UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA INFORMÁTICA

APLICACIÓN PARA LA MONITORIZACIÓN DE RESIDENCIAS DE ANCIANOS EN SITUACIONES DE RIESGO

APPLICATION FOR MONITORING NURSING HOMES IN AT-RISK SITUATIONS

Realizado por
Antonio Cabañas Zurita

Tutorizado por
Eduardo Guzmán De los Riscos

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, diciembre de 2022

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA INFORMÁTICA

Aplicación para la monitorización de residencias de ancianos en situaciones de riesgo

Application for monitoring nursing homes in at-risk situations

Realizado por
Antonio Cabañas Zurita

Tutorizado por
Eduardo Guzmán De los Riscos

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, Diciembre de 2022

Resumen

Este trabajo de fin de grado se centra en el desarrollo de una aplicación web que cubre las necesidades que los servicios de emergencia demandan para monitorizar las residencias de ancianos en situaciones de emergencia. El objetivo, en este sentido, es ofrecer una solución eficiente y adecuada para la necesidad a la que se quiere dar respuesta. En el proyecto se han marcado diferentes objetivos, la calidad del servicio, en relación con los usuarios que la usan, comunicación, gestión y productividad eficientes. Para el diseño de la aplicación se ha utilizado el patrón MVVM (Modelo-Vista-VistaModelo) para poder resolver y superar los retos establecidos. Las tecnologías utilizadas para el desarrollo front-end han sido Angular, TypeScript, HTML5, CSS para el back-end del sistema han sido PHP, Python o MySQL. Además, se han hecho uso de diferentes técnicas y algoritmos computacionales para resolver la lógica de la aplicación. Por ejemplo, la resolución de identificación segura de usuarios o la geolocalización de este. En general, este proyecto trata de incorporar las técnicas más actuales y eficaces para un desarrollo profesional y completo. También se busca hacer un sistema seguro que permita ser usado en entornos modernos, no solo para este sistema sino para cualquier otro que se implemente.

Palabras clave: Angular, PHP, Python, TypeScript.

Abstract

This final cumulative project displays the development of a web application that deals with the needs of emergency services when monitoring nursing homes in at-risk situations. The goal is finding the most efficient solution. There have been various objectives set in this project. Namely, the quality of service (QoS) for users, which facilitates efficient communication, management and productivity. The MVVM (Model-View-ViewModel) pattern has been used for the design of the application in order to solve and overcome the extant challenges. The technologies used for the front-end development included Angular, TypeScript, HTML5, and CSS. For the other side of the system, they include PHP, Python and MySQL. Furthermore, to solve the application's logical issues, assorted computational techniques and algorithms have been deployed. For example, the resolution of identification access and the geolocation of the user. In conclusion, this project tries to incorporate the most current techniques for a completely professional development. It also seeks a secure system that can be used in modern environments, and in any other system that it would be implemented in.

Keywords: **Angular, PHP, Python, TypeScript.**

Índice

1

Introducción

1.1 Motivación

Cada día se viven más situaciones de emergencia que afectan a entornos críticos y a personas vulnerables. No obstante, gracias a la prevención de riesgos se puede realizar un control de la situación y así paliar los posibles efectos que estos pueden provocar. Sin embargo, a pesar del proceso de transformación digital que están sufriendo en los últimos años muchos sectores de la sociedad, es todavía difícil de encontrar herramientas que ofrecen precisión e información que sirva para ayudar a los cuerpos de seguridad y a los especialistas en emergencias a desarrollar su labor. Estas herramientas son imprescindibles para la toma de decisiones y desarrollar la estrategia de actuación de los medios.

La pandemia causada por el COVID-19 afectó gravemente a la mayoría de las residencias de mayores de nuestro país. Por ello, surgió la necesidad de actualizar los datos de las residencias y monitorizar el estado de estas en entornos críticos. Aunque existen aplicaciones capaces de ofrecer herramientas de análisis intuitivo, carecen de capacidad para informar con precisión de catástrofes que afecten directamente al conjunto de residencias.

1.2 Objetivos

El proyecto se centra principalmente en desarrollar un sistema que monitorice y geolocalice las residencias durante catástrofes, además de mantener actualizados los datos de la plataforma. También se busca que los datos estén actualizados para la efectividad de los servicios de emergencia.

Será prioritario mostrar los datos de afección de la catástrofe, y aplicar una relación directa con las residencias. Será un sistema destinado a equipos de emergencias, por lo que se deberá presentar de forma clara e intuitiva la información para un buen análisis de los usuarios. Por otro lado, los centros residenciales deberán acceder al sistema para mantenerlo actualizado.

1.3 Solución Propuesta

Tras realizar un análisis de los objetivos y valorar la motivación inicial del proyecto, se ha propuesto realizar una aplicación web dedicada a la monitorización de las residencias. El sistema incorpora un mapa donde se ubican todos los centros, indicando las zonas afectadas. Este mapa incluye diferentes capas de visualización mostrando los diferentes riesgos a catástrofes naturales a los que están sometidas.

Por otro lado, la aplicación aloja una base de datos para mantener almacenada y actualizada toda la información necesaria para los servicios de emergencia.

1.4 Metodología usada

El proyecto se ha desarrollado haciendo uso de la metodología scrum. Esta es utilizada en entornos grupales para garantizar que en un proyecto se consigan resultados de calidad, gracias a la aplicación de buenas prácticas. Se desarrolla mediante una estrategia incremental iterativa evitando perder tiempo en la planificación y ejecución completa del sistema.

Este proyecto en particular se ha dividido en cuatro etapas. La primera de ellas ha sido investigar sobre las tecnologías más apropiadas para la aplicación, entre ellas se debía valorar el coste, funcionalidades y diseño. Una vez seleccionadas las herramientas correctas, continuamos con la segunda fase. En esta, se ha

desarrollado la aplicación incorporando todos los requisitos definidos, a la vez que se han aprendido aquellos aspectos de las tecnologías usadas. Además, se han ido realizando validaciones y pruebas constantes para ir verificando el buen funcionamiento del sistema.

Tras completar el sistema se llevan a cabo las comprobaciones globales en diferentes equipos y localizaciones de la aplicación para comprobar que cumple correctamente los requisitos funcionales y no funcionales definidos. Tras completar dicha fase, comenzamos con la última donde se redacta la memoria para la documentación correcta de la aplicación.

1.5 Descripción general del sistema

En este capítulo se dará a conocer la descripción de la aplicación desarrollada y su funcionamiento general. Así como las partes que lo conforman y su funcionamiento dentro del mismo. También se explicará su arquitectura.

El sistema está dedicado a la monitorización de residencias durante situaciones de emergencia, mostrando un mapa donde se geolocalizan estas. Incluyendo también capas que ofrecen información adicional, como los terremotos, zonas de riesgo a inundaciones, mareas, etc. Además, permite generar rutas a cualquier residencia desde la localización del usuario. Este mapa muestra también las catástrofes que suceden mediante zonas circulares y rojizas que indican el territorio afectado. Al seleccionar estas zonas se cargan marcos con información precisa sobre la catástrofe.

Otra funcionalidad de la aplicación es la representación de la información almacenada mediante listados. Estos disponen de capacidad de filtrado, búsqueda y algunos de ellos son exportables a documentos en formato pdf. Todos los datos son almacenados y procesados, gracias al registro de formularios que permiten grabar la información necesaria. Además, se mantiene actualizada por la posibilidad de modificación de dichos registros.

En el sistema se han definido roles que diferencian las funcionales permitidas para cada uno de los usuarios. El rol administrador, tiene la capacidad completa de uso

de la aplicación, y sus tareas son el mantenimiento y la comprobación del funcionamiento correcto del sistema.
Por otro lado, el técnico de emergencias podrá acceder a la plataforma y registrar catástrofes siendo los principales usuarios de la aplicación. Existe otro tipo de usuario, el editor de residencias. Este es el encargado de mantener actualizadas e incorporar residencias en el sistema.

1.6 Estructura de la memoria

Este documento está estructurado en diferentes capítulos: en el primero se ha introducido la idea principal de proyectos y los objetivos a cumplir, incluyendo una descripción de la metodología y del sistema.

El segundo mostrará las tecnologías y herramientas utilizadas para el desarrollo, definiendo Angular entre otras. A continuación, se realizarán las especificaciones y el análisis del sistema como son los requisitos, casos de uso, etc. El siguiente capítulo, abordará el diseño de la aplicación definiendo los usuarios, los roles, y los diferentes diagramas que indican como están estructurados los datos.

La implementación y pruebas se redactarán en el cuarto capítulo. Describiremos en qué ha consistido la fase de implementación y los retos superados durante el desarrollo. También se mostrarán los errores detectados durante las pruebas, y cómo se han solucionado.

Por último, se exponen las conclusiones obtenidas tras la elaboración completa del proyecto y posibles desarrollos futuros para la continua actualización del sistema.

2

Tecnologías y herramientas utilizadas

2.1 Angular

Angular es un framework de código abierto desarrollado por Google que permite crear páginas web dinámicas gracias al uso del patrón arquitectónico MVVM (Modelo-Vista-VistaModelo). Su programación se basa en TypeScript, permitiendo una conexión directa entre la estructura HTML y el código JavaScript que ejecuta las funcionalidades. Este framework facilita el desarrollo de una web adaptativa(*responsive*), permitiendo una correcta visualización desde diferentes dispositivos.

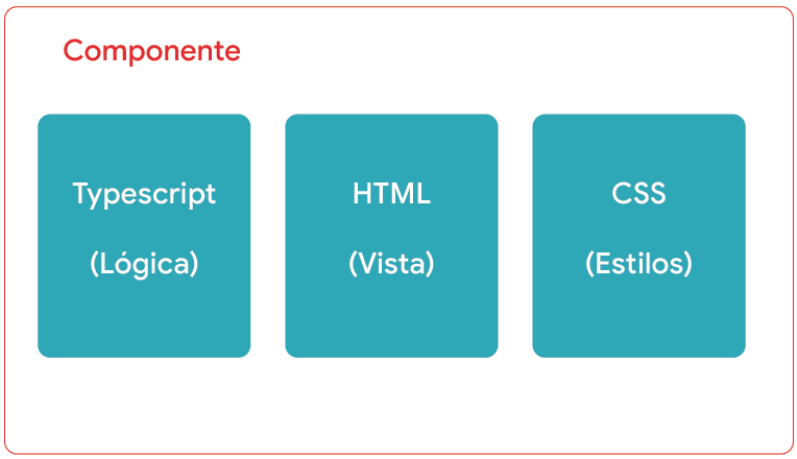
La decisión de usar esta tecnología en el proyecto es por su finalidad de creación de aplicaciones web, y la curva de aprendizaje es muy reducida con respecto a otros frameworks.

Desde la versión 2, Angular comenzó a ser un framework completo presentando grandes ventajas y funcionales respecto a librerías muy utilizadas como AJAX, test unitarios, etc. Algunas de estas son ofrecer todas las opciones de forma homogénea evitando tener que conectar y configurar librerías que permiten estas funcionalidades.

También cabe destacar la filosofía de poder incrustar código script en los ficheros HTML, a diferencia del resto que hacen lo contrario. Esta arquitectura permite trabajar en conjunto con todos los ficheros, aunque posteriormente sean módulos diferentes.

Es importante mencionar la estructura utilizada por esta tecnología para la gestión del contenido del código fuente. Se utilizan los componentes que forman el sistema web. El componente se define como el elemento reutilizable que forma parte de la aplicación y la estructura, siendo similar a las extremidades en el ser humano por la composición en subconjuntos más pequeños. Está compuesto por tres subconjuntos, el HTML o vista, donde se diseña la estructura y añaden elementos de la interfaz de usuario. El segundo subconjunto es CSS o estilos que aplica características de diseño y personalización al anterior. Y el último, el TypeScript o la lógica que capta las funcionalidades que realiza la interfaz.

Figura 1: Esquema componente angular.



Fuente: https://ngchallenges.gitbook.io/project/componentes

2.2 PHP

Es un lenguaje sencillo muy utilizado para el desarrollo web especialmente para el lado del servidor, caracterizado por la potencia, versatilidad, robustez y modularidad. Contiene multitud de funcionalidades para realizar peticiones web, y

conectar con bases de datos. Esta tecnología permite desarrollar web con gran complejidad, añadiendo aspectos importantes de seguridad gracias al gran número de módulos y librerías que aportan al desarrollador un gran número de herramientas libres.

Esta tecnología está dotada de controladores capaz de conectar con el framework angular y procesar correctamente las peticiones realizadas por la API, incluyendo mecanismos de autentificación.

2.3 Python

Es uno de los lenguajes más usados en la actualidad. Destaca por ser multiparadigma, permitiendo a los desarrolladores optar por un estilo de programación orientado a objetos. Otra característica que presenta es que usa tipado dinámico, permitiendo a una variable cambiar de tipo. Además, es muy usado para el tratamiento masivo de datos.

2.4 Apache

Apache es un software de servidor web gratuito y de código abierto para sistemas Unix, siendo un sistema de los más fiables y utilizados en este tipo de servicios.

Su principal función es conectar a clientes con un servidor mediante navegadores, es decir, haciendo uso de los protocolos http y https. Durante la comunicación, se comparten archivos, imágenes…, por lo que el software debe garantizar que las transmisiones sean fluidas y seguras entre ambas partes. Los usuarios que deciden usarlo, se benefician de sus características, entre ellas están la fiabilidad, el soporte y la continua actualización. Sin embargo, puede generar algunos fallos de rendimiento cuando el tráfico es demasiado alto. También cuenta con amplias opciones de configuración que pueden generar brechas de seguridad.

2.5 OpenStreetMap

*Open Street Map e*s un servicio *open source* de mapas, es un proyecto colaborativo a nivel mundial que tiene la finalidad de crear mapas libres, editables y detallados para su uso. Se puede incorporar sencillamente a proyectos web o móviles, además de contar con infinidad de herramientas que agregan funcionalidades extra, se pueden generar rutas, implementar capas sobre este, etc. Incluye librerías que

permiten mostrar y renderizar los mapas en cualquier dispositivo y tecnología. Es conocida como una de las mejores alternativas a Google Maps o Apple Maps.

2.6 Librerías

Se han utilizado diferentes librerías para cubrir los requisitos del sistema. Para el uso de mapas se ha utilizado Leaflet gracias a su sencillez y ligereza, conectando servicios como capas (WMS). También se pueden generar marcas y seleccionar zonas dentro del mapa.

Por otro lado, se utilizan utilidades de la librería de 8bootstrap 5 para dar estilo a los textos. Además, se utilizan formularios y tablas que permiten un diseño *responsive* y adecuado para la mayoría de dispositivos que van a ejecutar la aplicación.
Otra librería utilizada es la encargada de generar documentos PDF, está enfocada en el uso de JavaScript y destaca por su ligereza y la sencillez que facilita la creación de los informes o listados.

En angular disponemos de recursos que generan notificaciones visuales en la interfaz, facilitando al sistema la muestra de información emergente. Para ello, se hace uso de la librería 'ngxtoastr' desarrollada en JavaScript y con numerosas opciones para generar notificaciones. Se puede especificar el tiempo, el aspecto o incluso el contenido.

La seguridad del sistema aporta un valor, además de cumplir con normativas exigidas para poder poner en marcha cualquier software. Por esto, se aplica el encriptado de datos y la generación de tokens. Los tokens y encriptado son desarrollados gracias al uso del paquete 'jwTOKEN' que genera codificaciones únicas en base 64 recogiendo los datos de un usuario, por ejemplo. Además, se utiliza para cumplir el esquema general de un token y la identificación mediante este. En la Figura 2 se puede apreciar el esquema de acceso mediante el token.

Figura 2: Ciclo de vida de un token JWT

Fuente: https://openwebinars.net

2.7 Git

El desarrollo de esta aplicación exige mantener controladas las versiones que se generan, por ello esta herramienta ofrece el sistema de control necesario. El uso de estas tecnologías es muy común en cualquier empresa dedicada a desarrollo de software. Su capacidad para facilitar el trabajo en equipo, creando copias de seguridad y resolviendo conflictos, son características que destacan y se utilizan en proyectos de grandes capacidades.

Este tipo de herramientas se pueden encontrar con soporte y servicios de almacenamiento en la nube para mejorar sus prestaciones, y permitir la gestión centralizada de los proyectos en una única plataforma. En este caso se utilizan el alojamiento de GitHub al ofrecer un plan gratuito para proyectos como el que se ha desarrollado.

2.8 MariaDB

Los sistemas de gestión de base de datos son imprescindibles en la mayoría de las aplicaciones web que se desarrollan. En el proyecto se ha empleado un motor MySQL de dominio público que es MariaDB debido a la relación tan estrecha que mantienen ambos, aunque el último incorpora mejoras que hacen de las consultas algo más complejo y avanzado.

Describiendo el sistema de base de datos, es destacado por su velocidad en operaciones complejas mediante el uso de almacenamiento en caché. La compatibilidad con software existente que hacen uso de entornos similares o la gran ventaja de ser software libre son características diferenciadoras del software.

3

Especificación y Análisis

3.1 Catálogo de Requisitos

Esta sección será la encargada de definir los requisitos que se estipulan a lo largo del desarrollo del sistema para que funcione de forma adecuada y se cumpla el objetivo del proyecto. A continuación, se muestra el listado de dichos requisitos con información adicional como el tipo de requisito, título, código, dificultad e importancia.

3.1.1 Requisitos funcionales

Entre las tablas 1 a la 19 se especifican los requisitos funcionales de la aplicación.

Tabla 1: Requisito funcional RF-01

Código	RF-01
Título	Localización de residencias
Descripción	Permite la visualización de un mapa donde estén ubicadas todas las residencias del sistema mediante coordenadas.
Tipo	Funcional
Prioridad (Crítico, Importante, Secundario)	Crítico
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Significativo

Fuente: Elaboración propia

Tabla 2: Requisito funcional RF-02

Código	RF-02
Título	Visualización datos de residencias
Descripción	El sistema permite mostrar los datos definidos para las residencias, entre los que contiene dirección, teléfono, plazas, etc.
Tipo	Funcional
Prioridad (Crítico, Importante, Secundario)	Crítico
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Normal

Fuente: Elaboración propia

Tabla 3: Requisito funcional RF-03

Código	RF-03
Título	Registro de zona afectada
Descripción	El sistema permite la recolección de sucesos que impliquen una situación de riesgo para los centros residenciales, es decir, los servicios de emergencias deberán introducir una serie de datos para hacer efectiva la activación. Se recogerán datos como el área, tipo de catástrofe, etc.
Tipo	Funcional
Prioridad (Crítico, Importante, Secundario)	Crítico
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Significativo

Fuente: Elaboración propia

Tabla 4: Requisito funcional RF-04

Código	RF-04
Título	Monitorización zona afectada
Descripción	La aplicación debe ser capaz de localizar todas las residencias de ancianos afectadas en la zona de la catástrofe. Los datos de estas serán accesibles de forma automática tras el registro de un suceso nuevo mediante una lógica de búsqueda completa.
Tipo	Funcional
Prioridad (Crítico, Importante, Secundario)	Crítico
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Crítico

Fuente: Elaboración propia

Tabla 5: Requisito funcional RF-05

Código	RF-05
Título	Generar ruta a residencia
Descripción	El sistema debe de crear una ruta hacia la residencia deseada, por lo tanto, será posible la geolocalización para trazar la ruta en el mapa.
Tipo	Funcional
Prioridad (Crítico, Importante, Secundario)	Importante
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Significativo

Fuente: Elaboración propia

Tabla 6: Requisito funcional RF-06

Código	RF-06
Título	CRUD Residencias
Descripción	Se debe permitir la creación, lectura, actualización y eliminación de residencias. Estas modificaciones deben ser gestionadas por un sistema de roles para evitar cualquier cambio indebido.
Tipo	Funcional
Prioridad (Crítico, Importante, Secundario)	Importante
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Significativo

Fuente: Elaboración propia

Tabla 7: Requisito funcional RF-07

Código	RF-07
Título	CRUD Usuarios
Descripción	Se permite la creación, lectura, actualización y eliminación de los usuarios que debe contener el sistema para su uso. La gestión de los permisos se llevará a cabo por diferentes roles, evitando así modificaciones incorrectas.
Tipo	Funcional
Prioridad (Crítico, Importante, Secundario)	Importante
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Significativo

Fuente: Elaboración propia

Tabla 8: Requisito funcional RF-08

Código	RF-08
Título	Gestión Rol
Descripción	El sistema contará con una gestión de roles para distinguir permisos por tipos de usuario. Se crearán diferentes roles entre los que existirá el rol administrador, técnico de emergencias… Permitirá asignarle funcionalidades especificas a cada grupo de usuarios.
Tipo	Funcional
Prioridad (Crítico, Importante, Secundario)	Importante
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Normal

Fuente: Elaboración propia

Tabla 9: Requisito funcional RF-09

Código	RF-09
Título	Control de acceso
Descripción	La aplicación será accesible únicamente para usuarios registrados en ella. Debe contar con un acceso controlado por usuario y contraseña que previamente son facilitados.
Tipo	Funcional
Prioridad (Crítico, Importante, Secundario)	Importante
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Normal

Fuente: Elaboración propia

Tabla 10: Requisito funcional RF-10

Código	RF-10
Título	Menú principal
Descripción	Las principales funcionalidades del sistema serán accesibles mediante un menú que direccionará a todas las ventanas de la aplicación. Dicho menú debe informar sobre el usuario activo.
Tipo	Funcional
Prioridad (Crítico, Importante, Secundario)	Importante
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Significativo

Fuente: Elaboración propia

Tabla 11: Requisito funcional RF-11

Código	RF-11
Título	Autogestión del usuario
Descripción	A el usuario registrado le está permitido hacer cambios de sus datos personales, y de su contraseña.
Tipo	Funcional
Prioridad **(Crítico, Importante, Secundario)**	Importante
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Normal

Fuente: Elaboración propia

Tabla 12: Requisito funcional RF-12

Código	RF-12
Título	Mapa con capas
Descripción	El mapa del sistema debe implementar capas que muestren el riesgo sísmico, zonas inundables o efectos adversos que puedan acontecerse. Estas capas permiten evaluar el riesgo que sufren las residencias de ancianos ante catástrofes naturales u otros fenómenos.
Tipo	Funcional
Prioridad **(Crítico, Importante, Secundario)**	Importante
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Significativo

Fuente: Elaboración propia

Tabla 13: Requisito funcional RF-13

Código	RF-13
Título	Activar o desactivar emergencia
Descripción	El sistema permite activar o desactivar los sucesos que se registran en la aplicación para dar por finalizado estos. Esta funcionalidad permite a los servicios de emergencias recibir avisos para su actuación.
Tipo	Funcional
Prioridad **(Crítico, Importante, Secundario)**	Crítico
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Normal

Fuente: Elaboración propia

Tabla 14: Requisito funcional RF-14

Código	RF-14
Título	Control de actualizaciones
Descripción	Se debe llevar un registro de las actualizaciones de los datos de las residencias, para que los servicios de emergencias puedan visualizar la información actual.
Tipo	Funcional
Prioridad **(Crítico, Importante, Secundario)**	Secundario
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Normal

Fuente: Elaboración propia

Tabla 15: Requisito funcional RF-15

Código	RF-15
Título	Geolocalización del cliente
Descripción	Será posible geolocalizar al usuario y facilitar la generación de una ruta.
Tipo	Funcional
Prioridad **(Crítico, Importante, Secundario)**	Importante
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Significativo

Fuente: Elaboración propia

Tabla 16: Requisito funcional RF-16

Código	RF-16
Título	Generar listado en formato pdf
Descripción	El sistema permite generar documentos en formato pdf de los listados existentes en la aplicación. Estarán disponibles para su descarga directa desde la aplicación.
Tipo	Funcional
Prioridad **(Crítico, Importante, Secundario)**	Secundario
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Significativo

Fuente: Elaboración propia

Tabla 17: Requisito funcional RF-16

Código	RF-17
Título	Listados
Descripción	La aplicación permite mostrar listados con el conjunto de residencias, usuarios, etc. Estos serán creados mediante tablas dinámicas que permiten las búsquedas y la configuración de la cantidad de datos a mostrar.
Tipo	Funcional
Prioridad **(Crítico, Importante, Secundario)**	Importante
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Significativo

Fuente: Elaboración propia

Tabla 18: Requisito funcional RF-18

Código	RF-18
Título	Expiración de sesión
Descripción	La aplicación contará con un servicio de expirado de sesión para evitar dejar sesiones de usuario abiertas innecesariamente.
Tipo	Funcional
Prioridad **(Crítico, Importante, Secundario)**	Secundario
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Significativo

Fuente: Elaboración propia

Tabla 19: Requisito funcional RF-19

Código	RF-19
Título	Inserción mediante CSV
Descripción	Se permite la inserción de datos mediante ficheros en formato csv, para ello se define una estructura determinada. Esta utilidad permite realizar inserciones de datos masivas.
Tipo	Funcional
Prioridad **(Crítico, Importante, Secundario)**	Secundario
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Significativo

Fuente: Elaboración propia

3.1.2 Requisitos No Funcionales

Entre las tablas 20 a la 25 se especifican los requisitos no funcionales de la aplicación.

Tabla 20: Requisito no funcional RNF-01

Código	RNF-01
Título	Cumplir la legislación de LOPD
Descripción	La aplicación debe cumplir con la actual ley orgánica de la protección de datos personales que van a ser tratados en ella.
Tipo	No Funcional
Prioridad **(Crítico, Importante, Secundario)**	Crítico
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Normal

Fuente: Elaboración propia

Tabla 21: Requisito no funcional RNF-02

Código	RNF-02
Título	Tiempo de repuesta
Descripción	El tiempo de respuesta debe ser el mínimo posible para ofrecer la información lo más rápido y completa posible.
Tipo	No Funcional
Prioridad (Crítico, Importante, Secundario)	Importante
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Crítico

Fuente: Elaboración propia

Tabla 22: Requisito no funcional RNF-03

Código	RNF-03
Título	Disponibilidad 24/7
Descripción	El sistema debe estar disponible y activo todos los días del año durante las 24 horas.
Tipo	No Funcional
Prioridad (Crítico, Importante, Secundario)	Crítico
Nivel de Riesgo / Dificultad (Crítico, Significativo, Normal)	Significativo

Fuente: Elaboración propia

Tabla 23: Requisito no funcional RNF-04

Código	RNF-04
Título	Interfaz adaptativa (*responisve*)
Descripción	La interfaz del sistema será totalmente responsive, será adaptable a cualquier dispositivo y su contenido se visualiza correctamente independientemente del sistema operativo y tamaño de pantalla.
Tipo	No Funcional
Prioridad **(Crítico, Importante, Secundario)**	Importante
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Significativo

Fuente: Elaboración propia

Tabla 24: Requisito no funcional RNF-05

Código	RNF-05
Título	Facilidad de uso
Descripción	La interfaz de la aplicación será sencilla y amigable con el usuario. Se adaptará a cualquier tipo de usuario ya que es una aplicación destinada a usuario de características muy diferentes.
Tipo	No Funcional
Prioridad **(Crítico, Importante, Secundario)**	Importante
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Significativo

Fuente: Elaboración propia

Tabla 25: Requisito no funcional RNF-06

Código	RNF-06
Título	Integridad de los datos
Descripción	Los datos que se recogen deben ser correctos y completos, aumentado así la seguridad de los datos y la fiabilidad de la información ofrecida.
Tipo	No Funcional
Prioridad **(Crítico, Importante, Secundario)**	Importante
Nivel de Riesgo / Dificultad **(Crítico, Significativo, Normal)**	Significativo

Fuente: Elaboración propia

3.2 Catálogo de Usuarios

En esta sección, a través de la tabla 26, se describen todos los roles que se han creado para el manejo de la aplicación, incluyendo las funcionalidades y labores que deben realizar.

Tabla 26: Catalogo de usuarios

Código	Denominación	Descripción	Autor
Rol-01	Administrador	El nivel de privilegios de acceso del administrador es el máximo. Son distinguibles el cambio de rol y la creación de usuarios entre otros. Además, será el responsable del mantenimiento del sistema y de que este funcione correctamente. Dispondrán gestión total de los usuarios del sistema	Antonio Cabañas
Rol-02	Editor Residencias	El rol de editor de residencias dispondrá de permisos para gestionar las residencias. Es el responsable de mantener actualizadas las residencias de ancianos. Este tipo de usuario principalmente estará ocupado por personal administrativo de centros de ancianos. También se les permite registrar catástrofes en la aplicación, ya que son los primeros afectados.	Antonio Cabañas
Rol-03	Técnico de Emergencias	Los técnicos de emergencias ocuparán el puesto más importante. Ellos podrán registrar catástrofes y desactivarlas ya que son los encargados de actuar ante cualquier suceso.	Antonio Cabañas

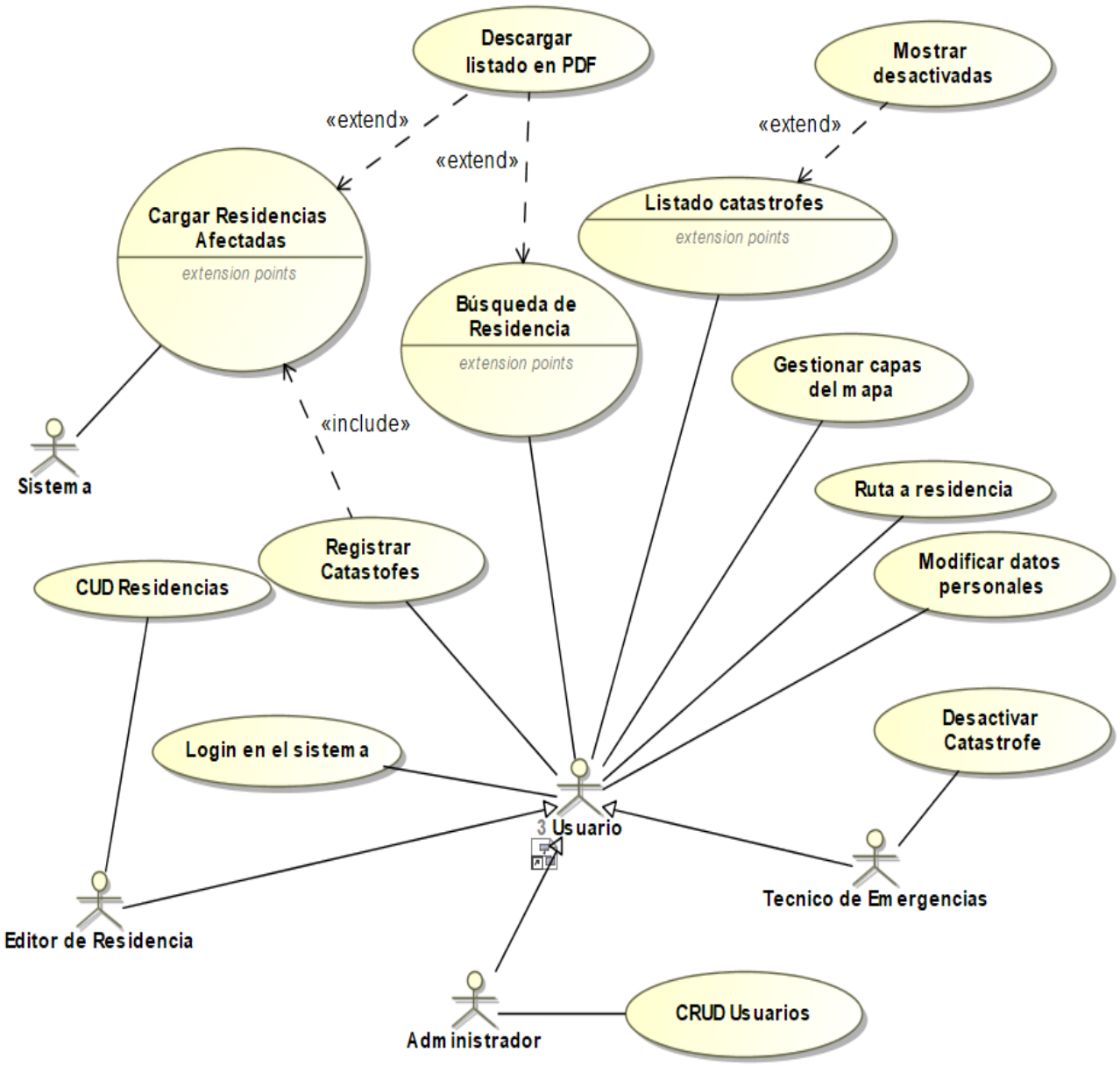
Fuente: Elaboración propia

3.3 Casos de Uso

En este apartado se comenzarán a explicar las características y tareas del sistema, serán detallados los roles que intervienen en los procesos. Además, se detallarán los roles y tareas heredadas con sus respectivas asociaciones.

La realización de la sección se ha realizado gracias al software MagicDraw y representado mediante el lenguaje UML, específico para el modelado de software. Seguidamente en la Figura 3, se exponen los modelos de casos de uso.

Figura 3: Diagrama de casos de uso



Fuente: Elaboración propia

3.4 Especificaciones de los casos de uso

Esta sección especificará los casos de uso previamente modelados en la Figura 3, se describen a continuación.

- Inicio de sesión en el sistema

Escenario básico:

1. El usuario registrado previamente en el sistema accede a la URL.
2. Introduce su nombre de usuario y contraseña.
3. Seguidamente pulsa el botón de login.
4. El sistema valida la información introducida.
5. El usuario accede a la pantalla de inicio de la aplicación web.

- CRUD de Usuarios

Escenario básico:

1. El usuario con el rol de administrador accede al sistema.
2. Accede a usuarios desde el menú.
3. Mediante el botón insertar usuarios accede al formulario de registro.
4. Rellena los datos del formulario y pulsa crear.
5. El sistema valida los datos, creando y almacenando al nuevo usuario.

- CRUD Residencias

Escenario básico:

1. El usuario accede a la aplicación, este debe tener asignado el rol editor de residencias.
2. Selecciona en el mapa la residencia deseada, y se muestra la información de esta.
3. Selecciona el botón editar y se le muestra un formulario con los datos actuales de la residencia seleccionada.
4. Edita los campos que desee, y pulsa el botón editar.

5. La aplicación notificara mediante un mensaje push que se ha realizado correctamente.

- Registrar catástrofe

Escenario básico:

1. El usuario accede a la aplicación y pulsa el botón de registrar catástrofe.
2. La aplicación carga el formulario, y el usuario rellena confirmando con el botón registrar.
3. El sistema registra la catástrofe en la base de datos y confirma mediante una notificación push.
4. Tras la nueva anotación se lanza un algoritmo para calcular las residencias afectadas.
5. El sistema muestra en el mapa la zona afectada, y permite mostrar las residencias afectadas.

- Búsqueda de residencia

Escenario básico:

1. El usuario accede a la aplicación y pulsa el botón listado que muestra las residencias registradas en el sistema.
2. El usuario dispone de un campo de búsqueda en la tabla de las residencias, introduce el dato deseado para filtrar.
3. El sistema muestra automáticamente los resultados que coincidan con el campo de búsqueda.

- Modificar datos personales

Escenario básico:

1. El usuario accede al sistema.
2. El usuario pulsa el desplegable del menú donde aparece su nombre.

3. Selecciona la opción de editar, y se muestra un formulario con su información.
4. El usuario modifica los datos deseados, y selecciona el botón editar.
5. El sistema confirma la actualización de los datos personales gracias a una notificación emergente.

- Desactivar catástrofe

Escenario básico:

1. El técnico de emergencias accede al sistema.
2. Este usuario selecciona la catástrofe que desea desactivar.
3. El sistema muestra la información de la catástrofe, y el usuario pulsa el botón desactivar.
4. El sistema recarga la ventana, y elimina del mapa la zona marcada.

- Gestionar capas del mapa

Escenario básico:

1. El usuario accede a la aplicación.
2. El usuario en la página principal, dentro del mapa en la esquina superior derecha hay un icono que representa las capas.
3. Selecciona este icono, y señala la capa o capas deseadas y se cargan sobre el mapa.

- Descargar listado PDF

Escenario básico:

1. El usuario accede a la aplicación, y carga un listado de residencias.
2. El sistema le muestra una tabla con el listado correspondiente de residencias.
3. La cabecera de la tabla muestra un botón de descarga, el usuario pulsa este.
4. La aplicación crea un documento en formato pdf con los datos de la tabla, y lanza la descarga en el navegador.

- Listado catástrofes

Escenario básico:

1. El usuario accede al sistema.
2. Pulsa el botón de catástrofes del menú superior.
3. La aplicación le muestra un listado con las catástrofes activas.

- Mostrar desactivadas

Escenario básico:

1. El usuario accede al sistema.
2. Pulsa el botón de catástrofes del menú superior.
3. La aplicación le muestra un listado con las catástrofes activas.
4. En el encabezado de la tabla, el usuario pulsa el botón de mostrar desactivadas.
5. El sistema carga en la tabla las catástrofes que ya han sido desactivadas indicado la fecha de desactivación de estas.

- Ruta a residencia

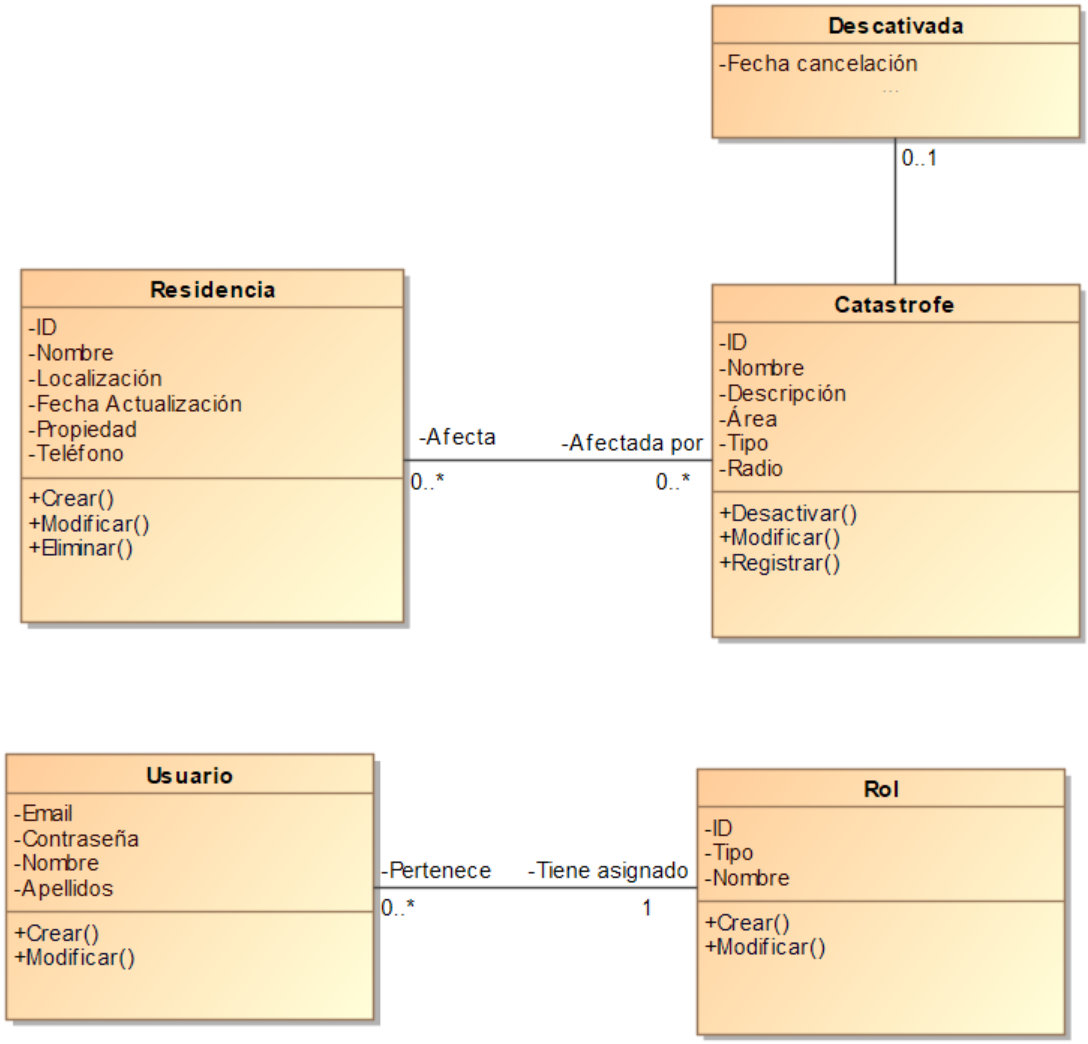
Escenario básico:

1. El usuario accede al sistema, y se le muestra el mapa.
2. El usuario selecciona la residencia a la que desea generar la ruta, y seguidamente pulsa el botón de cómo llegar.
3. El navegador web le solicitará permiso para acceder su ubicación.
4. El sistema genera las trazas correspondientes en el mapa y muestra un listado con las indicaciones correctas para llegar al destino, estimando un tiempo e informando de la distancia.

3.5 Diagrama de clases

Este apartado muestra el modelo de clases del sistema web que se ha desarrollado, y representa cómo se modela la información que constituye la aplicación y la estructuración de los datos.

Figura 4: Diagrama de clases



Fuente: Elaboración propia

Se aprecia en la Figura 4 cómo hay cinco entidades que conforman el sistema. La residencia y catástrofe se conectan mediante la relación directa de que una catástrofe se compone de ninguna, una o un conjunto de residencias. Al igual que la residencia puede pertenecer o no, a una o más catástrofes.

Por otro lado, se modela si la catástrofe está desactivada o no mediante una relación entre estas. Además, existe una relación entre usuarios y roles que determina que todo usuario debe tener asignado un rol, pero un rol puede estar o no asignado a algunos usuarios.

4

Diseño

4.1 Modelado de arquitectura

El sistema esta principalmente basado en el patrón de diseño arquitectónico Modelo-Vista-VistaModelo (MVVM).

Este patrón, el MVVM, nos divide el sistema en tres conjuntos de componentes, Modelos, Vistas y VistaModelo. El uso de esta estructura permite generar enlace directo con el VistaModelo, haciendo que las solicitudes del usuario sean bidireccionales sobre la vista y la VistaModelo.

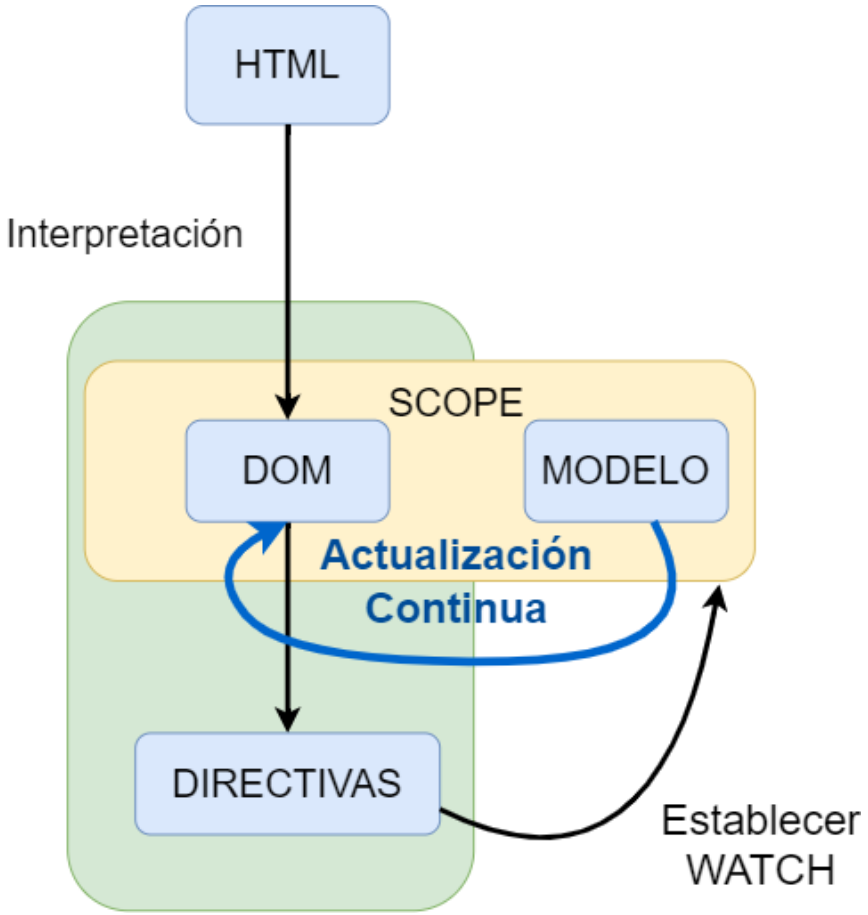
En el proyecto, se ha utilizado AngularJS siendo el framework más novedoso ya que la filosofía en la que se basa es una extensión del código HTML. La comunidad que utiliza tecnologías como esta ha sufrido de un crecimiento exponencial, facilitando el aprendizaje. Los principales conceptos que utiliza son las directivas, servicios proveedores o factorías. El primero de estos, son una serie de directrices que permiten al desarrollador vincular el modelo de Angular a la vista con la VistaModelo. Es muy común encontrar esta característica en los formularios ya que le permiten asociar propiedades del modelo al componente del formulario.

Por otro lado, está la idea de servicios en este framework, su principal utilidad es como proveedor de datos. Esto permite mantener la lógica de acceso a estos y las funcionalidades y operativa que llevan a cabo dentro de la aplicación. Los servicios están destinados a ser utilizados por los componentes, delegando la responsabilidad de acceso a la información y las operaciones sobre los datos.

Dentro de estos podemos tener contenedores que implementan librerías o almacenan datos denominados factorías. Los proveedores o dependencias son inyecciones que se deben realizar en módulos específicos o en componentes para que un servicio funcione correctamente.

AngularJS permite y facilita el diseño en aquellas aplicaciones que adoptan un esbozo fluido y *responsive*, aplicando el modelo MVVM de manera especial. A diferencia del resto de frameworks, este está en constante actualización de la vista, utilizando los conocidos watch. Estos son objetos de Angular encargados de controlar los cambios que se producen en una variable para su reacción inmediata a estos cambios.

Figura 5: Diagrama de flujo Angular

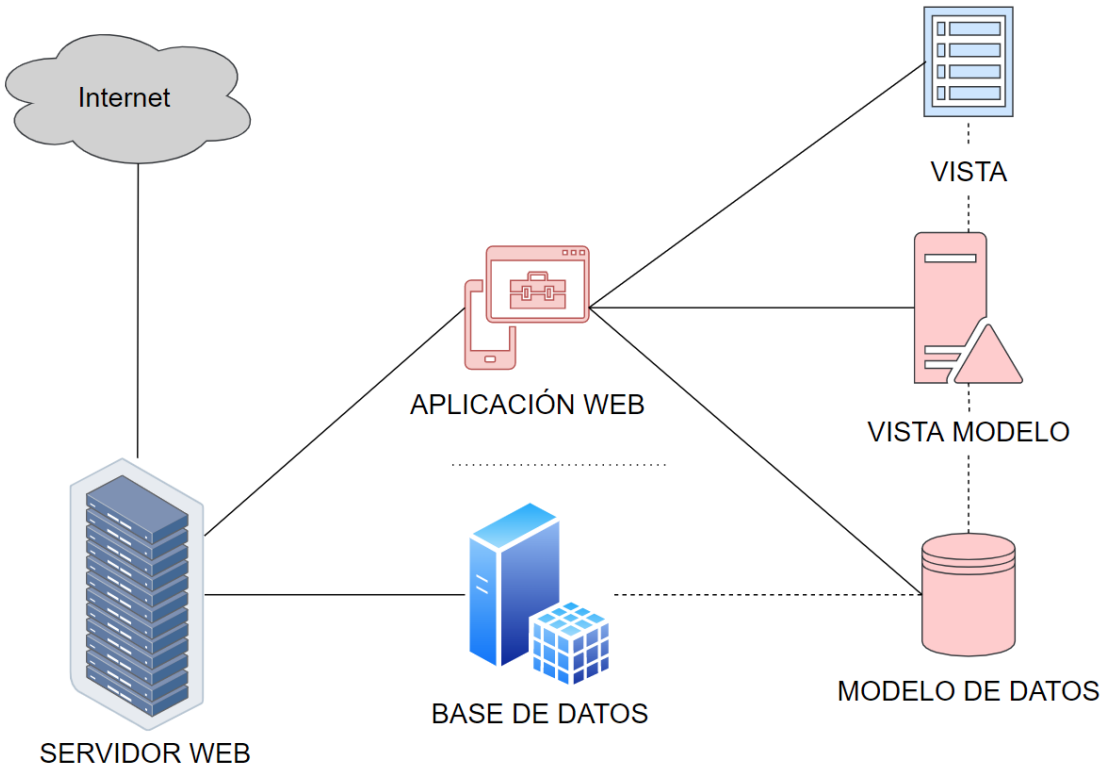


Fuente: Elaboración propia

El modelo representado en la Figura5 presenta una estructura más específica de cómo se ha diseñado arquitectónicamente el front-end de la aplicación. Sin embargo, el sistema se define mediante un modelo arquitectónico más completo donde se recoge también la estructura del servidor y su conexión con la aplicación.

Para ello, se define un modelo completo en el que se muestra cómo la web se implementa y se usa en el entorno final o productivo. En el modelo de la Figura 6, se muestra cómo la base de datos se relaciona con el servidor y el modelo de datos que gestiona la aplicación web. También se muestra cómo el MVVM se integra en el resto de servicios que nos ofrece el servidor web o la base de datos.

Figura 6: Esquema aplicación web



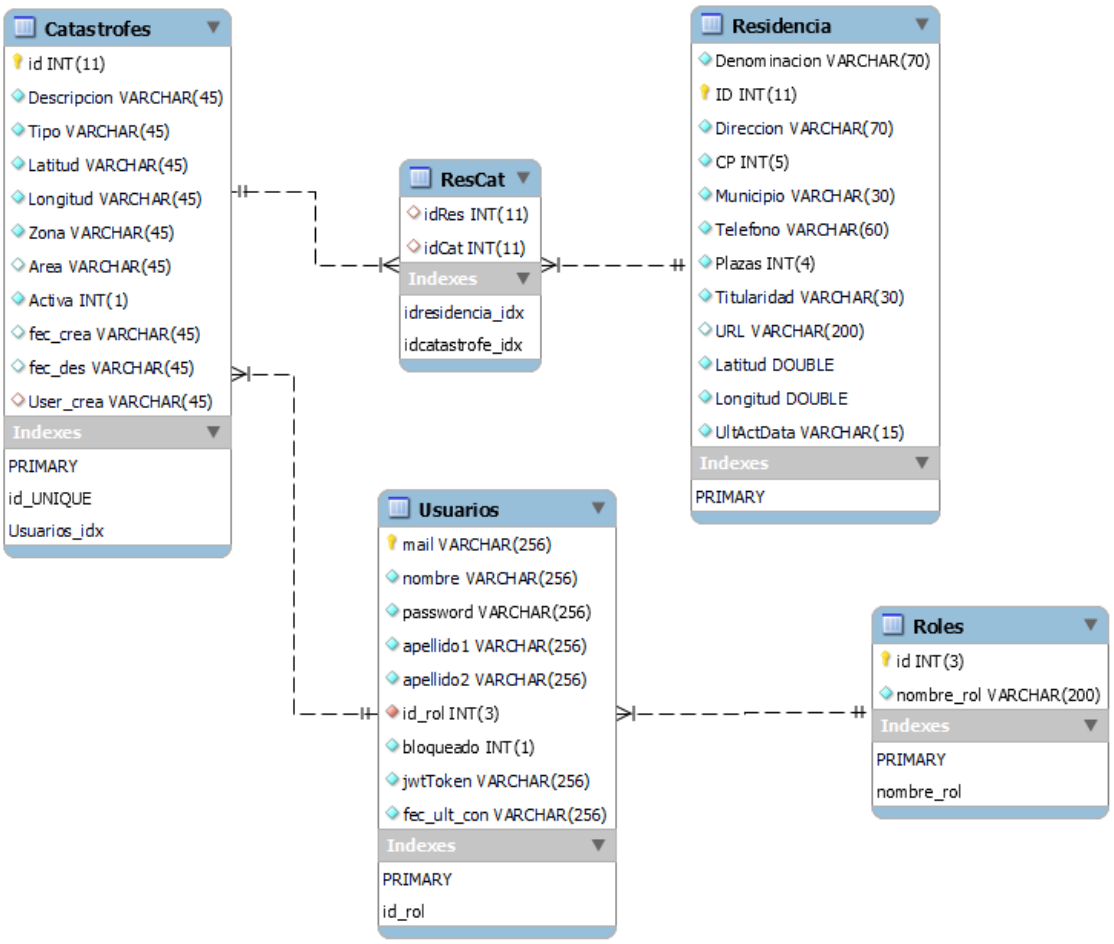
Fuente: Elaboración propia

4.2 Modelado de datos

El proyecto alberga una base de datos relacional utilizada como almacenamiento de datos. Este es uno de los componentes que conforman el modelo del sistema como se explicaba en el capítulo anterior.

La base de datos contiene una serie de tablas que permiten que el almacenamiento de los datos sea el correcto. Gracias a la utilidad de ingeniería inversa de MySQL Workbench se ha realizado el diagrama de la base de datos que representa la Figura 7:

Figura 7: Esquema base de datos



Fuente: MySQLWorkbench

El diagrama representa las tablas que contiene la base de datos y para entenderlas mejor, se describe cada una de ellas a continuación:

- La tabla ***Catástrofes*** representa los sucesos que se registran en el sistema, y se insertan datos como la descripción, el tipo, latitud, longitud, etc. La clave primaria es autogenerada según se van aumentado las catástrofes registradas. Se lleva a cabo también un control de la fecha de creación e inhabilitación de estas. También registra el usuario que la añade en la base de datos manteniendo una relación con la tabla usuarios para asegurar que la crea un usuario del sistema.

- La tabla **Usuarios** contiene todos los perfiles que acceden a la plataforma. Incluye todos los datos personales, nombre, apellidos… En este caso, se ha definido como clave primaria el mail que además es usado como el id

de acceso. Cada usuario debe tener asignado un rol definido en la base de datos, como bien indica el diagrama, se agrega una relación entre dichas tablas.

- La tabla ***Residencia*** se utiliza para tener registradas todos los centros de mayores, y recogen toda la información. Se debe registrar la descripción, la dirección, titularidad, número de plazas, localización mediante coordenadas, entre otros. El identificador único o clave primaria de la tabla es un id numérico generado automáticamente que crece secuencialmente según se insertan residencias.

- La tabla ***Roles*** define los diferentes grupos de permisos o competencias que son asignadas a un grupo de usuarios. Para ello se define un nombre de rol y se le asigna un identificador único. Cada uno de ellos es compartido por un número de usuarios del sistema.

- La tabla ***ResCat*** se utiliza para generar una relación entre residencias y catástrofes. Tras el registro de una catástrofe, el sistema añade entradas en la tabla que relacionan cada catástrofe con un número determinado de residencias. Debido a lo anterior, se crean relaciones entre esta tabla y catástrofes y de la misma forma con la tabla residencias para que la inserción sea la adecuada y de elementos existentes en el sistema.

4.3 Interfaz de Usuario

La interfaz de usuario es la parte visual y el medio que permite la interacción entre el usuario y el sistema, software, dispositivo o sitio web. Por lo tanto, debe ser sencilla de usar, rápida de comprender y aprender para que se intuitivo mejorando la interacción del usuario.

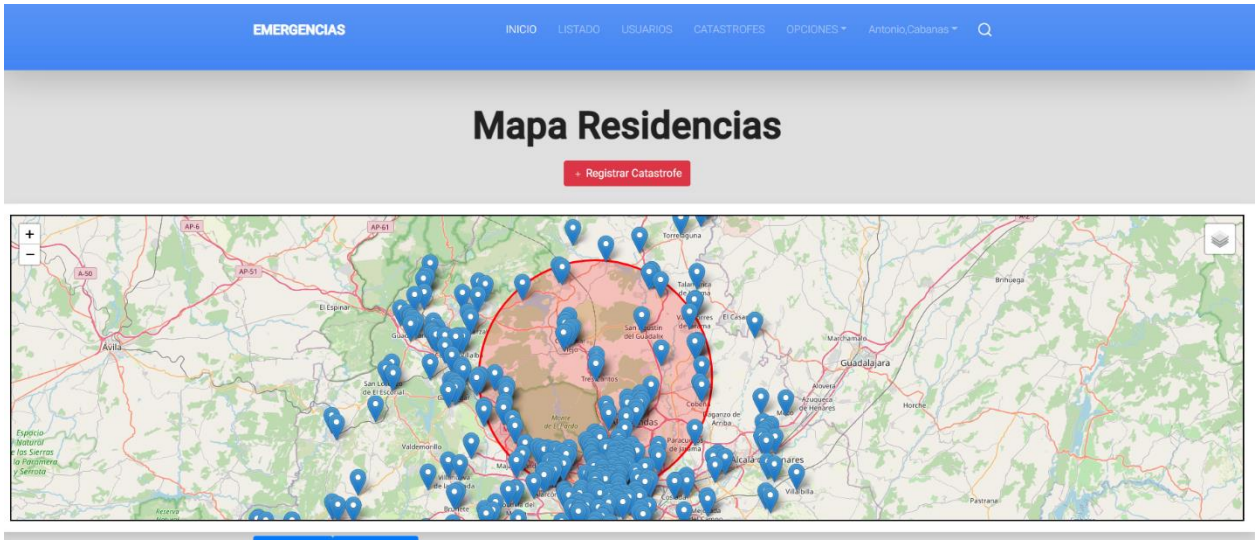
Esta interfaz se compone de un conjunto de elementos y características, como botones, iconos, cuadros de texto, imágenes… Para realizar un buen desarrollo de la

vista se tendrán en cuenta las técnicas de diseño visual y gráfico. Siguiendo estos patrones se harán buenas elecciones de colores, representando el significado y las emociones adecuadas. La tipografía es muy importante también ya que debe ser de lectura sencilla, y se consigue ajustando los tamaños y el tipado.

Los principales aspectos que se deben seguir en el diseño son la estructura que mantiene la organización, la simplicidad que elimina la complejidad y la visibilidad que evita mostrar información irrelevante. Existen otros también con gran importancia como la retroalimentación, informando al usuario de todos los movimientos que va realizando para evitar que cometa errores, o la reutilización que replica los componentes o comportamientos de la interfaz reduciendo la necesidad de recordar diferentes métodos al usuario.

Tras analizar las características más importantes que deben aplicarse en el diseño mostraremos diferentes pantallas para mostrar las características necesarias.

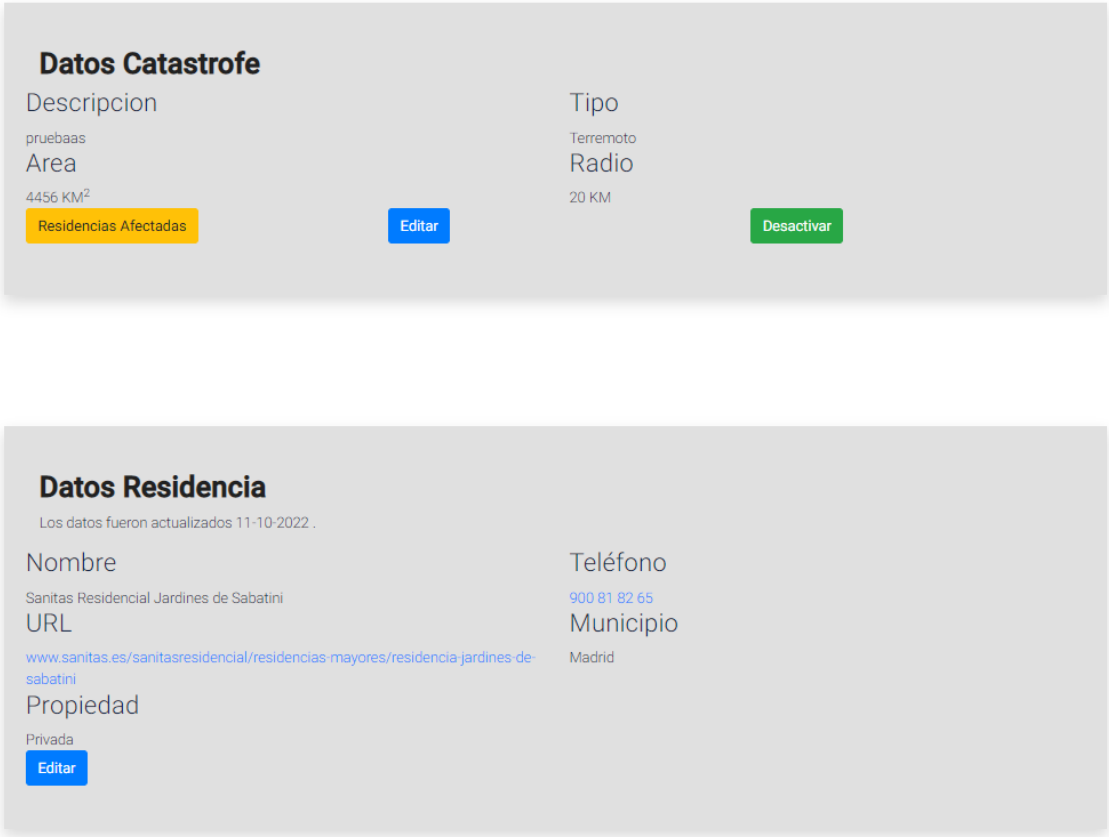
Figura 8: Interfaz principal del mapa



Fuente: Aplicación web

La pantalla de la Figura 8 nos muestra la vista principal donde se ve el mapa, y representa en su interior el área afectada por la catástrofe mediante un círculo rojo. La sencillez del mapa permite al usuario que se informe rápidamente de las zonas en estado de emergencia. También se muestra un menú principal en la aplicación que permite navegar al usuario, y contiene todas las secciones del sistema.

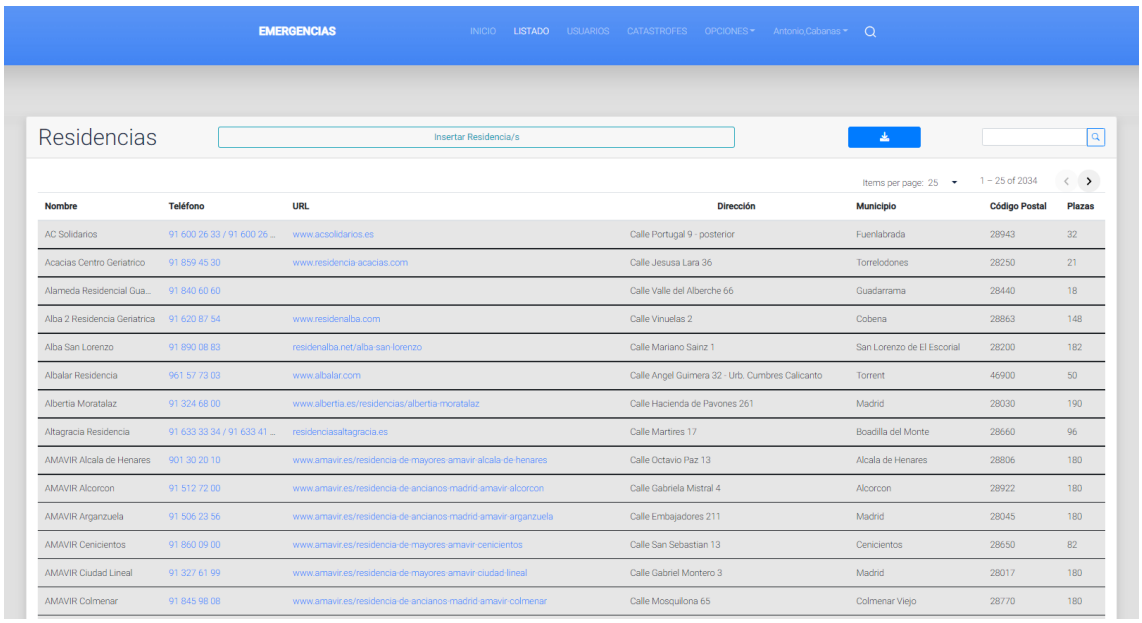
Figura 9: Cuadros de información



Fuente: Aplicación web

La información de las residencias y las catástrofes se muestran en la parte inferior del mapa representando los datos básicos de estas como se muestra en la Figura 9. Se incluyen botones que permiten realizar alguna funcionalidad. La sencillez que se aplica en este caso evita la carga de información y complejidad de uso para el usuario.

Figura 10: Listado de residencias



Fuente: Aplicación web

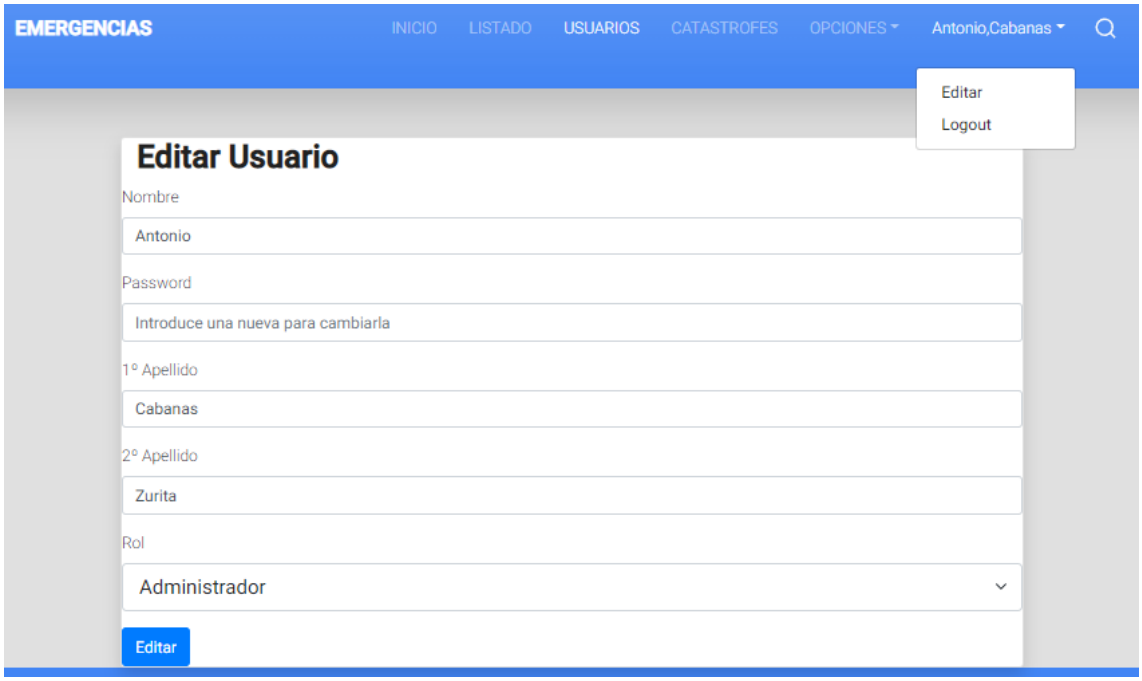
Los listados de la aplicación (Figura 10) adaptan la vista de la imagen superior, se muestran los datos en una tabla dinámica que incluye búsqueda y el número de filas que se muestran o funcionalidades para pasar de página. Por otro lado, la cabecera de la tabla incluye un botón para acceder al formulario de inserción de datos y otro para la descarga en formato PDF de los datos visualizados en la tabla.

Figura 11: Formulario insertar residencia

Fuente: Aplicación web

El sistema incluye formularios (Figura 11) que permiten introducir los datos necesarios en cada caso. La imagen superior muestra el formulario que registra una residencia. Este formulario es más completo e incluye un registro masivo de datos mediante un fichero CSV facilitando al usuario el formato que debe tener.

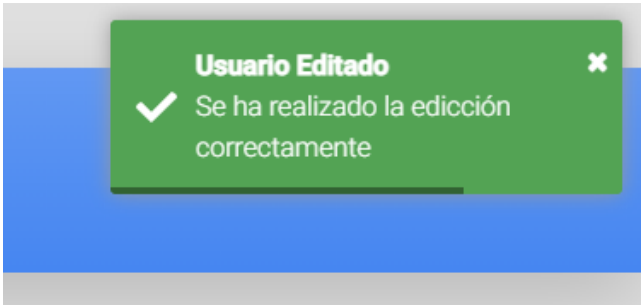
Figura 12: Formulario de edición



Fuente: Aplicación web

El usuario dispone de una vista completa de su perfil mostrando sus datos en la aplicación y con capacidad de modificar su contraseña de acceso o nombre. El sistema informa que usuario ha accedido al sistema, indicado su nombre en el menú principal. Además, se incluye un desplegable que permite seleccionar un rol de los existentes y cuenta con una herramienta de notificaciones push (Figura 13):

Figura 13: Notificación push



Fuente: Aplicación web

5

Implementación y Pruebas

5.1 Desarrollo

La fase de desarrollo del proyecto ha sido la más extensa, debido a la cantidad de tareas definidas en esta sección. El sistema se ha programado siguiendo las estructuras y tecnologías definidas en capítulos anteriores.
En este capítulo se describe cómo se ha desarrollado cada uno de los componentes que conforman el sistema completo.

En primer lugar, se aborda el desarrollo front-end explicando cada uno de los componentes que lo conforman.

- **Mapa:**
  El mapa se define con *OpenStreetMap* y la librería leaflet creando el componente que lo forma. En primer lugar, se deben asignar los parámetros

correctos para generar el mapa adecuadamente, indicando el ancho, tamaño, centro, etc. Se pueden ver estos datos básicos del mapa en la Figura 14.

Figura 14: Definición del mapa

```
var iconDefault = this.mapService.L.icon({
  iconRetinaUrl,
  iconUrl,
  shadowUrl,
  iconSize: [25, 41],
  iconAnchor: [12, 41],
  popupAnchor: [1, -34],
  tooltipAnchor: [16, -28],
  shadowSize: [41, 41]
});
  this.mapService.L.Marker.prototype.options.icon = iconDefault;


  this.map =  this.mapService.L.map('map', {
    center: [DEFAULT_LAT, DEFAULT_LON],
    attributionControl: false,
    zoom: 10
  });
```

Fuente: Código fuente

Después de incluir las características básicas del mapa, se pueden añadir marcas o chinchetas. En este caso van a representar las residencias registradas en el sistema y se le asignarán funcionalidades para que al pulsar en estas muestre su información. Los marcadores de las residencias en el mapa se definen en la aplicación según la Figura 15.

Figura 15: Definición de residencias en el mapa

```
this.apiService.searchResidencias().subscribe((residencias: Residencia[]) => {
  this.residencias = residencias;
//Creamos una marcaa para cada residencia y agregamos la funcion a realizar en el click.

this.residencias.forEach((element: Residencia) => {

  this.mapService.L.marker([element.Latitud,element.Longitud]).bindPopup(element.Denominacion).addTo(this.map).on('click',(e: { latlng: any; }) => {
    this.ResidenciainfoService.toggle(element);
    this.map.setView(new L.LatLng(element.Latitud, element.Longitud),12);
    this.markerlat = element.Latitud;
    this.markerlon = element.Longitud;
  });
});
});
```

Fuente: Código fuente

Otro elemento importante son las zonas afectadas por la catástrofe que se muestran en el mapa. Se agrega otro elemento de la librería leaflet, el 'cricle' donde se indican los parámetros de la catástrofe para su representación en el mapa como se aprecia en la Figura 16.

Figura 16: Definición de catástrofe en el mapa

```
this.apiService.searchCatastrofes().subscribe((catastrofes: Catastrofe[]) => {
  this.catastrofes = catastrofes;

  this.catastrofes.forEach((element: Catastrofe) => {
    if(element.Activa == 1){
      const mark = this.mapService.L.circle([element.Latitud,element.Longitud],{color: 'red', radius: (element.Zona*1000)}).addTo(this.map);
      mark.on(
        "click",
        () => {
          this.CatastrofeinfoService.toggle1(element);
        });
      mark.bindPopup(element.Descripcion);
      mark.addTo(this.map);
    }
  });
});
//radius directamente son los metros del circulo
```

Fuente: Código fuente

La última utilidad agregada al mapa de la aplicación son las capas que muestran diferentes características del terreno o de la zona sobre la que se dibujan. Estas capas son agregadas en la interfaz como se muestra en la Figura 17. La información que representan es obtenida desde servicios web que están disponibles para su uso, algunos de ellos se han obtenido de sitos web del gobierno de España con información en abierto.

Figura 17: Definición de capas del mapa

```
const fallas3 = this.mapService.L.tileLayer.wms("https://mapas.igme.es/gis/services/BasesDatos/IGME_QAFI/MapServer/WMSServer?", {
    layers: "3",//nombre de la capa (ver get capabilities)
    format: 'image/png',
    transparent: true,
    version: '1.3.0',//wms version (ver get capabilities)
    attribution: "PNOA WMS. Cedido por © Instituto Geográfico Nacional de España"
});


const inundable10 = this.mapService.L.tileLayer.wms("https://servicios.idee.es/wms-inspire/riesgos-naturales/inundaciones?service=WMS&", {
    layers: "NZ.Flood.FluvialT10",//nombre de la capa (ver get capabilities)
    format: 'image/png',
    transparent: true,
    version: '1.3.0',//wms version (ver get capabilities)
    attribution: "Peligrosidad asociada a las Zonas Inundables por inundación fluvial, correspondientes a un escenario de probabilidad alta de inundación (periodo
});
```

Fuente: Código fuente

Además, el sistema es capaz de generar rutas desde un punto a otro utilizando el trazado correcto y permitiendo seguir esta ruta mediante vehículos. Para ello se ha utilizado el servicio web de rutas Project-OSMR (*Open Source Routing Machine*, Máquina de Enrutamiento de Código Libre) es un servidor de enrutamiento diseñado para usarse con los datos de OpenStreetMap), creando una guía en texto que nos indica el camino a seguir. También ha sido necesario obtener la localización del usuario mediante la dirección IP para poder generar la ruta correcta entre el usuario y la localización de la residencia. El algoritmo que hace posible la generación de la ruta se describe en la Figura 18.

Figura 18: Definición de la ruta

```
public ruteLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition((position) => {
      if (position) {
        console.log("Latitude: " + position.coords.latitude +
          "Longitude: " + position.coords.longitude);
        this.latloc = position.coords.latitude;
        this.lngloc = position.coords.longitude;

        if(this.routingline === undefined){

        }else{
          this.routingline.remove(this.map);
        }
       this.routingline = this.mapService.L.Routing.control({
          router: this.mapService.L.Routing.osrmv1({
            serviceUrl: `https://router.project-osrm.org/route/v1/`
          }),
          showAlternatives: true,
          fitSelectedRoutes: true,
          show: true,
          routeWhileDragging: true,
          waypoints: [
            this.mapService.L.latLng(this.latloc, this.lngloc),
            this.mapService.L.latLng(this.markerlat,this.markerlon)
          ]
        }).addTo(this.map);
      }
    },
      (error) => console.log(error));
  } else {
    alert("Geolocation is not supported by this browser.");
  }
}
```

Fuente: Código fuente

- **Login**

  El sistema de acceso a la aplicación se encarga de autenticar al usuario. Se ha desarrollado utilizando un token aumentado las características de seguridad. Se utiliza un formulario que recibe los datos introducidos, y los envía al back-end del sistema para verificar la autenticidad de la información. Tras procesar los datos se gestiona si se puede acceder a la plataforma. Se puede observar en la Figura 19 cómo se ha programado el incio de sesión, haciendo uso del servicio de AuthService que se define en angular para gestionar si el usuario ha accedido al sistema o no.

Figura 19: Definición de inicio de sesión

```
onSubmit(): void {
  const { username, password } = this.form;
  this.apiService.login(username, password).subscribe(
    (data:any[]) => {
      if(data[0]){
        this.authService.sendToken(data[0]["jwtToken"], data[0]["mail"], data[0]["nombre"], data[0]["apellido1"], data[0]["apellido2"], data[0]["nombre_rol"]);
        let tokens = this.authService.getToken();
        this.router.navigate(['/mapa']).then(() => {
         window.location.reload();
        });
        this.isLoginFailed = false;
        this.isLoggedIn = true;
        console.log("me he logeado");
        location.reload();
       }else{
        this.isLoginFailed = true;
       }
    },
    (err:any) => {
      if(err === 'Locked'){
        this.errorMessage = "Usuario no registrado";
      } else{
        this.errorMessage = "Contraseña incorrecta"
      }
      this.isLoginFailed = true;
  });
}
```

Fuente: Código fuente

El servicio mencionado anteriormente, AuthService, se ocupa de gestionar la autenticidad del usuario durante el uso de la aplicación. Este servicio genera un token que comparte e identifica al usuario, permitiendo al resto de componentes que lo usen para sus funciones. Se incluyen también diferentes métodos que resuelven cuestiones relacionadas con el token o con características identificativas mostrados en la Figura 20.

Figura 20: Función login

```
login(username: string, password: string): Observable<any> {
  return this.http.post<any>(`${environment.baseUrl}/api/login`, { username, password }, { withCredentials: true })
      .pipe(map(user => {
          this.userSubject.next(user);
          this.startRefreshTokenTimer();
          console.log("Llamo al login de auth con user:");
          console.log(user);
          return user;
      }));


}
private stopRefreshTokenTimer() {
  clearTimeout(this.refreshTokenTimeout);

apilogout() {
  this.http.post<any>(`${environment.baseUrl}/users/revoke-token`, {}, { withCredentials: true }).subscribe();
  this.stopRefreshTokenTimer();
  this.router.navigate(['/login']);


sendToken(token: string, mail: string, name: string, surname1: string, surname2: string,
  rol: string) {
  localStorage.setItem("LoggedInUser", token)
  localStorage.setItem("id_user", mail)
  localStorage.setItem("name_user", name)
  localStorage.setItem("surname_user", surname1)
  localStorage.setItem("surname_user2", surname2)
  localStorage.setItem("role",rol)
  console.log("envio token");

}
```

Fuente: Código fuente

- **Información de la catástrofe o residencia**

  El sistema muestra ventanas que se cargan tras la selección de una residencia o una catástrofe en el mapa. Estas ventanas son dinámicas y se muestran cuando se detecta el clic del ratón en la catástrofe o residencia determinada. Se muestra en la Figura 21 el servicio que comunica estos cambios y transfiere los datos necesarios y la propiedad EventEmitter que detecta estos cambios.

Figura 21: Evento de residencia

```
@Output() change: EventEmitter<boolean> = new EventEmitter();

toggle1(catastrofe:any) {
  this.isOpen1 = !this.isOpen1;
  this.catastrofe = catastrofe;
  this.change.emit(this.isOpen1);
  this.change.emit(this.catastrofe);
  console.log(catastrofe);
  console.log("toggle1");
}
```

Fuente: Código fuente

Este componente contiene métodos que realizan funcionalidades sobre la aplicación. En la Figura 22 se presentan las funciones que permiten desactivar las catástrofes o eliminarlas, se usa la misma metodología para las residencias. Para llevar a cabo estos métodos es utilizado un identificador que distingue entre el conjunto de datos posible para mostrar.

Figura 22: Desactivar una catástrofe

```
public desac(idcat: any){
  this.apiService.desactivarCatastrofe(idcat).subscribe((res:any) => {
    if(res == 200){
      console.log("correcta");
      this.toastrService.success('Se ha desactivado la incidencia', 'Catastrofe Desactivada');
      setTimeout(() => {
        location.reload();
    }, 5000);
    }else{
      this.toastrService.error('Se ha detectado un error', 'Descactivación Fallida');
    }
  });
}
public  editar(id:number) {
  this.router.navigateByUrl('/editar-catastrofe', { state: {id : id.toString()} });

}
```

Fuente: Código fuente

- **Listado**

  Los listados en la aplicación son habituales para representar un conjunto de datos y estructurarlos dentro de esta. La Figura 23 es un ejemplo de carga del

listado mediante un vector de datos representado con HTML y CSS en la web. Sin embargo, no es suficiente para que la tabla se adapte correctamente a la web. Por lo tanto, son utilizados métodos que gestionan el número de filas o columnas mostradas.

Figura 23: Mostrar listado residencias

```
ngOnInit(): void {
  this.apiService.searchResidenciasOrder().subscribe((residencias: Residencia[]) => {
    this.residencias = residencias;
  });
}
handlePage(e: PageEvent){
  this.page_size = e.pageSize;
  this.page_number = e.pageIndex+1;
}
```

Fuente: Código fuente

Otra característica destacable es la búsqueda dentro del listado que permite al usuario filtrar por cualquier valor. Esta utilidad se aplica mediante los condicionales de Angular inyectados en código HTML mostrados en la Figura 24.

Figura 24: Tabla del listado de residencias

```
<table id="tablalistado" class="table table-striped table-hover" *ngIf="(residencias | filter: searchText).length > 0; else noResults">
    <thead>
        <tr>
            <th>Nombre</th>
            <th>Teléfono</th>
            <th>URL</th>
            <th class="text-center">Dirección</th>
            <th>Municipio</th>
            <th>Código Postal</th>
            <th>Plazas</th>
        </tr>
    </thead>
    <tbody *ngFor="let item of residencias | filter: searchText | paginate:page_size:page_number;">
        <tr>
            <td class="text-truncate" style="max-width: 200px;">{{item.Denominacion}}</td>
            <td class="text-truncate" style="max-width: 200px;"><a href="{{'tel:+34' + item.Telefono}}">{{item.Telefono}}</a></td>
            <td><a href="{{'https://' + item.URL}}">{{item.URL}}</a></td>
            <td>{{item.Direccion}}</td>
            <td>{{item.Municipio}}</td>
            <td>{{item.CP}}</td>
            <td>{{item.Plazas}}</td>
```

Fuente: Código fuente

- **Insertar**

  El registro de datos es muy importante para mantener la aplicación con la información correcta. Se ha desarrollado utilizando formularios y métodos que validan la información, gracias a Angular se simplifica a elaboración de estos. Se utiliza los 'angular forms' que incluyen la utilidad 'validators' permitiendo verificar y recoger los datos introducidos en el formulario. La

Figura 25 hace uso de las utilidades mencionadas, y muestra la definición de los campos del formulario.

Figura 25: Formulario para insertar una residencia

```
export class InsertarResidenciaComponent implements OnInit {

  denominacion =  new FormControl('', [Validators.required]);
  titularidad =  new FormControl('', [Validators.required]);
  municipio =  new FormControl('', [Validators.required]);
  direccion =  new FormControl('', [Validators.required]);
  cp =  new FormControl('', [Validators.required]);
  telefono =  new FormControl('', [Validators.required]);
  plazas =  new FormControl('', [Validators.required]);
  url =  new FormControl('',);
  longitud =  new FormControl('', [Validators.required]);
  latitud =  new FormControl('', [Validators.required]);
```

Fuente: Código fuente

También mediante el 'FormControl' de Angular utilizado en la Figura 26 es creado el controlador que recolecta la información de los cuadros de texto entrantes.

Figura 26: Obtención datos formulario

```
ngOnInit(): void {
  if(this.auth.getRole() == "Administrador" || this.auth.getRole() == "Editor Residencias"){

  }else{
    this.route.navigate(['mapa']);
  }
  this.denominacion.setValue('');
  this.titularidad.setValue('');
  this.municipio.setValue('');
  this.direccion.setValue('');
  this.cp.setValue('');
  this.telefono.setValue('');
  this.plazas.setValue('');
  this.url.setValue('');
  this.longitud.setValue('');
 this.latitud.setValue('');

}
```

```
onsubmit() {
  this.apiService.insertarres(this.denominacion.value, this.titularidad.value,
    this.municipio.value, this.direccion.value, this.cp.value, this.telefono.value,
    this.plazas.value, this.url.value, this.longitud.value, this.latitud.value ).subscribe(
    (res:any) => {
      if(res == 200){
        console.log("correcta");
        this.toastrService.success('Se ha creado correctamente', 'Residencia Creada');
      }else{
        this.toastrService.error('Se ha detectado un error', 'Creación Fallida');
      }
    }
  );
}
```

Fuente: Código fuente

- **Importar CSV**

  La aplicación permite la inserción de información mediante ficheros en el formato CSV predefinido. Para ello se ha empleado un formulario y métodos que captan ficheros introducidos en el navegador mostrados en la Figura 27.

Figura 27: Importación mediante formato csv

```
uploadFile(event:any){
  const file = event.target.files ? event.target.files[0] : '';
  console.log(file);
  this.form.patchValue({
    image: file
  });
  this.form.get('image')?.updateValueAndValidity();
}
```

Fuente: Código fuente

El método encargado de recibir el fichero captura el evento que lo carga para asignar los valores correctos.

- **Editar**

  El sistema permite editar los datos registrados para una catástrofe, usuario o residencia. La Figura 28 muestra la carga del formulario con los datos de la residencia que se desea modificar. El resto de objetos a modificar utilizan la misma metodología.

Figura 28: Formulario de edición

```
this.router.paramMap.subscribe ( parametro =>
  {
    this.id = history.state.id;
    this.apiService.searchResidenciaID(this.id).subscribe((residencias: Residencia[]) => {
     this.residencia = residencias[0];
     this.denominacion.setValue(this.residencia.Denominacion),
    this.titularidad.setValue(this.residencia.Titularidad),
    this.municipio.setValue(this.residencia.Municipio),
    this.direccion.setValue(this.residencia.Direccion),
    this.cp.setValue(this.residencia.CP),
    this.telefono.setValue(this.residencia.Telefono),
    this.plazas.setValue(this.residencia.Plazas),
    this.url.setValue(this.residencia.URL),
    this.longitud.setValue(this.residencia.Longitud),
   this.latitud.setValue(this.residencia.Latitud)
    });


});
```

Fuente: Código fuente

Este método es similar al registro de datos por lo tanto utiliza el mismo recurso de los formularios de angular para recolectar los datos por pantalla. Sin embargo, se deben mostrar los datos que contiene el objeto para ello utilizan el servicio de *router* que comparte el identificador del objeto. Y tras obtener del back-end la información, se le agrega esta información a cada campo del formulario.

- **Menú**

  El panel de menú desarrollado en la Figura 29 permite navegar dentro de la aplicación, y lo hace mediante los vínculos indicados por el servicio *router*.

Figura 29: Menú de la aplicación

```
<ul *ngSwitchCase="'listado'" class="ms-auto navbar-nav">
    <li class="nav-item"> <a class="nav-link px-lg-3 py-lg-4 text-uppercase" aria-current="page" (click)="viewMode = 'inicio'" routerLink="/mapa"><p style="
    </li>
    <li class="nav-item"> <a class="active nav-link px-lg-3 py-lg-4 text-uppercase" (click)="viewMode = 'listado'" routerLink="/listado">Listado</a>
    </li>

    <li class="nav-item"> <a class="nav-link px-lg-3 py-lg-4 text-uppercase" (click)="viewMode = 'catastrofes'" routerLink="/catastrofes">Catastrofes</a>
    </li>
```

Fuente: Código fuente

Además, se han definido estilos diferentes por cada opción del menú, angular permite generar casos específicos y actuar de forma diferente.

- **Datos del usuario**

  La aplicación muestra los datos del usuario que está haciendo uso de la aplicación, se muestra desde el menú usando los recursos del servicio AuthService con la función mostrada en la Figura 30.

Figura 30: Función que comprueba datos de usuario

```
if(this.auth.isLoggedIn()){
  this.usuario = this.auth.getNameUser() +','+this.auth.getSurnameUser();
}
```

Fuente: Código fuente

Además, el usuario es capaz de modificar algunos de sus datos como la contraseña o el nombre. Se hará de igual forma que al editar un usuario mediante el formulario de edición, pero en este caso solo podrá acceder a sus datos.

- **Gestión del rol**

  El sistema cuenta con un conjunto de roles diferenciados para aplicar funciones diferentes a cada grupo de usuarios algunas de ellas mostradas en la Figura 31. La plataforma lo gestiona desde el servicio AuthService que almacena los datos del usuario, obteniendo el rol de este.

Figura 31: Función que comprueba el rol

```
ngOnInit(): void {
  if(this.auth.getRole() == "Administrador"){

  } else {
    this.route.navigate(['mapa']);
  }
  this.apiService.searchRolesOrder().subscribe((roles: Rol[]) => {
    this.roles = roles;
  });
 //bloquear el campo de rol si no es admin
  if(this.auth.getRole() != "Administrador"){
    this.rol.disable();
  }
}
```

Fuente: Código fuente

También se aplican cambios en la vista del sistema modificando los componentes según el rol que tiene definido. Hay dos metodologías aplicadas, la primera de ellas es directamente sobre el control definiendo funcionalidades. Y la segunda, es inyectando las condiciones al fichero de la vista que aplica recursos de angular sobre este. En la Figura 32 se muestra.

Figura 32: Bloque de control para roles

```
<ng-template #elseBlock>
    <div *ngIf="rol(); then rolBlock else elserol"></div>
    <ng-template #rolBlock>
    <div class="bg-light py-1">
    <div class="container">
        <div class="align-items-center g-0 row">
            <div class="col">
                <div class="py-1">
```

Fuente: Código fuente

La figura superior muestra cómo se gestiona el bloque condicional que muestra el bloque que cumpla la condición.

- **Descarga PDF**

  El usuario puede obtener un PDF de algunos listados de la aplicación, y para ello se han utilizado librerías jsPDF. Este recurso obtiene la tabla completa y es capaz de representarla completa generando un pdf con la función descrita en la Figura 33.

Figura 33: Función descarga PDF

```
download() {
  var pdf = new jsPDF('l');
  (pdf as any).autoTable({
    orientation: 'landscape',
  html: '#tablalistado'
  })
  // Open PDF document in browser's new tab
  //pdf.output('dataurlnewwindow')
  // Download PDF doc
  pdf.save('Residencias.pdf');
}
```

Fuente: Código fuente

- **Router service**

  La aplicación realiza la navegación que el usuario demanda, y se deben definir las rutas y los ficheros relacionados a cada uno de ellos. La estructura de las aplicaciones Angular como se ha definido en capítulos anteriores, están formadas por un conjunto de componentes. Este servicio crear vínculos a cada componente como se aprecia en la Figura 34.

Figura 34: Definición de rutas estáticas

```
const routes: Routes = [
  { path: 'insert', component: InsertarResidenciaComponent,  canActivate: [AuthGuard] },
  { path: 'listado', component: ListadoComponent,  canActivate: [AuthGuard]},
  { path: 'mapa', component: MapaComponent,  canActivate: [AuthGuard] },
  { path: 'editar-residencia', component: EditarResidenciaComponent, canActivate: [AuthGuard]},
  { path: 'editar-usuario', component: EditarUsuarioComponent, canActivate: [AuthGuard]},
  { path: 'insertar-usuario', component: InsertarUsuarioComponent, canActivate: [AuthGuard]},
  { path: 'usuarios', component: ListadoUsuariosComponent, canActivate: [AuthGuard]},
  { path: 'afectadas', component: ListadoAfectadasComponent, canActivate: [AuthGuard]},
  { path: 'insertar-catastrofe', component: InsertarCatastrofeComponent, canActivate: [AuthGuard]},
  { path: 'catastrofes', component: ListadoCatastrofesComponent, canActivate: [AuthGuard]},
  { path: 'editar-catastrofe', component: EditarCatastrofeComponent, canActivate: [AuthGuard]},
  { path: '**', component: LoginComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes, { useHash: true} )],
  exports: [RouterModule]
})
```

Fuente: Código fuente

Existen más aplicaciones del servicio, añadiendo identificadores en su interior y permitir al componente actuar correctamente indicado en la Figura 35. Se realiza añadiendo un estado a la ruta.

Figura 35: Ruta con identificador incrustado

```
public  editar(id:number) {
  this.router.navigateByUrl('/editar-residencia', { state: {id : id.toString()} });

}
```

Fuente: Código fuente

Se puede obtener el parámetro en el componente destino mediante la historia del servicio, y acceder al estado mediante los datos definidos en la Figura 36.

Figura 36: Obtención de identificador de la ruta

```
this.router.paramMap.subscribe ( parametro =>
  {
    this.id = history.state.id;
```

Fuente: Código fuente

- **ToastService**

  Este servicio es utilizado por el sistema para generar notificaciones push y poder informar al usuario del resultado de una tarea ejecutada en la aplicación. Se realiza directamente llamando al servicio e indicando el tipo de notificación que deseamos y le agregamos la información que debe mostrar como muestra la Figura 37.

Figura 37: Servicio de notificaciones

```
(res:any) => {
  if(res == 200){
    console.log("correcta");
    this.toastrService.success('Se ha realizado la edicción correctamente', 'Residencia Editada');
  }else{
    this.toastrService.error('Se ha detectado un error', 'Edicción Fallida');
  }
}
```

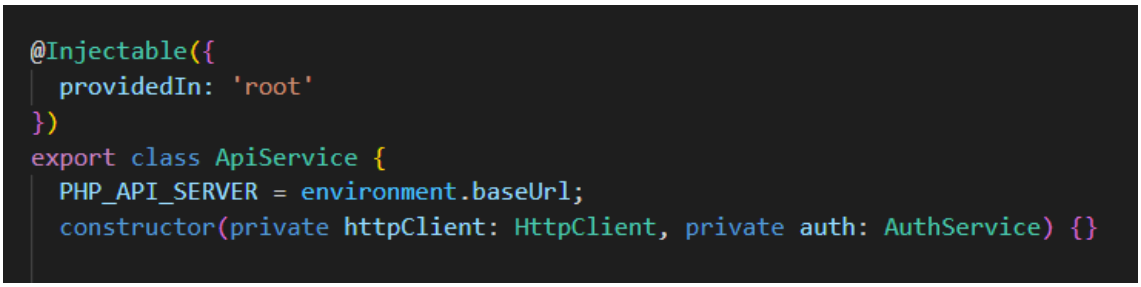
Fuente: Código fuente

- **ApiService**

  El servicio definido en la Figura 38, es el controlador principal del sistema. Se utiliza para conectar las peticiones del sistema con el back-end, transfiriendo los datos entre ambas partes.

El desarrollo del servicio incluye recursos del cliente http de angular que permite realizar peticiones POST, GET o PUT para el intercambio de información.

Figura 38: Servicio HTTP



```
@Injectable({
  providedIn: 'root'
})
export class ApiService {
  PHP_API_SERVER = environment.baseUrl;
  constructor(private httpClient: HttpClient, private auth: AuthService) {}
```

Fuente: Código fuente

Para el uso del servicio, primero se han creado los métodos que realizan las peticiones donde se indica la cabecera definida en la Figura 39, se incluye el token de autorización, y las opciones de la llamada.

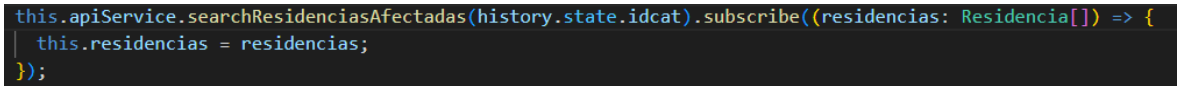
Figura 39: Petición realizada a la API

```
searchResidenciasAfectadas(idcat:string):any {
  const headers = { 'Authorization': 'Bearer '+this.auth.getToken() }
  let options = { headers: headers };

  return this.httpClient.post(`${this.PHP_API_SERVER}/api/buscarResidenciasAfectadas.php`, {
    idcat: idcat,
  }, options);
}
```

Fuente: Código fuente

La Figura 40 muestra los métodos que se llaman desde los componentes con una serie de parámetros para obtener de esta petición los datos deseados. Almacenando los datos recogidos en un objeto adecuado para estos.

Figura 40: Ejemplo de uso de la API



```
this.apiService.searchResidenciasAfectadas(history.state.idcat).subscribe((residencias: Residencia[]) => {
  this.residencias = residencias;
});
```

Fuente: Código fuente

En segundo lugar, tras exponer todas las características que conforman el front-end, continuamos con el desarrollo back-end del sistema. Este contenido se ha

desarrollado en diferentes lenguajes de programación para abordar todas las tareas necesarias.

Este parte de la estructura es la encargada de realizar las consultas a base de datos, realizar los cálculos y análisis adecuados de la información. En esta sección se aborda la lógica interna del sistema que desconoce el usuario, y realiza las funcionalidades vitales para que los datos se guarden o se muestren los que este busca.

Del mismo modo se explicarán cada uno de los elementos que estructuran el back-end.

- **Configuración**

  El conjunto de módulos que se utilizan para realizar las funcionalidades en servidor utiliza ficheros de configuración que definen datos de conexión y abren las conexiones con la BBDD mostrados en la Figura 41. Para este proyecto se ha utilizado un fichero de 'database' que define los datos para conectar a la base de datos. Desde PHP se disponen de métodos que facilitan abrir las conexiones enviando los parámetros correctos. Además, se utiliza el estándar UFT-8 para la codificación de caracteres bajo un estándar.

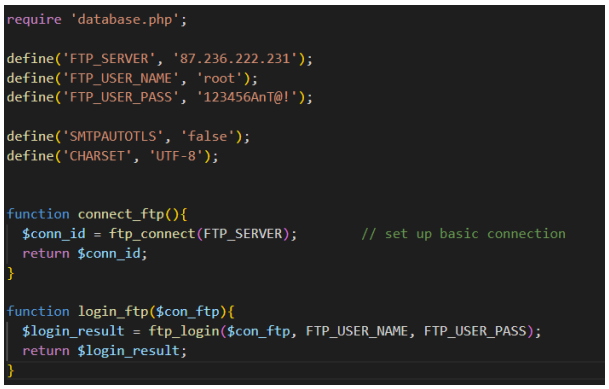
Figura 41: Definición parámetros de conexión

```php
//PRODUCCION
define('DB_HOST', 'localhost');
define('DB_USER', 'emergencias');
define('DB_PASS', 'Emergencias123');
define('DB_NAME', 'Emergencias');


function connect(){
  $connect = mysqli_connect(DB_HOST ,DB_USER ,DB_PASS ,DB_NAME);
  if (!$connect) {
    echo mysqli_connect_errno();
    echo mysqli_connect_error();
    exit;

  }

  mysqli_set_charset($connect, "utf8");

  return $connect;
}


$con = connect();
```

Fuente: Código fuente

Y el otro fichero mostrado en la Figura 42 define las conexiones FTP para la transmisión de documentación, igualmente el lenguaje utilizado simplifica el uso del protocolo.

Figura 42: Definición parámetros FTP



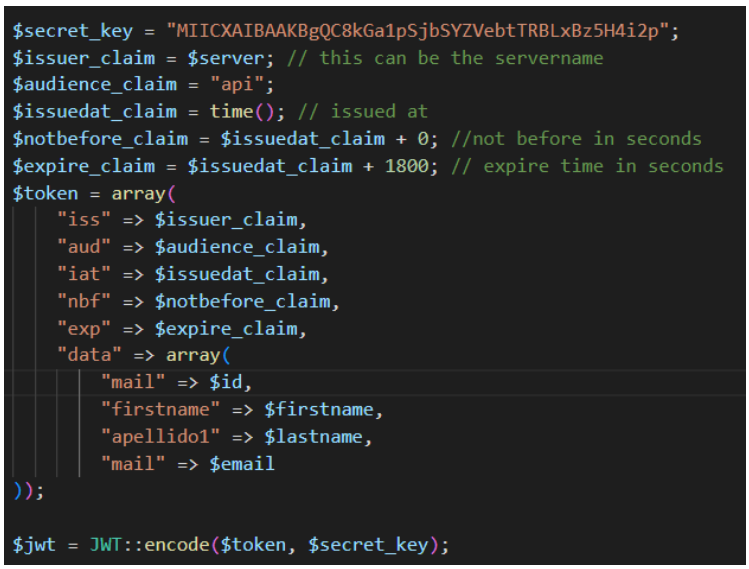
```php
require 'database.php';

define('FTP_SERVER', '87.236.222.231');
define('FTP_USER_NAME', 'root');
define('FTP_USER_PASS', '123456AnT@!');

define('SMTPAUTOTLS', 'false');
define('CHARSET', 'UTF-8');


function connect_ftp(){
  $conn_id = ftp_connect(FTP_SERVER);        // set up basic connection
  return $conn_id;
}

function login_ftp($con_ftp){
  $login_result = ftp_login($con_ftp, FTP_USER_NAME, FTP_USER_PASS);
  return $login_result;
}
```

Fuente: Código Fuente

- **Login**

  El acceso a la aplicación como se ha explicado anteriormente en el capítulo es vital para el uso de la aplicación y gestionar las funcionalidades que cada usuario tiene permitidas realizar. Para hacer un sistema robusto y seguro, se ha implementado la generación de un token al que se le aplica el algoritmo de encriptación SHA256, y está configurado en la Figura 43.

Figura 43: Configuración del token



```php
$secret_key = "MIICXAIBAAKBgQC8kGa1pSjbSYZVebtTRBLxBz5H4i2p";
$issuer_claim = $server; // this can be the servername
$audience_claim = "api";
$issuedat_claim = time(); // issued at
$notbefore_claim = $issuedat_claim + 0; //not before in seconds
$expire_claim = $issuedat_claim + 1800; // expire time in seconds
$token = array(
    "iss" => $issuer_claim,
    "aud" => $audience_claim,
    "iat" => $issuedat_claim,
    "nbf" => $notbefore_claim,
    "exp" => $expire_claim,
    "data" => array(
        "mail" => $id,
        "firstname" => $firstname,
        "apellido1" => $lastname,
        "mail" => $email
));

$jwt = JWT::encode($token, $secret_key);
```

Fuente: Código fuente

Tras cada acceso el sistema genera un nuevo token que hace único el acceso del usuario. Y lo almacena en la base de datos para mantener un control sobre

los registros del usuario. Además, se obtiene de la base de datos la relación entre usuario y rol para que la aplicación sea consciente de las funcionalidades que le están permitidas al usuario, se ve un ejemplo en la Figura 44.

Figura 44: Consulta a la BBDD

```
$safe_user = mysqli_real_escape_string($con,$user);
$sql = "SELECT Usuarios.mail,Usuarios.nombre,Usuarios.password,Usuarios.apellido1,Usuarios.apellido2,Usuarios.bloqueado,
Roles.nombre_rol,Usuarios.jwtToken FROM Usuarios JOIN Roles ON Usuarios.id_rol = Roles.id WHERE Usuarios.mail ='$safe_user'
AND Usuarios.password ='$pass_code256';";
$sqlcount = "SELECT count(*) FROM Usuarios JOIN Roles ON Usuarios.id_rol = Roles.id WHERE Usuarios.mail ='$safe_user' AND
Usuarios.password ='$pass_code256';";
$sqlcount1 = "SELECT count(*) FROM Usuarios JOIN Roles ON Usuarios.id_rol = Roles.id WHERE Usuarios.mail ='$safe_user';";
```

Fuente: Código fuente

Otra característica importante del acceso es comprobar si el usuario ha sido bloqueado por el administrador para evitar conexiones a la aplicación, permite tener registros del usuario evitando a este el uso a la aplicación, la comprobación es realizada en la Figura 45.

Figura 45: Comprobación de usuario bloqueado

```
if($row['bloqueado'] == '1'){
  return http_response_code(423);
```

Fuente: Código fuente

- **Inserción**

  El registro de información en el sistema se realiza para añadir residencias nuevas, usuarios o catástrofes. Por ello, en angular se recogen los datos y se envía en formato JSON a la API. Cuando se recibe la información en lado del servidor como se muestra en la Figura 46 se trata el conjunto de datos recibidos aplicando un filtrado exhaustivo evitando inyecciones dañinas sobre la base de datos.

Figura 46: Recepción de los datos de entrada

```
$postdata = file_get_contents("php://input");
$request = json_decode($postdata);

$mail= mysqli_real_escape_string($con,trim($request->mail));;
$nombre= mysqli_real_escape_string($con,trim($request->nombre));;
$apellido1= mysqli_real_escape_string($con,trim($request->apellido1));;
```

Fuente: Código fuente

Posterior al filtrado, se estructuran los datos en una consulta en lenguaje SQL mostrada en la Figura 47, permitiendo la ejecución en la base de datos la inserción de la información recolectada.

Figura 47: Filtrado en la BBDD

```
$safe_user = mysqli_real_escape_string($con,$user);
$sql = "INSERT INTO Usuarios (Mail,Nombre,Apellido1,Apellido2,id_rol,password,jwtToken,fec_ult_con)
VALUES ('$mail','$nombre','$apellido1','$apellido2',$id_rol,'$pass_code256','$pass','01-01-0000');";
```

Fuente: Código fuente

- **Importación CSV**

La inserción no únicamente es posible mediante el registro de un formulario, está disponible para la adicción masiva de elementos. Se realiza mediante un fichero de datos estructurado en formato CSV que recibe la API. El primer paso es obtener el fichero que se ha subido al servidor. Para ello debemos obtener la ruta en la que se encuentra, mostrado en la Figura 48, y comprobar que el formato del fichero es el correcto.

Figura 48: Recepción fichero

```
//move the file to target folder
if (move_uploaded_file($_FILES['fileToUpload']['tmp_name'], "/var/www/html/uploads/".$actual_fname.".".$ext)) {

$result["status"]=1;
$result["message"]="Uploaded file successfully.";
```

Fuente: Código fuente

A continuación, la Figura 49 muestra cómo se procesa la información. En primer lugar, se deben leer los datos de este para poder enviarlos a la base

de datos. En segundo, se abre el fichero y se lee cada línea estructurando la información que contiene en su interior.

Figura 49: Procesando fichero CSV

```php
//Ahora insertamos los datos de las residencias
if($ext == 'csv'){
  $csv_file = "/var/www/html/uploads/".$actual_fname.".".$ext;

  $csv_file = fopen($csv_file, 'r');
  $theData = fgets($csv_file);
  $i = 0;
  while (!feof($csv_file)){
    date_default_timezone_set('Europe/Madrid');
    $fecha = date("d-m-Y");
    $csv_data[] = fgets($csv_file, 1024);
    $csv_array = explode(",", $csv_data[$i]);
    $insert_csv = array();
    $insert_csv['Denominacion'] = $csv_array[0];
    $insert_csv['Direccion'] = $csv_array[1];
    $insert_csv['CP'] = $csv_array[2];
    $insert_csv['Municipio'] = $csv_array[3];
    $insert_csv['Telefono'] = $csv_array[4];
    $insert_csv['Plazas'] = $csv_array[5];
    $insert_csv['Titularidad'] = $csv_array[6];
    $insert_csv['URL'] = $csv_array[7];
    $insert_csv['Latitud'] = $csv_array[8];
    $insert_csv['Longitud'] = substr(substr($csv_array[9],0,-2),0,-2);
    $insert_csv['UltActData'] = $fecha;
```

Fuente: Código fuente

Cuando se almacena la información en variables de ejecución, se reorganizan para formular la petición de inserción correcta a la base de datos. La validación de los datos indicada en la Figura 50, también se realiza para que todos los campos obligatorios se añadan y no causen errores.

Figura 50: Insertando los datos del CSV en BBDD

```php
if( $insert_csv['Denominacion'] != "" &&  $insert_csv['Latitud'] != "" && $insert_csv['Longitud'] != ""){
  $query = "INSERT INTO Residencia(Denominacion,Direccion,CP,Municipio,Telefono,Plazas,Titularidad,URL,Latitud,Longitud,UltActData)
  VALUES('".$insert_csv['Denominacion']."','".$insert_csv['Direccion']."','".$insert_csv['CP']."','".$insert_csv['Municipio']."',
  '".$insert_csv['Telefono']."','".$insert_csv['Plazas']."','".$insert_csv['Titularidad']."','".$insert_csv['URL']."',"
  .$insert_csv['Latitud'].",".$insert_csv['Longitud'].",'".$insert_csv['UltActData']."')";
  $nquery = mysqli_query($con, $query);
```

Fuente: Código fuente

- **Edición**

  El sistema también es capaz de editar registros, mediante la actualización de la información de la base de datos. El proceso de edición es similar a la inserción, la principal diferencia es sobre la consulta que va a procesar la base de datos que afecta sobre un registro existente. Se debe indicar el

elemento concreto a sobrescribir como se muestra en la consulta de la Figura 51.

Figura 51: Actualizando registro de la BBDD

```
$safe_user = mysqli_real_escape_string($con,$user);
$sql = "UPDATE Residencia SET Denominacion = '$denom', Direccion = '$direc', CP = $codp,
 Municipio = '$muni', Telefono = '$tele', Plazas = $plaz, Titularidad = '$titu',
URL = '$nelace', Latitud = $lat, Longitud = $long,UltActData = '$fecha' WHERE `Residencia`.`ID` = $id;";
```

Fuente: Código fuente

En ningún párrafo anterior se explica cómo se verifican que las peticiones a la API e identifican al usuario registrado en la aplicación que ejecuta dicha tarea. Por lo tanto, la Figura 52 muestra el fichero de verificación de la cabecera de la petición que se realiza, para realizar una decodificación del token que incluye y comprobarlo con la clave secreta que lo generó.

Figura 52: Verificación del token

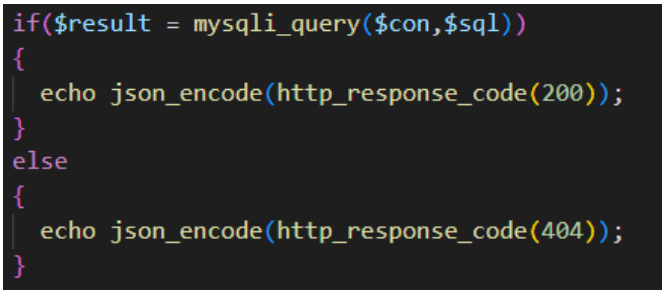
```
require_once(__DIR__ . "/../lib/php-jwt/vendor/autoload.php");
use \Firebase\JWT\JWT;

function protectedAuth(){
    return true;
    $secret_key = "MIICXAIBAAKBgQC8kGa1pSjbSYZVebtTRBLxBz5H4i2p";
    $data = json_decode(file_get_contents("php://input"));
    $jwt = null;
    //$token = mysqli_real_escape_string($con,trim($request->token));
    $authHeader = $_SERVER['HTTP_AUTHORIZATION'];
    $arr = explode(" ", $authHeader);
    $jwt = $arr[1];
    if($jwt){
        try {
            $decoded = JWT::decode($jwt, $secret_key, array('HS256'));
            return true;
        }catch (Exception $e){
            return false;
        }
    }else{
        return false;
    }
}
$protectedAuth = protectedAuth();
```

Fuente: Código fuente

Otro aspecto que siempre se debe tener en cuenta durante el desarrollo de cualquier software es el tratamiento de errores. Para ello se muestra la Figura 53 como se realizan las respuestas al usuario cundo hacen uso erróneo del token, la ejecución de petición se resuelve arrojando un código

de error. Esta metodología se aplica a todas las peticiones del sistema para hacer el entorno seguro y confiable ante la ley de protección de datos, entre otras.

Figura 53: Respuesta JSON de la API



```php
if($result = mysqli_query($con,$sql))
{
  echo json_encode(http_response_code(200));
}
else
{
  echo json_encode(http_response_code(404));
}
```

Fuente: Código fuente

Además, la metodología aplicada para transferir la información en las peticiones son ficheros JSON que incluyen los datos de la petición y el resultado de su ejecución.

- **Registro de catástrofe**

  Las catástrofes son un aspecto fundamental de la aplicación, y son registradas en la aplicación como se ha descrito en el punto anterior sobre su inserción. Sin embargo, se realizan más tareas asociadas que implican generar un registro de las relaciones que existen entre esta y las residencias a las que les afecta. Mediante los datos cumplimentados en la catástrofe se aplica un algoritmo que gestiona las residencias que se encuentra en la zona indicada. Se ha desarrollado un algoritmo en Python para obtener esta información, ya que este lenguaje aumenta el rendimiento en el proceso de datos. La petición de generación de catástrofe y el sistema la registra, tras esto, la Figura 54 muestra cómo se realiza una llamada al script que verifica las posibles relaciones existentes.

Figura 54: Inserción de catástrofe

```
if($result = mysqli_query($con,$sql))
{
  if($result1 = mysqli_query($con,$sql1))
{
  $command = escapeshellcmd('python3 /var/www/Python/InsertarResAfectadas.py
  '.mysqli_fetch_assoc($result1)['LAST_INSERT_ID()']);
    $output = shell_exec($command);
    echo $output;
    echo json_encode(http_response_code(200));
}
```

Fuente: Código fuente

El script recibe el identificador de la catástrofe que se ha registrado, y realiza una consulta a la base de datos que obtiene las características principales de esta. También recopila la información del conjunto de residencias del sistema. A continuación, el algoritmo procesa secuencialmente todas las ubicaciones de las residencias y calcula la distancia entre dos puntos (Figura 55), el centro de la catástrofe y la ubicación de los geriátricos, y verifica si es inferior o superior al radio de influencia.

Figura 55: Algoritmo para calcular la distancia entre dos puntos

```
#recorrer myresult1
for x in myresult1:
    idres = x[1]
    latres = x[9]
    lonres = x[10]

    #calcular distancia entre dos puntos
    dlat = math.radians(float(latcat)-float(latres))
    dlon = math.radians(float(loncat)-float(lonres))
    a = math.sin(dlat/2) * math.sin(dlat/2) + math.cos(math.radians(lat1)) * math.cos(math.radians(lat2)) * math.sin(dlon/2) * math.sin(dlon/2)
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
    d = R * c
    #convertir a metros
    d = d * 1000
    #comparar distancia si es menor a 100 metros
    if d < (int(zoncat)*1000):
        mycursorinse.execute("insert into ResCat (idCat, idRes) values ("+str(idcat)+", "+str(idres)+")")

#ejecutar inserciones
```

Fuente: Código fuente

Tras obtener si la residencia está afectada, se inserta en la base de datos la implicación de esta, generando la relación en el sistema.

- **Búsqueda**

  La búsqueda de elementos es muy utilizada para obtener la información necesaria que se gestiona en la aplicación web. Los métodos de búsqueda son muy similares, pero existen diferencias entre los resultados obtenidos. En algunos casos se recibe un parámetro de entrada por la API para filtrar la búsqueda. Y estructuramos los datos cómo indica la Figura 56 para transmitirlos posteriormente en la respuesta de la petición.

Figura 56: Búsqueda de un listado en la BBDD

```
$catastrofes = [];
$sql = "SELECT * FROM Catastrofes WHERE Activa = 1;";
date_default_timezone_set('Europe/Madrid');
$fecha = date("Y-m-d H:i:s");
if($result = mysqli_query($con,$sql))
{
  $i = 0;
  while($row = mysqli_fetch_assoc($result))
  {
    //if($row['fec_fin'] == null || $fecha <  date("Y-m-d",strtotime($row['fec_fin']."+5 days
      $catastrofes[$i]['ID']             = $row['id'];
      $catastrofes[$i]['Descripcion']                    = $row['Descripcion'];
      $catastrofes[$i]['Tipo']               = $row['Tipo'];
      $catastrofes[$i]['Latitud']                  = $row['Latitud'];
      $catastrofes[$i]['Longitud']              = $row['Longitud'];
      $catastrofes[$i]['Zona']               = $row['Zona'];
      $catastrofes[$i]['Area']                 = $row['Area'];
      $catastrofes[$i]['Activa']             = $row['Activa'];
      $catastrofes[$i]['Fecha_Creacion']                   = $row['fec_crea'];
      $catastrofes[$i]['Fecha_Desact']               = $row['fec_des'];
      $catastrofes[$i]['User_Creacion']              = $row['User_crea'];

      $i++;
    //}

  }
```

Fuente: Código fuente

5.2 Casos de prueba y resultados

Tras aplicar todo el desarrollo de la aplicación, se deben realizar las pruebas correspondientes que verifican el correcto funcionamiento del sistema, y que los requisitos definidos inicialmente se cumplen.

Esta fase es una de las más importantes durante el desarrollo de un sistema como el que se proyecta. Será una etapa donde se analizan e identifican aquellos errores y problemas que se han generado en el desarrollo del software. Por lo tanto, se comienza comprobando los requisitos establecidos y demostrando que cumplen su función.

A continuación, se exponen algunas pruebas que se han llevado a cabo:

Tabla 27: Prueba 1 – Acceso a la aplicación

Prueba 1 – Acceso a la aplicación	
Condiciones previas	El usuario debe estar creado por el administrador en el sistema.
Acciones	El usuario accede a la web e introduce sus datos de inicio de sesión en el formulario habilitado para ello, pulsando el botón de login.
Resultado esperado	La aplicación valida la información introducida y le muestra el panel principal del mapa.

Fuente: Elaboración propia

Tabla 28: Prueba 2 – Registro de catástrofe

Prueba 2 – Registro de catástrofe	
Condiciones previas	El usuario de haber accedido a la plataforma.
Acciones	El técnico accede al formulario de registro de catástrofe e introduce todos los datos sobre esta. Y pulsar el botón insertar para hacer efectivo el registro.
Resultado esperado	El sistema comprueba que se han introducido los datos necesarios para el registro, seguidamente se añade a la base de datos. Además, se aplica el algoritmo que genera las relaciones con las residencias afectadas. En último lugar, se muestra en pantalla el resultado.

Fuente: Elaboración propia

Tabla 29: Prueba 3 – Visualización de catástrofes desactivadas

Prueba 3 – Visualización de catástrofes desactivadas	
Condiciones previas	El cliente debe haber accedido a la aplicación, y el sistema debe tener registradas catástrofes.
Acciones	El usuario accede desde el menú a la sección de catástrofes, y tras mostrarse, debe pulsar el botón 'Mostrar Desactivadas'
Resultado esperado	La plataforma muestra un listado donde se cargan las catástrofes activas y desactivadas, después de realizar la consulta a la base de datos.

Fuente: Elaboración propia

Tabla 30: Prueba 4- Modificar contraseña del usuario

Prueba 4 – Modificar contraseña del usuario	
Condiciones previas	El cliente debe acceder al sistema.
Acciones	El usuario despliega las opciones sobre su nombre mostrado en el menú, y selecciona editar. Se muestra un formulario donde introduce la nueva contraseña y pulsa editar para confirmar el cambio.
Resultado esperado	El sistema identifica al usuario y valida la nueva contraseña introducida, se actualiza en la base de datos y por último le muestra al usuario el resultado de la modificación.

Fuente: Elaboración propia

Tabla 31: Prueba 5 – Generación de ruta

Prueba 5 – Generación de ruta	
Condiciones previas	El cliente tendrá que acceder a la aplicación y el navegador debe permitir la geolocalización.
Acciones	El usuario hace clic sobre una residencia del mapa, y posteriormente pulsa el botón de cómo llegar.
Resultado esperado	El sistema recibe las acciones del usuario y dibuja unas trazas sobre el mapa que representan la ruta, además muestra un listado de las direcciones que se deben tomar.

Fuente: Elaboración propia

Tabla 32: Prueba 6 – Mostrar datos residencia

Prueba 6 – Mostrar datos residencia	
Condiciones previas	El usuario debe acceder a la aplicación, el sistema tiene registrada al menos una residencia.
Acciones	El usuario pulsa sobre la marca que identifica a la residencia que desea ver su información en el mapa.
Resultado esperado	El sistema identifica la residencia y habilita una sección que muestra la información de la residencia.

Fuente: Elaboración propia

Tabla 34: Prueba 7 – Descargar listado de residencias afectadas

Prueba 7 – Descargar listado de residencias afectadas	
Condiciones previas	El sistema tiene al menos una catástrofe activa, y el usuario debe iniciar sesión en la aplicación.
Acciones	El usuario selecciona una catástrofe del mapa, se cargan las características y opciones de esta. Pulsa el botón de residencias afectadas.
Resultado esperado	La aplicación carga los datos de la catástrofe y recibe la petición de mostrar las afectadas. Seguidamente, muestra un listado con todas las residencias afectadas por la catástrofe que ha solicitado el usuario. Tras la solicitud de descarga del usuario, se genera un documento PDF que se abre en una nueva pestaña y se descarga automáticamente.

Fuente: Elaboración propia

Figura 35: Prueba 8 – Crear usuario

Prueba 8 – Crear usuario	
Condiciones previas	El administrador debe acceder al sistema.
Acciones	El administrador accede a la pestaña usuarios del sistema, y se carga la lista de usuarios. Este pulsa el botón de crear usuario y carga un formulario donde se introducen los datos del nuevo usuario. Por último, se pulsa el botón crear.
Resultado esperado	El sistema recibe los datos del nuevo usuario, y los válida para comprobar si están correctos y completos. Después realiza la inserción en la base de datos y genera una notificación mostrando el resulta.

Fuente: Elaboración propia

Tabla 36: Prueba 9 – Búsqueda en la tabla residencias

Prueba 9 – Búsqueda en la tabla residencias	
Condiciones previas	El usuario debe acceder a la aplicación y el sistema debe tener registradas residencias.
Acciones	El usuario accede a la opción residencias del menú, y se carga el listado. Después el usuario introduce en el campo de búsqueda de la tabla el parámetro de filtrado.
Resultado esperado	El sistema carga el listado de residencias y según recibe el valor de búsqueda va filtrando los resultados, eliminando aquellos que no coinciden con este.

Fuente: Elaboración propia

Tabla 37: Prueba 10 – Desactivar catástrofe

Prueba 10 – Desactivar catástrofe	
Condiciones previas	El sistema debe tener catástrofes activas, y el usuario debe acceder a este.
Acciones	El usuario selecciona una catástrofe en el mapa, y se muestran los datos de esta. En las opciones mostradas, selecciona desactivar.
Resultado esperado	La aplicación recibe el identificador de la catástrofe a desactivar, realizando la actualización de esta en la base de datos. Del mismo modo, elimina las relaciones existentes con las residencias afectadas por esta. Se muestra al usuario una notificación que indica el resultado.

Fuente: Elaboración propia

Tabla 38: Prueba 11- Aplicar capas al mapa

Prueba 11 – Aplicar capas al mapa	
Condiciones previas	El cliente debe acceder al sistema.
Acciones	El usuario selecciona alguna de las capas disponibles en el mapa y la habilita.
Resultado esperado	El sistema detecta la capa seleccionada por el usuario, y muestra sobre este las características específicas que contiene el servicio de la capa.

Fuente: Elaboración propia

Tabla 39: Prueba 12 – Inserción masiva mediante CSV

Prueba 12 – Inserción masiva mediante CSV	
Condiciones previas	El usuario debe entrar al sistema y tener un fichero CSV en el formato adecuado.
Acciones	El usuario accede al listado de residencias y selecciona la opción insertar de la tabla. Cuando se carga el formulario, el cliente sube el fichero y pulsa el botón insertar.
Resultado esperado	El sistema recibe la petición de inserción y el fichero validando la información. Seguidamente procesa los datos del fichero y realiza las inserciones en la base de datos. Le muestra al usuario una notificación indicando el resultado del registro.

Fuente: Elaboración propia

Tras definir los casos de prueba se realizan cada una de ellas por un grupo de usuarios recurrentes en el sistema para obtener un resultado. En la tabla 40 donde se indican los resultados obtenidos para cada una de ellas.

Tabla 40: Resultados de las pruebas

Número de Prueba	Resultado de la aplicación
1	Resultado esperado
2	Resultado esperado
3	Resultado esperado
4	Resultado esperado
5	Resultado esperado
6	Resultado esperado
7	Resultado esperado
8	Resultado esperado
9	Resultado esperado
10	Resultado esperado
11	Resultado esperado
12	Resultado esperado

Fuente: Elaboración propia

Los resultados obtenidos en las pruebas demuestran que el funcionamiento de la aplicación es el correcto y que los requisitos que se han especificado para este sistema se cumplen. Por lo tanto, supone un caso de éxito en el desarrollo del proyecto.

5.3 Despliegue de la aplicación

El desarrollo de la aplicación necesita ser implementado y desplegado para poder hacer un sistema productivo y totalmente funcional. Inicialmente, el sistema se desplegó en un servidor local para su funcionamiento. Sin embargo, cuando se comenzaron las pruebas se observó que el rendimiento no era el adecuado ya que no soportaba a todos los usuarios.

Por lo tanto, se ha decidido desplegar la aplicación en un servidor online. El servidor online es una máquina virtual Linux que despliega el servicio Apache para ejecutar la aplicación.
Además, se ha instalado la base de datos en el servidor para aumentar el rendimiento y las peticiones recurrentes de un gran número de usuarios.

La aplicación web está alojada en el servidor con dirección IP: 87.236.222.231 que tiene asignada una DNS pública registrada: https://www.residenciasenemergencia.es la cual permite el acceso a los navegadores.

El sistema se ha desarrollado sin incorporar datos de residencias para su uso, pero cuando se ha desplegado se han agregado residencias utilizando datos obtenidos del CSIC. Pero se han desarrollado scripts que procesan los datos obtenidos y los carga en ficheros CSV en el formato establecido en la aplicación, se muestra en la Figura 57:

Figura 57: Script de formateado de ficheros

```
wb = openpyxl.load_workbook("/Users/elzur/Downloads/completas/"+fichero)
sheet = wb.get_sheet_by_name(nombre2)
csvFile = open("C:/Users/elzur/Downloads/encsv/"+nombre+".csv", 'w', newline='')
csvWriter = csv.writer(csvFile)
for rowNum in range(1, sheet.max_row + 1):
    rowData = []
    rowdd = []
    for colNum in range(1, sheet.max_column + 1):
        #si colNum es 2
        if colNum == 2:
            rowdd.append(sheet.cell(row=rowNum, column=colNum).value)

            print(sheet.cell(row=rowNum, column=colNum).value)

        #si es igual a 3
        elif colNum == 3:
            rowdd.append(sheet.cell(row=rowNum, column=colNum).value)

            print(sheet.cell(row=rowNum, column=colNum).value)

        elif colNum == 4:
            rowdd.append(sheet.cell(row=rowNum, column=colNum).value)
```

Fuente: Código fuente

6

Conclusiones y Líneas Futuras

Tras la elaboración del proyecto donde se ha diseñado y desarrollado un sistema de monitorización para las residencias de ancianos, se han cubierto el conjunto de requisitos y objetivos establecidos al inicio del trabajo. La ofrece una solución eficiente y completa que cubre las necesidades del cliente aportando al mercado un nuevo sistema para este ámbito, permitiendo gestionar catástrofes de diferente índole como la gestión de los geriátricos frente a eventualidades que puedan surgir.

El proyecto demuestra exitosamente la solución propuesta, ya que se ha realizado un desarrollo con un enfoque eficiente ya que no limita el número de incidencias, ni tipología que monitoriza. Los servicios de emergencias cuentan con un sistema sencillo para su uso y con la información necesaria para realizar con rapidez y éxito sus labores. Además, la aplicación habilita acceso a los servicios administrativos de los centros residenciales el registro y actualización de los datos manteniendo un sistema actualizado y completo.
El desarrollo del sistema se ha enfocado para un mantenimiento sencillo y una escalabilidad que incorpore nuevas mejoras.

El desarrollo se ha llevado a cabo utilizando multitud de tecnologías punteras como Python, MySQL, PHP o Angular combinando el patrón MVVM. En los estudios del grado de Ingeniería Informática no se ofrece formación específica en estas

tecnologías, pero la base proporcionada ha facilitado la comprensión y el estudio de cualquier nueva tecnología. No obstante, para desarrollar el proyecto se ha requerido de una fase de aprendizaje de las tecnologías usadas, llevando un periodo amplio de tiempo y horas para poder hacer frente la complejidad del proyecto. Estas tecnologías son algunas de las más demandadas en el sector de desarrollo de software de aplicaciones web *fullstack*.

El proyecto completa la fase formativa del grado, pero también ha permitido cumplir el objetivo personal de desarrollar un sistema que aporta una soluciona real a un entorno determinado, incluso si es posible poder continuar el desarrollo y proponerle al cliente que mostró la necesidad la misma. Además, el proyecto ha permitido dominar las tecnologías utilizadas que añaden valor de cara al entorno profesional.

La aplicación ha sido implementada y desarrollada con éxito, pero existen algunas mejoras que incrementan su valor y la funcionalidad del sistema. Debido a que es una de las primeras plataformas que cubren las necesidades planteadas, el uso de la aplicación generará nuevas ampliaciones, o tras alguna catástrofe de índole desconocido, se crearan nuevas tipologías que deban adaptar el sistema. A pesar de ello, surgen mejoras que pueden ser implementadas. A continuación, se abordan algunas posibles mejoras.

La aplicación actualmente cuenta con el idioma castellano, ya que es la única preferencia definida. Sin embargo, el sistema puede ser extensible a otros países que hagan uso de otros idiomas. Esta funcionalidad puede ser importante puesto que en catástrofes que haya colaboradores internacionales es muy habitual que se reúnan expertos del área de diferentes instituciones alrededor del mundo.

Otra característica del sistema son las notificaciones, ya que todas las funcionalidades se basan en notificaciones push. Sin embargo, se podría crear un sistema de notificaciones más avanzado que conecte con los centros de emergencia mediante mensajes móviles o llamadas que crean alertas más informativas. Además, se puede generar un gestor de notificaciones de correo que recuerde a los centros residenciales la actualización de los datos.

El sistema actualmente muestra información alojada en la base de datos que muestra los datos principales de las residencias, pero existen datos más específicos y útiles para los servicios de emergencia como los planes de prevención o planas de emergencia. Estos documentos son elaborados en los centros residenciales junto a técnicos de emergencias. Por lo tanto, es una buena mejora el implementar la gestión de un sistema de documentación que aloje todos los planes que se han desarrollado en los centros.

Referencias

Maaoui, K. (2019). *Why Angular is your best choice for your next projects ?* Medium. Recuperado el 1 de diciembre de 2022, de https://medium.com/@maaouikimo/why-angular-is-your-best-choice-for-you-next-projects-9d754fb18f91

Coppola, M. (2022). *¿Qué es Angular? Características y ventajas* HubSpot. Recuperado el 1 de diciembre de 2022, de https://blog.hubspot.es/website/que-es-angular

Hugh, E. W., David, L. (2002). *Web Database Applications with PHP, and MySQL* O'REILLY. Recuperado el 1 de diciembre de 2022, https://learning.oreilly.com/library/view/web-database-applications/0596000413/

Ibarra, M., Setter, M. (2021). *PHP Authorization with JWT (JSON Web Tokens)* SitePoint. Recuperado el 1 de diciembre de 2022, https://www.sitepoint.com/php-authorization-jwt-json-web-tokens/

Chavan, B. (2018). *Using OpenStreetMap inside Angular (v6)* Medium. Recuperado el 1 de diciembre de 2022, https://balramchavan.medium.com/using-openstreetmap-inside-angular-v6-3d42cbf03e57

Visus, A. (2020). *¿Para qué sirve Python? Razones para utilizar este lenguaje de programación* ESIC. Recuperado el 1 de diciembre de 2022, https://www.esic.edu/rethink/tecnologia/para-que-sirve-python

Shachee, S. (2021). *A Beginner's Guide to Creating a Map Using Leaflet.js* SitePoint. Recuperado el 1 de diciembre de 2022, https://www.sitepoint.com/leaflet-create-map-beginner-guide/#:~:text=Leaflet%20is%20a%20framework%20for%20presenting%20map%20data.,browser%20support%2C%20default%20interactivity%2C%20panning%20and%20zooming%20capabilities

Raj, J. (2018). *How to Create Angular Toastr Notifications* JSCrambler. Recuperado el 1 de diciembre de 2022, https://blog.jscrambler.com/how-to-create-angular-toastr-notifications

Ranwa, S. (2019). *Implement Toastr Notification In Angular 7* C#Corner. Recuperado el 1 de diciembre de 2022, https://www.c-sharpcorner.com/article/implement-toastr-notification-in-angular-7/

Vincy (2022). *jsPDF AutoTable example - Table to PDF* PHPPOT. Recuperado el 1 de diciembre de 2022, https://phppot.com/javascript/jspdf-autotable/#:~:text=The%20jsPDF%20AutoTables%20plugin%20is%20for%20converting%20a,around%20the%20PDF%20generation%20process%20using%20this%20library.

Agafonki, V. (2022). *Using WMS and TMS services - Leaflet - a JavaScript library for interactive maps* LeafletJS. Recuperado el 1 de diciembre de 2022, https://leafletjs.com/examples/wms/wms.html

Gravelle, R. (2018). *The Model-View-ViewModel Pattern and Angular Development* HTMLGoodies. Recuperado el 1 de diciembre de 2022, https://www.htmlgoodies.com/javascript/the-model-view-viewmodel-pattern-and-angular-development/

Roznovsky, A. (2019). *WHY USE PHP? MAIN ADVANTAGES AND DISADVANTAGES* LIGHT IT. Recuperado el 1 de diciembre de 2022, https://light-it.net/blog/why-use-php-main-advantages-and-disadvantages/

Jordana, A. (2022). *What Is Bootstrap?* Hostinger. Recuperado el 10 de diciembre de 2022, https://www.hostinger.com/tutorials/what-is-bootstrap/#:~:text=Bootstrap%20is%20a%20free%20and%20open-source%20web%20development,need%20to%20worry%20about%20basic%20commands%20and%20functions

Shay, T. (2018). *MariaDB vs MySQL - Key Differences - Comparing MySQL 8.0 with MariaDB 10.5* EverSQL. Recuperado el 11 de diciembre de 2022, https://www.eversql.com/mariadb-vs-mysql/

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA