Universidad Zaragoza
1542

# Trabajo Fin de Grado

## Aplicación móvil para acompañar en la visita a las Minas Olvidadas de Aragón

## Mobile application to accompany during the visit to the Forgotten Mines of Aragon


Autor

Héctor Bara

Directores

José Ignacio Canudo Sanagustín

Francisco Javier Zarazaga Soria

Escuela de Ingeniería y Arquitectura



2022/2023

# Aplicación móvil para acompañar en la visita a las Minas Olvidadas de Aragón

## Resumen

Aragón dispone de un patrimonio oculto, en forma de cientos de pequeñas explotaciones mineras abandonadas repartidas por todo el territorio. El proyecto de las Minas Olvidadas de Aragón pretende recuperar la memoria perdida de las pequeñas minas aragonesas y generar una exposición sobre estas minas y minerales de Aragón. Para ello, el Museo de Ciencias Naturales de la Universidad de Zaragoza ha creado una web que recoge la información de las minas y minerales anteriormente nombradas, la aplicación creada en este TFG pretende ayudar a las personas interesadas en visitar estas minas, facilitándoles la visita a las minas que les interesen y poniendo a su disposición toda la información necesaria estén donde estén, tengan o no una conexión a internet.

El objetivo de este TFG ha sido disponer de una aplicación móvil que permita que las personas que la usen puedan llevar encima la información que sobre minas olvidadas de Aragón publica el Museo de Ciencias de la Universidad de Zaragoza. Esta aplicación debe poder convivir, sin molestar, con el sistema de publicación web de este proyecto. De este modo, accederá a la información del mismo como sistema legado. Así mismo, deberá ayudar a que se planifiquen y se lleven a cabo visitas a las minas en donde no se tiene garantía de acceso a conexiones de datos.

Los problemas abordados en este TFG se basan principalmente en el acceso a datos sin conexión de distintos sistemas, entre ellos los diferentes servicios geográficos como los servidores de capas de OpenStreetMap y PNOA y de enrutamiento de Project OSRM, o directamente la base de datos desplegada en el servidor creado para dar información a la aplicación. Todo esto obligando a sincronizar los datos y guardarlos en la aplicación móvil, por diferentes medios, siendo caché, base de datos o variables persistentes. Otro problema abordado es la obtención de los datos sobre las minas de la página web del proyecto, que al no tener una API que ofrezca los datos de las minas, lo que desemboca en la obligación de crear un servicio que navegue por esa web y extraiga los datos necesarios, este proceso es conocido como *webscrap*.

Este proyecto supone una colaboración entre dos importantes entidades pertenecientes a la Universidad de Zaragoza, el grupo de Sistemas de Información Avanzados (IAAA), grupo con gran presencia en proyectos en los que la información geográfica es relevante, y el Museo de Ciencias Naturales.

# DECLARACIÓN DE AUTORÍA

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

## DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe remitirse a seceina@unizar.es dentro del plazo de depósito)

D./Dª. Héctor Bara Lles ,

en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de Estudios de la titulación de Grado en Ingeniería Informática (Título del Trabajo)

Aplicación móvil para acompañar en la visita a las Minas Olvidadas de Aragón

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 05/06/2023

Fdo: Héctor Bara Lles

# ÍNDICE

# AGRADECIMIENTOS

Antes de entrar en el grueso del tema, toca agradecer a toda la gente que ha hecho posible mi trayectoria a lo largo de los cursos de Ingeniería Informática.

En primer lugar a F. Javier Zarazaga Soria y a José Ignacio Canudo por haber aceptado tutorizarme y ofrecerme este interesante proyecto. En segundo lugar a mis padres y a mi pareja por haberme apoyado durante toda la carrera. Por último pero no menos importante a mis compañeros, que a pesar de haber trabajado con ellos hemos podido entablar lazos de amistad fuera de los ámbitos académicos.

# 1 INTRODUCCIÓN

## 1.1 Contexto del Trabajo

El Museo de Ciencias Naturales de la Universidad de Zaragoza fue propuesto por esta institución en 2013, y su creación fue autorizada en 2014 por el Departamento de Educación, Universidad, Cultura y Deporte del Gobierno de Aragón. Esta autorización implica su integración en el Sistema de Museos de Aragón, lo que lo convierte en un museo de carácter público. Más allá de sus colecciones y sus diferentes espacios expositivos, el Museo ha hecho una apuesta por su integración en la sociedad aragonesa. De este modo, una de sus líneas de trabajo se sustenta en el desarrollo de iniciativas de ciencia ciudadana.

La ciencia ciudadana (también conocida como ciencia comunitaria o colectiva) es un tipo de trabajo científico realizado por personal aficionado en colaboración o bajo la dirección de científicos profesionales o instituciones científicas. Las iniciativas de este tipo también se han descrito como de "participación pública en la investigación científica". Y sus resultados son especialmente interesantes por dos motivos. Por un lado, a menudo suponen avances importantes en la investigación científica. Por otro, contribuyen a aumentar la comprensión de la ciencia por parte del público general. Una de las primeras iniciativas de este tipo puestas en marcha ha sido el proyecto de Minas olvidadas de Aragón.

Aragón ha sido territorio minero desde que la historia es historia. Las grandes explotaciones de minerales son escasas, pero las pequeñas abundan por todo el territorio aragonés desde los Pirineos hasta el sur de Teruel. El Museo cuenta con un número importante de ejemplares minerales entre sus colecciones, pero no existe una exposición dedicada a las minas y minerales de Aragón. Este proyecto pretende devolver a la vida a estas minas olvidadas, recolectando información acerca de sus nombres, su localización (coordenadas o como llegar), los minerales que se extraían, fotografías, y esas pequeñas historias que encierran y se han perdido o son desconocidas para la mayoría. En los anexos se dan más detalles de la presencia en la Web de este proyecto.

El grupo de Sistemas de Información Avanzados (IAAA) nace en 1993 con el foco puesto en la creación de tecnología y conocimiento orientados al desarrollo de los sistemas de información con capacidad de integrar soluciones y tecnologías de inteligencia artificial (IA). En 1998 se inicia un acercamiento al ámbito de la información geográfica en donde los recursos de información son grandes y, aquellos que son propiedad de las administraciones públicas, comienzan a estar accesibles por desarrollos normativos. En este escenario, el IAAA mantiene una activa política de colaboración con otros grupos de investigación y unidades de gestión de la Universidad de Zaragoza en proyectos en los que la parte geográfica de la información cobra un valor relevante.

## 1.2 Contexto Tecnológico

### *Tecnologías*

El proyecto se aborda con el objetivo de que el Museo pueda ofrecer una aplicación móvil de consulta de la información que tiene sobre minas abandonadas. Para ello, se han hecho uso de las siguientes tecnologías:

#### *Aplicaciones Cliente*

Para las tecnologías destinadas al cliente podemos distinguir dos partes, destinadas a distintos tipos de usuarios:

1. El cliente Web para tareas de administración, desarrollado con ReactJs[1], una biblioteca JavaScript de código abierto que permite crear interfaces de usuario de aplicaciones de una sola página.
2. El cliente móvil hace uso de la plataforma de desarrollo Ionic[2], este permite la creación de aplicaciones multiplataforma proporcionando una amplia gama de componentes y herramientas de desarrollo. Esta plataforma permite usar código de *frameworks* ampliamente usados: Vue, React y Angular[3]. Angular es un *framework* de desarrollo en TypeScript que permite construir aplicaciones robustas y escalables ofreciendo una arquitectura modular, debido a esto y a que se usa ampliamente en combinación con Ionic se eligió esta tecnología. En esta aplicación era imprescindible utilizar una biblioteca como Leaflet[4], que permite crear mapas interactivos, está desarrollada en JavaScript y cuenta con muchos *plugins* necesarios para la aplicación, como el que permite trazar rutas o el que guarda mapas en memoria caché. Las capas que presenta Leaflet en esta aplicación son la del PNOA[5] (Plan Nacional de Ortofotografía Aérea) y la capa estándar de OpenStreetMap[6].

#### *Servidor*

La base de la tecnología usada en el lado del servidor ha sido Node.js[7], este es un entorno de ejecución de JavaScript que cuenta con un ecosistema muy rico gracias al administrador de paquetes npm[8] (el cual también se ha usado en ambos clientes), el cual ofrece una gran cantidad de herramientas de código abierto para facilitar el desarrollo.
El módulo principal de Node.js en el servidor ha sido Express[9], que facilita el enrutamiento y el manejo de solicitudes y respuestas, todo esto facilita la creación de una API que permita a ambos clientes enviar y recibir información del servidor.

[1] https://legacy.reactjs.org/
[2] https://ionicframework.com/
[3] https://angular.io/
[4] https://leafletjs.com/
[5] https://pnoa.ign.es/
[6] https://www.openstreetmap.org/
[7] https://nodejs.org/es
[8] https://www.npmjs.com/
[9] https://expressjs.com/es/

Los paquetes npm más importantes para el desarrollo del cliente y el servidor se pueden encontrar en el Anexo III.

### *Datos*

El almacenamiento de datos se ha basado totalmente en SQLite[10], que permite almacenar información de forma rápida y sencilla en equipos con pocas capacidades de *hardware*. Esta base de datos ha sido utilizada tanto en la parte del servidor, como en el cliente móvil, ya que requiere tener toda la información de las minas sin conexión.
En el cliente móvil también se utiliza PouchDB[11], base de datos diseñada para funcionar en navegadores web y aplicaciones móviles que permite la sincronización de datos *offline*. Ha sido necesario para la funcionalidad de caché de mapas en Leaflet.

### *HERRAMIENTAS*

En cuanto a las herramientas utilizadas, todas las tareas de programación se han llevado a cabo en Visual Studio Code[12], un editor de texto con una gran variedad de extensiones que facilitan dichas tareas. Para el control de versiones se ha utilizado GitHub[13], principalmente con su extensión del Visual Studio. También, con el fin de automatizar el despliegue, se ha utilizado Docker[14], que es una aplicación que permite introducir aplicaciones en contenedores *software*, más ligeros que las máquinas virtuales. Además, para la prueba de la aplicación se han utilizado algunas herramientas de emulación, cabe destacar el Android SDK[15], que permite simular y depurar un dispositivo Android en un ordenador y los Chrome DevTools[16], que tienen una funcionalidad en la página que permite inspeccionar aplicaciones Android como si fueran aplicaciones web. Por último, para la visualización y edición manual de la base de datos se ha usado DB Browser for SQLite[17].

La carga máxima de trabajo que necesitaba aguantar la máquina para desarrollar el proyecto sería de tener un móvil emulado, y un contenedor de Docker con dos aplicaciones en su interior, además de un navegador que busque toda la información. Para ello se ha dispuesto de un portátil con un procesador AMD Ryzen 5 5600H (3.3GHz-4.2GHz, 6 *core*-12 *threads*), 16 GB RAM y 1TB de almacenamiento SSD, por lo que no ha habido problemas en este aspecto. Para el despliegue se ha utilizado un servidor privado virtual, con 2 *cores*, 4 GB RAM y 80 GB de disco duro, alojado en un proveedor de nube comercial para la parte del servidor, la parte del cliente dependerá del dispositivo móvil que tenga cada usuario de la aplicación.

En este apartado se han nombrado las herramientas utilizadas desde un punto de vista de desarrollo, pero también se han utilizado herramientas de Google como Drive[18] para el almacenamiento y compartición de ficheros, Gmail[19] para el intercambio de correos con los

[10] https://www.sqlite.org/index.html
[11] https://pouchdb.com/
[12] https://code.visualstudio.com/
[13] https://github.com/
[14] https://www.docker.com/
[15] https://developer.android.com/studio
[16] chrome://inspect/#devices (URL en Chrome)
[17] https://sqlitebrowser.org/
[18] https://drive.google.com/
[19] https://mail.google.com/

tutores, Meet[20] para realizar reuniones telemáticas o Docs[21] para la escritura de documentos. También se han utilizado herramientas gráficas como Lucidchart[22] y Diagrams.net[23] para hacer gran variedad de diagramas, o Figma[24] para prototipos de aplicación e historias de usuario.

## 1.3 Alcance, objetivos y limitaciones

El objetivo de este TFG es disponer de una aplicación móvil que permita que las personas que la usen puedan llevar encima la información que sobre minas olvidadas de Aragón publica el Museo de Ciencias de la Universidad de Zaragoza. Esta aplicación debe poder convivir, sin molestar, con el sistema de publicación web de este proyecto. De este modo, accederá a la información del mismo como sistema legado. Así mismo, deberá ayudar a que se planifiquen y se lleven a cabo visitas a las minas en donde no se tiene garantía de acceso a conexiones de datos.

El alcance de este proyecto viene fijado por la disponibilidad de contar con dos versiones de la aplicación móvil, una para iOS y otra para Android, así como de un *back-end* de gestión. Todo el código generado quedará a disposición del Museo y del IAAA en un repositorio de GitHub dentro del ámbito del espacio que este grupo de investigación tiene. Se espera que este proyecto quede abierto a futuras evoluciones del mismo por lo que, tal y como se menciona más adelante, es necesaria llevar a cabo una tarea de transferencia de tecnología al menos a uno de los integrantes del IAAA.

Como limitaciones fundamentales, el proyecto tiene el conjunto de datos de minas ya publicado, y las cuentas de la Universidad de Zaragoza para subir aplicaciones a los correspondientes *markets*. En el primer caso, aunque se cuenta con información de casi 1.000 espacios relevantes, ésta tiene que ser organizada, maquetada y presentada en la nueva web del proyecto. Es por ello que, en el momento de cierre de esta memoria, tan solo ha sido posible experimentar con 8 espacios. Aunque funcionalmente es irrelevante el número que se disponga (1.000 espacios mineros relevantes no es una dimensión de datos que suponga un problema manejar), sí que se pierde riqueza en la presentación de listados, mapas, etc. Por otro lado, en el momento de inicio del proyecto, la Universidad de Zaragoza no contaba con cuenta propia en los *markets* de iOS y Android. Al cierre de esta memoria, la Universidad está en proceso de dotación de estas cuentas y, por lo tanto, todavía no ha sido posible subir las aplicaciones a los *markets*.

Otra limitación es la velocidad de algunos servicios externos, más específicamente del servidor de capas de PNOA, cuyo intercambio de datos es notablemente más lento que los de OpenStreetMap. Entre esto y que el caché implementado ralentiza la carga de capas por el procesamiento extra, la carga de las imágenes de las capas de la aplicación puede ser relativamente lenta (Esto puede variar dependiendo de la potencia del móvil y la calidad de la conexión).

[20] https://meet.google.com/
[21] https://docs.google.com/
[22] https://lucid.app/
[23] https://app.diagrams.net/ También conocido como draw.io
[24] https://www.figma.com/

## 1.4 Esquema general de la memoria del proyecto

En la memoria de este trabajo de fin de grado se puede encontrar al principio una introducción que incluye una puesta en contexto sobre el trabajo a realizar, otra sobre las tecnologías de desarrollo y auxiliares utilizadas y las metas y restricciones del proyecto. Posteriormente se explica cómo se ha desarrollado el trabajo, desde los requisitos, pasando por la arquitectura y diseño del sistema, con su texto acompañado de diagramas para tener una descripción más visual y acabando con los aspectos más complejos abordados.

En la siguiente sección se describen los conocimientos adquiridos con las ideas a implementar en un futuro, seguidos por las conclusiones sobre el proyecto. Para finalizar, se incluye la bibliografía utilizada como apoyo y los anexos.

El Anexo I presenta detalles sobre la situación actual de presencia del proyecto de Minas olvidadas de Aragón en la Web. El Anexo II hace un rápido repaso al proyecto OpenStreetMap y las funcionalidades del mismo que se ha utilizado. En el Anexo III se hace un resumen de los paquetes npm más importantes utilizados en el desarrollo de la aplicación. El Anexo IV relaciona los requisitos de la aplicación con las historias de usuario propuestas en un principio. En el Anexo V se puede ver gráficamente la relación entre los campos de la tabla de minas de la base de datos con los campos de la web del proyecto. El Anexo VI explica la API de *routing* que se ha usado para calcular las rutas y cómo se ha usado en la aplicación. Por último en el Anexo VII hay una explicación de cómo se ha hecho la automatización del despliegue de algunos componentes de la aplicación.

# 2 TRABAJO DESARROLLADO

## 2.1 Requisitos del sistema

En esta sección se recogen los requisitos funcionales del sistema orientados tanto a los usuarios finales como a los administradores y también los requisitos no funcionales. Además después de los requisitos se describen algunos términos en el diccionario de datos.

Requisitos funcionales:

| | |
|---|---|
| RF1 | El usuario debe poder registrarse en la aplicación. |
| RF2 | El usuario debe poder iniciar sesión en la aplicación con una cuenta existente. |
| RF3 | El usuario debe poder cerrar sesión en la aplicación. |
| RF4 | El usuario debe poder cambiar su foto de perfil. |
| RF5 | El usuario debe poder cambiar su nombre de perfil. |
| RF6 | El usuario debe poder ir a la web del museo de ciencias naturales desde la aplicación. |
| RF7 | El usuario debe poder navegar por el mapa de la aplicación. |
| RF8 | El usuario debe poder cambiar de capa del mapa. |
| RF9 | El usuario debe poder descargar los datos actualizados de las minas. |
| RF10 | El usuario debe poder ver la información de la mina. |
| RF11 | El usuario debe poder buscar una mina mediante un buscador de texto. |
| RF12 | El usuario debe poder filtrar minas mediante varios parámetros. |
| RF13 | El usuario debe poder enviar un comentario sobre una mina. |
| RF14 | El usuario debe poder organizar una ruta desde una ubicación hasta una mina. |
| RF15 | El usuario debe poder consultar el estado de sus comentarios. |
| RF16 | El administrador debe poder iniciar sesión en la aplicación de administración. |
| RF17 | El administrador debe poder aceptar comentarios. |
| RF18 | El administrador debe poder rechazar comentarios. |
| RF19 | El administrador debe poder banear usuarios. |
| RF20 | El administrador debe poder actualizar los datos de las minas automáticamente. |

*Tabla 1: Requisitos funcionales del sistema.*

Requisitos no funcionales:

| | |
|---|---|
| RNF1 | Las capas usadas deben ser de OpenStreetMap y de PNOA. |
| RNF2 | Las zonas que ya hayan sido visitadas en el mapa de la aplicación deben ser accesibles sin conexión. |
| RNF3 | La ruta planificada debe ser accesible sin conexión. |
| RNF4 | Las contraseñas de los usuarios deben estar encriptadas en la base de datos. |
| RNF5 | Las rutas a las minas deben poder calcularse desde la ubicación actual o desde una ubicación a elegir. |
| RNF6 | Las ruta planificada debe poder ser escondida o eliminada. |
| RNF7 | La información de minas y sus imágenes deben ser accesibles sin conexión siempre que hayan sido descargadas previamente. |
| RNF8 | El sistema debe poder usar imágenes como mínimo en formato jpg, png, svg y gif. |

*Tabla 2: Requisitos no funcionales del sistema.*

Diccionario de datos:

- Administrador: Además de poder acceder a las funcionalidades de la aplicación móvil como los usuarios normales, los administradores son usuarios con la responsabilidad de encargarse de la moderación y de la actualización de datos.
- Ruta planificada: Es la ruta que puede organizar un usuario desde una ubicación hasta una mina o viceversa, será accesible desde la aplicación sin conexión, pero la creación de la ruta requiere conexión.

En la fase más temprana del proyecto, los requisitos del sistema se hicieron en forma de historias de usuario. En el Anexo IV se pueden encontrar las historias de usuario, ligadas a los requisitos actuales. Algunos de los requisitos de la Tabla 1 no tienen representación de historias de usuario, así como todos los requisitos no funcionales de la Tabla 2.

## 2.2 Arquitectura software del sistema

Al ser un sistema que utiliza muchos datos externos, hay que definir una arquitectura de despliegue, para ello se hizo primero un diagrama a modo de concepto, que más tarde se completó a partir del despliegue real final.

En la aproximación inicial o aproximación conceptual, que podemos ver en la Figura 1, se plantearon como sistemas externos de los que extraer información la aplicación web del proyecto de minas olvidadas de aragón y los servicios geográficos, dejados en este diagrama como un conjunto de servicios, ya que todavía no podía saberse a ciencia cierta cuáles iban a ser necesarios y cómo estaban distribuidos por la red.

Respecto a los componentes a desarrollar, se puede ver en el centro el servidor de aplicaciones, que conecta con un servicio de *webscrap* que consigue datos de la aplicación web nombrada anteriormente y se los devuelve al servidor. Además de eso las aplicaciones del proyecto interactúan con este servidor, la de administración interactúa para tareas de moderación y la aplicación móvil interactúa para tareas de sincronización de datos, sesión y comentarios.

Esta última aplicación también interactúa con los servicios geográficos nombrados anteriormente, otra aproximación podría haber sido que el servidor fuera el que interactuara con los servicios geográficos y se los sirviera más fácilmente a la aplicación móvil, pero eso hubiera complicado enormemente el desarrollo, además de poner una carga mucho más grande en el servidor.

Finalmente, cabe destacar que tanto el servidor de aplicaciones como la aplicación móvil tendrán base de datos, que al sincronizarse comparten la mayoría de datos, salvo algunas excepciones con fines de aumentar la seguridad y eficiencia del sistema.

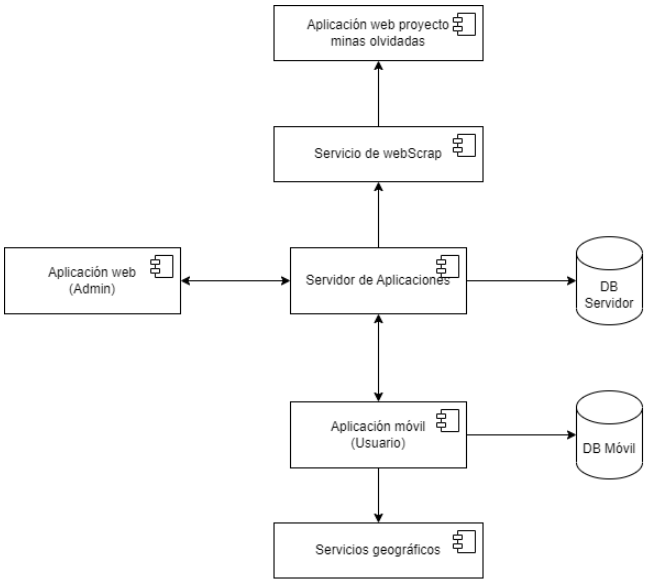


*Figura 1: Diagrama conceptual del despliegue de las aplicaciones.*

El despliegue final no ha sido muy diferente al concepto, en la Figura 2 podemos ver un diagrama de despliegue que mantiene la mayoría de ideas principales del concepto, pero más detallado.

Hay algunos cambios notables respecto al concepto, uno de ellos es el detalle de los servicios geográficos a los que accede al cliente móvil, siendo los servidores de capas que utiliza para mostrar las diferentes capas en el mapa del móvil y el servidor de routing, para calcular las rutas entre coordenadas geográficas. Otro de los cambios notables es la inexistencia de un servicio de *webscrap* separado del servidor, en el despliegue final forma parte de la aplicación del servidor, como uno de los componentes más importantes.

También cabe destacar el despliegue de un mismo contenedor dentro de un servidor privado virtual para la aplicación del servidor con su base de datos y la aplicación web de

administración para facilitar el despliegue. También en este servidor hay otros contenedores con otras aplicaciones del departamento.

Respecto a el cliente móvil, este utiliza dos bases de datos, la primera que ha sido llamada “DB Cliente” es la que se sincroniza con el servidor de la aplicación y la otra, llamada “DB Caché” es la que guarda las imágenes proporcionadas por los servidores de capas para poder utilizar esas capas sin conexión.

Por último, en el diagrama se pueden ver las tecnologías utilizadas en el desarrollo de los distintos componentes del sistema, los protocolos mediante los que intercambian información y los componentes externos al sistema desarrollado en otro color. En el Anexo VII se explica brevemente cómo se ha desplegado el contenedor de la aplicación web de administración y el servidor con Docker.

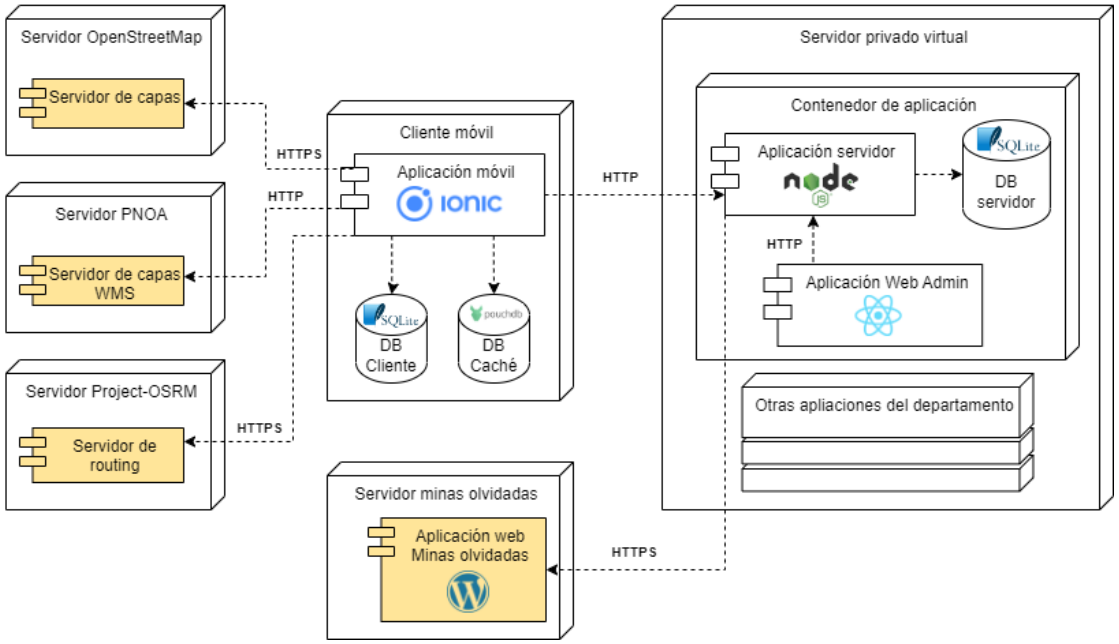


*Figura 2: Diagrama de despliegue final.*

## 2.3 Diseño del sistema

El diseño del sistema ha ido evolucionando durante todo el desarrollo del proyecto, gracias a nuevas ideas, cambios en las aplicaciones externas o descubrimiento de nuevos retos. En esta sección se puede ver el diseño a nivel de interfaz, de software y de base de datos.

### *Navegación de la interfaz de usuario*

La navegación final ha seguido fielmente los pasos que se seguían en el prototipo hecho al principio del proyecto (el cual se puede ver en parte en las figuras del Anexo IV), por lo que en esta sección se presenta el mapa de navegación de la interfaz de usuario final de la aplicación, la cual se puede ver en la Figura 3.

Las imágenes hablan por sí solas, pero hay que tener en cuenta algunas aclaraciones para entender bien el mapa de navegación:

- La página “3 Web museo” es accesible desde cualquier parte de la aplicación en la que sea visible el logo del Museo de Ciencias Naturales de Zaragoza, pero sólo se ha plasmado en la navegación de la página 2 a la página 3 por falta de espacio.
- La división de flechas o el texto “6 o 7” que representa la navegación a la página de perfil se debe a que si el usuario tiene una sesión iniciada irá a la página “7 Perfil con sesión iniciada” y en el caso contrario irá a “6 Perfil sin sesión iniciada”.
- La división de flechas o el texto “5 o 10” que representa la navegación a la página en la que se ve la ruta planificada se debe a que si el sistema tiene una ruta guardada irá a la página “10 Ruta planificada” y en el caso contrario irá a “5 Ruta planificada vacía”.
- La navegación desde la página 6 a la página 7, lógicamente no es directa, pasando por formularios de registro o inicio de sesión, esta parte no se ha plasmado en el mapa de navegación por ser poco relevante.

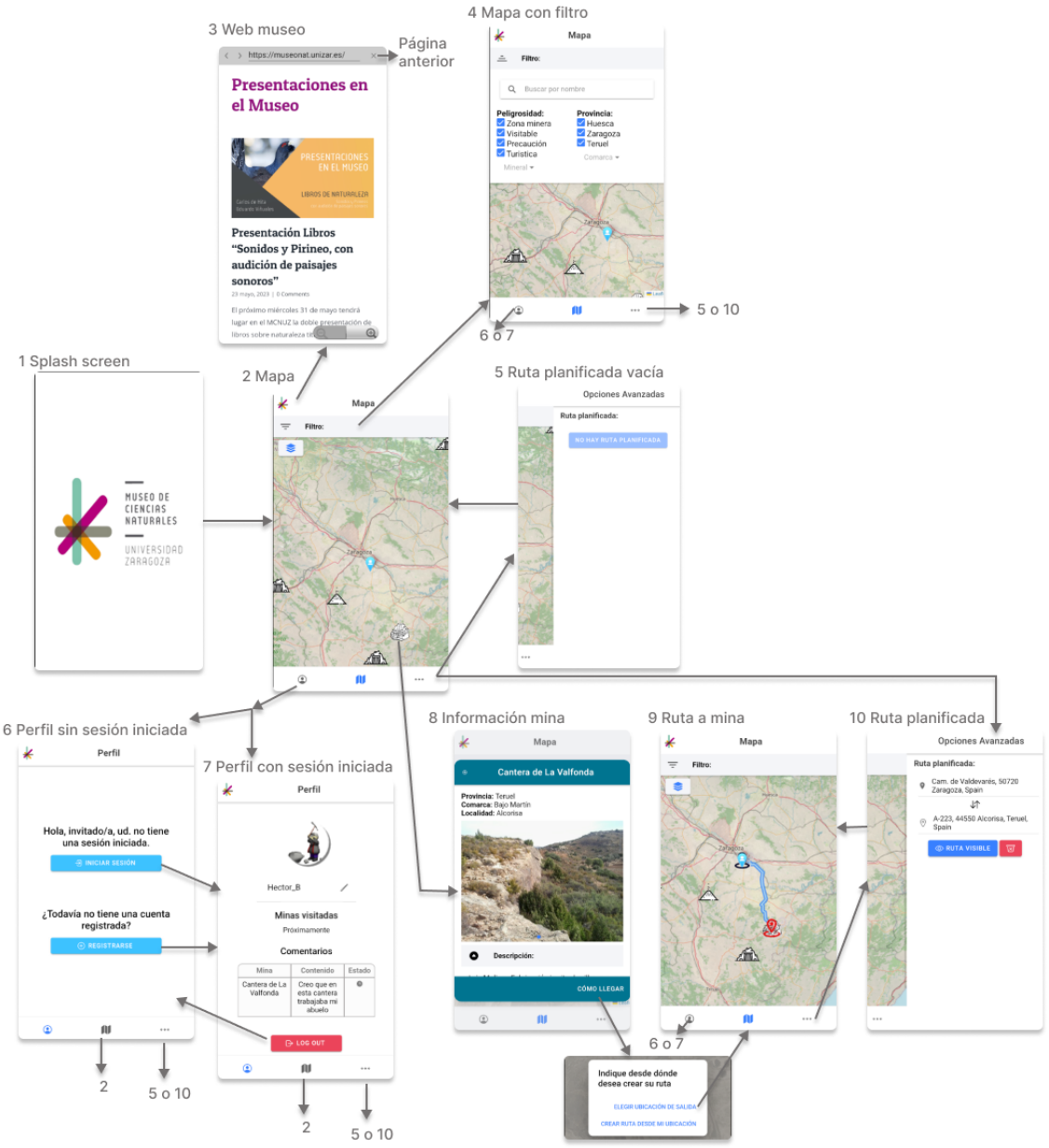


*Figura 3: Mapa de navegación básico.*

Este mapa de navegación no muestra todas las funcionalidades ni todos los caminos posibles a seguir, pero sirve para ver a un nivel medio de detalle como se ve y cómo funciona la navegación del sistema. Para ver el mapa de navegación completo de la aplicación, se puede seguir este enlace a Drive.

### *Diseño del software*

El diseño a nivel de software de la aplicación del proyecto se describe en esta sección mediante dos tipos de diagramas. El primer tipo es el diagrama de clases, que nos ayuda a ver cómo está estructurado el código de la aplicación y las dependencias entre las distintas clases del sistema. El otro tipo es el diagrama de secuencia, que nos muestra a un nivel muy cercano al código el funcionamiento de distintas partes del sistema.

Como podemos ver en el diagrama de clases del *back-end* de la Figura 4, se pueden ver representadas como cajas las diferentes clases del sistema y como paquetes los directorios que las contienen, las flechas indican que la clase de origen de la flecha usa a la clase de destino. Se puede intuir que *app* es la clase principal, de la que parte toda la funcionalidad, esta usa *database* para inicializar el servicio de base de datos, que se encarga de la configuración de *sequelize* y el servicio de *passport*, que sirve para verificar un usuario con nombre y contraseña (Por ello usa la clase *User*). También importa *index*, que es la clase que contiene todo el enrutamiento, ésta asimismo importa *jwtHelper*, que se usa para verificar los *tokens* de sesión y los controladores de *controller*, que se explicarán más adelante.

El paquete *controller* contiene a las clases que proporcionan las funciones accesibles desde la api gracias a *index*, estas clases son muy descriptivas con su función en el sistema y cada una de ellas usa uno o varias clases de *model*, ya que algunas funciones necesitan acceder a varias tablas distintas de la base de datos. Además algunas clases de *controller* utilizan funciones de la clase *versionController* para actualizar las versiones cuando hay cambios en las tablas que utilizan.

Por último el paquete *model* contiene las clases que representan las tablas de las bases de datos del sistema, por ello todas importan la clase *database*. También se usan entre ellas, para declarar las relaciones que mantienen entre tablas, la información de estas clases implementa el diseño de la base de datos que se puede ver en la Figura 8.

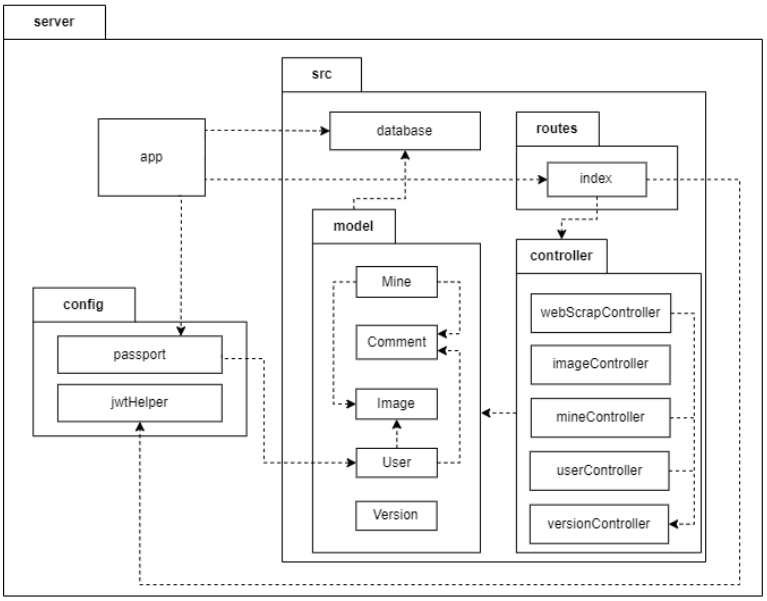


*Figura 4: Diagrama de clases resumido del back-end.*

Para el diagrama de clases del *front-end* del móvil, se ha usado la misma representación que con el del *back-end* y podemos verlo en la Figura 5.

Como se puede observar, en cada paquete de la aplicación hay un patrón, cada módulo principal usa un módulo de *routing*[25] y uno de *component* o *page*, que representa la lógica y presentación de cada página (En el código cada *page* o *component* está compuesto por un fichero TypeScript con la lógica, otro html para la presentación y otro scss para los estilos). Los módulos y *component* de app se encargan de importar los paquetes necesarios e inicializar el *routing*, pero casi toda la lógica está en las demás clases. El enrutamiento principal está en *tabs*, que constituye la base de la navegación y también se encarga de mostrar la ruta planificada guardada. Las clases de *profile* se encargan de presentar y gestionar la lógica de perfil. Lo mismo con *maps* con la lógica de los mapas y el *routing*, aunque este es más complejo internamente y usa *modal-mine* para mostrar la información de las minas. Por último, hay un paquete de servicios, que contiene a *database.service*, este es usado por una gran cantidad de clases para la importación y uso de datos de la base de datos de la aplicación. Además de usarse también para el intercambio de mensajes entre algunas páginas.

[25] En esta parte, el *routing* se refiere a la navegación de la app móvil.

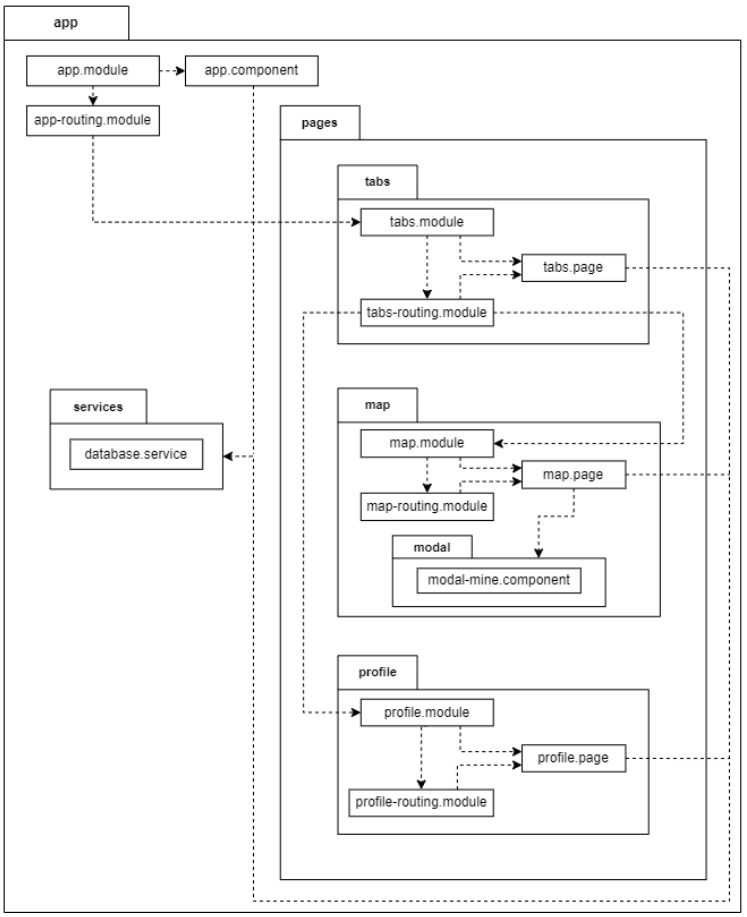


*Figura 5: Diagrama de clases resumido del front-end móvil.*

Para el *front-end* web de administración no ha sido necesario crear un diagrama de clases, ya que su complejidad ha sido muy baja, formado por 4 clases sencillas.

Los diagramas de clases no han incluido atributos o funciones internas de las clases, para hacerlos más sencillos y legibles, por ello se les ha llamado "diagramas de clases resumidos". Para ver con más detalle cómo funcionan las partes del sistema más complejas e importantes, a continuación se muestran y explican los diagramas de secuencia que las representan.

En el diagrama de secuencia de la Figura 6 representa la sincronización de datos del cliente móvil con el servidor, con la frontera marcada entre los dos. De ello se encarga la función *checkUpdates()* al lanzarse *AppComponent*, el primer paso es ver si hay conexión haciendo una llamada a *Network*, en caso negativo se presenta un aviso al cliente (esto no está plasmado en el diagrama de secuencia). Si hay conexión se le pide a *Storage*[26] las versiones y posteriormente también se piden las versiones al servidor, primero la versión de las minas e imágenes "*bigVersion*", que si es mayor la del servidor que la del cliente se añadirá un elemento a la variable buttons. Después de esto se pide la versión de comentarios y usuarios "*lightVersion*", que si es mayor la del servidor que la del cliente sigue el flujo de dentro de la caja en rojo.

En el caso nombrado anteriormente, *AppComponent* llama a una función de *DatabaseService* (El gestor de la BD SQLite de la aplicación), que borra los datos de usuarios y comentarios y los

[26] Storage: Componente que se encarga del almacenamiento de variables persistentes.

pide al servidor, introduciendo los datos devueltos en la base de datos local. El servicio al acabar la tarea, no devuelve un valor sino que manda un mensaje a un hilo que está constantemente escuchando para que guarde en *Storage* la nueva "*lightVersion*".

Para finalizar, en el caso que haya algún elemento en botón (Que es el caso en el que la "*bigVersion*" está desactualizada) se presentará una alerta con un botón, que si se pulsa actualizará los datos de minas e imágenes, repitiendo el proceso de dentro del bloque rojo, pero cambiando *users* y *comments* por *mines* e *images* y actualizando al final "*bigVersion*" en lugar de "*lightVersion*".

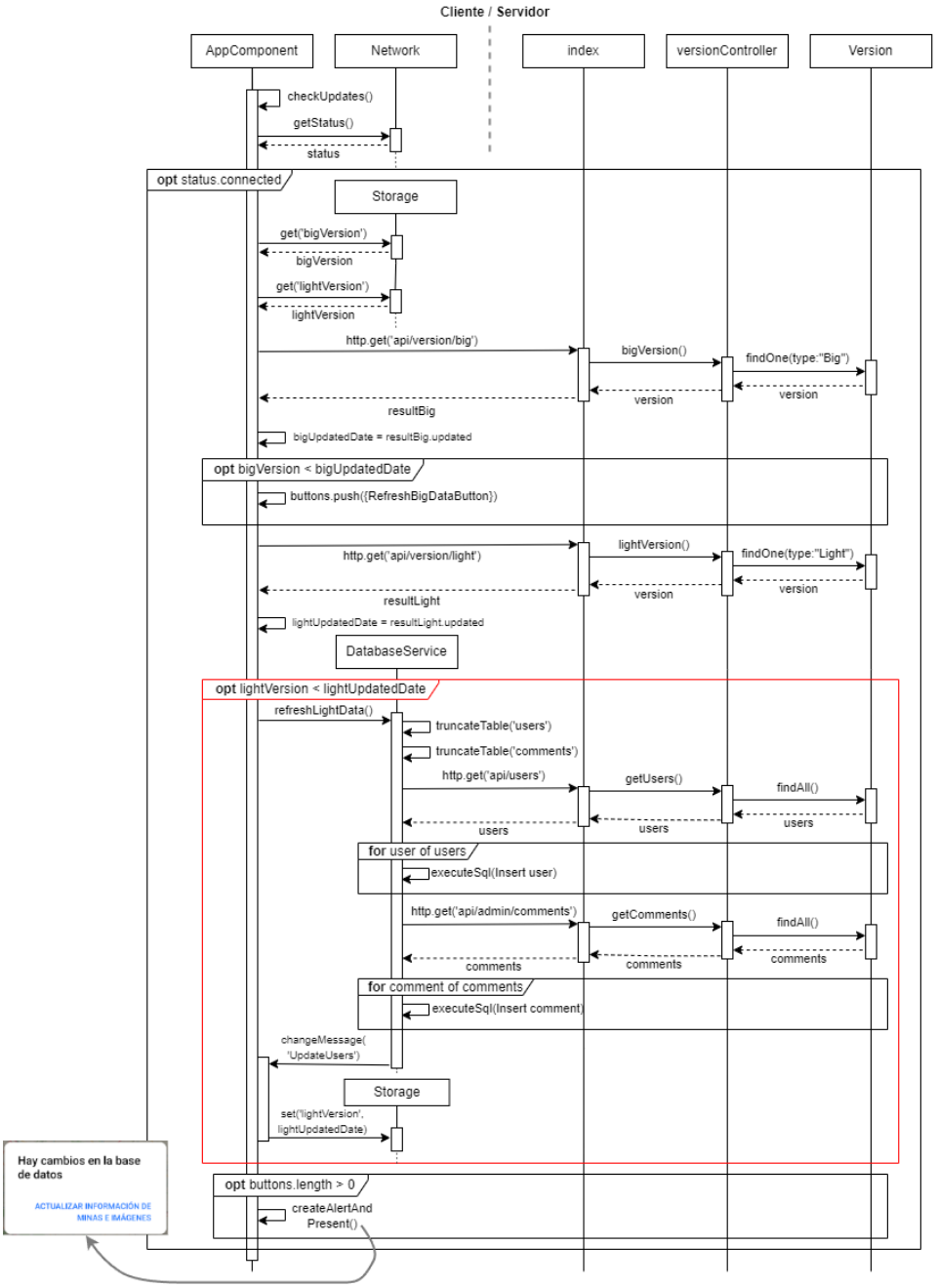


*Figura 6: Diagrama de secuencia de sincronización de datos.*

El otro diagrama de secuencia, visible en la Figura 7 se centra en el enrutamiento. En él se puede ver la frontera entre el cliente móvil y el servidor de *routing*, la secuencia que se ve se inicia cuando se inicia la función *showRoute()* en *MapPage*, lo primero que se hace es preparar y enviar una petición al servidor de *routing*, devolviendo este un resultado, del que nos interesa sólo la geometría de la ruta, la cual se pasa como parámetro a la función *routeLayerShow()*, que se encarga de decodificar la geometría (convirtiéndola a un *array* de coordenadas), convertirla en un objeto que pueda usar Leaflet, eliminar la anterior ruta si la hay y presentarla en el mapa. Además, la geometría y las coordenadas de origen y destino son guardadas en *Storage* y enviadas a *TabsPage* con una llamada a *DatabaseService*, *TabsPage* al recibirla lleva a cabo la función *geocode*, que consiste en conseguir la dirección de las coordenadas de origen y destino, para presentarlas como se ven en la pantalla 10 del mapa de navegación de la Figura 3. La parte más compleja de la implementación de esta funcionalidad ha sido entender cómo funciona la API de *routing*, descrita brevemente en el Anexo VI.

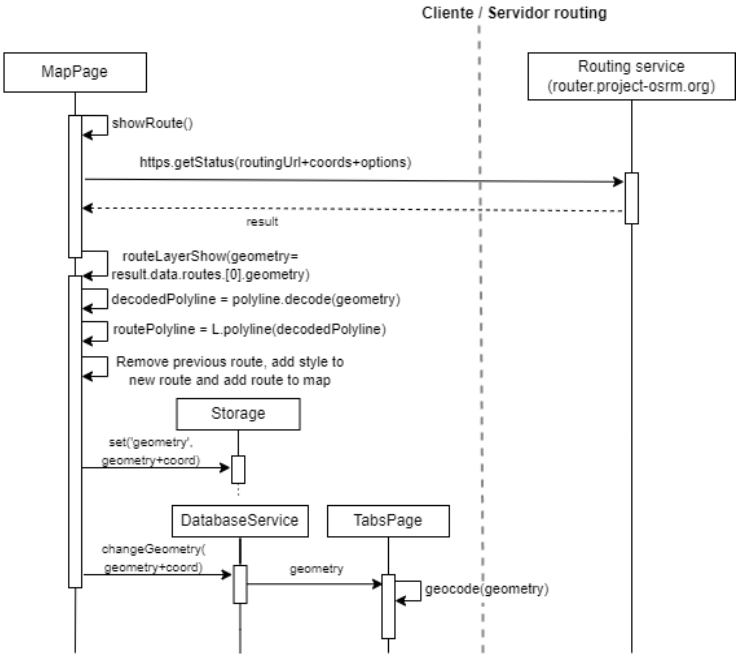


*Figura 7: Diagrama de secuencia de creación de ruta.*

Finalmente, algunos detalles del diseño software de la funcionalidad del *webscrap* se pueden ver en el Anexo V, aunque éste trate principalmente de explicar la relación entre atributos de la base de datos y los campos de la web.

### *Diseño de la base de datos*

Las bases de datos desarrolladas en el proyecto (“DB Cliente” y “DB Servidor”) están basadas en un esquema relacional e implementadas en SQLite. Las dos tienen tablas y campos muy parecidos, pero con algunas diferencias bastante significativas para facilitar su implementación.

El diagrama Entidad Relación de la Figura 8 muestra todas las tablas con sus relaciones y sus campos, sin incluir los que añade SQLite por defecto. A continuación se describen las tablas, su función, la explicación de sus relaciones y de sus campos:

- *User*: Esta tabla guarda los datos de los usuarios, con su campo *username*, un campo booleano *admin* que marca si el usuario es administrador y *password* y *saltSecret* para las contraseñas encriptadas. Como se puede ver en el diagrama, hay una relación 1:N con los comentarios y una relación 1:1 con las imágenes.
- *Mine*: Esta tabla representa la información que se puede encontrar en la web de las minas olvidadas de Aragón. Tiene una relación 1:N tanto con las minas como con los comentarios.
- *Image*: Guarda el nombre de la imagen, los bytes de la imagen en forma de *Blob* y se relaciona 1:1 con *User* y N:1 con *Mine*. Las imágenes están diseñadas para pertenecer a un usuario o a una mina, pero no a las dos al mismo tiempo.
- *Comment*: Guarda el contenido del comentario, su estado (pendiente=0, aceptado=1, rechazado=2) y se relaciona N:1 con *User* y N:1 con *Mine*. Los comentarios siempre pertenecerán a un usuario y a una mina.
- *Version*: Esta tabla no se relaciona con ninguna de las demás, ya que sirve para marcar la última fecha de actualización de las distintas tablas, el campo *type* guarda el tipo de versión (Ahora "*light*" si es de usuarios y comentarios y "*big*" si es de imágenes y minas) y el campo update representa la última fecha de actualización de un dato de ese tipo.

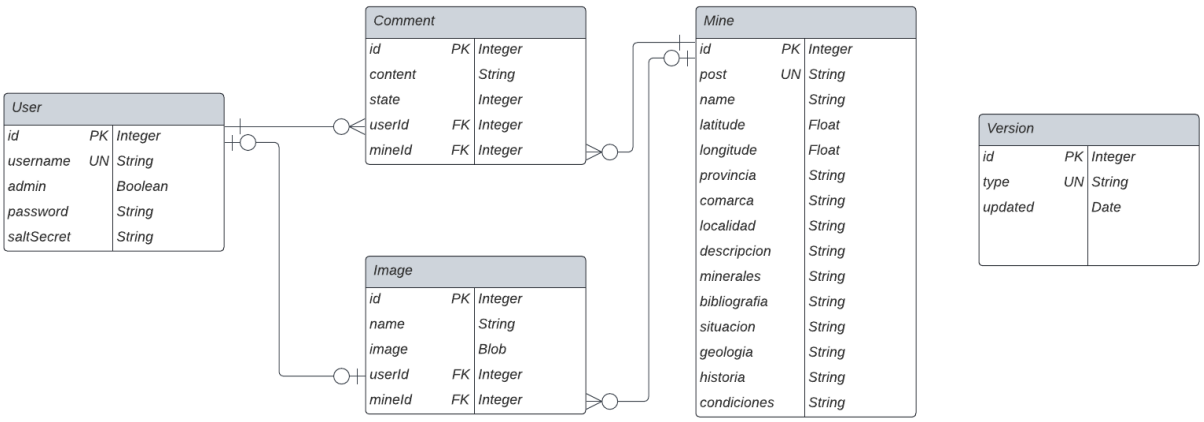


*Figura 8: Diagrama Entidad Relación base de datos Servidor.*

Para cerrar esta sección, en la Figura 9 se puede ver el diagrama Entidad Relación correspondiente a la base de datos del cliente. Es una aproximación más sencilla, de la que se han eliminado las relaciones, para evitar errores en algunos casos, a modo de ejemplo, por motivos de diseño los usuarios y comentarios se sincronizan automáticamente, pero las imágenes y las minas no. Esto quiere decir que en el caso de que en la base de datos del servidor esté la mina con id X pero ese usuario no la tenga en su base de datos, al sincronizar los comentarios, habría un error en caso de que se descargara un comentario asociado a esa mina.

Las otras diferencias notables son la falta de los campos *password* y *saltSecret* en la tabla *User*, obviamente por motivos de seguridad y la falta de *userId* en la tabla *Image*. El motivo de la eliminación de este campo es que no se descargan las imágenes de los usuarios, ya que pueden aumentar enormemente el tamaño de la base de datos. Para poder ver las imágenes de los

demás usuarios se hace una petición al servidor, esto hace que no estén disponibles sin conexión. Por último, la tabla *Version* no está en la base de datos de cliente, pero está implementada mediante el uso de variables de storage-angular, descrito en el Anexo III.

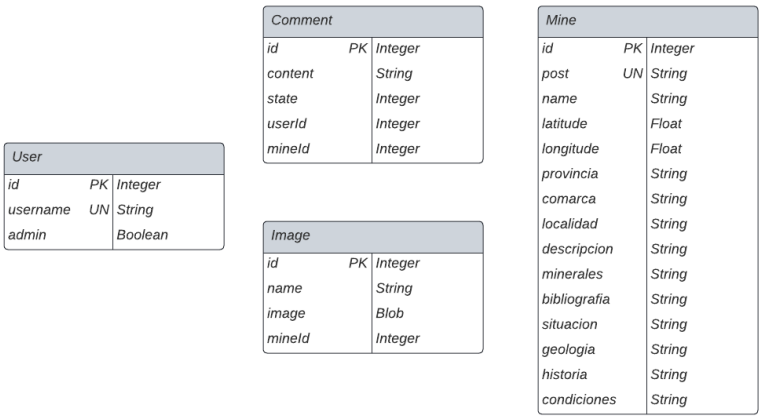


*Figura 9: Diagrama Entidad Relación base de datos Cliente.*

## 2.4 Dimensión del trabajo realizado

En esta sección se hace una comparación de la planificación inicial con el resultado final del porcentaje de horas dedicado a cada tarea relacionada con el proyecto.

Inicialmente, como se puede ver en el gráfico de la Figura 10, se plantearon 5 tipos de tareas:

- "*Front-end*": trabajos relacionados con el desarrollo exclusivo de las aplicaciones móviles;
- "*Back-end*": trabajos relacionados con el desarrollo de back-end, su base de datos y el sistema de scrapping;
- "Integración y despliegue": trabajos relacionados con integrar los diferentes sistemas y desplegar las aplicaciones móviles y las aplicaciones del lado del servidor;
- "Gestión de proyecto": trabajos vinculados a reuniones técnicas y administrativas con los directores;
- "Documentación": tareas de creación de diagramas y escritura de la memoria.


*Figura 10: Gráfico con el porcentaje de horas teóricas a dedicar por tipo de tarea.*


*Figura 11: Gráfico con el porcentaje de horas dedicadas por tipo de tarea.*

Finalmente, en el gráfico de la Figura 11 se ve que los tipos de tareas han quedado intactos, pero comparando los dos gráficos se ven algunas diferencias notables, siendo el "*Back-end*" y la "Gestión de proyecto" los cálculos menos errados.:

- "Front-end": En la estimación inicial se subestimó la dimensión de esta parte, ya que finalmente fue la más compleja, teniendo además un front-end web de administración, que no fue tenido en cuenta al principio.
- "Integración y despliegue": En este tipo de tarea se había tenido en cuenta el tiempo que costaría subir las versiones de las aplicaciones móviles a los distintos *markets*, cosa que no se ha conseguido hacer, aún así ha ocupado más tiempo de lo esperado por falta de experiencia en la tecnología y por algunas conversiones de datos.
- "Documentación": Inicialmente se pensaba que la redacción de la memoria iba a evolucionar a la vez que el desarrollo, lo que se cumplió sólo parcialmente, por lo que se ha evitado gran cantidad de refactorizaciones de los textos introducidos en la memoria escribiendo la mayoría de ellos al dar por terminado el desarrollo.

## 2.5 Aspectos más complejos abordados

Como en todo proyecto de desarrollo con funcionalidades poco frecuentes, algunos elementos pueden ser complicados de implementar, en el caso de este podemos describir varios:

- Capacitor vs Cordova: Tanto Capacitor como Cordova son dos frameworks que actúan como una capa intermedia entre Ionic y las API nativas de los móviles. Aprendiendo Ionic en las fases tempranas del proyecto, se eligió usar Capacitor, ya que actualmente se utiliza más y tiene mejor soporte. El problema surgió cuando algunos paquetes necesarios para el proyecto parecían estar sólo disponibles para Cordova. Finalmente se descubrió que no había un impedimento en usar paquetes de Cordova para un proyecto de Capacitor, sólo había que cambiar ligeramente la forma de instalarlo e importarlo.
- Datos sin conexión para el cliente móvil: Esta funcionalidad añadía cierta complejidad al desarrollo tanto del cliente móvil como del servidor, ya que dejaba de ser la típica aplicación móvil que recoge los datos del servidor a demanda en cualquier momento. Se tuvo que cambiar el diseño, ya descrito en esta memoria, basándose en tener una base de datos en el cliente móvil, con unas variables guardadas que representan la versión de usuarios y comentarios y la de minas e imágenes. Con esto se sincroniza mediante llamadas al servidor y si el usuario tiene conexión y una versión antigua la puede descargar.
- Trabajar dependiendo de un proyecto en construcción: En sí el *webscrap* no fue complicado de implementar, gracias a la experiencia adquirida en otra asignatura de la carrera. El problema fueron los distintos cambios en la estructura de las minas de la web originados debido a que el proyecto de la web de minas olvidadas ha madurado en el mismo período de tiempo que el TFG, por lo que se tuvo que hacer cambios en el *webscrap* cada vez que había un cambio de estructura de la web.
- Caché de capas: El mapa sin conexión fue aparcado durante varios meses desde que se planteó, ya que no se podía encontrar ningún paquete de Ionic o de Angular que ayudara a guardar en caché las imágenes de las capas. Después de cambiar el enfoque se encontró el paquete de npm leaflet.tilelayer.pouchdbcached, descrito en el Anexo III. Este paquete actúa como un *plug-in* de Leaflet, modificando su comportamiento para lograr una funcionalidad de caché de imágenes del mapa, dio algunos problemas para importarlo en el proyecto, pero se pudo solucionar dedicándole algunas horas. Aunque no es una solución perfecta, ya que ralentiza ligeramente la carga de capas cuando hay conexión.

Finalmente, fuera de los problemas de desarrollo, que han sido resueltos de manera satisfactoria, ha habido un problema referente a las tiendas de aplicaciones. Se ha intentado durante meses conseguir cuentas con el nombre de la Universidad para subir la aplicación a los diferentes *markets* de los sistemas operativos más usados, pero no ha sido posible hasta la fecha.

# 3 LECCIONES APRENDIDAS Y CONCLUSIONES

## 3.1 Conocimientos adquiridos

Mediante la realización del presente Trabajo Fin de Grado se han consolidado habilidades de desarrollo *full-stack* general, repasando tecnologías como Docker, Node.js, ReactJs y bases de datos SQLite.

También se han aprendido algunas nuevas tecnologías, principalmente Angular e Ionic, junto con el uso de numerosos paquetes interesantes para estos mismos.

En este proyecto se ha mejorado también en el ámbito de encontrar soluciones tecnológicas existentes y a adaptarlas a las necesidades de cada aplicación. Pero no solo ha habido que encontrarlas, ya que la dificultad real en muchos casos ha sido aprender a usar esas tecnologías, ya sea como en el caso de Leaflet, que al tener una API muy extensa hay que navegar por un mar de atributos y funciones, o el caso del caché o la API de *routing*, que poseían documentación poco extensa y pocos ejemplos de uso.

Finalmente, fuera del nivel de tecnología también se ha adquirido experiencia en gestión de tiempo y de tareas y la interacción con clientes, ya que José Ignacio Canudo ha tenido un papel muy parecido a un cliente real.

## 3.2 Ideas Futuras

Este es un proyecto vivo que va a ser mantenido más allá del tiempo de vida de este Trabajo de Fin de Grado. Por ello, se ha añadido esta sección en la que se recogen una serie de posibles mejoras y nuevas funcionalidades que podrían ser útiles para usuarios y administradores.

- Mantenimiento del *webscrap*: De todas las mejoras planteadas, la única que podría llamarse obligatoria, sería la constante evolución del *webscrap*, ya que la web de Minas olvidadas de Aragón no ha sido terminada todavía y es muy probable que presente cambios que requieran hacer cambios en el código que implementa esta funcionalidad.
- Interacción de usuarios: La interacción de los usuarios actualmente se limita a la creación de comentarios en minas, en un futuro podría plantearse la posibilidad de añadir algunas funcionalidades para aumentar esa interacción. Una de las funcionalidades planteadas es añadir una imagen con los comentarios, para enriquecer la galería de la mina. También se plantea un caso de uso que poseen muchas aplicaciones con sección de comentarios, que es la respuesta a los comentarios de otros usuarios. Otra funcionalidad que enriquecería mucho la aplicación sería el añadido de puntos de interés por parte de los usuarios, para que estos marcaran en el mapa zonas interesantes que tengan que ver con el tema.
- Mejora de rutas: La implementación final de la ruta planificada ha sido muy sencilla, creando una ruta desde un punto del mapa a la mina. Se podría mejorar añadiendo la posibilidad de guardar varias rutas y poner más paradas en cada ruta. Esta

aproximación se puede ver en el Anexo IV, en las historias de usuario de Añadir a la ruta una mina y Calcular ruta planificada. También otra mejora de las rutas que se puede plantear es el intento de añadir la ruta andando hasta la mina, actualmente la ruta acaba hasta donde se puede llegar conduciendo, pero la API del sistema de enrutamiento usado ofrece la posibilidad de calcular la ruta a pie, se ha probado y no presenta un cambio significativo, pero podría suponer una mejora.

- Comunidad de interesados en el tema: Otra aproximación y mejora de la interacción de usuarios puede ser un foro en el que las personas puedan compartir ideas o incluso quedar para visitar minas, ya sea dentro de la aplicación móvil o en la web.
- Moderación: Todas las mejoras de interacción planteadas, requerirán también de nuevas funcionalidades de moderación para los administradores.
- Rollback: Actualmente si después de un *webscrap* se quedan muchos datos en mal estado por el cambio en la estructura de la web, no hay vuelta atrás, por ello se podría plantear una base de datos auxiliar de minas e imágenes para volver a esa versión si ha habido algún problema.

## 3.3 Conclusiones

Los objetivos de este Trabajo de Fin de Grado se basaban en la creación de una aplicación que acompañara a los usuarios durante la visita de las minas olvidadas, lo que se tradujo en unos requisitos que pueden verse en la sección Requisitos del sistema. Estos han sido cumplidos, tanto los funcionales, como los no funcionales, pero ningún trabajo es perfecto y siempre quedan aspectos a mejorar, vistos en la anterior sección y todavía quedan problemas que solucionar, como la subida de la aplicación a los distintos *markets*.

Todo lo anteriormente expuesto no reduce mi satisfacción final con el desarrollo de este TFG, con el que he podido ampliar mis conocimientos de estudiante de Ingeniería Informática en la rama de Ingeniería del Software.

# Anexo I. El proyecto Minas Olvidadas en la Web

Como se ha mencionado al inicio de esta memoria, el proyecto Minas Olvidadas de Aragón pretende devolver a la vida a estas minas olvidadas, recolectando información acerca de sus nombres, su localización (coordenadas o como llegar), los minerales que se extraían, fotografías, y esas pequeñas historias que encierran y se han perdido o son desconocidas para la mayoría. La vía elegida para este retorno a la vida es a través de la divulgación de esta información a través de la Web, que se puede apreciar en la Figura 12.


*Figura 12: Página web del proyecto dentro del espacio del Museo*

En la actualidad, el proyecto tiene registrados casi 1.000 minas, concesiones mineras, canteras (incluidas históricas) y lugares con minerales. El factor de Ciencia Ciudadana es muy importante y se sustenta en más de 80 colaboradores y colaboradoras, desde aficionados y aficionadas, naturalistas, amigos y amigas del museo o empresas privadas, hasta instituciones públicas como las direcciones provinciales de minas de la DGA o los archivos históricos.


*Figura 13: Web del proyecto al inicio del TFG*

Aunque inicialmente se pensó en dar cabida al proyecto (y toda la información que se dispone) dentro del espacio Web que el Museo tiene en el dominio de la Universidad, su crecimiento ha llevado a considerar que la ergonomía que las webs de la Universidad exigen hacen difícil el poder apreciar en su importancia todos los contenidos que se exponen. Es por ello que desde principios del curso 22/23 se está en proceso de desarrollo de una web específica para el proyecto, esta web ha cambiado significativamente a lo largo del tiempo de vida de este TFG, pudiendo ver en la Figura 13 cómo era la web del proyecto al inicio y en la Figura 14 cómo es al final del TFG.

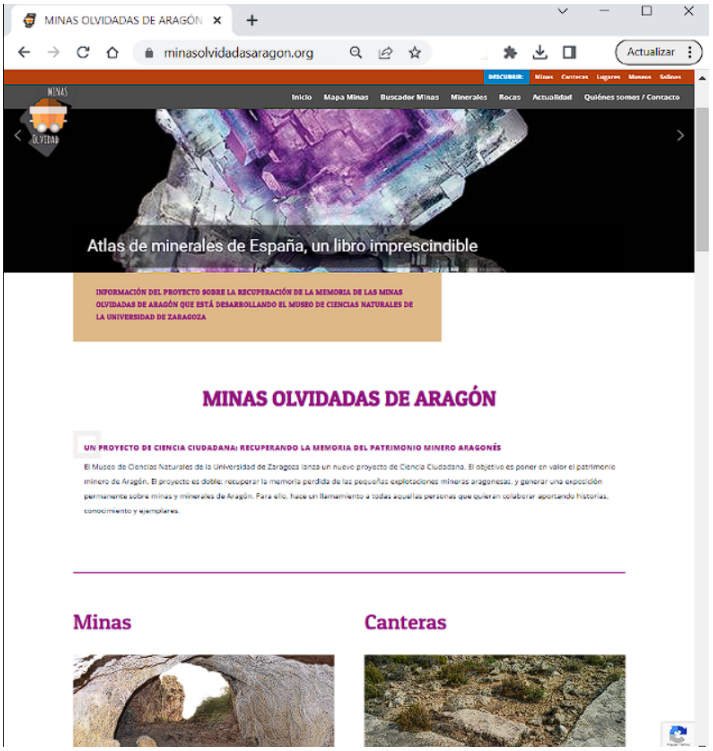


*Figura 14: Web del proyecto al final del TFG.*

# ANEXO II: INTRODUCCIÓN A OPENSTREETMAP

OpenStreetMap (también conocido como OSM) es un proyecto colaborativo para crear mapas editables y libres. En lugar del mapa en sí, los datos generados por el proyecto se consideran su salida principal. Los mapas se crean utilizando información geográfica capturada con dispositivos GPS móviles, ortofotografías[27] y otras fuentes libres. Esta cartografía, tanto las imágenes creadas como los datos vectoriales almacenados en su base de datos, se distribuye bajo licencia abierta Licencia Abierta de Bases de Datos[28] (en inglés ODbL). Los usuarios registrados pueden subir sus trazas desde el GPS y crear y corregir datos vectoriales mediante herramientas de edición creadas por la comunidad OpenStreetMap. Cada semana se añaden 90.000 km de nuevas carreteras con un total de casi 24.000.000 km de viales, eso sin contar otros tipos de datos (pistas, caminos, puntos de interés, etc.). El tamaño de la base de datos (llamada planet.osm) se situaba en julio de 2017 por encima de los 800 gigabytes (58 GB con compresión bzip2).

OpenStreetMap utiliza una estructura de datos topológica[29]. Los datos se almacenan en el *datum*[30] WGS84 lat/lon (EPSG:4326) de proyección de Mercator[31]. Los datos primitivos o elementos básicos de la cartografía de OSM son:

- Los nodos (*nodes*). Son puntos que recogen una posición geográfica dada.
- Las vías (*ways*). Son una lista ordenada de nodos que representa una polilínea o un polígono (cuando una polilínea empieza y finaliza en el mismo punto).
- Las relaciones (*relations*). Son grupos de nodos, vías u otras relaciones a las que se pueden asignar determinadas propiedades comunes. Por ejemplo, todas aquellas vías que forman parte del Camino de Santiago.
- Las etiquetas (*tags*). Se pueden asignar a nodos, caminos o relaciones y constan de una clave (*key*) y de un valor (*value*). Por ejemplo: *highway=trunk* define una vía como carretera troncal.
- Los atributos de los datos siguen un modelo más elaborado que las folksonomías[32] de indexación social. La ontología de las características del mapa (principalmente el significado de las etiquetas) se mantiene mediante una wiki.

El proyecto facilita varias API destinadas a la edición y ampliación de su base de datos, todas ellas se descartan por existir una exclusivamente para el consumo de información. No obstante, esta tiene impuesto el límite de dos peticiones cada 5 minutos y 50000 nodos por cada una de ellas, luego ha sido necesario recurrir a servicios externos que se apoyan en la misma fuente de datos.

[27] https://es.wikipedia.org/wiki/Ortofotograf%C3%ADa
[28] https://es.wikipedia.org/wiki/Licencia_Abierta_de_Bases_de_Datos
[29] https://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n_geogr%C3%A1fica#Modelo_topol.C3.B3gico
[30] https://es.wikipedia.org/wiki/Datum
[31] https://es.wikipedia.org/wiki/Proyecci%C3%B3n_de_Mercator

[32] https://es.wikipedia.org/wiki/Folcsonom%C3%ADa

El consumo de la API de OpenStreetMap en este proyecto se hace mediante *tileLayer* de Leaflet que funciona haciendo varias peticiones HTTPS con esta forma:

https://{**s**}.tile.openstreetmap.org/{**z**}/{**x**}/{**y**}.png

- S: Subdominio al que se pide la imagen, esto sirve para hacer un balanceo de carga y que no se pidan demasiadas imágenes a un mismo servidor.
- Z: El nivel de *zoom*.
- X e Y: Coordenadas de la imagen del mapa al nivel de *zoom* especificado, a más nivel de *zoom* más grandes pueden ser estos números.

# ANEXO III. LOS PAQUETES MÁS IMPORTANTES DE NPM

En este anexo se enumeran los paquetes de npm usados más importantes, con una breve explicación de su papel en el proyecto.

### *Proyecto de back-end Node.js:*

- bcryptjs: Usado para la encriptación y comprobación de las contraseñas de los usuarios mediante *hash* y *salt*.
- compress-images: Este paquete ha proporcionado la funcionalidad de la compresión de imágenes, pudiendo comprimir en formato jpg, png, svg e incluso gif con distintos algoritmos de compresión.
- express: En este paquete se basa la API del *back-end*, gracias a este se puede realizar el enrutamiento de los diferentes servicios.
- fs: Es un módulo incorporado en Node, permite interactuar de forma sencilla con el sistema de archivos local, se ha usado para el *webscrap*, gestionando las imágenes sacadas de la web.
- fs-extra: Paquete que añade funcionalidades del sistema de archivos que no tiene fs, se ha usado para eliminar todas las imágenes de fuera de la base de datos al terminar el *webscrap*.
- jsdom: Ofrece una herramienta que permite simular un entorno de navegador en Node.js, esta herramienta ofrece funcionalidades de búsqueda y parseo de elementos dentro de páginas en HTML, por lo que ha sido la clave de conseguir los datos necesarios mediante el *webscrap*.
- jsonwebtoken: Necesario para la autenticación de usuarios mediante un *token* de sesión, es de los paquetes de npm más utilizados.
- passport, passport-local: Estos paquetes se usan para la autenticación de peticiones, en el caso de esta aplicación se utiliza específicamente para el inicio de sesión.
- puppeteer: Permite automatizar funciones que un humano podría hacer utilizando un navegador, otra herramienta imprescindible para el *webscrap*.
- sequelize: Este paquete ofrece un ORM (Object Relational Mapper, permite manipular tablas de la base de datos como si fueran objetos de un programa) sencillo de utilizar que ha facilitado enormemente la creación y mantenimiento de la base de datos utilizada en el *back-end*.
- sqlite3: Es el *driver* de node para bases de datos SQLite3, la necesita sequelize para trabajar con la base de datos SQLite.

### *Proyecto de front-end de administración React:*

- react: Es una biblioteca de JavaScript para crear interfaces de usuario, se ha usado en combinación con react-dom para hacer el cliente web. Incluye varios paquetes útiles.
- axios: Paquete que permite hacer peticiones, se ha usado en el proyecto para hacer peticiones al back-end desde el cliente de administración.
- universal-cookie: Paquete que facilita el trabajo con las *cookies* del navegador, sólo se ha necesitado para guardar el *token* de autenticación de usuario.

### *Proyecto de front-end móvil Ionic y Angular:*

- @ionic/angular: Es una construcción de Angular que permite utilizar sus componentes en aplicaciones Ionic. Es la base de la interfaz de usuario de la aplicación.
- @capawesome/capacitor-file-picker: Permite al usuario seleccionar un fichero, con este paquete se ha implementado el cambio de foto de perfil.
- @ionic-native/geolocation: Es un paquete que permite geolocalizar el dispositivo que ejecuta la aplicación, se ha usado para mostrar en el mapa dónde está el usuario y para el enrutamiento.
- @ionic-native/in-app-browser: Con este paquete se puede visitar una página web desde la misma aplicación, en nuestro caso la web del Museo de Ciencias Naturales de la Universidad de Zaragoza.
- @ionic-native/native-geocoder: Permite convertir coordenadas geográficas en direcciones y viceversa. Se ha utilizado para hacer más intuitiva la consulta de la ruta.
- @ionic-native/sqlite: Paquete necesario para la creación y uso de bases de datos sqlite en la aplicación. Imprescindible para los datos sin conexión.
- @ionic/storage-angular: Es un paquete que permite almacenar variables en el dispositivo, ha sido útil para el almacenamiento del token de sesión y de la ruta planificada entre otros.
- ionicons: Paquete con gran variedad de iconos muy fáciles de introducir en la interfaz de usuario.
- leaflet: Una biblioteca que permite implementar mapas con gran variedad de funciones en una aplicación. Es la base de todas las funcionalidades en la que se interactúa con el mapa.
- leaflet.tilelayer.pouchdbcached: Es un paquete que modifica la clase L.TileLayer para que pueda guardar en caché las imágenes del mapa, además de guardar en caché partes del mapa que aún no se han mostrado.
- polyline: Permite traducir datos geográficos, ha sido necesario para convertir datos que devuelve la API de enrutamiento a datos que pueda utilizar la biblioteca de Leaflet para mostrar la ruta.
- pouchdb: Es una base de datos ideal para sincronización, ha sido necesaria para que el paquete leaflet.tilelayer.pouchdbcached pueda funcionar.
- rxjs: Es un paquete que ofrece funcionalidades para hacer una aplicación reactiva. Se ha utilizado para pasar mensajes entre distintas clases de la aplicación del cliente.

Además muchos paquetes de @angular y @ionic que se utilizan para pura estética, por lo que se van a omitir.

# Anexo IV. Historias de usuario ligadas a los requisitos

La primera aproximación de los requisitos de la aplicación fueron las historias de usuario, a continuación se van a definir las relaciones entre los requisitos funcionales de la Tabla 1 y las historias de usuario, que se describen más adelante.

- RF1: El requisito funcional se traduce de la historia de Crear perfil, visible en la Figura 15.
- RF2: El requisito funcional se traduce de la historia de Iniciar sesión, visible en la Figura 16.
- RF3: Este requisito funcional se traduce de la historia de Cerrar sesión, visible en la Figura 17.
- RF4: Este requisito funcional se traduce de la historia de Cambiar foto de perfil, visible en la Figura 18.
- RF5: Este requisito funcional se traduce de la historia de Cambiar nombre, visible en la Figura 19.
- RF6: Este requisito funcional se traduce de la historia de Ir a la web, visible en la Figura 21.
- RF8: Este requisito funcional se traduce de la historia de Cambio capa, visible en la Figura 24.
- RF10: Este requisito funcional se traduce de la historia de Ver mina, visible en la Figura 20.
- RF11: Este requisito funcional se traduce de la historia de Buscar mina, visible en la Figura 22.
- RF12: Este requisito funcional se traduce de la historia de Filtrar minas, visible en la Figura 23.
- RF13: Este requisito funcional se traduce de la historia de Comentar en mina, visible en la Figura 25.
- RF14: Este requisito funcional tiene gran similitud con la historias de usuario de Añadir a la ruta una mina y Calcular ruta planificada, de la Figura 26 y la Figura 27 respectivamente, pero fueron descartadas debido a un cambio de diseño en la forma en la que se planifican las rutas. Actualmente sólo se puede calcular una ruta desde un punto hasta una mina, ya que la funcionalidad planteada en las historias de usuario nombradas no iba a ser usada frecuentemente y añadía bastante complejidad.

Los requisitos que no han sido nombrados en esta sección no fueron planteados como historias de usuario en su momento. A continuación se pueden ver todas las historias de usuario planteadas al principio del proyecto, como se puede ver desde el mapa de navegación de la Figura 3, en muchos casos la interfaz del prototipo es bastante similar a la desarrollada finalmente.

### *Crear perfil*

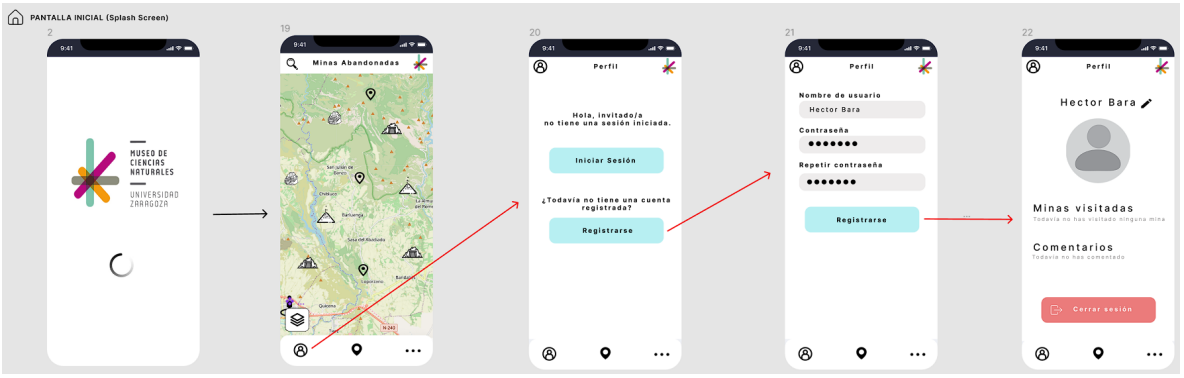


*Figura 15: Historia de usuario Crear perfil*

Para crearse un perfil en la aplicación el usuario después de cargar la aplicación tendrá que pulsar el botón del perfil para ir a la pestaña de perfil, pulsar en el botón “Registrarse” e introducir un nombre de usuario y una contraseña (Y repetir esta en otro campo). Después de esto, pulsa en el nuevo botón “Registrarse” y se creará el perfil y en la misma vista se mostrará la página de su nuevo perfil.

Notas:

- Se necesita haber realizado este paso para llevar a cabo “Iniciar sesión”, “Cerrar sesión”, “Cambiar foto de perfil”, “Cambiar nombre” o “Comentar en mina".
- Si el nombre de usuario que se introduce ya existe en el sistema se mostrará un mensaje de error que pedirá al usuario poner otro nombre.
- Si las contraseñas del campo “contraseña” y “repetir contraseña” no coinciden se mostrará un mensaje de error que pedirá al usuario que los dos campos coincidan.

### *Iniciar sesión*

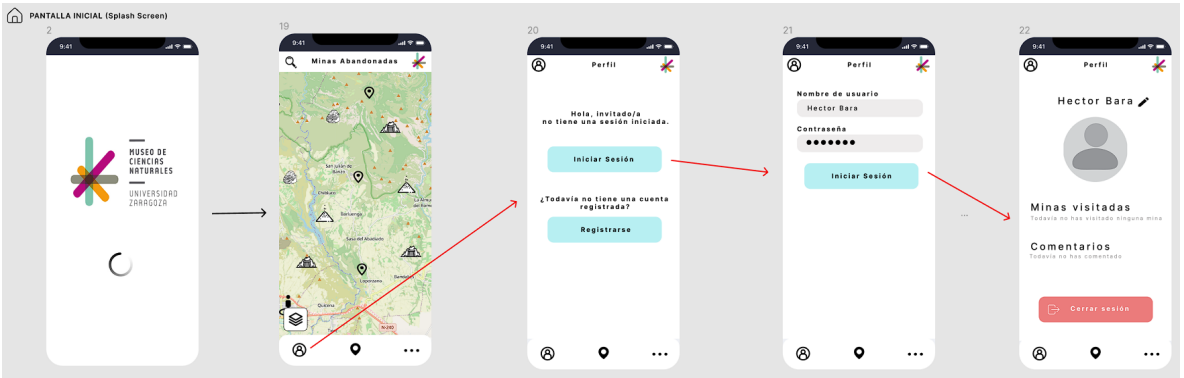


*Figura 16: Historia de usuario Iniciar sesión*

Para iniciar sesión en la aplicación el usuario después de cargar la aplicación tendrá que pulsar el botón del perfil para ir a la pestaña de perfil, pulsar en el botón “Iniciar sesión” e introducir su nombre de usuario y contraseña. Después de esto, en la misma vista saldrá la página de su perfil.

Notas:

- Se necesita haber realizado este paso o el de “Crear perfil” para llevar a cabo “Cerrar sesión”, “Cambiar foto de perfil”, “Cambiar nombre” o “Comentar en mina”.

- Se mostrará un mensaje de error si el nombre de usuario o la contraseña son incorrectos.

### *Cerrar sesión*

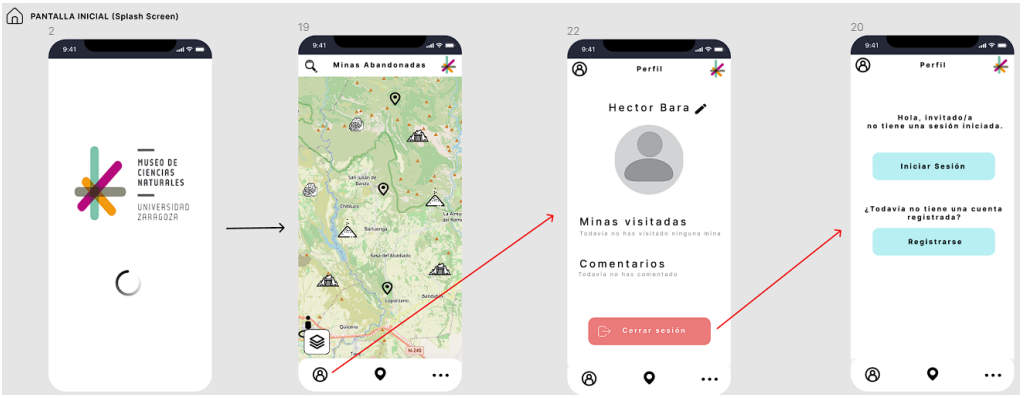


*Figura 17: Historia de usuario Cerrar sesión*

Para cerrar sesión en la aplicación el usuario después de cargar la aplicación tendrá que pulsar el botón del perfil para ir a la pestaña de perfil y pulsar en el botón “Cerrar sesión”. Después de esto, en la misma vista saldrá la página que pide al usuario iniciar sesión o registrarse.
Notas:

- Se necesita haber realizado el paso de “Iniciar sesión” o el de “Crear perfil” para llevar a cabo este paso.

### *Cambiar foto de perfil*

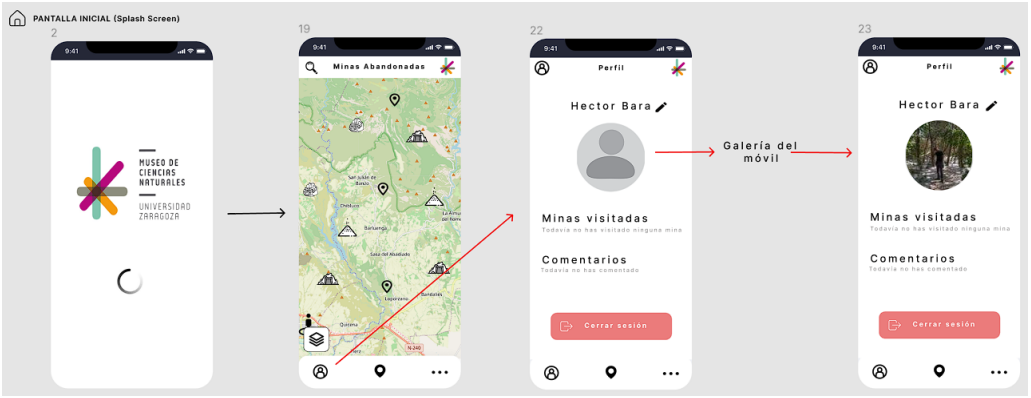


*Figura 18: Historia de usuario Cambiar foto de perfil*

Para cambiar su foto de perfil el usuario después de cargar la aplicación tendrá que pulsar el botón del perfil para ir a la pestaña de perfil y pulsar en el círculo en el que está su foto de perfil. Esto le redirigirá a la galería de su móvil (No se ha puesto captura ya que pueden cambiar mucho dependiendo del SO y modelo del teléfono), tras elegir una imagen de su galería se mostrará el perfil con la foto de perfil cambiada.
Notas:

- Se necesita haber realizado el paso de “Iniciar sesión” o el de “Crear perfil” para llevar a cabo este paso.
- Sólo se permitirán los formatos de foto más usados.

### *Cambiar nombre*

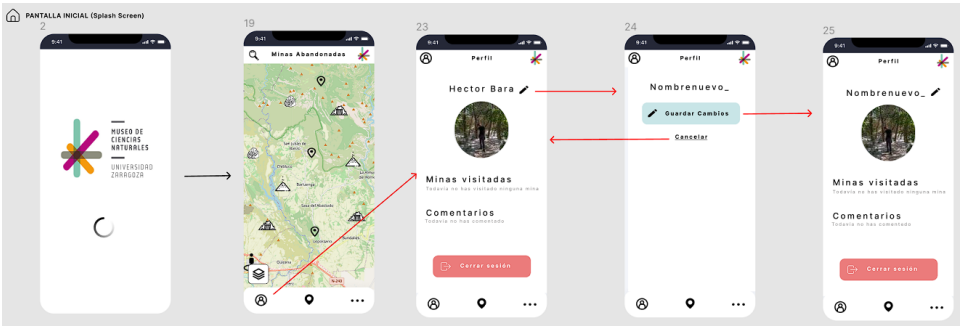


*Figura 19: Historia de usuario Cambiar nombre*

Para cambiar su nombre el usuario después de cargar la aplicación tendrá que pulsar el botón del perfil ☺ para ir a la pestaña de perfil y pulsar en el botón en forma de lápiz que está al lado de su nombre. Después de esto introducirá su nuevo nombre y puede guardar los cambios o cancelar el proceso. En el primer caso su perfil será guardado y mostrado con el nuevo nombre introducido y en el segundo su perfil estará en su estado anterior

Notas:

- Se necesita haber realizado el paso de “Iniciar sesión” o el de “Crear perfil” para llevar a cabo este paso.
- Si el nombre de usuario que se introduce ya existe en el sistema se mostrará un mensaje de error que pedirá al usuario poner otro nombre.

### *Ver mina*

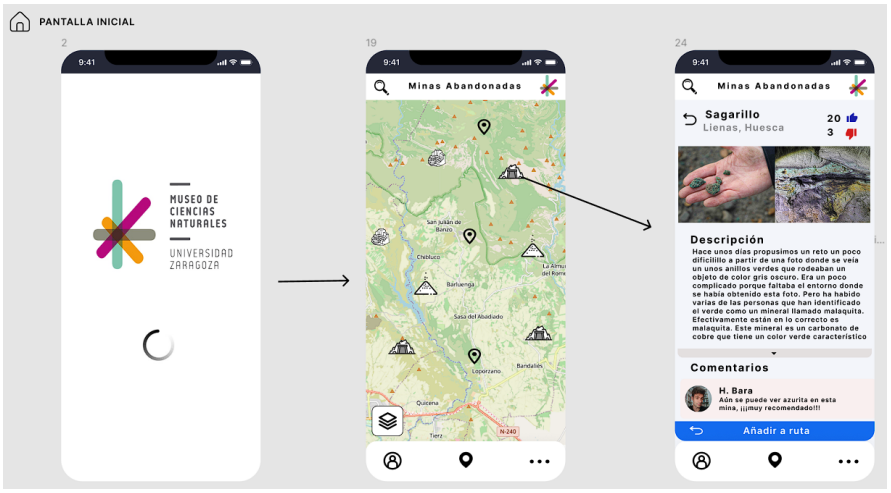


*Figura 20: Historia de usuario Ver mina*

Para ver una mina el usuario después de cargar la aplicación tendrá que pulsar una de las minas que se le muestran en el mapa. Al hacer esto se le mostrará toda la información de la mina, los comentarios sobre esta y la opción de añadir a ruta.

Notas:

- Para buscar minas en concreto, hacer antes el paso de “Buscar mina”.
- Para ver minas que cumplan ciertas condiciones, hacer antes el paso de “Filtrar minas”.

### *Ir a la web*

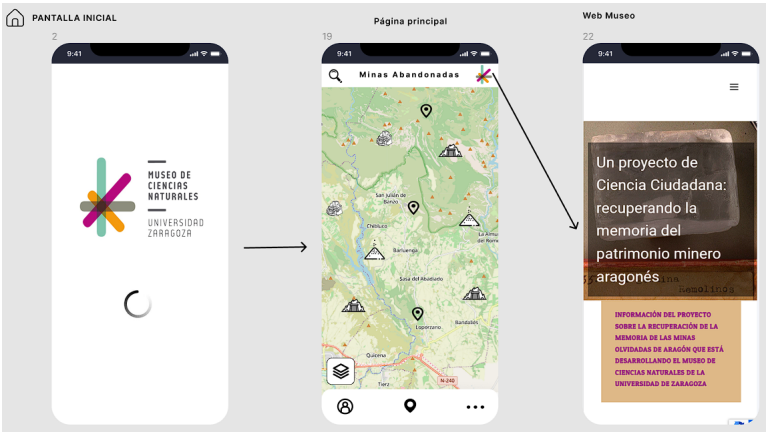


*Figura 21: Historia de usuario Ir a la web*

Para ver la web del museo, el usuario después de cargar la aplicación tendrá que pulsar el botón con el icono del museo de ciencias naturales de unizar. Después de esto se mostrará la web del museo en formato móvil.

Notas:

- Puede que se cambie esta funcionalidad de sitio para poner las opciones extra de ir a https://minasolvidadasaragon.org/ y otras webs que tengan relación con el tema.

### *Buscar mina*

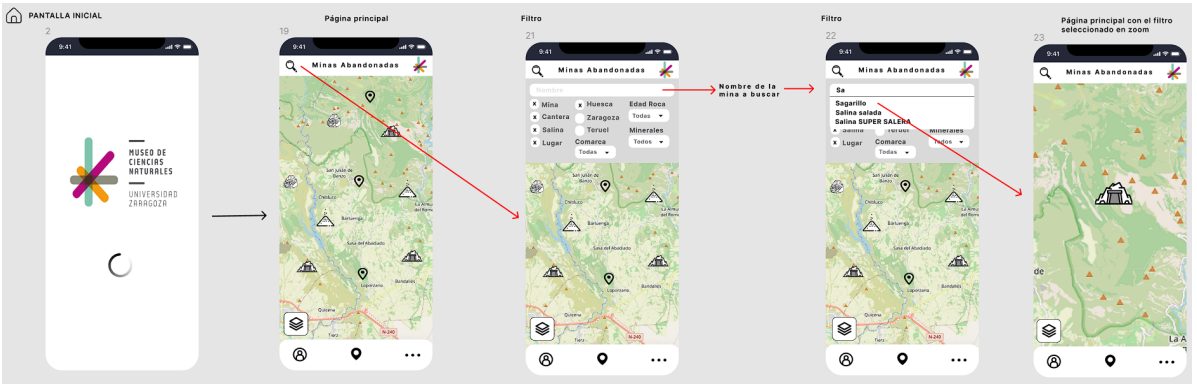


*Figura 22: Historia de usuario Buscar mina*

Para buscar una mina el usuario después de cargar la aplicación tendrá que pulsar el botón en forma de lupa 🔍. Esto mostrará una caja con un cuadro de texto y diferentes opciones de filtrado, el usuario podrá introducir el nombre de la mina o parte de el nombre para que se le muestren los nombres de las diferentes minas que coinciden, si pulsa uno de esos nombres se cerrará la caja y el mapa hará *zoom* a esa mina.

Notas:

- Si se quiere ver la información de esa mina, bastará con pulsarla.

### *Filtrar minas*

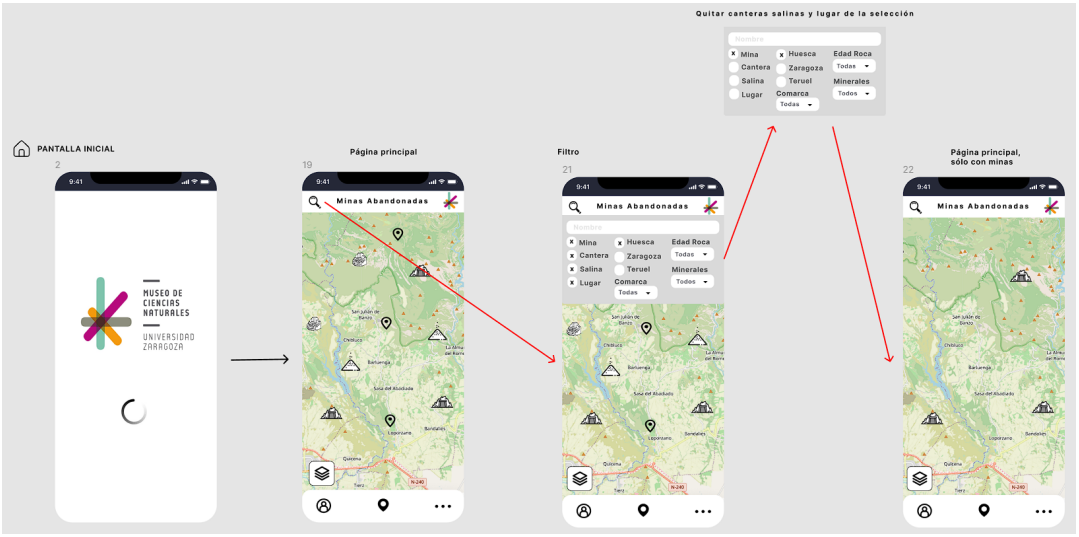


*Figura 23: Historia de usuario Filtrar minas*

Para filtrar las minas el usuario después de cargar la aplicación tendrá que pulsar el botón en forma de lupa 🔍. Esto mostrará una caja con un cuadro de texto y diferentes opciones de filtrado, el usuario podrá editar los distintos campos para encontrar las minas que le interesen.

Notas:

- Si se quiere ver la información de alguna de esas minas, bastará con pulsarla.
- Aún quedan por determinar esas opciones de filtrados, pero se puede hacer una idea con la imagen.

### *Cambio capa*

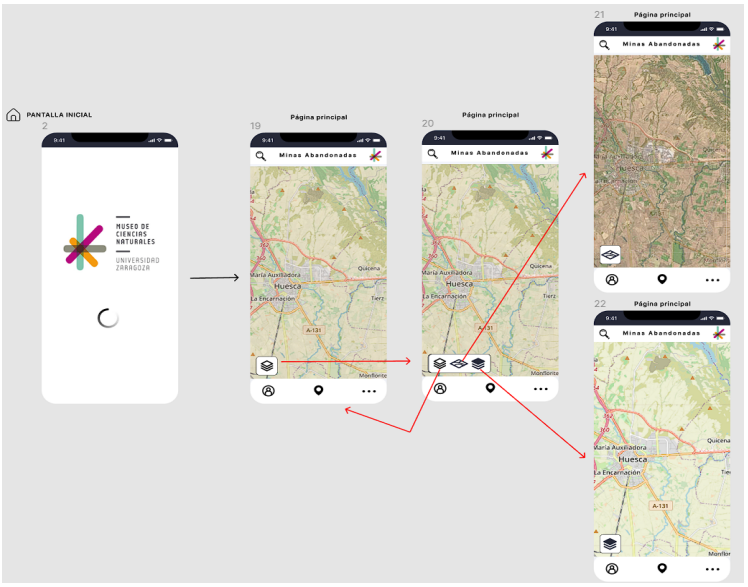


*Figura 24: Historia de usuario Cambio capa*

Para cambiar de capa el usuario después de cargar la aplicación tendrá que pulsar el botón de capas . Esto mostrará las demás opciones de capas que hay, el usuario deberá elegir una y pulsarla, después de esto se mostrará la nueva capa.

Notas:

- En principio hay 3 opciones de capas, la que muestra una mezcla del mapa de openstreetmap con las ortofotos del pnoa, la que muestra sólo el mapa de openstreetmap y la que muestra sólo las ortofotos del pnoa.
- Como predeterminado se mostrará la capa que mezcla los mapas

### *Comentar en mina*

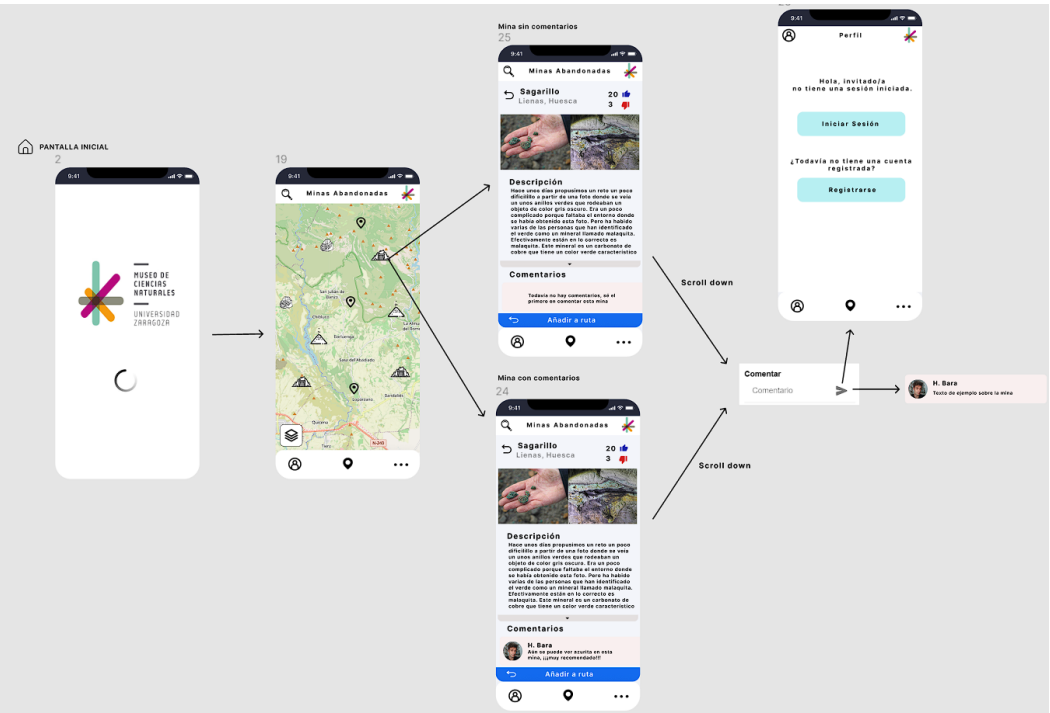


*Figura 25: Historia de usuario Comentar en mina*

Para comentar en una mina el usuario tendrá que seguir los pasos de “Ver mina”. Después de esto, realizará un scroll hacia abajo hasta encontrar la caja de comentarios, en esta escribirá el comentario que desee y le dará a la flecha (que es el botón de enviar), el comentario se guardará y se mostrará la página de la mina con el comentario añadido.
Notas:

- En la imagen se ve un caso en el que ya hay comentarios y otro en el que todavía no hay ningún comentario en esa mina.
- Se necesita haber realizado el paso de “Iniciar sesión” o el de “Crear perfil” para llevar a cabo este paso. Si no se ha hecho y se intenta enviar un comentario se redireccionará a la página del perfil para que el usuario inicie sesión o se registre.

### *Añadir a la ruta una mina*

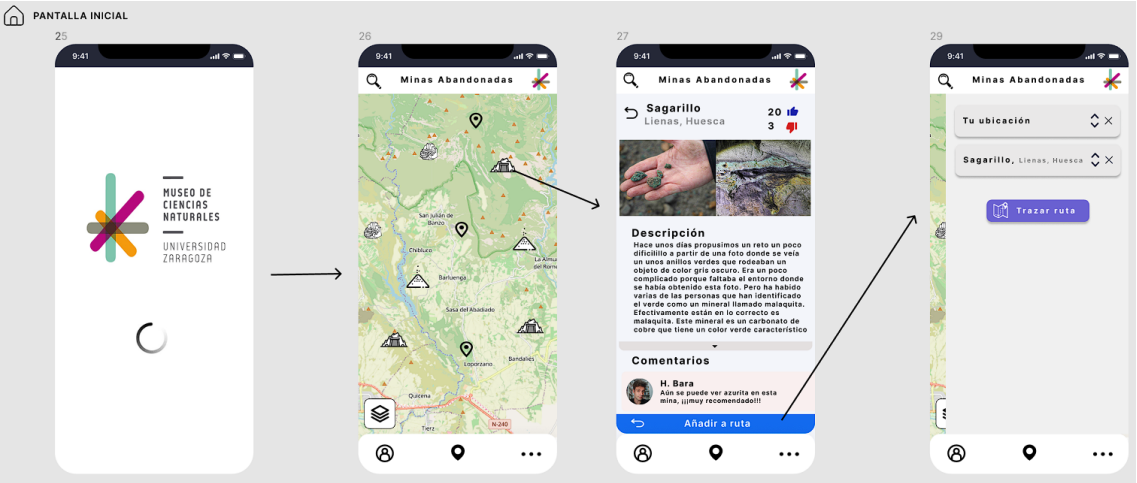


*Figura 26: Historia de usuario Añadir a la ruta una mina*

Para añadir una mina a la ruta el usuario tendrá que seguir los pasos de "Ver mina". Después de esto, pulsará el botón "Añadir a ruta" y a continuación se abrirá un menú en el que se mostrarán los destinos actuales de la ruta planificada.

Notas:

- Más información en "Calcular ruta planificada".

### *Calcular ruta planificada*

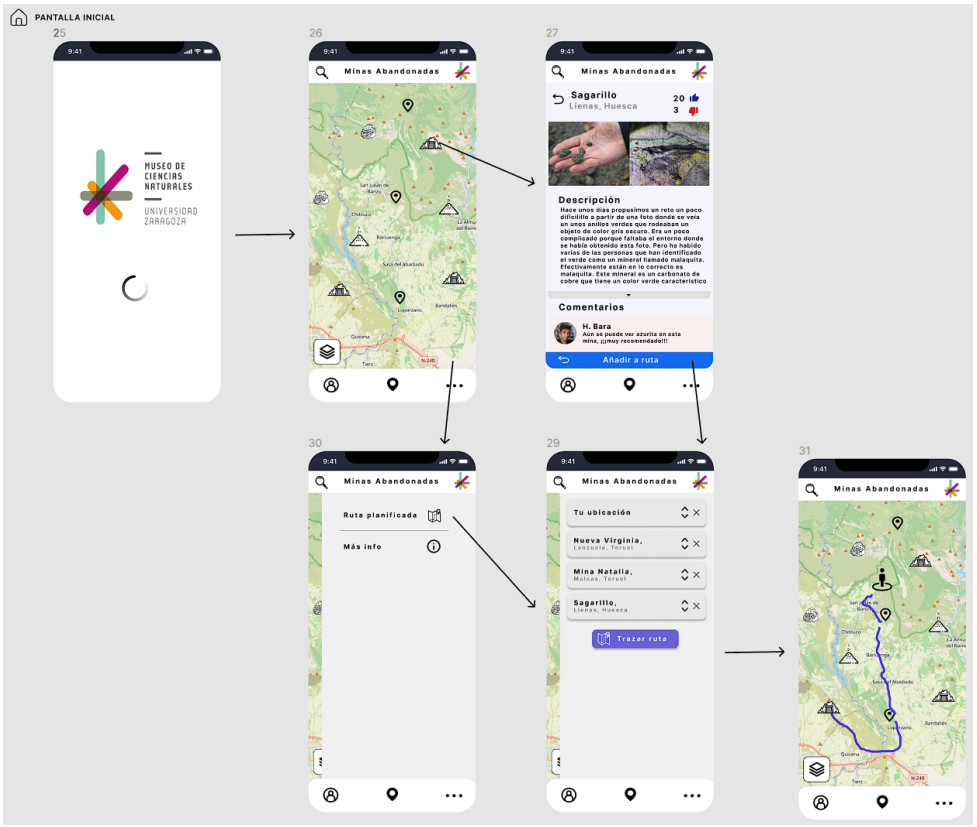


*Figura 27: Historia de usuario Calcular ruta planificada*

Para esta historia hay dos posibles flujos. Uno de ellos es seguir los pasos de “Añadir a la ruta una mina”, el otro es, que después de cargar la aplicación, el usuario pulse el botón de opciones ⋮ y seleccione la opción de “Ruta planificada”. A partir de aquí los flujos convergen, y el usuario podrá cambiar el orden de la ruta con las flechas ⇕ o eliminar una mina de la ruta con la ✕. Cuando la ruta esté al gusto del usuario, podrá pulsar el botón de “Trazar ruta” y se le mostrará el mapa con el trayecto planificado.

Notas:

- El segundo flujo descrito requiere haber añadido alguna mina a la ruta planificada previamente.

# ANEXO V. RELACIÓN DE LOS CAMPOS DE LA BD CON LA WEB DE MINAS OLVIDADAS

Para hacer el *webscrap* hay que tener claras las relaciones entre los campos que hay en la web y los campos de la base de datos. La mayoría de campos son traducciones directas, pero hay dos campos que son un poco más complejos. En la Figura 28 se puede ver la tabla de minas de la base de datos y las partes importantes de la web, con sus campos relacionados mediante números.

Descripción (8.): En algunos casos está separado entre varias etiquetas html, por lo que se hace una búsqueda de todas las etiquetas <p> que no estén vacías ni pertenezcan a los demás campos.

Condiciones (14.): Había varias implementaciones posibles, ya fuera hacer otra tabla de condiciones con relación N:N con las minas, o poner varios campos por cada condición que hubiera en la misma tabla. Finalmente la aproximación se basó en crear un string en la base de datos con una posición por cada condición, haciendo que si está activada esa condición en la posición correspondiente del string haya un 1 (En el caso de la Figura 28 sólo habría un 1 en la posición correspondiente a “Visitable”).

Otra tabla de la que se consiguen datos de la web es la de imágenes, para cargar todos sus datos mediante *webscrap* se buscan todas las etiquetas <img>, pero sólo se guardan las necesarias, ya que hay algunas imágenes que no nos interesa guardar, como los logos de las entidades responsables del proyecto o los de las condiciones. Estas imágenes se descargan a una carpeta, se comprimen y se guardan en otra y finalmente se borran después de introducirlas en la base de datos.

Por último, cabe puntualizar que la forma de encontrar todas las minas de la aplicación para hacer el *webscrap* se hace mediante puppeteer, descrito en en Anexo III. Con puppeteer se navega desde la página https://minasolvidadasaragon.org/minas/ hasta todas las páginas de las minas, que se pueden ver en la Figura 29 y se saca la información de cada una de ellas.

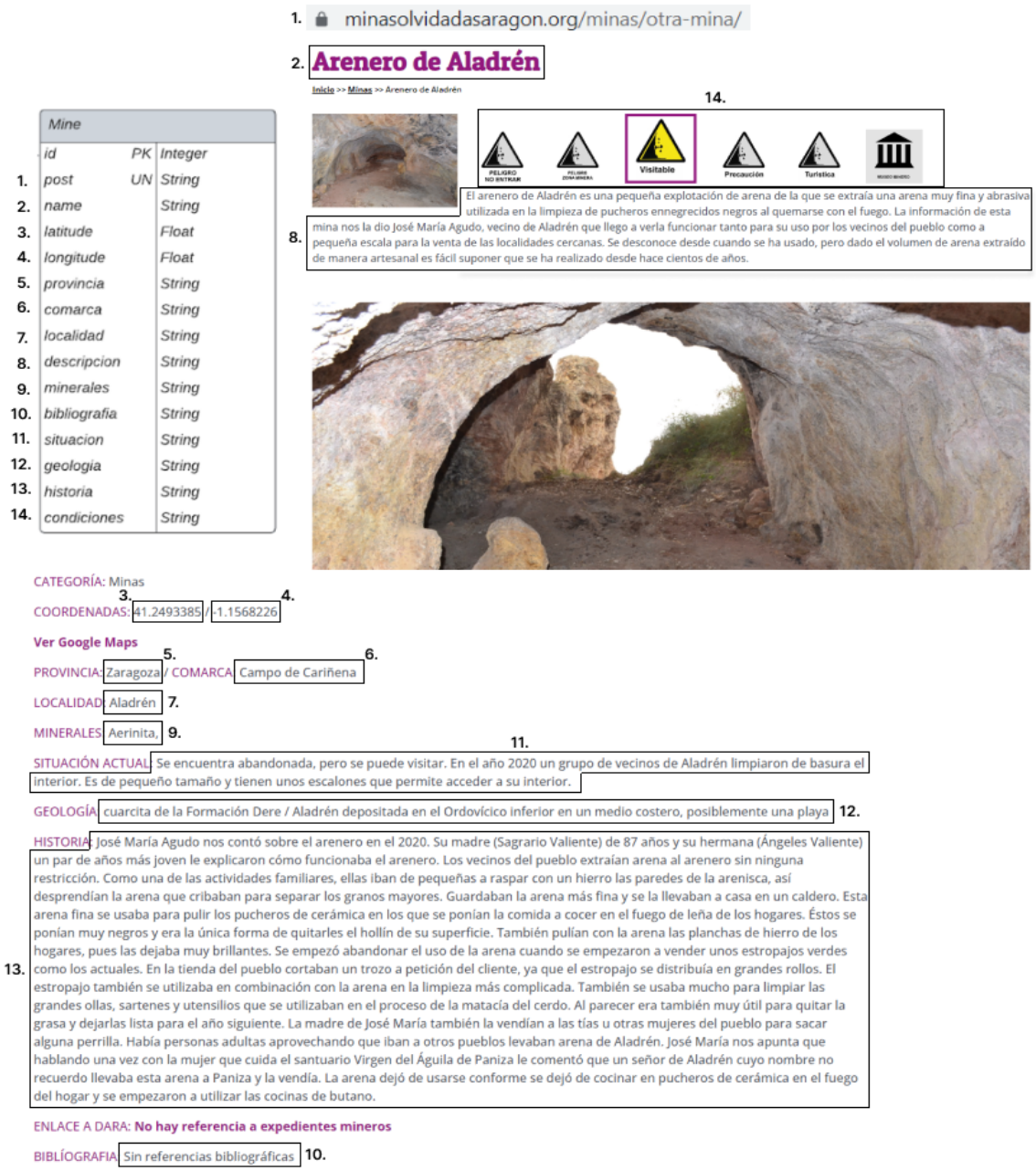


*Figura 28: Relación campos de la tabla minas con campos de la web*

**Arenero de Aladrén**
por **José Ignacio Canudo** | 31/03/2023 | **Minas**

El arenero de Aladrén es una pequeña explotación de arena de la que se extraía una arena muy fina y abrasiva utilizada en la limpieza de pucheros ennegrecidos negros al quemarse con el fuego. La información de esta mina nos la dio José María Agudo, vecino de Aladrén...

**Cantera de La Valfonda**
por **José Ignacio Canudo** | 12/04/2023 | **Canteras**

Luis Moliner: Fabricación in situ de sillares, arcos, soportales y peldaños para su utilización en la arquitectura urbana local y también en las algunas masías. En el núcleo urbano de Alcorisa pueden reconocerse en la escalinata de acceso a la puerta norte de la...

**Mina Ana**
por **José Ignacio Canudo** | 02/04/2023 | **Minas**

Es una importante mina de galena como demuestra el gran el volumen de material extraído y el tamaño de las escombreras. Los diferentes autores que han publicado sobre ella apuntan que probablemente se explotaban desde hace siglos, pero no está documentada su uso...

**Mina de plata de Aliaga**
por **José Ignacio Canudo** | 12/04/2023 | **Minas**

Desde el siglo XVIII hay referencias a una enigmática mina de plata en la localidad de Aliaga. Hay documentos históricos que la citan, pero los materiales geológicos del entorno de Aliaga no son los adecuados para contener mineralizaciones de plata. Recientemente lo...

**Mina Emancipadora / Coto Minero de Esera**
por **José Ignacio Canudo** | 11/04/2023 | **Minas**

Guillermo Aparicio fue quien nos dio la primera noticia de esta importante mina de piritas en el Valle de Benasque (Pirineo de Huesca). Lo que queda de ella se encuentra a unos 600 m al Noreste de Cerler. Se puede acceder por un camino que comienza detrás del Hotel HG...

**Mina La Pedraza / Ceres / Marcial / La independiente**
por **José Ignacio Canudo** | 11/04/2023 | **Minas**

El geólogo Alejandro Lorenzo García ha realizado su TFG en el año 2023 sobre las minas de La Pedraza en Bubierca (Zaragoza), con la dirección de los profesores Isabel Fanlo y Alfonso Yuste. Nos ha enviado una interesante nota sobre estas minas. Se encuentran al...

**Mina Rosario**
por **José Ignacio Canudo** | 02/04/2023 | **Minas**

La explotación de la fluorita en el entorno del puerto de Portalet en Sallent de Gállego (Huesca) posiblemente comenzó en el siglo XVIII. Aunque la extracción sistemática se hizo a principios de la década de 1960. Fue el momento en que se explotaban varias minas a...

**Minas de Libros**
por **José Ignacio Canudo** | 12/04/2023 | **Minas**

El conjunto de minas de azufre de Libros están actualmente abandonadas, pero llegaron a tener una gran importancia como atestigua el gran desarrollo de sus galerías y las miles de toneladas de azufre que se obtenían cada año. Son también famosas por haberse encontrado...

*Figura 29: Página desde donde se ven todas las minas.*

# ANEXO VI. API DE ENRUTAMIENTO

En la documentación de la OSRM API[33] hay varias llamadas interesantes, como una llamada que te devuelve puntos de interés cercanos de OpenStreetMap, u otra que devuelve capas vectoriales con geometrías de carreteras, pero la que necesita este proyecto es el servicio de rutas. La llamada a este servicio es una petición GET que se muestra de la siguiente manera en la documentación:
*/route/v1/{**profile**}/{**coordinates**}?**alternatives**={true|false|number}&**steps**={true|false}*
*&**geometries**={polyline|polyline6|geojson}&**overview**={full|simplified|false}*
*&**annotations**={true|false}*

Como se puede ver hay bastantes variables, a continuación se explica qué significa cada una y que opción se ha escogido:

- *Profile*: Es el modo de transporte, en nuestro caso "*driving*" ya que pueden ser largas distancias, aunque en un futuro se podría complementar con "*foot*" hasta donde no llegue el coche.
- *Coordinates*: Formato longitudOrigen, latitudOrigen; longitudDestino, latitudDestino.
- *Alternatives*: Booleano o número, que marca si se quieren enviar rutas alternativas, no es nuestro caso así que en la petición no se pone esta variable ya que está a *false* de manera predeterminada.
- *Steps*: Devuelve los pasos a seguir en la ruta, no los necesitamos por lo que hacemos como en el caso de *alternatives*.
- *Geometries*: Formato de la geometría a devolver, puede ser *polyline*, *polyline6* o *geojson*. No se ha usado la variable porque el predeterminado es *polyline*, que nos sirve.
- *Overview*: Nivel de detalle de la línea, *full, simplified* o *false*. La hemos puesto a *full*, ya que la predeterminada que es *simplified* ofrece un nivel de detalle muy pobre.
- *Annotations*: Metadatos adicionales, mismo caso que *alternatives* y *steps*.

Independientemente de las opciones que pongamos, la API siempre devolverá los siguientes atributos:

- *Code*: Código de estado para marcar si la respuesta ha sido correcta o ha fallado.
- *Waypoints*: Un *array* de puntos de ruta en orden. No se ha necesitado usar.
- *Routes*: Un *array* de objetos Route ordenados en orden de recomendación, con la opción de alternativas a *false* no se devuelve más que una, que es nuestro caso.

[33] https://project-osrm.org/docs/v5.24.0/api/#

# ANEXO VII. DESPLIEGUE CON DOCKER

Para desplegar el contenedor de Docker en un principio se pensó meter dentro de un servicio de docker tanto el front-end web como el *back-end*, pero al dar problemas al instalar los paquetes de Docker se tomó la decisión de separarlo en dos servicios diferentes en el mismo contenedor, lo que tenía mucho más sentido.

Para esta aproximación final, se ha creado un fichero docker-compose.yml, que con el comando "docker compose up --build" se crean dos servicios copiando las carpetas necesarias desde local hasta el contenedor, inicializando las variables de entorno necesarias (puertos, ip del servidor privado virtual etc.). Los dos servicios se crean a partir de sus respectivas dockerfiles, front.Dockerfile y back.Dockerfile. En estos dockerfiles están los comandos a ejecutar, ambos crean un espacio de trabajo con el comando "WORKDIR", se copian los ficheros de dependencias de Node con "COPY", se instalan los paquetes con "RUN npm install" y finalmente se copian todos los demás ficheros y se lanzan las respectivas aplicaciones con "CMD [ "npm", "run", "prod" ]".

La única diferencia entre los dos dockerfiles, ha sido una pequeña configuración que se ha hecho en back.Dockerfile para que funcione el *webscrap* correctamente. En la Figura 30 se puede ver el contenedor de docker con los servicios lanzados con el comando "docker compose up --build" anteriormente nombrado en Docker Desktop.

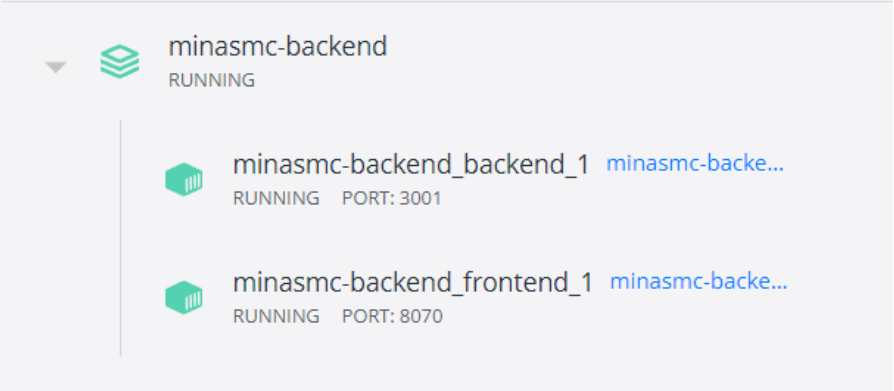


*Figura 30: Visualización de los servicios desplegados en Docker Desktop.*