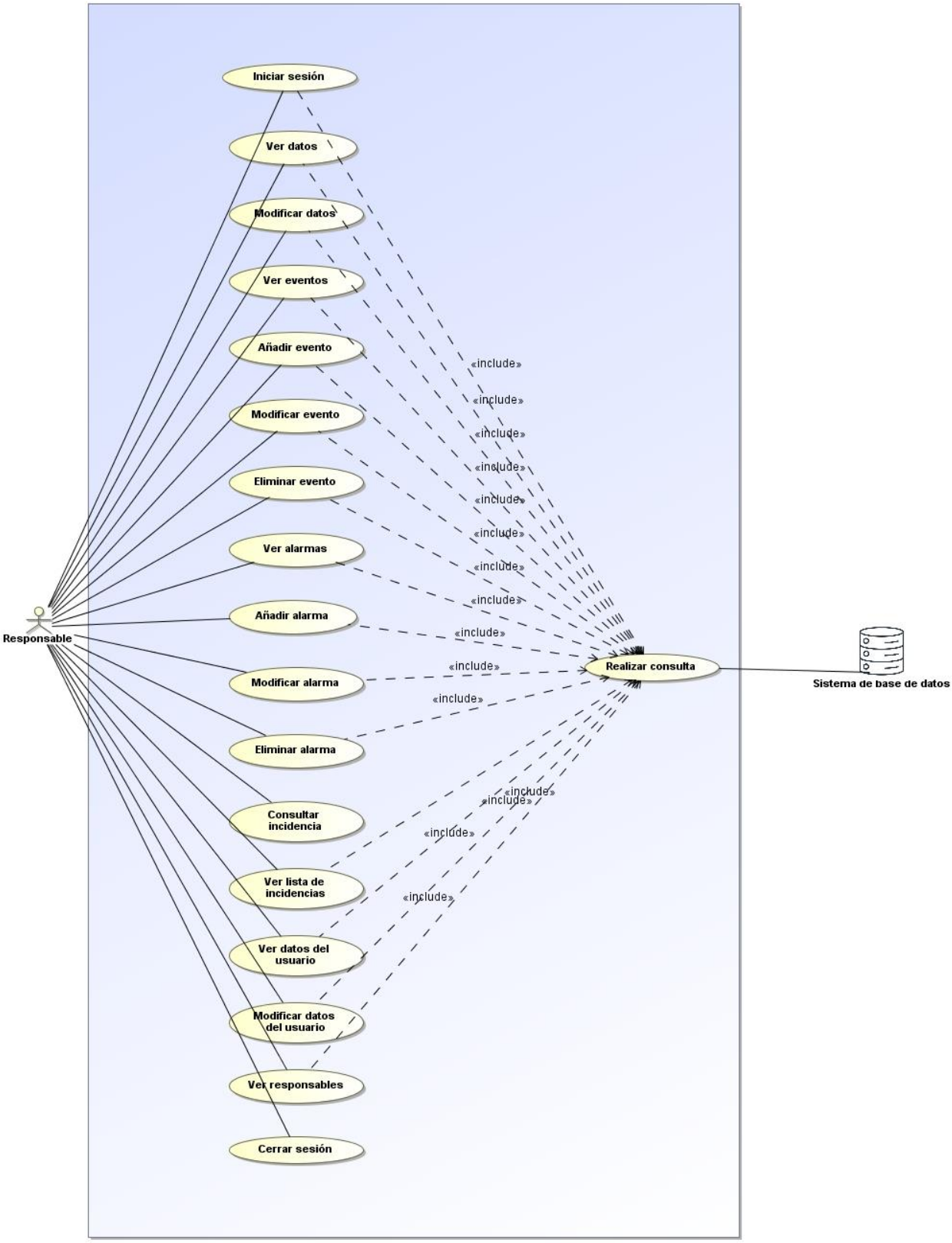

Figura 5-1. Casos de uso aplicación SensApp Control[4]

[4] En el diagrama representado se han omitido las relaciones de extensión, por simplificación. Estas quedan reflejadas en las tablas de cada uno de los casos de usos.

5.1 Funcionalidades

Se detallan en este apartado todas las funcionalidades ofrecidas por la aplicación SensApp Control, detallando para cada una sus casos de uso. En la mayoría de ellos, el actor principal es el responsable.

5.1.1 Inicio de sesión

Al igual que en la aplicación explicada en el capítulo anterior, es necesario que el responsable esté registrado para poder acceder a las funcionalidades ofrecidas. Este proceso de inicio de sesión ofrece seguridad, garantizando que solo los usuarios autorizados puedan acceder a información sensible. La interfaz de usuario correspondiente a esta funcionalidad se muestra en la figura 5-2.

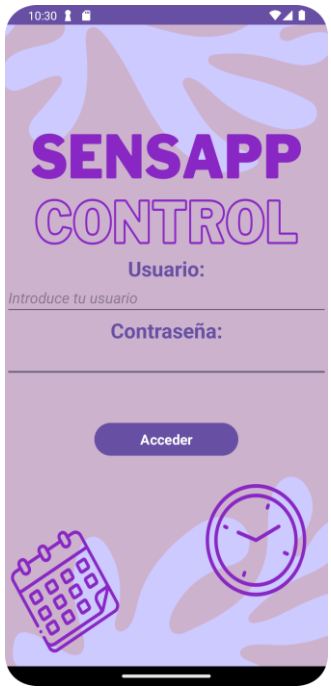

Figura 5-2. Pantalla de inicio de sesión

5.1.1.1 CDU – Iniciar sesión

Este caso de uso es similar al CDU – 1.01 – Iniciar sesión, descrito en el capítulo anterior. Su objetivo es dar acceso a un responsable registrado al resto de funcionalidades de la aplicación. Se detalla en la tabla 5-1.

Tabla 5-1. CDU-2.01 – Iniciar sesión

CDU-2.01	**Iniciar sesión**	
Descripción	El responsable introduce sus credenciales para acceder a la aplicación. Si estas son correctas, el responsable podrá acceder a la pantalla principal de la aplicación, si no, se muestra un mensaje de error.	
Precondición	El responsable debe estar registrado en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El responsable introduce un nombre de usuario y una contraseña en la pantalla de inicio de sesión y pulsa el botón “Iniciar sesión”.
	2	Se llama al CDU – 2.18 para realizar una petición al servicio REST que permite comprobar si existe el usuario mediante una consulta a la

		BBDD.
	3	Si los datos introducidos se corresponden con un responsable registrado en el sistema, se da acceso al usuario a la pantalla principal de la aplicación.
Postcondición	El responsable puede acceder a todas las funcionalidades de la aplicación.	
Excepciones	Paso	Acción
	3	Si los datos no son correctos, se muestra un mensaje de error al usuario.

5.1.2 Pantalla principal

Cuando el responsable inicia sesión exitosamente accede a un menú desde el cual puede acceder a todas las funcionalidades disponibles. Estas se dividen en cinco secciones: Datos, Eventos, Alarmas, Ver incidencias y Ver usuario, como se puede ver en la figura 5-3.

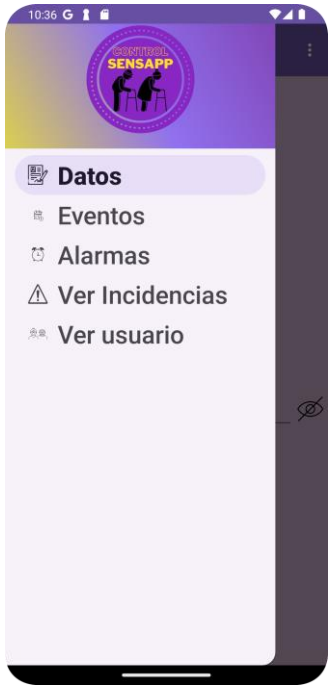

Figura 5-3. Menú desplegable lateral Responsables

En los siguientes apartados se describen las funcionalidades disponibles desde cada una de las secciones de este menú, detallando cada caso de uso.

5.1.3 Datos

Esta primera sección es la que se muestra por defecto al acceder a la aplicación tras iniciar sesión. Desde ella el responsable puede consultar y modificar sus datos personales. Esto es fundamental para mantener actualizada la información personal relevante. La interfaz disponible para el responsable se muestra en la figura 5-4.

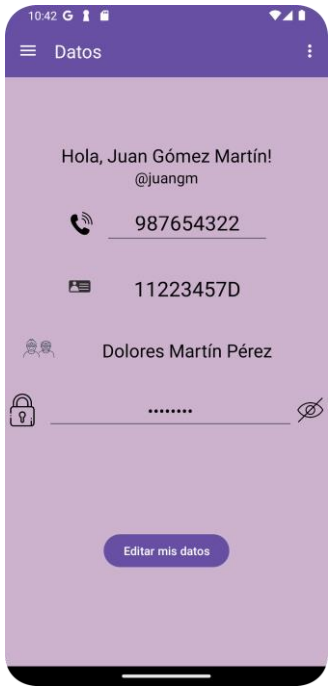

Figura 5-4. Datos del responsable

A continuación, se van a describir los casos de uso asociados a esta sección.

5.1.3.1 CDU – Ver datos

El objetivo de este caso de uso es permitir al responsable consultar su información personal de manera simple.

Tabla 5-2. CDU-2.02 – Ver datos

CDU-2.02	**Ver datos**	
Descripción	Permite al responsable consultar sus datos personales.	
Precondición	El responsable debe haber iniciado sesión con éxito (CDU – 2.01).	
Secuencia normal	**Paso**	**Acción**
	1	El responsable accede a la sección “Datos” del menú lateral.
	2	Se llama al CDU – 2.18 para realizar una petición al servicio REST que obtenga los datos del responsable mediante una consulta a la BBDD.
	2.1	Los datos del responsable se muestran en pantalla.
Postcondición	El responsable puede consultar toda su información personal.	

5.1.3.2 CDU – Modificar datos

El objetivo de este caso de uso es permitir al responsable actualizar sus datos personales de forma simple.

Tabla 5-3. CDU-2.03 – Modificar datos

CDU-2.03	**Modificar datos**

<table>
<tr><td>Descripción</td><td>Permite al responsable actualizar algunos de sus datos personales (número de teléfono y contraseña).</td></tr>
<tr><td>Precondición</td><td>El responsable debe haber iniciado sesión con éxito (CDU – 2.01) y estar en la sección “Datos” del menú desplegable lateral (CDU – 2.02).</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El responsable modifica uno o más campos de texto de los mostrados en pantalla.</td></tr>
<tr><td>2</td><td>El responsable pulsa el botón “Editar mis datos”.</td></tr>
<tr><td>3</td><td>Se llama al CDU – 2.18 para realizar una petición al servicio REST que actualice los datos del responsable mediante una consulta a la BBDD.</td></tr>
<tr><td>3.1</td><td>Los datos actualizados del responsable se muestran en pantalla.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>El responsable puede ver toda su información personal actualizada.</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>2</td><td>Si algún campo de texto está vacío, se muestra un mensaje de error al responsable, indicando que se deben rellenar todos los campos.</td></tr>
</table>
</td></tr>
</table>

5.1.4 Eventos

En esta sección se pueden encontrar las funcionalidades que permiten gestionar los eventos, es decir, visualizar una lista de estos, modificar o eliminar un evento en concreto y crear uno nuevo. Cada uno de los casos de uso existentes se detallan en los siguientes subapartados.

La interfaz principal de esta sección se puede ver en la figura 5-5.

Figura 5-5. Ver eventos

5.1.4.1 CDU – Ver eventos

El objetivo de este caso de uso es permitir al responsable visualizar la lista de eventos existente de una manera muy intuitiva, ya que se representan en orden cronológico y además cada evento se representa de un color dependiendo de su estado, verde si ha sido completado y rojo si no. Esto permite al responsable poder llevar un seguimiento de las actividades realizadas por el usuario.

Tabla 5-4. CDU-2.04 – Ver eventos

<table>
<tr><td>CDU-2.04</td><td colspan="2">Ver eventos</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al responsable ver una lista de eventos registrados para el usuario.</td></tr>
<tr><td>Precondición</td><td colspan="2">El responsable debe haber iniciado sesión con éxito (CDU – 2.01)</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El responsable accede a la sección “Eventos” del menú lateral.</td></tr>
<tr><td>2</td><td>Se llama al CDU – 2.18 para realizar una petición al servicio REST que obtenga una lista paginada de los eventos del usuario correspondiente mediante una consulta a la BBDD.</td></tr>
<tr><td>2.1</td><td>Los eventos se muestran en pantalla de forma ordenada.</td></tr>
<tr><td>3</td><td>El responsable se desplaza por la pantalla (realiza scroll[5] en la lista).</td></tr>
<tr><td>4</td><td>Cuando el usuario llega hasta el final de la lista, se llama de nuevo al CDU – 2.18 para obtener la siguiente página de la lista mediante una petición al servicio REST que realiza la consulta correspondiente en la BBDD.</td></tr>
<tr><td>4.1</td><td>Los eventos se añaden a la lista mostrada en pantalla, de forma ordenada.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El responsable puede consultar todos los eventos.</td></tr>
</table>

5.1.4.2 CDU – Añadir evento

Este caso de uso permite añadir un nuevo evento para el usuario, definiendo sus detalles. Para ofrecer esta funcionalidad se define una interfaz que se muestra en la figura 5-6.

[5] Realizar scroll es desplazarse por el contenido de la aplicación móvil para permitir visualizar información que no cabe en la pantalla visible.

Figura 5-6. Añadir evento

El caso de uso se detalla en la tabla 5-5.

Tabla 5-5. CDU-2.05 – Añadir evento

<table>
<tr><td>CDU-2.05</td><td colspan="2">Añadir evento</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al responsable añadir un nuevo evento para el usuario.</td></tr>
<tr><td>Precondición</td><td colspan="2">El responsable debe haber iniciado sesión con éxito (CDU – 2.01) y estar en la sección “Eventos” del menú desplegable lateral (CDU – 2.04).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El responsable pulsa el botón “Crear evento”.</td></tr>
<tr><td>2</td><td>Se navega hasta un nuevo fragmento dedicado a esta funcionalidad.</td></tr>
<tr><td>3</td><td>El responsable rellena el campo de texto ‘nombre’, selecciona un tipo de evento en el spinner y selecciona la fecha del evento en el Date Picker[6].</td></tr>
<tr><td>4</td><td>El responsable pulsa el botón “Crear evento”.</td></tr>
<tr><td>5</td><td>Se llama al CDU – 2.18 para realizar una petición al servicio REST que permita crear el evento mediante una consulta a la BBDD.</td></tr>
<tr><td>6</td><td>Se navega hasta el fragmento correspondiente a editar el evento (CDU – 2.07) y se muestran los datos del evento creado en pantalla.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El responsable puede ver los detalles del evento creado en pantalla.</td></tr>
</table>

[6] En Android Studio un Date Picker es un componente que permite al usuario de la aplicación seleccionar una fecha mediante un diálogo de calendario.

Excepciones	Paso	Acción
	4	Si el campo de texto se encuentra vacío, se muestra un mensaje de error al responsable, indicando que se debe introducir un nombre para el evento.
	4	Si el nombre introducido para el nuevo evento ya existe, se muestra un mensaje de error al responsable, indicando que ya existe un evento con ese nombre.

5.1.4.3 CDU – Modificar evento

El objetivo de este caso de uso es permitir modificar un evento ya existente. La interfaz utilizada para esto se muestra a continuación en la figura 5-7.

Figura 5-7. Modificar evento

Este caso de uso se detalla en la tabla 5-6.

Tabla 5-6. CDU-2.06 – Modificar evento

CDU-2.06	**Modificar evento**	
Descripción	Permite al responsable modificar los detalles de un evento existente.	
Precondición	El responsable debe haber iniciado sesión con éxito (CDU – 2.01) y estar en la sección "Eventos" del menú desplegable lateral (CDU – 2.04).	
Secuencia normal	**Paso**	**Acción**
	1	El responsable pulsa sobre el evento de la lista que desee editar.
	2	Se navega hasta un nuevo fragmento dedicado a modificar eventos, pasando el evento seleccionado en el bundle[7] de la navegación.

[7] Bundle es un contenedor de pares clave-valor utilizado para almacenar y transferir información entre los fragmentos de una aplicación.

	2.1	Se muestran los detalles del evento en pantalla.
	3	El responsable realiza los cambios que desee.
	4	El responsable pulsa el botón “Editar evento”.
	5	Se llama al CDU – 2.18 para realizar una petición al servicio REST que permita actualizar el evento mediante una consulta a la BBDD.
	5.1	Se muestran los datos del evento actualizado en pantalla.
Postcondición	El responsable puede ver los detalles del evento actualizado en pantalla.	
Excepciones	Paso	Acción
	4	Si el campo de texto correspondiente al nombre del evento se encuentra vacío, se muestra un mensaje de error al responsable, indicando que se debe introducir un nombre.
	4	Si ya existe un evento con el nombre introducido, se muestra un mensaje de error al responsable, indicando que ya existe un evento con ese nombre.

5.1.4.4 CDU – Eliminar evento

Con este caso de uso se ofrece al usuario la posibilidad de eliminar un evento existente. Para evitar posibles errores, se muestra un diálogo de confirmación para llevar a cabo la eliminación del evento, como se puede ver en la figura 5-8.

Figura 5-8. Eliminar evento

Este caso de uso se detalla en la tabla 5-7.

Tabla 5-7. CDU-2.07 – Eliminar evento

CDU-2.07	**Eliminar evento**

<table>
<tr><td>Descripción</td><td>Permite al responsable eliminar un evento existente.</td></tr>
<tr><td>Precondición</td><td>El responsable debe haber iniciado sesión con éxito (CDU – 2.01) y estar en la sección “Eventos” del menú desplegable lateral (CDU – 2.04).</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El responsable realiza un long click sobre el evento de la lista que desea eliminar.</td></tr>
<tr><td>2</td><td>Se muestra un diálogo al responsable para confirmar que se quiere eliminar el evento.</td></tr>
<tr><td>3</td><td>El responsable pulsa “Sí” en el diálogo.</td></tr>
<tr><td>4</td><td>Se llama al CDU – 2.18 para realizar una petición al servicio REST para eliminar el evento seleccionado mediante una consulta a la BBDD.</td></tr>
<tr><td>4.1</td><td>Se muestra la lista actualizada en pantalla.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>El responsable puede ver la lista de eventos actualizada.</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>3</td><td>Si el responsable pulsa “No”, el evento no se elimina.</td></tr>
</table>
</td></tr>
</table>

5.1.5 Alarmas

Esta sección es análoga a la anterior, se pueden encontrar las funcionalidades que permiten gestionar las alarmas, en lugar de los eventos como en la anterior. Permite por tanto, visualizar una lista de alarmas, modificar o eliminar una en concreto y crear una nueva. La interfaz principal de esta sección se puede observar en la figura 5-9.

Figura 5-9. Ver alarmas

Los casos de uso se explican en los siguientes subapartados.

5.1.5.1 CDU – Ver alarmas

Al igual que en el caso de uso Ver eventos, el objetivo de este caso de uso es permitir al responsable visualizar la lista de alarmas existentes. Estas alarmas se representan en orden cronológico y en función de su estado, las no completadas se muestran al principio de la lista y las completadas al final. Cada una de ellas se representa en un color dependiendo de su estado, verde para las que han sido completadas, azul para aquellas que aún no se han activado y rojo para las que no han sido completadas y cuya hora ya ha pasado. Esta forma de representación permite al responsable visualizar de forma rápida si el usuario ha realizado o no sus actividades.

Tabla 5-8. CDU-2.08 – Ver alarmas

<table>
<tr><td>CDU-2.08</td><td colspan="2">Ver alarmas</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al responsable ver una lista de las alarmas existentes y sus estados.</td></tr>
<tr><td>Precondición</td><td colspan="2">El responsable debe haber iniciado sesión con éxito (CDU – 2.01).</td></tr>
<tr><td>Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>El responsable accede a la sección “Alarmas” del menú lateral.</td></tr>
<tr><td></td><td>2</td><td>Se llama al CDU – 2.18 para realizar una petición al servicio REST que permita obtener las alarmas del usuario correspondiente mediante una consulta a la BBDD.</td></tr>
<tr><td></td><td>2.1</td><td>Las alarmas se muestran en pantalla de forma ordenada.</td></tr>
<tr><td></td><td>3</td><td>El responsable realiza scroll en pantalla.</td></tr>
<tr><td></td><td>4</td><td>Cuando el usuario llega hasta el final de la lista, se llama de nuevo al CDU – 2.18 para obtener la siguiente página de la lista mediante una petición al servicio REST que realiza la consulta correspondiente en la BBDD.</td></tr>
<tr><td></td><td>4.1</td><td>Las alarmas se añaden a la lista mostrada en pantalla, de forma ordenada.</td></tr>
<tr><td>Postcondición</td><td colspan="2">El responsable puede consultar todas las alarmas.</td></tr>
</table>

5.1.5.2 CDU – Añadir alarma

Este caso de uso permite añadir una nueva alarma, del mismo modo que el caso de uso Añadir evento permitía añadir un evento. Para ofrecer esta funcionalidad se define una interfaz que se muestra en la figura 5-10.

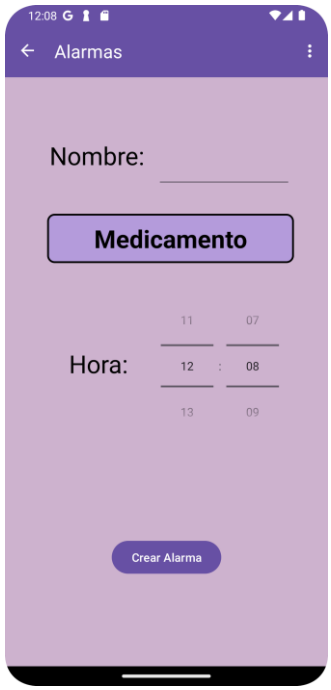

Figura 5-10. Añadir alarma

El caso de uso se detalla en la tabla 5-9.

Tabla 5-9. CDU-2.09 – Añadir alarma

<table>
<tr><td>CDU-2.09</td><td colspan="2">Añadir alarma</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al responsable añadir una alarma.</td></tr>
<tr><td>Precondición</td><td colspan="2">El responsable debe haber iniciado sesión con éxito (CDU – 2.01) y estar en la sección “Alarmas” del menú desplegable lateral (CDU – 2.08).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El responsable pulsa el botón “Crear evento”.</td></tr>
<tr><td>2</td><td>Se navega hasta un nuevo fragmento dedicado a esta funcionalidad.</td></tr>
<tr><td>3</td><td>El responsable rellena el campo de texto “nombre”, selecciona un tipo de alarma en el spinner y selecciona la fecha del evento en el Time Picker[8].</td></tr>
<tr><td>4</td><td>El responsable pulsa el botón “Crear alarma”.</td></tr>
<tr><td>5</td><td>Se llama al CDU – 2.18 para realizar una petición al servicio REST que permita crear el evento mediante una consulta a la BBDD.</td></tr>
<tr><td>5.1</td><td>Se navega hasta el fragmento correspondiente a editar la alarma (CDU – 2.10) y se muestran los datos de la alarma creada en pantalla.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El responsable puede ver los detalles de la alarma creada en pantalla.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table>
<tr><td>Paso</td><td>Acción</td></tr>
</table></td></tr>
</table>

[8] En Android Studio un Time Picker es un componente que permite al usuario de la aplicación seleccionar una hora específica.

	4	Si el campo de texto se encuentra vacío, se muestra un mensaje de error al responsable, indicando que se debe introducir un nombre.
	4	Si el nombre introducido para la alarma ya existe, se muestra un mensaje de error al responsable, indicando que ya existe una alarma con ese nombre.

5.1.5.3 CDU – Modificar alarma

El propósito de este caso de uso es ofrecer la posibilidad de editar una alarma existente. La interfaz correspondiente para llevar a cabo esta acción se puede observar en la figura 5-11.

Figura 5-11. Modificar alarma

En la tabla 5-10 se detalla este caso de uso.

Tabla 5-10. CDU-2.10 – Modificar alarma

CDU-2.10	**Modificar alarma**	
Descripción	Permite al responsable modificar los detalles de una alarma existente.	
Precondición	El responsable debe haber iniciado sesión con éxito (CDU – 2.01) y estar en la sección "Alarmas" del menú desplegable lateral (CDU – 2.08).	
Secuencia normal	**Paso**	**Acción**
	1	El responsable pulsa sobre la alarma de la lista que desee editar.
	2	Se navega hasta un nuevo fragmento dedicado a esta funcionalidad, pasando la alarma seleccionada en el bundle de la navegación.
	2.1	Se muestran los detalles de la alarma en pantalla.
	3	El responsable realiza los cambios que desee.

	4	El responsable pulsa el botón “Editar alarma”.
	5	Se llama al CDU – 2.18 para realizar una petición al servicio REST que permita actualizar la alarma mediante una consulta a la BBDD.
	5.1	Se muestran los datos de la alarma actualizada en pantalla.
Postcondición	El responsable puede ver los detalles de la alarma actualizada en pantalla.	
Excepciones	Paso	Acción
	4	Si el campo de texto correspondiente al nombre de la alarma se encuentra vacío, se muestra un mensaje de error al responsable, indicando que se debe introducir un nombre.
	4	Si ya existe una alarma con el nombre introducido, se muestra un mensaje de error al responsable, indicando que ya existe una alarma con ese nombre.

5.1.5.4 CDU – Eliminar alarma

La finalidad de este caso de uso es habilitar la posibilidad de eliminar alarmas existentes en el sistema. Se ha configurado un diálogo de confirmación para evitar posibles errores, este se puede ver en la figura 5-12.

Figura 5-12. Eliminar alarma

Este caso de uso se detalla a continuación.

Tabla 5-11. CDU-2.11 – Eliminar alarma

CDU-2.11	**Eliminar alarma**
Descripción	Permite al responsable eliminar una alarma existente.
Precondición	El responsable debe haber iniciado sesión con éxito (CDU – 2.01) y estar en la sección “Alarmas” del menú desplegable lateral (CDU – 2.08).

Secuencia normal	**Paso**	**Acción**
	1	El responsable realiza un long click sobre la alarma de la lista que desee eliminar.
	2	Se muestra un diálogo al responsable para confirmar que se quiere eliminar la alarma.
	3	El responsable pulsa “Sí” en el diálogo.
	4	Se llama al CDU – 2.18 para realizar una petición al servicio REST que permita eliminar la alarma seleccionada mediante una consulta a la BBDD.
	4.1	Se muestran la lista de alarmas actualizada en pantalla.
Postcondición	El responsable puede ver la lista de alarmas actualizada.	
Excepciones	Paso	Acción
	3	Si el responsable pulsa “No”, la alarma no se elimina.

5.1.6 Ver Incidencias

La tercera sección del menú está dedicada a las incidencias del usuario, se permite consultar una lista de todas ellas, donde se puede ver el estado según el color en el que se representan (rojo para incidencias no resueltas y verde para la resueltas) y consultar en detalle cada una de ellas. Además, la lista de incidencias se presenta ordenada en orden cronológico. La interfaz principal de esta sección se muestra en la figura 5-13.

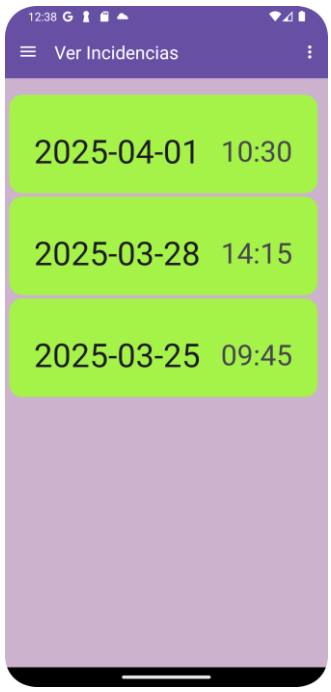

Figura 5-13. Ver incidencias

Los casos de uso se explican en los siguientes subapartados.

5.1.6.1 CDU – Ver lista de incidencias

Este caso de uso tiene como finalidad permitir visualizar la lista de todas las incidencias reportadas para un usuario. A continuación se detalla el caso de uso en la tabla 5-12.

Tabla 5-12. CDU-2.12 – Ver lista de incidencias

CDU-2.12	**Ver lista de incidencias**	
Descripción	Permite al responsable ver una lista de las incidencias reportadas asociadas al usuario.	
Precondición	El responsable debe haber iniciado sesión con éxito (CDU – 2.01).	
Secuencia normal	**Paso**	**Acción**
	1	El responsable accede a la sección "Ver incidencias" del menú lateral.
	2	Se llama al CDU – 2.18 para realizar una petición al servicio REST que permita obtener las incidencias del usuario correspondiente mediante una consulta a la BBDD.
	2.1	Las incidencias se muestran en pantalla de forma ordenada.
	3	El responsable realiza scroll en pantalla.
	4	Cuando el usuario llega hasta el final de la lista, se llama de nuevo al CDU – 2.18 para obtener la siguiente página de la lista mediante una petición al servicio REST que realiza la consulta correspondiente en la BBDD.
	4.1	Las incidencias se añaden a la lista mostrada en pantalla, de forma ordenada.
Postcondición	El responsable puede consultar todas las incidencias.	

5.1.6.2 CDU – Consultar incidencia

El caso de uso Consultar incidencia ofrece la posibilidad de consultar los detalles de una incidencia. La interfaz correspondiente para llevar a cabo esta acción se puede observar en la figura 5-14.

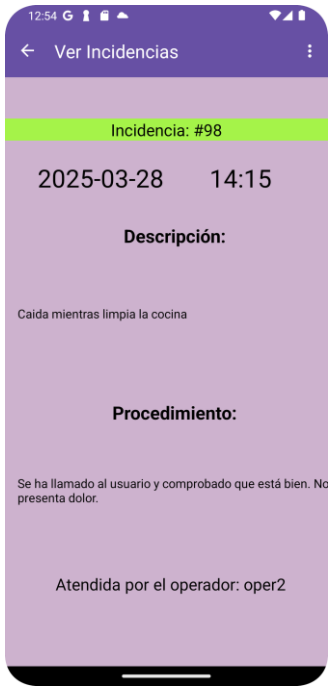

Figura 5-14. Consultar incidencia

En la tabla 5-13 se detalla este caso de uso.

Tabla 5-13. CDU-2.13 – Consultar incidencia

<table>
<tr><td>CDU-2.13</td><td colspan="2">Consultar incidencia</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al responsable consultar los detalles de una incidencia.</td></tr>
<tr><td>Precondición</td><td colspan="2">El responsable debe haber iniciado sesión con éxito (CDU – 2.01) y estar en la sección “Ver incidencias” del menú desplegable lateral (CDU – 2.08).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El responsable pulsa sobre la incidencia de la lista que desee consultar.</td></tr>
<tr><td>2</td><td>Se navega hasta un nuevo fragmento dedicado a esta funcionalidad, pasando la incidencia seleccionada en el bundle de la navegación.</td></tr>
<tr><td>2.1</td><td>Se muestran los detalles de la incidencia en pantalla.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El responsable puede ver los detalles de la incidencia en pantalla.</td></tr>
</table>

5.1.7 Ver usuario

La última sección es “Ver usuario”, en esta se puede consultar y modificar los datos personales del usuario. Esto resulta crucial para el sistema, ya que mantener actualizado el número de teléfono del usuario es de vital importancia para que los operadores puedan resolver las incidencias reportadas. La interfaz ofrecida al usuario se puede observar en las figuras 5-15 y 5-16, ya que, para ver todos los elementos de la pantalla es necesario hacer scroll.

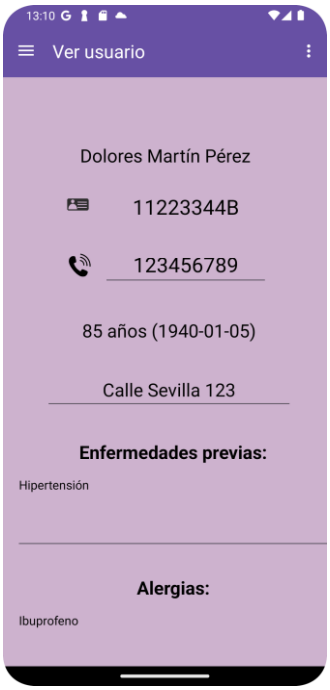

Figura 5-16. Ver usuario (1/2)

Figura 5-15. Ver usuario (2/2)

A continuación, se van a describir los casos de uso asociados a esta última sección.

5.1.7.1 CDU – Ver datos del usuario

El objetivo de este caso de uso es permitir al responsable consultar la información personal del usuario de manera simple.

Tabla 5-14. CDU-2.14 – Ver datos del usuario

<table>
<tr><td>CDU-2.14</td><td colspan="2">Ver datos del usuario</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al responsable ver los datos personales del usuario.</td></tr>
<tr><td>Precondición</td><td colspan="2">El responsable debe haber iniciado sesión con éxito (CDU – 2.01).</td></tr>
<tr><td>Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>El responsable accede a la sección “Ver usuario” del menú lateral.</td></tr>
<tr><td></td><td>2</td><td>Se llama al CDU – 2.18 para realizar una petición al servicio REST que permita obtener los datos del usuario mediante una consulta a la BBDD.</td></tr>
<tr><td></td><td>2.1</td><td>Los datos del usuario se muestran en pantalla.</td></tr>
<tr><td>Postcondición</td><td colspan="2">El responsable puede consultar toda la información personal del usuario.</td></tr>
</table>

5.1.7.2 CDU – Modificar datos del usuario

El objetivo de este caso es permitir al responsable actualizar los datos personales del usuario.

Tabla 5-15. CDU-2.15 – Modificar datos del usuario

<table>
<tr><td>CDU-2.15</td><td colspan="2">Modificar datos del usuario</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al responsable actualizar algunos de los datos personales del usuario (número de teléfono, domicilio, enfermedades previas, alergias y contraseña).</td></tr>
<tr><td>Precondición</td><td colspan="2">El responsable debe haber iniciado sesión con éxito (CDU – 2.01) y estar en la sección “Ver usuario” del menú desplegable lateral (CDU – 2.14).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El responsable modifica uno o más campos de texto de los mostrados en pantalla.</td></tr>
<tr><td>2</td><td>El responsable pulsa el botón “Editar datos”</td></tr>
<tr><td>3</td><td>Se llama al CDU – 2.18 para realizar una petición al servicio REST que permita actualizar los datos del usuario mediante una consulta a la BBDD.</td></tr>
<tr><td>3.1</td><td>Los datos actualizados del usuario se muestran en pantalla.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El responsable puede ver toda la información personal del usuario actualizada.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>2</td><td>Si algún campo de texto está vacío y no admite un valor nulo, como por ejemplo el número de teléfono, se muestra un mensaje de error al usuario, indicando que se deben rellenar los campos.</td></tr>
</table></td></tr>
</table>

5.1.7.3 CDU – Ver responsables

Este caso de uso ofrece la funcionalidad de consultar una lista con los nombres y números de teléfono de todos los responsables asociados al usuario. La interfaz gráfica usada para esto se puede ver en la figura 5-17.

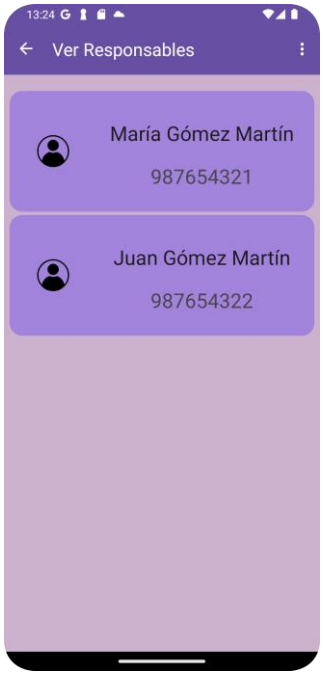

Figura 5-17. Ver responsables

El caso de uso se detalla en la tabla 5-16.

Tabla 5-16. CDU-2.16 – Ver responsables

<table>
<tr><td>CDU-2.16</td><td colspan="2">Ver responsables</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al responsable consultar una lista con todos los responsables del usuario.</td></tr>
<tr><td>Precondición</td><td colspan="2">El responsable debe haber iniciado sesión con éxito (CDU – 2.01) y estar en la sección “Ver usuario” del menú desplegable lateral (CDU – 2.14).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El responsable pulsa el botón “Ver responsables”.</td></tr>
<tr><td>2</td><td>Se llama al CDU – 2.18 para realizar una petición al servicio REST que permita obtener todos los responsables del usuario mediante una consulta a la BBDD.</td></tr>
<tr><td>2.1</td><td>La lista de responsables se muestra en pantalla.</td></tr>
<tr><td>3</td><td>El responsable realiza scroll en la lista.</td></tr>
<tr><td>4</td><td>Cuando el usuario llega hasta el final de la lista, se llama de nuevo al CDU – 2.18 para obtener la siguiente página de la lista mediante una petición al servicio REST que realiza la consulta correspondiente en la BBDD.</td></tr>
<tr><td>4.1</td><td>Los responsables se añaden a la lista mostrada en pantalla, de forma ordenada.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El responsable puede ver todos los responsables del usuario.</td></tr>
</table>

5.1.8 Cerrar sesión

La última funcionalidad ofrecida en esta aplicación es cerrar sesión, esta permite volver a la actividad de inicio de sesión. Está disponible desde el menú de la esquina superior derecha, como se puede ver en la figura 5-18.

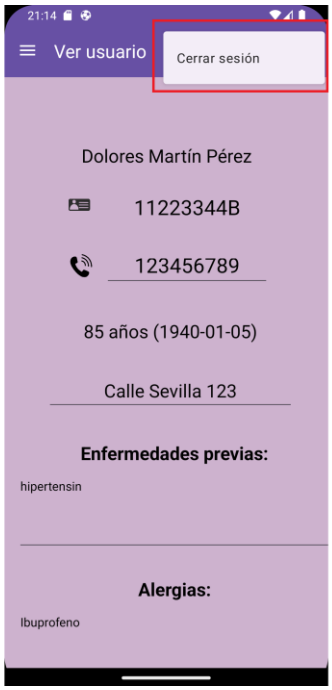

Figura 5-18. Cerrar sesión

5.1.8.1 CDU – Cerrar sesión

Este caso de uso se detalla en la tabla 5-17.

Tabla 5-17. CDU-2.17 – Cerrar sesión

CDU-2.17	**Cerrar sesión**	
Descripción	Permite al responsable cerrar sesión en la aplicación.	
Precondición	El responsable debe haber iniciado sesión con éxito (CDU – 2.01).	
Secuencia normal	**Paso**	**Acción**
	1	El responsable accede al menú superior derecho pulsando en los tres puntos que aparecen.
	2	El responsable pulsa la opción “Cerrar sesión”.
	3	Se navega hasta la actividad encargada de iniciar sesión.
Postcondición	El responsable se encuentra en la actividad de inicio de sesión.	

5.1.9 Petición al Servicio REST y consulta a la BBDD

5.1.9.1 CDU- Realizar consulta

Este caso de uso se aplica en la mayoría de los anteriores, y permite realizar una consulta a la base de datos utilizada por el sistema para consultar, modificar, crear o eliminar algún dato. Se detalla en la tabla 5-18.

Tabla 5-18. CDU-2.18 – Realizar consulta

CDU-2.18	**Realizar consulta**

<table>
<tr><td>Descripción</td><td>Permite realizar una consulta a la base de datos.</td></tr>
<tr><td>Actor principal</td><td>Sistema de base de datos</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>Se realiza una petición al servicio REST para obtener, modificar, crear o eliminar algún dato de la BBDD.</td></tr>
<tr><td>2</td><td>Se ejecuta la consulta SQL.</td></tr>
<tr><td>3</td><td>Se devuelve el resultado del método REST invocado.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>Se ha realizado una consulta a la BBDD.</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>1</td><td>Si no hay conexión a internet no se podrá realizar la petición correctamente.</td></tr>
<tr><td>2</td><td>Si no hay acceso a la BBDD no se podrá realizar la consulta.</td></tr>
</table>
</td></tr>
</table>

5.2 Estructura de ficheros de la aplicación

En este apartado se describe la estructura de ficheros de la aplicación, explicando el contenido de los principales directorios. Esta está pensada para facilitar el mantenimiento de la aplicación y el uso de reutilización.

Los principales ficheros de código y de configuración de la aplicación se encuentran en el directorio src/main/. En esta carpeta podemos encontrar también el fichero de configuración AndroidManifest.xml, que contiene la configuración necesaria para el correcto funcionamiento de la aplicación.

También podemos encontrar en este dos grandes directorios, java/com/example/appresponsables/, donde se almacena todo el código fuente, y res/, donde se almacenan los diferentes recursos. Estos se explican en los siguientes subapartados.

5.2.1 Directorio del código fuente de la aplicación

En este directorio podemos encontrar el código fuente de la aplicación. En sus ficheros se define toda la lógica de las funcionalidades descritas en apartados anteriores y de toda la aplicación. En la figura 5-19 se puede observar su estructura.

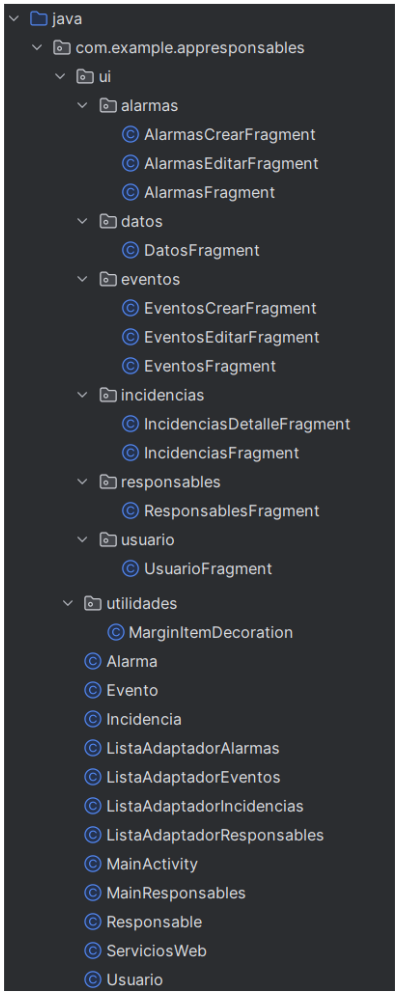

Figura 5-19. Estructura del directorio java/com/example/appresponsables/

Los principales archivos de este directorio son los siguientes:

- Alarma.java, Evento.java, Incidencia.java, Responsable.java y Usuario.java: JavaBeans utilizados para representar las entidades Alarma, Evento, Incidencia, Responsable y Usuario, respectivamente, en la aplicación. Todas contienen los atributos y métodos getter y setter correspondientes según lo visto en el capítulo 3 de este documento.
- ListaAdaptadorAlarmas.java, ListaAdaptadorEventos.java, ListaAdaptadorIncidencias.java y ListaAdaptadorResponsables.java: Clases encargadas de adaptar la visualización de una lista de alarmas, eventos, incidencias o responsables, respectivamente, para poder ofrecer al usuario una interfaz gráfica más atractiva y simple.
- ServiciosWeb.java: Clase encargada de gestionar una única instancia de la cola de peticiones al servidor REST mediante el patrón Singleton. Esta clase permite realizar las peticiones de manera centralizada, lo que garantiza que solo existirá una cola de solicitudes.
- MainActivity.java: Actividad encargada de gestionar el inicio de sesión del responsable y de redirigirlo, en caso de éxito, a la pantalla principal (CDU -2.01 – Iniciar sesión).
- MainResponsables.java: Actividad principal de la aplicación. Esta funciona como contenedor de los fragmentos que ofrecen las funcionalidades descritas en este capítulo. También es la encargada de ofrecer la funcionalidad de cerrar sesión (CDU -2.17 – Cerrar sesión).
- Utilidades/MarginItemDecoration.java: Clase encargada de agregar márgenes personalizados entre los elementos de un RecyclerView, mejorando la apariencia de las listas mostradas en pantalla.

El directorio ui/ contiene los componentes de la interfaz de usuario. Se organizan en subdirectorios según la funcionalidad que ofrece. Cada uno de ellos contiene los fragmentos que permiten ofrecer las funcionalidades descritas en el apartado anterior. La estructura de este directorio es la siguiente:

- alarmas/: Contiene los fragmentos relacionados con la gestión de alarmas:
    - AlarmasCrearFragment.java: Fragmento encargado de ofrecer la funcionalidad de añadir una nueva alarma (CDU -2.09 – Añadir alarma).
    - AlarmasEditarFragment.java: Fragmento que ofrece la posibilidad de editar una alarma existente en el sistema (CDU -2.10 – Modificar alarma).
    - AlarmasFragment.java: Fragmento encargado de ofrecer las funcionalidades de consultar la lista de todas las alarmas y de eliminarlas (CDU -2.08 – Ver alarmas y CDU -2.11 – Eliminar alarma).
- datos/: Contiene el fragmento relacionado con la gestión de los datos personales del responsable:
    - DatosFragment.java: Fragmento que maneja la visualización y la actualización de los datos del responsable (CDU -2.02 – Ver datos y CDU -2.03 – Modificar datos).
- eventos/: Contiene los fragmentos relacionados con la gestión de eventos.
    - EventosCrearFragment.java: Fragmento encargado de ofrecer la funcionalidad de añadir un nuevo evento (CDU -2.05 – Añadir evento).
    - EventosEditarFragment.java: Fragmento que ofrece la posibilidad de editar un evento existente en el sistema (CDU -2.06 – Modificar evento).
    - EventosFragment.java: Fragmento encargado de ofrecer las funcionalidades de consultar la lista de todos los eventos y de eliminarlos (CDU -2.04 – Ver eventos y CDU -2.07 – Eliminar evento).
- incidencias/: Contiene los fragmentos relacionados con la visualización de incidencias.
    - IncidenciasDetalleFragment.java: Este fragmento ofrece la posibilidad de consultar en detalle una determinada incidencia (CDU -2.13 – Consultar incidencia).
    - IncidenciasFragment.java: Fragmento encargado de ofrecer la funcionalidad de consultar la lista de todas las incidencias (CDU -2.12 – Ver lista de incidencias).
- responsables/: Contiene el fragmento relacionado con la visualización de la lista de responsables.
    - ResponsablesFragment.java: Este fragmento ofrece la posibilidad de consultar la lista de todos los responsables existentes de un usuario (CDU -2.16 – Ver responsables).
- usuario/: Contiene el fragmento relacionado con la gestión de los datos personales del usuario.
    - UsuarioFragment.java: Fragmento que maneja la visualización y la actualización de los datos del usuario (CDU -2.14 – Ver datos del usuario y CDU -2.15 – Modificar datos del usuario).

5.2.2 Directorio de recursos

Este directorio contiene los recursos usados en la aplicación. A su vez, este directorio contiene subdirectorios específicos según el tipo de recurso que se almacena dentro, como se puede ver en la figura. 5-20.

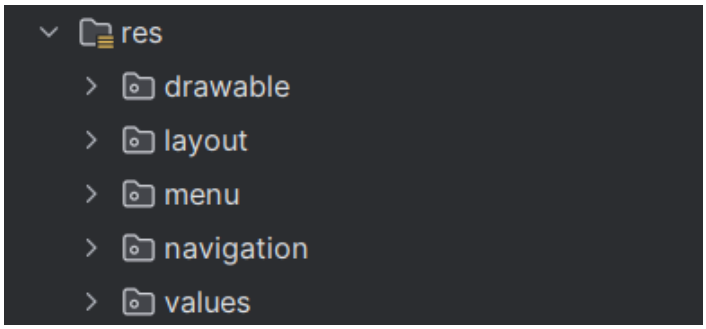

Figura 5-20. Estructura del directorio res/

Cada una de estas carpetas almacena recursos de un tipo concreto:

- Drawable: Contiene los recursos gráficos de los que se hacen uso en el proyecto. Principalmente se almacenan en formato .png o .xml.
- Layout: Contiene todos los ficheros XML que definen las interfaces de usuario, ya sean pantallas de la aplicación o recursos que se usan en las pantallas.
- Menu: Contiene los ficheros XML que definen las opciones de los menús de la aplicación, es decir, el menú desplegable lateral y el menú de la esquina superior derecha para cerrar sesión.
- Navigation: Contiene el fichero XML que define la navegación de la aplicación. En este fichero se especifican los flujos entre fragmentos y se establece como destino inicial el fragmento DatosFragment.
- Values: Contiene ficheros XML que definen valores globales, como cadenas de texto (strings.xml), colores (colors.xml) o estilos (styles.xml), que son usados en toda la aplicación.

5.3 Configuración del Proyecto

La configuración necesaria para que el proyecto funcione correctamente se encuentra en los ficheros build.gradle y AndroidManifest.xml, en ellos se definen las dependencias del proyecto, las configuraciones, los permisos necesarios y todo lo necesario para el uso correcto de la aplicación.

5.3.1 Dependencias del build.gradle

Este archivo se encarga de definir las dependencias de la aplicación. Se especifican todas las librerías y versiones que se requieren, además de la configuración necesaria para construir la aplicación.

Entre las dependencias más relevantes para esta aplicación, podemos encontrar las siguientes librerías:

- com.android.volley:volley:1.2.1: Utilizada para realizar peticiones HTTP, en esta aplicación se usa para poder enviar las peticiones al servicio REST. Se ha elegido esta biblioteca debido a que permite realizar las peticiones de forma muy eficiente.
- com.google.code.gson:gson:2.8.8: Permite la serialización y deserialización de objetos Java JSON y viceversa. Es crucial para permitir una correcta comunicación entre el servicio REST y la aplicación.
- androidx.navigation:navigation-fragment:2.7.7: Facilita la navegación entre fragmentos.
- androidx.navigation:navigation-ui:2.7.7: Esta biblioteca es utilizada en conjunto con la anterior, para gestionar la navegación entre fragmentos basada en la interfaz de usuario.
- com.fatboyindustrial.gson-javatime-serialisers:gson-javatime-serialisers:1.1.0: Permite trabajar con objetos de tipo java.time en conjunto con Gson. Facilita la conversión de los objetos de tipo LocalDate y LocalDateTime, muy usados para gestionar las alarmas y eventos.

5.3.2 Permisos del AndroidManifest.xml

En el archivo AndroidManifest.xml se definen los permisos que requiere la aplicación para poder interactuar

con el sistema operativo del dispositivo móvil. El principal permiso necesario para esta aplicación es el de internet. Este permite a la aplicación hacer uso de conexiones a internet y es necesario para realizar las peticiones HTTP al servicio REST. En la figura 5-21 podemos ver un fragmento de este fichero donde se define este permiso.

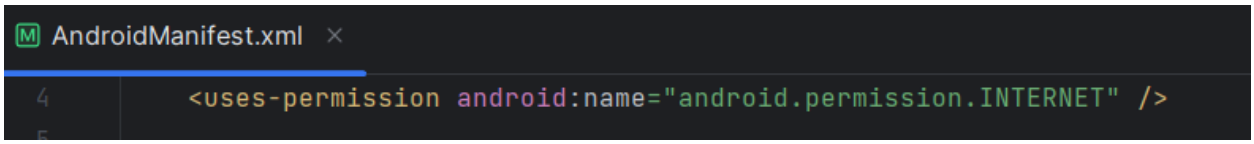

Figura 5-21. Fragmento AndroidManifest.xml de SensApp Control

6 APLICACIÓN SENSAPP PLUS

El hombre que mueve montañas empieza
cargando pequeñas piedras.

Confucio

La aplicación SensApp Plus está dirigida a dos tipos de usuarios. Por un lado, los administrativos, estos podrán ver y modificar los datos de usuarios, responsables y operadores, además de crear nuevos cuando sea necesario. También podrán acceder a toda la información correspondiente a las incidencias registradas en el sistema.

Por otro lado, los operadores, estos pueden consultar los datos de los usuarios, ver las incidencias que han atendido y atender las que aún no han sido resueltas.

Esta tercera aplicación se centra en la gestión de datos y en la resolución de incidencias. Para ofrecer esto la aplicación está compuesta por tres actividades principales, una dedicada al inicio de sesión y otras dos, cada una de estas funciona como contenedor de diferentes fragmentos que conforman los menús disponibles, uno para administrativos y otro para operadores.

En este capítulo se detallan las funcionales comunes a los dos tipos de usuarios y las específicas de cada uno de ellos, así como la estructura de ficheros del proyecto y los aspectos determinantes de la configuración.

En las siguientes figuras se representan los diagramas de caso de uso de la aplicación, en la figura 6-1 podemos ver los casos de uso para el actor administrativo y en la figura 6-2 para el operador.

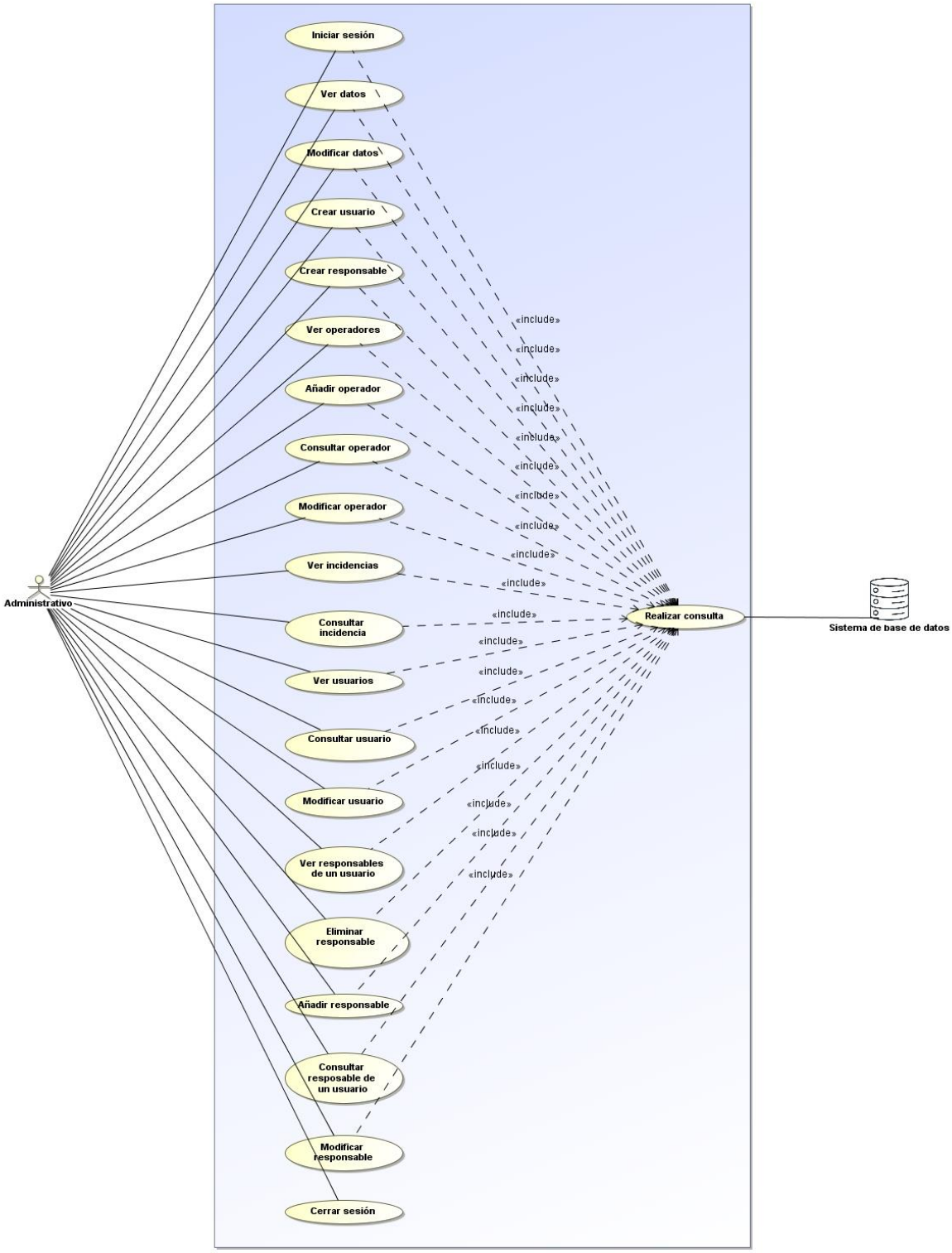

Figura 6-1. Casos de uso aplicación SensApp Plus (administrativos)[9]

[9] En el diagrama representado se han omitido las relaciones de extensión, por simplificación. Estas quedan reflejadas en las tablas de cada uno de los casos de usos.

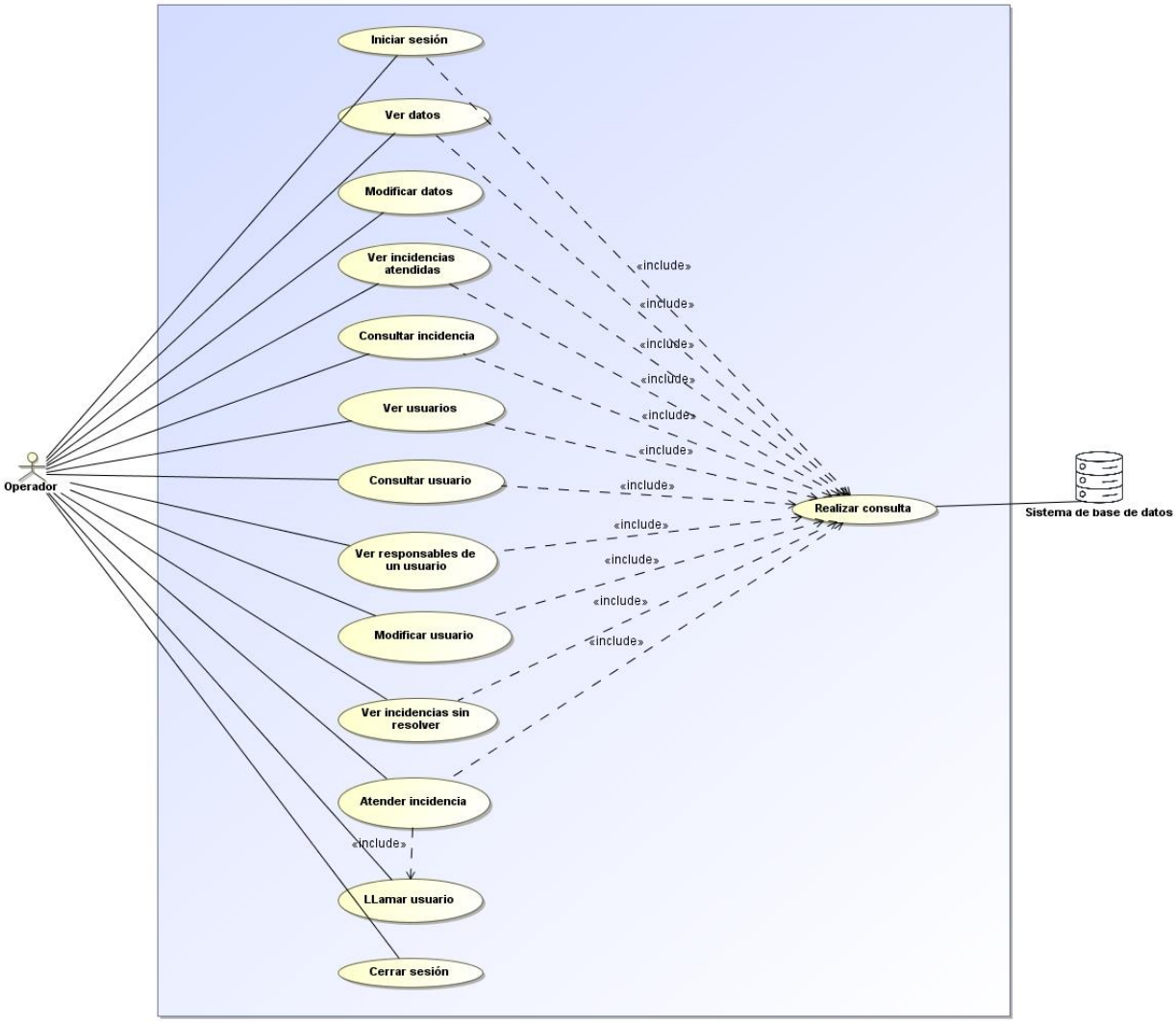

Figura 6-2. Casos de uso aplicación SensApp Plus (operadores)[10]

6.1 Funcionalidades comunes

Hay una serie de funcionalidades comunes a los dos tipos de usuarios que usan SensApp Plus por lo que se detallan en este apartado los casos de uso correspondientes a estas.

6.1.1 Inicio de sesión

La actividad que proporciona la funcionalidad de iniciar sesión es común para los administrativos y para los operadores. Su objetivo es ofrecer seguridad y garantizar el acceso a las demás opciones de la aplicación a los usuarios autorizados. La interfaz de esta actividad se muestra en la figura 6-3.

[10] En el diagrama representado se han omitido las relaciones de extensión, por simplificación. Estas quedan reflejadas en las tablas de cada uno de los casos de usos.

Figura 6-3. Inicio de sesión

6.1.1.1 CDU – Iniciar sesión

Este caso de uso es similar a los ya vistos en las aplicaciones SensApp y SensApp Control. Su objetivo es dar acceso al menú de administrativos o al de operadores según el tipo de usuario registrado que acceda a la aplicación. Se detalla en la tabla 6-1.

Tabla 6-1. CDU-3.01 – Iniciar sesión

<table>
<tr><td>CDU-3.01</td><td colspan="2">Iniciar sesión</td></tr>
<tr><td>Descripción</td><td colspan="2">El actor introduce sus credenciales para acceder a la aplicación. Si estas son correctas, si el actor es un administrativo podrá acceder al menú de administrativos, si es un operador podrá acceder al menú de operadores. Si las credenciales no son correctas, se muestra un mensaje de error.</td></tr>
<tr><td>Actor principal</td><td colspan="2">Administrativo y Operador</td></tr>
<tr><td>Precondición</td><td colspan="2">El actor que acceda a la aplicación debe estar registrado en el sistema.</td></tr>
<tr><td>Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>El actor introduce un nombre de usuario y una contraseña en la pantalla de inicio de sesión y pulsa el botón “Iniciar sesión”.</td></tr>
<tr><td></td><td>2</td><td>Se llama al CDU – 3.05 para realizar una petición al servicio REST que permite comprobar si existe el administrativo o el operador mediante una consulta a la BBDD.</td></tr>
<tr><td></td><td>3.1</td><td>Si los datos introducidos se corresponden con un administrativo registrado en el sistema, se da acceso al administrativo al menú principal de administrativos.</td></tr>
<tr><td></td><td>3.2</td><td>Si los datos introducidos se corresponden con un operador registrado</td></tr>
</table>

			en el sistema, se da acceso al operador al menú principal de operadores.
		3.2.1	En el caso de que el actor sea un operador, este se actualiza, para que su estado pase a ser activo, haciendo uso del CDU – 3.05 para realizar la petición.
Postcondición	El actor puede acceder a todas las funcionalidades disponibles en el menú que le corresponda según sea un administrativo o un operador.		
Excepciones	Paso	Acción	
	3	Si los datos no son correctos, se muestra un mensaje de error al actor.	

6.1.2 Datos

Esta es la primera sección tanto del menú de administrativos como del operador y es la que se muestra por defecto al acceder a la aplicación tras iniciar sesión. Desde ella el administrativo u operador puede consultar y modificar sus datos personales. Esto es fundamental para mantener actualizada la información personal relevante. En la figura 6-5 podemos ver la interfaz disponible para el administrativo y en la 6-4 la disponible para el operador, ambas tienen la misma estructura.

Figura 6-5. Datos del administrativo

Figura 6-4. Datos del operador

A continuación, se van a describir los casos de uso asociados a esta sección.

6.1.2.1 CDU – Ver datos

Este caso de uso tiene por objetivo permitir al administrativo o al operador consultar su información personal.

Tabla 6-2. CDU-3.02 – Ver datos

CDU-3.02	**Ver datos**

Descripción	Permite al actor consultar sus datos personales.
Actor principal	Administrativo y Operador
Precondición	El actor debe haber iniciado sesión con éxito (CDU – 3.01).
Secuencia normal	**Paso** / **Acción** 1 / El actor accede a la sección "Mis datos" del menú lateral. 2 / Se llama al CDU – 3.05 para realizar una petición al servicio REST que obtenga los datos del actor mediante una consulta a la BBDD. 2.1 / Los datos del actor se muestran en pantalla.
Postcondición	El actor puede consultar toda su información personal.

6.1.2.2 CDU – Modificar datos

Este caso de uso permite al administrativo o al operador actualizar algunos de sus datos personales de forma simple.

Tabla 6-3. CDU-3.03 – Modificar datos

CDU-3.03	**Modificar datos**
Descripción	Permite al actor actualizar algunos de sus datos personales (contraseña).
Actor principal	Administrativo y Operador
Precondición	El actor debe haber iniciado sesión con éxito (CDU – 3.01) y estar en la sección "Mis datos" del menú desplegable lateral (CDU – 3.02).
Secuencia normal	**Paso** / **Acción** 1 / El actor modifica el campo de texto 'contraseña' mostrado en pantalla. 2 / El actor pulsa el botón "Editar mis datos". 3 / Se llama al CDU – 3.05 para realizar una petición al servicio REST que actualice los datos del actor mediante una consulta a la BBDD. 3.1 / Los datos actualizados del actor se muestran en pantalla.
Postcondición	El actor puede ver toda su información personal actualizada.
Excepciones	Paso / Acción 2 / Si el campo de texto está vacío, se muestra un mensaje de error al actor, indicando que se debe rellenar el campo.

6.1.3 Cerrar sesión

La funcionalidad de cerrar sesión también es común a los dos tipos de usuarios. Esta permite volver a la actividad

de inicio de sesión. Está disponible desde el menú de la esquina superior derecha, tanto desde el menú para administrativos como desde el de operadores. Se puede ver en la figura 6-6.

Figura 6-6. Cerrar sesión

6.1.3.1 CDU – Cerrar sesión

En la tabla 6-4 se detalla este caso de uso.

Tabla 6-4. CDU-3.04 – Cerrar sesión

<table>
<tr><td>CDU-3.04</td><td colspan="2">Cerrar sesión</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al actor cerrar sesión en la aplicación.</td></tr>
<tr><td>Actor principal</td><td colspan="2">Administrativo y Operador</td></tr>
<tr><td>Precondición</td><td colspan="2">El actor debe haber iniciado sesión con éxito (CDU – 3.01).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El actor accede al menú superior derecho pulsando en los tres puntos que aparecen.</td></tr>
<tr><td>2</td><td>El actor pulsa la opción “Cerrar sesión”.</td></tr>
<tr><td>3</td><td>Se navega hasta la actividad encargada de iniciar sesión.</td></tr>
<tr><td>3.1</td><td>En el caso de que el actor sea un operador, este se actualiza, para que su estado pase a ser no activo, haciendo uso del CDU – 3.05 para realizar la petición.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El actor se encuentra en la actividad de inicio de sesión.</td></tr>
</table>

6.1.4 Petición al Servicio REST y consulta a la BBDD

6.1.4.1 CDU - Realizar consulta

Este caso de uso se aplica en la mayoría de los anteriores y en los siguientes, ya que permite realizar una consulta a la base de datos utilizada por el sistema para consultar, modificar, crear o eliminar algún dato. Se detalla en la tabla 6-5.

Tabla 6-5. CDU-3.05 – Realizar consulta

<table>
<tr><td>CDU-3.05</td><td colspan="2">Realizar consulta</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite realizar una consulta a la base de datos.</td></tr>
<tr><td>Actor principal</td><td colspan="2">Sistema de base de datos</td></tr>
<tr><td>Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>Se realiza una petición al servicio REST para obtener, modificar, crear o eliminar algún dato de la BBDD.</td></tr>
<tr><td></td><td>2</td><td>Se ejecuta la consulta SQL.</td></tr>
<tr><td></td><td>3</td><td>Se devuelve el resultado del método REST invocado.</td></tr>
<tr><td>Postcondición</td><td colspan="2">Se ha realizado una consulta a la BBDD.</td></tr>
<tr><td>Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>Si no hay conexión a internet no se podrá realizar la petición correctamente.</td></tr>
<tr><td></td><td>2</td><td>Si no hay acceso a la BBDD no se podrá realizar la consulta.</td></tr>
</table>

6.2 Funcionalidades Administrativos

En este apartado se detallan las funcionalidades específicas de los administrativos en la aplicación SensApp Plus, es decir, las correspondientes a los casos de uso representados en el diagrama de la figura 6-2. Como se ha explicado en la introducción del capítulo, la aplicación se basa en dos menús desde donde se accede a las distintas funcionalidades. Al menú correspondiente a los administrativos se accede después de haber iniciado sesión correctamente con los datos de un administrativo registrado en el sistema. Las secciones de este menú son cinco: Mis datos, Ver usuarios, Añadir usuarios, Ver operadores y Ver incidencias, como se puede ver en la figura 6-7.

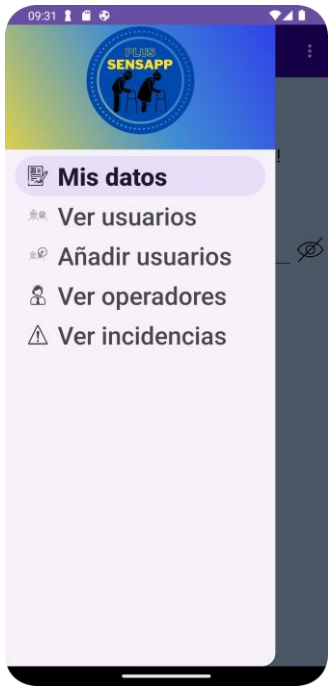

Figura 6-7. Menú desplegable lateral Administrativos

Todos los casos de uso de este menú tienen como actor principal a un administrativo y se detallan en los siguientes apartados, dividiéndolos según la sección del menú a la que pertenecen.

6.2.1 Ver usuarios

En esta sección se pueden consultar y modificar todos los usuarios registrados en el sistema. También se puede consultar y modificar los datos de los responsables asociados a cada usuario y añadir nuevos responsables. En la interfaz principal de esta sección también se permite filtrar la lista de usuarios según el DNI, escribiendo el número de DNI deseado en la barra de búsqueda superior, como se puede ver en la figura 6-8.

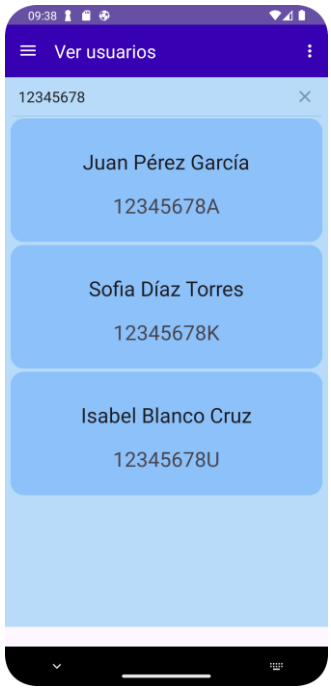

Figura 6-8. Ver usuarios

6.2.1.1 CDU – Ver usuarios

El primero de los casos de uso asociados a esta sección permite visualizar la lista de todos los usuarios registrados en el sistema, haciendo uso para ello de una lista paginada. Se hace uso de este caso desde los dos menús de la aplicación, por lo que tendrá dos actores: administrativo y operador. Se detalla en la tabla 6-6.

Tabla 6-6. CDU-3.06 – Ver usuarios

<table>
<tr><td>CDU-3.06</td><td colspan="2">Ver usuarios</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al actor consultar una lista con todos los usuarios existentes.</td></tr>
<tr><td>Actor principal</td><td colspan="2">Administrativo y Operador</td></tr>
<tr><td>Precondición</td><td colspan="2">El actor debe haber iniciado sesión con éxito (CDU – 3.01).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El actor accede a la sección “Ver usuarios” del menú lateral.</td></tr>
<tr><td>2</td><td>Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita obtener los usuarios mediante una consulta a la BBDD.</td></tr>
<tr><td>2.1</td><td>La lista de usuarios se muestra en pantalla.</td></tr>
<tr><td>3</td><td>El actor realiza scroll en la lista.</td></tr>
<tr><td>4</td><td>Cuando el actor llega hasta el final de la lista, se llama de nuevo al CDU – 3.05 para obtener la siguiente página de la lista mediante una petición al servicio REST que realiza la consulta correspondiente en la BBDD.</td></tr>
<tr><td>4.1</td><td>Los usuarios se añaden a la lista mostrada en pantalla, de forma ordenada.</td></tr>
<tr><td>5</td><td>El actor introduce un número de DNI en la barra de búsqueda.</td></tr>
<tr><td>6</td><td>Se filtra la lista mostrada en pantalla para mostrar solo los usuarios cuyo DNI contenga el introducido en la barra de búsqueda.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El actor puede ver todos los usuarios que coincidan con su criterio de búsqueda.</td></tr>
</table>

6.2.1.2 CDU – Consultar usuario

Este caso de uso tiene como objetivo permitir al administrativo ver los datos personales de un determinado usuario. Se hace uso de este caso desde los dos menús de la aplicación, por lo que tendrá dos actores: administrativo y operador. La interfaz usada en este menú se puede observar en las figuras 6-9 y 6-10, ya que para visualizar todos los elementos es necesario deslizar la pantalla.

Figura 6-10. Consultar usuario (1/2)

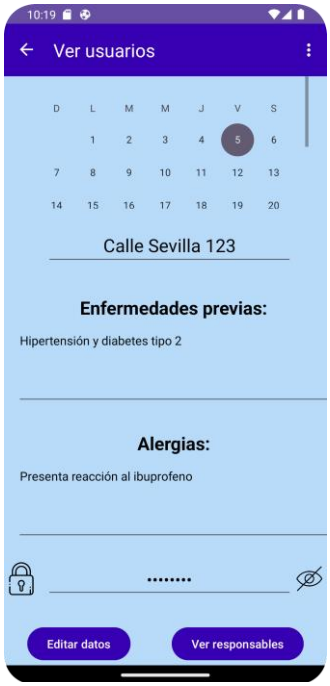

Figura 6-9. Consultar usuario (2/2)

Tabla 6-7. CDU-3.07 – Consultar usuario

<table>
<tr><td>CDU-3.07</td><td colspan="2">Consultar usuario</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al actor consultar los datos personales de un usuario.</td></tr>
<tr><td>Actor principal</td><td colspan="2">Administrativo y Operador</td></tr>
<tr><td>Precondición</td><td colspan="2">El actor debe haber iniciado sesión con éxito (CDU – 3.01) y estar en la sección “Ver usuarios” del menú desplegable lateral (CDU – 3.06).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El actor pulsa sobre el usuario que quiere consultar.</td></tr>
<tr><td>2</td><td>Se navega hasta un nuevo fragmento dedicado a ver los detalles de los usuarios, pasando el usuario seleccionado en el bundle de la navegación.</td></tr>
<tr><td>2.1</td><td>Se muestran los detalles del usuario en pantalla.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El actor puede ver los datos personales del usuario seleccionado en pantalla.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>2.1</td><td>Si el actor es un operador no se mostrará la información correspondiente a la contraseña del usuario.</td></tr>
</table></td></tr>
</table>

6.2.1.3 CDU – Modificar usuario

La función de este caso de uso es permitir al administrativo modificar algunos de los datos personales de un usuario en concreto.

Tabla 6-8. CDU-3.08 – Modificar usuario

CDU-3.08	**Modificar usuario**	
Descripción	Permite al administrativo modificar algunos de los datos de un usuario.	
Precondición	El administrativo debe haber iniciado sesión con éxito (CDU – 3.01), estar en la sección "Ver usuarios" del menú desplegable lateral (CDU – 3.06) y haber seleccionado algún usuario (CDU – 3.07).	
Secuencia normal	**Paso**	**Acción**
	1	El administrativo modifica uno o varios de los campos de texto disponibles en pantalla o el Date Picker para modificar algunos de los datos del usuario (nombre, apellidos, número de teléfono, domicilio, enfermedades previas, alergias y contraseña).
	2	El administrativo pulsa el botón "Editar datos".
	3	Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita actualizar el usuario mediante una consulta a la BBDD.
	3.1	Se muestran los datos actualizados del usuario en pantalla.
Postcondición	El administrativo puede ver los datos actualizados del usuario en pantalla.	
Excepciones	Paso	Acción
	2	Si alguno de los campos de texto que no puedan ser nulos está vacío (nombre, apellidos, número de teléfono, domicilio o contraseña), se muestra un mensaje de error al administrativo, indicando que se deben rellenar los campos.

6.2.1.4 CDU – Ver responsables de un usuario

También se ofrece la posibilidad de ver todos los responsables asociados a un usuario, como se puede ver en la figura 6-11. Este caso se usa en los dos menús de la aplicación, por lo que tendrá dos actores: administrativo y operador. Se detalla en la tabla 6-9

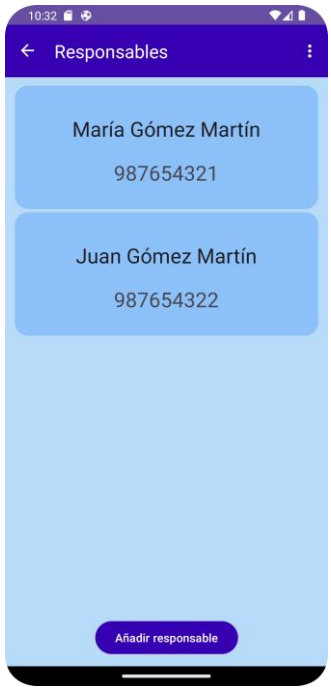

Figura 6-11. Ver responsables de un usuario

Tabla 6-9. CDU-3.09 – Ver responsables de un usuario

<table>
<tr><td>CDU-3.09</td><td colspan="2">Ver responsables de un usuario</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al actor consultar la lista de todos los responsables de un usuario.</td></tr>
<tr><td>Actor principal</td><td colspan="2">Administrativo y Operador</td></tr>
<tr><td>Precondición</td><td colspan="2">El actor debe haber iniciado sesión con éxito (CDU – 3.01), estar en la sección “Ver usuarios” del menú desplegable lateral (CDU – 3.06) y haber seleccionado algún usuario (CDU – 3.07).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El actor pulsa el botón “Ver responsables”.</td></tr>
<tr><td>2</td><td>Se navega hasta un nuevo fragmento dedicado a ver la lista de responsables de un usuario.</td></tr>
<tr><td>2.1</td><td>Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita obtener los responsables del usuario seleccionado mediante una consulta a la BBDD.</td></tr>
<tr><td>2.2</td><td>La lista de responsables se muestra en pantalla.</td></tr>
<tr><td>3</td><td>El actor realiza scroll en la lista.</td></tr>
<tr><td>4</td><td>Cuando el actor llega hasta el final de la lista, se llama de nuevo al CDU – 3.05 para obtener la siguiente página de la lista mediante una petición al servicio REST que realiza la consulta correspondiente en la BBDD.</td></tr>
<tr><td>4.1</td><td>Los responsables se añaden a la lista mostrada en pantalla, de forma</td></tr>
</table></td></tr>
</table>

	ordenada.	
Postcondición	El actor puede ver todos los responsables del usuario seleccionado.	

6.2.1.5 CDU – Consultar responsable de un usuario

Este caso de uso, detallado en la tabla 6-10, permite al administrativo ver los datos personales de un responsable de uno de los usuarios. La interfaz usada se puede observar en la figura 6-12.

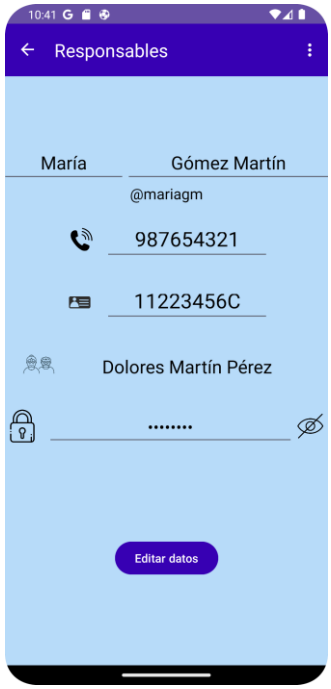

Figura 6-12. Consultar responsable de un usuario

Tabla 6-10. CDU-3.10 – Consultar responsable de un usuario

CDU-3.10	**Consultar responsable de un usuario**	
Descripción	Permite al administrativo consultar los datos personales de uno de los responsables asociados a un usuario en concreto.	
Precondición	El administrativo debe haber iniciado sesión con éxito (CDU – 3.01), estar en la sección "Ver usuarios" del menú desplegable lateral (CDU – 3.06), haber seleccionado algún usuario (CDU – 3.07) y haber seleccionado la opción de "Ver responsables" (CDU – 3.09).	
Secuencia normal	**Paso**	**Acción**
	1	El administrativo pulsa sobre el responsable que quiere consultar.
	2	Se navega hasta un nuevo fragmento dedicado a ver los detalles de los responsables, pasando el responsable seleccionado en el bundle de la navegación.
	2.1	Se muestran los detalles del responsable en pantalla.
Postcondición	El administrativo puede ver los datos personales del responsable seleccionado en pantalla.	

6.2.1.6 CDU – Modificar responsable

También es posible modificar algunos de los datos de los responsables, de esta función se encarga el caso de uso detallado en la tabla 6-11.

Tabla 6-11. CDU-3.11 – Modificar responsable

<table>
<tr><td>CDU-3.11</td><td colspan="2">Modificar responsable</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al administrativo modificar algunos de los datos de un usuario.</td></tr>
<tr><td>Precondición</td><td colspan="2">El administrativo debe haber iniciado sesión con éxito (CDU – 3.01), estar en la sección “Ver usuarios” del menú desplegable lateral (CDU – 3.06), haber seleccionado algún usuario (CDU – 3.07), haber seleccionado la opción de “Ver responsables” (CDU – 3.09) y haber seleccionado algún usuario en concreto (CDU – 3.10).</td></tr>
<tr><td>Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>El administrativo edita uno o varios de los campos de texto disponibles en pantalla para modificar algunos de los datos del usuario (nombre, apellidos, número de teléfono, DNI y contraseña).</td></tr>
<tr><td></td><td>2</td><td>El administrativo pulsa el botón “Editar datos”.</td></tr>
<tr><td></td><td>3</td><td>Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita actualizar el responsable mediante una consulta a la BBDD.</td></tr>
<tr><td></td><td>3.1</td><td>Se muestran los datos actualizados del responsable en pantalla.</td></tr>
<tr><td>Postcondición</td><td colspan="2">El administrativo puede ver los datos actualizados del responsable en pantalla.</td></tr>
<tr><td>Excepciones</td><td>Paso</td><td>Acción</td></tr>
<tr><td></td><td>2</td><td>Si alguno de los campos de texto está vacío, se muestra un mensaje de error al administrativo, indicando que se deben rellenar todos los campos.</td></tr>
</table>

6.2.1.7 CDU – Eliminar responsable

Se ofrece la posibilidad de eliminar los responsables de un usuario. Para evitar posibles errores se ha configurado un diálogo de confirmación para realizar la operación, como se puede ver en la figura 6-13.

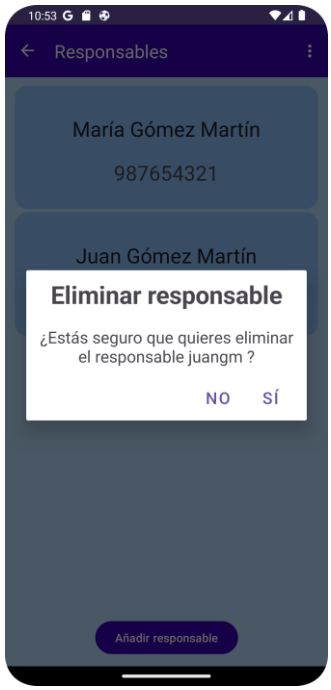

Figura 6-13. Eliminar responsable

Este caso de uso se detalla en la tabla 6-12.

Tabla 6-12. CDU-3.12 – Eliminar responsable

<table>
<tr><td>CDU-3.12</td><td colspan="2">Eliminar responsable</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al administrativo eliminar un determinado responsable de un usuario seleccionado.</td></tr>
<tr><td>Precondición</td><td colspan="2">El administrativo debe haber iniciado sesión con éxito (CDU – 3.01), estar en la sección “Ver usuarios” del menú desplegable lateral (CDU – 3.06), haber seleccionado algún usuario (CDU – 3.07) y haber seleccionado la opción de “Ver responsables” (CDU – 3.09).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El administrativo realiza un long click sobre el responsable de la lista que desee eliminar.</td></tr>
<tr><td>2</td><td>Se muestra un diálogo al administrativo para confirmar que se quiere eliminar el responsable.</td></tr>
<tr><td>3</td><td>El administrativo pulsa “Sí” en el diálogo.</td></tr>
<tr><td>4</td><td>Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita eliminar el responsable seleccionado mediante una consulta a la BBDD.</td></tr>
<tr><td>4.1</td><td>Se muestran la lista de responsables actualizada en pantalla.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El administrativo puede ver la lista de responsables actualizada.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table>
<tr><td>Paso</td><td>Acción</td></tr>
</table></td></tr>
</table>

	3	Si el administrativo pulsa “No”, el responsable no se elimina.

6.2.1.8 CDU – Añadir responsable

La última de las funcionalidades ofrecidas en esta sección es añadir un responsable asociado a un usuario seleccionado. Para esto se utiliza la interfaz mostrada en la figura 6-14.

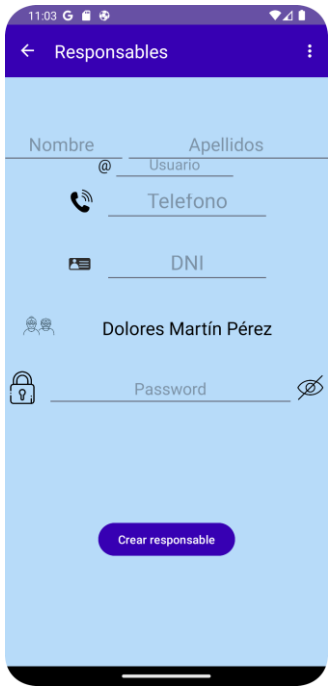

Figura 6-14. Añadir responsable

En la tabla 6-13 se detalla el caso de uso que ofrece esta funcionalidad.

Tabla 6-13. CDU-3.13 – Añadir responsable

CDU-3.13	**Añadir responsable**	
Descripción	Permite al administrativo añadir un nuevo responsable para un determinado usuario.	
Precondición	El administrativo debe haber iniciado sesión con éxito (CDU – 3.01), estar en la sección “Ver usuarios” del menú desplegable lateral (CDU – 3.06), haber seleccionado algún usuario (CDU – 3.07) y haber seleccionado la opción de “Ver responsables” (CDU – 3.09).	
Secuencia normal	**Paso**	**Acción**
	1	El administrativo pulsa el botón “Añadir responsable”.
	2	Se navega hasta un nuevo fragmento dedicado a esta funcionalidad.
	3	El administrativo rellena los campos de texto disponibles en pantalla.
	4	El administrativo pulsa el botón “Crear responsable”.
	5	Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita crear el responsable mediante una consulta a la BBDD.
	6	Se navega hasta el fragmento correspondiente a consultar un

		responsable (CDU – 3.10) y se muestran los datos del responsable creado en pantalla.
Postcondición	El administrativo puede ver los datos del responsable creado en pantalla.	
Excepciones	Paso	Acción
	4	Si alguno de los campos de texto se encuentra vacío, se muestra un mensaje de error al administrativo, indicando que se deben rellenar.
	4	Si ya existe un responsable con el nombre de usuario introducido, se muestra un mensaje de error al administrativo, indicando que ya existe un responsable con ese nombre de usuario.

6.2.2 Añadir usuarios

En esta sección se permite crear nuevos usuarios y crear responsables asociados a ellos. La interfaz que se utiliza para crear los usuarios se puede ver en las figuras 6-15 y 6-16.

Figura 6-16. Añadir usuarios (1/2)

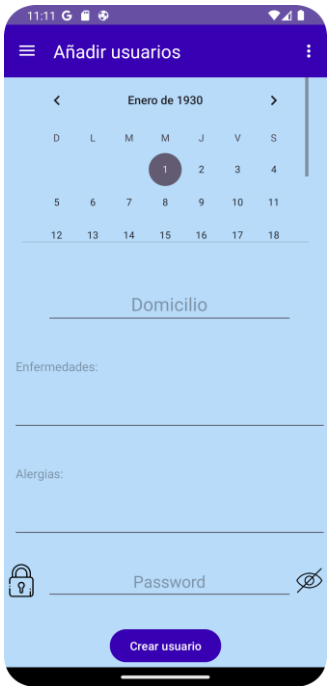

Figura 6-15. Añadir usuarios (2/2)

A continuación, se detallan los casos de uso asociados a esta sección.

6.2.2.1 CDU – Crear usuario

Este caso de uso, detallado en la tabla 6-14, ofrece la posibilidad de registrar un nuevo usuario en el sistema.

Tabla 6-14. CDU-3.14 – Crear usuario

CDU-3.14	**Crear usuario**
Descripción	Permite al administrativo registrar un nuevo usuario.

<table>
<tr><td>Precondición</td><td>El administrativo debe haber iniciado sesión con éxito (CDU – 3.01).</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El administrativo accede a la sección “Añadir usuario” del menú lateral.</td></tr>
<tr><td>2</td><td>Se navega hasta un nuevo fragmento dedicado a esta funcionalidad.</td></tr>
<tr><td>3</td><td>El administrativo rellena los campos de texto disponibles en pantalla.</td></tr>
<tr><td>4</td><td>El administrativo pulsa el botón “Crear usuario”.</td></tr>
<tr><td>5</td><td>Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita crear el usuario mediante una consulta a la BBDD.</td></tr>
<tr><td>6</td><td>Se muestra un diálogo al administrativo para consultar si quiere añadir un responsable.</td></tr>
<tr><td>6.1</td><td>El administrativo pulsa “Si”.</td></tr>
<tr><td>6.2</td><td>Se llama al CDU- 3.15 para crear un responsable.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>El administrativo puede añadir un responsable para el usuario creado.</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>4</td><td>Si alguno de los campos de texto que no puedan ser nulos está vacío (nombre, apellidos, número de teléfono, domicilio o contraseña), se muestra un mensaje de error al administrativo, indicando que se deben rellenar los campos.</td></tr>
<tr><td>4</td><td>Si ya existe un usuario con el DNI introducido, se muestra un mensaje de error al administrativo, indicando que ya existe un usuario con ese DNI.</td></tr>
<tr><td>6.1</td><td>Si el administrativo pulsa “No” se navega hasta el fragmento en el que se muestra la lista de usuarios existentes (CDU – 3.06).</td></tr>
</table>
</td></tr>
</table>

6.2.2.2 CDU – Crear responsable

Tras añadir un nuevo usuario se ofrece la posibilidad de añadir un responsable asociado a este usuario. Para ello, tras completar el registro del usuario se muestra el diálogo mostrado en la figura 6-17.

Figura 6-17. Diálogo para añadir un responsable

Después de añadir un responsable se muestra otro diálogo similar a este que, tras pulsar la opción "Si", permite añadir otro.

Este caso de uso, por tanto, ofrece la funcionalidad de crear uno o varios responsables asociados al nuevo usuario y se detalla en la tabla 6-15.

Tabla 6-15. CDU-3.15 – Crear responsable

CDU-3.15	**Crear responsable**
Descripción	Permite al administrativo crear uno o varios responsables para un nuevo usuario.
Precondición	El administrativo debe haber iniciado sesión con éxito (CDU – 3.01), estar en la sección "Añadir usuarios" del menú desplegable lateral (CDU – 3.06), haber registrado un nuevo usuario y haber indicado que se desea añadir un responsable en el diálogo correspondiente (CDU – 3.14).
Secuencia normal	**Paso** / **Acción** 1 / Se navega hasta un nuevo fragmento dedicado a crear un responsable. 2 / El administrativo rellena los campos de texto disponibles en pantalla. 3 / El administrativo pulsa el botón "Crear responsable". 4 / Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita crear el responsable mediante una consulta a la BBDD. 5 / Se muestra un diálogo al administrativo para consultar si quiere añadir otro responsable. 5.1 / El administrativo pulsa "No". 6 / Se navega hasta el fragmento correspondiente a consultar la lista de usuarios (CDU – 3.07).

Postcondición	El administrativo puede consultar una lista de los usuarios del sistema.
Excepciones	<table><tr><td>Paso</td><td>Acción</td></tr><tr><td>4</td><td>Si alguno de los campos de texto se encuentra vacío, se muestra un mensaje de error al administrativo, indicando que se deben rellenar.</td></tr><tr><td>4</td><td>Si ya existe un responsable con el nombre de usuario introducido, se muestra un mensaje de error al administrativo, indicando que ya existe un responsable con ese nombre de usuario.</td></tr><tr><td>5.1</td><td>Si se pulsa “Si” se repite el CDU – 3.15 para permitir añadir otro responsable.</td></tr></table>

6.2.3 Ver operadores

En esta sección se agrupan las funcionalidades relacionadas con la gestión de los operadores, es decir, visualizar los operadores existentes, modificarlos, eliminarlos o añadir nuevos. Estos casos de uso se detallan en los siguientes apartados.

6.2.3.1 CDU – Ver Operadores

Este caso de uso permite visualizar una lista paginada de todos los operadores existentes. La lista se ordena para visualizar primero los operadores que están activos y no están ocupados, representados en color verde, en segundo lugar se representan los activos y ocupados, en color azul, y por último los no activos, en color rojo. Esto permite a los administrativos conocer el estado de los operadores de una forma sencilla. También se permite filtrar la lista según el nombre de usuario de los operadores. Todo esto se puede ver en la figura 6-18.

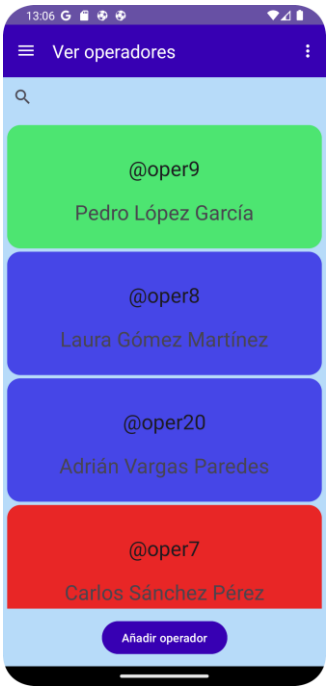

Figura 6-18. Ver operadores

El caso de uso correspondiente a esta funcionalidad se detalla en la tabla 6-16.

Tabla 6-16. CDU-3.16 – Ver operadores

CDU-3.16	**Ver operadores**

<table>
<tr><td>Descripción</td><td>Permite al administrativo consultar una lista de todos los operadores y consultar el estado en el que se encuentran de manera gráfica.</td></tr>
<tr><td>Precondición</td><td>El administrativo debe haber iniciado sesión con éxito (CDU – 3.01).</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El administrativo accede a la sección “Ver operadores” del menú lateral.</td></tr>
<tr><td>2</td><td>Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita obtener los operadores mediante una consulta a la BBDD.</td></tr>
<tr><td>2.1</td><td>La lista de operadores se muestra en pantalla.</td></tr>
<tr><td>3</td><td>El administrativo realiza scroll en la lista.</td></tr>
<tr><td>4</td><td>Cuando el administrativo llega hasta el final de la lista, se llama de nuevo al CDU – 3.05 para obtener la siguiente página de la lista mediante una petición al servicio REST que realiza la consulta correspondiente en la BBDD.</td></tr>
<tr><td>4.1</td><td>Los operadores se añaden a la lista mostrada en pantalla, de forma ordenada.</td></tr>
<tr><td>5</td><td>El administrativo introduce un nombre de usuario en la barra de búsqueda.</td></tr>
<tr><td>6</td><td>Se filtra la lista mostrada en pantalla para mostrar solo los operadores cuyo nombre de usuario contenga el introducido en la barra de búsqueda.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>El administrativo puede ver todos los operadores que coincidan con su criterio de búsqueda y el estado en el que se encuentran.</td></tr>
</table>

6.2.3.2 CDU – Consultar operador

Este caso de uso permite consultar los datos de un determinado operador. La interfaz utilizada se muestra en la figura 6-19.

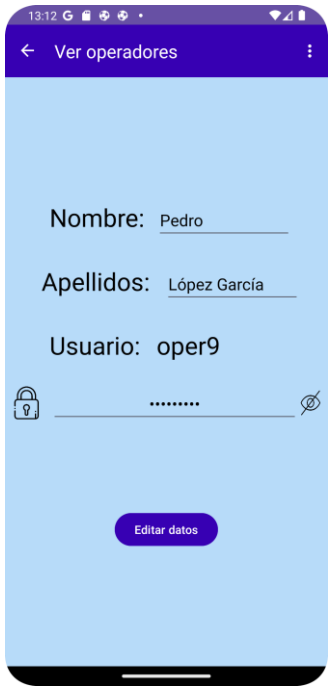

Figura 6-19. Consultar operador

Se detalla en la tabla 6-17.

Tabla 6-17. CDU-3.17 – Consultar operador

CDU-3.17	**Consultar operador**	
Descripción	Permite al administrativo consultar los datos personales de uno de los operadores.	
Precondición	El administrativo debe haber iniciado sesión con éxito (CDU – 3.01) y estar en la sección "Ver operadores" del menú desplegable lateral (CDU – 3.16).	
Secuencia normal	**Paso**	**Acción**
	1	El administrativo pulsa sobre el operador que quiere consultar.
	2	Se navega hasta un nuevo fragmento dedicado a ver los detalles de los operadores, pasando el operador seleccionado en el bundle de la navegación.
	2.1	Se muestran los detalles del operador en pantalla.
Postcondición	El administrativo puede ver los datos personales del operador seleccionado en pantalla.	

6.2.3.3 CDU – Modificar operador

También se permite modificar algunos de los datos de los operadores, esto se detalla en la tabla 6-18.

Tabla 6-18. CDU-3.18 – Modificar operador

CDU-3.18	**Modificar operador**
Descripción	Permite al administrativo modificar algunos de los datos de un operador.

<table>
<tr><td>Precondición</td><td>El administrativo debe haber iniciado sesión con éxito (CDU – 3.01), estar en la sección “Ver operadores” del menú desplegable lateral (CDU – 3.16) y haber seleccionado algún operador (CDU – 3.17).</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El administrativo modifica uno o varios de los campos de texto disponibles en pantalla para modificar algunos de los datos del usuario (nombre, apellidos y contraseña).</td></tr>
<tr><td>2</td><td>El administrativo pulsa el botón “Editar datos”.</td></tr>
<tr><td>3</td><td>Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita actualizar el operador mediante una consulta a la BBDD.</td></tr>
<tr><td>3.1</td><td>Se muestran los datos actualizados del operador en pantalla.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>El administrativo puede ver los datos actualizados del operador en pantalla.</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>2</td><td>Si alguno de los campos de texto está vacío, se muestra un mensaje de error al administrativo, indicando que se deben rellenar todos los campos.</td></tr>
</table>
</td></tr>
</table>

6.2.3.4 CDU – Añadir operador

Este caso de uso ofrece la posibilidad de registrar un nuevo operador en el sistema, haciendo uso para ello de la interfaz que se muestra en la figura 6-20. Se detalla en la tabla 6-19.

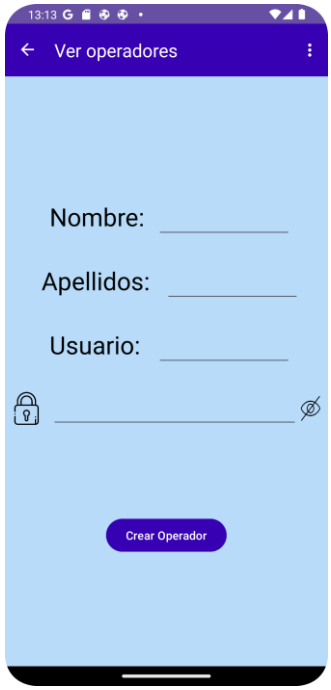

Figura 6-20. Añadir operador

Tabla 6-19. CDU-3.19 – Añadir operador

<table>
<tr><td>CDU-3.19</td><td colspan="2">Añadir operador</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al administrativo registrar un nuevo operador.</td></tr>
<tr><td>Precondición</td><td colspan="2">El administrativo debe haber iniciado sesión con éxito (CDU – 3.01) y estar en la sección “Ver operadores” del menú desplegable lateral (CDU – 3.16).</td></tr>
<tr><td>Secuencia normal</td><td colspan="2"><table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El administrativo pulsa el botón “Añadir operador”.</td></tr>
<tr><td>2</td><td>Se navega hasta un nuevo fragmento dedicado a esta funcionalidad.</td></tr>
<tr><td>3</td><td>El administrativo rellena los campos de texto disponibles en pantalla.</td></tr>
<tr><td>4</td><td>El administrativo pulsa el botón “Crear operador”.</td></tr>
<tr><td>5</td><td>Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita crear el operador mediante una consulta a la BBDD.</td></tr>
<tr><td>6</td><td>Se navega hasta el fragmento correspondiente a consultar el operador (CDU – 3.17) y se muestran los datos del operador creado en pantalla.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td colspan="2">El administrativo puede visualizar los datos del nuevo operador en pantalla.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table>
<tr><td>Paso</td><td>Acción</td></tr>
<tr><td>4</td><td>Si alguno de los campos de texto está vacío, se muestra un mensaje de error al administrativo, indicando que se deben rellenar todos los campos.</td></tr>
<tr><td>4</td><td>Si ya existe un operador con el nombre de usuario introducido, se muestra un mensaje de error al administrativo, indicando que ya existe un operador con ese nombre de usuario.</td></tr>
</table></td></tr>
</table>

6.2.4 Ver incidencias

La última sección del menú de los administrativos está dedicada a visualizar las incidencias reportadas. Se permite consultar una lista de todas ellas, donde se puede ver su estado según el color en el que se representan (rojo para incidencias no resueltas y verde para la resueltas) y consultar en detalle cada una de ellas. Además, la lista de incidencias se presenta en orden cronológico inverso y se permite filtrar según el operador que ha atendido cada incidencia. Esto se puede ver en la figura 6-21.

Figura 6-21. Ver incidencias

Los casos de uso se explican en los siguientes subapartados.

6.2.4.1 CDU – Ver incidencias

La finalidad de este caso de uso es permitir visualizar la lista de todas las incidencias reportadas. A continuación, se detalla en la tabla 6-20.

Tabla 6-20. CDU-3.20 – Ver incidencias

CDU-3.20	**Ver incidencias**	
Descripción	Permite al administrativo ver una lista de todas las incidencias reportadas.	
Precondición	El administrativo debe haber iniciado sesión con éxito (CDU – 3.01).	
Secuencia normal	**Paso**	**Acción**
	1	El administrativo accede a la sección “Ver incidencias” del menú lateral.
	2	Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita obtener las incidencias mediante una consulta a la BBDD.
	2.1	La lista de incidencias se muestra en pantalla.
	3	El administrativo realiza scroll en la lista.
	4	Cuando el administrativo llega hasta el final de la lista, se llama de nuevo al CDU – 3.05 para obtener la siguiente página de la lista mediante una petición al servicio REST que realiza la consulta correspondiente en la BBDD.
	4.1	Las incidencias se añaden a la lista mostrada en pantalla, de forma ordenada.

	5	El administrativo introduce el nombre de usuario de un operador en la barra de búsqueda.	
	5.1	Se filtra la lista mostrada en pantalla para mostrar solo las incidencias cuyo operador contenga el introducido en la barra de búsqueda.	
Postcondición	El administrativo puede consultar todas las incidencias que coincidan con su criterio de búsqueda.		

6.2.4.2 CDU – Consultar incidencia

En este caso de uso se ofrece la posibilidad de consultar los detalles de una incidencia en concreto. Este caso se usa en los dos menús, por lo que tendrá dos actores: administrativo y operador. Para implementarlo se usa la interfaz mostrada en la figura 6-22.

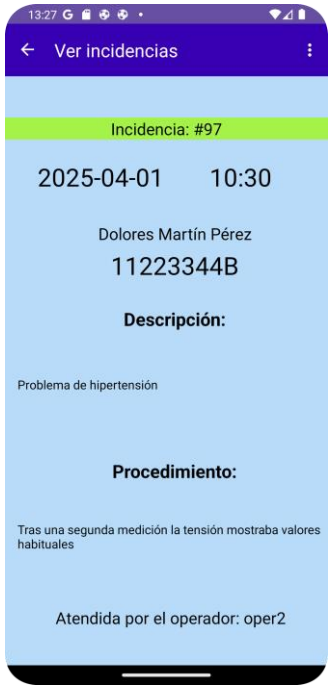

Figura 6-22. Consultar incidencia

En la tabla 6-21 se detalla el caso de uso.

Tabla 6-21. CDU-3.21 – Consultar incidencia

CDU-3.21	**Consultar incidencia**		
Descripción	Permite al actor consultar los detalles de una incidencia.		
Actor principal	Administrativo y Operador		
Precondición	El actor debe haber iniciado sesión con éxito (CDU – 3.01) y estar en la sección “Ver incidencias” o “Mis incidencias”, según corresponda, del menú desplegable lateral (CDU – 3.20).		
Secuencia normal	**Paso**	**Acción**	
	1	El actor pulsa sobre la incidencia de la lista que desee consultar.	

	2	Se navega hasta un nuevo fragmento dedicado a esta funcionalidad, pasando la incidencia seleccionada en el bundle de la navegación.	
	2.1	Se muestran los detalles de la incidencia en pantalla.	
Postcondición	El actor puede ver los detalles de la incidencia en pantalla.		

6.3 Funcionalidades Operadores

Las demás funcionalidades de la aplicación son las específicas de los operadores, es decir, las correspondientes a los casos de uso representados en el diagrama de la figura 6-2. Estos se dividen en varias secciones en el menú de los operadores: Mis datos, Mis incidencias, Ver usuarios y Atender, como se puede ver en la figura 6-23.

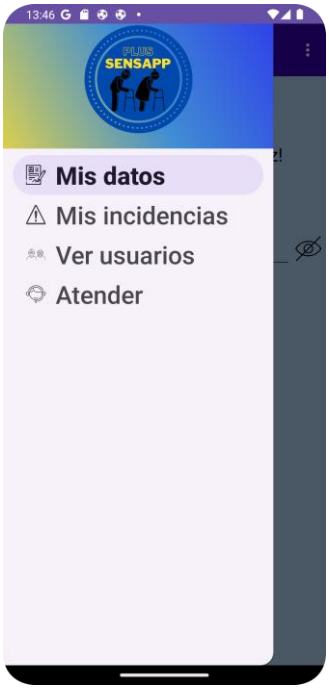

Figura 6-23. Menú desplegable lateral Operadores

En los casos de uso de este apartado el actor principal es un operador. Se detallan en los siguientes apartados, dividiéndolos según la sección del menú a la que pertenecen.

6.3.1 Incidencias

En esta sección se permite al operador consultar una lista de las incidencias que han sido atendidas por él. Se le permite filtrar según el DNI del usuario al que corresponde la incidencia, escribiéndolo en la barra de búsqueda, como se puede ver en la figura 6-24.

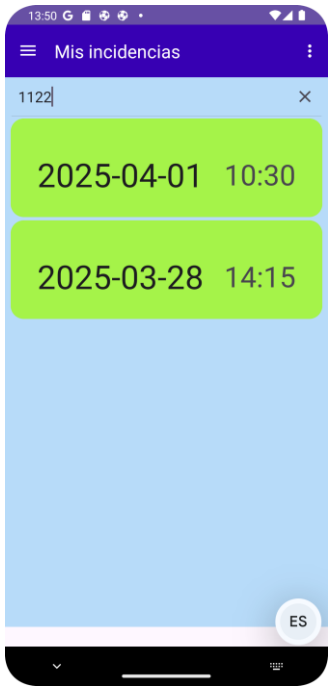

Figura 6-24. Mis incidencias

6.3.1.1 CDU – Ver incidencias atendidas

En este primer caso de uso de la sección se permite visualizar la lista de incidencias atendidas por el operador. Se detalla en la tabla 6-22.

Tabla 6-22. CDU-3.22 – Ver incidencias atendidas

CDU-3.22	**Ver incidencias atendidas**	
Descripción	Permite al operador ver una lista de todas las incidencias que han sido atendidas por él.	
Precondición	El operador debe haber iniciado sesión con éxito (CDU – 3.01).	
Secuencia normal	**Paso**	**Acción**
	1	El operador accede a la sección “Mis incidencias” del menú lateral.
	2	Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita obtener las incidencias del operador mediante una consulta a la BBDD.
	2.1	La lista de incidencias se muestra en pantalla.
	3	El operador realiza scroll en la lista.
	4	Cuando el operador llega hasta el final de la lista, se llama de nuevo al CDU – 3.05 para obtener la siguiente página de la lista mediante una petición al servicio REST que realiza la consulta correspondiente en la BBDD.
	4.1	Los incidencias se añaden a la lista mostrada en pantalla, de forma ordenada.

	5	El operador introduce un DNI en la barra de búsqueda.	
	5.1	Se filtra la lista mostrada en pantalla para mostrar solo las incidencias cuyo usuario tenga un DNI que contenga el introducido en la barra de búsqueda.	
Postcondición	El operador puede consultar todas las incidencias que coincidan con su criterio de búsqueda.		

6.3.1.2 CDU – Consultar incidencia

En este caso de uso se corresponde con el explicado anteriormente en las opciones del administrativo, CDU – 3.21 – Consultar incidencia. Permite consultar los detalles de una incidencia concreta pulsando en la lista que se muestra en el CDU – 3.22 – Ver incidencias atendidas.

6.3.2 Ver usuarios

Esta sección reúne todas las funcionalidades acerca de los usuarios del sistema disponibles desde este menú. Permite consultar una lista de todos los usuarios registrados en el sistema, modificar algunos de los datos de estos usuarios y consultar una lista con los nombres y números de teléfonos de sus responsables.

En la interfaz principal se permite filtrar la lista de usuarios según su DNI, escribiendo el número de DNI buscado en la barra de búsqueda superior, esto se puede observar en la figura 6-25.

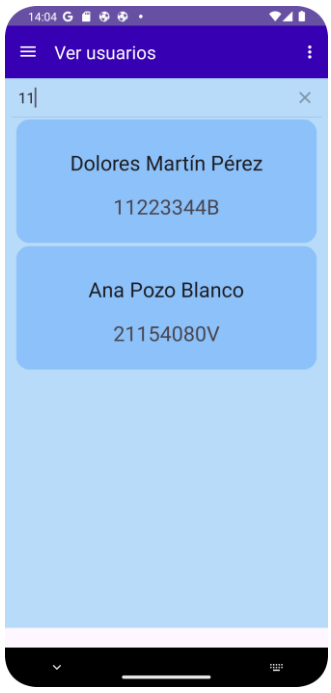

Figura 6-25. Ver usuarios

6.3.2.1 CDU – Ver usuarios

Este caso de uso se corresponde con el explicado anteriormente en las opciones del administrativo, CDU – 3.06 – Ver usuarios. Permite consultar la lista de todos los usuarios registrados en el sistema, haciendo uso de una lista paginada.

6.3.2.2 CDU – Consultar usuario

Del mismo modo que antes, este caso de uso se corresponde con su análogo del menú de administrativos, CDU – 3.07 – Consultar usuario. Permite consultar los datos personales de un usuario concreto, con la pequeña

diferencia de que en esta ocasión no se representa la contraseña del usuario.

6.3.2.3 CDU – Modificar usuario

Este caso de uso es muy similar al CDU – 3.08 – Modificar usuario, ambos permiten modificar algunos de los datos de un usuario en concreto. La diferencia es que en este solo se puede modificar los datos correspondientes a las enfermedades previas y a las alergias del usuario. Se detalla a continuación en la tabla 6-23.

Tabla 6-23. CDU-3.23 – Modificar usuario

<table>
<tr><td>CDU-3.23</td><td colspan="2">Modificar usuario</td></tr>
<tr><td>Descripción</td><td colspan="2">Permite al operador modificar algunos de los datos de un usuario.</td></tr>
<tr><td>Precondición</td><td colspan="2">El operador debe haber iniciado sesión con éxito (CDU – 3.01), estar en la sección “Ver usuarios” del menú desplegable lateral (CDU – 3.06) y haber seleccionado algún usuario (CDU – 3.07).</td></tr>
<tr><td>Secuencia normal</td><td>Paso</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>El operador modifica uno o los dos campos de texto disponibles en pantalla para modificar algunos de los datos del usuario (enfermedades previas y alergias).</td></tr>
<tr><td></td><td>2</td><td>El operador pulsa el botón “Editar datos”.</td></tr>
<tr><td></td><td>3</td><td>Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita actualizar el usuario mediante una consulta a la BBDD.</td></tr>
<tr><td></td><td>3.1</td><td>Se muestran los datos actualizados del usuario en pantalla.</td></tr>
<tr><td>Postcondición</td><td colspan="2">El operador puede ver los datos actualizados del usuario en pantalla.</td></tr>
</table>

6.3.2.4 CDU – Ver responsables de un usuario

Este caso de uso se corresponde con el explicado en las opciones del administrativo, CDU – 3.09 – Ver responsables de un usuario. Permite consultar la lista con los nombres y números de teléfono de todos los responsables asociados a un usuario.

6.3.3 Atender

Esta última sección del menú está dedicada a atender las incidencias que se reportan en el sistema. Cuenta con dos interfaces, una en la que se puede ver una lista de todas las incidencias no resueltas existentes que actualmente no tienen ningún operador asignado, y otra en la que se puede resolver una incidencia. Si un operador tiene asignada una incidencia sin resolver solo podrá acceder a esta segunda interfaz. En la figura 6-26 se muestra la primera interfaz y en las figuras 6-27 y 6-28, la otra.

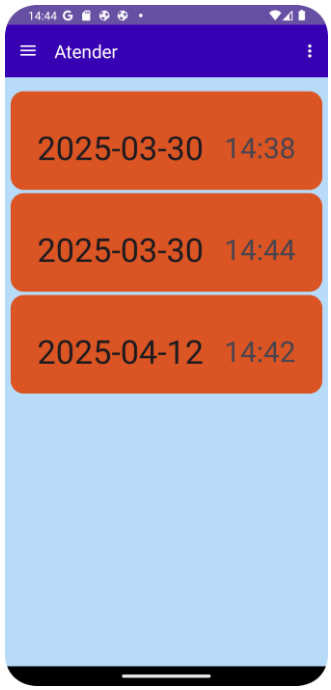

Figura 6-26. Ver incidencia sin resolver

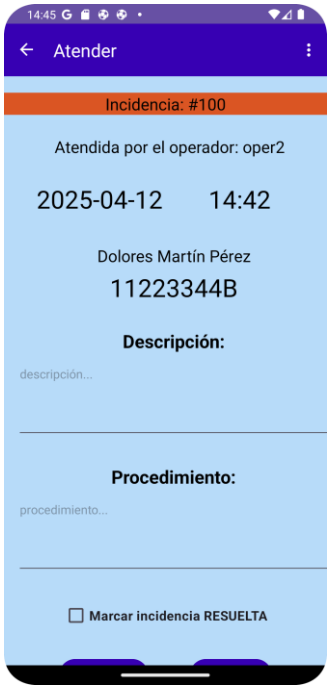

Figura 6-28. Atender incidencia (1/2)

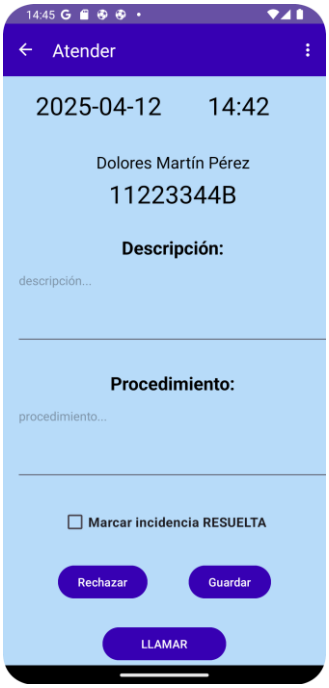

Figura 6-27. Atender incidencia (2/2)

En los siguientes subapartados se detallan los casos de uso existentes para ofrecer a los operadores la funcionalidad de atender incidencias.

6.3.3.1 CDU – Ver incidencias sin resolver

Este caso de uso permite a los operadores consultar la lista de incidencias sin resolver que aún no tienen asignado ningún operador. Esta lista se muestra en orden cronológico, para que las incidencias con más tiempo de evolución se atiendan antes. En la tabla 6-24 se detalla el caso.

Tabla 6-24. CDU-3.24 – Ver incidencias sin resolver

CDU-3.24	**Ver incidencias sin resolver**	
Descripción	Permite al operador ver una lista de todas las incidencias sin resolver y sin operador asignado.	
Precondición	El operador debe haber iniciado sesión con éxito (CDU – 3.01).	
Secuencia normal	**Paso**	**Acción**
	1	El operador accede a la sección "Atender" del menú lateral.
	2	Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita obtener las incidencias sin resolver y sin operador mediante una consulta a la BBDD.
	2.1	La lista de incidencias se muestra en pantalla.
	3	El operador realiza scroll en la lista.
	4	Cuando el operador llega hasta el final de la lista, se llama de nuevo al CDU – 3.05 para obtener la siguiente página de la lista mediante una petición al servicio REST que realiza la consulta correspondiente en la BBDD.
	4.1	Las incidencias se añaden a la lista mostrada en pantalla, de forma ordenada.
Postcondición	El operador puede consultar todas las incidencias sin resolver que aún no tienen un operador asignado.	
Excepciones	Paso	Acción
	1	Si el operador tiene asignada una incidencia sin resolver cuando accede a este menú se llamará directamente al CDU – 2.25 – Atender incidencia.

6.3.3.2 CDU – Atender incidencia

Este caso de uso ofrece la funcionalidad para atender una incidencia. Se detalla en la tabla 6-25.

Tabla 6-25. CDU-3.25 – Atender incidencia

CDU-3.25	**Atender incidencia**	
Descripción	Permite al operador atender una incidencia sin resolver y sin operador asignado.	
Precondición	El operador debe haber iniciado sesión con éxito (CDU – 3.01) y estar en la sección "Atender" del menú lateral (CDU – 3.24 – Ver incidencias sin resolver).	
Secuencia normal	**Paso**	**Acción**
	1	El operador pulsa la incidencia que desea atender.
	1.1	Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita actualizar el operador de la incidencia mediante una

		consulta a la BBDD.
	1.2	Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita actualizar el estado del operador mediante una consulta a la BBDD.
	2	Se navega hasta un nuevo fragmento dedicado a atender la incidencia.
	3	El operador pulsa el botón "LLAMAR".
	3.1	Se llama al CDU – 3.26 – Llamar usuario.
	4	El operador rellena los campos de texto mostrados en pantalla y marca la casilla para actualizar la incidencia como resuelta.
	5	El operador pulsa el botón "Guardar".
	5.1	Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita actualizar la incidencia mediante una consulta a la BBDD.
	5.2	Se llama al CDU – 3.05 para realizar una petición al servicio REST que permita actualizar el estado del operador mediante una consulta a la BBDD.
	6	Se navega hasta el fragmento correspondiente al CDU – 3.24 – Ver incidencias sin resolver.
Postcondición	El operador puede consultar todas las incidencias sin resolver que aún no tienen un operador asignado.	
Excepciones	Paso	Acción
	1	Si el operador que accede ya tiene una incidencia asignada se pasa directamente al paso 2.
	1	Si la incidencia seleccionada ya ha sido elegida por otro operador se muestra un diálogo que indique esto al operador (ver figura 6-29) y, tras esto, se muestra la lista actualizada (CDU – 3.24 – Ver lista de incidencias sin resolver).
	5	Si la casilla para actualizar la incidencia como completada no está marcado se actualizarán los datos de la incidencia pero no se marcará como completa, no se liberará al operador y no se realizará ninguna navegación.
	5	Si el operador pulsa el botón "Rechazar" la incidencia se actualizará con los datos introducidos en pantalla pero sin operador asignado, también se liberará el operador y se navegará al fragmento correspondiente al CDU – 3.24 – Ver incidencias sin resolver.

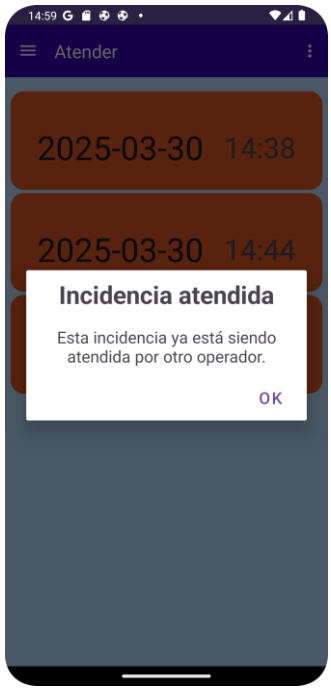

Figura 6-29. Diálogo incidencia atendida

6.3.3.3 CDU – Llamar usuario

Este caso de uso permite al operador realizar una llamada al número de teléfono asociado al usuario del cual está atendiendo una incidencia. La primera vez que se quiera usar esta opción será necesario conceder los permisos necesarios para poder realizarlo, como se observa en la figura 6-30. En la figura 6-31 podemos ver la realización de una llamada.

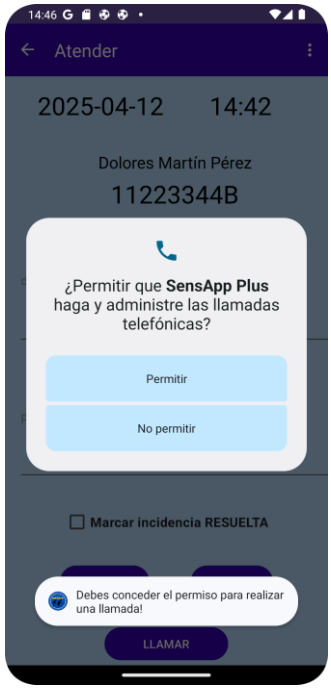

Figura 6-30. Permisos llamada

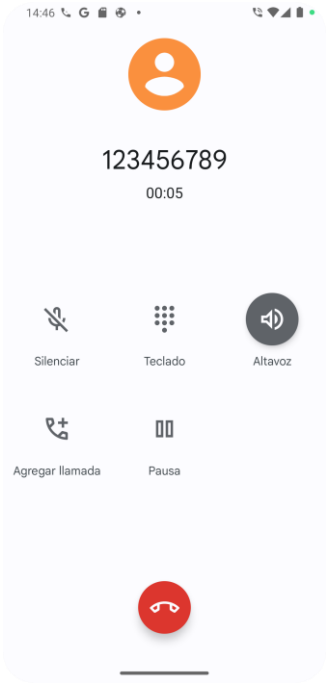

Figura 6-31. Llamar usuario

En la tabla 6-26 se detalla el caso de uso.

Tabla 6-26. CDU-3.26 – Llamar usuario

<table>
<tr><td>CDU-3.26</td><td>Llamar usuario</td></tr>
<tr><td>Descripción</td><td>Permite al operador realizar una llamada al número de teléfono del usuario.</td></tr>
<tr><td>Precondición</td><td>El operador debe haber iniciado sesión con éxito (CDU – 3.01) y estar atendiendo una incidencia (CDU – 3.25).</td></tr>
<tr><td>Secuencia normal</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>El operador pulsa el botón “LLAMAR”.</td></tr>
<tr><td>2</td><td>Se navega a la aplicación “Teléfono” y se realiza automáticamente una llamada al número de teléfono que tenga configurado el usuario al que pertenece la incidencia que está siendo atendida.</td></tr>
<tr><td>3</td><td>Tras finalizar la llamada, se navega al fragmento correspondiente al CDU – 3.25 – Atender incidencia.</td></tr>
</table>
</td></tr>
<tr><td>Postcondición</td><td>El operador ha realizado una llamada al usuario correspondiente.</td></tr>
<tr><td>Excepciones</td><td>
<table>
<tr><th>Paso</th><th>Acción</th></tr>
<tr><td>1</td><td>La primera vez que se haga uso de esta opción será necesario permitir que SensApp Plus haga y administre las llamadas telefónicas (ver figura 6-31).</td></tr>
<tr><td>2</td><td>Si no se conceden los permisos necesarios no se realizará ni la navegación ni la llamada, se mostrará un mensaje de error al operador indicando que es necesario conceder los permisos para realizar la llamada.</td></tr>
</table>
</td></tr>
</table>

6.4 Estructura de ficheros de la aplicación

En este apartado se describe la estructura de ficheros de SensApp Plus y se explica el contenido de sus principales directorios.

Los principales ficheros de código y de configuración de la aplicación se encuentran en el directorio src/main/. En esta carpeta podemos encontrar también el fichero de configuración AndroidManifest.xml, que contiene la configuración necesaria para el correcto funcionamiento de la aplicación.

También podemos encontrar en este dos grandes directorios, java/com/example/apptrabajadores/, donde se almacena todo el código fuente, y res/, donde se almacenan los diferentes recursos. Estos se explican en los siguientes subapartados.

6.4.1 Directorio del código fuente de la aplicación

En este directorio podemos encontrar el código fuente de la aplicación. En sus ficheros se define toda la lógica de las funcionalidades descritas en apartados anteriores y de toda la aplicación. En la figura 6-32 podemos ver su estructura.

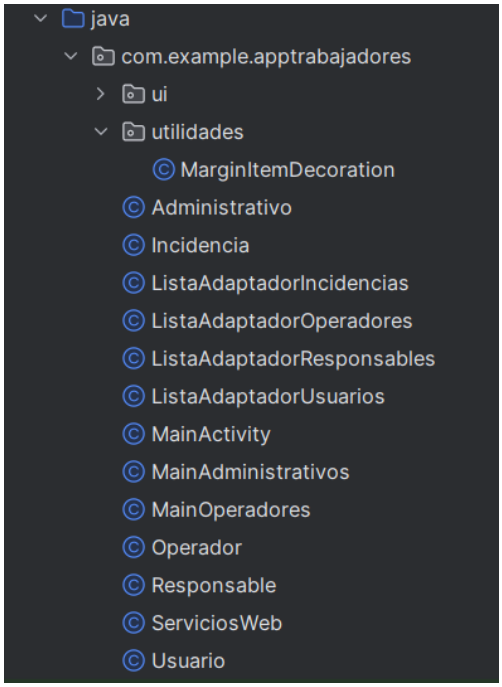

Figura 6-32. Estructura del directorio java/com/example/apptrabajadores/

Los principales archivos de este directorio son los siguientes:

- Administrativo.java, Incidencia.java, Operador.java, Responsable.java y Usuario.java: JavaBeans utilizados para representar las entidades Administrativo, Incidencia, Operador, Responsable y Usuario, respectivamente, en la aplicación. Todas contienen los atributos y métodos getter y setter correspondientes según lo visto en el capítulo 3 de este documento.
- ListaAdaptadorIncidencias.java, ListaAdaptadorOperadores.java, ListaAdaptadorResponsables.java y ListaAdaptadorUsuarios.java: Clases encargadas de adaptar la visualización de una lista de incidencias, operadores, responsables o usuarios, respectivamente, para poder ofrecer al usuario una interfaz gráfica más atractiva y simple.
- ServiciosWeb.java: Clase encargada de gestionar una única instancia de la cola de peticiones al servidor REST mediante el patrón Singleton. Esta clase permite realizar las peticiones de manera centralizada, lo que garantiza que solo existirá una cola de solicitudes.
- MainActivity.java: Actividad encargada de gestionar el inicio de sesión de los administrativos y operadores y de redirigirlos, en caso de éxito, a la pantalla principal del menú que les corresponda (CDU

-3.01 – Iniciar sesión).

- MainAdministrativos.java: Actividad principal del menú de administrativos de la aplicación. Esta funciona como contenedor de los fragmentos que ofrecen las funcionalidades disponibles para los administrativos. También es la encargada de ofrecer la funcionalidad de cerrar sesión (CDU - 3.04 – Cerrar sesión).
- MainOperadores.java: Actividad principal del menú de operadores de la aplicación. Esta funciona como contenedor de los fragmentos que ofrecen las funcionalidades disponibles para los operadores. También es la encargada de ofrecer la funcionalidad de cerrar sesión (CDU - 3.04 – Cerrar sesión).
- Utilidades/MarginItemDecoration.java: Clase encargada de agregar márgenes personalizados entre los elementos de un RecyclerView, mejorando la apariencia de las listas mostradas en pantalla.

El directorio ui/ contiene los componentes de la interfaz de usuario. Se organizan en subdirectorios según la funcionalidad que ofrece. Cada uno de ellos contiene los fragmentos que permiten ofrecer las funcionalidades descritas en el apartado anterior. La estructura de este directorio se puede observar en la figura 6-33.

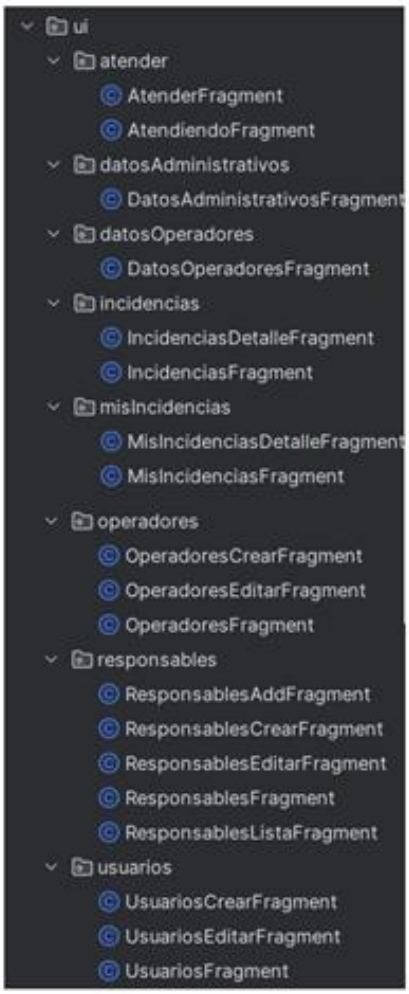

Figura 6-33. Estructura del directorio ui/

- atender/: Contiene los fragmentos relacionados con la funcionalidad de atender incidencias:
    - AtenderFragment.java: Fragmento encargado de ofrecer la funcionalidad de consultar una lista de las incidencias sin resolver (CDU - 3.24 – Ver lista de incidencias sin resolver).
    - AtendiendoFragment.java: Fragmento encargado de ofrecer la funcionalidad de atender una incidencia y de llamar a un usuario (CDU - 3.25 – Atender incidencia y CDU – 3.26 – Llamar usuario).
- datosAdministrativos/: Contiene el fragmento relacionado con la gestión de los datos personales del

administrativo:

  - DatosAdministrativosFragment.java: Fragmento que maneja la visualización y la actualización de los datos del administrativo (CDU - 3.02 – Ver datos y CDU - 3.03 – Modificar datos).

- datosOperadores/: Contiene el fragmento relacionado con la gestión de los datos personales del operador:
  - DatosOperadoresFragment.java: Fragmento que maneja la visualización y la actualización de los datos del operador (CDU - 3.02 – Ver datos y CDU - 3.03 – Modificar datos).
- incidencias/: Contiene los fragmentos relacionados con la visualización de todas las incidencias.
  - IncidenciasDetalleFragment.java: Fragmento que ofrece la posibilidad de consultar los datos de una incidencia en concreto (CDU - 3.21 – Consultar incidencia).
  - IncidenciasFragment.java: Fragmento que ofrece la posibilidad de consultar la lista de todas las incidencias (CDU - 3.20 – Ver incidencias).
- misIncidencias/: Contiene los fragmentos relacionados con la visualización de las incidencias asociadas a un operador:
  - MisIncidenciasDetalleFragment.java: Fragmento que ofrece la posibilidad de consultar los datos de una incidencia en concreto (CDU - 3.21 – Consultar incidencia).
  - MisIncidenciasFragment.java: Fragmento que ofrece la posibilidad de consultar la lista de incidencias de un operador en concreto (CDU - 3.22 – Ver incidencias atendidas).
- operadores/: Contiene los fragmentos relacionado con la gestión de los operadores:
  - OperadoresCrearFragment.java: Fragmento que ofrece la posibilidad de crear nuevos operadores (CDU - 3.19 – Añadir operador).
  - OperadoresEditarFragment.java: Fragmento que maneja la visualización y la actualización de los datos de un operador (CDU - 3.17 – Consultar operador y CDU - 3.18 – Modificar operador).
  - OperadoresFragment.java: Fragmento que ofrece la posibilidad de consultar la lista de todos los operadores (CDU - 3.16 – Ver operadores).
- responsables/: Contiene los fragmentos relacionados con la gestión de los responsables:
  - ResponsablesAddFragment.java y ResponsablesCrearFragment.java: Fragmentos que ofrecen la posibilidad de crear nuevos responsables para un determinado usuario (CDU - 3.13 – Añadir responsable y CDU – 3.15 – Crear responsable).
  - ResponsablesEditarFragment.java: Fragmento que maneja la visualización y la actualización de los datos de un responsable (CDU – 3.10 – Consultar responsable de un usuario y CDU – 3.11 – Modificar responsable).
  - ResponsablesFragment.java y ResponsablesListaFragment.java: Estos fragmentos ofrecen la posibilidad de consultar la lista de todos los responsables existentes de un usuario y de eliminar uno en concreto (CDU - 3.09 – Ver responsables de un usuario y CDU – 3.12 – Eliminar responsable).
- usuarios/: Contiene los fragmentos relacionados con la gestión de los usuarios.
  - UsuariosCrearFragment.java: Fragmento que ofrece la posibilidad de crear nuevos usuarios (CDU - 3.14 – Crear usuario).
  - UsuariosEditarFragment.java: Fragmento que maneja la visualización y la actualización de los datos de un usuario (CDU - 3.07 – Consultar usuario, CDU - 3.08 – Modificar usuario y CDU – 3.23 – Modificar usuario).
  - UsuariosFragment.java: Fragmento que ofrece la posibilidad de consultar la lista de todos los usuarios (CDU - 3.06 – Ver usuarios).

6.4.2 Directorio de recursos

En este directorio se almacenan los recursos usados por la aplicación. Se dividen en subdirectorios según el tipo de recurso que se almacena dentro. La estructura de este se puede ver en la figura 6-34.

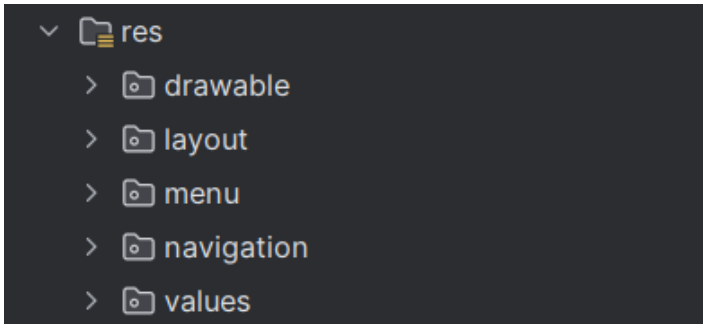

Figura 6-34. Estructura del directorio res/

Cada una de estas carpetas almacena recursos de un tipo concreto, como ya se ha explicado en capítulos anteriores.

6.5 Configuración del Proyecto

Como se explicaba en los capítulos 4 y 5 de este documento, la configuración necesaria para que el proyecto funcione correctamente se encuentra en los ficheros build.gradle y AndroidManifest.xml, en ellos se definen las dependencias del proyecto, las configuraciones, los permisos necesarios y todo lo necesario para el uso correcto de la aplicación.

6.5.1 Dependencias del build.gradle

En este fichero se definen las dependencias de la aplicación, se especifican las librerías y versiones que se requieren y la configuración necesaria para construir la aplicación.

Las dependencias más relevantes para esta aplicación, son las mismas que en la aplicación SensApp Control, es decir, las librerías explicadas en el apartado Dependencias del build.gradle del capítulo 5 de este documento.

6.5.2 Permisos del AndroidManifest.xml

En el archivo AndroidManifest.xml se definen los permisos que requiere la aplicación para poder interactuar con el sistema operativo del dispositivo móvil. Los principales permisos necesarios para esta aplicación son los siguientes:

- Internet: Este permiso permite a la aplicación hacer uso de conexiones a internet. Es necesario para realizar las peticiones HTTP al servicio REST.
- Call_phone: Este permiso permite a la aplicación hacer y gestionar llamadas telefónicas. No es suficiente con declarar el permiso aquí, será necesario que el usuario que use la aplicación conceda el permiso de forma manual la primera vez que se haga uso de una función que lo requiera.

En la figura 6-35 podemos ver un fragmento de este fichero donde se definen los permisos.

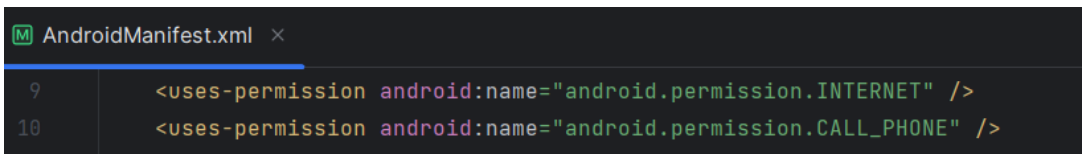

Figura 6-35. Fragmento AndroidManifest.xml de SensApp

7 CONCLUSIONES Y LÍNEAS FUTURAS

La mejor manera de predecir el futuro es crearlo.

Peter Drucker

Para concluir este documento, se detallan en este último capítulo los hitos alcanzados durante la realización del proyecto, junto con un análisis sobre su impacto y las posibles futuras líneas de mejora del sistema, tanto a nivel funcional como en términos de seguridad del sistema. Este Trabajo Fin de Grado, no solo representa el fin de una importante etapa académica, sino también la consolidación de conocimientos y habilidades adquiridas durante la misma.

7.1 Conclusiones personales

Este proyecto supone el broche final de mi paso por el grado en Ingeniería de las Tecnologías de Telecomunicación, algo que, sin duda, me emociona y enorgullece a partes iguales. Tener la oportunidad de desarrollar un trabajo como este, que me ha motivado personalmente desde el inicio, me ha permitido asentar conocimientos adquiridos durante la carrera.

El desarrollo de este sistema ha supuesto una experiencia enriquecedora, no solo a nivel académico, sino también a nivel personal, ya que durante su desarrollo me he podido enfrentar al proceso de diseñar y desarrollar un proyecto desde cero, permitiéndome así conocer más de cerca el proceso de desarrollo completo de una solución técnica, algo que, sin duda, será de gran utilidad en mi trayectoria profesional.

7.2 Conclusiones técnicas

La solución técnica desarrollada es un sistema completamente centrado y adaptado a las necesidades del usuario, haciendo especial hincapié en la accesibilidad y usabilidad del mismo. Tras haber finalizado el desarrollo de este proyecto, se puede concluir que los objetivos iniciales se han cumplido, logrando una aplicación accesible para personas mayores o en situación de dependencia, que les permite visualizar sus alarmas y eventos, y reportar incidencias. Por otro lado, se ha desarrollado la funcionalidad correspondiente al responsable de un usuario, con el objetivo de facilitarles el uso de las nuevas tecnologías a los usuarios.

El principal logro de este proyecto ha sido el desarrollo de una arquitectura clara y con una alta disponibilidad, basada en un servicio REST, que actúa como intermediario entre las aplicaciones Android desarrolladas y la base de datos creada. Además, el uso de esta arquitectura facilita la accesibilidad de este sistema, ya que para usarlo solo es necesario tener acceso a un teléfono móvil Android, cosa que, a día de hoy, es común en la inmensa mayoría de hogares de nuestro país.

Además, el despliegue del servicio REST y de la base de datos PostgreSQL en AWS ha sido un aspecto muy destacable, ya que permite ofrecer fiabilidad y escalabilidad al sistema desarrollado. El uso de este tipo de plataformas en la nube es muy relevante, ya que se han convertido en herramientas ampliamente utilizadas por sus grandes beneficios y posibilidades en cuanto a mantenimiento y accesibilidad.

La realización del proyecto ha permitido poner en práctica conocimientos teóricos adquiridos, que han hecho

posible desarrollar una solución real que tiene un gran potencial para mejorar la calidad de vida de colectivos que se enfrentan a barreras tecnológicas frecuentemente. El trabajo realizado promueve un enfoque tecnológico inclusivo y accesible, fomentando la autonomía de las personas mayores o en situación de dependencia, a través de una herramienta que les permite tener un mayor control de sus rutinas y, en consecuencia, una mayor independencia, al no tener la necesidad de depender de la supervisión de otras personas para tareas presentes en su día a día.

7.3 Líneas futuras

Se exponen en este apartado las posibles mejoras que podrían implementarse en el sistema para conseguir una mayor seguridad, eficiencia y accesibilidad, mejorando así la experiencia del usuario y enriqueciendo el servicio ofrecido.

7.3.1 Seguridad

Para mejorar la seguridad de los usuarios y garantizar la protección de datos sensibles, como contraseñas o datos médicos, se plantean las siguientes mejoras:

- Uso de cifrado en la base de datos para los datos sensibles. Se podría implementar un cifrado en la base de datos para asegurar que, aunque los datos se vean comprometidos, los datos no serán legibles. Para esto se podría hacer uso del algoritmo de cifrado simétrico AES (Advanced Encryption Standard), ampliamente reconocido por su alta seguridad.
- Uso del protocolo HTTPS para garantizar comunicaciones seguras. Debería de implementarse HTTPS en todas las comunicaciones entre el cliente y el servicio REST, garantizando así que todos los datos enviados y recibidos estén protegidos. Esto conllevaría adaptar tanto el servicio como las aplicaciones Android para hacer uso de un certificado SSL/TLS.
- Uso de OAuth 2.0 [14] para autenticar a los usuarios. Esto permitiría a los usuarios autentificarse de forma segura y confiable haciendo uso de credenciales de otros servicios como, por ejemplo, Google. Esto mejora ampliamente la seguridad del sistema ya que delega la autentificación a servicios externos altamente confiables.
- Uso de doble factor de autenticación (2FA). La implementación de autenticación en dos pasos supone una mayor seguridad, al requerir un segundo método de verificación además de la contraseña. Para implementar esto se podría hacer uso de aplicaciones de autenticación reconocidas como Google Authenticator [15] o Microsoft Authenticator [16].
- Uso de autenticación biométrica. En dispositivos que lo permitan, que tengan reconocimiento facial o huella dactilar, se podrían usar estos métodos como segundo factor de autenticación para el acceso a las aplicaciones.

7.3.2 Funcionalidad

En cuanto a las posibles mejoras funcionales del sistema, se destacan algunas funciones que mejorarían altamente la experiencia del usuario y reforzarían la confianza en el sistema:

- Implementación de un servicio GPS. Hacer uso de la localización para registrar el lugar exacto donde se ha reportado una incidencia permitiría una mejor gestión de una incidencia, al proporcionar una información más detallada sobre esta.
- Implementación de notificaciones “push” para los responsables. Disponer de esta posibilidad aumentaría la confianza en el sistema ya que permitiría a los responsables conocer en tiempo real el estado de las incidencias recientemente reportadas por su usuario asociado. Además, también se podrían implementar notificaciones de emergencia para casos en los que una incidencia no se haya resuelto tras un tiempo determinado y/o personalizable, según las necesidades y características del usuario, lo que optimizaría la experiencia del usuario.
- Evaluación de la resolución de incidencias. Hacer uso de un sistema de evaluación para escuchar la

opinión de los usuarios sobre el proceso de resolución de incidencias, permitiría implementar posibles mejoras acordes con las necesidades reales de los usuarios.

- Integración de SensApp en relojes inteligentes. Esta posibilidad permitiría a los usuarios hacer un uso más cómodo y natural de la aplicación. Además, podría ser especialmente útil si el reloj inteligente cuenta con sensores biométricos, como frecuencia cardíaca, saturación de oxígeno o temperatura corporal, ya que se podría incorporar el reporte automático de incidencias cuando se detecten anomalías en los datos recogidos por estos sensores.

ANEXO: CÓDIGO DE LAS IMPLEMENTACIONES

A través del siguiente enlace de GitHub se puede acceder al repositorio donde se encuentra todo el código del proyecto.

https://github.com/marmangam/SensApp

En la rama main de este se pueden encontrar cuatro directorios principales:

- REST/: Contiene el código del servicio REST y de las sentencias necesarias para crear la base de datos PostgreSQL.
- SensApp/: Aplicación Android para los usuarios finales.
- SensAppControl/: Aplicación Android para los responsables.
- SensAppPlus/: Aplicación Android para los administrativos y operadores.

A) Despliegue de las aplicaciones Android

Para ejecutar las aplicaciones Android, basta con seguir los siguientes pasos:

1. Descargar el directorio correspondiente a la aplicación que se quiera desplegar desde el repositorio de GitHub.
2. Abrir el directorio como proyecto en Android Studio.
3. Esperar hasta que Android Studio sincronice el proyecto con Gradle.
4. Pulsar el botón “Run ‘app’” para compilar y ejecutar la aplicación.

B) Despliegue del servicio REST

Los pasos a seguir para desplegar el servicio REST de forma local[11] son los siguientes:

1. Descargar el directorio correspondiente (SensApp/REST/servicio/) desde el repositorio.
2. Acceder al directorio desde un terminal.
3. Ejecutar el siguiente comando[12]:

```
mvn spring-boot:run
```

Estos pasos se pueden ver en la figura Anexo-1.

[11] El despliegue realizado para el proyecto se ha realizado con AWS, como se explica en el capítulo 3.
[12] Es necesario tener Maven y Java instalados.

```
dit@localhost:~$ cd SensApp-main/REST/servicio/
dit@localhost:~/SensApp-main/REST/servicio$ mvn spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.example:rest-service >----------------------
[INFO] Building rest-service 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] >>> spring-boot-maven-plugin:2.4.3:run (default-cli) > test-compile @ res
t-service >>>
[INFO]
[INFO] --- maven-resources-plugin:3.2.0:resources (default-resources) @ rest-ser
vice ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] skip non existing resourceDirectory /home/dit/SensApp-main/REST/servicio/
src/main/resources
[INFO] skip non existing resourceDirectory /home/dit/SensApp-main/REST/servicio/
src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ rest-service
---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 15 source files to /home/dit/SensApp-main/REST/servicio/target/
```

Figura Anexo-1. Ejecución REST

C) Despliegue de la base de datos PostgreSQL

Los pasos a seguir para desplegar la base de datos PostgreSQL son los siguientes:

1. Descargar el directorio correspondiente (SensApp/REST/bbdd/) desde el repositorio. Este contiene los scripts .sql necesarios para la base de datos.
2. Acceder a PostgreSQL como usuario administrador.

   ```
   sudo -i -u postgres
   psql
   ```
3. Crear la base de datos sensapp.

   ```
   CREATE DATABASE sensapp;
   ```
4. Una vez creada, podemos comprobarlo con el comando:

   ```
   \l
   ```

La ejecución de estos pasos se muestra en la figura Anexo-2

```
dit@localhost:~$ cd SensApp-main/REST/bbdd/
dit@localhost:~/SensApp-main/REST/bbdd$ sudo -i -u postgres
postgres@localhost:~$ psql
psql (14.17 (Ubuntu 14.17-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# CREATE DATABASE sensapp;
CREATE DATABASE
postgres=# \l
postgres=#
```

Figura Anexo-2. Creación base de datos

5. Acceder a ella mediante el comando:

   ```
   psql -U 'usuario' -d sensapp
   ```
6. Ejecutar los scripts SQL desde el directorio SensApp/REST/bbdd/. El primero se usa para crear las tablas necesarias y el segundo, opcional, para introducir unos datos iniciales.

   ```
   \i sentencias.sql
   \i sentenciasInsert.sql
   ```

```
dit@localhost:~/SensApp-main/REST/bbdd$ psql -U dit -d sensapp
Password for user dit:
psql (14.17 (Ubuntu 14.17-0ubuntu0.22.04.1))
Type "help" for help.

sensapp=> \i sentencias.sql
psql:sentencias.sql:5: NOTICE:  table "usuarios" does not exist, skipping
DROP TABLE
psql:sentencias.sql:6: NOTICE:  table "responsables" does not exist, skipping
DROP TABLE
psql:sentencias.sql:7: NOTICE:  table "alarmas" does not exist, skipping
DROP TABLE
psql:sentencias.sql:8: NOTICE:  table "eventos" does not exist, skipping
DROP TABLE
psql:sentencias.sql:9: NOTICE:  table "administrativos" does not exist, skipping
DROP TABLE
psql:sentencias.sql:10: NOTICE:  table "operadores" does not exist, skipping
DROP TABLE
psql:sentencias.sql:11: NOTICE:  table "incidencias" does not exist, skipping
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
sensapp=> \i sentenciasInsert.sql
INSERT 0 3
INSERT 0 3
INSERT 0 3
INSERT 0 3
INSERT 0 3
INSERT 0 3
INSERT 0 3
```

Figura Anexo-3. Acceso a la BBDD y creación de tablas

7. Para comprobar que todas las tablas se han creado correctamente haciendo uso del comando:

   `\dt`

```
sensapp=> \dt
             List of relations
 Schema |      Name       | Type  | Owner
--------+-----------------+-------+-------
 public | administrativos | table | dit
 public | alarmas         | table | dit
 public | eventos         | table | dit
 public | incidencias     | table | dit
 public | operadores      | table | dit
 public | responsables    | table | dit
 public | usuarios        | table | dit
(7 rows)
```

Figura Anexo-4. Tablas creadas

Siguiendo estos pasos, se consigue desplegar de forma local el ecosistema SensApp, las aplicaciones Android, el servicio REST y la base de datos.

REFERENCIAS

[1] Junta de Andalucía, «Servicio Andaluz de teleasistencia» [En línea]. Available: https://www.juntadeandalucia.es/agenciadeserviciossocialesydependencia/index.php/m-teleasistencia

[2] MSI, «Especificaciones GF63-Thin-10SX-GTX» [En línea]. Available: https://es.msi.com/Laptop/GF63-Thin-10SX-GTX/Specification

[3] Huawei, «HUAWEI Mate 10 lite» [En línea]. Available: https://consumer.huawei.com/es/support/phones/mate10-lite/

[4] Android Studio, «Desarrolladores de Android» [En línea]. Available: https://developer.android.com/?hl=es-419

[5] Spring, «Spring makes Java simple» [En línea]. Available: https://spring.io/

[6] Maven, «Welcome to Apache Maven» [En línea]. Available: https://maven.apache.org/

[7] RESTClient, «A debugger for RESTful web services» [En línea]. Available: http://restclient.net/

[8] PostgreSQL, «PostgreSQL: The World's Most Advanced Open Source Relational Database» [En línea]. Available: https://www.postgresql.org/

[9] SQLite, «What Is SQLite?» [En línea]. Available: https://www.sqlite.org/index.html

[10] Swagger, «API Development for Everyone» [En línea]. Available: https://swagger.io/

[11] Visual Studio Code, «Your code editor» [En línea]. Available: https://code.visualstudio.com/

[12] Amazon Web Services, «AWS» [En línea]. Available: https://aws.amazon.com/es/?nc2=h_lg

[13] GitHub, «Welcome to tmux!» [En línea]. Available: https://github.com/tmux/tmux/wiki

[14] Auth0, «¿Qué es OAuth 2.0?» [En línea]. Available: https://auth0.com/es/intro-to-iam/what-is-oauth-2

[15] Google, «Google Authenticator» [En línea]. Available: https://support.google.com/accounts/answer/1066447?hl=es&co=GENIE.Platform%3DAndroid

[16] Microsoft, «Microsoft Authenticator» [En línea]. Available: https://www.microsoft.com/es-es/security/mobile-authenticator-app