Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

Diseño y desarrollo de un servicio de gestión de planos de edificios basado en OpenStreetMaps

Autor:
D. Iván González Rincón

Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

Diseño y desarrollo de un servicio de gestión de planos de edificios basado en OpenStreetMaps

Autor:
D. Iván González Rincón

Tutores:
Dr. Diego R. Llanos Ferraris
D. Antonio Román López

Resumen

El siguiente Trabajo de Fin de Grado consiste en el desarrollo de un sistema de gestión de planos de edificios basado en el estándar de OpenStreetMaps, compuesto por el desarrollo de una parte de lógica de servidor, una interfaz web que aproveche ese servicio y el despliegue e integración de todo el sistema con un conjunto de utilidades que actualmente se encuentran en producción.

A partir de esta premisa surgen las bases para este proyecto: utilidades que permitan a un administrador del sistema realizar toda la gestión del mismo, pudiendo realizar altas, bajas y modificaciones de todos los elementos que lo componen, permitiendo gestionar de forma sencilla y gráfica la información relacionada con proyectos y el contenido de los mismos. Además se permitirá la visualización completa del estado del sistema por parte del administrador, lo que facilitará su trabajo considerablemente.

Por otra parte, se permitirá usar los mapas generados en el resto de sistemas que se encuentran en producción, pudiendo visualizar las imágenes procesadas por el sistema antes de ser usadas por las otras partes que se encuentran ya desplegadas.

Se escogió realizar un sistema de estas características ya que hasta ahora la generación de elementos cartográficos no permitía su visualización hasta que no se usaban los mapas generados en otras partes del sistema, y además, toda la generación se realizaba manualmente, lo que resultaba considerablemente tedioso, dificultando todas las tareas de creación y gestión de mapas.

Abstract

The following Final Degree Project consists in the development of a building floorplan management system, based on the OpenStreetMaps standard, composed by the development of a server logic, a web interface that takes advantage of this service and the deployment and integration of the entire system with a set of utilities that are currently in production.

Based on this premise the features for this project arise: utilities that allow a system administrator to carry out all the management of the entire system, with which can make additions, cancellations and modifications of all the elements that compose the system, allowing to manage in a simple and graph the information related to projects and their content. In addition, it will allow the complete view of the system status by the administrator, which will facilitate the administration tasks considerably.

On the other hand, it will be allowed to use the maps generated in the rest of the systems that are in production, being able to visualize the images processed by the system before being used by the other parts that are already deployed.

It was chosen to make a system with these characteristics since until now the generation of cartographic elements didn't allow their visualization until the maps generated are used in other parts of the system, and also, the entire generation was done manually, which was considerably tedious, making all the tasks of creating and managing maps difficult.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto y motivación

Hoy en día los sistemas de posicionamiento y localización tienen una importancia más que notable en nuestra sociedad, ya que con la extensión masiva de los Smartphones se ha masificado el uso de sistemas GPS, cómo el famoso Google Maps. Todo este tipo de software está apoyado por sistemas de mapas, con los que se muestra la información que necesita el usuario de forma útil y precisa.

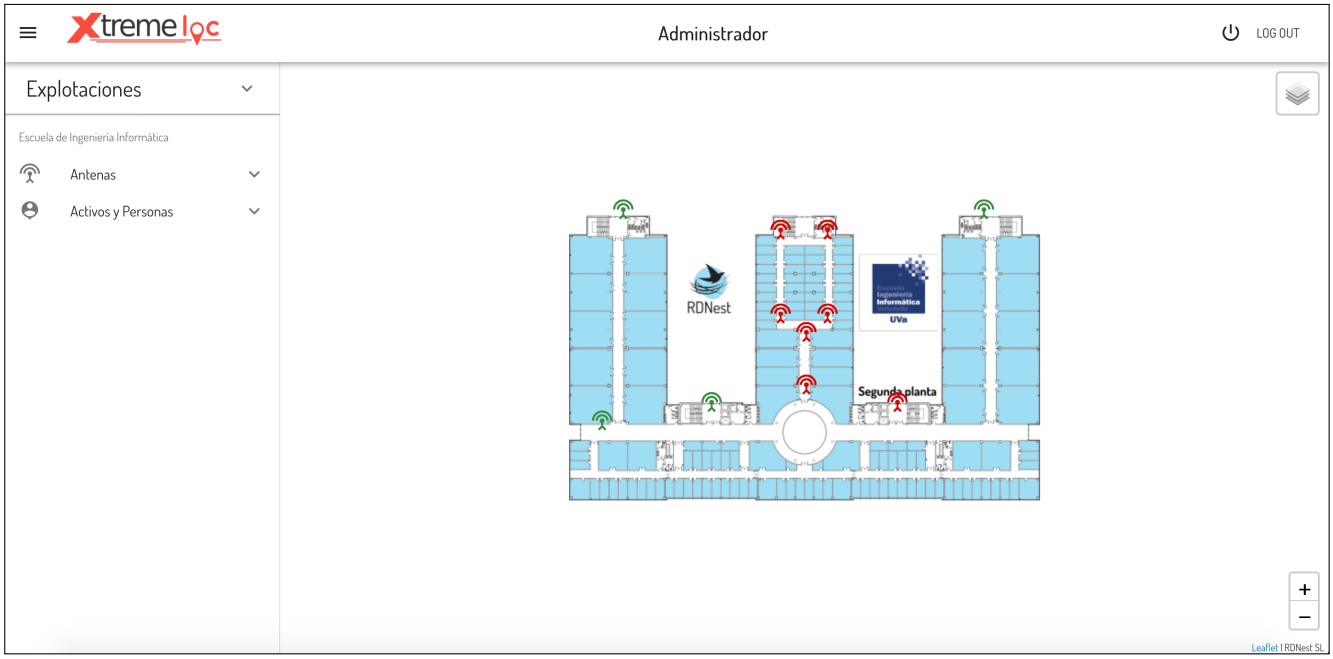

Figura 1.1: Mapa de la Escuela de Ingeniería Informática de Valladolid

El presente TFG está motivado precisamente por esta necesidad. Cualquier software que entre en la categoría anteriormente mencionada necesita una cartografía en la que poder basarse para mostrar su información al usuario. Este es el caso de la aplicación XtremeLoc, aplicación desarrollada por el Grupo Trasgo de la Universidad de Valladolid orientada a la localización en interiores y comercializada por RDNest SL, que se apoya en una serie de mapas para poder mostrar con detalle la información que presenta la aplicación(tal cómo se puede observar en la figura 1.1). Se buscará ampliar sus funcionalidades y llevar a cabo el desarrollo de un sistema auxiliar que facilite la tarea de generar los elementos necesarios para poder mostrar la información cartográfica que el sistema precisa.

A lo largo del presente documento se detallarán todas las fases que forman parte del desarrollo del proyecto, desde la elicitación de requisitos hasta la implementación total de la aplicación.

1.2. Objetivos

El objetivo del presente Trabajo de Fin de Grado es el de llevar a cabo el desarrollo de una herramienta de generación y gestión de mapas de interiores de edificios, basado en el estándas de OpenStreetMaps, de modo que facilite y mejore la administración de este tipo de contenido para sistemas de posicionamiento de interiores, entre los que se encuentra la citada aplicación XtremeLoc, ya que es un elemento considerablemente importante para que este sistema muestre correctamente su información al usuario. Por otro lado, también se busca mejorar el estado del servidor de mapas con el que actualmente se cuenta en producción, por lo que se trabajará en esta línea, en conjunto con el desarrollo del resto de características del sistema a implementar.

Para conseguir todo esto se deberá implementar una parte de lógica de servidor que se encargue de toda la carga de generación de información cartográfica, una interfaz web que aproveche esa implementación y permita de una forma fácil realizar las diversas tareas que requiere el sistema (generación y administración de mapas, administración de proyectos, etc...) y la integración de todo ello junto con las aplicaciones que actualmente se encuentran en producción, resultando todo ello en el despliegue del sistema en un entorno totalmente real. Las tareas a desarrollar son las siguientes:

- **Desarrollo de un back-end**: La función principal del sistema a desarrollar es la de facilitar la generación de mapas para un sistema que actualmente se encuentra en producción. Esta tarea será llevada a cabo por la parte correspondiente en la lógica del servidor, que se encargará de generar mapas de interiores de edificios en base a una imagen dada, y respetando el estándar que actualmente usa dicho sistema.

  Aunque lo principal es la carga de generación de mapas, el backend también permitirá gestionar información asociada a proyectos que contienen clientes, dichos clientes tendrán edificios en los que tienen activos los proyectos, y cada edificio tendra una serie de plantas, las cuáles podrán visualizarse y generarse con el sistema desarrollado.

- **Desarrollo de un front-end**: Apoyado por lo anterior, se llevará a cabo el desarrollo de una interfaz web sencilla, que permita realizar fácilmente todas las tareas de administración que permite el sistema.

- **Integración del sistema**: Cómo ya se ha mencionado, el presente TFG se trata de un añadido a un sistema que actualmente se encuentra en producción, por lo que una vez finalizado su desarrollo, deberá integrarse en su totalidad con las herramientas con las que se cuenta actualmente.

Dicho esto, el presente TFG se llevará a cabo por la necesidad existente de facilitar las tareas de administración en relación a los mapas que usa el sistema en producción ya mencionado, pues hasta ahora esta tarea resultaba considerablemente tediosa.

1.3. Punto de partida

Actualmente se parte de un sistema de generación y gestión de mapas manual, que está compuesto por un servidor Apache que se encarga de servir los mapas con los que se trabaja, y un Script en Python que se

encarga de la generación de los mismos. El funcionamiento actual consiste en generar información cartográfica haciendo uso del Script anteriormente citado y una imagen que represente el plano que se quiere generar, realizándose todo de forma manual y mediante un terminal. Una vez realizado esto, se cuenta con el conjuntos de datos necesarios para poder mostrar en el sistema que se precise la información del mapa generado, lo cuál no permite ver su estado hasta que esa información se integra en el sistema en cuestión.

Además, estos mapas están asociados a una serie de proyectos que precisan de este tipo de datos para poder mostrar su información de forma completa, como es el caso del proyecto *Xtremeloc*, en el que la información que se muestra en el mapa es la localización de edificios que tienen instalado el sistema, las antenas desplegadas dentro de las plantas de esos edificios, etc...

Estos detalles asientan las bases del trabajo a realizar, que no es otro que la tarea de automatizar y facilitar todo el proceso citado en el párrafo anterior. Con el conjunto de la parte de lógica de servidor y la interfaz web, junto con la integración de todo ello con las herramientas que ya se encuentran en producción, se conseguirá poder gestionar los proyectos y mapas con los que se necesite contar, de forma sencilla y gráfica, pudiendo usar los mapas generados en aquellos sistemas que se precise, y pudiendo visualizar su estado mediante la interfaz web una vez renderizados, además de toda la información asociada a los proyectos con los que cuente el sistema.

1.4. Estructura de esta memoria

La memoria del presente Trabajo de Fin de Grado está formada por una serie de capítulos que tratan cada uno un tema específico, a fin de organizar todo el trabajo realizado:

1. Análisis de requisitos, en el que se extrae del problema planteado las necesidades del sistema y se asientan una serie de requisitos sobre los que se apoyará el trabajo a realizar.

2. Plan de proyecto, dedicado a elaborar la planificación de todo el proceso de desarrollo del proyecto, indicando las fases por las que pasará el mismo y los riesgos existentes en cada una de ellas.

3. Modelo de análisis, en el que se detallan los diferentes casos de uso y el modelo de domino extraídos de los requisitos.

4. Diseño, dedicado a detallar la arquitectura del sistema y sus diagramas de secuencia, así como los bocetos de las vistas que compondrán la interfaz del sistema.

5. Reingeniería del sistema anterior, en el que se indican los cambios realizados en el sistema existente antes de la implementación de las nuevas funciones.

6. Implementación, dedicado a detallar los procesos seguidos durante el desarrollo así como la descripción de lo desarrollado.

7. Plan de pruebas y evaluación, en el que se planifican y elaboran una serie de pruebas de todo el sistema con la finalidad de comprobar que todos los requisitos planteados se han cubierto satisfactoriamente.

8. Manual del programador, dedicado a describir todos los detalles necesarios para que un desarrollador pueda ampliar o modificar el sistema implementado fácilmente.

9. Manual de instalación, en el que se detallan todos los procesos a seguir para la instalación del sistema en un nuevo entorno.

10. Manual de usuario, dedicado a describir a un usuario todo lo necesario del sistema para que sea capaz de usarlo fácilmente.

11. Conclusiones y trabajo futuro, en el que se exponen las conclusiones extraídas tras finalizar el trabajo realizado y los posibles caminos de trabajo futuro del sistema desarrollado.

Capítulo 2

Análisis de requisitos

Este capítulo se dedica al análisis de requisitos de la aplicación. En él se detalla la lista de requisitos funcionales, no funcionales y de información. El objetivo de este capítulo es contar con información sobre el sistema a desarrollar que facilite el desarrollo de las fases venideras dentro del proyecto.

2.1. Definición de los actores

Debido a la naturaleza propia del sistema a desarrollar, el único actor existente en el sistema será el administrador del mismo.

2.2. Requisitos de usuario

Los requisitos proporcionados por el usuario son los siguientes:

- RU-01: Se deberá realizar un API REST que permita automatizar y llevar a cabo todas las tareas relacionadas con la gestión de planos de edificios.
- RU-02: Se deberá realizar una interfaz web que aproveche el API REST anteriormente mencionado.

2.3. Requisitos funcionales

Listado de las funciones que debe realizar la aplicación.

- RF-01: El sistema deberá permitir al usuario iniciar sesión.
- RF-02: El sistema deberá permitir al usuario cerrar sesión.
- RF-03: El sistema deberá permitir al usuario poder seleccionar una imagen de su equipo para procesarla.
- RF-04: El sistema deberá permitir al usuario nombrar el mapa que se va a procesar.
- RF-05: El sistema deberá mostrar al usuario una previsualización del procesamiento de la imagen elegida.
- RF-06: El sistema deberá mostrar un mensaje al realizar el guardado del procesado de la imagen.
- RF-07: El sistema deberá mostrar una URL con la localización del procesado de la imagen.
- RF-08: El sistema deberá permitir al usuario copiar fácilmente la URL al portapapeles.

- RF-09: El sistema deberá permitir al usuario dar de alta un proyecto nuevo.
- RF-10: El sistema deberá permitir al usuario eliminar un proyecto.
- RF-11: El sistema deberá permitir al usuario modificar un proyecto.
- RF-12: El sistema deberá permitir al usuario dar de alta un cliente nuevo.
- RF-13: El sistema deberá permitir al usuario eliminar un cliente.
- RF-14: El sistema deberá permitir al usuario modificar un cliente.
- RF-15: El sistema deberá permitir al usuario dar de alta un edificio nuevo.
- RF-16: El sistema deberá permitir al usuario eliminar un edificio.
- RF-17: El sistema deberá permitir al usuario modificar un edificio.
- RF-18: El sistema deberá permitir al usuario dar de alta una planta nueva para un edificio.
- RF-19: El sistema deberá permitir al usuario eliminar una planta de un edificio.
- RF-20: El sistema deberá permitir al usuario modificar una planta de un edificio.
- RF-21: El sistema deberá permitir al usuario visualizar un listado con todos los proyectos.
- RF-22: El sistema deberá permitir al usuario visualizar un listado de los clientes de cada proyecto.
- RF-23: El sistema deberá permitir al usuario visualizar un listado de los edificios de cada cliente.
- RF-24: El sistema deberá permitir al usuario visualizar un listado de las plantas de un edificio.

2.4. Requisitos no funcionales

Listado de las características referentes al funcionamiento del sistema.

- RNF-01: El procesamiento de la imagen se realizará a través de un script en Python.
- RNF-02: El front-end de la aplicación se llevará a cabo en el framework de JavaScript Vue.js.
- RNF-03: El back-end de la aplicación se llevará a cabo en el framework de Python Flask.
- RNF-04: El despliegue de la aplicación se llevará a cabo mediante Docker.
- RNF-05: El contenedor de despliegue compartirá un volumen con el sistema existente.
- RNF-06: La persistencia de datos se realizará mediante una base de datos SQLite.
- RNF-07: Se deberá actualizar el presente servidor de mapas indoor.
- RNF-08: Se deberá actualizar la presente generación de tiles, con el objetivo de hacerla más óptima.

2.5. Requisitos de información

Listado de los datos necesarios para el correcto funcionamiento de la aplicación.

- RI-01: Para el cliente se almacenará:
    1. Nombre
- RI-02: Para cada proyecto se almacenará:
    1. Nombre
- RI-03: Para cada edificio se almacenará:
    1. Nombre

- RI-04: Para cada planta se almacenará:
    1. Nombre
    2. URL

Tras haber definido todos los requisitos necesarios para la realización del proyecto, continuamos con el siguiente capítulo, que se centrará en elaborar un plan de proyecto adecuado para el trabajo a desarrollar.

Capítulo 3

Plan de proyecto

3.1. Resumen del proyecto

En este apartado se detallará el plan de proyecto seguido para llevar a cabo la realización del servicio planteado en este TFG.

3.1.1. Propósito, alcance y objetivos

El objetivo del proyecto es el desarrollo de un servicio completo de administración y gestión de proyectos que usen información cartográfica, permitiendo la generación de este tipo de información en función de lo que se precise en cada instante. Las características que se incluirán son las siguientes (todas disponibles para el administrador, dada la naturaleza del sistema): Añadir, visualizar, editar y eliminar proyectos, clientes, edificios y mapas (plantas de los edificios dentro del sistema). Las funciones serán accesibles en su totalidad a través de la interfaz web desarrollada para el sistema.

3.1.2. Definiciones y acrónimos

A continuación se listarán un conjunto de definiciones y acrónimos que serán usados a lo largo del presente documento.

- REST: REpresentational State Transfer, trasnferencia de estado representacional.
- CRUD : Create, Read, Update and Delete / Crear, Leer, Actualizar y Borrar.
- OSM: OpenStreetMaps.
- API: Applicaction Programming Interface, Interfaz de programación de aplicaciones. Capa de abstracción formada por un conjunto de subrutinas, funciones y procedimientos ofrecidos por cierta biblioteca, para ser usados por otro software.
- Front-end: Principalmente es el desarrollo de una interfaz web, de modo que el usuario pueda interactuar con los datos que se manejan en una aplicación de forma gráfica.
- Back-end: Parte del desarrollo que corresponde a la parte del servidor, en la que se manejan y procesan los datos de una aplicación y/o servicio.
- ORM: Object-Relational mapping, mapeo objeto-relacional. Técnica de programación usada para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional.

- Tile: Franja o Loseta. En el contexto que nos vamos a mover durante el presente documento, indica parte del procesado de una imagen (una parte cuadrada de dicha imagen).
- Framework: Entorno de trabajo que establece unas pautas a seguir para enfocar un tipo de problema particular que sirve como referencia para afrontar otros problemas nuevos de índole similar.
- UML: Unified Modeling Language, Lenguaje Unificado de Modelado, lenguaje de modelado para sistemas software.
- CU: Caso de Uso.

3.1.3. Método escogido para el proyecto

Para llevar a cabo el desarrollo del proyecto se ha tomado la decisión de emplear un desarrollo basado en cascada, ya que la naturaleza del mismo y los requisitos a los que está sujeto son claros y no dan lugar a mucha interpretación que pueda desembocar en posibles cambios.

Este tipo de desarrollo se caracteriza por separar y distinguir completamente cada fase de especificación y desarrollo del proyecto. En el caso de este proyecto, se constarán de 5 fases: análisis, diseño, implementación, pruebas y mantenimiento. Tal y cómo indica el desarrollo en cascada, siempre se realizará de forma completa una fase antes de comenzar con la siguiente.

3.1.4. Evolución del plan

El plan de desarrollo de proyecto debe detallar cada una de las fases seguidas por las que pasará el proceso de implementación del sistema, y además detallar como se llevará a cabo la gestión de toda su evolución. Para conseguir esto y obtener un plan completo, se debe contemplar al menos lo siguiente:

- Detalle de los objetivos del proceso.
- Estimación de costes materiales y temporales necesarios para el desarrollo.
- Especificación de las tareas a realizar y su evolución temporal, así cómo determinar las fases por las que irá pasando el proyecto.
- Recursos que serán necesarios para llevar a cabo satisfactoriamente cada una de las fases.
- Estudio de los posibles riesgos que pueden surgir y un plan de contingencia asociado para poder actuar ante ellos.

3.1.5. Ciclo de vida del proyecto

Cómo se ha mencionado anteriormente en este capítulo, el ciclo de vida del proyecto constará de cinco fases perfectamente diferenciadas entre sí, que necesitan unos elementos de entrada y generan otros de salida. Para comenzar un fase será totalmente imprescindible haber terminado la anterior para poder disponer de sus artefactos de salida y así, poder usarlos como entrada para la fase actual. A continuación se detallan los objetivos a cumplir en cada una de las fases del proyecto:

- Fase de análisis: Se establece el contexto y alcance del proyecto, sus casos esenciales y la estimación temporal.
- Fase de diseño: Análisis del dominio del problema, elección de una arquitectura base adecuada y desarrollo de un plan de proyecto adecuado para la naturaleza del trabajo a realizar.

- Fase de implementación: Minimización de costes en la medida de lo posible y obtención de una calidad aceptable en el producto final.
- Fase de pruebas: Comprobar que el producto obtenido cumple con todos los requisitos establecidos de forma satisfactoria.
- Fase de mantenimiento: Conseguir que el usuario sea capaz de usar el producto y alcanzar un resultado final que sea rápido, eficiente y práctico, de modo que sea aceptado por el cliente, a base de refinar, si fuera necesario, dicho producto.

3.2. Gestión del proyecto

3.2.1. Plan de puesta en marcha

Para llevar a cabo el desarrollo del proyecto de forma adecuada se debe realizar una estimación de cada una de las fases, teniendo en cuenta la experiencia previa obtenida de proyectos similares. El desarrollo se realizará de forma individual, por lo que la gestión de personal no será necesaria para este contexto.

Antes de comenzar con el desarrollo del proyecto, será necesario tener cierto conocimiento de algunos aspectos clave para su elaboración. Se dividirá según los apartados que se van a desarrollar dentro del mismo, para tener una vista más clara de lo necesario para la resolución del proyecto:

Base de datos

- Conocimiento sobre bases de datos relacionales, tanto en su uso como en llevar a cabo un diseño de la misma.
- Conocimiento sobre SQLite3.

Back-end

- Conocimiento sobre Python 3.7.
- Conocimiento sobre el framework Flask.
- Conocimiento sobre el ORM SQLAlchemy.
- Conocimiento sobre el uso y procesado de OSM.
- Conocimiento sobre el procesado de imágenes para la generación de Tiles mediante pygdal y gdal2tiles.

Front-end

- Conocimiento sobre HTML, CSS y JavaScript.
- Conocimiento sobre el framework Vue.js.

Despliegue

- Conocimiento sobre Docker, así como todos los conceptos que engloba, necesarios para el despliegue correcto de un sistema.

En líneas más generales, conocimiento del diseño empleado en el resto de proyectos para que este cuente con una continuidad.

3.2.2. Plan de trabajo

En este apartado se detalla cada una de las fases del proyectos y las actividades que las componen.

Fase de análisis

ID: 01	**Inicio de la fase de Análisis**
Predecesoras:	-
Duración:	-
-	

Tabla 3.1: Actividad 01

ID: 02	**Elicitación de requisitos**
Predecesoras:	01
Duración:	3 días
Obtención de los 3 tipos de requisitos (funcionales, no funcionales y de información	

Tabla 3.2: Actividad 02

ID: 03	**Definición de Actividades**
Predecesoras:	02
Duración:	2 días
Definición de las actividades necesarias para el desarrollo del proyecto y su duración	

Tabla 3.3: Actividad 03

ID: 04	**Análisis de riesgos**
Predecesoras:	03
Duración:	2 días
Análisis de los riesgo que presenta el desarrollo junto con el plan de contención de cada uno de ellos	

Tabla 3.4: Actividad 04

ID: 05	**Preparación del calendario**
Predecesoras:	04
Duración:	1 día
Elaboración de una planificación de carácter temporal para la ejecución de todas las fases del proyecto	

Tabla 3.5: Actividad 05

ID: 06	**Aprendizaje de Vue.js**
Predecesoras:	05
Duración:	7 días
Familiarización con el framework que se va a usar para el desarrollo del front-end, tanto en su uso como en su arquitectura	

Tabla 3.6: Actividad 06

ID: 07	Aprendizaje de Flask
Predecesoras:	05
Duración:	7 días
Familiarización con el framework que se va a usar para el desarrollo del back-end, tanto en su uso como en su arquitectura	

Tabla 3.7: Actividad 07

ID: 08	Aprendizaje de Docker
Predecesoras:	05
Duración:	7 días
Familiarización con la tecnología Docker, que será usada para el despliegue de todo el sistema desarrollado	

Tabla 3.8: Actividad 08

ID: 09	Familiarización con OSM
Predecesoras:	05
Duración:	7 días
Estudio de la tecnología de OpenStreetMaps, así como de su procesado, despliegue y uso	

Tabla 3.9: Actividad 09

ID: 10	Desarrollo del plan de proyecto
Predecesoras:	06, 07, 08, 09
Duración:	2 días
Elaboración del documento que describe el plan de proyecto, incluyendo todas las actividades y fases, así como su planificación y la gestión de los riesgos asociados	

Tabla 3.10: Actividad 10

ID: 11	Fin de la fase de Análisis
Predecesoras:	10
Duración:	-
-	

Tabla 3.11: Actividad 11

Fase de Diseño

ID: 12	Inicio de la fase de Diseño
Predecesoras:	11
Duración:	-
-	

Tabla 3.12: Actividad 12

ID: 13	Elaboración de los casos de uso
Predecesoras:	12
Duración:	2 días
Elaboración del diagrama de casos de uso del sistema así como la especificación de cada uno de ellos, partiendo de los requisitos encontrados en la fase de Análisis	

Tabla 3.13: Actividad 13

ID: 14	Elaboración del modelo de dominio
Predecesoras:	12
Duración:	2 días
Elaboración del diagrama de clases correspondiente a las entidades que serán usadas en el sistema, únicamente incluyendo los atributos de éstas	

Tabla 3.14: Actividad 14

ID: 15	Elaboración del despliegue del sistema
Predecesoras:	12
Duración:	4 días
Elaboración del diagrama de despliegue del sistema, junto con las dependencias de cada apartado del conjunto	

Tabla 3.15: Actividad 15

ID: 16	Diseño de la arquitectura del sistema
Predecesoras:	13, 14, 15
Duración:	6 días
Elaboración de un diagrama que permita decidir la arquitectura que seguirá el sistema	

Tabla 3.16: Actividad 16

ID: 17	Diseño de los bocetos del sistema
Predecesoras:	16
Duración:	2 días
Elaboración de los bocetos que describan lo mejor posible las vistas que se van a implementar en el sistema, en base a los casos de uso obtenidos a través del diagrama de casos de uso	

Tabla 3.17: Actividad 17

ID: 18	Elaboración de diagramas de secuencia
Predecesoras:	16
Duración:	6 días
Elaboración de los diagramas de secuencia que detallen la realización de los casos de uso anteriormente planteados, tanto para el front-end como para el back-end	

Tabla 3.18: Actividad 18

ID: 19	Fin de la fase de Diseño
Predecesoras:	17, 18
Duración:	-

Tabla 3.19: Actividad 19

Fase de Implementación

ID: 20	Inicio de la fase de Implementación
Predecesoras:	19
Duración:	-
-	

Tabla 3.20: Actividad 20

ID: 21	Preparación del entorno de desarrollo
Predecesoras:	20
Duración:	3 días
Se instalarán todas las dependencias necesarias para poder llevar a cabo el desarrollo del back-end, el front-end y el despliegue de todo el sistema	

Tabla 3.21: Actividad 21

ID: 22	Reingeniería del sistema actual
Predecesoras:	21
Duración:	20 días
Se realizará un reingeniería del sistema actual, para ver las características con las que se cuentan y corrigiendo posibles fallos existentes, así la realización de una adaptación para la instauración del nuevo servicio	

Tabla 3.22: Actividad 22

ID: 23	Desarrollo del back-end
Predecesoras:	22
Duración:	30 días
Se llevará a cabo la implementación del nuevo servicio en base al análisis realizado del sistema existente, para la inclusión de nuevas opciones y la mejora de las ya existentes	

Tabla 3.23: Actividad 23

ID: 24	Desarrollo del front-end
Predecesoras:	22, 23
Duración:	30 días
Se llevará a cabo la implementación de una interfaz web para hacer uso del servicio implementado	

Tabla 3.24: Actividad 24

ID: 25	Fin de la fase de Implementación
Predecesoras:	24
Duración:	-
-	

Tabla 3.25: Actividad 25

Fase de Pruebas

ID: 26	Inicio de la fase de Pruebas
Predecesoras:	25
Duración:	-
-	

Tabla 3.26: Actividad 26

ID: 27	Realización de pruebas del sistema
Predecesoras:	26
Duración:	4 días
Realización de todas las pruebas necesarias para comprobar que el sistema cubre satisfactoriamente todos los requisitos establecidos	

Tabla 3.27: Actividad 27

ID: 28	Fin de la fase de Pruebas
Predecesoras:	27
Duración:	-
-	

Tabla 3.28: Actividad 28

Fase de Mantenimiento

ID: 29	Inicio de la fase de Mantenimiento
Predecesoras:	28
Duración:	-
-	

Tabla 3.29: Actividad 29

ID: 30	Despliegue del sistema
Predecesoras:	29
Duración:	4 días
Despliegue en producción del conjunto desarrollado	

Tabla 3.30: Actividad 30

ID: 31	Elaboración del manual de usuario
Predecesoras:	30
Duración:	2 días
Elaboración de un documento que indique de forma concreta y clara el correcto uso del sistema.	

Tabla 3.31: Actividad 31

ID: 32	Elaboración del manual de instalación
Predecesoras:	31
Duración:	2 días
Elaboración de un documento que indique de forma concreta y clara los pasos a seguir y los elementos necesarios para el despliegue del sistema por parte de un administrador.	

Tabla 3.32: Actividad 32

ID: 33	Elaboración del manual del programador
Predecesoras:	32
Duración:	1 día
Elaboración de un documento que incluya toda la información necesaria para que otro desarrollador pueda continuar con el desarrollo del sistema	

Tabla 3.33: Actividad 33

ID: 34	Fin de la fase de Mantenimiento
Predecesoras:	33
Duración:	-
-	

Tabla 3.34: Actividad 34

Diagramas de Gantt

En este apartado se mostrarán los Diagramas de Gantt para cada una de las fases por las que ha pasado el proyecto.

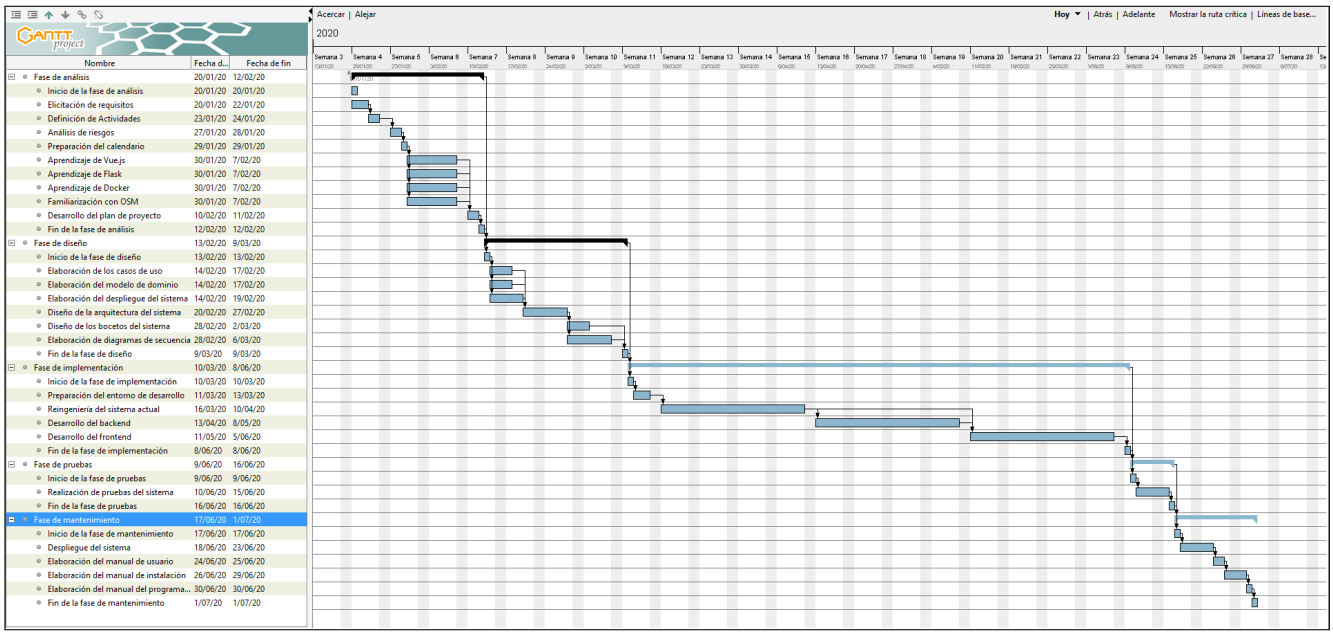

Figura 3.1: Vista general del plan de proyecto

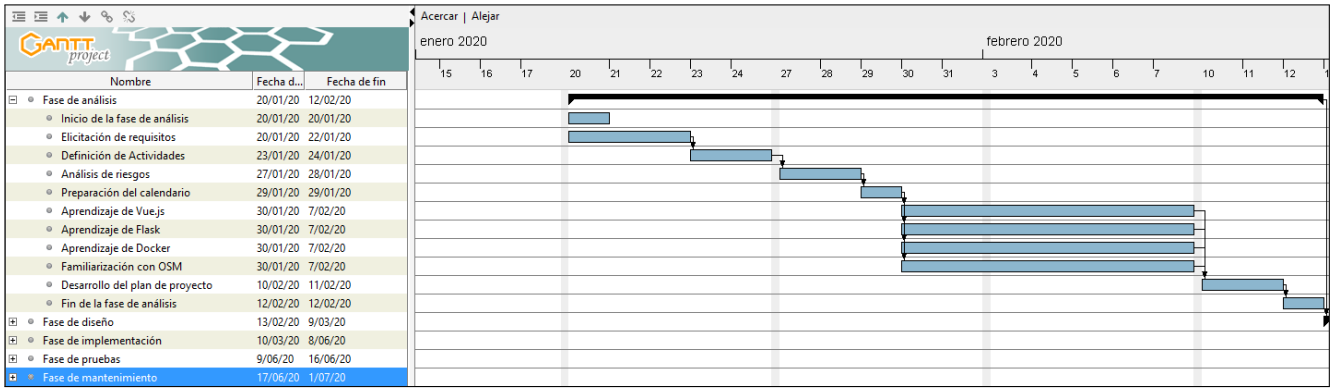

Figura 3.2: Fase de análisis

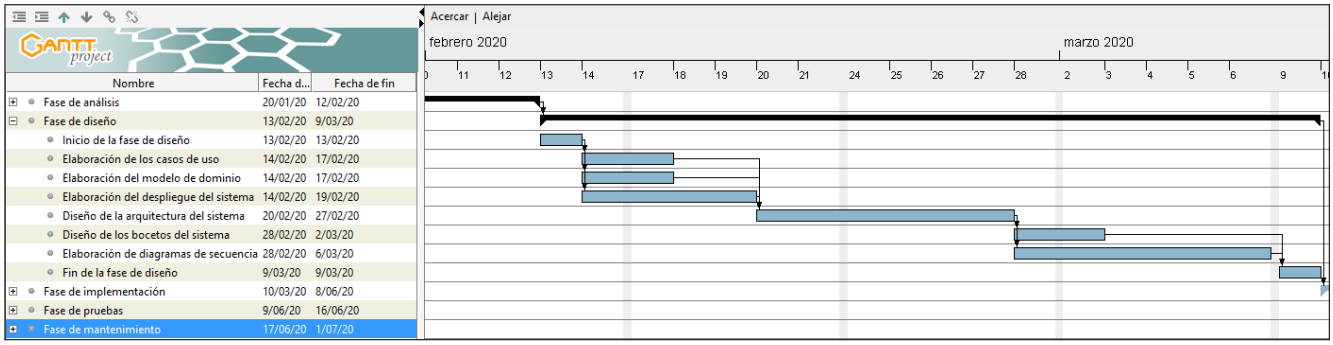

Figura 3.3: Fase de diseño

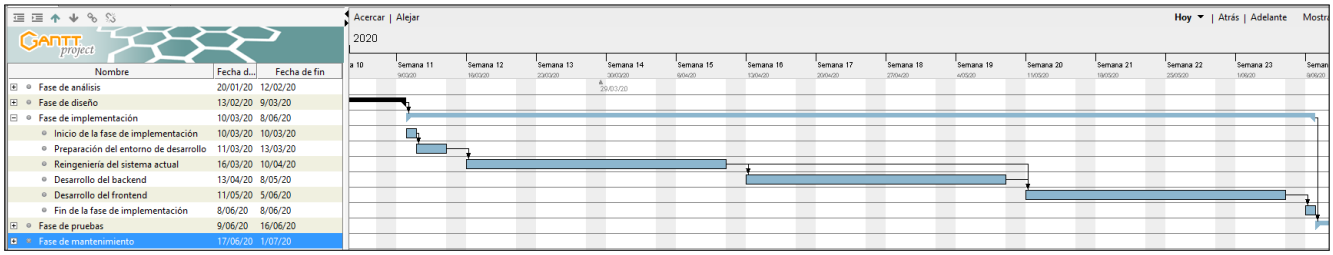

Figura 3.4: Fase de implementación

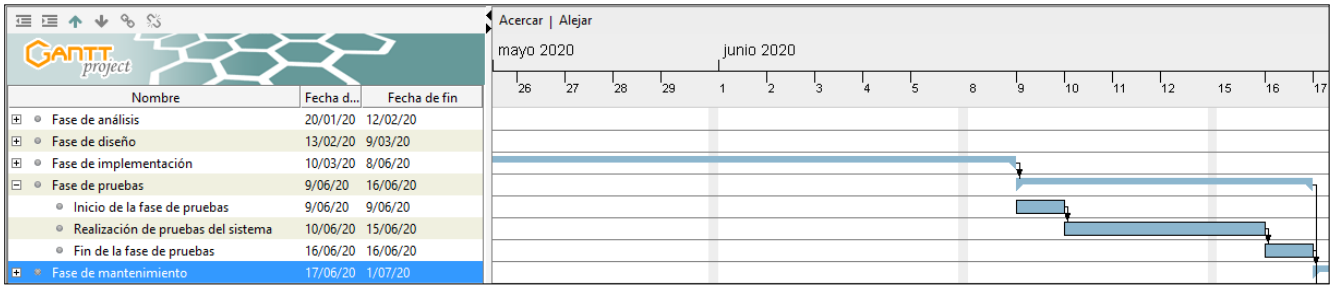

Figura 3.5: Fase de pruebas

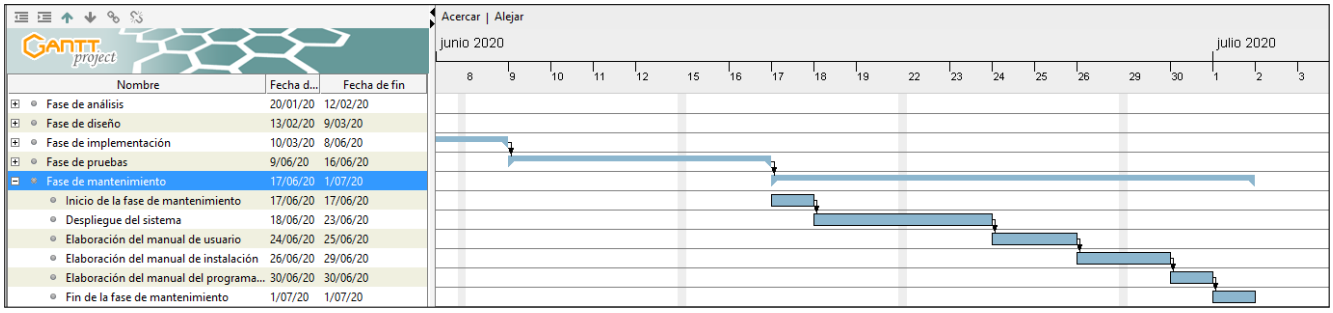

Figura 3.6: Fase de mantenimiento

3.2.3. Plan de control

Para asegurar un funcionamiento óptimo de todo el proceso de desarrollo, se realizará un plan de control basado tanto en los objetivos iniciales como en el planteamiento propuesto para su realización, de forma que puedan adaptarse en función de las necesidades y el contexto de cada fase del desarrollo.

Para poder llevar a cabo esta tarea, se elaborará un listado con los requisitos que previamente han sido analizados, ordenándolos por su prioridad. De esta forma, si en algún momento es necesario modificar alguno de los requisitos existentes o añadir alguno nuevo, podrá hacerse de manera sencilla mediante esta lista, teniendo en cuenta la prioridad y realizando una replanificación temporal si fuere necesario.

En cuanto a la planificación y control temporal de todo el proceso, al final de cada actividad planteada se ha decidido que se realizarán tanto un análisis como un seguimiento de la actividad finalizada, para poder comprobar que el marco temporal se adapta correctamente a la planificación realizada o que por el contrario, no se adapta al planteamiento propuesto y es necesario adaptarlo. Este método permitirá observar y detectar posibles contratiempos durante el desarrollo del proyecto, de modo que el proceso pueda dividirse y de ese modo admita correcciones en fragmentos de tiempo reducidos.

3.2.4. Plan de gestión de riesgos

R01 - Desconocimiento del sistema actual

Falta de conocimiento del sistema que se usa actualmente, provocando un retraso temporal en el desarrollo del proyecto.

Impacto Crítico

Probabilidad Alta

Exposición Media

Plan de Protección Reservar un tiempo en la fase de análisis del sistema para poder comprender totalmente su funcionamiento.

Plan de Contingencia En caso de producirse algún retraso temporal, adaptar la planificación realizada según se necesite.

R02 - Desconocimiento de las herramientas a utilizar para el desarrollo

Falta de conocimiento de los frameworks que van a ser usados para el desarrollo de la parte del servidor y de la aplicación web.

Impacto Crítico

Probabilidad Alta

Exposición Media

Plan de Protección Reservar un tiempo en la fase de análisis para poder realizar un aprendizaje previo de las herramientas.

Plan de Contingencia En caso de producirse algún retraso temporal, adaptar la planificación realizada según se necesite.

R03 - Desconocimiento de las herramientas a utilizar para el despliegue

Falta de conocimiento de las herramientas que van a ser usadas para el despliegue del sistema desarrollado.

Impacto Crítico

Probabilidad Media

Exposición Media

Plan de Protección Reservar un tiempo en la fase de análisis para poder realizar un aprendizaje previo de las herramientas.

Plan de Contingencia En caso de producirse algún retraso temporal, adaptar la planificación realizada según se necesite.

R04 - Fallos de despliegue

Fallos en el despliegue del sistema desarrollado.

Impacto Crítico

Probabilidad Media

Exposición Media

Plan de Protección Realizar un correcto diseño de despliegue del sistema que aproveche correctamente las herramientas con las que se cuenta, así como un plan de pruebas que contemple los posibles casos de fallo.

Plan de Contingencia En caso de producirse algún retraso temporal, adaptar la planificación realizada según se necesite y modificar los elementos del sistema que produzcan errores.

R05 - Carencia de alguna funcionalidad

Falta de alguna funcionalidad que debería haber sido cubierta.

Impacto Crítico

Probabilidad Baja

Exposición Alta

Plan de Protección Ajustarse a los requisitos definidos en la fase de análisis.

Plan de Contingencia Añadir la nueva funcionalidad que se precise.

R06 - Retraso en alguna de las fases

Retraso temporal de alguna de las fases planificadas anteriormente.

Impacto Crítico

Probabilidad Alta

Exposición Media

Plan de Protección Ajustarse a la planificación planteada lo máximo posible.

Plan de Contingencia En caso de producirse algún retraso temporal, adaptar la planificación realizada según se necesite.

R07 - Diseño erróneo

El diseño planteado no cumple con los requisitos esperados para el sistema desarrollado.

Impacto Crítico

Probabilidad Baja

Exposición Baja

Plan de Protección Llevar a cabo el diseño ajustándose lo máximo posible a la información obtenida en la fase de análisis.

Plan de Contingencia En caso de producirse algún retraso temporal, adaptar la planificación realizada según se necesite.

Durante el desarrollo del proyecto y tras haber llevado a cabo la realización del plan de proyecto tuvieron que incluirse 2 elementos a mayores, dada la crisis sanitaria surgida por el virus denominado como *covid-19* y tras el decreto del estado de alarma en todo el estado español el pasado 14 de Marzo de 2020.

R08 - Enfermedad

Dadas las circunstancias en las que se ha desarrollado el proyecto y que la elaboración del mismo cuenta con un sólo programador, se ha debido tener en cuenta esta posibilidad, ya que imposibilitaría la continuación del proyecto.

Impacto Leve

Probabilidad Baja

Exposición Alta

Plan de Protección No se puede elaborar un plan de contención para este caso.

Plan de Contingencia Estudiar y replanificar el plan de proyecto en función de los días de incapacidad que hayan causado pérdida de tiempo en el desarrollo del proyecto.

R09 - Cuarentena

Dadas las circunstancias en las que se ha desarrollado el proyecto, y en relación con el anterior punto, se debe tener en cuenta la posibilidad de tener que realizar una cuarentena por parte del desarrollador del proyecto.

Impacto Crítico

Probabilidad Baja

Exposición Alta

Plan de Protección No se puede elaborar un plan de contención para este caso.

Plan de Contingencia Preparar todo el material de trabajo en el equipo portátil del trabajador, y si fuere necesario, replanificar el plan de proyecto.

3.2.5. Seguimiento del proyecto

Teniendo en cuenta todo lo citado anteriormente, no ha surgido ninguna situación excepcional que haya supuesto un retraso significativo en el desarrollo planteado del proyecto, por lo que se ha podido cumplir con las fechas planteadas en el inicio del mismo.

3.3. Presupuesto

Teniendo en cuenta la planificación anteriormente planteada, la duración del proyecto será de 6 meses. A continuación se mostrará una tabla que contará con la información necesaria para ver el presupuesto total (simulado) del proyecto.

Presupuesto del Proyecto	
Duración del proyecto	6 meses.
Salario según convenio de un Ingeniero Informático en España	25.600€.
Costes sociales	8.450€.
Total	14.593'2€.

Tabla 3.35: Presupuesto (IVA no incluido)

Dada la naturaleza del proyecto, se debe indicar que el presupuesto real del mismo ha sido bastante menor que la anteriormente detallada, pues el contexto en el que se ha llevado a cabo ha sido totalmente académico aunque sostenido por una beca de investigación.

Tras finalizar el plan de desarrollo del proyecto, nos centraremos en el análisis del problema planteado con más profundidad en el capítulo siguiente.

Capítulo 4

Modelo de análisis

Este apartado detalla el análisis de la aplicación, partiendo de la información recogida en la elicitación de requisitos.

4.1. Casos de uso del sistema

Esta sección está centrada en el estudio de los casos de uso del sistema, tanto generales cómo específicos. Los casos de uso permiten describir de forma concreta qué es lo que hace el sistema, así como la interacción que el **usuario** (al que a partir de ahora denominaremos **actor**), tendrá con el sistema al utilizar cada una de las características con las que cuenta. Los casos de uso serán de gran utilidad como guía al llevar el desarrollo del sistema, así como apoyo para la realización del plan de pruebas(8) del mismo.

4.1.1. Diagrama general de casos de uso

Este diagrama es el primer paso para hacerse una idea de todas las interacciones que los actores van a tener con el sistema. Más adelante entraremos en detalle con cada uno de ellos.

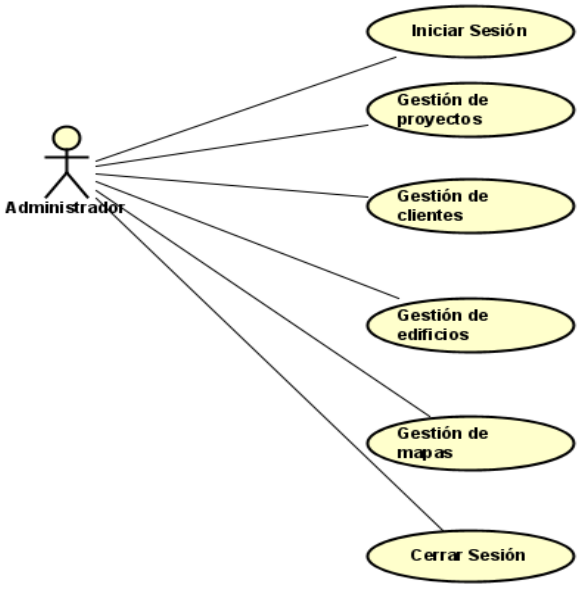

Figura 4.1: Diagrama general de casos de uso

4.1.2. Caso de uso 1: Iniciar sesión

CU-1.1	**Iniciar sesión**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador iniciar sesión.
Precondición	
Secuencia Normal	
Paso	**Acción**
1	El actor introduce las credenciales.
2	El sistema comprueba que las credenciales son correctas.
3	El actor inicia sesión correctamente en el sistema. El caso de uso termina.
Post-condición	El actor consigue entrar en el sistema.
Excepciones	
Variación	**Acción**
1b, 2b, 3b	El actor sale del sistema y el caso de uso queda sin efecto.
2c	El sistema determina que las credenciales no son correctas, se lo notifica al usuario y el caso de uso continúa en el paso 1.

Tabla 4.1: Caso de Uso 1.1 - Iniciar sesión

4.1.3. Caso de uso 2: Gestión de proyectos

Los casos de uso de la gestión de proyectos permiten al usuario ver, añadir, editar y eliminar proyectos.

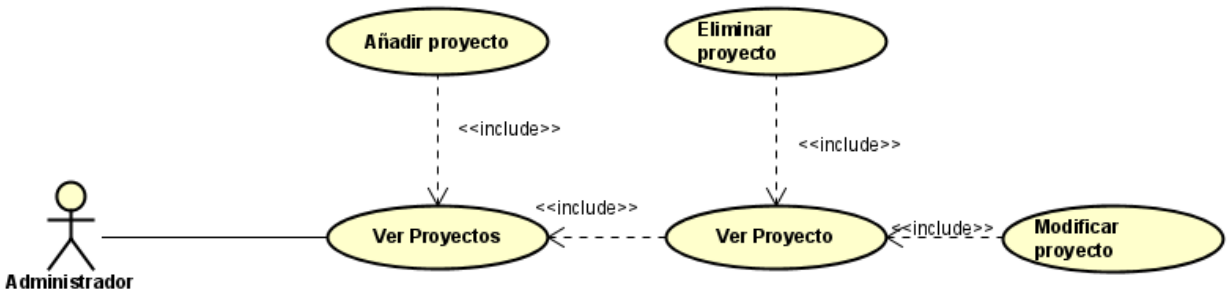

Figura 4.2: Diagrama en detalle de gestión de proyectos

CU-2.1	**Ver proyectos**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador visualizar un listado de todos los proyectos.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El actor entra en la página principal.
2	El sistema muestra un listado con los proyectos existentes. El caso de uso termina.
Post-condición	
Excepciones	
Variación	**Acción**
1b, 2b	El actor sale del sistema. El caso de uso queda sin efecto.
2c	El sistema no contiene proyectos que mostrar, notificándoselo al actor. El caso de uso termina.

Tabla 4.2: Caso de Uso 2.1 - Ver proyectos

CU-2.2	**Añadir proyecto**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador añadir proyectos.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver proyectos*.
2	El actor selecciona *AÑADIR PROYECTO*.
3	El sistema muestra los campos a rellenar.
4	El actor completa la información solicitada y pulsa el botón de *GUARDAR*.
5	El sistema comprueba que los datos son correctos.
6	El sistema muestra un mensaje de éxito. El caso de uso termina.
Post-condición	El proyecto se añade al sistema.
Excepciones	
Variación	**Acción**
2b, 3b, 4b	El actor sale del sistema y el caso de uso queda sin efecto.
4c	El actor selecciona la opción *CANCELAR* y el caso de uso queda sin efecto.
5b	El sistema detecta que los datos no son correctos, muestra un mensaje de error y el caso de uso continúa en el paso 3.

Tabla 4.3: Caso de Uso 2.2 - Añadir proyecto

CU-2.3	**Ver proyecto**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador visualizar el detalle de un proyecto del listado.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver proyectos*
2	El actor selecciona un proyecto de la lista.
3	El sistema carga los datos del proyecto seleccionado. El caso de uso termina.
Post-condición	
Excepciones	
Variación	**Acción**
2b, 3b	El actor sale del sistema. El caso de uso queda sin efecto.

Tabla 4.4: Caso de Uso 2.3 - Ver proyecto

CU-2.4	**Modificar proyecto**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador eliminar proyectos ya existentes.
Precondición	El actor ha iniciado sesión
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver proyecto.*
2	El actor selecciona *Acciones.*
3	El actor selecciona *Editar proyecto.*
4	El sistema muestra los datos actuales del proyecto para modificarlos.
5	El actor modifica los datos y pulsa el botón *GUARDAR.*
6	El sistema comprueba que los datos con correctos.
7	El sistema muestra un mensaje de éxito. El caso de uso termina.
Post-condición	Los datos del proyecto se modifican.
Excepciones	
Variación	**Acción**
2b, 3b, 4b, 5b, 6b	El actor sale del sistema y el caso de uso queda sin efecto.
4c, 5c	El actor selecciona la opción cancelar y el caso de uso queda sin efecto.
6c	El sistema detecta que los datos no son correctos, muestra un mensaje de error y el caso de uso queda sin efecto.

Tabla 4.5: Caso de Uso 2.4 - Modificar proyecto

CU-2.5	**Eliminar proyecto**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador eliminar proyectos ya existentes.
Precondición	El actor ha iniciado sesión
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver proyecto*.
2	El actor selecciona *Acciones*.
3	El actor selecciona *Eliminar proyecto*.
4	El sistema muestra un mensaje informando de lo que se va a realizar y solicita confirmación.
5	El actor pulsa el botón *Aceptar*.
6	El sistema elimina el proyecto. El caso de uso termina.
Post-condición	El proyecto se elimina del sistema.
Excepciones	
Variación	**Acción**
2b, 3b, 4b	El actor sale del sistema y el caso de uso queda sin efecto.
5b	El actor pulsa el botón de cancelar y el caso de uso queda sin efecto.

Tabla 4.6: Caso de Uso 2.5 - Eliminar proyecto

4.1.4. Caso de uso 3: Gestión de clientes

Los casos de uso de la gestión de clientes permiten al usuario ver, añadir, editar y eliminar clientes.

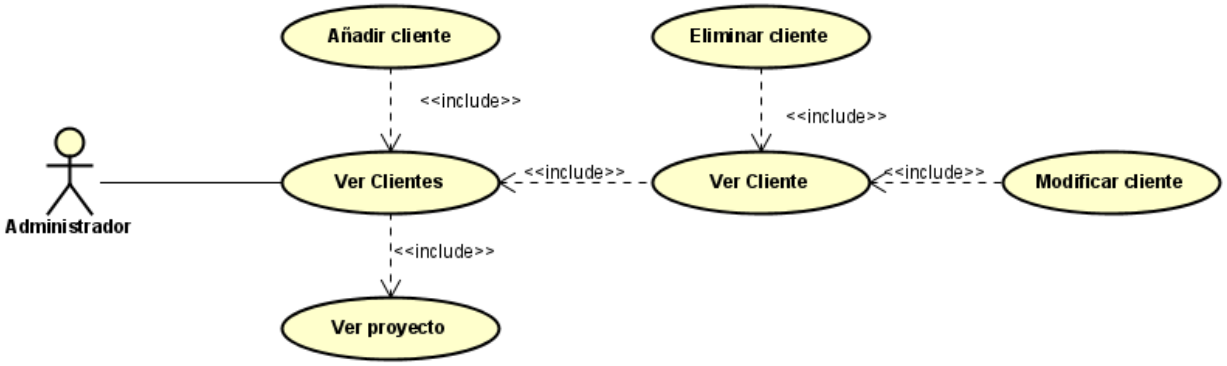

Figura 4.3: Diagrama en detalle de gestión de cliente

CU-3.1	**Ver clientes**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador visualizar un listado de todos los clientes.
Precondición	El actor ha iniciado sesión
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver proyecto*
2	El actor selecciona *Clientes*.
3	El actor selecciona un *Ver clientes*.
4	El sistema muestra un listado con los clientes existentes. El caso de uso termina.
Post-condición	
Excepciones	
Variación	**Acción**
2b, 3b	El actor sale del sistema y el caso de uso queda sin efecto.
4b	El sistema no contiene clientes asociados al proyecto que mostrar, notificándoselo al actor. El caso de uso termina.

Tabla 4.7: Caso de Uso 3.1 - Ver clientes

CU-3.2	**Añadir cliente**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador añadir clientes.
Precondición	El actor ha iniciado sesión
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver clientes*.
2	El actor selecciona *AÑDIR CLIENTE*.
3	El sistema muestra los campos a rellenar.
4	El actor completa la información solicitada y pulsa el botón de *GUARDAR*.
5	El sistema comprueba que los datos son correctos.
6	El sistema muestra un mensaje de éxito. El caso de uso termina.
Post-condición	El cliente se añade al sistema.
Excepciones	
Variación	**Acción**
2b,3b, 4b	El actor sale del sistema y el caso de uso queda sin efecto.
4c	El actor selecciona la opción *CANCELAR* y el caso de uso queda sin efecto.
5b	El sistema detecta que los datos no son correctos, muestra un mensaje de error y el caso de uso continúa en el paso 3.

Tabla 4.8: Caso de Uso 3.2 - Añadir cliente

CU-3.3	**Ver cliente**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador visualizar el detalle de un cliente del listado.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver clientes*
2	El actor selecciona un cliente de la lista.
3	El sistema carga los datos del cliente seleccionado. El caso de uso termina.
Post-condición	
Excepciones	
Variación	**Acción**
2b, 3b	El actor sale del sistema. El caso de uso queda sin efecto.

Tabla 4.9: Caso de Uso 3.3 - Ver cliente

CU-3.4	**Modificar cliente**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador modificar clientes ya existentes.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver cliente*.
2	El actor selecciona *Acciones*.
3	El actor selecciona *Editar cliente*.
4	El sistema muestra los datos actuales del cliente para modificarlos.
5	El actor modifica los datos y pulsa el botón *GUARDAR*.
6	El sistema comprueba que los datos con correctos.
7	El sistema muestra un mensaje de éxito. El caso de uso termina.
Post-condición	Los datos del cliente se modifican.
Excepciones	
Variación	**Acción**
2b, 3b, 4b, 5b, 6b	El actor sale del sistema y el caso de uso queda sin efecto.
4c, 5c	El actor selecciona la opción cancelar y el caso de uso queda sin efecto.
6c	El sistema detecta que los datos no son correctos, muestra un mensaje de error y el caso de uso queda sin efecto.

Tabla 4.10: Caso de Uso 3.4 - Modificar cliente

CU-3.5	**Eliminar cliente**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador eliminar clientes ya existentes.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver cliente*.
2	El actor selecciona *Acciones*.
3	El actor selecciona *Eliminar cliente*.
4	El sistema muestra un mensaje informando de lo que se va a realizar y solicita confirmación.
5	El actor pulsa *Aceptar*.
6	El sistema elimina el cliente. El caso de uso termina.
Post-condición	El cliente se elimina del sistema.
Excepciones	
Variación	**Acción**
2b, 3b, 4b	El actor sale del sistema y el caso de uso queda sin efecto.
5b	El actor pulsa el botón de cancelar y el caso de uso queda sin efecto.

Tabla 4.11: Caso de Uso 3.5 - Eliminar cliente

4.1.5. Caso de uso 4: Gestión de edificios

Los casos de uso de la gestión de edificios permiten al usuario ver, añadir, editar y eliminar edificios.

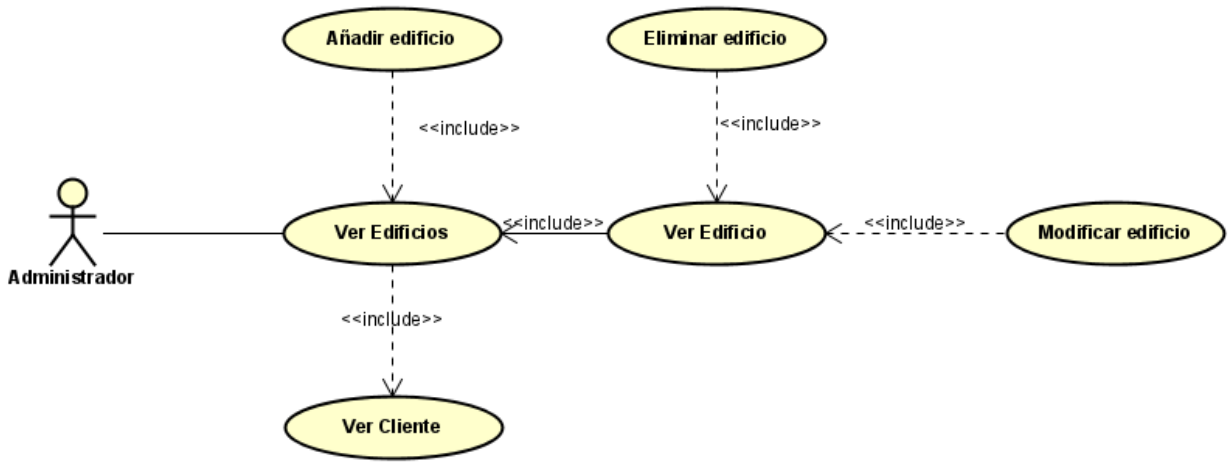

Figura 4.4: Diagrama en detalle de gestión de edificios

CU-4.1	**Ver edificios**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador visualizar un listado de todos los edificios.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver cliente*.
2	El actor selecciona *Edificios*.
3	El actor selecciona *Ver edificios*.
4	El sistema muestra un listado con los edificios existentes. El caso de uso termina.
Post-condición	
Excepciones	
Variación	**Acción**
2b, 3b	El actor sale del sistema y el caso de uso queda sin efecto.
4b	El sistema no contiene edificios asociados al cliente que mostrar, notificándoselo al actor. El caso de uso termina.

Tabla 4.12: Caso de Uso 4.1 - Ver edificios

CU-4.2	**Añadir edificio**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador añadir edificios.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver edificios*.
2	El actor selecciona *AÑADIR EDIFICIO*.
3	El sistema muestra los campos a rellenar.
4	El actor completa la información solicitada y pulsa el botón de *GUARDAR*.
5	El sistema comprueba que los datos son correctos.
6	El sistema muestra un mensaje de éxito. El caso de uso termina.
Post-condición	El edificio se añade al sistema.
Excepciones	
Variación	**Acción**
2b, 3b, 4b	El actor sale del sistema y el caso de uso queda sin efecto.
4c	El actor selecciona la opción *CANCELAR* y el caso de uso queda sin efecto.
5b	El sistema detecta que los datos no son correctos, muestra un mensaje de error y el caso de uso continúa en el paso 3.

Tabla 4.13: Caso de Uso 4.2 - Añadir edificio

CU-4.3	**Ver edificio**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador visualizar el detalle de un edificio del listado.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver edificios*
2	El actor selecciona un edificio de la lista.
3	El sistema carga los datos del edificio seleccionado. El caso de uso termina.
Post-condición	
Excepciones	
Variación	**Acción**
2b, 3b	El actor sale del sistema. El caso de uso queda sin efecto.

Tabla 4.14: Caso de Uso 4.3 - Ver edificio

CU-4.4	**Modificar edificio**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador modificar edificios ya existentes.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver edificio*.
2	El actor selecciona *Acciones*.
3	El actor selecciona *Editar edificio*.
4	El sistema muestra los datos actuales del edificio para modificarlos.
5	El actor modifica los datos y pulsa el botón *GUARDAR*.
6	El sistema comprueba que los datos con correctos.
7	El sistema muestra un mensaje de éxito. El caso de uso termina.
Post-condición	Los datos del edificio se modifican.
Excepciones	
Variación	**Acción**
2b, 3b, 4b, 5b, 6b	El actor sale del sistema y el caso de uso queda sin efecto.
4c, 5c	El actor selecciona la opción cancelar y el caso de uso queda sin efecto.
6c	El sistema detecta que los datos no son correctos, muestra un mensaje de error y el caso de uso queda sin efecto.

Tabla 4.15: Caso de Uso 4.4 - Modificar edificio

CU-4.5	**Eliminar edificio**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador eliminar edificios ya existentes.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver edificio*.
2	El actor selecciona *Acciones*.
3	El actor selecciona *Eliminar edificio*.
4	El sistema muestra un mensaje informando de lo que se va a realizar y solicita confirmación.
5	El actor pulsa *Aceptar*.
6	El sistema elimina el edificio. El caso de uso termina.
Post-condición	El edificio se elimina del sistema.
Excepciones	
Variación	**Acción**
2b, 3b, 4b	El actor sale del sistema y el caso de uso queda sin efecto.
5b	El actor pulsa el botón de cancelar y el caso de uso queda sin efecto.

Tabla 4.16: Caso de Uso 4.5 - Eliminar edificio

4.1.6. Caso de uso 5: Gestión de mapas

Los casos de uso de la gestión de mapas permiten al usuario ver, añadir, editar y eliminar mapas. Los mapas están asociados a plantas, se ha elegido llamar así a este caso de uso porque es el eje principal del proyecto desarrollado.

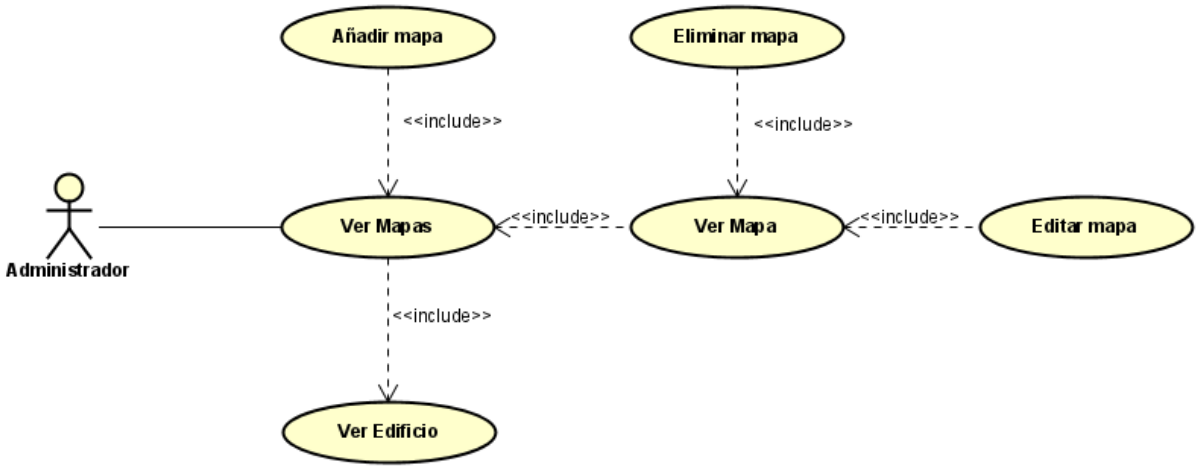

Figura 4.5: Diagrama en detalle de gestión de mapas

CU-5.1	**Ver mapas**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador visualizar un listado de todos los mapas.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver edificio*.
2	El actor selecciona *Plantas*.
3	El actor selecciona *Ver plantas*.
4	El sistema muestra un listado con los mapas existentes. El caso de uso termina.
Post-condición	
Excepciones	
Variación	**Acción**
2b, 3b	El actor sale del sistema y el caso de uso queda sin efecto.
4b	El sistema no contiene mapas asociados al edifico que mostrar, notificándoselo al actor. El caso de uso termina.

Tabla 4.17: Caso de Uso 5.1 - Ver mapas

CU-5.2	**Añadir mapa**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador añadir mapas.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver mapas*.
2	El actor selecciona *AÑADIR PLANTA*.
3	El sistema muestra los campos a rellenar.
4	El actor completa la información solicitada y pulsa el botón de *GUARDAR*.
5	El sistema comprueba que los datos son correctos.
6	El sistema muestra un mensaje de éxito. El caso de uso termina.
Post-condición	El mapa se añade al sistema.
Excepciones	
Variación	**Acción**
2b, 3b, 4b	El actor sale del sistema y el caso de uso queda sin efecto.
4c	El actor selecciona la opción *CANCELAR* y el caso de uso queda sin efecto.
5b	El sistema detecta que los datos no son correctos, muestra un mensaje de error y el caso de uso continúa en el paso 3.

Tabla 4.18: Caso de Uso 5.2 - Añadir mapa

CU-5.3	**Ver mapa**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador visualizar el detalle de un mapa del listado.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver mapas*
2	El actor selecciona un mapa de la lista.
3	El sistema carga los datos del mapa seleccionado. El caso de uso termina.
Post-condición	
Excepciones	
Variación	**Acción**
2b, 3b	El actor sale del sistema. El caso de uso queda sin efecto.

Tabla 4.19: Caso de Uso 5.3 - Ver mapa

CU-5.4	**Modificar mapa**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador modificar mapas ya existentes.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver mapa*.
2	El actor selecciona *Acciones*.
3	El actor selecciona *Editar Planta*.
4	El sistema muestra los datos actuales del mapa para modificarlos.
5	El actor modifica los datos y pulsa el botón *GUARDAR*.
6	El sistema comprueba que los datos son correctos.
7	El sistema muestra un mensaje de éxito. El caso de uso termina.
Post-condición	Los datos del mapa se modifican.
Excepciones	
Variación	**Acción**
2b, 3b, 4b, 5b, 6b	El actor sale del sistema y el caso de uso queda sin efecto.
4c, 5c	El actor selecciona la opción cancelar y el caso de uso queda sin efecto.
6c	El sistema detecta que los datos no son correctos, muestra un mensaje de error y el caso de uso queda sin efecto.

Tabla 4.20: Caso de Uso 5.4 - Modificar mapa

CU-5.5	**Eliminar mapa**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador eliminar mapas ya existentes.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	Se realiza el caso de uso *Ver mapa*.
2	El actor selecciona *Acciones*.
3	El actor selecciona *Eliminar Planta*.
4	El sistema muestra un mensaje informando de lo que se va a realizar y solicita confirmación.
5	El actor pulsa *Aceptar*.
6	El sistema elimina el mapa. El caso de uso termina.
Post-condición	El mapa se elimina del sistema.
Excepciones	
Variación	**Acción**
2b, 3b, 4b	El actor sale del sistema y el caso de uso queda sin efecto.
5b	El actor pulsa el botón de cancelar y el caso de uso queda sin efecto.

Tabla 4.21: Caso de Uso 5.5 - Eliminar mapa

4.1.7. Caso de uso 6: Cerrar sesión

CU-6.1	**Cerrar sesión**
Actor	**Administrador**
Descripción	El sistema deberá permitir al administrador cerrar sesión.
Precondición	
Secuencia Normal	
Paso	**Acción**
1	El actor selecciona la opción *ADMIN* en el menú superior.
2	El actor selecciona *CERRAR SESIÓN*.
3	El sistema devuelve al actor a la página de iniciar sesión. El caso de uso termina.
Post-condición	El actor consigue cerrar la sesión en el sistema.
Excepciones	
Variación	**Acción**
1b, 2b, 3b	El actor sale del sistema y el caso de uso queda sin efecto.

Tabla 4.22: Caso de Uso 6.1 - Cerrar sesión

4.2. Modelo de dominio

En este apartado se muestra un diagrama con los elementos pertenecientes al dominio del sistema, que son usados en las características añadidas.

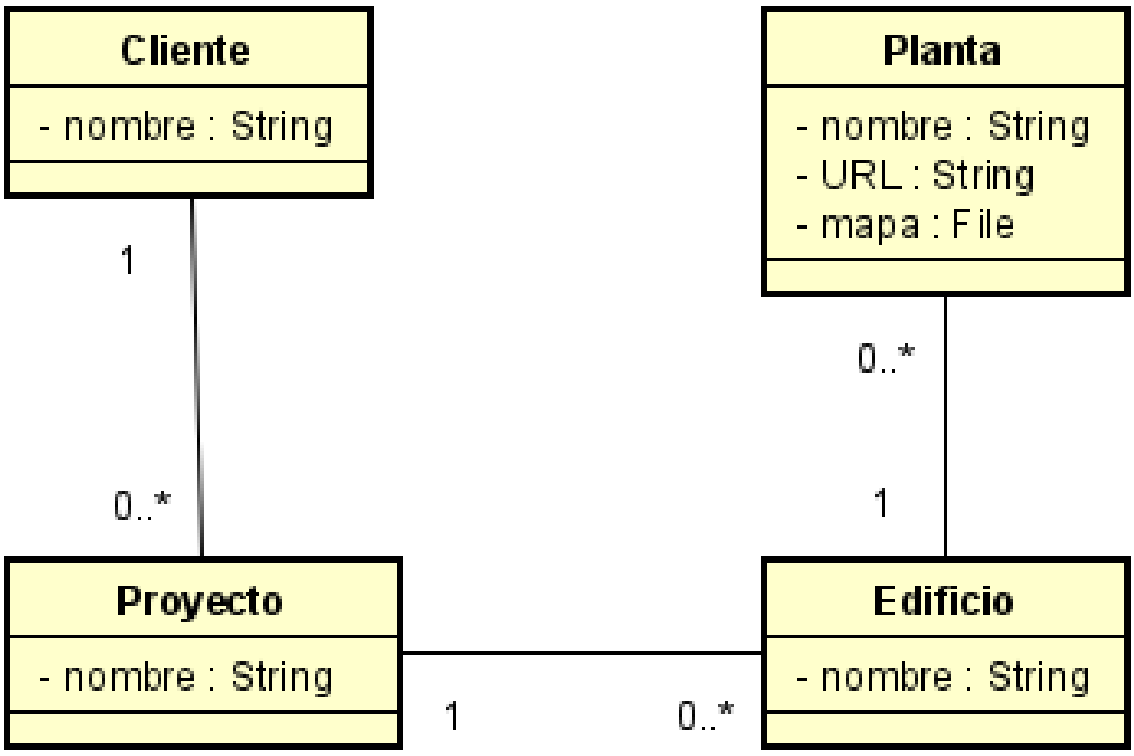

Figura 4.6: Modelo de dominio

Después de llevar a cabo todo el análisis del problema, el siguiente capítulo estará centrado en la elaboración del diseño del sistema, tanto desde el punto de vista de la arquitectura como del visual.

Capítulo 5

Diseño

En este capítulo se detallarán los distintos modelos generados durante la fase de diseño de las distintas partes del sistema, tanto para la parte del servidor como de la interfaz web. Los distintos diagramas y modelos obtenidos se basan en el modelo de análisis, con el objetivo de extenderlo y así mostrar la arquitectura final del sistema, así cómo las interacciones que se consideran más relevantes entre el usuario y los distintos servicios desarrollados.

5.1. Arquitectura del sistema

En este punto se hablará y detallará la arquitectura elegida para cada parte del sistema que se ha desarrollado. Para llevar a cabo el diseño de la aplicación el servicio se ha dividido en dos partes, implementando una arquitectura multicapa cliente-servidor.

5.1.1. Arquitectura del Back-end: Flask

La arquitectura elegida para llevar a cabo la implementación de la parte del servidor es una arquitectura de capas, con 3 capas estrictas y una capa relajada [1]. En una arquitectura de este tipo se permite que la comunicación entre capas sea más flexible, de modo que una capa de nivel n pueda acceder a los servicios de una capa de nivel *n-1*, aunque para nuestro contexto sólo se permite esa acción de las capas estrictas hacia la capa relajada(*utils*).

Se ha elegido usar este tipo de arquitectura para facilitar la implementación de la lógica de parte del servidor, siempre evitando las posibles dependencias circulares que pudieran surgir entre los componentes de la aplicación.

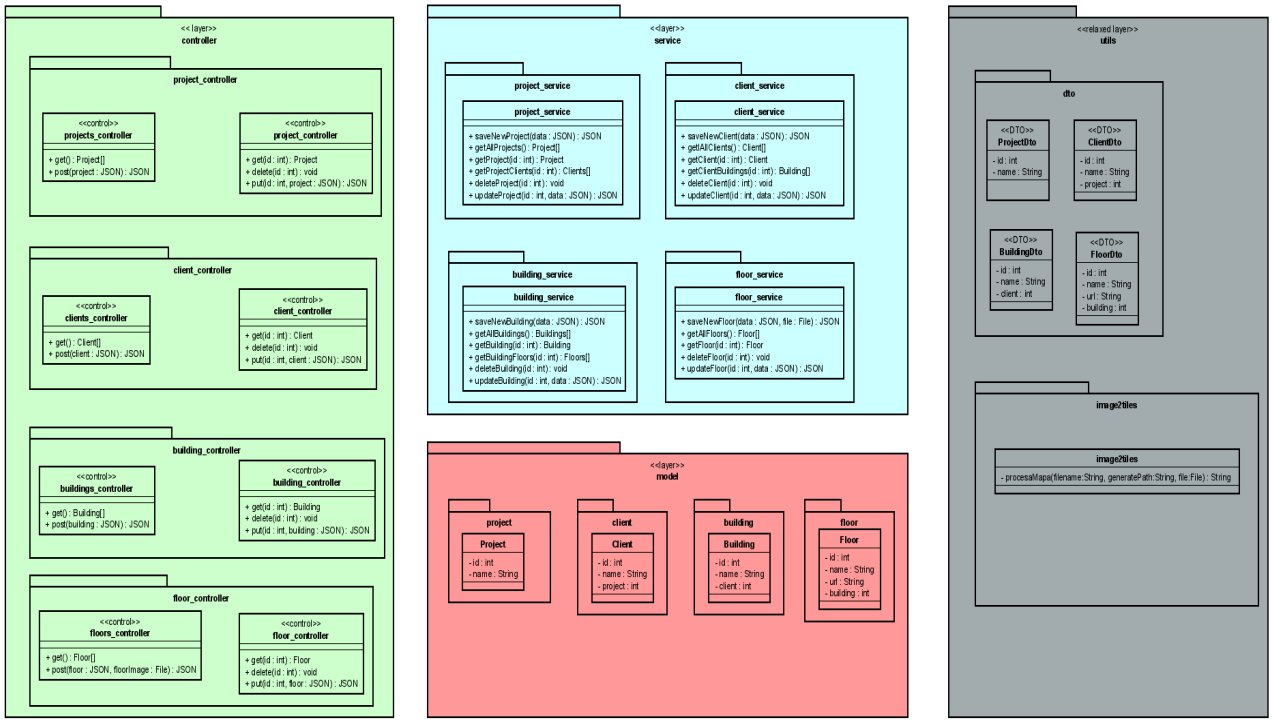

Figura 5.1: Arquitectura usada para el Back-end

5.1.2. Arquitectura Front-end: Vue.js

La arquitectura del sistema ha sido diseñada usando Vuex [25], que es un patrón que gestiona el estado de la interfaz web junto con una librería que extiende el framework Vue.js, adaptando la arquitectura MVC (Modelo-Vista-Controlador).

Los componentes principales de esta arquitectura son los siguientes:

- Estado (**State**): Son los atributos propiamente dichos. Se encargan de almacenar información relacionada con la aplicación.
- **Getters**: Se encargan de obtener los valores almacenados en el *State*.
- Setters (**Mutations**): Se encargan de editar los valores del *State*.
- Funciones (**Actions**): Se encargan de gestionar operaciones relacionadas con la interfaz de usuario.

El orden de acción y comunicación entre los componentes de Vuex se basa en lo siguiente:

- El usuario realiza una interacción con la interfaz y dicha interacción dispara una *acción*. Esa *acción* se llama mediante la directiva *dispatch*.
- La acción disparada se encarga de realizar las propiedades oportunas y "muta" el estado de un atributo del *state*, llamando a una mutation mediante la directiva *commit*.
- En la *mutation* correspondiente se asigna el valor oportuno al *state* que deba asignarse, recibido desde el *commit*.
- Mediante el *getter* correspondiente del estado mutado, podemos acceder a su nuevo valor.

A continuación se muestra un diagrama de la arquitectura indicada, tanto el esquema general del funcionamiento de Vuex como la arquitectura genérica del front-end.

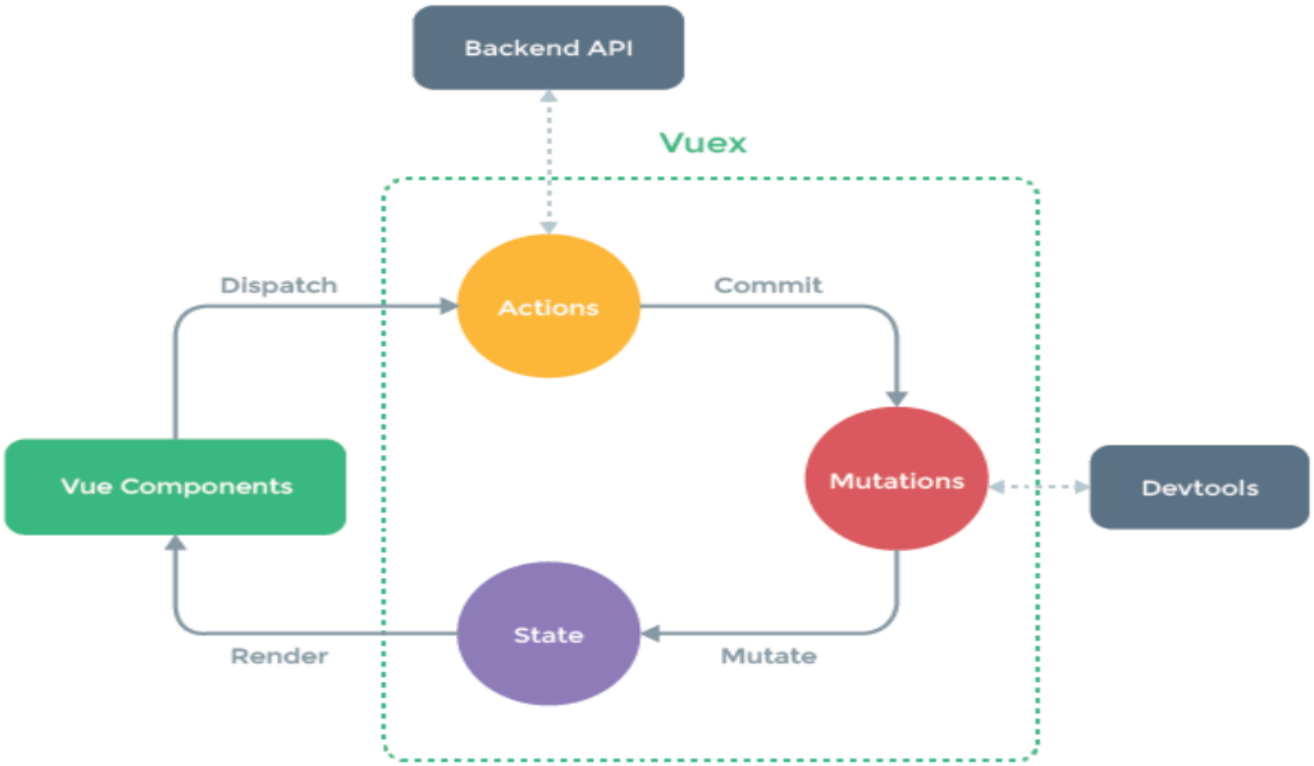

Figura 5.2: Arquitectura usada para Vue: Patrón de Vuex

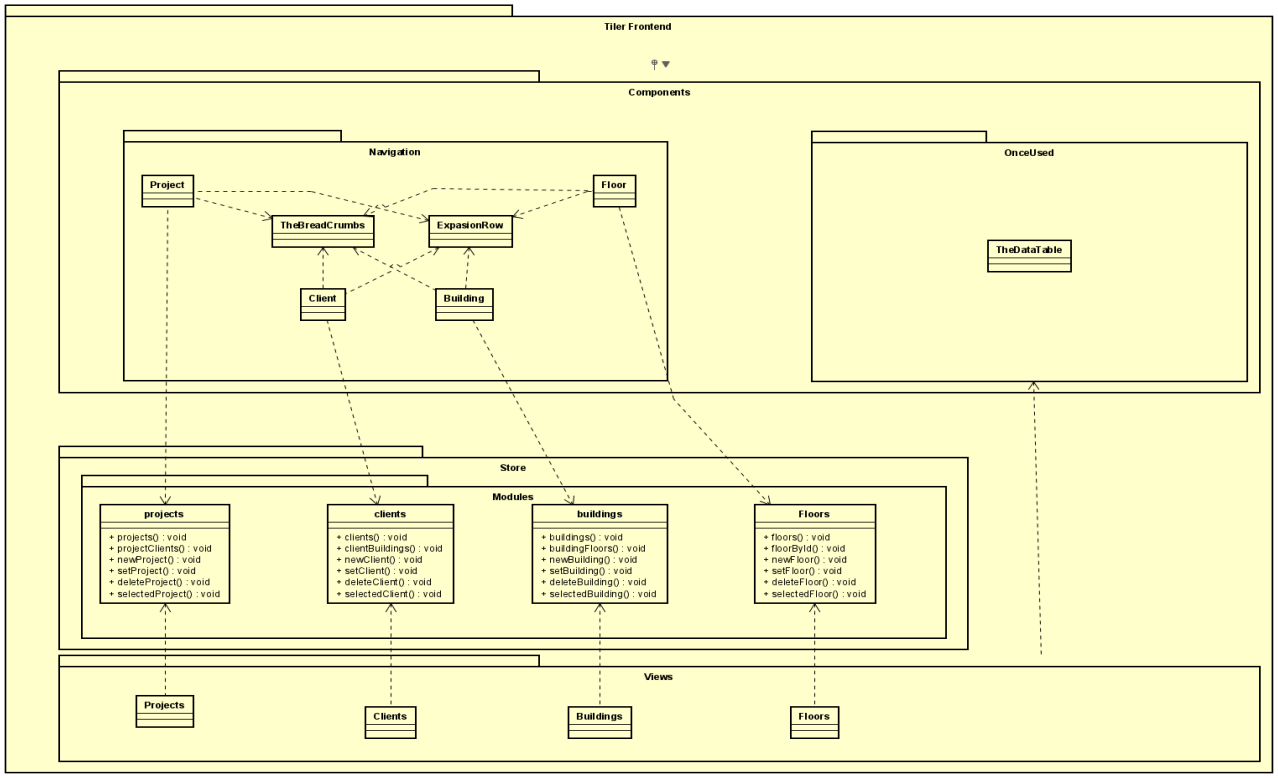

Figura 5.3: Arquitectura usada para el Front-end

5.2. Interfaz de usuario

Los bocetos de este apartado se han realizado utilizando la herramienta web Balsamiq [4]. Se han incluido las vistas principales de la aplicación.

5.2.1. Vista sobre inicio de sesión

Se incluye la vista para iniciar sesión en la aplicación

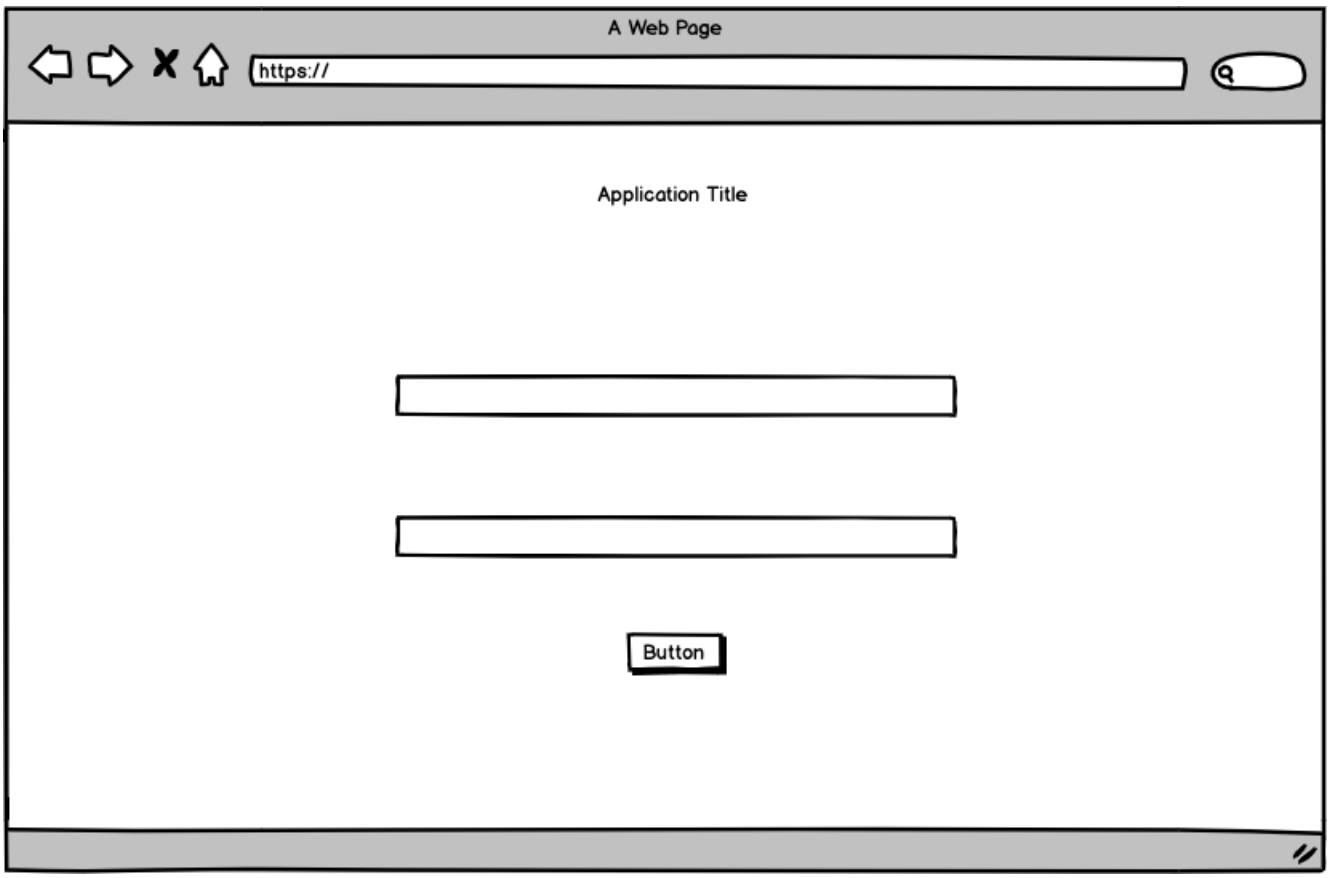

Figura 5.4: Inicio de sesión

5.2.2. Vistas sobre gestión de mapas

Se incluyen las vistas relevantes de este apartado de la aplicación. Son las vistas asociadas a la creación y procesado de mapas, visualizado de datos, modificación y eliminación.

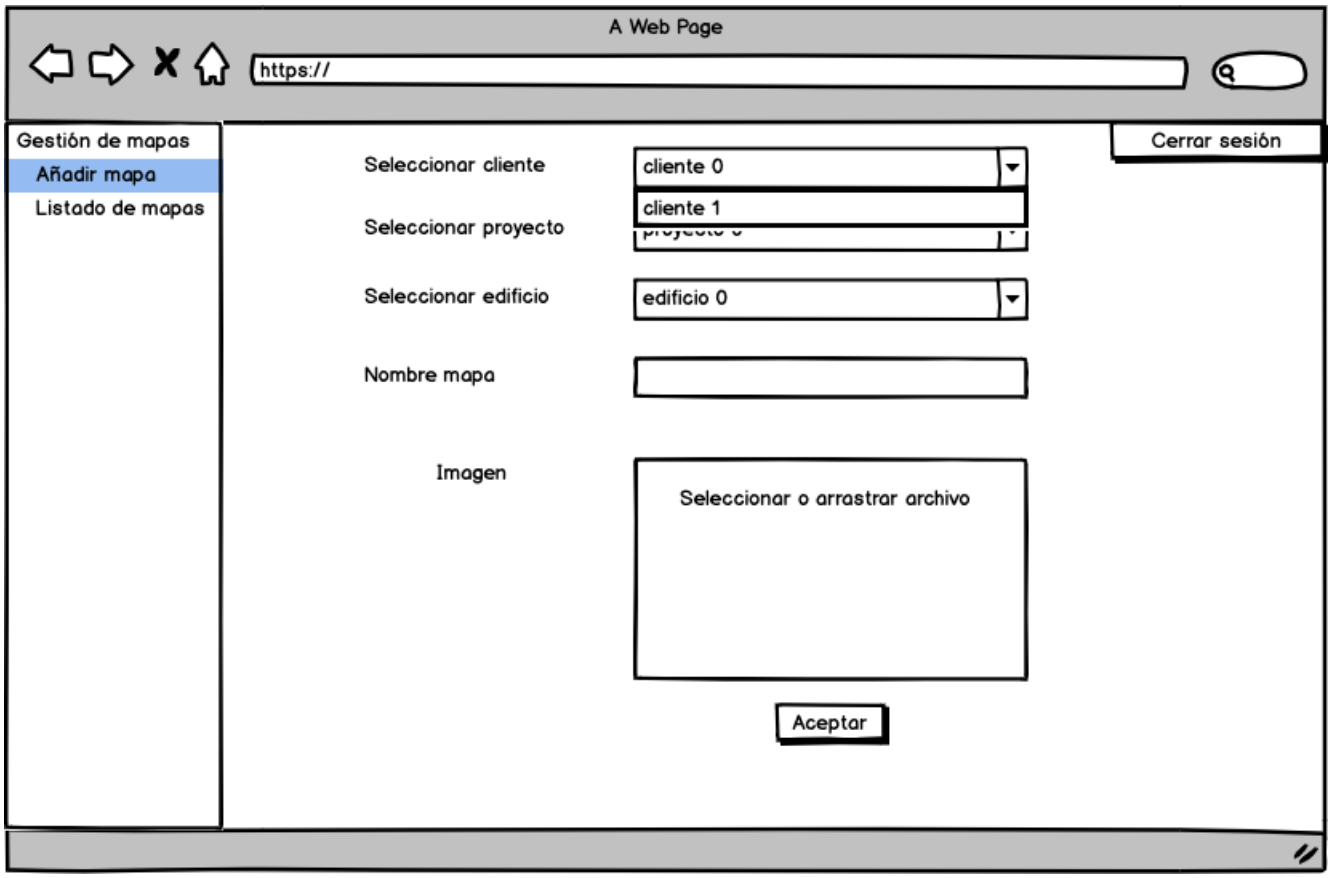

Figura 5.5: Añadir mapa

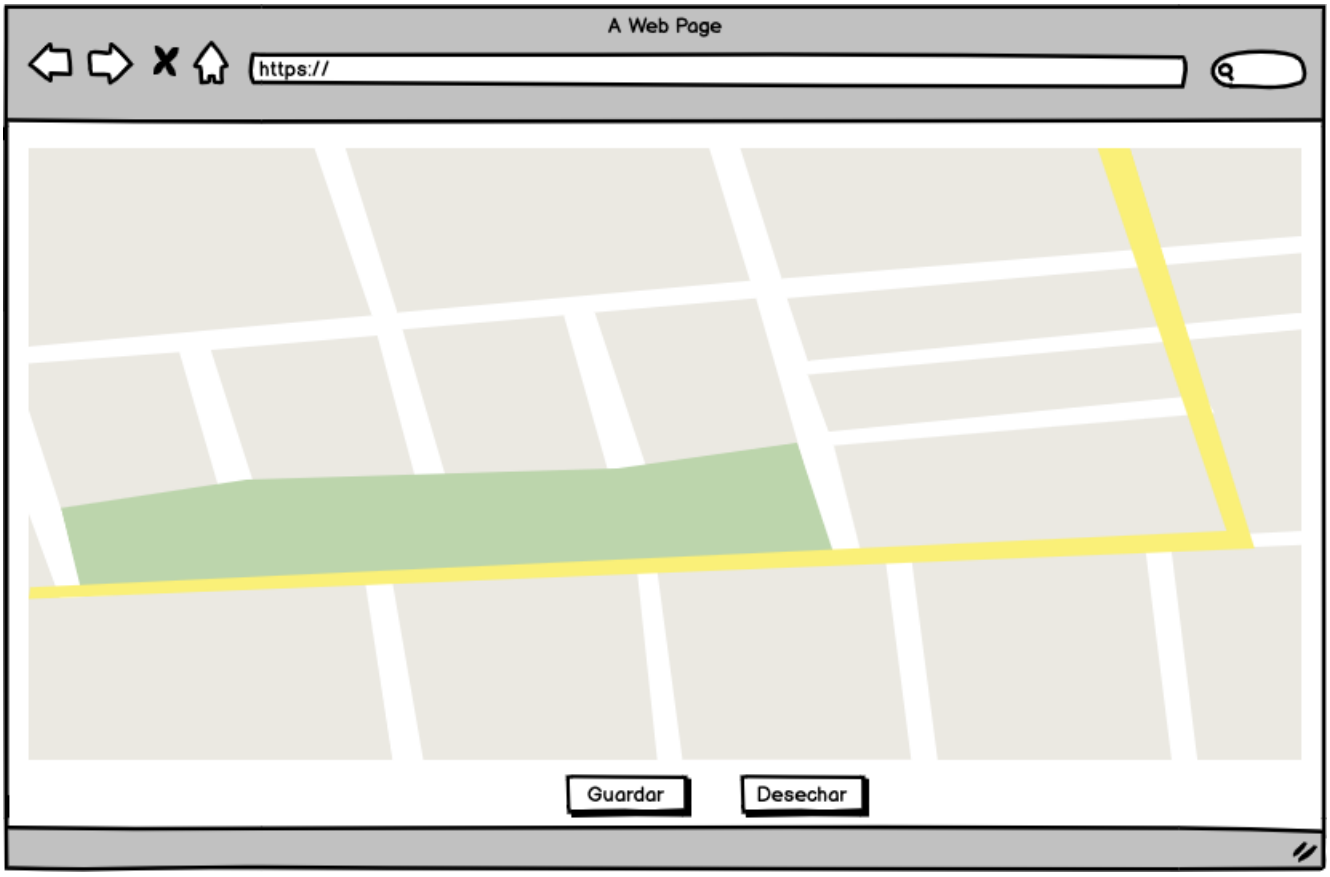

Figura 5.6: Previsualización del mapa añadido

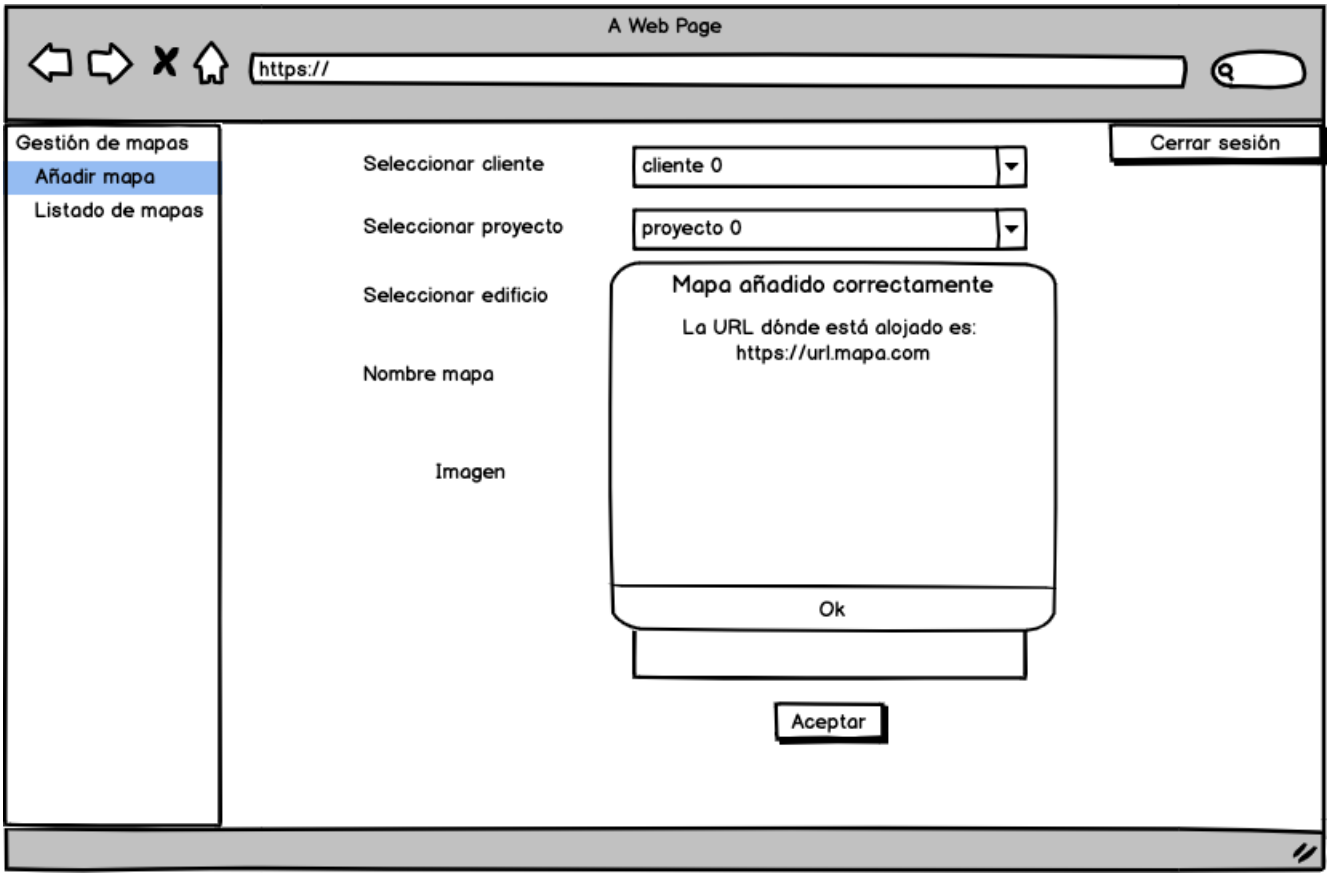

Figura 5.7: Mensaje de confirmación del añadido del mapa

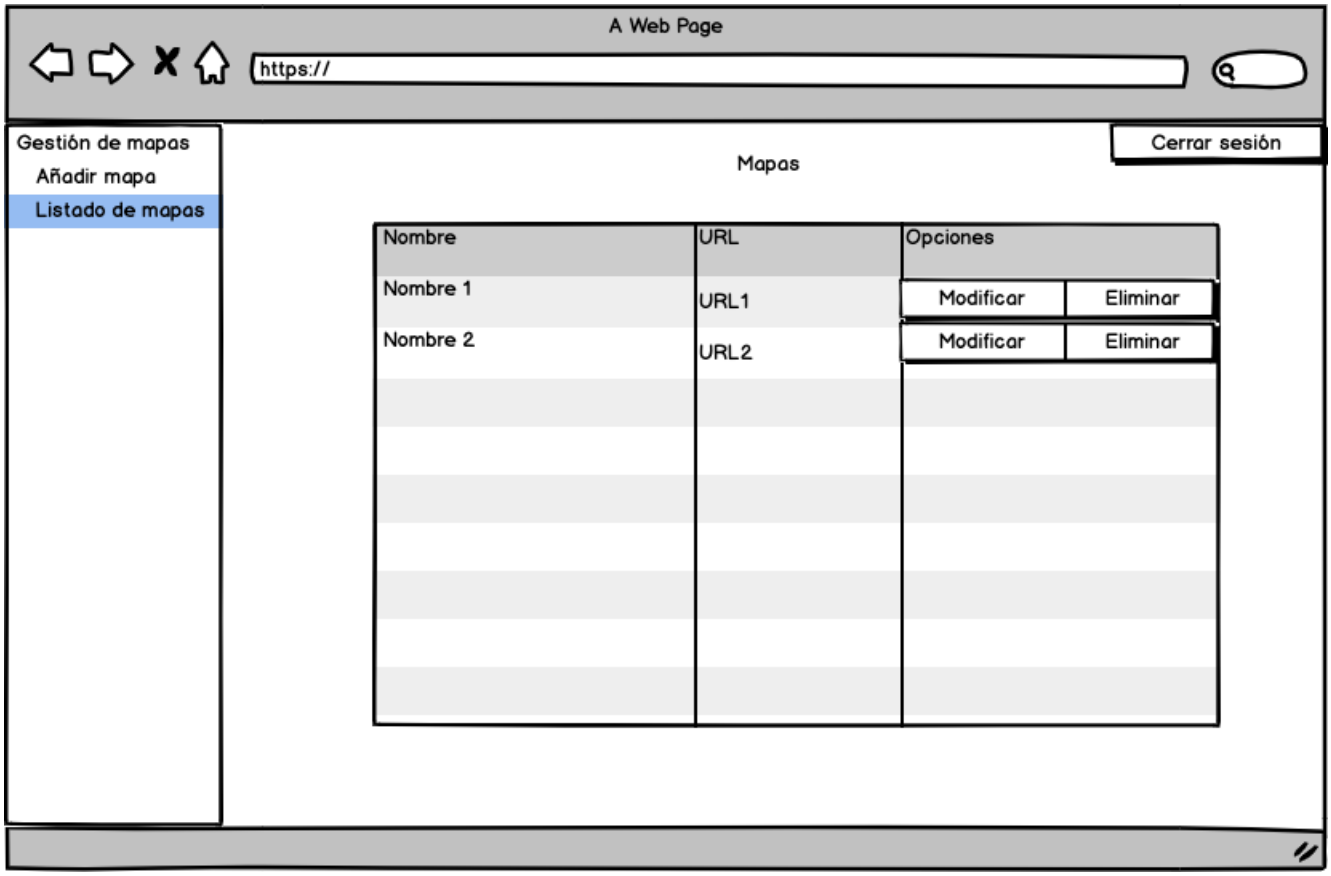

Figura 5.8: Listado de los mapas disponibles en el sistema

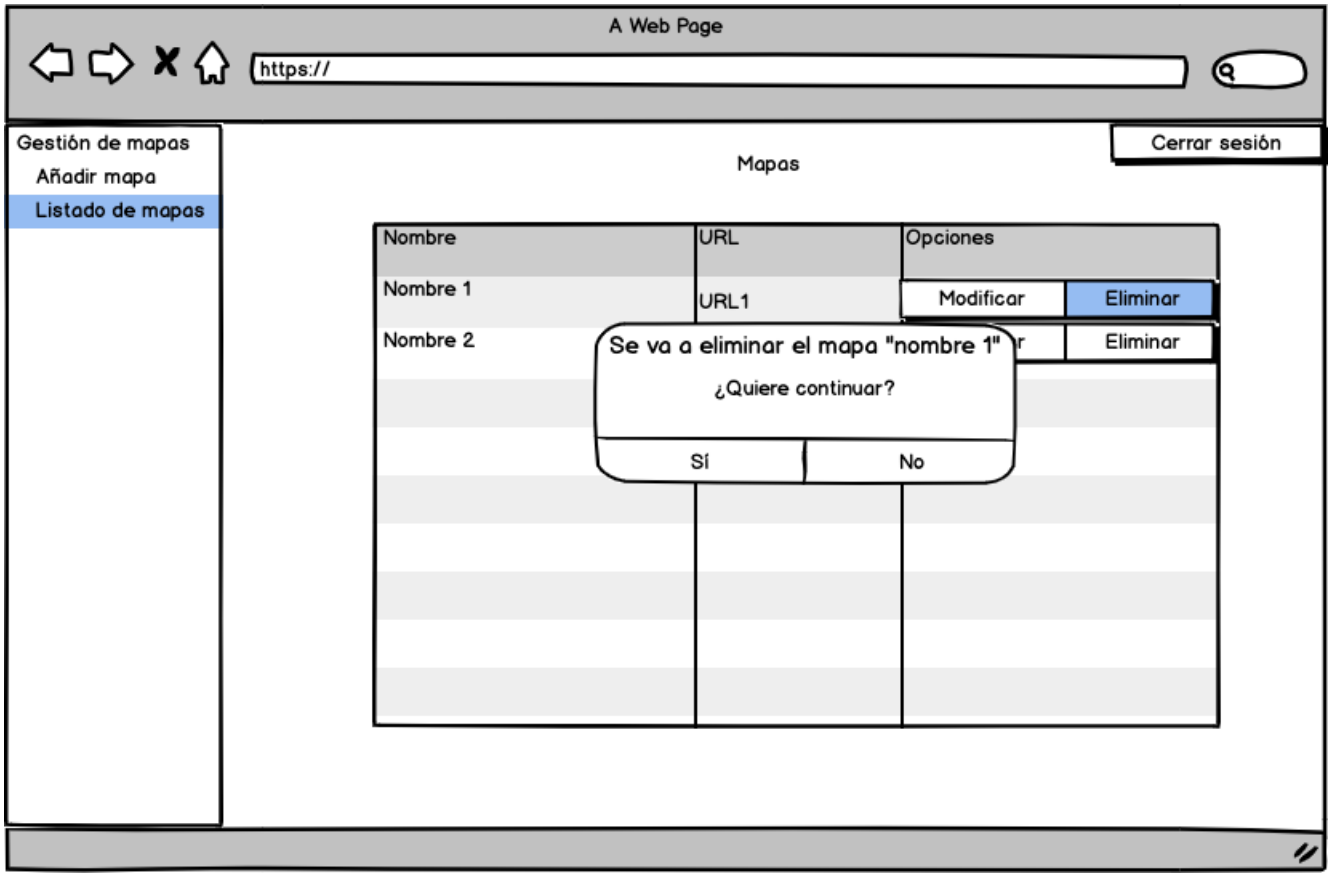

Figura 5.9: Mensaje de confirmación del añadido del mapa

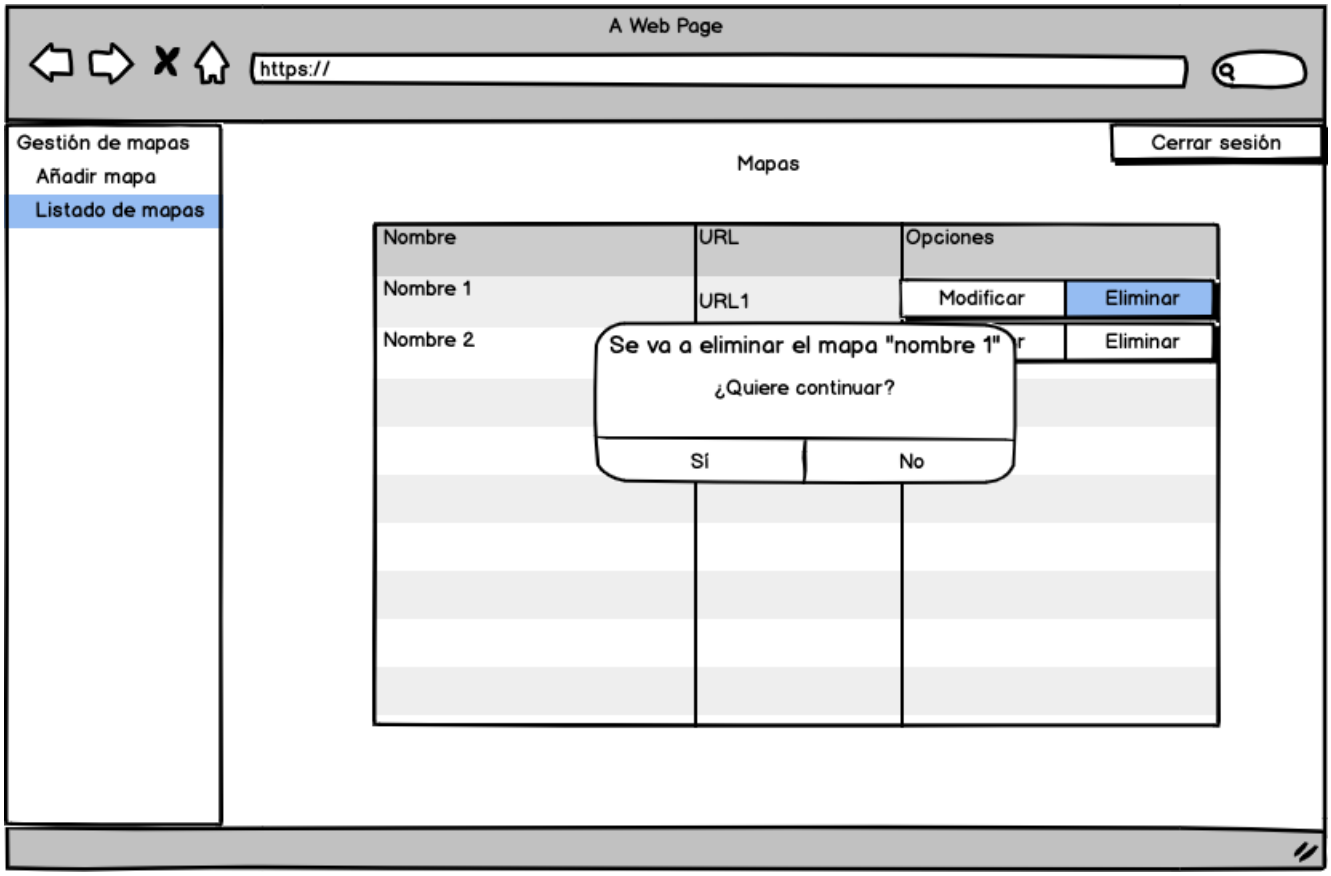

Figura 5.10: Mensaje de confirmación del añadido del mapa

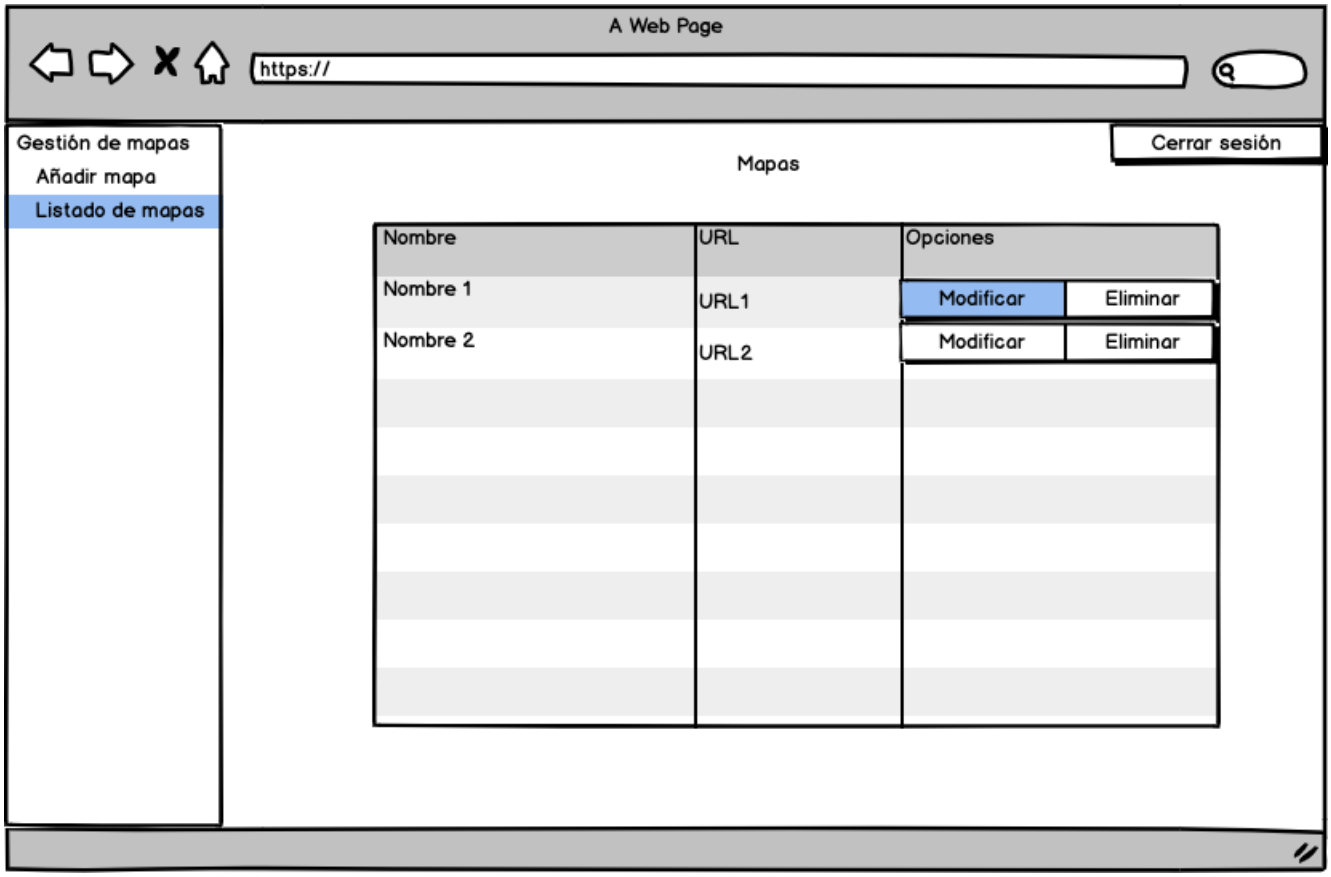

Figura 5.11: Elección de modificación de mapa

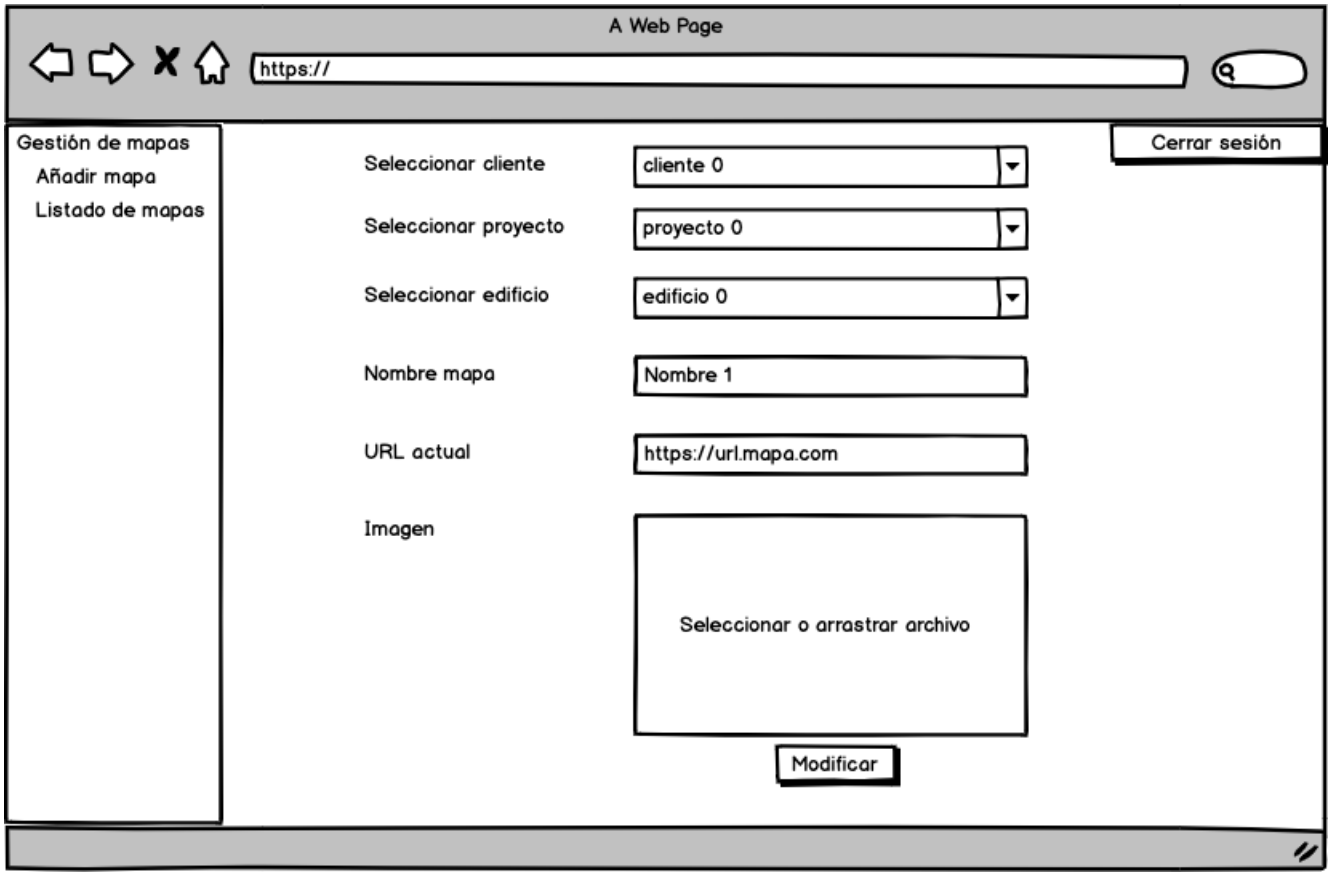

Figura 5.12: Elección de modificación de mapa

5.2.3. Vistas sobre gestión de clientes

Se incluyen las vistas relevantes de este apartado de la aplicación. Son las vistas asociadas a la creación, visualizado de datos, modificación y eliminación de clientes.

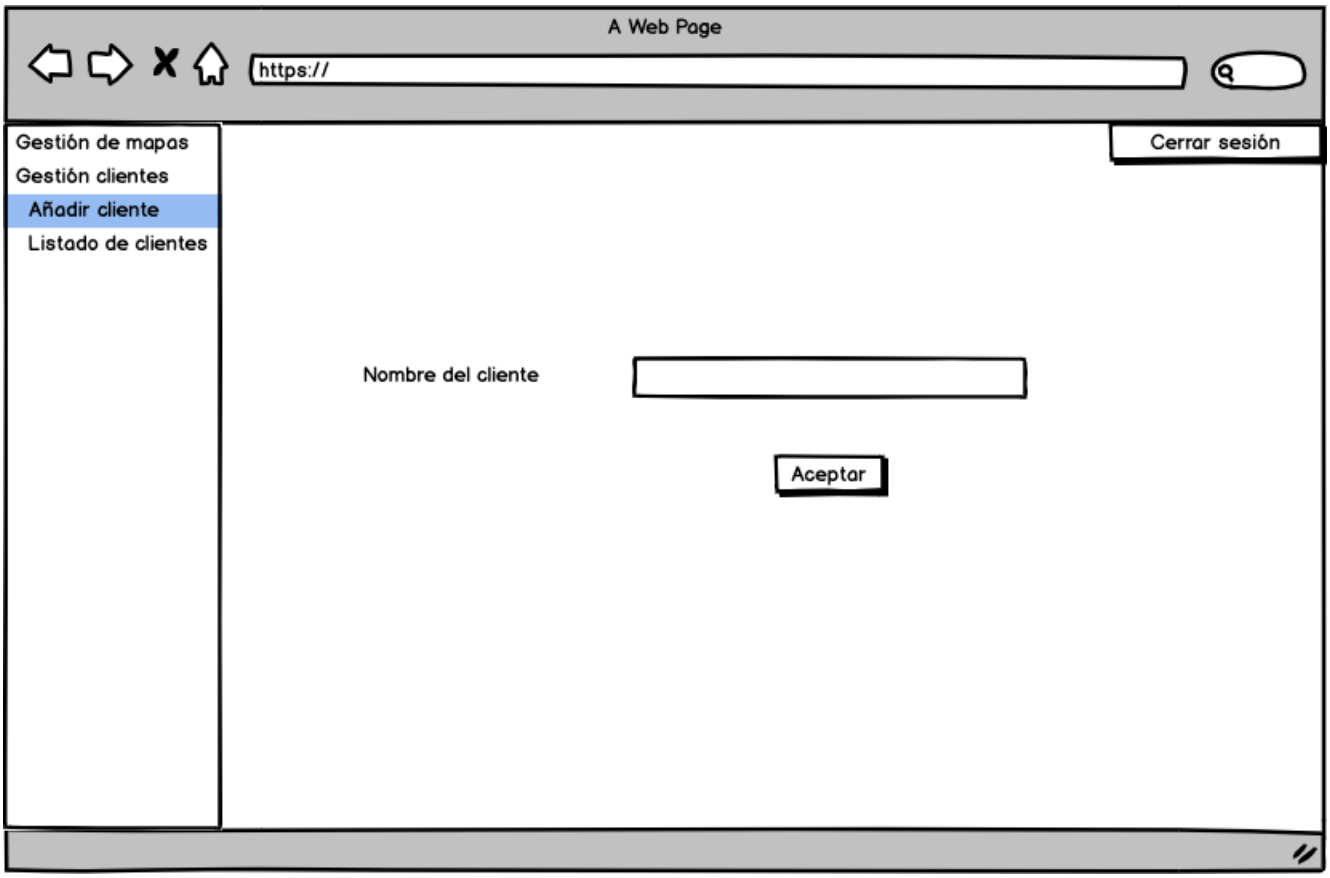

Figura 5.13: Añadir un cliente

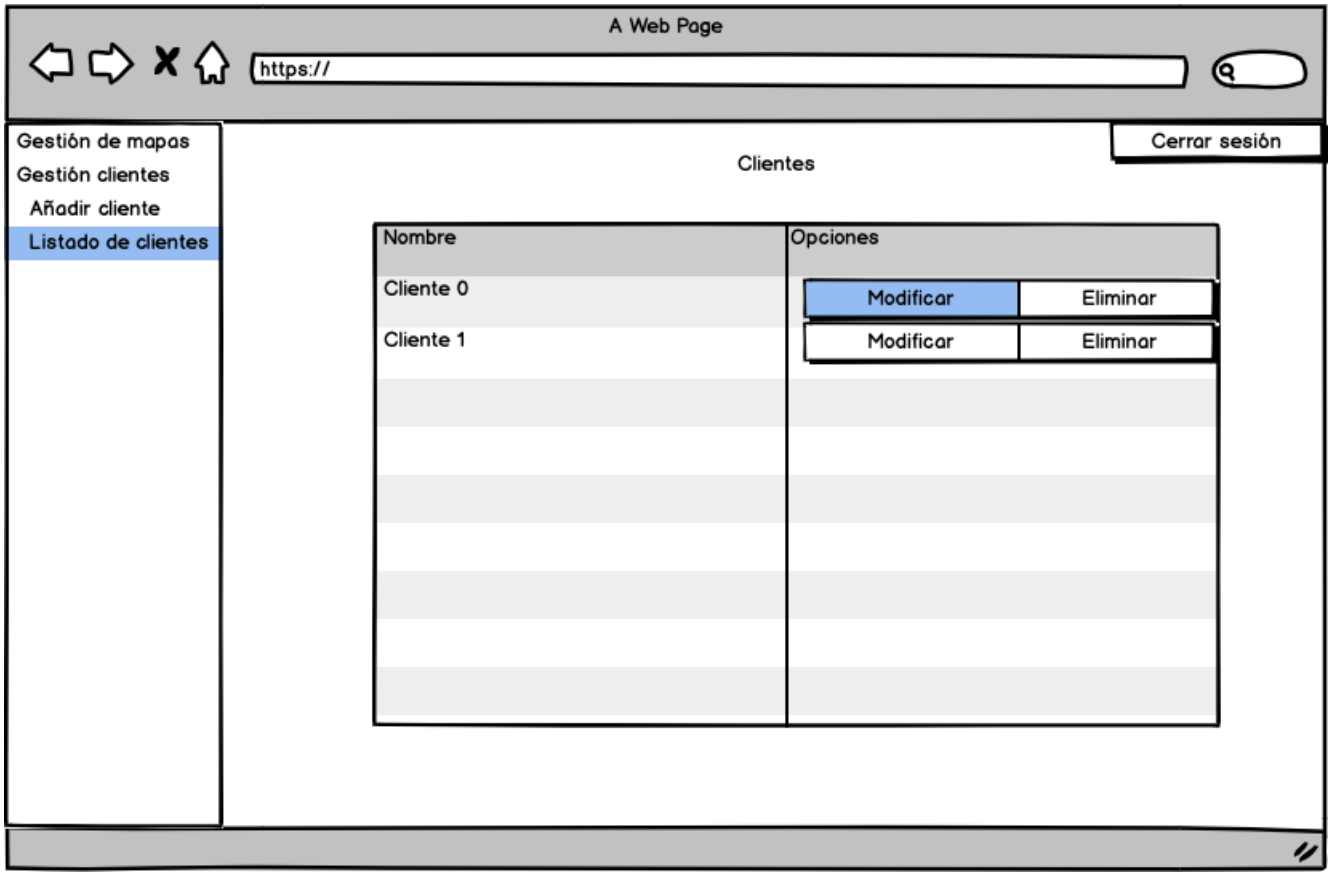

Figura 5.14: Listado de clientes

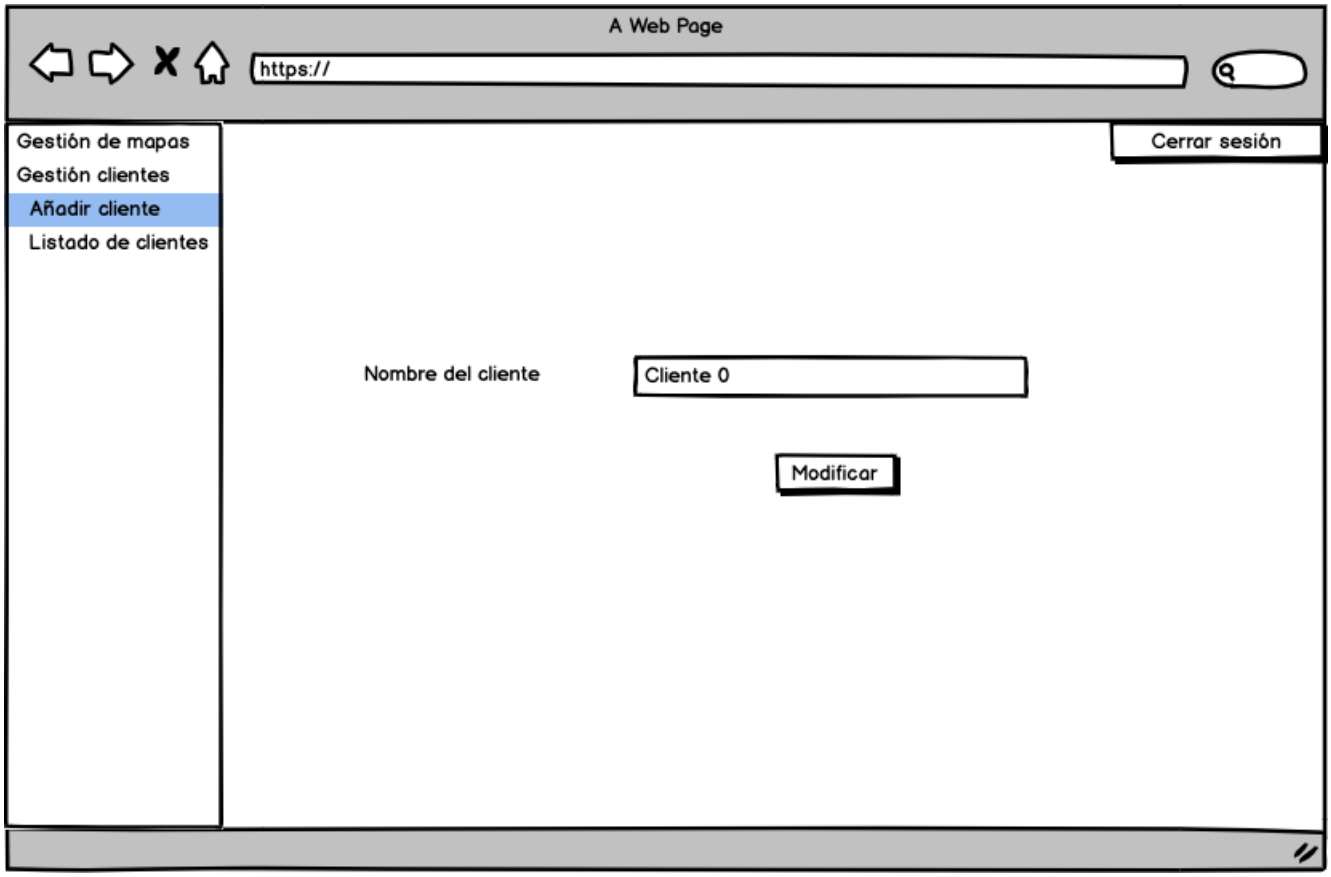

Figura 5.15: Modificar un cliente

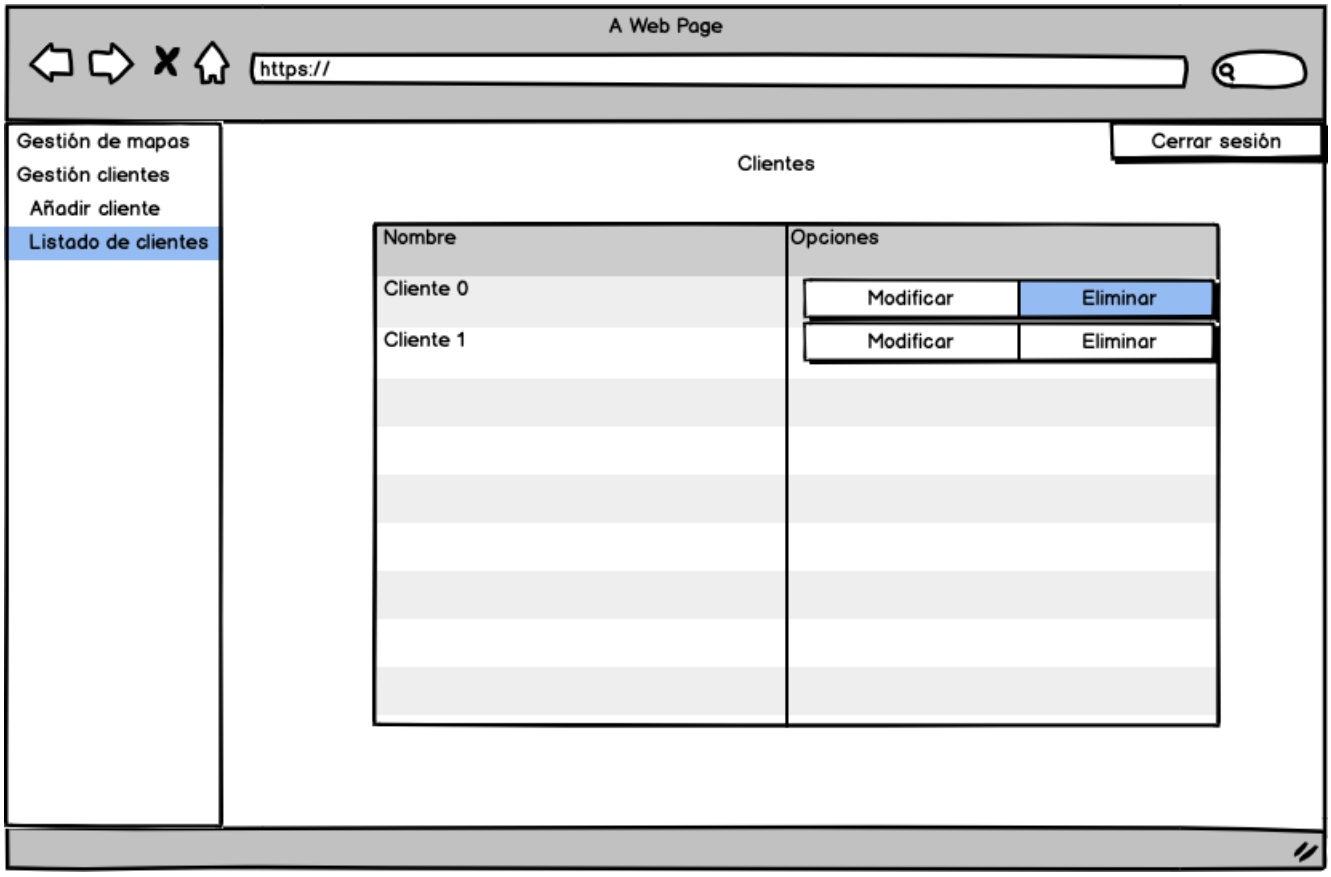

Figura 5.16: Selección de eliminar cliente

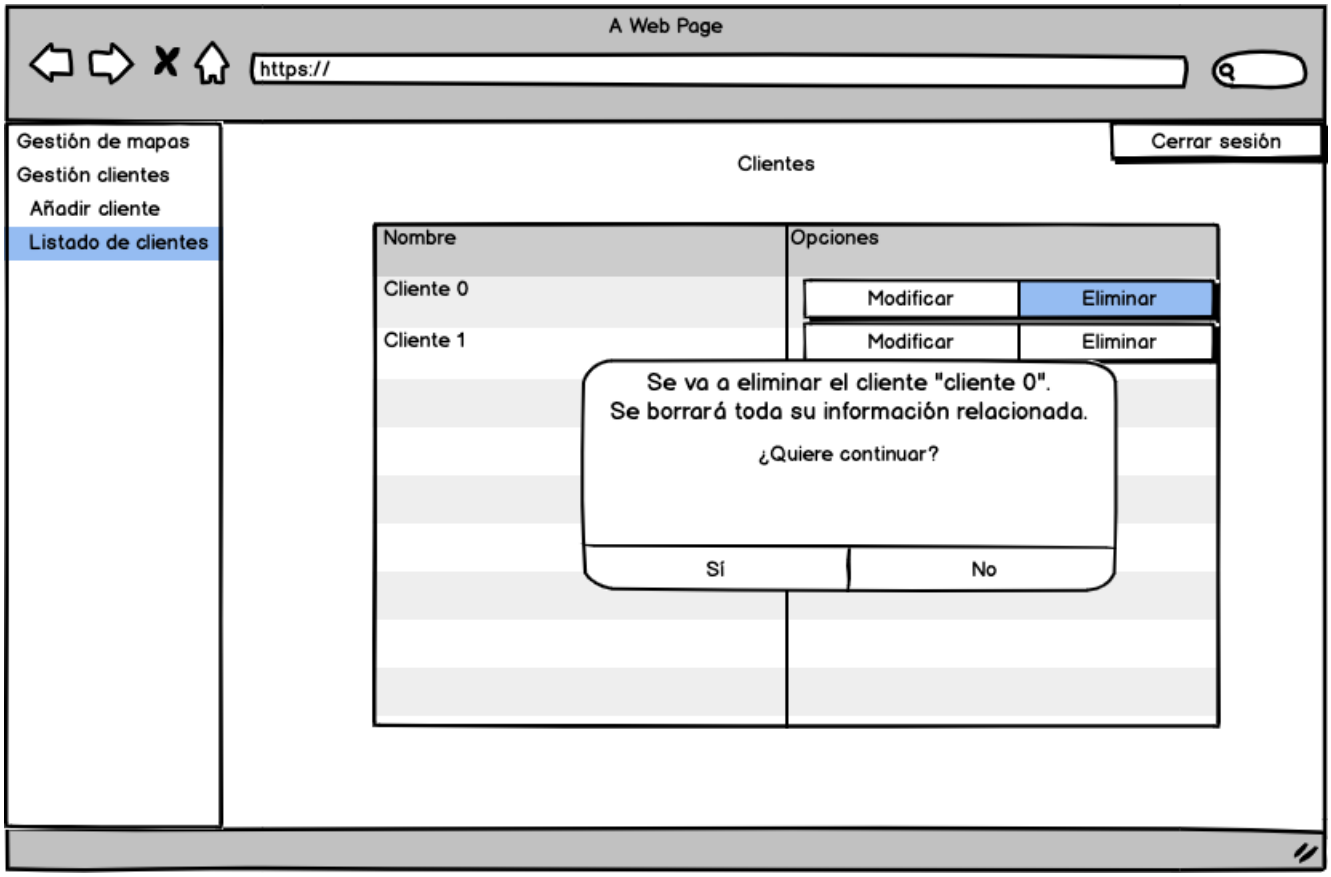

Figura 5.17: Eliminar cliente

5.2.4. Vista sobre gestión de proyectos

Se incluyen las vistas relevantes de este apartado de la aplicación. Son las vistas asociadas a la creación, visualizado de datos, modificación y eliminación de proyectos.

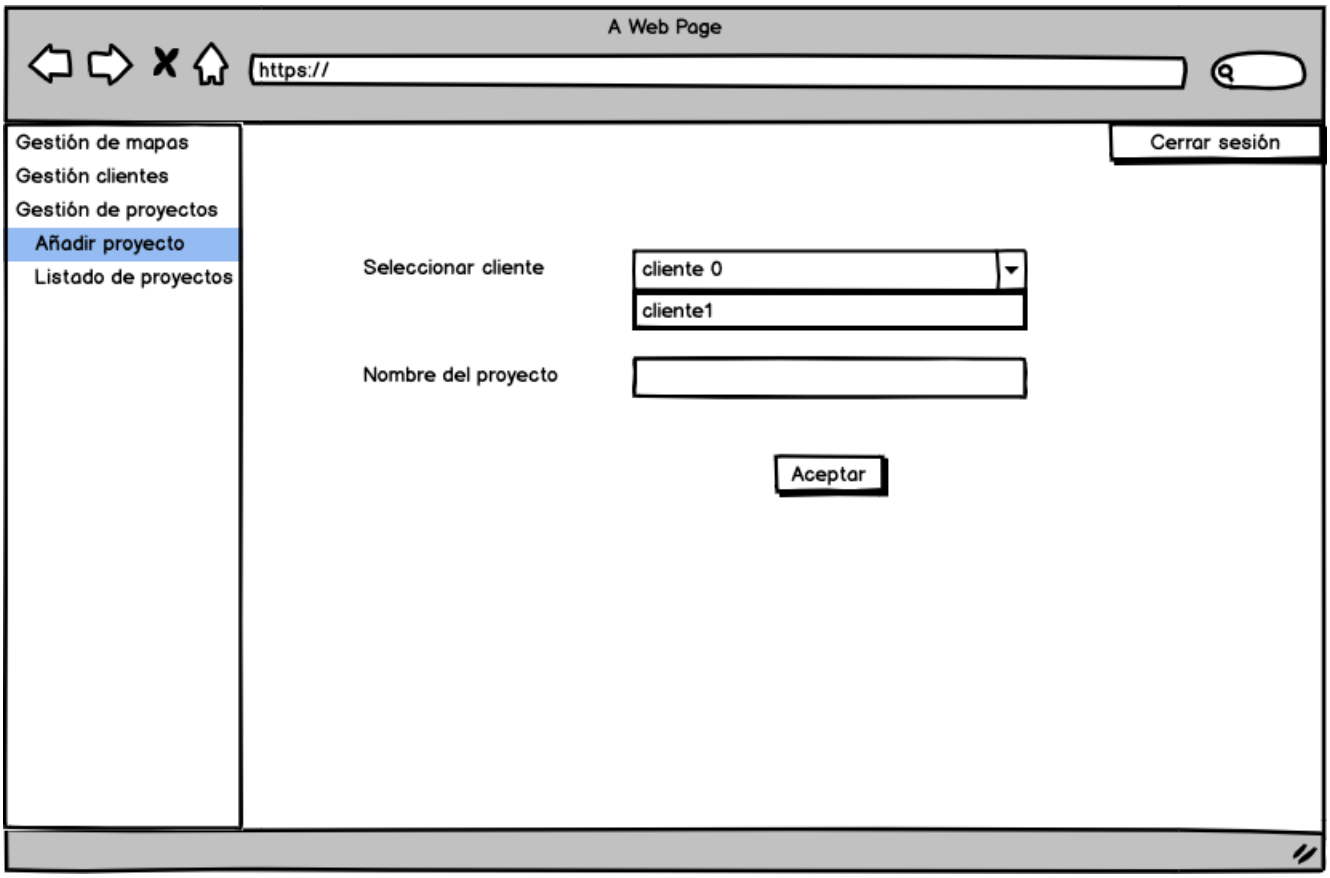

Figura 5.18: Añadir un proyecto

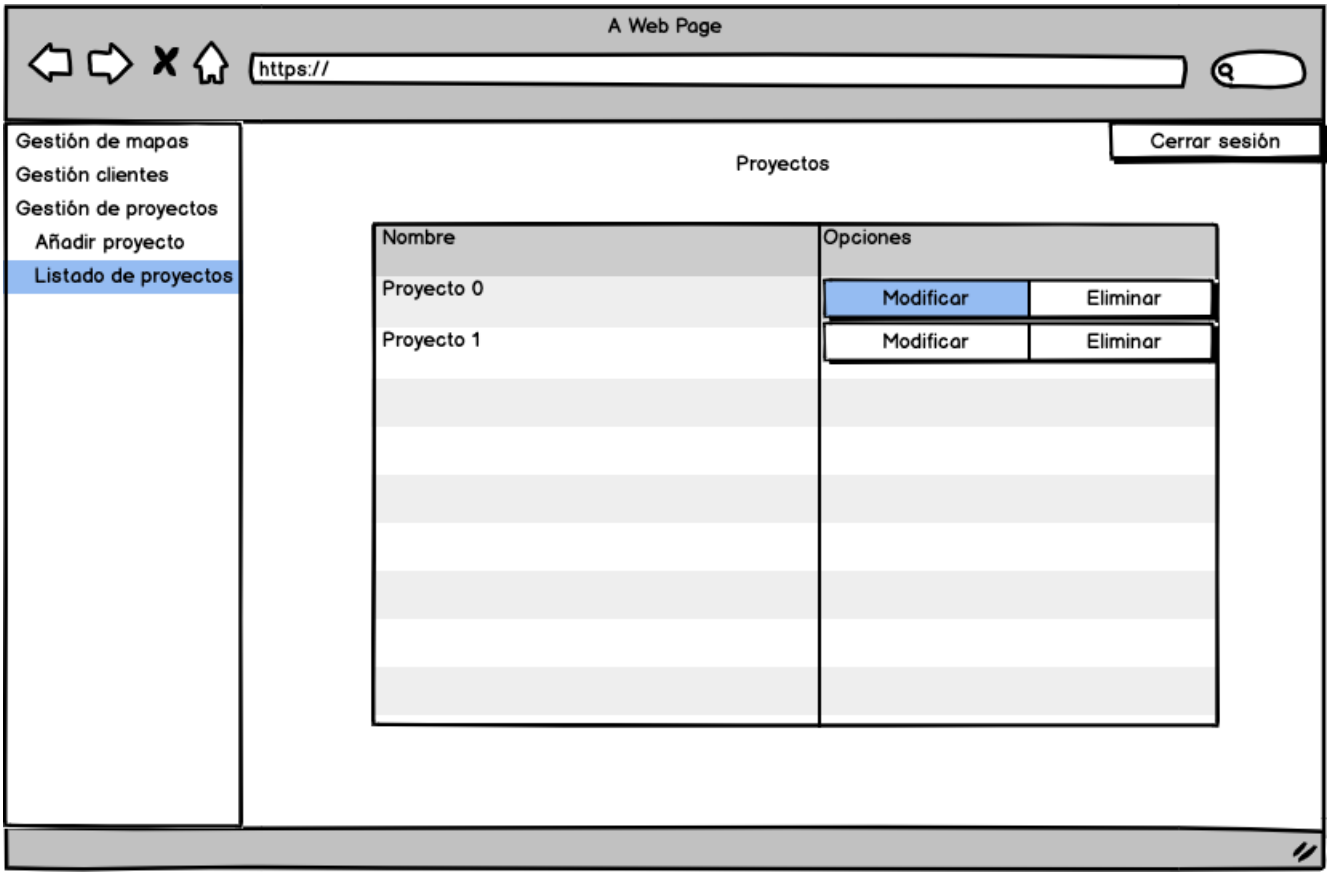

Figura 5.19: Listado de proyectos

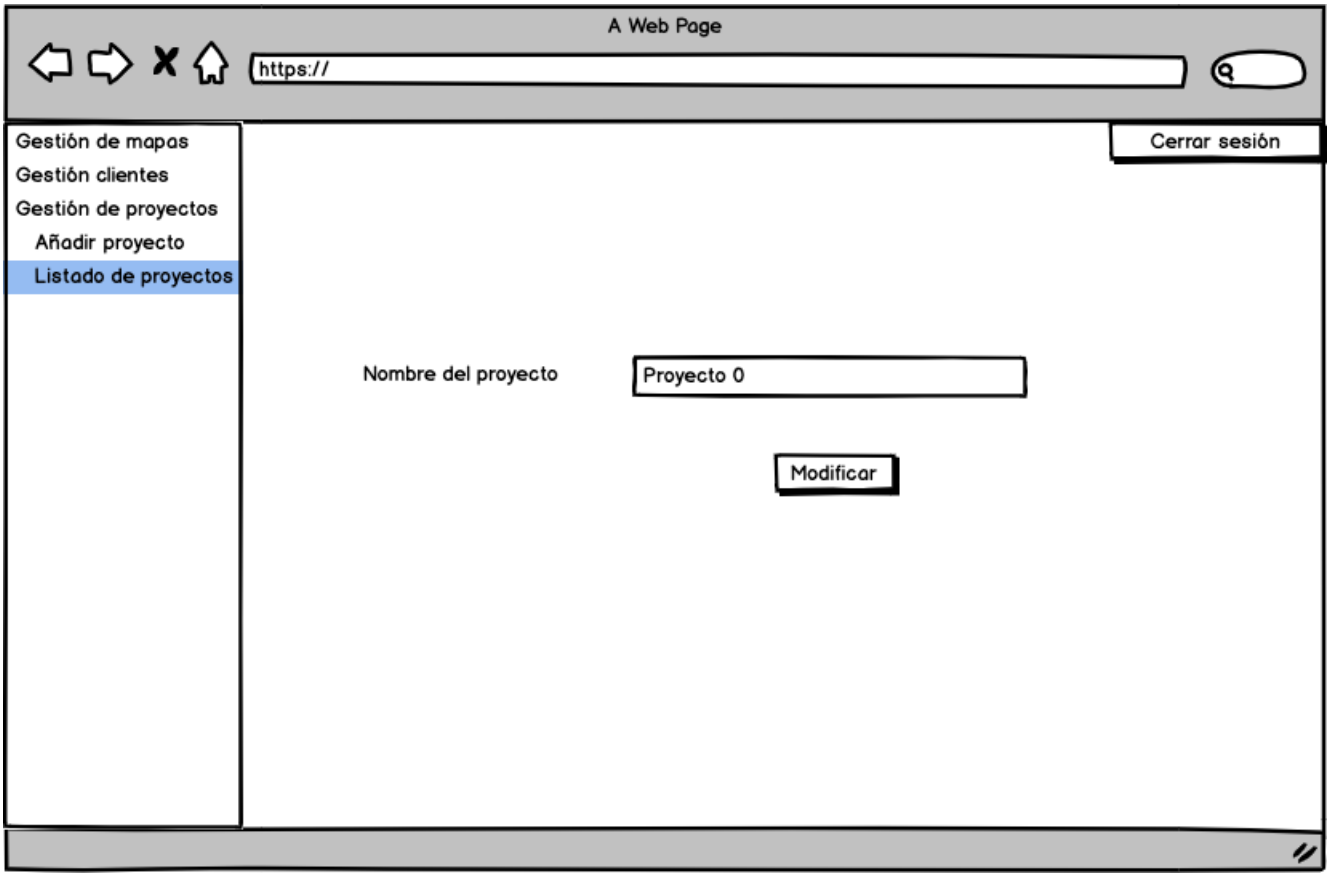

Figura 5.20: Modificar un proyecto

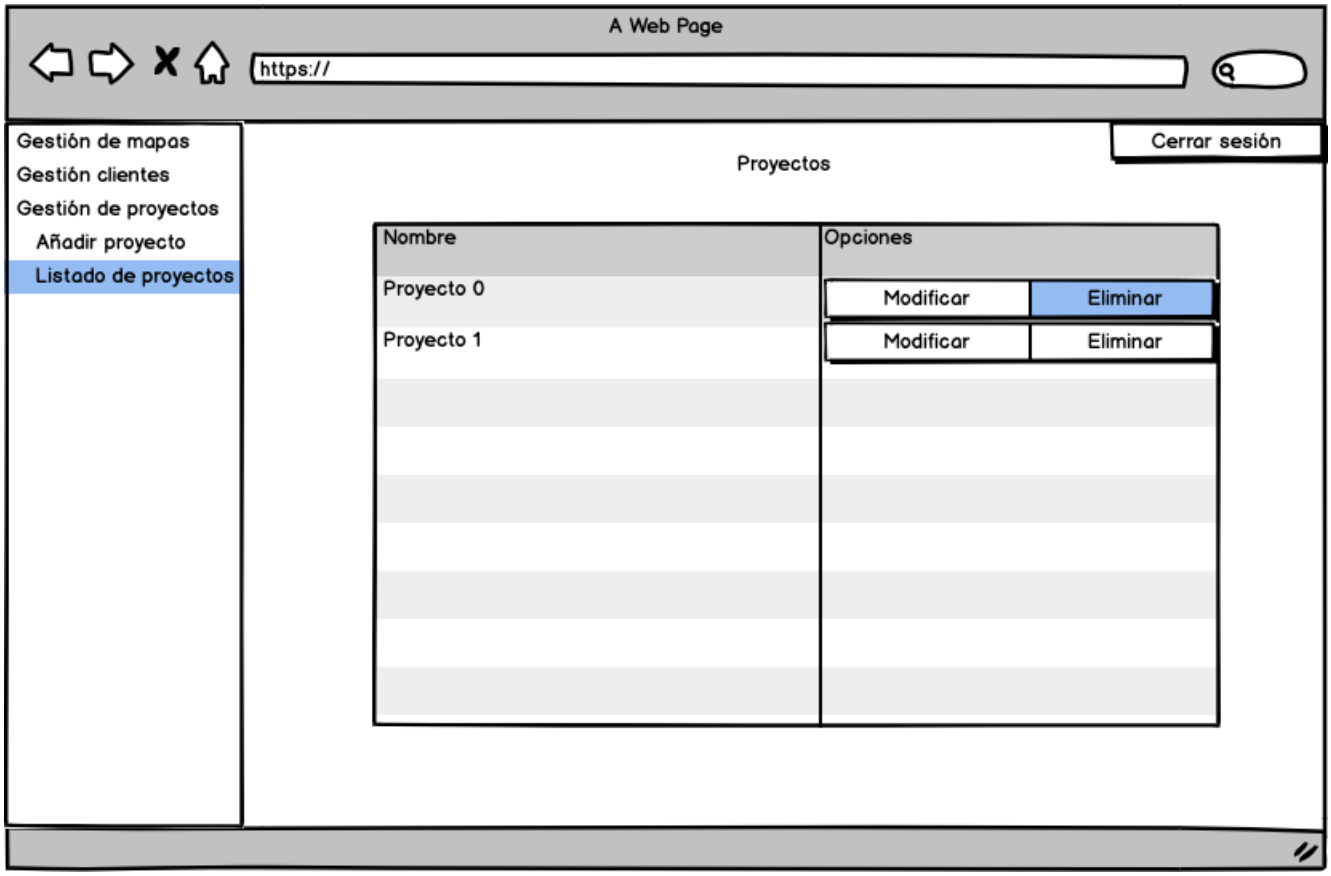

Figura 5.21: Selección de eliminar proyecto

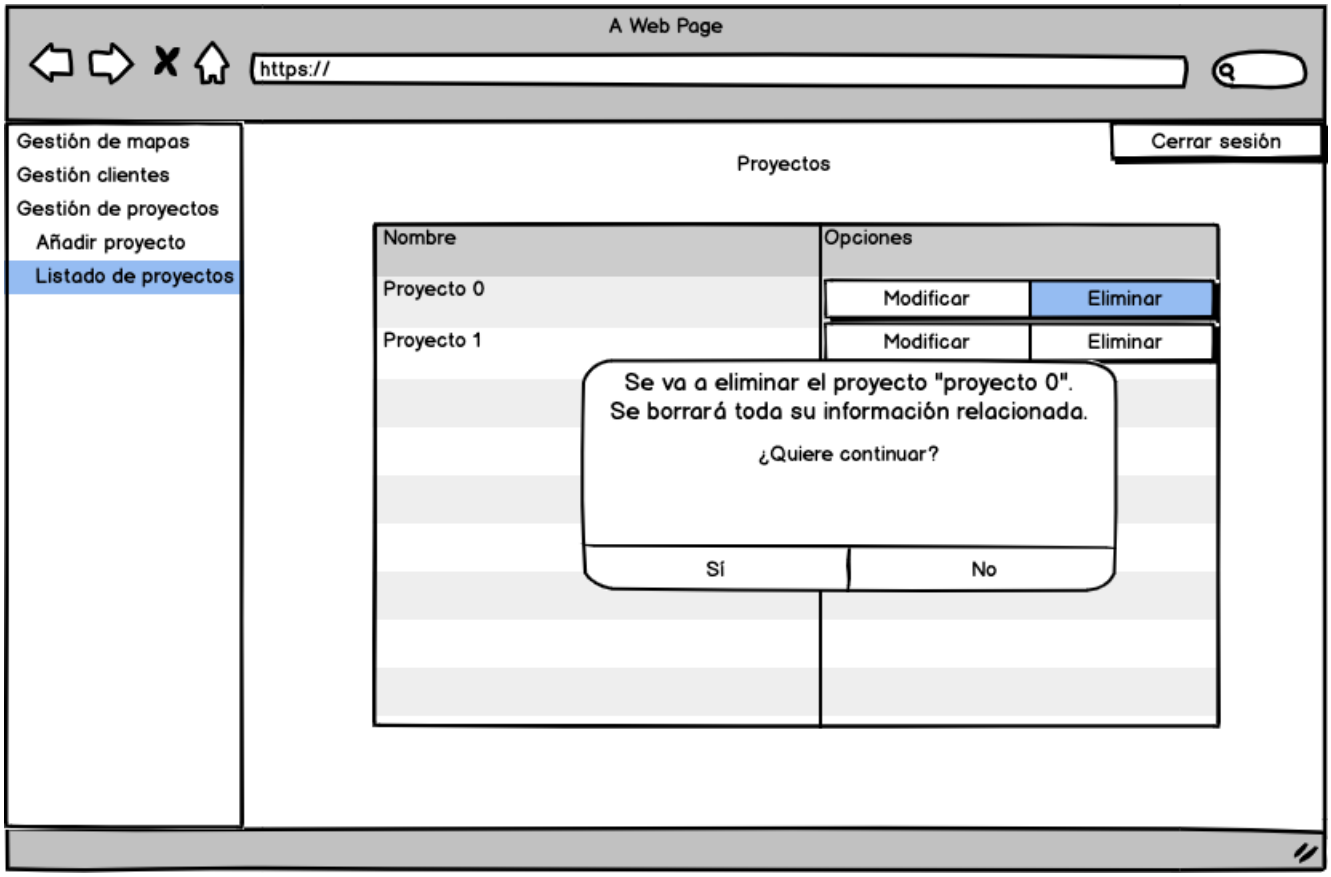

Figura 5.22: Eliminar proyecto

5.2.5. Vista sobre gestión de edificios

Se incluyen las vistas relevantes de este apartado de la aplicación. Son las vistas asociadas a la creación, visualizado de datos, modificación y eliminación de edificios.

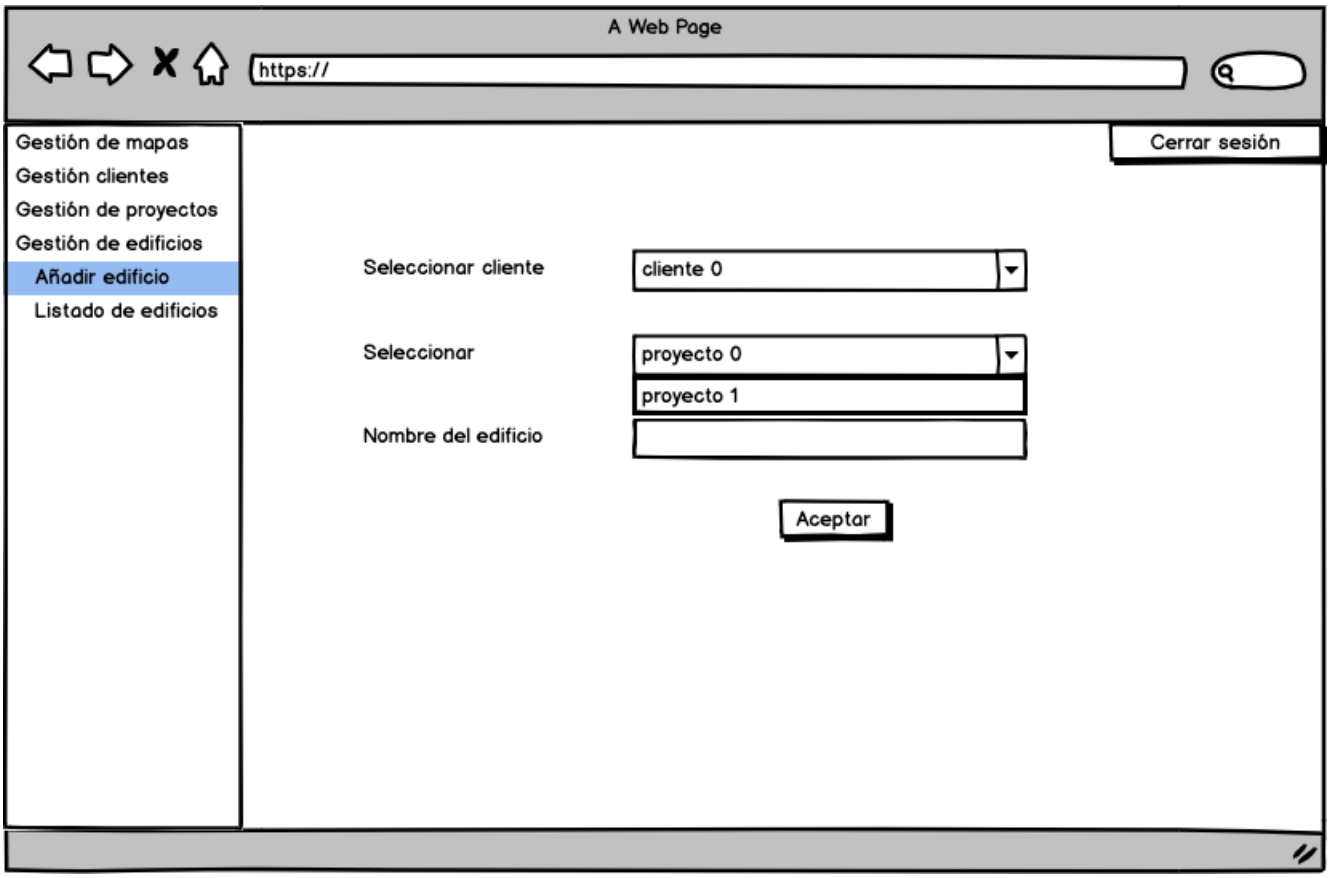

Figura 5.23: Añadir un edificio

Figura 5.24: Listado de edificios

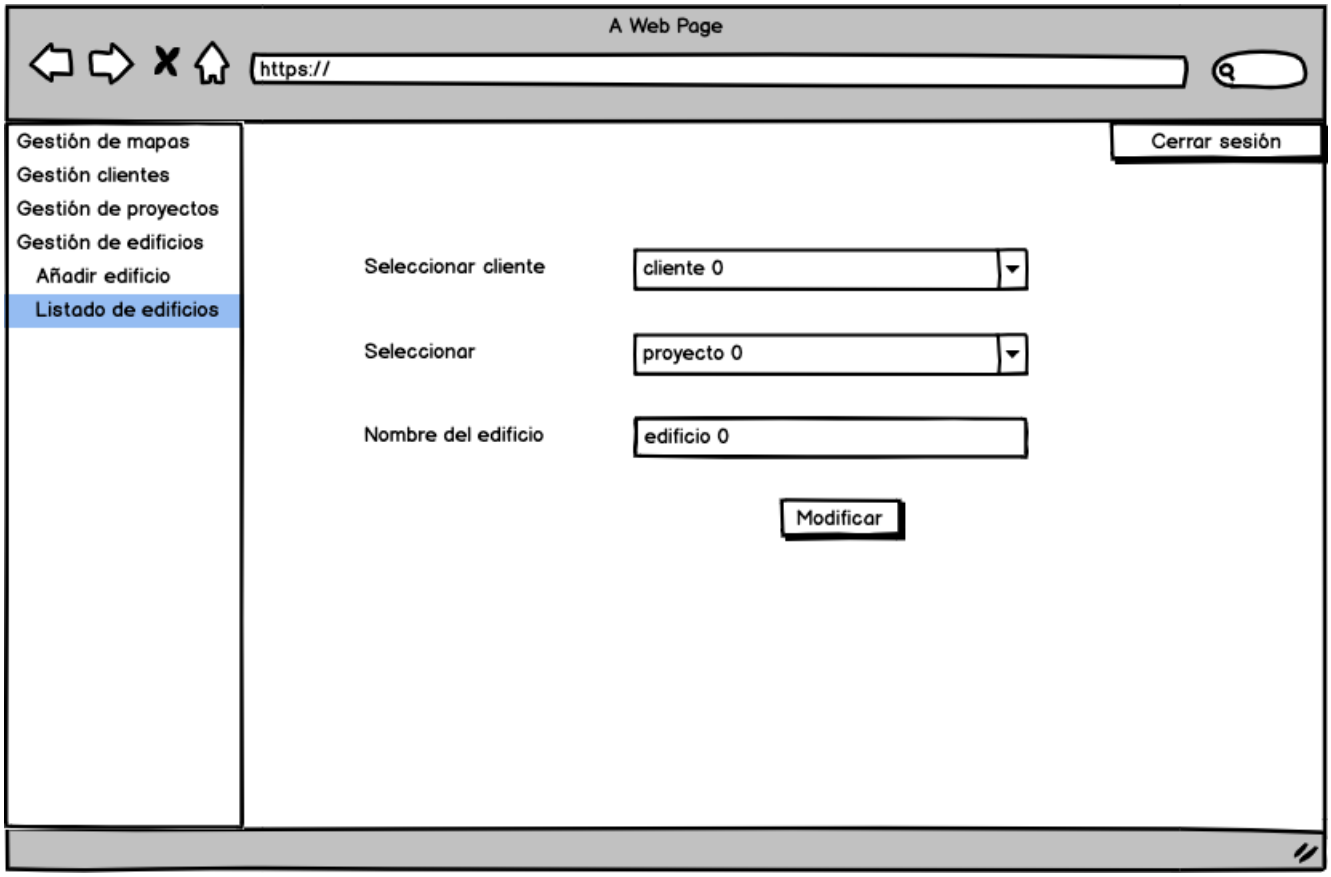

Figura 5.25: Modificar un edificio

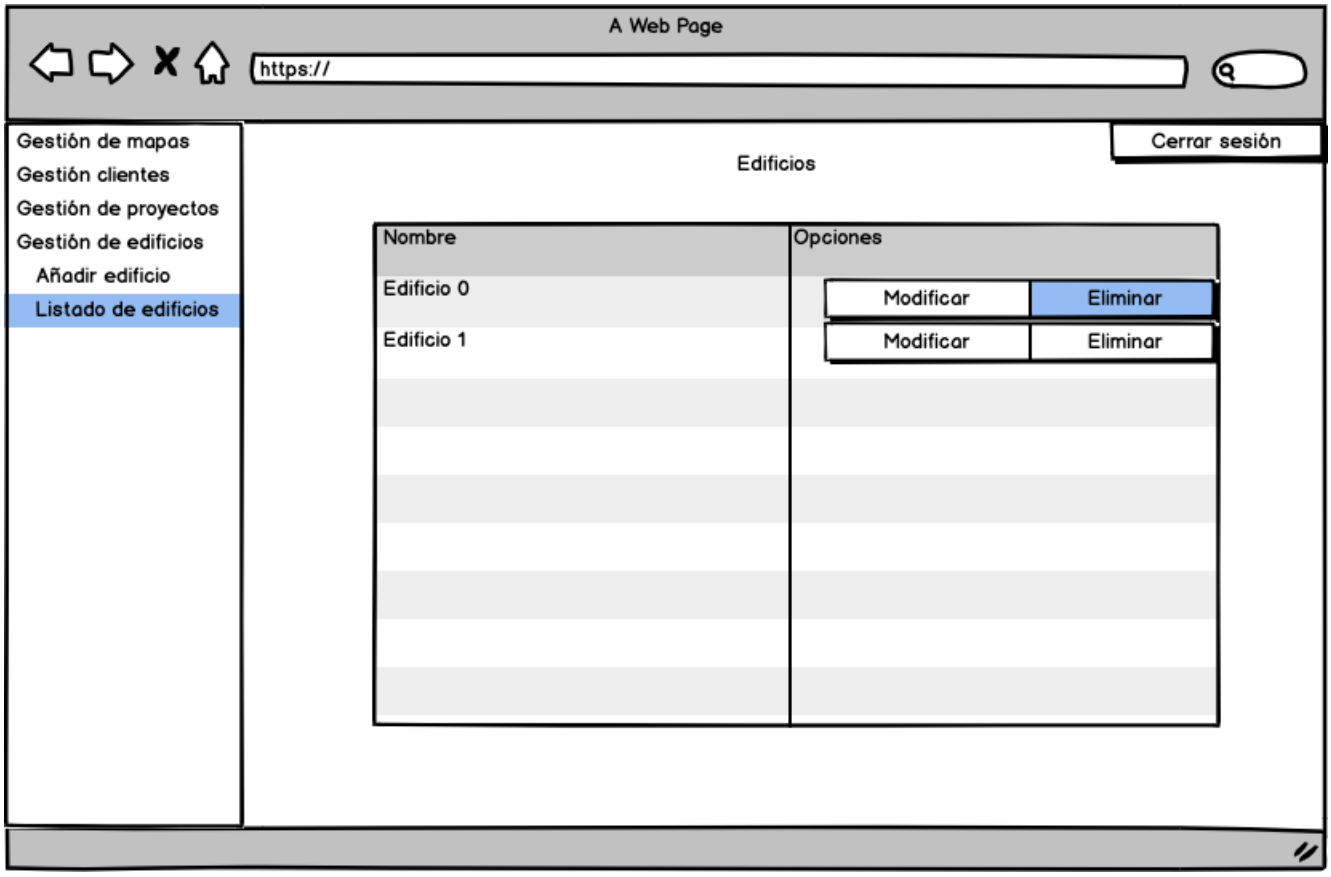

Figura 5.26: Selección de eliminar edificio

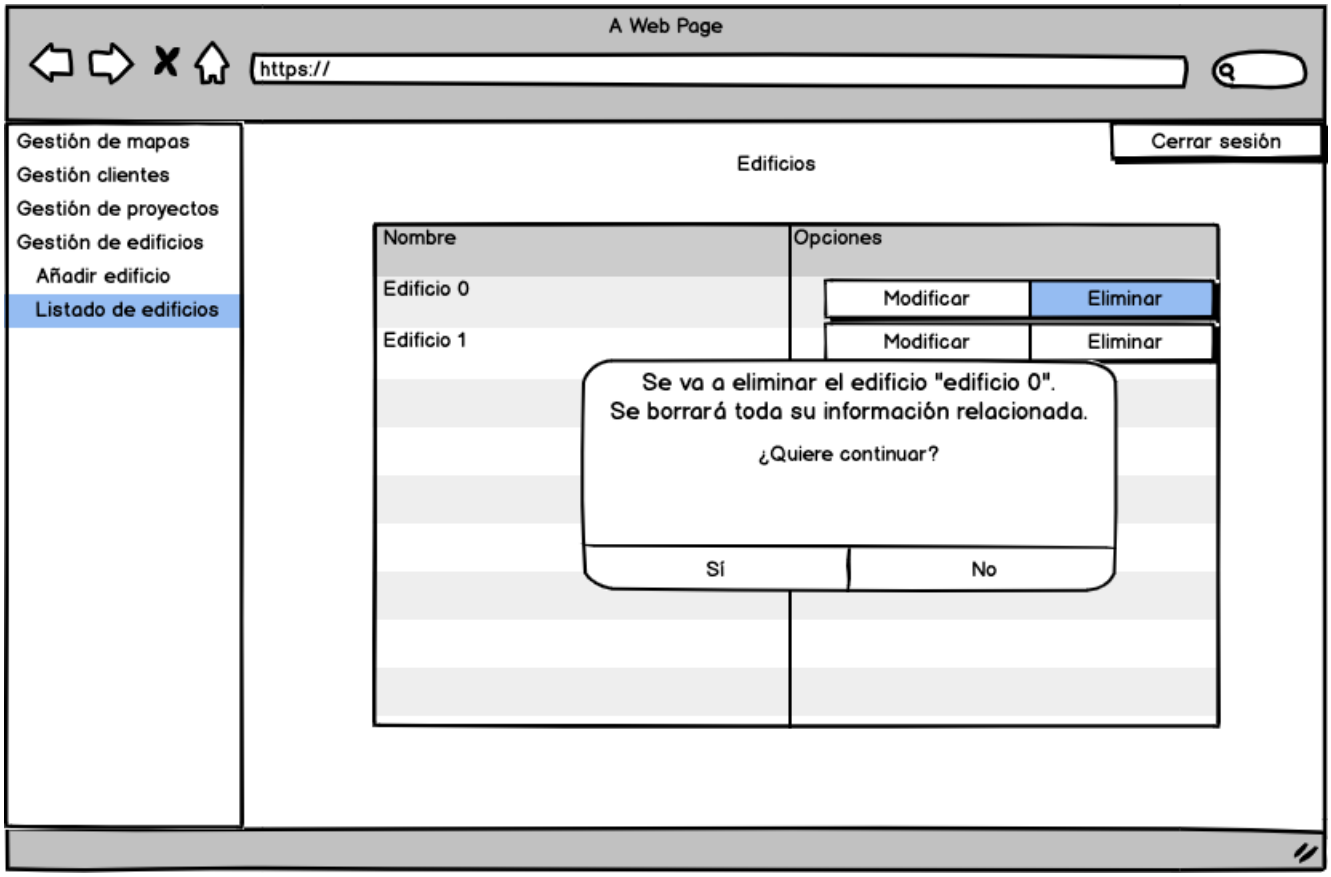

Figura 5.27: Eliminar edificio

Tras terminar este punto, cabe destacar que la interfaz final dista en ciertos aspectos de los bocetos planteados (se puede ver en el manual de usuario[11]), ya que durante el desarrollo se observó que era mejor cambiar algunos detalles, de modo que el uso de la aplicación resultase más fácil e intuitivo.

5.3. Diagrama de paths de la aplicación

Esta sección mostrará el diagrama de paths de la interfaz. Este tipo de diagrama permite ver en detalle las rutas de las que dispone el sistema, de modo que muestre desde un punto de vista a alto nivel las posibles operaciones que puede realizar el usuario.

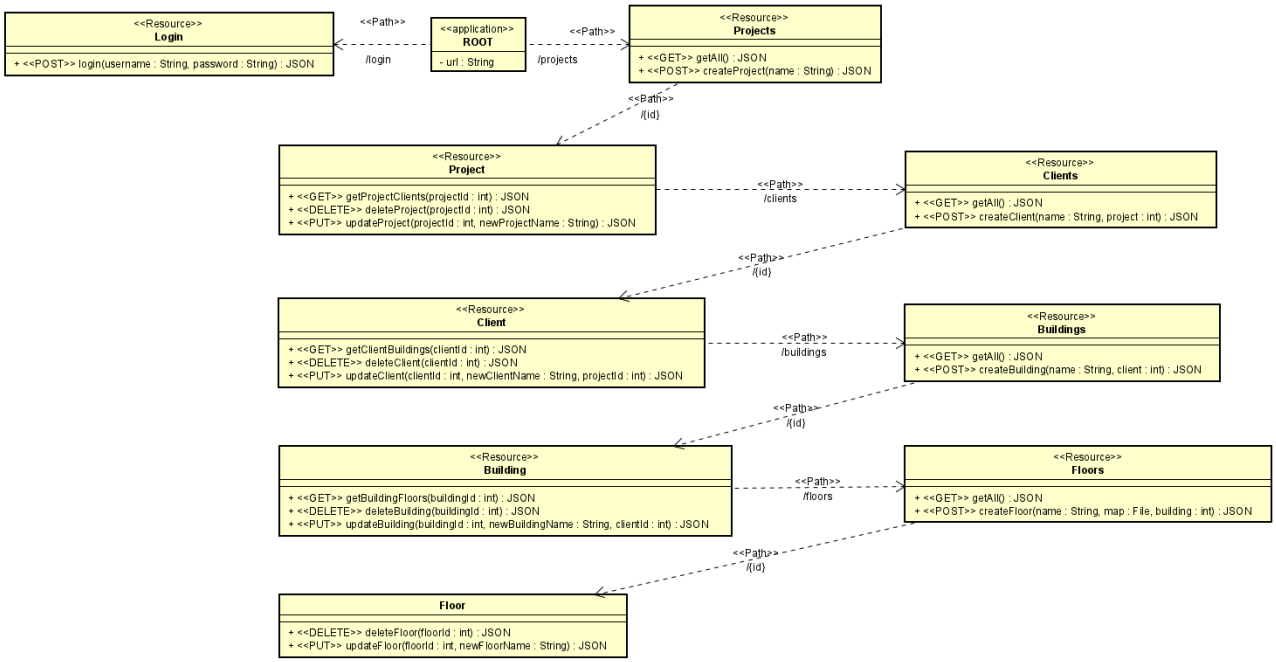

Figura 5.28: Diagrama de Paths de la aplicación

5.4. Diagramas de secuencia en diseño

En esta sección se mostrarán los diagramas de secuencia en diseño para los casos de uso que han sido considerados más relevantes de cara a la comprensión del funcionamiento del sistema. Dado que el proyecto cuenta con la parte de servidor y de cliente, el presente punto se dividirá en dos secciones, cada una correspondiente a la parte que detalle del servicio implementado.

Antes de comenzar, destacar que todos los casos de uso son muy similares entre sí, por lo que sólo se va a hacer el detalle del caso de uso "*Gestión de mapas*", que es como el resto pero con algún matiz. Todos los casos de uso tienen las mismas tareas: Visualizar el elemento del sistema, crearlo, editarlo o eliminarlo. El caso de los mapas, aunque es un poco más extenso, en esencia es lo mismo.

En algunas secuencias habrá ciertas referencias a otros subdiagramas, que por la naturaleza de la tecnología usada para gestionar las consultas a base de datos, no se implementará en detalle (se ha usado un ORM [32] para facilitar esta tarea).

A continuación se muestra el detalle de las distintas operaciones a realizar con los mapas, tanto en back-end como en front-end.

5.4.1. Diagramas de secuencia en diseño del Back-end

Para este apartado se mostrarán los diagramas de secuencia en diseño para los casos de uso que se han considerado más relevantes de la parte del servidor. Cómo ya se ha mencionado anteriormente, únicamente se tendrán en cuenta los diagramas para el caso de uso de Gestión de mapas, por la semejanza existente entre todos los casos de uso del sistema.

Se han marcado en color verde las clases pertenecientes a la capa del controlador, en azul las pertenecientes a la de servicio, en rojo las pertenecientes a la de modelo y en gris las pertenecientes a la capa de servicios comunes o *utils*.

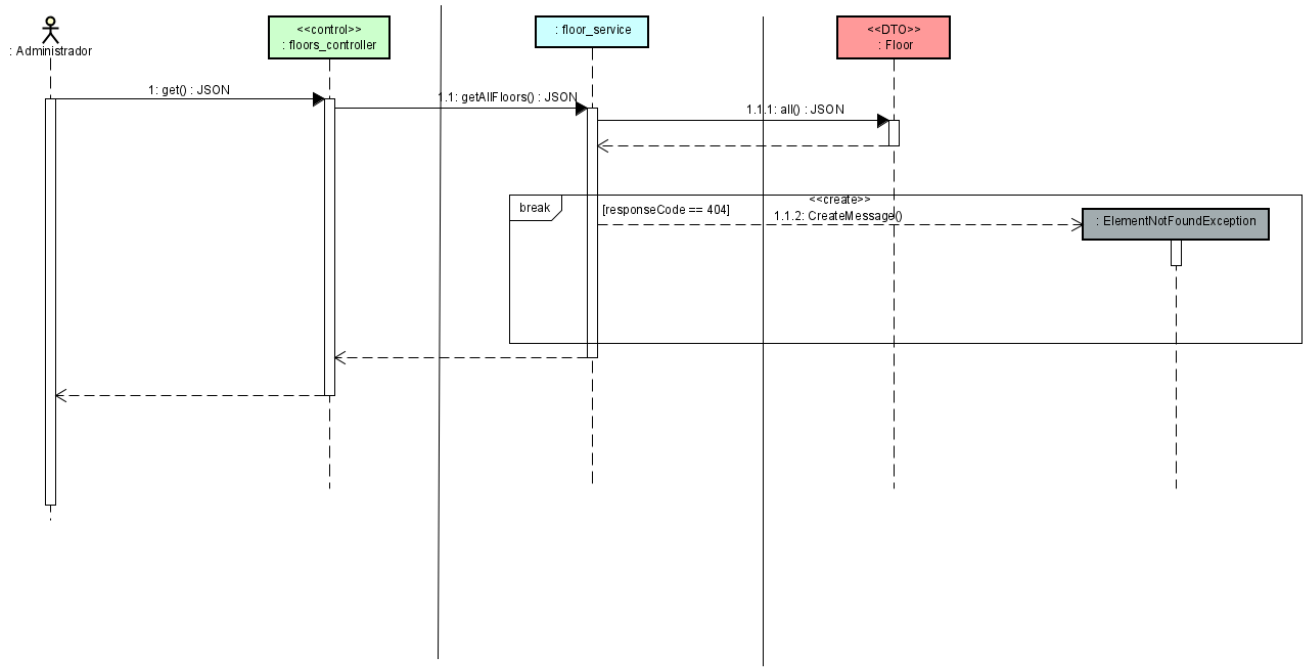

Figura 5.29: Ver las plantas de un edificio

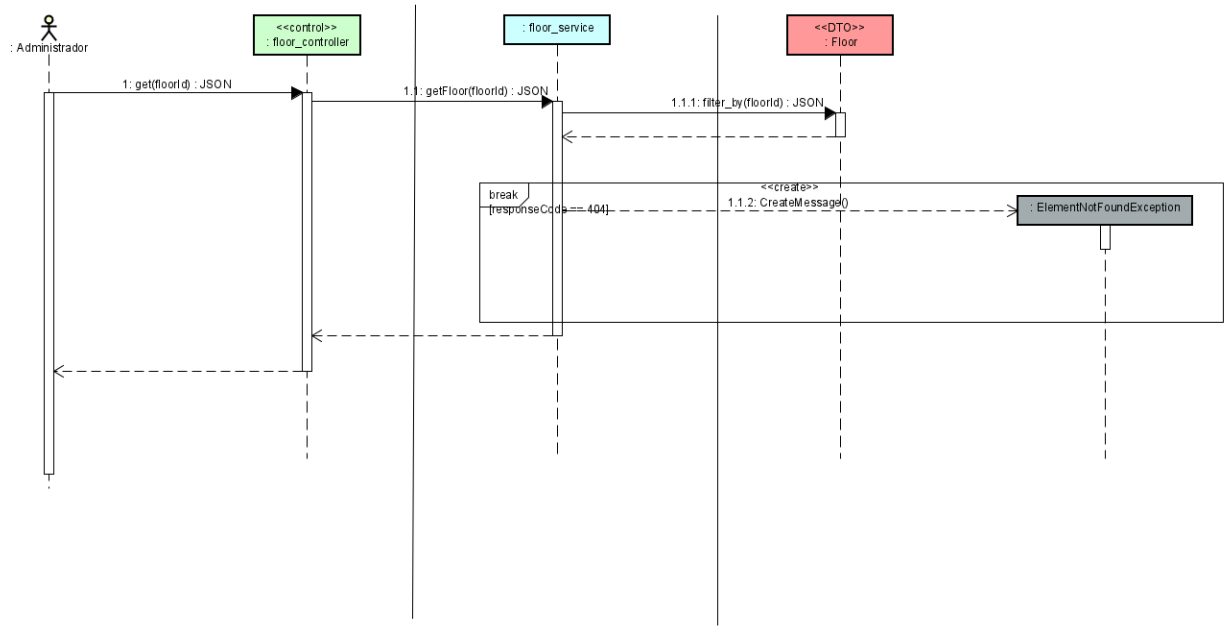

Figura 5.30: Ver una planta de un edificio

En la siguiente imagen [5.31] se puede ver la secuencia relacionada con la creación de una planta. Cómo se puede observar, hay un fragmento combinado en relación al fichero que necesita la consulta para poder completarse satisfactoriamente. En caso de no haber fichero, no se ha indicado nada, y la razón es porque se controla a nivel de interfaz, con unas reglas indicadas en el código de la aplicación.

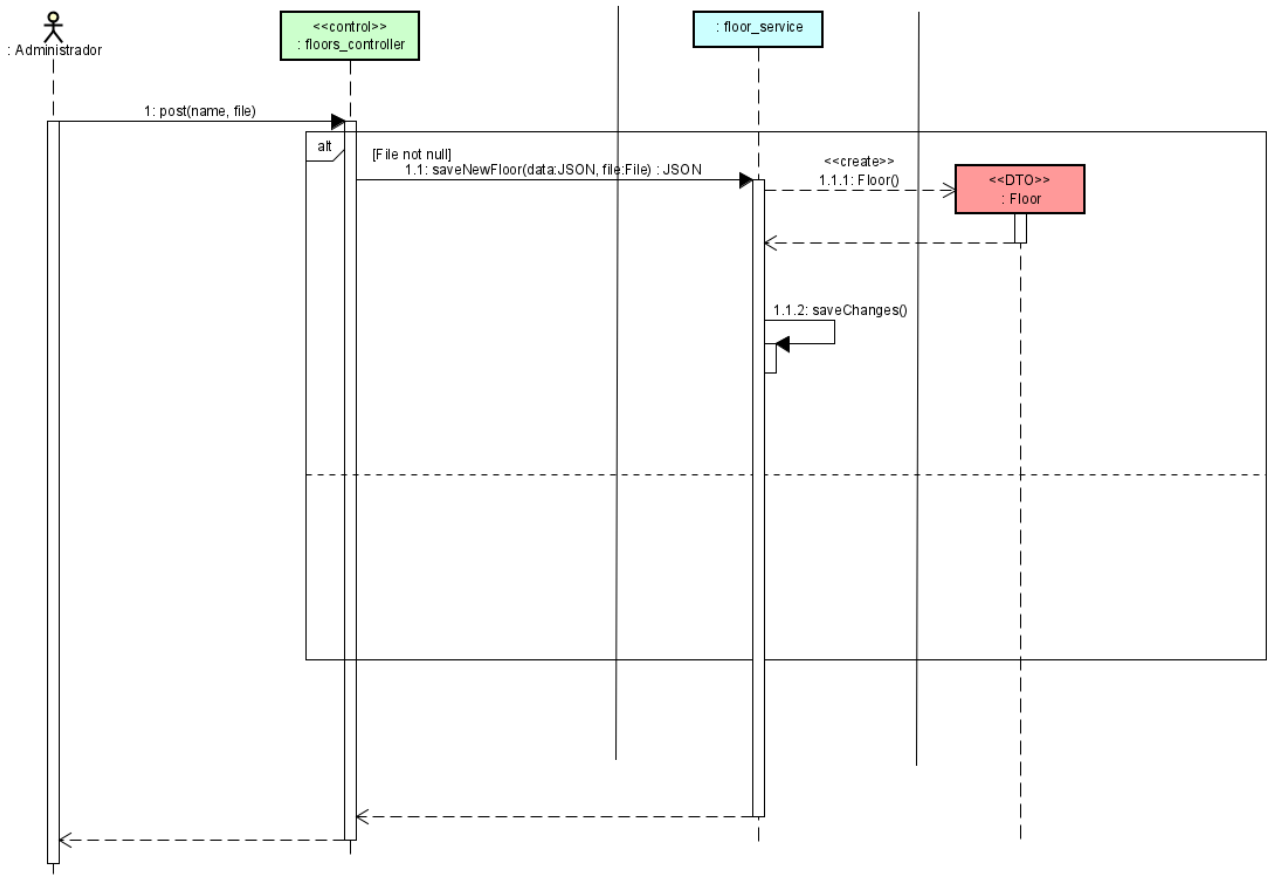

Figura 5.31: Crear una planta

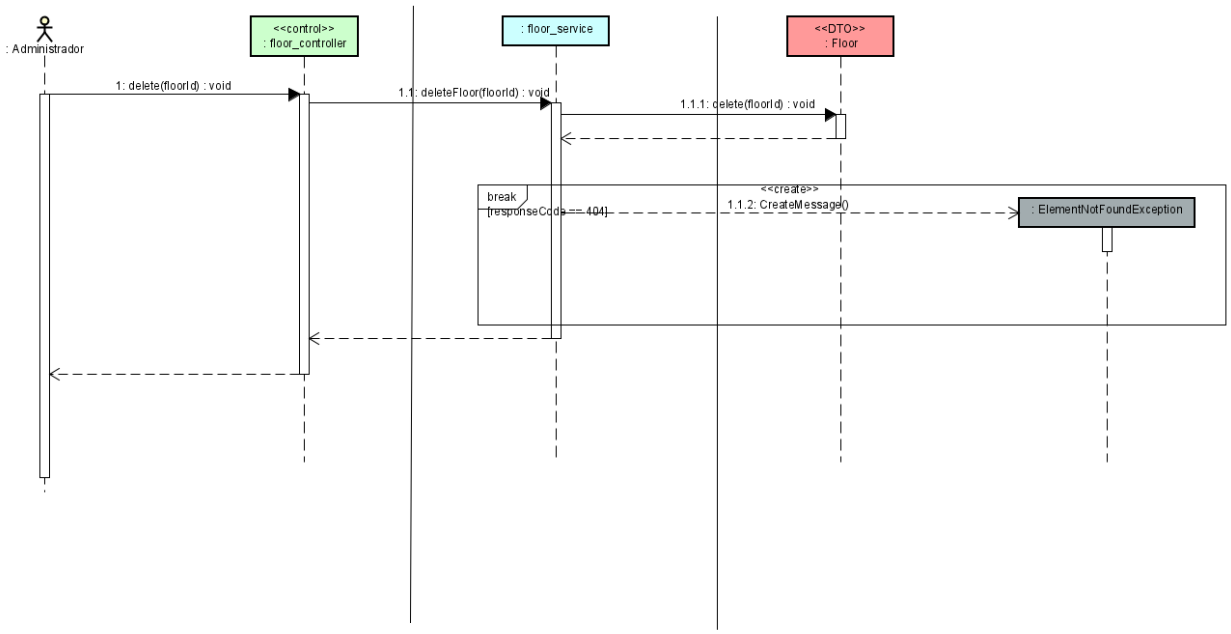

Figura 5.32: Eliminar una planta

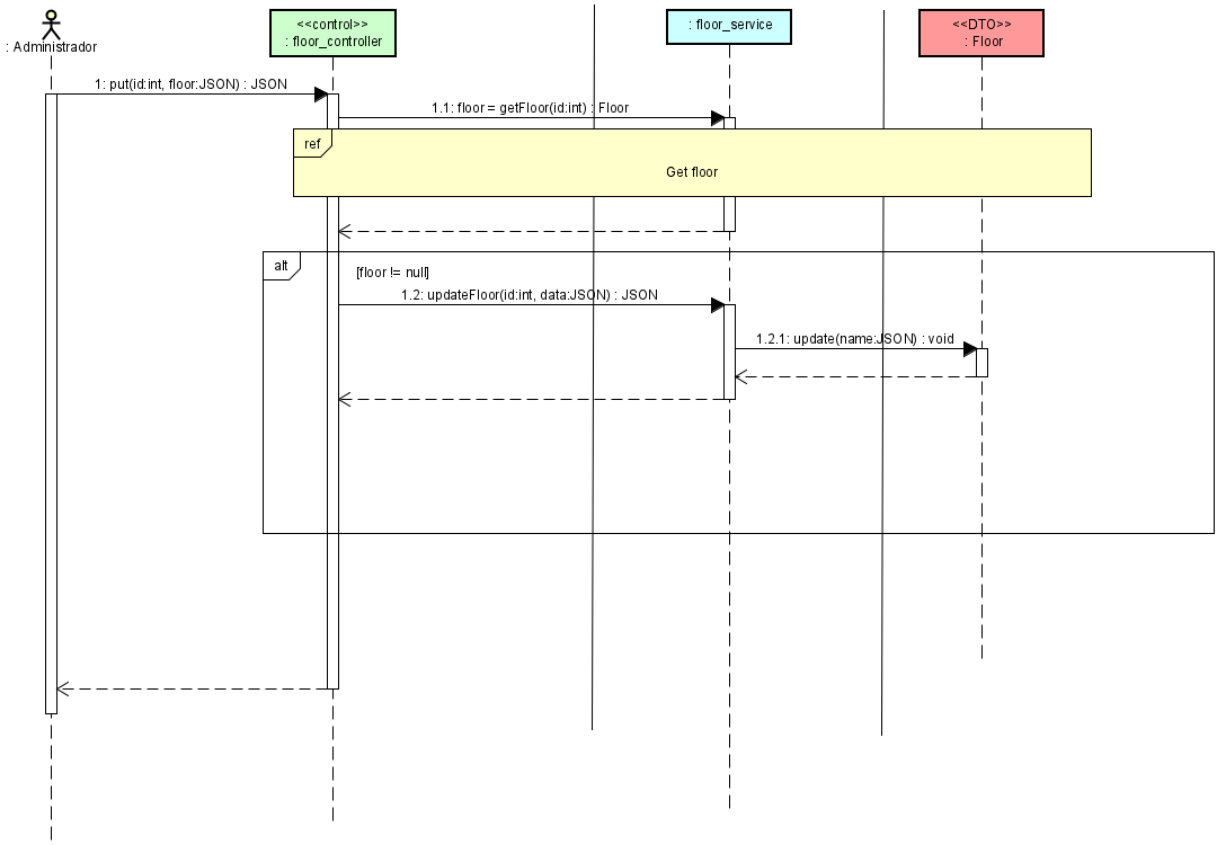

Figura 5.33: Editar una planta

5.4.2. Diagramas de secuencia en diseño del Front-end

Para este apartado se mostrarán los diagramas de secuencia en diseño para los casos de uso que se han considerado más relevantes de la parte del cliente. Cómo ya se ha mencionado anteriormente, únicamente se tendrán en cuenta los diagramas para el caso de uso de Gestión de mapas, por la semejanza existente entre todos los casos de uso del sistema.

La arquitectura que se ha elegido no opera bajo orientación a objetos, por los que la definición de las clases sigue una nomenclatura del tipo *Carpeta padre: Nombre del archivo hijo*.

Se han marcado en color naranja los componentes del sistema, en amarillo los elementos del store y en azul la parte de Axios. Se ha de tener en cuenta que los diagramas presentados no muestran un paso de mensajes entre las distintas líneas de vida dada la naturaleza del sistema desarrollado, reflejando en su lugar el flujo de comunicación entre los distintos componentes que forman parte de la aplicación.

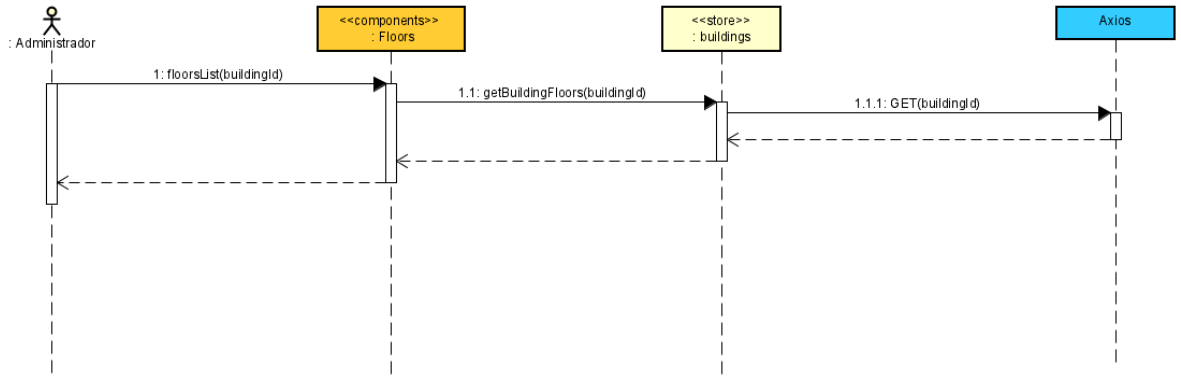

Figura 5.34: Ver plantas de un edificio

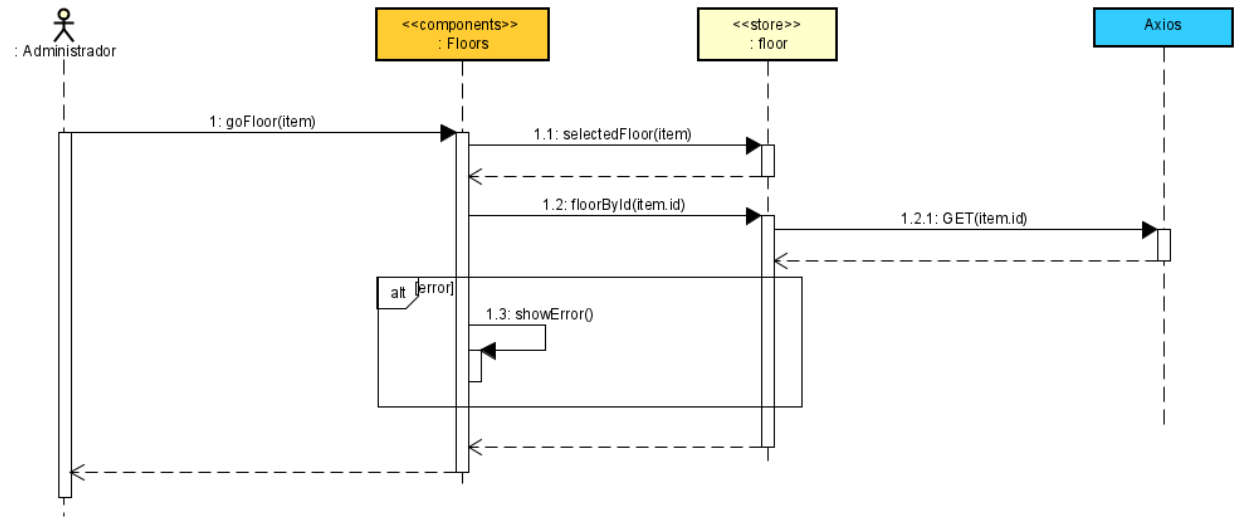

Figura 5.35: Ver una planta de un edificio

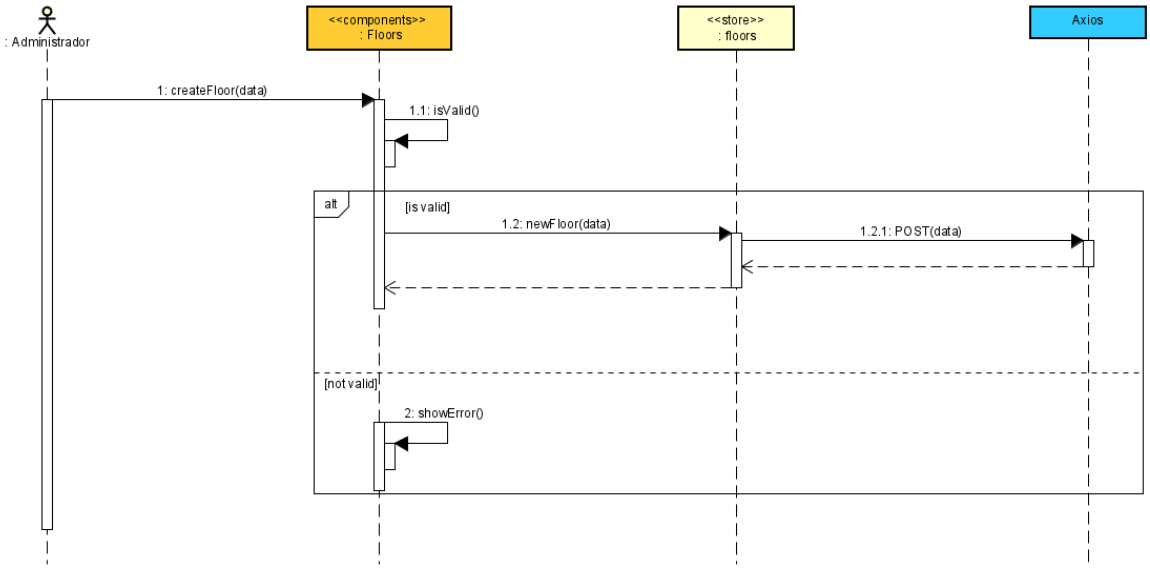

Figura 5.36: Crear una planta

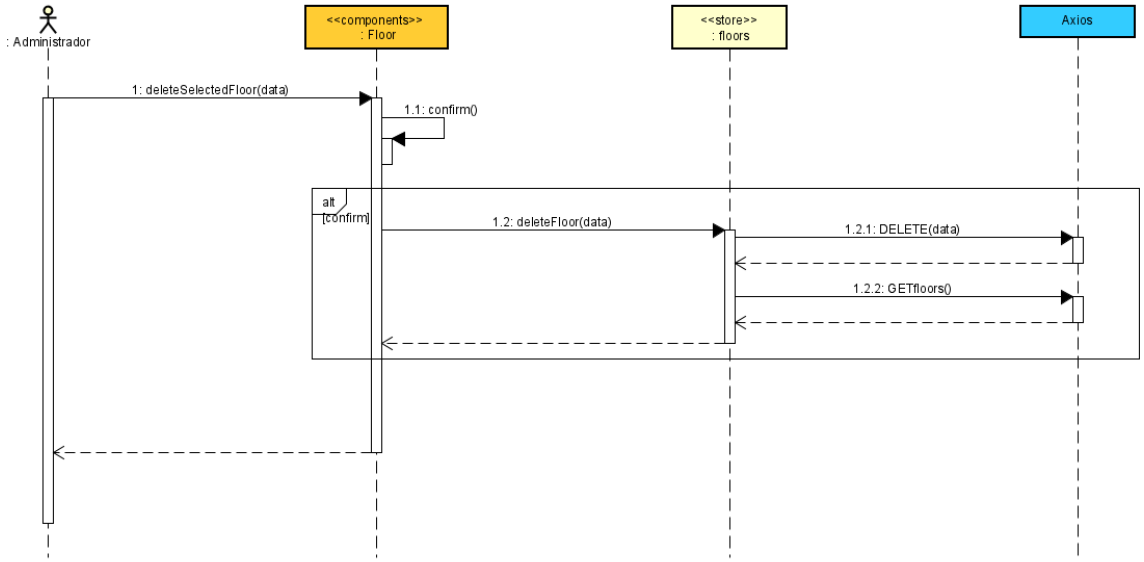

Figura 5.37: Eliminar una planta

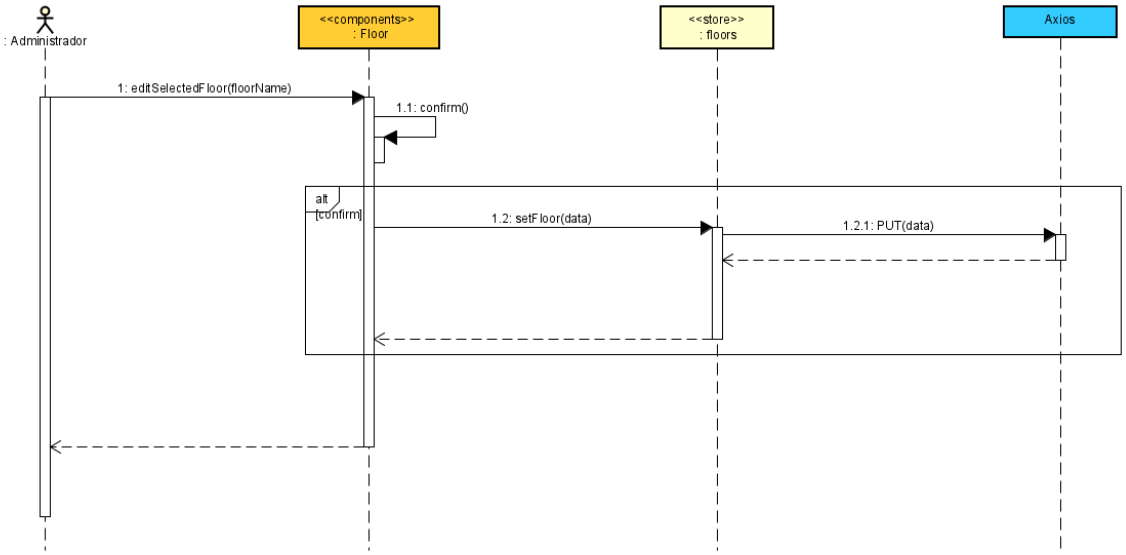

Figura 5.38: Editar una planta

Capítulo 6

Modificaciones en XtremeLoc para la utilización del nuevo sistema desarrollado

En este capítulo se hablará sobre los cambios previos realizados en el sistema, para así asegurar su correcto funcionamiento antes de comenzar con el desarrollo de las nuevas funcionalidades.

6.1. Arreglos y modificaciones realizadas en el sistema existente

Cómo ya se ha mencionado anteriormente a lo largo del presente TFG, el sistema a desarrollar será usado por cualquier aplicación en producción que requiera información cartográfica basada en OpenStreetMaps, siendo ahora la única que aprovecha esta característica la aplicación *XtremeLoc*.

El sistema XtremeLoc cuenta con dos apartados en cuanto a mapas se refiere: Los mapas outdoor y los mapas indoor, que a continuación se explicarán ambos de modo resumido en que consisten y qué los diferencia, para así poder asentar las bases del trabajo que se ha llevado a cabo.

- **Mapas outdoor**: Estos mapas se caracterizan por ser los que muestran al usuario zonas geográficas amplias como ciudades, provincias, etc. En el contexto de XtremeLoc se precisan estos mapas por su amplitud geográfica, que aún siendo un sistema de localización en interiores, se necesita tener un control de en qué zonas está desplegado, de modo que esa información facilite las tareas de administración.

- **Mapas indoor**: Estos mapas se caracterizan por ser planos concretos de explotaciones localizadas en los mapas outdoor. Simbolizan una planta concreta de una explotación (edificio) que tiene implantado el sistema XtremeLoc.

Junto a estas diferencias, se debe indicar también que se cuentan con dos elementos desplegados: el servidor de mapas y la base de datos postgis (base de datos PostgreSQL con el añadido de ciertas características para poder almacenar información espacial).

Nuestro contexto de trabajo serán los mapas indoor, que requieren un despliegue de una versión nueva del servidor que actualmente se encuentra en producción.

6.1.1. Servidor indoor

El trabajo a realizar era el despliegue de una versión nueva del servidor de mapas indoor, con el objetivo de hacerlo más óptimo y actualizarlo, respecto al que estaba ya en producción. La razón de querer realizar el despliegue mediante Docker era para que el servidor fuese escalable, de modo que su mantenimiento y actualización requiriese menos esfuerzo en un futuro.

Versión 2

Antes de comenzar el trabajo en esta línea, ya se había intentado realizar la actualización del servidor, consiguiendo únicamente actualizar el servidor de mapas outdoor. Por tanto, se debía leer y entender toda la documentación con la que se contaba, tanto del servidor outdoor como del servidor indoor. Ambos servidores se encontraban preparados para ser desplegados en Docker, por lo que también era necesario entender toda esta tecnología y los conceptos que la acompañan.

Una vez conseguidas las bases necesarias para poder comenzar con el trabajo, se observaron varios problemas en el servidor al intentar desplegarlo:

- Al realizar el despliegue en local, hacía que la máquina de trabajo fuese considerablemente lenta.
- El nivel de zoom generado fallaba para niveles muy cercanos (a partir del nivel de zoom 7).
- Levantando el servicio mediante docker compose, el contenedor se reiniciaba de forma continua.

Se llevó a cabo la compilación de varias imágenes de Docker, para poder dar con el fallo que surgía al desplegar el servicio, aunque sin éxito. Hay que destacar que el despliegue de todo el sistema se realizaba mediante un script que también generaba errores.

Tras revisar dicho script, se realizaron los cambios pertinentes que consiguieron realizar un despliegue óptimo en local, sin conseguir arreglar el fallo de la generación de los tiles a partir del zoom de nivel 7.

6.1.2. Script de generación de tiles

Anteriormente se contaba con un Script desarrollado en Python, que tomaba un imagen como entrada y realizaba de forma manual la generación de planos, descargándose una librería [20] para este lenguaje, la cuál se encargaba de toda la tarea de procesado de la imagen. Los arreglos que se han llevado a cabo son los siguientes:

- Sustitución de la descarga por código de la librería de procesado de imagen, usando un módulo [18] para Python3 que incluye las funciones que se necesitan para hacerlo de una forma más limpia y eficiente.

6.2. Trabajo conseguido en el servidor de mapas

Aunque ya se ha mencionado que no se consiguió finalizar de forma satisfactoria la actualización y despliegue del servidor de mapas, se entendió bastante en el proceso, entendimiento que ayudaría al resto del conjunto del proyecto desarrollado. Se consiguió lo siguiente:

- Arreglar el servidor para poder desplegar mediante docker-compose.
- Arreglar el script de configuración del sistema de modo que no diera fallos al realizar el despliegue.

Todo el proceso realizado ayudó a entender en más o menos profundidad la tecnología Docker, ya que se tuvo que realizar la compilación y despliegue de muchas imágenes y contenedores en esta tecnología, la revisión de sus logs para investigar el por qué de los fallos que surgían, etc.

6.3. Nuevas características implementadas en la generación de tiles

1. Función de centrado de imagen. Esto se necesita de modo que en el posterior procesado de la misma, las partes que se van a usar para renderizar los planos aparezcan centradas correctamente.

2. Función de reescalado de la imagen, de modo que la lógica implementada permita realizar una imagen cuadrada a partir de la imagen de entrada. Esto se debe a la teoría que usa OpenStreetMaps para su renderizado [26], ya que para un zoom de nivel n, la imagen se divide en 4^n franjas (o tiles de aquí en adelante) cuadradas.
   Esto se realizada usando una combinación de la función anteriormente mencionada junto con esta, de modo que:

   a) La imagen de entrada cuenta con una anchura x y una altura y.

   b) Se obtiene el máximo de esas dos medidas.

   c) Con el máximo de esas dos medidas se genera un fondo cuadrado transparente (para que la imagen se mantenga en un formato correcto).

   d) La imagen se "pinta"por encima de ese fondo, centrada, de modo que al comenzar el procesado de la misma, la lógica del programa cuente con una imagen cuadrada y centrada.

Capítulo 7

Implementación

En este apartado se detallarán los aspectos prácticos del proceso de desarrollo que han sido considerados más relevantes. Todo el código desarrollado para el proyecto se encuentra en el siguiente enlace, tanto la parte del servidor cómo la de la interfaz web, así cómo los ficheros de despliegue e integración continua: `https://www.infor.uva.es/~diego/tfg/ivan-gonzalez-sep-20.tgz`

7.1. Diagrama de despliegue

En este diagrama se puede ver cómo está deplegado el sistema y la integración con la parte del servidor de mapas que ya se encontraba en producción (en color naranja).

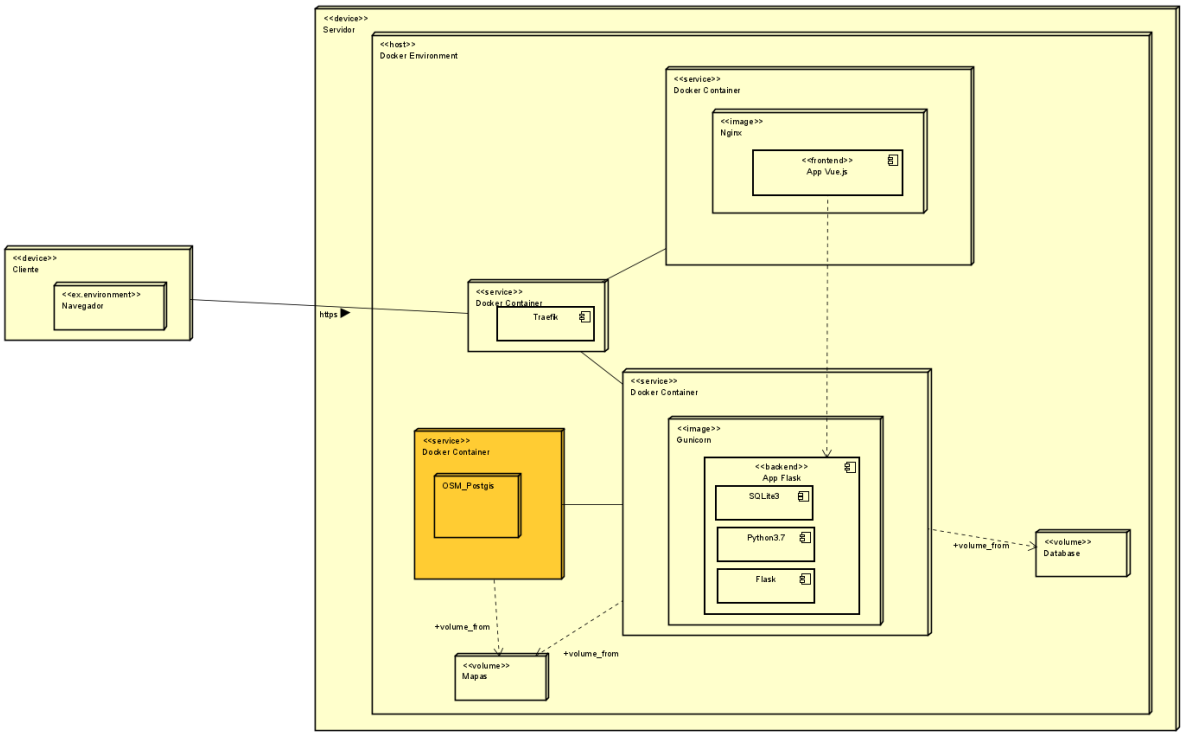

Figura 7.1: Diagrama de Despliegue de la aplicación

7.2. Vista lógica de la implementación

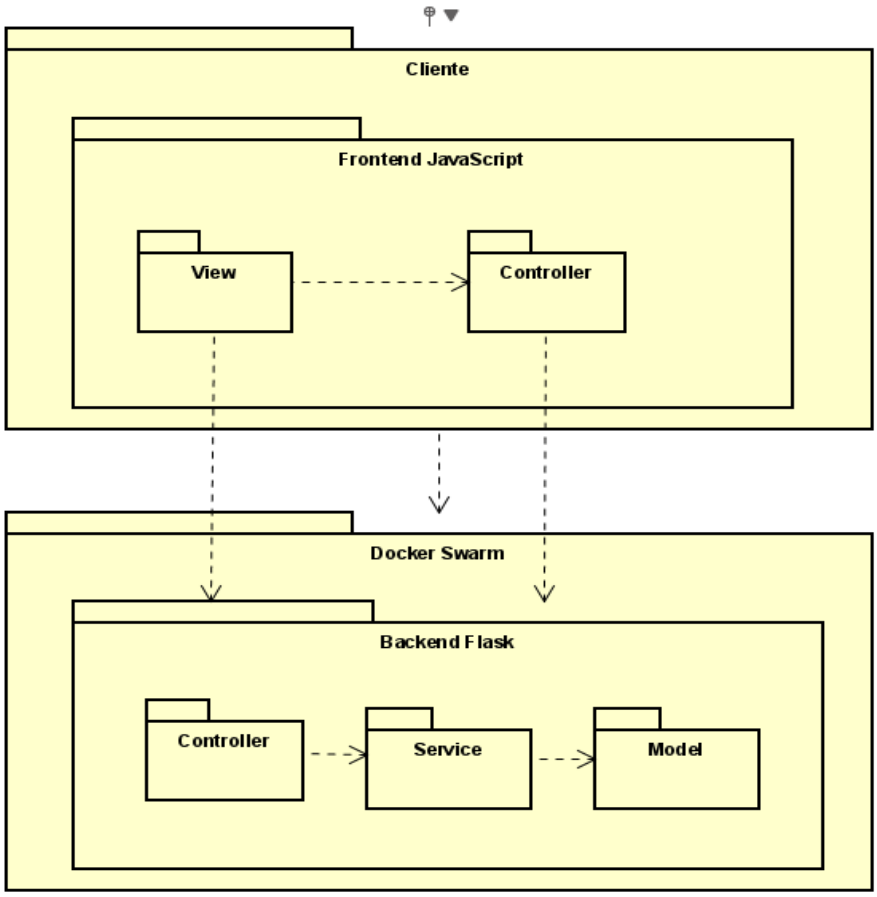

Figura 7.2: Vista lógica de la aplicación

7.3. Entorno de desarrollo

Para el desarrollo tanto del frontend como del backend se ha usado Visual Studio Code, un editor de texto gratuito desarrollado por Microsoft, que permite añadir extensiones al gusto del programador, para poder personalizar el entorno en función de la tarea que vaya a realizarse con el mismo. Las extensiones usadas han sido algunas relacionadas con JavaScript, Vue.js y alguna más para Flask y Python.

7.4. Versiones necesarias de software

Al tratarse de una aplicación web, sólo será necesario un equipo con conexión a Internet y uno de los siguientes navegadores:

- Google Chrome.
- Mozilla Firefox.
- Microsoft Edge.
- Safari.
- Opera.

7.5. Herramientas utilizadas

A lo largo del desarrollo del sistema se han utilizado diversas herramientas, que se dividirán en la parte correspondiente a la interfaz web (front-end) y la parte del servidor (back-end). Las más importantes son las indicadas a continuación.

7.5.1. Front-end

- Vue.js [23] es un framework progresivo para el desarrollo de interfaces de usuario. Se diferencia de otros frameworks monolíticos en que está diseñado desde abajo para que resulte escalable. Se ha empleado en el desarrollo de todo el apartado de la interfaz de gráfica de usuario y en el control de la misma.
- Node.js [16] es un entorno de ejecución multiplataforma y de código abierto basado en JavaScript que opera de forma asíncrona, basado en el motor V8 de Google [22]. Se ha usado como herramienta para ejecutar Vue.js, ya que es un entorno eficiente, ligero y escalable.
- Axios [34] es un cliente HTTP basado en promesas de JavaScript, destinado para ser usado en navegadores web y con node.js. Ha sido empleado como gestor para realizar las peticiones al API REST desarrollado.
- Vuetify [24] es un framework progresivo de componentes para Vue.js basado en Material Design [14]. Se ha elegido por dos razones:
  - La primera es que, al ser un framework soportado por Google, se asegura la continuidad y soporte del diseño de la aplicación.
  - La segunda es que, como la interfaz se ha desarrollado como parte de un conjunto de herramientas ya en producción, elaboradas con este mismo framework, se ha querido dar una sensación de unidad usando la misma base.
- Leaflet [13] es una librería JavaScript de código abierto para mapas interactivos basados en OpenStreetMaps. Se ha usado para poder mostrar en la interfaz web los planos de los edificios una vez procesados, de modo que se pueda interactuar con ellos.

7.5.2. Back-end

- Flask [17] es un framework para aplicaciones web tipo WSGI [27] ligero. Está destinado a poder comenzar con el desarrollo de forma rápida y sencilla, con la capacidad de escalar hacia aplicaciones más complejas.
- SQLAlchemy [32] es un conjunto de herramientas de código abierto basadas en SQL para Python, usada para facilitar el acceso a bases de datos relacionales, permitiendo aligerar la carga que el programador incluye en su código para poder realizar operaciones con bases de dato de este tipo.
- Gdal2tiles [20] es una herramienta de procesado que permite la generación de Tiles para ser usados en sistemas de despliegue de mapas cartográficos. La versión que usada será el módulo de Python [18].
- SQLite3 [33] es una base de datos relacional compacta y portable, que facilita la implementación de este tipo de servicios en sistemas que no requieren una base de datos muy potente, por la naturaleza del servicio que se va a realizar o por el poco volumen de datos que se va a manejar. Su uso hace que sea sencillo migrar un sistema basado en esta herramienta a versiones que usen otro tipo de bases de datos, como por ejemplo MySQL.

7.5.3. Despliegue

Tanto para el caso del Front-end como el del Back-end, se han usado las herramientas que se citan a continuación.

- Docker [29] es un proyecto de código abierto que permite automatizar el despliegue de aplicaciones dentro de contenedores software, proporcionando una capa adicional de abstracción y automatización de aplicaciones en múltiples sistemas operativos. De esta manera, se consigue que un sistema pueda ser portable y fácil de desplegar en cualquier sistema que cuente con esta tecnología instalada, consiguiendo un despliegue sólido y universal sin grandes requerimientos hardware.
- Traefik [9] es un proxy que se usa para llevar a cabo la configuración de conexiones https.

7.6. Problemas encontrados y soluciones aportadas

En esta sección se hablará sobre los distintos problemas que han ido surgiendo a lo largo del desarrollo del proyecto, así como de las soluciones que se han aportaron para cada uno de ellos. La forma de exponerlo será mediante una pequeña introducción al desarrollo de la parte correspondiente de la aplicación, para posteriormente enumerar por pares de Problema-Solución todo lo que se considere de interés para este punto.

7.6.1. Back-end

El desarrollo de la parte del servidor del servicio que se ha llevado a cabo tiene dos pilares fundamentales:

- El procesamiento de imágenes usando el estándar de OpenStreetMaps, basado en gdal [11], con el que se generan los tiles de los distintos niveles de zoom disponibles.
- El desarrollo de la tecnología del back-end propiamente dicho.

Los problemas más destacables que han surgido durante el desarrollo de esta parte han sido los siguientes:

- **Problema**: Gestión de dependencias necesarias para poder desarrollar los módulos del backend encargados del procesamiento de imágenes. Dado el volumen de requerimientos necesarios para poder usar las funciones de GDAL y sus elementos relacionados, era preciso disponer de ellas de forma local.
- Solución: Instalación en Linux [11] y posterior creación de un contenedor Docker, para poder probar las funcionalidades del servicio una vez desplegado.
- **Problema**: Gestión de la política CORS. Esta política define la forma en la que el cliente y el servidor pueden interactuar, y de ese modo, cómo el primero puede solicitar recursos al segundo. El problema surgió en que al comienzo del desarrollo del front-end y sin tener en cuenta esto, el back-end bloqueaba algunas peticiones realizadas desde el cliente.
- Solución: En primera instancia y únicamente para realizar las pruebas previo al despliegue del sistema, hay 2 opciones:
    1. Desactivar la política en el navegador desde el que se realizan las pruebas.
    2. Instalar una extensión (para Chrome en este caso) que incluya las cabeceras necesarias para el servidor.

En cualquiera de los dos casos, una vez desplegado el sistema en su totalidad este problema desaparece, por lo que hay que tener en cuenta estas soluciones en la fase de desarrollo y de realización de pruebas del sistema.

- **Problema**: Integración del back-end desarrollado con el sistema que se encuentra en producción. Dada la naturaleza del proyecto desarrollado, era imperativo que se pudiese incluir cómo una funcionalidad del sistema ya desplegado, por lo que debería trabajar en armonía con volúmenes Docker que ya estaban en uso. Al incluir esto, el contenedor de la imagen asociada a la implementación realizada se desplegaba erróneamente, provocando que al rehacer la recompilación de la imagen no se actualizase con los cambios que se iban introduciendo.

- Solución: El problema venía de intentar montar un volumen en un fichero, concretamente el volumen para mantener los datos de la base de datos de forma persistente, de modo que al reiniciar el contenedor, la base de datos quedase intacta. Esto se solucionó copiando el fichero de la base de datos de dentro del contenedor y enlazándolo posteriormente con el que se encontraba dentro del contenedor, de modo que los datos persistieran al volver a desplegar el sistema.

7.6.2. Front-end

El desarrollo de la parte de la interfaz web se centra en la implementación de una herramienta de administración que aproveche el servicio implementado de procesamiento de imágenes, de manera que resulte sencillo para un administrador del sistema llevar a cabo sus tareas con esta herramienta.

Los problemas más destacables que han surgido durante el desarrollo de esta parte han sido los siguientes:

- **Problema**: Despliegue en Docker de la herramienta. Al llegar el momento de realizar los ficheros de configuración necesarios para poder levantar la aplicación, esta se podía desplegar en producción de forma correcta en local, pero a la hora de compilar la imagen con la configuración indicada, la compilación fallaba.

- Solución: Inclusión de dependencias de npm que faltaban a nivel de proyecto para poder desplegar correctamente.

7.7. Control de versiones

Como control de versiones se ha usado git [12], concretamente dentro de la página de Gitlab [2], usando un repositorio privado para parte del proyecto.

La forma de trabajo llevada a cabo ha sido la descrita por Gitflow [28], en el que se definen 3 ramas principales: master, develop y feature.

1. Rama master: Contiene las versiones en producción de los distintos servicios.

2. Rama develop: Contiene la integración de las nuevas funcionalidades implementadas en los distintos servicios hasta que se publica la siguiente versión.

3. Rama feature: Contiene el código relacionado con las nuevas funcionalidades en desarrollo del sistema.

Cabe destacar que se ha aprovechado la posibilidad con la que cuenta Gitlab de llevar a cabo Integración Continua [3] para llevar a cabo el despliegue de los servicios que conforman el proyecto(se explicará en detalle en el manual de instalación[10])

7.8. Servidor de la aplicación

En este apartado del punto de implementación se separará la parte del servidor para el Front-end y para el Back-end.

7.8.1. Back-end

La parte del API REST del servicio se encuentra desplegada en un contenedor docker que contiene todas las dependencias necesarias, usando traefik [9] para poder crear una conexión entre el contenedor y el exterior, de modo que sea accesible.

7.8.2. Front-end

La parte de la aplicación web, igual que el servidor, se encuentra alojado en un contenedor docker, usando nginx [30] como servidor web y de nuevo traefik como proxy inverso de alto rendimiento.

Tras haber visto el proceso de implementación del proyecto pasaremos a las pruebas a realizar sobre los casos de uso extraídos en análisis, apara asegurar que todo se ha desarrollado satisfactoriamente.

Capítulo 8

Plan de pruebas y evaluación

En este apartado se detallarán las pruebas a realizar por cada apartado implementado en la aplicación, para comprobar que funcionan correctamente y cumplen con el objetivo planteado.

Se llevarán a cabo pruebas unitarias que comprueben meticulosamente los requisitos recogidos en el capítulo 2, de forma que a través de ellas se verifique tanto que el servidor como la interfaz web desarrollados permiten efectuar todas las funcionalidades satisfactoriamente.

Antes de comenzar en detalle con las pruebas, indicar que contarán con los pasos necesarios para poder ser realizadas correctamente, es decir:

- Para el back-end, la URL correspondiente, los endpoints a los que realizar las peticiones y la configuración concreta de cada petición.
- Para el front-end, el apartado concreto de la aplicación y la información necesaria que requiere el sistema.

8.1. Pruebas para el back-end

En este punto se detallarán las pruebas correspondientes a la parte del servidor del servicio desarrollado, con todas las operaciones CRUD necesarias para comprobar el correcto funcionamiento del API REST. Las pruebas se realizarán en Postman [31], una aplicación que sirve para poder realizar peticiones al servidor una vez desplegado. Basta con conocer su URL y formar la petición necesaria rellenando una serie de campos.

La URL base a la que realizar las peticiones es la siguiente: **https://maps.tiler.rdnest.com/**. A esta URL habrá que añadir el endpoint correspondiente al que realizar la petición con **/nombreEndpoint**.

8.1.1. Pruebas sobre gestión de proyectos

En este apartado se realizarán las pruebas correspondientes al correcto funcionamiento sobre la gestión de proyectos del sistema, que incluyen su creación eliminación, modificación y listado, así como el listado de clientes asociado a un proyecto dado (requisitos RF-09, RF-10, RF-11, RF-21, RF-22).

Prueba 1.1 (RF-09)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la creación de un proyecto.
Endpoint	/projects
Tipo de petición	POST
Parámetros del Body	name
Acción	Añadir un nuevo proyecto indicando su nombre.
Resultado Esperado	El servidor contesta con un 201, indicando que se ha creado correctamente.

Tabla 8.1: Prueba 1.1

Prueba 1.2 (RF-21)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento del listado de los proyectos.
Endpoint	/projects
Tipo de petición	GET
Parámetros del Body	-
Acción	Listar los proyectos existentes en el sistema.
Resultado Esperado	El servidor debe responder devolviendo un JSON que contenga todos los proyectos existentes.

Tabla 8.2: Prueba 1.2

Prueba 1.3 (RF-10)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la eliminación de un proyecto.
Endpoint	/projects/id
Tipo de petición	DELETE
Parámetros del Body	-
Acción	Eliminar un proyecto concreto indicando su id en la petición.
Resultado Esperado	El servidor debe responder con un JSON vacío, indicando que se ha eliminado correctamente.

Tabla 8.3: Prueba 1.3

Prueba 1.4 (RF-11)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la modificación de un proyecto.
Endpoint	/projects/id
Tipo de petición	PUT
Parámetros del Body	name
Acción	Modificar el nombre de un proyecto, indicando su id en la petición y pasando un nuevo nombre como parámetro del body.
Resultado Esperado	El servidor contesta con un 200, indicando que se ha modificado correctamente.

Tabla 8.4: Prueba 1.4

Prueba 1.5 (RF-22)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de listar los clientes de un proyecto.
Endpoint	/projects/id
Tipo de petición	GET
Parámetros del Body	-
Acción	Listar los clientes asociados a un proyecto.
Resultado Esperado	El servidor debe responder devolviendo un JSON que contenga los clientes asociados al proyecto con el id indicado.

Tabla 8.5: Prueba 1.5

8.1.2. Pruebas sobre gestión de clientes

En este apartado se realizarán las pruebas correspondientes al correcto funcionamiento sobre la gestión de clientes del sistema, que incluyen su creación, eliminación, modificación, así como el listado de sus edificios asociados(requisitos RF-12, RF-13, RF-14, RF-23)

Prueba 2.1 (RF-12)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la creación de un cliente.
Endpoint	/clients
Tipo de petición	POST
Parámetros del Body	name, project
Acción	Añadir un nuevo cliente indicando su nombre y el proyecto al que está asociado.
Resultado Esperado	El servidor contesta con un 201, indicando que se ha creado correctamente.

Tabla 8.6: Prueba 2.1

Prueba 2.2 (RF-13)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la eliminación de un cliente.
Endpoint	/clients/id
Tipo de petición	DELETE
Parámetros del Body	-
Acción	Eliminar un cliente concreto indicando su id en la petición.
Resultado Esperado	El servidor debe responder con un JSON vacío, indicando que se ha eliminado correctamente.

Tabla 8.7: Prueba 2.2

Prueba 2.3 (RF-14)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la modificación de un cliente.
Endpoint	/clients/id
Tipo de petición	PUT
Parámetros del Body	name, project
Acción	Modificar el nombre de un cliente, indicando su id en la petición y pasando un nuevo nombre como parámetro del body.
Resultado Esperado	El servidor contesta con un 200, indicando que se ha modificado correctamente.

Tabla 8.8: Prueba 2.3

Prueba 2.4 (RF-23)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de listar los edificios de un cliente.
Endpoint	/clients/id
Tipo de petición	GET
Parámetros del Body	-
Acción	Listar los edificios asociados a un cliente.
Resultado Esperado	El servidor debe responder devolviendo un JSON que contenga los edificios asociados al cliente con el id indicado.

Tabla 8.9: Prueba 2.4

8.1.3. Pruebas sobre gestión de edificios

En este apartado se realizarán las pruebas correspondientes al correcto funcionamiento sobre la gestión de edificios del sistema, que incluyen su creación, eliminación, modificación, así como el listado de sus plantas asociadas (requisitos RF-15, RF-16, RF-17, RF-24)

Prueba 3.1 (RF-15)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la creación de un edificio.
Endpoint	/buildings
Tipo de petición	POST
Parámetros del Body	name, client
Acción	Añadir un nuevo edificio indicando su nombre y el cliente al que está asociado.
Resultado Esperado	El servidor contesta con un 201, indicando que se ha creado correctamente.

Tabla 8.10: Prueba 3.1

Prueba 3.2 (RF-16)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la eliminación de un edificio.
Endpoint	/buildings/id
Tipo de petición	DELETE
Parámetros del Body	-
Acción	Eliminar un edificio concreto indicando su id en la petición.
Resultado Esperado	El servidor debe responder con un JSON vacío, indicando que se ha eliminado correctamente.

Tabla 8.11: Prueba 3.2

Prueba 3.3 (RF-17)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la modificación de un edificio.
Endpoint	/buildings/id
Tipo de petición	PUT
Parámetros del Body	name, client
Acción	Modificar el nombre de un edificio, indicando su id en la petición y pasando un nuevo nombre como parámetro del body.
Resultado Esperado	El servidor contesta con un 200, indicando que se ha modificado correctamente.

Tabla 8.12: Prueba 3.3

Prueba 3.4 (RF-23)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de listar las plantas de un edificio.
Endpoint	/buildings/id
Tipo de petición	GET
Parámetros del Body	-
Acción	Listar las plantas asociadas a un cliente.
Resultado Esperado	El servidor debe responder devolviendo un JSON que contenga las plantas asociadas al edificio con el id indicado.

Tabla 8.13: Prueba 3.4

8.1.4. Pruebas sobre gestión de plantas

En este apartado se realizarán las pruebas correspondientes al correcto funcionamiento sobre la gestión de plantas del sistema, que incluyen su creación, eliminación y modificación (requisitos RF-18, RF-19, RF-20)

Prueba 4.1 (RF-18)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la creación de una planta.
Endpoint	/floors
Tipo de petición	POST
Parámetros del Body	name, building, file
Acción	Añadir una nueva planta indicando su nombre y el edificio al que está asociada, así como el plano a procesar.
Resultado Esperado	El servidor contesta con un 201, indicando que se ha creado correctamente.

Tabla 8.14: Prueba 4.1

Prueba 4.2 (RF-19)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la eliminación de una planta.
Endpoint	/floors/id
Tipo de petición	DELETE
Parámetros del Body	-
Acción	Eliminar una planta concreta indicando su id en la petición.
Resultado Esperado	El servidor debe responder con un JSON vacío, indicando que se ha eliminado correctamente.

Tabla 8.15: Prueba 4.2

Prueba 4.3 (RF-20)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la modificación de una planta.
Endpoint	/floors/id
Tipo de petición	PUT
Parámetros del Body	name, building.
Acción	Modificar el nombre de una planta, indicando su id en la petición y pasando un nuevo nombre como parámetro del body.
Resultado Esperado	El servidor contesta con un 200, indicando que se ha modificado correctamente.

Tabla 8.16: Prueba 4.3

8.2. Pruebas para el front-end

En este punto se detallarán las pruebas correspondientes a la parte de la interfaz del servicio desarrollado, con los pasos y operaciones necesarias a realizar para comprobar el correcto funcionamiento de la aplicación.

8.2.1. Pruebas sobre gestión de sesión

En este apartado se realizarán las pruebas correspondientes al correcto funcionamiento sobre la gestión de sesión del usuario (requisitos RF-01 y RF-02)

Prueba 5.1 (RF-01)	
Descripción	Esta prueba está destinada a comprobar el funcionamiento de inicio de sesión del usuario en el sistema.
Acción	Introducir las credenciales de administrador en la ventana principal.
Resultado Esperado	El usuario ha conseguido identificarse correctamente en el sistema y está en la vista de proyectos.

Tabla 8.17: Prueba 5.1

Prueba 5.2 (RF-02)	
Descripción	Esta prueba está destinada a comprobar el funcionamiento de cierre de sesión del usuario en el sistema.
Acción	Hacer click en el nombre de usuario que se encuentra en la barra superior de la interfaz y seleccionar "Cerrar sesión".
Resultado Esperado	El usuario ha terminado su sesión y se encuentra en la página inicial.

Tabla 8.18: Prueba 5.2

8.2.2. Pruebas sobre proyectos

En este apartado se realizarán las pruebas correspondientes al correcto funcionamiento sobre la gestión de proyectos del sistema, que incluyen su creación, eliminación, modificación y listado (requisitos RF-09, RF-10, RF-11, RF-21)

En el apartado de listado de proyectos

Prueba 6.1 (RF-09)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la creación de un proyecto.
Acción	Añadir un nuevo proyecto indicando su nombre.
Resultado Esperado	El proyecto aparece añadido en la lista.

Tabla 8.19: Prueba 6.1

Prueba 6.2 (RF-21)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento del listado de los proyectos.
Acción	Una vez iniciada sesión, aparece la lista indicada.
Resultado Esperado	Deben aparecer todos los proyectos del sistema, listados de 5 en 5 y con opción a mostrar todos los existentes.

Tabla 8.20: Prueba 6.2

En el detalle de un proyecto

Prueba 6.3 (RF-10)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la eliminación de un proyecto.
Acción	Dentro de acciones, seleccionar la opción de eliminar el proyecto y confirmar.
Resultado Esperado	El proyecto ya no aparece en la lista.

Tabla 8.21: Prueba 6.3

Prueba 6.4 (RF-11)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la modificación de un proyecto.
Acción	Dentro de acciones, seleccionar la opción editar proyecto e introducir el nuevo nombre.
Resultado Esperado	El proyecto aparece con el nuevo nombre en el listado.

Tabla 8.22: Prueba 6.4

8.2.3. Pruebas sobre clientes

En este apartado se realizarán las pruebas correspondientes al correcto funcionamiento sobre la gestión de clientes del sistema, que incluyen su creación, eliminación, modificación y listado (requisitos RF-12, RF-13, RF-14, RF-22)

En el apartado de listado de clientes

Para acceder a este apartado, deberemos seleccionar un proyecto de la lista de proyectos y dentro de la opciones disponibles, "Ver clientes".

Prueba 7.1 (RF-12)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la creación de un cliente.
Acción	Añadir un nuevo cliente indicando su nombre.
Resultado Esperado	El cliente aparece añadido en la lista.

Tabla 8.23: Prueba 7.1

Prueba 7.2 (RF-22)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento del listado de los clientes.
Acción	Seleccionar la opción "Ver clientes" dentro del detalle de un proyecto.
Resultado Esperado	Deben aparecer todos los clientes asociados al proyecto seleccionado, listados de 5 en 5 y con opción a mostrar todos los existentes.

Tabla 8.24: Prueba 7.2

En el detalle de un cliente

Prueba 7.3 (RF-13)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la eliminación de un cliente.
Acción	Dentro de acciones, seleccionar la opción de eliminar el cliente y confirmar.
Resultado Esperado	El cliente ya no aparece en la lista.

Tabla 8.25: Prueba 7.3

Prueba 7.4 (RF-14)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la modificación de un cliente.
Acción	Dentro de acciones, seleccionar la opción editar cliente e introducir el nuevo nombre.
Resultado Esperado	El cliente aparece con el nuevo nombre en el listado.

Tabla 8.26: Prueba 7.4

8.2.4. Pruebas sobre edificios

En este apartado se realizarán las pruebas correspondientes al correcto funcionamiento sobre la gestión de edificios del sistema, que incluyen su creación, eliminación, modificación y listado (requisitos RF-15, RF-16, RF-17, RF-23)

En el apartado de listado de edificios

Para acceder a este apartado, deberemos seleccionar un cliente de la lista de clientes y dentro de la opciones disponibles, "Ver edificios".

Prueba 8.1 (RF-15)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la creación de un edificio.
Acción	Añadir un nuevo edificio indicando su nombre.
Resultado Esperado	El edificio aparece añadido en la lista.

Tabla 8.27: Prueba 8.1

Prueba 8.2 (RF-23)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento del listado de los edificios.
Acción	Seleccionar la opción "Ver edificios" dentro del detalle de un cliente.
Resultado Esperado	Deben aparecer todos los edificios asociados al cliente seleccionado, listados de 5 en 5 y con opción a mostrar todos los existentes.

Tabla 8.28: Prueba 8.2

En el detalle de un edificio

Prueba 8.3 (RF-16)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la eliminación de un cliente.
Acción	Dentro de acciones, seleccionar la opción de eliminar el cliente y confirmar.
Resultado Esperado	El cliente ya no aparece en la lista.

Tabla 8.29: Prueba 8.3

Prueba 8.4 (RF-17)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la modificación de un edificio.
Acción	Dentro de acciones, seleccionar la opción editar edificio e introducir el nuevo nombre.
Resultado Esperado	El edificio aparece con el nuevo nombre en el listado.

Tabla 8.30: Prueba 8.4

8.2.5. Pruebas sobre plantas

En este apartado se realizarán las pruebas correspondientes al correcto funcionamiento sobre la gestión de plantas del sistema, que incluyen su creación, eliminación, modificación y listado (requisitos RF-18, RF-19, RF-20, RF-24)

En el apartado de listado de plantas

Para acceder a este apartado, deberemos seleccionar un edificio de la lista de edificios y dentro de la opciones disponibles, "Ver plantas".

Prueba 9.1 (RF-18)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la creación de una planta.
Acción	Añadir un nuevo edificio indicando su nombre y seleccionando el plano correspondiente.
Resultado Esperado	La planta aparece añadida en la lista.

Tabla 8.31: Prueba 9.1

Prueba 9.2 (RF-24)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento del listado de las plantas.
Acción	Seleccionar la opción "Ver plantas" dentro del detalle de un edificio.
Resultado Esperado	Deben aparecer todas las plantas asociadas al cliente seleccionado, listadas de 5 en 5 y con opción a mostrar todas las existentes.

Tabla 8.32: Prueba 9.2

En el detalle de una planta

Prueba 9.3 (RF-19)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la eliminación de una planta.
Acción	Dentro de acciones, seleccionar la opción de eliminar la planta y confirmar.
Resultado Esperado	La planta ya no aparece en la lista.

Tabla 8.33: Prueba 9.3

Prueba 9.4 (RF-20)	
Descripción	Esta prueba está destinada a comprobar el correcto funcionamiento de la modificación de una planta.
Acción	Dentro de acciones, seleccionar la opción editar planta e introducir el nuevo nombre y/o seleccionar el nuevo plano de la planta.
Resultado Esperado	La planta aparece con el nuevo nombre en el listado.

Tabla 8.34: Prueba 9.4

Capítulo 9

Manual de programador

En este capítulo se detalla la información necesaria para un desarrollador que quiera trabajar en el servicio desarrollado, ya sea para realizar su mantenimiento, una modificación o incluir nuevas funcionalidades. Se incluirá todo lo necesario tanto para el back-end como para el front-end.

9.1. Back-end

Antes de comenzar con este punto, indicar que al final se incluirá un apartado con la suite de pruebas realizadas en Postman [31], con un ejemplo de cómo realizar cada una de las peticiones básicas(CRUD) al API REST.

9.1.1. Directivas generales

El código fuente desarrollado se encuentra en el directorio del proyecto "/app/main", encontrándose a su vez dividido en 4 subcarpetas: 'controller', 'model', 'service' y 'utils'.

- **Carpeta controller**: Contiene los endpoints a los que se realizarán las peticiones correspondientes. Cada endpoint corresponde de uno de los tipos básicos de operación CRUD. Devuelven una serie de códigos HTTP [10] en función de la operación realizada y la respuesta esperada del servidor en base al resultado a esa operación, y en caso de necesitarse, un JSON con la información concreta solicitada.
- **Carpeta model**: Contiene las entidades de la base de datos, definidas tal y cómo estarán almacenadas, es decir, cada campo definido en cada una de las clases corresponderá a una columna de la tabla correspondiente en la base de datos.
- **Carpeta service**: Contiene las clases que implementan las funciones necesarias para operar con la base de datos (*queries*) y la lógica de la aplicación..
- **Carpeta utils**: Contiene una clase para definir las columnas de cada una de las entidades que se implementan en las clases de la carpeta *model* y otra clase para definir las funciones del procesado de imágenes para generar los tiles del sistema.

9.1.2. Carpeta controller

En la carpeta controller se encuentran los endpoints a los que la aplicación realizará peticiones. Hay un total de 4, cada uno correspondiente a un apartado de la aplicación y siguiendo la siguiente jerarquía:

$$Floors \subset Buildings \subset Clients \subset Projects$$

En otras palabras, un proyecto tiene clientes, un cliente tiene edificios, etc. Las operaciones que se pueden realizar a los distintos endpoints son análogas, salvo para el caso de las plantas, que será explicado a parte y para un caso particular de un GET del apartado de los proyectos:

- **GET elements (projects)**: Obtiene todos los proyectos existentes en el sistema.
- **GET elements in element**: Obtiene los elementos que pertenecen a uno superior en la jerarquía mediante su id. Por ejemplo, en el caso de un proyecto con *id = 1*, obtendría todos sus clientes.
- **POST create new element**: Crea un nuevo elemento con los datos requeridos.
    - En el caso de las **plantas** se requiere como argumento para su creación un fichero asociado al plano de dicha planta. Mediante la creación, el fichero se subirá al servidor y se procesará convenientemente.
- **GET element by id**: Dado el id de un elemento, obtiene su información.
- **PUT edit element**: Mediante el identificador de un elemento seleccionado, se pueden modificar sus datos.
- **DELETE element**: Mediante el identificador de un elemento seleccionado, podemos eliminar del sistema dicho elemento. Si se borra un elemento que tiene otros por debajo dada la jerarquía del sistema, esos elementos también se eliminarán.

9.1.3. Carpeta model

En la carpeta model se encuentran definidas las entities de la base de datos, habiendo una por cada tabla: **project**, **client**, **building** y **floor**. Dentro de cada clase se definen sus atributos, que equivalen a las columnas de la tabla correspondiente de la base de datos, así como directivas en caso de modificaciones o borrados.

9.1.4. Carpeta service

En la carpeta service se encuentran las clases que proveen de consultas y funciones de tratamiento de los datos al controller correspondiente, habiendo uno por cada entidad del dominio: **project_service**, **client_service**, **building_service** y **floor_service**.

Dentro de cada clase se encuentran consultas para obtener todos los elementos de la tabla correspondiente, un elemento por su id, crear, editar y eliminar entradas de la base de datos. Un caso especial es el del servicio para las plantas, **floor_service** que también cuenta con funciones para el procesado de imágenes y la subida de archivos al servidor.

9.1.5. Carpeta utils

En la carpeta utils encontramos los siguientes archivos Python:

- **dto.py**, que se encarga de definir clases, una por cada tabla de la base de datos, para poder operar con ella través de los modelos que definimos en la carpeta **model**.
- **image2tiles.py**, que se encarga del preprocesado de imagen que recibe el servidor al crear una planta y de la generación de tiles. Concretamente tiene funciones para hacer lo siguiente:
    1. Subir la imagen al servidor (**uploadImage(file)**).

2. Comprobar que la imagen coincida con una de las extensiones permitidas (**allowedFile(filename)**).
3. Redimensionar la imagen de entrada para poder procesarla correctamente (**resizeEntryImage(img, exitPath)**).
4. Centrar la imagen redimensionada para que los tiles se generen correctamente (**centerImage(img, resultName)**).
5. Procesar la imagen redimensionada y centrada para generar los tiles (**process_map(filename, generatePath, file)**).

9.1.6. Otros ficheros de interés

En esta sección se indicarán otros ficheros de interés que se encuentran en el proyecto, que sirven para configuración, despliegue, etc. Se indicará su ubicación mediante la ruta relativa al proyecto, siendo "/" la raíz del mismo.

- **/manage.py**: Configuraciones básicas como inicio de la base de datos, dirección donde se ejecutará el servidor y arranque de este.
- **/Makefile**: Contiene una serie de directivas para hacer más fácil la gestión del proyecto. En el fichero **/README.md** se puede ver en detalle el uso de cada una de ellas.
- **/Dockerfile** y **/docker-compose.yml**: Ficheros de configuración y despliegue del contenedor Docker.
- **/app/__init__.py**: Inicializa el API, indicando los namespaces de cada uno de los endpoints de la parte de la carpeta **controller**.

9.1.7. Pruebas en Postman

Tal cómo se indicó al comienzo de esta sección, a continuación se indicará cómo hacer una serie de peticiones de prueba para testear el funcionamiento del backend. La suite de pruebas vendrá incluida en el repositorio, la cuál se puede importar en Postman mediante "File - Import". Se ejemplificará como hacer las pruebas al endpoint de **proyectos**.

Antes de comenzar, en la siguiente imagen[9.1] se ve el inicio del programa, con la suite de prueba denominada *Tiler* importada. Si se desea añadir una petición nueva, bastaría con hacer click derecho en el lugar dónde se quiera crear y seleccionar "Add request". Si por el contrario, se desea crear un conjunto de pruebas nuevo, habría que seleccionar la opción *New collection*.

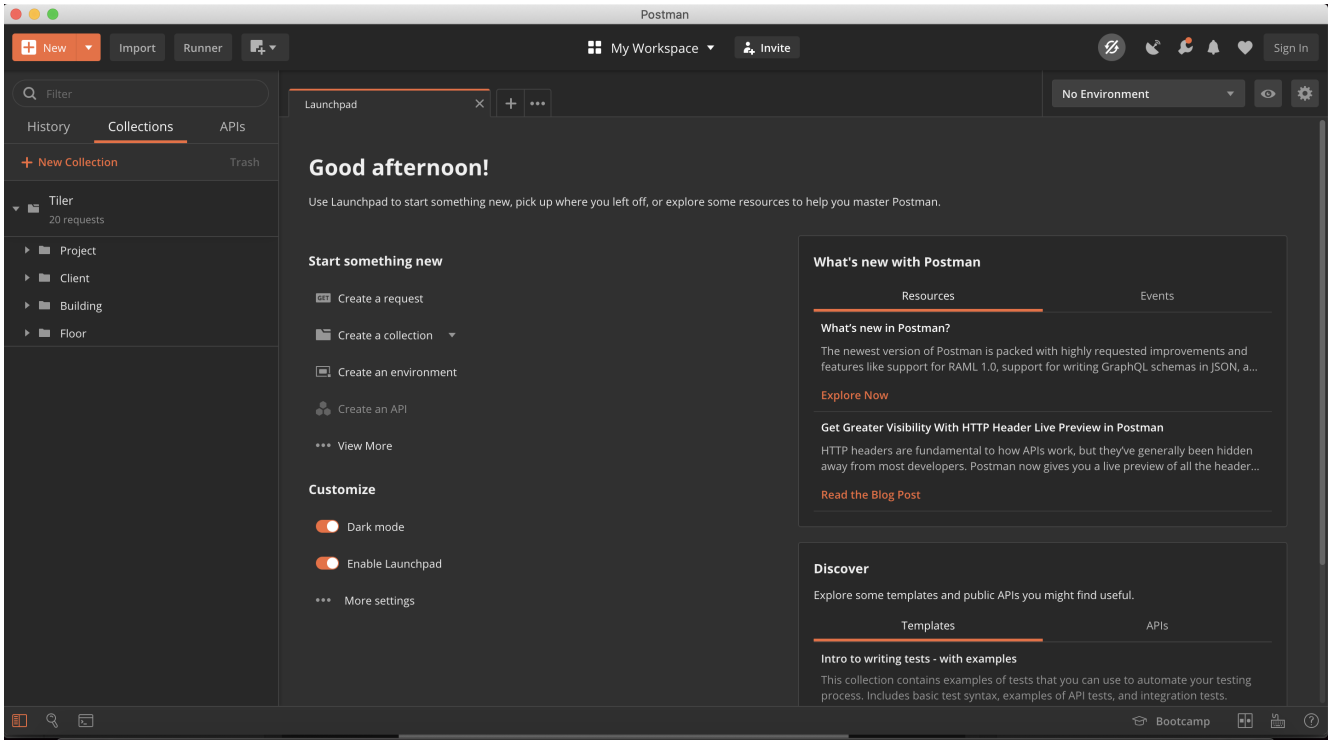

Figura 9.1: Pantalla de inicio de Postman

En todas las peticiones a realizar se debe indicar una **URL** y un **endpoint**. En el este caso, la URL base sería **http://0.0.0.0:5000/**, y el endpoint sería **/projects**.

Obtener todos los proyectos

Con esta petición obtendremos todos los proyectos del sistema.

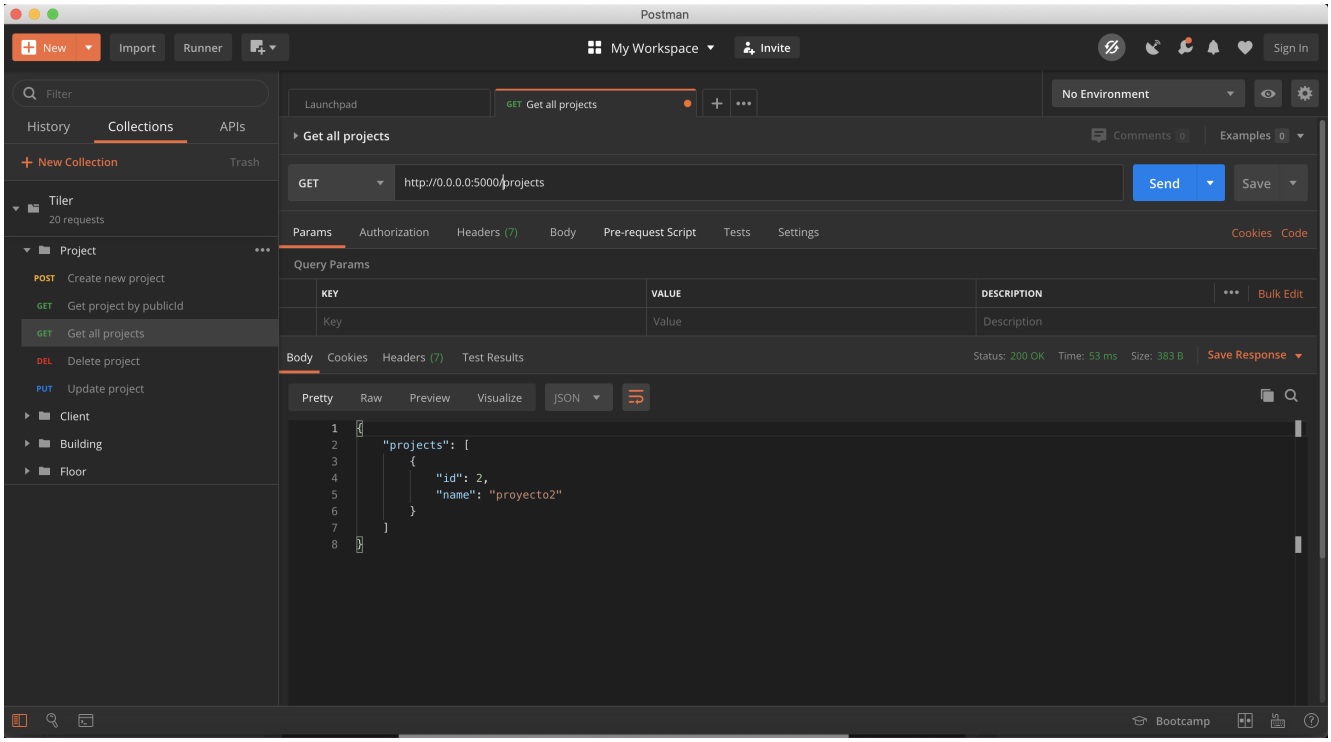

Figura 9.2: GET todos los proyectos

Crear un nuevo proyecto

Con esta petición crearemos un nuevo proyecto. En la segunda imagen[9.4] se pueden ver los parámetros que se mandan en la petición. Para poder mandarlos correctamente, debemos realizar lo siguiente:

- Seleccionar pestaña *body*.
- Dentro de esa pestaña, seleccionar el radiobutton que indica *raw*.
- En el desplegable a la derecha de los radiobuttons, seleccionar formato *JSON*.
- Para este caso, el JSON a construir tiene el siguiente formato:

```
1    {
2        "name" : "nombreProyecto"
3    }
4
```

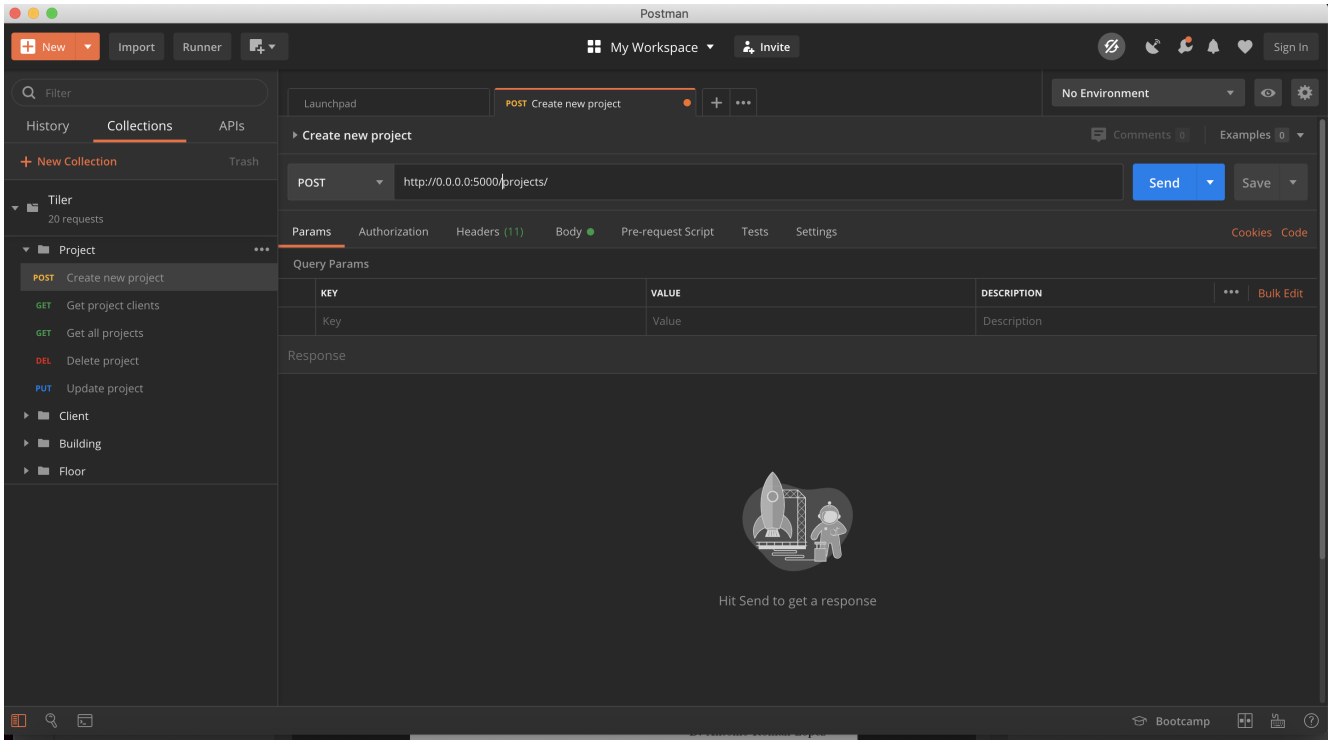

Figura 9.3: POST crear un nuevo proyecto

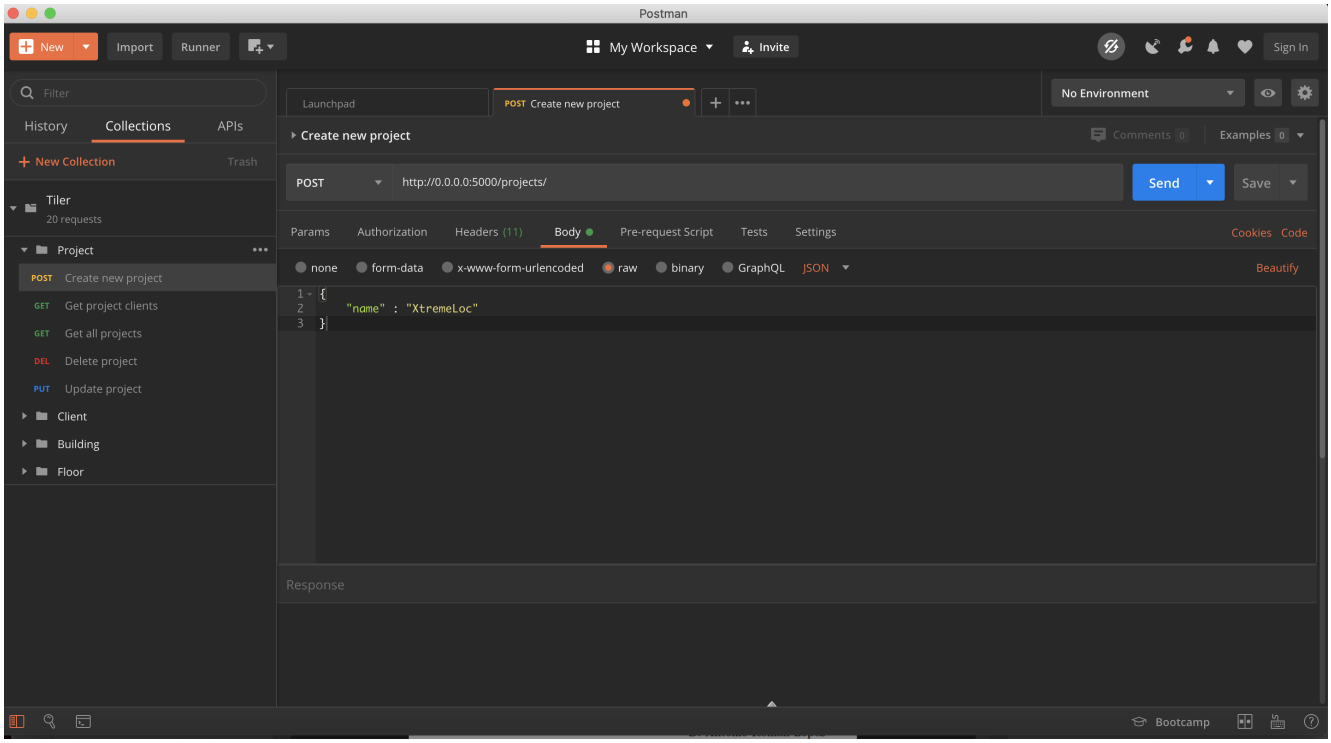

Figura 9.4: POST parámetros para crear un nuevo proyecto

Resto de peticiones

Para el resto de peticiones no se incluyen imágenes porque se realizan de forma análoga a la creación de un elemento. Simplemente, en el desplegable que indica *POST* se debería seleccionar el tipo de petición que se desee realizar, y en caso de tener parámetros, indicarlos en el body, igual que para el *POST*.

9.2. Front-end

En este punto se detallará lo necesario para la comprensión del trabajo realizado actualmente en el front-end del servicio.

9.2.1. Directivas generales

En la raíz del proyecto nos encontramos con varias carpetas (el código de la aplicación se encuentra en la denominada como 'src'). De la misma forma, esta carpeta se encuentra dividida en subcarpetas, de forma que el código se encuentre agrupado por tipo y función dentro de la aplicación: 'components', 'router', 'store' y 'views'.

- **Carpeta components**: Dentro de esta carpeta se encuentran los componentes Vue de la aplicación, divididos por tipos y utilidad. Los componentes Vue tienen 3 etiquetas principales:

  1. La etiqueta **template**, en la que se define el código HTML convencional, aunque en este caso se han usado principalmente componentes del plugin Vuetify [24].
  2. La etiqueta **script**, en la que se define la lógica y funciones del componente, en JavaScript.
  3. La etiqueta **style**, en la que se especifica el código CSS del componente para aplicarle algún tipo de estilo.

- **Carpeta router**: Contiene un único fichero 'index.js' escrito en JavaScript en el que se definen las rutas de la aplicación, de modo que así se pueda manejar mejor la navegabilidad entre los componentes que conforman el sistema.

- **Carpeta store**: Esta carpeta contiene todos los ficheros necesarios para definen el controlador de la aplicación, escritos también es JavaScript. Dentro de esta carpeta encontramos otra subcarpeta denominada 'modules' en las que encontramos las funciones de cada punto de la aplicación, en los que se describen propiedades de los componentes y/o vistas asociados, así como funciones y llamadas al API REST. La estructura de cada fichero dentro de store es principalmente la siguiente:

  - **State**: Son los atributos propiamente dichos. Se encargan de almacenar información relacionada con la aplicación.
  - **Getters**: Se encargan de obtener los valores almacenados en el *State*.
  - **Mutations**: Se encargan de editar los valores del *State*.
  - **Actions**: Funciones asíncronas que realizan una tarea concreta y que solicitan la modificación del estado de la aplicación.
  - **Modules**: Este apartado es opcional. Dentro de él se indican los módulos que hemos desarrollado, de modo que nuestro código quede más limpio y legible. Para poder usar módulos dentro de este apartado, debemos importarlos mediante '*import nombreModulo from rutaModulo*'.

- **Carpeta views**: En esta carpeta se encuentran los componentes Vue que simbolizan cada una de las vistas principales de la aplicación, en otras palabras, son los contenedores principales del resto de componentes.

9.2.2. Carpeta components

Dentro de la carpeta components encontramos varias subcarpetas. Antes de comenzar con el detalle de cada una, volver a mencionar que la división está hecha de forma que el cometido de cada componente quede más claro mediante esta agrupación.

La **subcarpeta BreadCrumbs** define un componente de este tipo. Un BreadCrumb, literalmente "miga de pan", es un tipo de componente interactivo que marca una ruta absoluta dentro de la navegación de una aplicación, de modo que el usuario pueda ver el recorrido de sus acciones, permitiendo que pueda ir a componentes visitados haciendo click en una de sus rutas.

En la **subcarpeta Navigation** encontramos otras 5 subcarpetas, 4 de ellas nombradas igual que el componente que contienen y la restante para un componente de visualización concreto, **ExpansionRow**. El objetivo de estos componentes es mostrar el detalle del elemento de la herramienta que representan. Dentro de estos componentes se hace uso de otros componentes que se citarán a continuación, entre los que se encuentra el componente **BreadCrumb** anteriormente detallado.

Las 4 subcarpetas correspondientes **Projects**, **Clients**, **Buildings** y **Floors** contienen cada una un componente nombrado igual, y construidos de forma similar. Los 3 primeros muestran el nombre del elementos en concreto (un proyecto en particular del sistema, por ejemplo, en el componente. **Project**, ídem con el resto), y dos *expansion panels*: El primero para ir al siguiente apartado de al aplicación y el segundo, para editar o eliminar el elemento en el que nos encontramos.

Dentro de la **subcarpeta OnceUsed** encontrado una tabla de datos genérica, que se mostrará dentro de los componentes contenidos en la carpeta **views**, que se detallará más adelante. Esta tabla sirve para mostrar todas las entidades de un determinado tipo dentro del sistema (el tipo de datos que muestra la tabla viene dado por el componente que la muestra).

Por último, las subcarpetas **PopUpComponents** y **Toolbar** contienen respectivamente los diálogos de datos que se muestran a lo largo de la aplicación (para añadir y editar datos del sistema), y la barra de la aplicación junto con el **NavigationDrawer**, que muestra una lista de proyectos con la opción de poder filtrar por nombre.

9.2.3. Carpeta router

En la carpeta router, como ya se ha mencionado anteriormente, encontramos un único fichero con extensión **.js** que incluye todas las rutas a los distintos componentes que forman parte de la aplicación, generando una URL para cada uno de ellos. Los componentes contemplados en esta parte son considerados como subpáginas dentro de la aplicación, ya que son componentes "alcanzables" mediante una URL. La implementación de Router se ha realizado siguiendo la sugerida por la propia página de Router [21].

9.2.4. Carpeta store

Dentro de la carpeta store se encuentran 3 elementos: 2 ficheros aislados y una carpeta llamada **modules**.

- **api.js**: Este fichero declara la URL base de las peticiones a realizar al API REST.
- **index.js**: En este fichero hay propiedades y funciones que se usan de forma general en toda la aplicación, y se hace la importación de los archivos contenidos en la carpeta **modules**.
- **Carpeta modules**: En esta carpeta encontramos 5 ficheros con extensión **.js**, que contienen cada uno atributos y funciones específicas para el apartado análogo de la aplicación que contiene su nombre. Contienen atributos y propiedades de sus respectivos componentes, y funciones para realizar operaciones con ellos y peticiones al API REST. Las operaciones a realizar hacia el API REST son las básicas de un esquema CRUD: CREATE(POST), READ(GET), UPDATE(PUT) y DELETE, además de contener otro tipo de funciones.

9.2.5. Carpeta views

Dentro de esta carpeta se encuentran 5 componentes Vue, cada uno nombrado de forma que hace referencia a la parte de la aplicación que representa.

9.2.6. Otros ficheros de interés

Un fichero a destacar es el componente **App.vue**, que se encuentra dentro de la **carpeta src**. Este componente es el punto de entrada de la aplicación dentro del framework Vue, y en ella se deben definir y usar los elementos que son comunes a toda la web. En este caso la **barra de navegación** y una etiqueta llamada **router-view**, que es la etiqueta que marca dónde se van a cargar los componentes declarados dentro del archivo de configuración de **router**.

En el siguiente capítulo se detallará lo necesario para poder instalar, desplegar y usar el sistema en cualquier equipo informático.

Capítulo 10

Manual de instalación

En este apartado se detallarán los pasos y dependencias necesarios para poder tener el conjunto del proyecto en la máquina que se precise y poder hacerlo funcionar. Antes de comenzar, destacar que el despliegue y gestión de todas las dependencias se han centralizado en Docker, por facilitar la portabilidad y despliegue de todo el conjunto del sistema.

Aún con esto, se indicarán los pasos necesarios para poder tener el sistema y poder desplegarlo sin inconvenientes.

Nota:La instalación y despliegue del sistema puede realizarse en cualquier sistema operativo siempre que se trate de un Linux, un MacOS X o un Windows. Cómo ya se ha mencionado, el despliegue de todo el sistema está centralizado en Docker, por lo que la gestión de dependencias del mismo se explicará en base a esta tecnología.

Ya que se aportan los ficheros necesarios para poder desplegar el sistema en su totalidad, se comenzará indicando cómo instalar las distintas dependencias necesarias a nivel local para que sea posible trabajar con los ficheros propios de Docker.

10.1. Instalación de Docker

10.1.1. Requisitos de software

1. Sistema operativo: Linux,MacOS X o Windows.
2. Docker.
3. Docker-compose.

10.1.2. Pasos a realizar

Para instalar las dependencias necesarias citadas anteriormente, deberemos realizar lo siguiente (puede variar según el sistema operativo. Las instrucciones a continuación indicadas han sido probadas en un sistema Unix):

1. Entrar en la página de descarga de Docker [8] y seguir los pasos necesarios para el sistema operativo en el que queramos instalarlo.
2. Una vez realizada la instalación podremos comprobar si se ha realizado correctamente abriendo un terminal y poniendo el siguiente comando: *docker –version*.

Si el comando nos devuelve información asociada a la versión de la instalación, querrá decir que se ha instalado correctamente.

3. Tras instalar Docker, deberemos instalar docker-compose, herramienta necesaria para usar ficheros de definición de contenedores y poder desplegarlos mediante un comando sencillo. Para instalar esta herramienta entraremos en la página de Docker [6] y seguiremos los pasos necesarios para el sistema operativo destino.

4. Una vez realizada la instalación podremos comprobar si se ha realizado correctamente abriendo un terminal y poniendo el siguiente comando: *docker-compose –version*. Si el comando nos devuelve información asociada a la versión de la instalación, querrá decir que se ha instalado correctamente.

Aún habiendo indicado que el despliegue se centra en Docker, a continuación se indicarán las dependencias necesarias tanto para la parte del servidor como de la interfaz web, para poder instalarlas y configurarlas en un host físico. En el apartado **Despliegue**(10.4) se explicará en detalle las configuraciones Docker realizadas.

Antes de comenzar con la explicación de los distintos pasos para la instalación del sistema, cabe **destacar** lo siguiente: Para poder realizar un despliegue correcto, las imágenes que se compilen deberán llevar un tag concreto, de lo contrario, los scripts de lanzamiento fallarán al no encontrar la imagen que tienen asociada en su definición (el nombre de estas imágenes se pueden ver en cada uno de los docker-compose del front-end y el back-end, en la etiqueta *image*).

10.2. Base de datos y Back-end

Debido a la naturaleza de la base de datos empleada, esta se encuentra dentro del propio back-end. SQLite genera una base de datos dentro de un fichero (*database.db*) en el que se encuentra toda la información de la misma. De esta manera, puede usarse en cualquier dispositivo sin necesidad de crear la base de datos localmente. A continuación, se indicarán los pasos necesarios para contar con todo el paquete y poder desplegarlo correctamente.

10.2.1. Requisitos de software

1. Python 3.
2. SQLite3.
3. GDAL.

10.2.2. Pasos a realizar

Los pasos a continuación detallados se han seguido para instalar las dependencias en un sistema Linux, por lo que se recomienda instalarlo en un sistema de estas características:

1. Abrimos un terminal.
2. Seguimos los pasos indicados en este enlace [19].
3. Una vez comprobado que se ha instalado correctamente Python, pasamos a instalar SQLite3, estando en el mismo terminal.
4. Introducimos el comando *apt-get install sqlite3*. Para comprobar que se ha instalado correctamente bastará con escribir el comando *sqlite3* en el terminal, y si lo reconoce, nos devolverá información relacionada con la versión instalada.
5. Para instalar las dependencias necesarias de GDAL, seguimos los pasos indicados en este enlace [15].
6. Clonar el proyecto *Tiles/tiles-back* si no se ha clonado ya.
7. Para el resto de configuraciones seguir el archivo README, que hace referencia a una serie de comandos indicados en el Makefile del proyecto.

10.3. Frontend

10.3.1. Requisitos software

1. npm

10.3.2. Pasos a realizar

1. Clonar el proyecto *Tiles/tiles-frontend* si no se ha clonado ya.
2. Seguir las instrucciones detalladas en el README del proyecto. Dentro del primer punto, no realizar el comando *vue init*, ya que el proyecto ya está creado y no es necesario iniciarlo después de clonarlo.

10.4. Despliegue

10.4.1. Requisitos software

Los requisitos software para este apartado son los indicados en el punto de **Docker**(10).

10.4.2. Pasos a realizar

Los pasos serán separados para la parte de front-end y la parte de back-end, de modo que sea más sencillo realizar el despliegue de cada una de las partes del sistema.

Back-end

1. Comprobar que todos los componentes necesarios para Docker se han instalado correctamente(10) si no se ha hecho anteriormente.
2. Clonar el proyecto *Tiles/tiles-back* si no se ha clonado ya.
3. En un terminal, nos situamos en la ruta del proyecto anteriormente clonado.

4. Una vez en dicha ruta (/ruta/de/clonado/tiles-back), contamos con una serie de ficheros. Los que nos interesan son el Dockerfile y el docker-compose.

    - **Dockerfile**: Indica todas las dependencias necesarias del contenedor. Para más información al respecto, visitar la documentación oficial [7].
    - **Docker-compose**: Establece la información asociada al contenedor para comenzar su ejecución. Para más información al respecto, visitar la documentación oficial [5].

5. Para compilar la imagen deberemos usar el comando *"docker build -t nombreContenedor ."*. Este comando compila la imagen del contenedor que posteriormente desplegaremos, dando como nombre a dicho contenedor *nombreContenedor* (en nuestro caso lo nombraremos como *tiler*).

6. Una vez compilada la imagen, podremos levantar el contenedor mediante el comando *"docker-compose up -d"*. El flag -d ejecuta el contenedor en modo *detach*, es decir, en segundo plano, de modo que quede ejecutándose sin "ocupar espacio" en el terminal.

Front-end

1. Comprobar que todos los componentes necesarioas para docker se han instalado correctamente(10) si no se ha hecho anteriormente.

2. Clonar el proyecto *Tiles/tiles-frontend* si no se ha clonado ya.

3. En un terminal, nos situamos en la ruta del proyecto anteriormente clonado.

4. Una vez en dicha ruta (/ruta/de/clonado/tiles-frontend), contamos con una serie de ficheros. Los que nos interesan son el Dockerfile y el docker-compose.

5. Para compilar la imagen deberemos usar el comando "docker build -t nombreContenedor .". Este comando compila la imagen del contenedor que posteriormente desplegaremos, dando como nombre a dicho contenedor *nombreContenedor* (en nuestro caso lo nombraremos como tiler-frontend).

6. Una vez compilada la imagen, podremos levantar el contenedor mediante el comando "docker-compose up -d". El flag -d ejecuta el contenedor en modo *detach*, es decir, en segundo plano, de modo que quede ejecutándose sin "ocupar espacio" en el terminal.

Una vez realizado el manual de instalación y expuestos todos los pasos que hay que realizar para contar con todas las dependencias del sistema, pasaremos a la última parte de manuales con el manual de usuario, con el que se explicará el funcionamiento del sistema mediante la interfaz web desarrollada.

10.5. Integración Continua

Los pasos indicados anteriormente sirven para realizar una instalación y despliegue manual de todo el sistema. Esto se facilita y evita con el uso de la integración continua, característica que ha sido integrada en este proyecto, aprovechando que el repositorio usado ha sido GitLab y cuenta con la opción de incluir un fichero de configuración de estas características.

Los proyectos están configurados de tal manera para que cada vez que se hace un push a una rama en concreto se realicen una serie de instrucciones. Para nuestro caso, se diferencian dos:

- Rama develop: Para esta rama está incluido el build de las imágenes de docker, de modo que al hacer un push o un merge de una rama en ésta, la imagen de docker asociada se compile automáticamente y quede guardada en el registry del repositorio.
- Rama master: Además de contar con la misma opción que la rama develop, cuenta con la automatización del despliegue del sistema en el servidor correspondiente, evitando realizar de forma manual todo el proceso de descarga de imagen de docker y posterior despliegue de los contenedores.

Capítulo 11

Manual de usuario

En este apartado de la documentación se detallará paso por paso mediante apoyo de imágenes, la forma de usar la interfaz del servicio desarrollado durante el transcurso del proyecto. Todas las acciones que a continuación se indican están destinadas para ser realizadas por un administrador del sistema.

11.1. Acceso a la aplicación

La aplicación tiler será accesible mediante un navegador web a través de un dominio dado. Para poder acceder a las funcionalidades del sistema, el usuario deberá estar registrado en el sistema como administrador e iniciar sesión en el mismo.

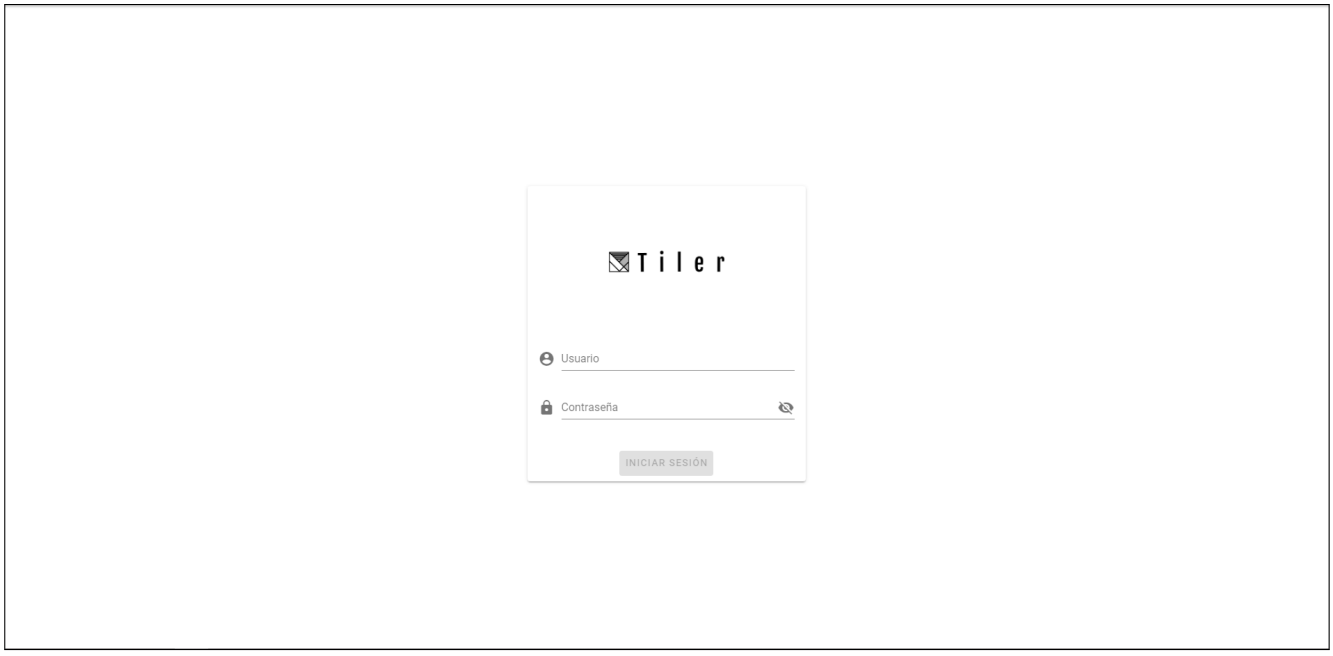

Figura 11.1: Inicio de sesión en Tiler

11.2. Navegación entre elementos

Según se acceda a las distintas secciones de la aplicación, se generará una ruta con un formato determinado (que se detallará a continuación) con la que se podrá ver de forma gráfica en qué apartado concreto nos encontramos y mediante la que nos podremos desplazar a dichos apartados.

La ruta tiene el siguiente formato: **Proyectos - nombreProyecto - Clientes - nombreCliente -**

Edificios - nombreEdificio - Plantas - nombrePlanta. Esta ruta se genera de esta forma dada la jerarquía que tienen los elementos dentro del sistema. En ella, las etiquetas que son del tipo *nombreElemento* marcan el elemento que se ha mostrado en detalle en algún momento.

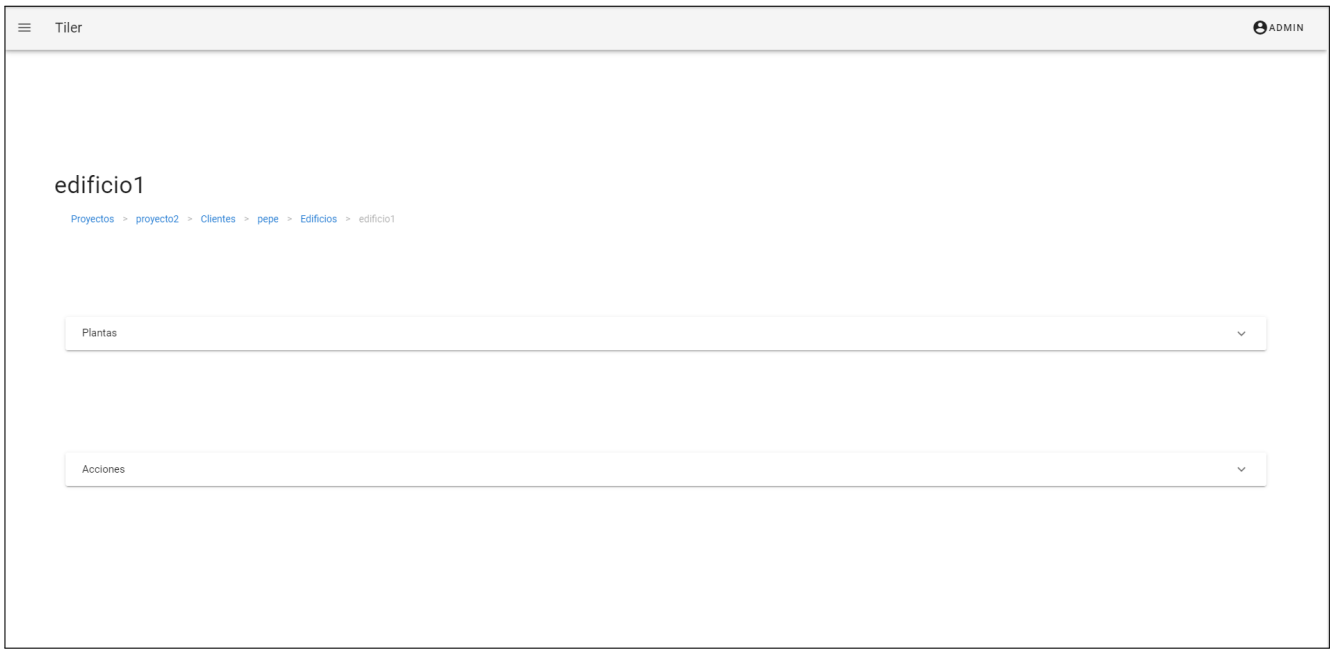

Figura 11.2: Ruta de navegación de la aplicación

Para facilitar la selección del detalle de otro proyecto, una vez que hemos navegado en profundidad sobre uno concreto, tenemos un menú lateral que se despliega haciendo click en las 3 líneas horizontales que están al lado del texto de la barra superior *Tiler*.

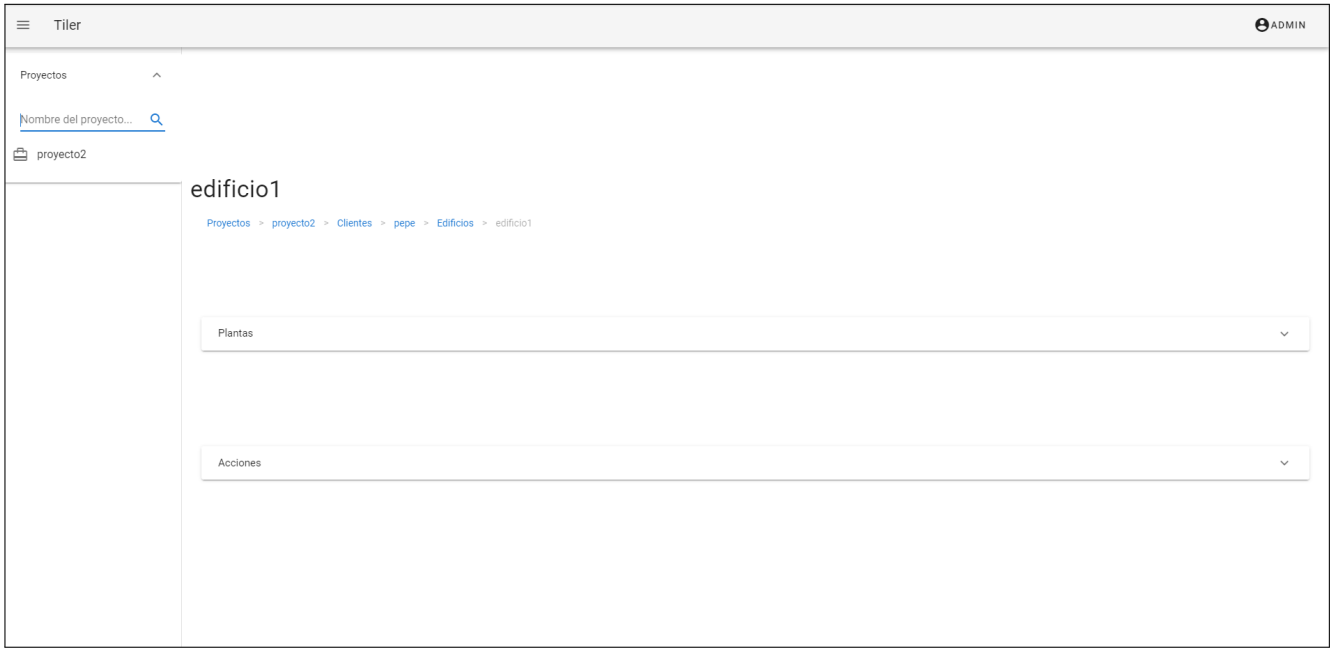

Figura 11.3: Menú lateral de selección de Proyectos

Para poder seleccionar un proyecto dentro de este menú, hacemos click en **Proyectos** y se desplegará una barra de búsqueda junto con los proyectos que hay en el sistema. La barra de búsqueda facilitará el encontrar un proyecto concreto si conocemos su nombre.

11.3. Administración de proyectos

En este apartado se detallarán las operaciones que se podrán realizar con los proyectos del sistema.

11.3.1. Listado de proyectos

Dentro de este apartado de la aplicación se podrán ver los proyectos con los que cuenta el sistema. Desde esta vista podrá añadirse un proyecto nuevo seleccionando **Añadir proyecto**. Si hacemos click sobre uno de los proyectos de la lista, entraremos en el detalle de dicho proyecto, dónde podremos **editarlo**, **eliminarlo** o ver sus **clientes**.

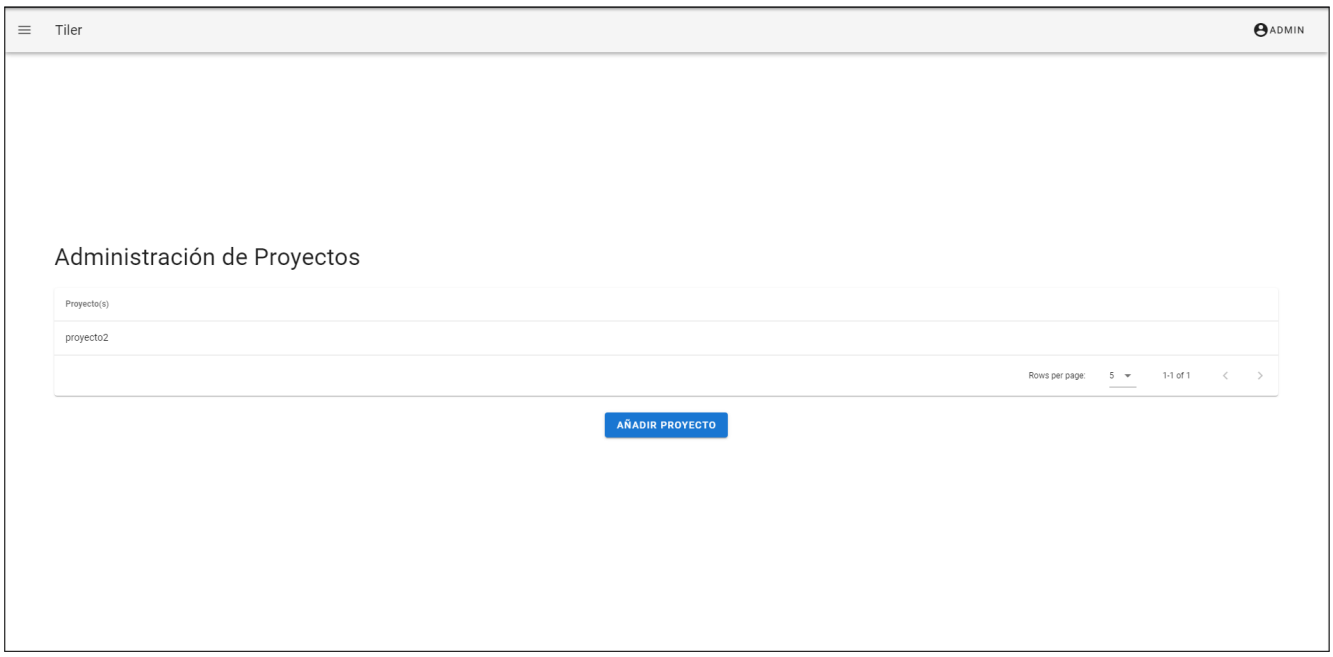

Figura 11.4: Listado de proyectos del sistema

11.3.2. Añadir un proyecto nuevo

Para poder añadir un proyecto nuevo haremos click, como hemos mencionado anteriormente, en **AÑADIR PROYECTO**, y se nos desplegará un diálogo que nos solicitará la información requerida para añadir un proyecto nuevo (el nombre en este caso).

Para que el proyecto se guarde correctamente en el sistema, deberemos introducir un nombre válido (uno que no esté registrado ya) y dar en la opción **GUARDAR**. Si por el contrario se quiere cancelar el añadido de un proyecto se deberá seleccionar **CANCELAR**.

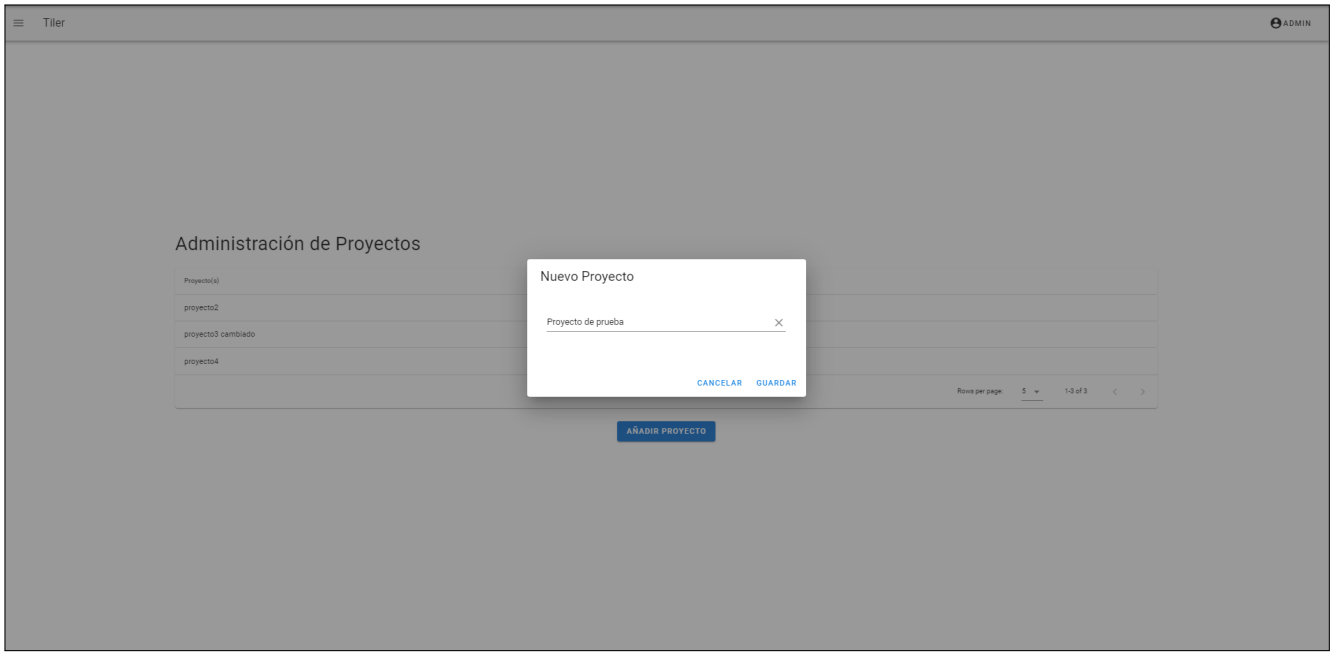

Figura 11.5: Añadir un nuevo proyecto

11.3.3. Detalle de proyecto

Dando click en cualquiera de los proyectos del listado anterior, podremos ver el detalle de la información de dicho proyecto. Este detalle incluye algunas operaciones adicionales sobre el proyecto (**editar** y **eliminar**), así como la opción de ver los **Clientes** de dicho proyecto. Para mostrar esas opciones contamos con dos menús desplegables: **Clientes**, que contiene la opción **Ver clientes**, y **Acciones**, que muestra las opciones de **editar** y **eliminar** el proyecto.

Para realizar cualquiera de estas operaciones deberemos hacer click sobre el icono que se ve en pantalla, a la izquierda del texto asociado a la operación.

Figura 11.6: Detalle de un proyecto

11.3.4. Editar un proyecto

Cuándo seleccionamos la opción de editar un proyecto, se nos mostrará un diálogo que contiene el nombre del proyecto que estamos viendo en detalle. Para cambiarlo bastará con borrar ese nombre, introducir uno nuevo (que no esté ya en el sistema) y dar a la opción **GUARDAR**. Si por el contrario no queremos editar el nombre del proyecto, deberemos hacer click en **CANCELAR**.

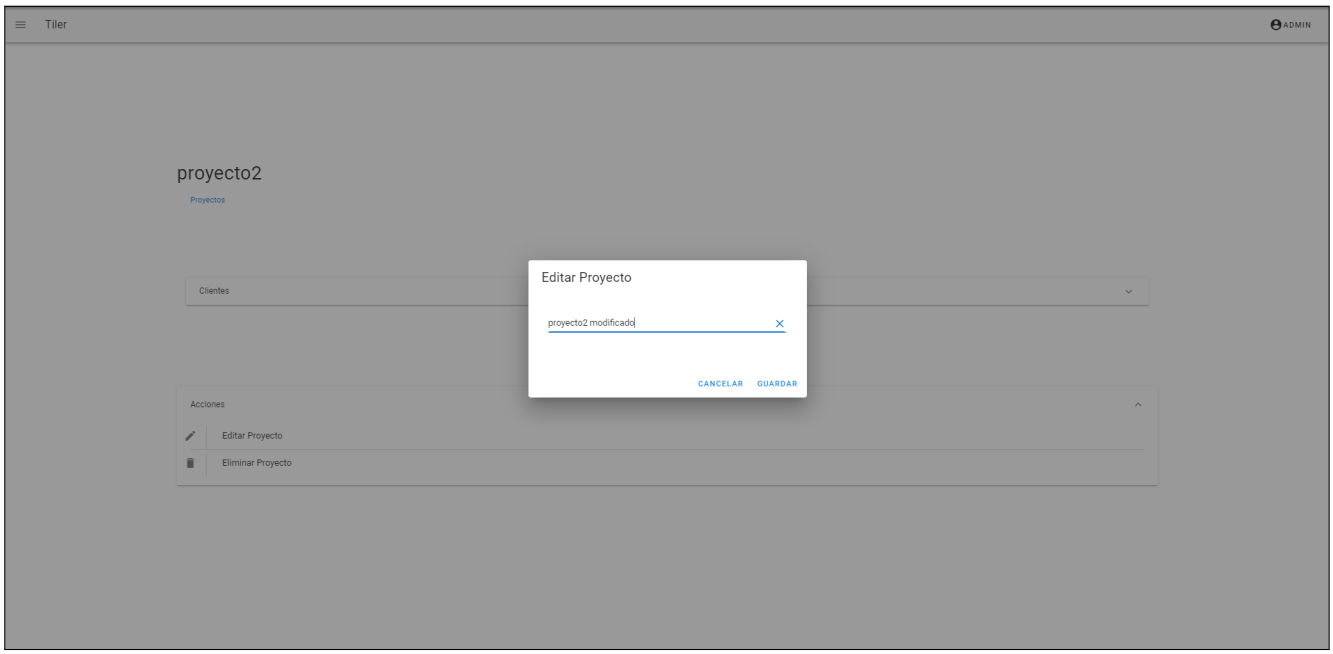

Figura 11.7: Editar un proyecto

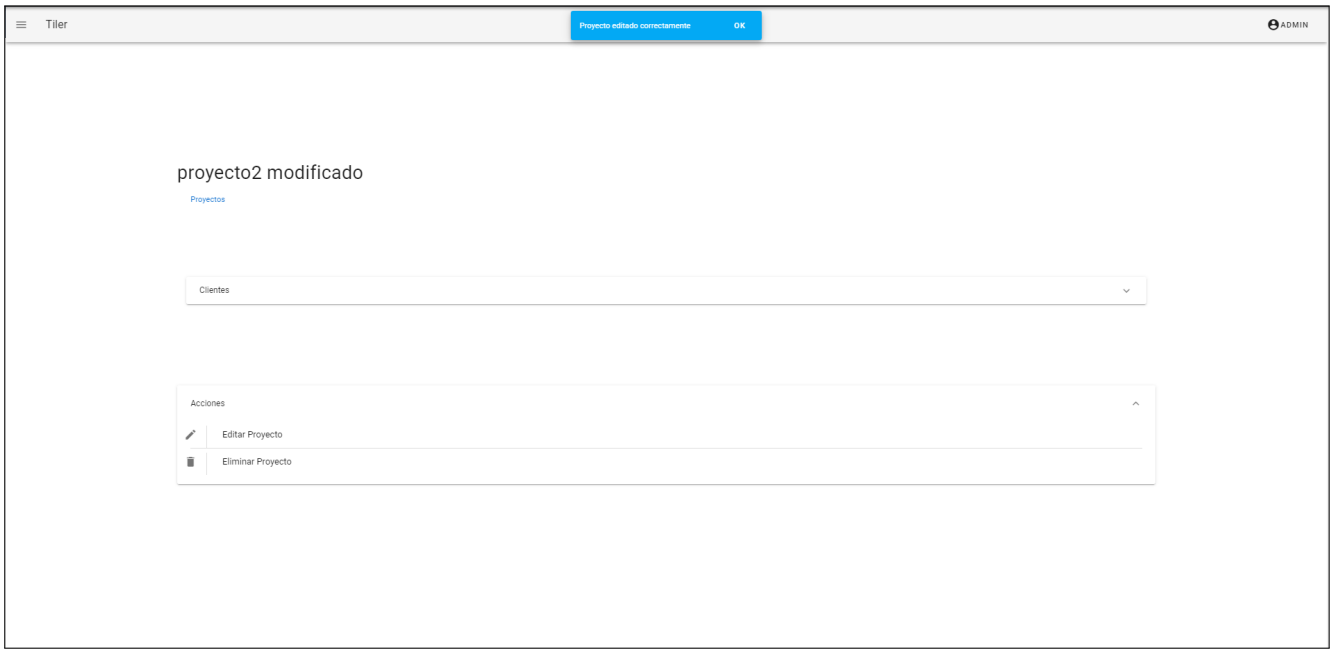

Figura 11.8: Proyecto editado

11.3.5. Eliminar un proyecto

De forma análoga a la opción de editar el proyecto, si queremos eliminar el proyecto que estamos viendo en detalle, se nos mostrará un mensaje en forma de alerta del navegador que nos pedirá confirmar esta acción. Si se quiere confirmar la acción, deberemos seleccionar **Aceptar**, mientras que si hemos dado

por error o simplemente no queremos eliminarlo, deberemos hacer click en **Cancelar**.

Nota importante: Hay que tener en cuenta que esta acción es irreversible, y si se realiza, se borrará toda la información relacionada con el proyecto: sus clientes, edificios y plantas.

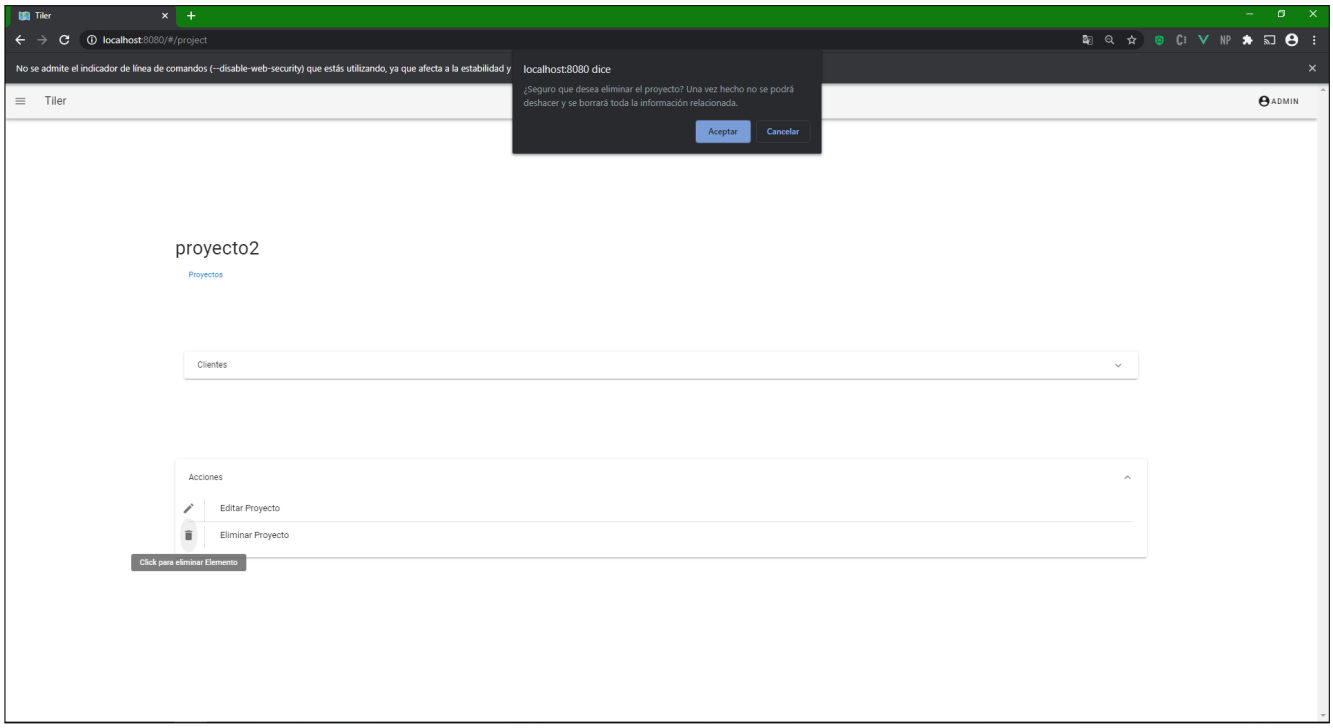

Figura 11.9: Eliminar un proyecto

11.4. Administración de clientes

De igual manera, al seleccionar la opción de **Ver clientes** de un proyecto, accederemos al apartado de administración de los clientes de ese proyecto. Las operaciones a realizar con los clientes son las mismas que con los proyectos:

- **Listado de clientes**
- **Añadir un nuevo cliente**
- **Ver el detalle de un cliente**
- **Editar un cliente**
- **Eliminar un cliente**
- **Ver los edificios de un cliente seleccionado**

No se han adjuntado capturas porque las distintas vistas asociadas a este apartado de la aplicación son idénticas al del apartado de administración de proyectos y las operaciones se realizan de la misma forma y cumpliendo las mismas restricciones.

11.5. Administración de edificios

De igual manera, al seleccionar la opción de **Ver edificios** de un cliente, accederemos al apartado de administración de los edificios de ese cliente. Las operaciones a realizar con los edificios son las mismas que con los proyectos:

- **Listado de edificios**
- **Añadir un nuevo edificio**
- **Ver el detalle de un edificio**
- **Editar un edificio**
- **Eliminar un edificio**
- **Ver las plantas de un edificio seleccionado**

No se han adjuntado capturas porque las distintas vistas asociadas a este apartado de la aplicación son idénticas al del apartado de administración de proyectos y las operaciones se realizan de la misma forma y cumpliendo las mismas restricciones.

11.6. Administración de plantas

De igual manera, al seleccionar la opción de **Ver plantas** de un edificios, accederemos al apartado de administración de las plantas de ese edificio. Las operaciones a realizar con las plantas son las mismas que con los proyectos:

- **Listado de plantas**
- **Añadir una nueva planta**
- **Ver el detalle de una planta**
- **Editar una planta**
- **Eliminar una planta**

En este apartado se adjuntarán capturas de cómo realizar el añadido de una planta nueva, ya que es una operación que cambia respecto a las anteriores, y también se mostrará una captura del detalle de una planta, dónde se podrá ver e interactuar con el plano de la planta seleccionada.

11.6.1. Añadir una nueva planta

Para poder añadir una planta nueva haremos click en **AÑADIR PLANTA**, y se nos desplegará un diálogo que nos solicitará la información requerida para añadir una planta nueva (el nombre de la planta y la imagen que representará la misma).

Para que la planta se guarde correctamente en el sistema, deberemos introducir un nombre válido (uno que no esté registrado ya) y dar en la opción **GUARDAR**. Si por el contrario se quiere cancelar el añadido de una planta se deberá seleccionar **CANCELAR**.

Al seleccionar la opción **GUARDAR**, podremos observar que el botón que antes contenía **AÑADIR PLANTA** ahora tiene una carga circular, que simboliza que está procesando el archivo que representa la planta. Cuándo esta operación termine, la interfaz notificará mediante un mensaje.

Para poder ver la planta en detalle, bastará con hacer click en el nombre de la planta que deseemos ver, y que mostrará cierta información que será detallada en el siguiente apartado.

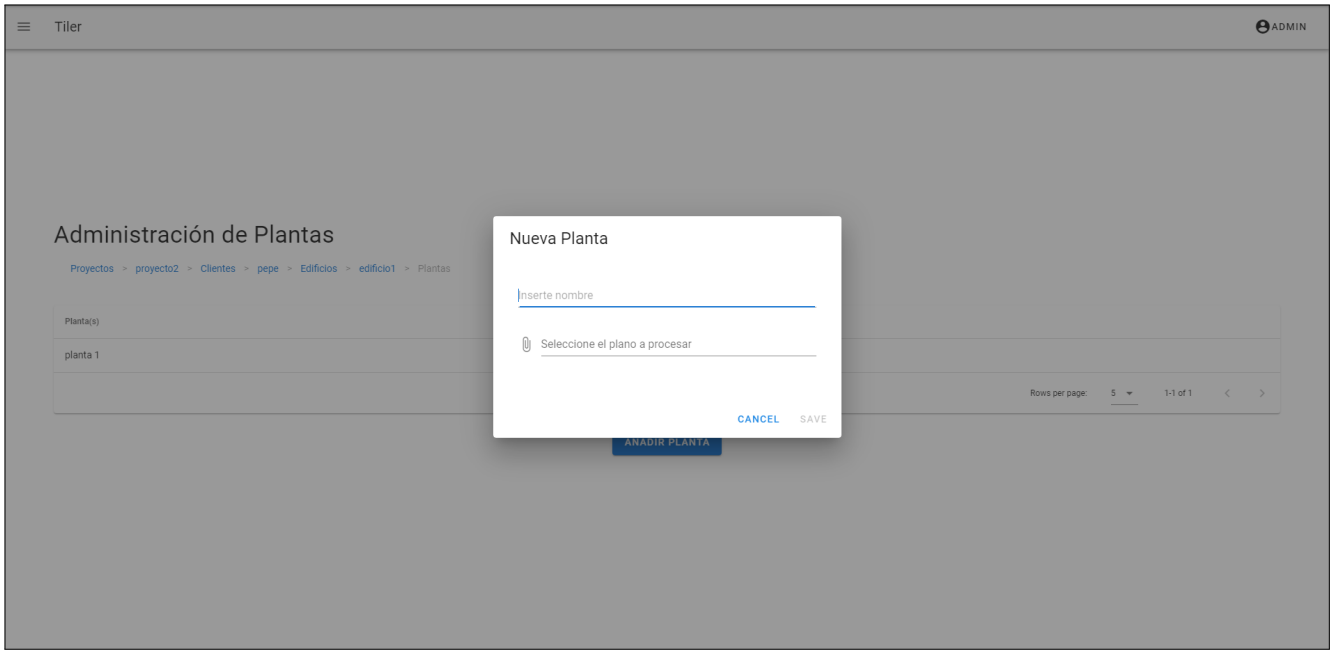

Figura 11.10: Añadir una planta nueva

11.6.2. Detalle de planta

Dando click en cualquiera de las plantas del listado anterior, podremos ver el detalle de la información de dicha planta. Este detalle incluye algunas operaciones adicionales sobre la planta (**editar** y **eliminar**), así como la opción de ver y copiar la **URL** de dicha planta. Esta funcionalidad se ha incluido para que la imagen procesada pueda usarse directamente en otras aplicaciones que usen el mismo formato de URL's, como por ejemplo Xtremeloc.

Para mostrar esas opciones contamos con dos menús desplegables: **URL de la planta**, que contiene la información de la URL y la posibilidad de copiar directamente al portapapeles (haciendo click en el icono a la izquierda de la URL), y **Acciones**, que muestra las opciones de **editar** y **eliminar** la planta.

Para realizar cualquiera de estas operaciones deberemos hacer click sobre el icono que se ve en pantalla, a la izquierda del texto asociado a la operación.

Por último, podremos ver e interactuar con la imagen procesada en el marco inferior. En dicho marco se podrá ver la imagen procesada que se seleccionó al añadir la planta que estamos viendo en detalle.

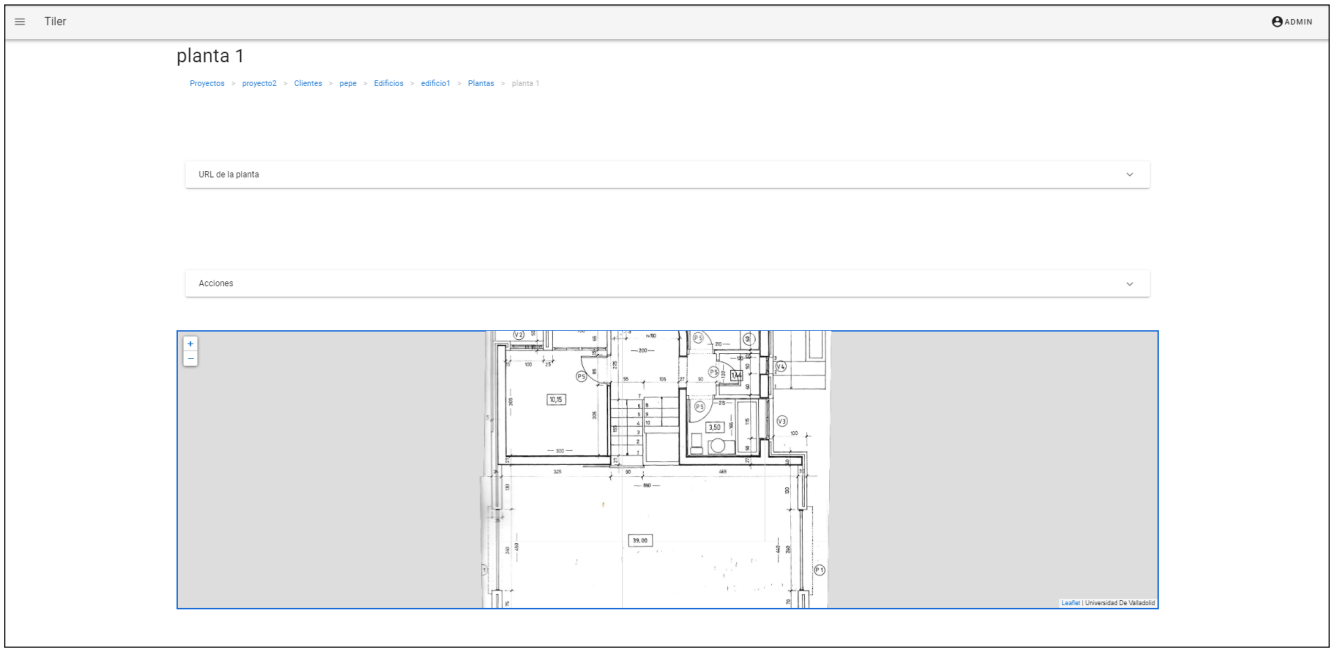

Figura 11.11: Detalle de una planta

11.7. Salir del sistema

Si en cualquier momento se quiere cerrar sesión, bastará con hacer click en la barra superior, en la parte derecha, dónde está el texto **ADMIN**. Se desplegará un menú que dará la opción de **Cerrar sesión** y a través de la cuál, volveremos a la pantalla de inicio de sesión del sistema.

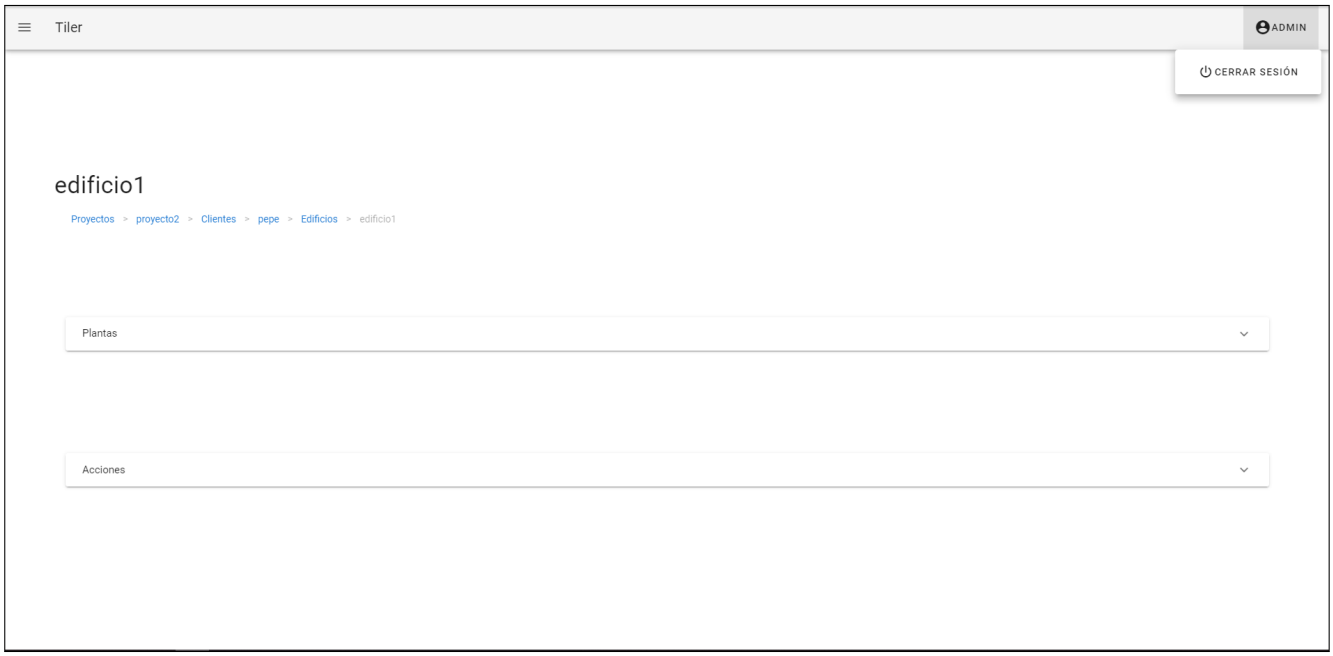

Figura 11.12: Cierre de sesión en Tiler

Capítulo 12

Conclusiones y trabajo futuro

12.1. Conclusiones

Durante la realización de este TFG se ha llevado a cabo la implementación de distintas funcionalidades para realizar una aplicación de administración sencilla y funcional, que permita gestionar elementos cartográficos para herramientas que actualmente se encuentran en producción. Antes de la elaboración de este proyecto, estas tareas se realizaban de forma manual, lo que resultaba muy tedioso y complejo, ya que durante el proceso podían surgir errores varios.

Gracias al conjunto de servicios implementados, entre los que se encuentra una parte de lógica de servidor y una interfaz web que aprovecha dicho servicio, se consigue que estas tareas se hagan de forma sencilla y gráfica, pudiendo añadir, eliminar, modificar y visualizar cada uno de los elementos que forman parte del sistema.

A continuación se presentará de forma resumida las tareas que se han llevado a cabo durante la realización del proyecto:

- Se ha llevado a cabo el análisis, diseño, plan de proyecto y de pruebas, así cómo una serie de manuales para el mantenimiento, instalación y uso del sistema.
- Se ha realizado una investigación para comprender el funcionamiento del estándar OpenStreetMaps.
- Se ha realizado un API REST que funcione usando la tecnología OpenStreetMaps, mediante el procesado de imágenes que representen plantas de edificios para el resto de herramientas que se encuentran en producción.
- Se ha realizado una investigación y entendimiento de la tecnología Docker.
- El despliegue de la aplicación se realiza mediante Docker, pues todos los sistemas que iban a hacer uso del proyecto a desarrollar se encuentran desplegados haciendo uso de esta tecnología.
- Se ha hecho uso del servidor de mapas que había anteriormente, usando la nomenclatura de direcciones de imágenes ya existente.
- Se ha adaptado el sistema de generación de mapas que había anteriormente, para poder implementar las nuevas funcionalidades y asegurar el correcto funcionamiento de la herramienta que se iba a desarrollar.
- Se ha desarrollado una aplicación de administración para todos los elementos cartográficos de los sistemas en producción que requieran este tipo de recursos al que sólo tiene acceso el administrador.

- Se ha implementado un sistema que permite dar de alta nuevos proyectos.
- Se ha implementado un sistema que permite dar de alta nuevos clientes, asociados a proyectos.
- Se ha implementado un sistema que permite dar de alta nuevos edificios, asociados a clientes.
- Se ha implementado un sistema que permite dar de alta nuevas plantas y procesar sus planos, asociadas a edificios.
- Se ha implementado un sistema que permite visualizar las imágenes procesadas de las plantas de los edificios.
- Se ha llevado a cabo la integración del sistema desarrollado con herramientas que ya se encontraban en producción, ampliando las funcionalidades de dichos sistemas.

Finalmente, mencionar que la herramienta desarrollada cumple con su cometido, facilitando las tareas de administración que requieren el uso de elementos de cartografía (plantas de edificios para nuestro contexto), pudiendo gestionar dichos elementos de una forma sencilla y rápida.

12.2. Trabajo futuro

Para terminar, se abren varias líneas de trabajo mediante las que se puede seguir extendiendo y mejorando el funcionamiento y funcionalidades del sistema que se ha desarrollado cómo fruto del trabajo realizado durante este TFG. Los puntos que a continuación se muestran son algunas de ellas que resultan especialmente interesantes:

- Desarrollo de una aplicación móvil para realizar las mismas tareas que se pueden realizar con el sistema realizado pero mediante un smartphone.
- Realizar un desarrollo multiplataforma, de modo que el sistema pueda usarse en cualquier dispositivo.

Referencias

[1] RJ Code Advance, Ultimo acceso: Mayo 2020.

[2] https://about.gitlab.com/, Ultimo acceso: Agosto 2020.

[3] https://about.gitlab.com/stages-devops-lifecycle/continuous integration/, Ultimo acceso: Agosto 2020.

[4] https://balsamiq.com/, Ultimo acceso: Febrero 2020.

[5] https://docs.docker.com/compose/, Ultimo acceso: Julio 2020.

[6] https://docs.docker.com/compose/install/, Ultimo acceso: Junio 2020.

[7] https://docs.docker.com/engine/reference/builder/, Ultimo acceso: Julio 2020.

[8] https://docs.docker.com/get docker/, Ultimo acceso: Junio 2020.

[9] https://docs.traefik.io/, Ultimo acceso: Mayo 2020.

[10] https://es.wikipedia.org/wiki/Anexo:C %C3 %B3digos_de_estado_HTTP, Ultimo acceso: Mayo 2020.

[11] https://gdal.org/programs/gdal2tiles.html, Ultimo acceso: Mayo 2020.

[12] https://git scm.com/, Ultimo acceso: Febrero 2020.

[13] https://leafletjs.com, Ultimo acceso: mayo 2019.

[14] https://material.io/design, Ultimo acceso: mayo 2020.

[15] https://mothergeo-py.readthedocs.io/en/latest/development/how-to/gdal-ubuntu pkg.html, Ultimo acceso: Junio 2020.

[16] https://nodejs.org/es/, Ultimo acceso: mayo 2020.

[17] https://palletsprojects.com, Ultimo acceso: Febrero 2020.

[18] https://pypi.org/project/gdal2tiles/, Ultimo acceso: Mayo 2020.

[19] https://python-guide es.readthedocs.io/es/latest/starting/install3/linux.html, Ultimo acceso: Junio 2020.

[20] https://raw.githubusercontent.com/commenthol/gdal2tiles leaflet/master/gdal2tiles.py, Ultimo acceso: Mayo 2020.

[21] https://router.vuejs.org/, Ultimo acceso: mayo 2019.

[22] https://v8.dev/, Ultimo acceso: mayo 2020.

[23] https://vuejs.org/, Ultimo acceso: Febrero 2020.

[24] https://vuetifyjs.com/en/, Ultimo acceso: mayo 2019.

[25] https://vuex.vuejs.org/, Ultimo acceso: mayo 2020.

[26] https://wiki.openstreetmap.org/wiki/Zoom_levels, Ultimo acceso: Mayo 2020.

[27] https://wsgi.readthedocs.io, Ultimo acceso: Febrero 2020.

[28] https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow workflow, Ultimo acceso: Agosto 2020.

[29] https://www.docker.com/, Ultimo acceso: Junio 2020.

[30] https://www.nginx.com/, Ultimo acceso: Mayo 2020.

[31] https://www.postman.com/, Ultimo acceso: Mayo 2020.

[32] https://www.sqlalchemy.org/, Ultimo acceso: Mayo 2020.

[33] https://www.sqlite.org/version3.html, Ultimo acceso: Mayo 2020.

[34] Cecilio Álvarez Caules. Axios js una librería de promesas, dec 2016.