UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Máster Universitario en Ingeniería Web

Trabajo Fin de Máster

Desarrollo de una aplicación web para la gestión de bienes inmobiliarios

Autor

Oliver Fernández García

Tutor

Santiago Alonso Villaverde

junio de 2024

AGRADECIMIENTOS

Quiero expresar mi agradecimiento a mi familia y amigos, quienes en las buenas y, sobre todo, en las malas, siempre han demostrado su apoyo incondicional.

Gracias a Santiago, mi guía en este proyecto, por su dedicación y atención, incluso con la carga de dirigir múltiples trabajos.

También quiero reconocer al resto de profesores del máster. Cada uno, con su especialidad, transmite sus conocimientos con pasión y así lo reflejan cada fin de semana de clase.

Finalmente, gracias a todas aquellas personas que, de una forma u otra, han contribuido a este logro. Su apoyo y aportaciones han hecho que alcanzar mis objetivos sea más sencillo.

RESUMEN

En el actual contexto socioeconómico, donde los precios de los inmuebles alcanzan máximos históricos, el alquiler se ha convertido en la única opción viable para muchos, especialmente para los jóvenes y familias en situación de vulnerabilidad. La creciente demanda del arrendamiento tradicional, junto con el aumento de la popularidad de plataformas como Booking o Airbnb, ha llevado a los propietarios a ser cada vez más selectivos y estrictos con el alquiler de sus propiedades.

En este proyecto se diseña y desarrolla una nueva aplicación web dirigida a estos propietarios, que buscan gestionar sus bienes inmobiliarios de manera activa, eficiente y desde un único lugar, sin necesidad de intermediarios como gestores. La aplicación ofrecerá funcionalidades para gestionar distintos tipos de propiedades (casas, pisos, garajes), así como la administración de los ingresos y gastos asociados (alquileres, seguros, cuotas de comunidad, etc.). Además, facilitará la gestión de documentos relacionados, permitirá la generación de balances financieros para evaluar la rentabilidad y ofrecerá asistencia fiscal para la realización de la declaración de la renta.

Para gestionar el proyecto se utiliza la metodología ágil Scrum, lo que permite adaptarse a los cambios y necesidades del proyecto de manera eficiente. La interfaz de usuario se desarrollará utilizando el *framework* Angular, diseñando y desarrollando los distintos módulos para una navegación eficiente. La parte servidora de la aplicación se creará con Node.js, utilizando el *framework* Express para desarrollar una API REST. Esto incluirá la definición de la arquitectura del servidor, la implementación de *endpoints* RESTful para la gestión de datos y la integración con una base de datos MySQL, garantizando la seguridad mediante mecanismos de autenticación y autorización.

El ecosistema software se complementará con herramientas de control de versiones, Git y GitHub, integradas con la herramienta de gestión Jira. Además, se definirá un flujo de integración continua con GitHub Actions, permitiendo lanzar pruebas automáticas ante una liberación de código y analizar el código de forma estática con Sonar. Finalmente, se configurará un entorno de pruebas unitarias y de integración utilizando el *framework* Jest y la biblioteca Supertest.

En este documento se presenta una introducción que abarca el contexto actual y los objetivos del proyecto, junto con el marco teórico. Se detalla el *inception deck*, que incluye, entre otros, la motivación, el análisis de riesgos e incertidumbres, la planificación global o los costos del proyecto. Posteriormente, se describe el *product backlog* y el desarrollo del sistema a lo largo de varios *sprints*, cubriendo el diseño de la arquitectura, la interfaz de usuario, los *endpoints*, las pruebas, así como las revisiones y retrospectivas de cada *sprint*. Finalmente, se presentan las conclusiones y las posibles líneas futuras de desarrollo, seguidas de la bibliografía utilizada.

Palabras clave

Inversión inmobiliaria, alquiler, Scrum, Angular, Node.js, API REST, MySQL

ABSTRACT

In the current socioeconomic context, where property prices are reaching historical highs, renting has become the only viable option for many, especially for young people and families in vulnerable situations. The growing demand for traditional rentals, along with the increasing popularity of platforms such as Booking or Airbnb, has led property owners to become increasingly selective and strict with renting out their properties.

This project involves the design and development of a new web application aimed at these property owners, who seek to manage their real estate assets actively, efficiently, and from a unique place, without the need for intermediaries such as managers. The application will offer functionalities to manage various types of properties (houses, flats, garages), as well as the administration of associated income and expenses (rentals, insurance, community fees, etc.). Additionally, it will facilitate the management of related documents, enable the generation of financial statements to assess profitability, and offer tax assistance for income tax filing.

The project will be managed using the agile Scrum methodology, allowing efficient adaptation to the project's changes and needs. The user interface will be developed using the Angular framework, designing and developing various modules for efficient navigation. The server-side of the application will be created with Node.js, using the Express framework to develop a REST API. This will include defining the server architecture, implementing RESTful endpoints for data management, and integrating with a MySQL database, ensuring security through authentication and authorization mechanisms.

The software ecosystem will be complemented with version control tools, Git and GitHub, integrated with the management tool Jira. Furthermore, a continuous integration flow will be defined with GitHub Actions, enabling automated testing upon code release and static code analysis with Sonar. Finally, a unit and integration testing environment will be configured using the Jest framework and the Supertest library.

This document presents an introduction covering the current context and project objectives, along with the theoretical framework. It details the inception deck, including, among other aspects, motivation, risk and uncertainty analysis, overall planning, and project costs. Subsequently, it describes the product backlog and the system development over several sprints, covering the architecture design, user interface, endpoints, testing, as well as sprint reviews and retrospectives. Finally, conclusions and possible future development lines are presented, followed by the bibliography used.

Keywords

Real estate investment, rental, Angular, Node.js, API REST, MySQL

TABLA DE CONTENIDO

1. INTRODUCCIÓN

En este primer capítulo de la memoria se presenta la situación actual, es decir, el punto de partida necesario para comprender el escenario en el que se desenvuelve el estudio. Además, se establecen claramente los objetivos generales y específicos que se pretende alcanzar con el trabajo. Una vez analizado la situación actual y los objetivos, es crucial establecer el marco teórico para contextualizar adecuadamente la realización del proyecto.

1.1 Situación actual

En la actualidad, los precios de la vivienda en España han alcanzado máximos históricos, superando incluso los máximos registrados durante la burbuja inmobiliaria de 2007. Según el último informe de Idealista, en mayo de 2024 el precio medio del metro cuadrado de vivienda de segunda mano se situó en 2120 euros, con un incremento del 7,3% con respecto al año anterior.

El mercado de la vivienda en España está experimentando presiones significativas debido a una oferta insuficiente. La construcción no ha podido mantener el ritmo de la creciente demanda, especialmente en ciudades como Madrid y Barcelona, lo que contribuye al incremento de los precios. Además, las nuevas leyes destinadas a regular el mercado inmobiliario no cumplen su objetivo, y los precios siguen en ascenso lo que imposibilita la compra de viviendas.

La situación es igual de complicada para el alquiler, ya que también los precios del arrendamiento alcanzan niveles récord. A pesar de esto, el elevado precio para la compra de inmuebles fuerza a que el alquiler sea la única opción. Esto se acrecienta en lo más jóvenes, según el portal inmobiliario pisos.com, el 35% de los pertenecientes a la Generación Z vive de alquiler.

Esta creciente demanda del alquiler tradicional, acompañada también con las nuevas formas de explotación de bienes inmobiliarios con plataformas como Booking o Airbnb, hace que los propietarios sean más estrictos. Por tanto, estos arrendadores, cada vez más exigentes, dan prioridad a inquilinos que garanticen mayor estabilidad legal y capacidad financiera para evitar riesgos de impago. Esta situación, junto a los elevados precios, hace que para familias y grupos en situación de vulnerabilidad sea muy difícil el acceso a la vivienda.

Desde el punto de vista de la ingeniería del *software*, más en concreto orientada a la web, existen multitud de plataformas que ofrecen sus servicios relacionados con el mercado inmobiliario en la web moderna. Las ya nombradas webs de Booking o Airbnb se sitúan como las plataformas líderes para la reserva y alquiler de alojamientos. Otras aplicaciones web como Idealista, pisos.com o Fotocasa, se especializa en la búsqueda y publicación de anuncios de compra, venta y alquiler de bienes inmobiliarios.

En este contexto se sitúa el trabajo fin de máster, que consiste en el desarrollo de una aplicación web para ofrecer un servicio dentro del mercado inmobiliario. A diferencia de las plataformas mencionadas anteriormente, esta aplicación está dirigida a ese propietario de bienes inmuebles, que es cada vez más selectivo debido a la creciente

demanda de alquiler. El propósito de la aplicación es facilitar una gestión proactiva y eficiente de sus propiedades, como se detallará en la siguiente sección.

1.2. Objetivos

Para abordar de manera efectiva el proyecto, es esencial definir claramente los objetivos, ya que estos establecerán las bases y directrices necesarias para el desarrollo de la aplicación. En este sentido, se han definido un objetivo general y varios objetivos específicos que detallan las metas que se pretenden alcanzar con este trabajo.

1.2.1. Objetivo general

El objetivo general de este proyecto es el diseño y desarrollo de una aplicación web innovadora destinada a inversionistas inmobiliarios y, en general, a propietarios de inmuebles que explotan sus propiedades rentándolas.

El resultado que se desea obtener es una nueva plataforma mediante la cual se permita al usuario realizar una gestión proactiva y efectiva de sus propiedades alquiladas, desde un único lugar. Esto implica la administración de ingresos (provenientes de alquileres tradicionales, Airbnb, etc.) así como de los gastos asociados (seguros, cuotas de comunidad, etc.). Además, la aplicación facilitará la gestión de documentos relacionados, la generación de balances financieros con los que poder evaluar la rentabilidad y dispondrá de una ayuda fiscal para realizar la declaración de la renta.

1.2.2. Objetivos específicos

Para alcanzar el objetivo general y demostrar las competencias adquiridas durante el máster, se han definido una serie de objetivos específicos:

- Desarrollar un proyecto *software* desde cero utilizando la metodología ágil Scrum para gestionar las tareas y adaptarse a los cambios y necesidades del proyecto
- Implementar la interfaz de usuario utilizando el *framework* Angular, diseñando y desarrollando los distintos módulos para una navegación eficiente.
- Crear la parte servidora de la aplicación utilizando Node.js. Esto incluirá la definición de la arquitectura del servidor, la implementación de *endpoints* RESTful para la gestión de datos, y la integración con una base de datos MySQL. Se garantizará la seguridad de la aplicación mediante la implementación de autenticación.
- Montar un ecosistema *software* con herramientas para el sistema de control de versiones, con Git y GitHub, conectado con la herramienta de gestión Jira. Y definir un flujo de integración continua, con GitHub Actions, que permita lanzar las pruebas de forma automática ante una liberación de código, así como analizar el código de forma estática con Sonar.
- Configurar un entorno de pruebas tanto unitarias como de integración utilizando el *framework* Jest y la biblioteca Supertest.

1.3. Marco teórico

1.3.1. Metodologías ágiles

En 2001, con el aumento significativo del desarrollo *software*, el modelo de desarrollo en “cascada” demostró ser ineficaz para los equipos de *software* debido a la naturaleza cambiante del trabajo y a la evolución constante de las necesidades de los clientes. Fue

en este contexto que surgieron las metodologías ágiles, formalizadas en el Manifiesto Ágil (*"The Agile Manifesto")*. Este manifiesto, compuesto por 4 valores y 12 principios, transformó radicalmente la gestión de proyectos de desarrollo de *software*.

El enfoque ágil es una metodología iterativa, es decir, se realizan entregas cíclicas en las que se cubre todas las fases del ciclo de desarrollo: desde la recopilación de requisitos, pasando por el diseño, hasta la verificación y la entrega final. Cada una de las iteraciones de este marco de trabajo para la gestión de proyectos se conoce como "*sprint*". Por tanto, a diferencia de los antiguos modelo en cascada, en los procesos ágiles, se entrega valor de manera continua y se recibe retroalimentación a lo largo de todo el proyecto.

Los valores fundamentales de Agile incluyen:

- **Individuos e interacciones sobre procesos y herramientas**: Se destaca la importancia de la comunicación y la colaboración humana.
- ***Software* funcional frente a documentación exhaustiva**: Se centra en la entrega frecuente de *software* operativo.
- **Colaboración con el cliente en lugar de negociación de contratos**: Promueve la participación y el *feedback* continuo del cliente.
- **Responder al cambio sobre seguir un plan**: Subraya la adaptabilidad y flexibilidad ante la evolución de los requisitos.

Las metodologías ágiles engloban varios marcos y prácticas específicas, entre los cuales Scrum es uno de los más ampliamente adoptados.

1.3.1.1 Scrum

Scrum es uno de los marcos ágiles más adoptados, diseñado para facilitar la colaboración de equipos en proyectos complejos. Su objetivo es ayudar a los equipos a trabajar de manera más eficiente y a entregar productos de alta calidad. Trabaja con equipos pequeños multidisciplinares en ciclos iterativos centrados en el cliente, creando productos de forma incremental. La metodología Scrum se organiza en torno a un conjunto de roles, eventos y artefactos que guían el proceso de desarrollo.

Roles:

- **Product Owner**: Es responsable de definir y priorizar el *producto backlog*, asegurándose de que el equipo trabaje en las tareas más valiosas y alineadas con los objetivos del negocio.
- **Scrum Master**: Es la persona que dirige los distintos eventos de Scrum y apoya al equipo en su adopción y prácticas. Elimina impedimentos y asegura que el equipo siga los principios y prácticas de Scrum.
- **Equipo de desarrollo**: Es un grupo autoorganizado, multifuncional y colaborativo que trabaja en la creación de los incrementos del producto.

Eventos:

- ***Sprint***: Es el corazón de Scrum, un ciclo de trabajo que dura entre una y cuatro semanas, en el cual se desarrolla un incremento del producto que sea potencialmente entregable.

- ***Sprint planning***: Reunión al inicio de cada *sprint* donde el equipo planifica el trabajo a realizar durante el *sprint*, definiendo el objetivo del *sprint* y seleccionando los elementos del *backlog* del producto que se van a desarrollar.
- ***Daily***: Reunión diaria de 15 minutos donde el equipo de desarrollo sincroniza actividades y planifica las próximas horas, identificando posibles obstáculos.
- ***Sprint review***: Al final del *sprint*, el equipo presenta el trabajo completado a los *stakeholders* (partes interesadas) y recopila *feedback* que se incorpora en el siguiente *sprint*.
- ***Sprint restrospective***: Después de la revisión del *sprint*, el equipo reflexiona sobre el proceso y discute posibles mejoras para el próximo *sprint*.

Artefactos:

- ***Product backlog***: Es una lista ordenada de todo lo que podría ser necesario en el producto y es la única fuente de requisitos para cualquier cambio a realizar en el producto.
- ***Sprint backlog***: Es la lista de tareas seleccionadas del *product backlog* para completarse en el *sprint*, junto con un plan para entregar el incremento del producto y alcanzar el objetivo del *sprint*.
- **Incremento**: Es la suma de todos los elementos del *product backlog* completados durante un *sprint* y los *sprints* anteriores. Cada incremento debe ser un producto funcional que cumpla con la definición de "terminado" (DoD, D*efinition of Done*) del equipo.

Como herramienta para Scrum se utiliza el *Burn-down*, que consiste en un gráfico utilizado para mostrar el trabajo pendiente del *sprint.* En el gráfico, las unidades de trabajo se representan en el eje vertical, mientras que el tiempo se muestra en el eje horizontal. A medida que avanza el *sprint*, la cantidad de trabajo pendiente disminuye.

1.3.2. Tecnologías y herramientas

1.3.2.1. Desarrollo Front-End: Angular

Angular es un *framework* para el desarrollo de aplicaciones web de una sola página sofisticadas y de alto rendimiento. Mantenido por un equipo dedicado en Google, Angular ofrece un conjunto amplio de herramientas, APIs y bibliotecas que simplifican y optimizan el flujo de trabajo de desarrollo.

Con una arquitectura basada en componentes, Angular fomenta la reutilización de código y permite una separación clara de responsabilidades. Utiliza TypeScript, un superset de JavaScript con tipado estático, que mejora la robustez y la mantenibilidad del código, proporcionando además características avanzadas como interfaces y decoradores. Angular también implementa un sistema de inyección de dependencias que facilita la gestión de servicios y componentes, promoviendo una arquitectura modular y escalable.

El ecosistema de Angular incluye herramientas esenciales como el CLI (*Command Line Interface*), que permite a los desarrolladores generar, configurar y gestionar proyectos de manera efectiva. Angular también soporta la gestión avanzada de rutas, la validación de formularios y la comunicación con APIs, integrando fácilmente funcionalidades complejas en las aplicaciones.

En este proyecto se combina Angular con Bootstrap, un popular *framework* de CSS, que permite a los desarrolladores crear interfaces de usuario atractivas de manera sencilla. Bootstrap proporciona una amplia gama de componentes predefinidos y estilos que pueden ser fácilmente integrados en una aplicación Angular, acelerando el proceso de diseño y asegurando una apariencia coherente y profesional. La integración de Bootstrap con Angular permite aprovechar las fortalezas de ambos *frameworks*: la modularidad y robustez de Angular, junto con el diseño intuitivo y adaptativo de Bootstrap.

1.3.2.2. Desarrollo Back-end: Node.js, npm y JavaScript

Node.js es un entorno de ejecución de código abierto que permite a los desarrolladores ejecutar JavaScript del lado del servidor. Utiliza el motor V8 de Google Chrome para ejecutar JavaScript de manera rápida y eficiente fuera del contexto del navegador web. Fue creado en 2009, y desde entonces ha ganado popularidad debido a su capacidad para construir aplicaciones escalables y de alto rendimiento.

Uno de los componentes clave de Node.js es npm (*Node Package Manager*), el gestor de paquetes predeterminado que facilita la instalación, actualización y gestión de bibliotecas y dependencias. Con una vasta colección de paquetes disponibles, npm permite a los desarrolladores ampliar fácilmente la funcionalidad de sus aplicaciones, promoviendo la reutilización de código y la colaboración comunitaria. La integración de npm en el flujo de trabajo de desarrollo agiliza la gestión de dependencias y la configuración del entorno, aumentando la productividad.

El desarrollo Back-End con Node.js se basa en JavaScript, un lenguaje ampliamente conocido y utilizado tanto en el lado del cliente como en el del servidor. Para potenciar aún más la eficiencia y la organización del código, se utiliza el *framework* Express, una herramienta minimalista y flexible que facilita la creación de aplicaciones web y APIs robustas. Express simplifica la gestión de rutas y el manejo de solicitudes y respuestas HTTP, permitiendo a los desarrolladores centrarse en la lógica de negocio y en la creación de funcionalidades avanzadas.

1.3.2.3. Base de datos: MySQL

MySQL es un sistema de gestión de bases de datos relación (RDBMS, por sus siglas en inglés) muy popular y ampliamente utilizado en el desarrollo de aplicaciones web y empresariales. MySQL es ampliamente reconocido por su confiabilidad, velocidad y facilidad de uso, lo que lo convierte en una elección común para aplicaciones que requieren almacenamiento estructurado y consultas complejas.

Se utiliza Sequelize como ORM (Object-Relational Mapping), una herramienta que facilita la interacción con la base de datos MySQL desde Node.js. Sequelize simplifica la manipulación de datos al mapear objetos JavaScript a tablas de la base de datos y viceversa, permitiendo a los desarrolladores utilizar métodos y consultas de alto nivel en lugar de SQL directo. Esto promueve un desarrollo más rápido y estructurado, además de mejorar la mantenibilidad del código al abstraer detalles de la base de datos.

La combinación de MySQL como sistema de gestión de base de datos, Sequelize como ORM, sumado a la herramienta HeidiSQL para administración y visualización de la base de datos MySQL, proporciona un entorno robusto y productivo para el almacenamiento y manipulación de datos en este proyecto, garantizando tanto la seguridad de los datos como el rendimiento óptimo de la aplicación.

1.3.2.4. Control de versiones: Git y GitHub

Git es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo *software* por su rendimiento, seguridad y flexibilidad. Permite a los desarrolladores trabajar en paralelo mediante ramas y fusiones, facilitando la colaboración e integración de cambios. Además, Git utiliza un sistema de confirmaciones que crea instantáneas del estado del proyecto en momentos específicos, permitiendo revertir cambios y mantener un historial detallado del desarrollo.

Por su parte, GitHub es una plataforma de alojamiento de código que utiliza el sistema de control de versiones Git. GitHub permite a los desarrolladores almacenar, compartir y colaborar en proyectos de software, facilitando la gestión de versiones y la integración de cambios mediante características como *pull requests* y revisiones de código. La plataforma también ofrece la integración con servicios de terceros, por ejemplo, en este proyecto, con Jira para la conexión de las actividades de desarrollo *software* con la gestión del proyecto, o con SonarCloud para el análisis estático de código.

1.3.2.5. Integración continua: GitHub Actions

GitHub Actions se trata de una plataforma para la integración continua (CI) y entrega continua (CD) gracias a la cual se pueden automatizar flujos de trabajo directamente desde los repositorios de GitHub. Con GitHub Actions, se puede definir *pipelines* mediante archivos YAML, especificando una serie de pasos que se ejecutan en respuesta a eventos en el repositorio (como *commits*, *pull requests*, etc.)

Realizar la automatización del proyecto con GitHub Actions, facilita la detección temprana de errores, la ejecución de pruebas y la implementación de aplicaciones, mejorando la calidad del software y acelerando el ciclo de desarrollo. Además, GitHub Actions ofrece una amplia gama de acciones predefinidas y la posibilidad de crear acciones personalizadas, lo que proporciona flexibilidad y adaptabilidad a las necesidades específicas del proyecto.

1.3.2.6. Pruebas automatizadas: Jest

El *framework* de pruebas Jest, desarrollado por Facebook, está diseñado para realizar pruebas unitarias y de integración en JavaScript. Jest destaca por su facilidad de uso, su capacidad para realizar pruebas asíncronas y su integración con otras herramientas y bibliotecas del ecosistema JavaScript.

Además de Jest, se emplea la biblioteca Supertest para realizar pruebas de integración en las APIs desarrolladas con Node.js y Express. Supertest facilita la simulación de solicitudes HTTP y la verificación de las respuestas del servidor, permitiendo a los desarrolladores asegurar que las distintas partes de la aplicación interactúan correctamente entre sí. De esta forma, el entorno de pruebas automatizadas garantiza que tanto las unidades individuales de código como las integraciones entre componentes funcionen según lo esperado.

1.3.2.7. Análisis estático: Sonar

SonarCloud es una plataforma de análisis de código basada en la nube que permite a los desarrolladores detectar problemas de calidad y seguridad en su código fuente. Esta herramienta proporciona análisis automáticos de ramas y *pull requests*, ofreciendo informes detallados sobre *bugs*, vulnerabilidades y malas prácticas de codificación. SonarCloud soporta una amplia gama de lenguajes de programación y se integra

fácilmente con plataformas de DevOps, en el caso de este proyecto, con GitHub Actions.

2. INCEPTION DECK

Una vez descrita la situación actual, los objetivos del proyecto y el marco teórico, en este segundo capítulo se realiza el *inception deck*, también conocido como *agile inception*. El *inception deck* agrupa una serie de técnicas diseñadas para guiar el inicio del proyecto y alinear a todas las personas involucradas en él, con el objetivo de mitigar incertidumbres, detectar los riesgos más notorios y alinear las expectativas.

2.1. Motivación

El primer apartado del *inception deck* se trata de responder a la pregunta de "¿por qué estamos aquí?", para descubrir el motivo principal del proyecto.

La gestión de una o varias propiedades, además de todo el tema de la explotación de las mismas, puede resultar algo complejo y exigir un considerable esfuerzo por parte del propietario. La gran cantidad de ingresos y gastos derivados del alquiler, hipoteca, seguro de la vivienda, facturas, recibos del ayuntamiento, averías, entre otros, genera una multiplicidad de operaciones y documentos que se incrementa exponencialmente con el número de propiedades

Para enfrentar estos desafíos, se busca llevar un control exhaustivo de las propiedades, analizar su rentabilidad, mantener todos los documentos organizados y recibir asistencia para la declaración de la renta, todo ello sin depender de gestorías o inmobiliarias, evitando así sus costos adicionales. Con esta motivación, se quiere desarrollar una aplicación web diseñada específicamente para personas que poseen bienes inmuebles y desean gestionarlos de manera sencilla, efectiva y cómoda.

2.2. Elevator pitch

La segunda actividad se trata del discurso del ascensor, también conocido como *elevator pitch*, que se trata de una presentación breve con el objetivo de persuadir al interlocutor para captar su atención y crear una oportunidad de contacto o negocio.

"¿Te has sentido abrumado por la gran cantidad de papeles derivados del alquiler de tus propiedades? ¿Has deseado poder llevar un control de tus ingresos y gastos sin tener que buscar durante horas entre carpetas?

Bienvenido a nuestra web, donde puedes mantener todos tus documentos organizados, analizar la rentabilidad de tus inversiones y hasta realizar la declaración de la renta, todo en un solo lugar y sin complicaciones.

Ya no tendrás que pagar costosas gestorías para que administren tus bienes. Con esta plataforma, podrás tomar el control de tus activos inmobiliarios de manera autónoma y totalmente gratis, sin los costos asociados a intermediarios."

2.3. Caja del producto

En este apartado se pretende diseñar la caja del producto como si fuese a ser empaquetado y expuesto en una tienda física, junto con sus eslóganes principales. Con este concepto abstracto, relacionado con el marketing, se puede revelar información importante acerca del producto, así como dar una perspectiva directa y visual.

Eslóganes:

- ✓ Centraliza la gestión de tus propiedades alquiladas
- ✓ Organiza tus facturas y documentos sin complicaciones
- ✓ Visualiza la rentabilidad con balances claros

Propuesta de diseño para la caja del producto:

Logos:

2.4. Lista de noes

Con la lista de "noes" se identifica claramente lo que no incluirá el proyecto. Se establecen límites para mantener el enfoque en las metas principales.

Para ello se construye la siguiente tabla con tres zonas principales: en alcance, fuera de alcance y por decidir.

En alcance	Por decidir	Fuera de alcance
Gestión de usuarios	Ayuda en declaración de la renta	Creación de nuevos roles
Gestión de propiedades	Filtrado y búsqueda avanzada	Comparativa de rentabilidades
Gestión de operaciones		Exportación de datos
Generación de balances		Chat en tiempo real con experto
Gestión de documentos		

En alcance:

- **Gestión de usuarios**: Se realizará la gestión de los usuarios de la aplicación de forma que se pueda dar de alta, modificar y dar de baja usuarios, así como garantizar la seguridad con mecanismo de autenticación y autorización.
- **Gestión de propiedades**: Un usuario podrá ver sus propiedades, así como crearlas, editarlas y borrarlas. Además, un usuario podrá ver sus propiedades, tanto en un listado, como en un mapa integrando Google Maps.
- **Gestión de operaciones**: Un usuario puede ver, crear, editar y borrar operaciones (ingresos/gastos) asociadas a sus propiedades.
- **Generación de balances**: Un usuario podrá generar balances asociados a sus propiedades en un rango de fechas para consultar la rentabilidad de esa propiedad en un determinado tiempo.
- **Gestión de documentos**: Un usuario dispondrá de un repositorio donde subir, ver, descargar y borrar documentos relacionados con sus propiedades como facturas contratos, etc.

Por decidir:

- **Ayuda en declaración de la renta**: Un usuario podrá solicitar ayuda para realizar la declaración de la renta gracias a las operaciones adjuntadas.
- **Filtrado y búsqueda avanzada**: Implementar opciones de filtrado y búsqueda para que los usuarios puedan encontrar fácilmente propiedades, operaciones, documentos… basados en determinados criterios.

Fuera de alcance:

- **Creación de nuevos roles**: Se incluyen nuevos roles en la aplicación, como administrador, con permisos totales para la gestión de usuarios, propiedades, etc.

Así como de gestor, que se entiende como un usuario más avanzado con la idea de que pueda asesorar a múltiples clientes

- **Comparativa de rentabilidades**: Los usuarios a parte de generar balances para controlar la rentabilidad de sus inmuebles, podrán compararlos para extraer más información. Esto extiende la funcionalidad de generación de balances, pudiendo no solo crearlos, si no generar 2 balances simultáneos para la misma propiedad o diferentes, dentro de un rango de fechas.
- **Exportación de datos:** Los usuarios pueden exportar los distintos datos introducidos en la aplicación (propiedades, operaciones, documentos, balances…) al formato más adecuado en cada caso (pdf, xls…).
- **Chat en tiempo real con experto**: Los usuarios podrán contactar a través de un chat con un experto inversionista para asesorías y preguntas.

2.5. Conoce a tus vecinos

Este apartado se identifica todos los agentes involucrados en el proyecto, se enfatiza la importancia de la colaboración y la interacción entre partes para el éxito conjunto.

Se identifican los actores involucrados en el proyecto:

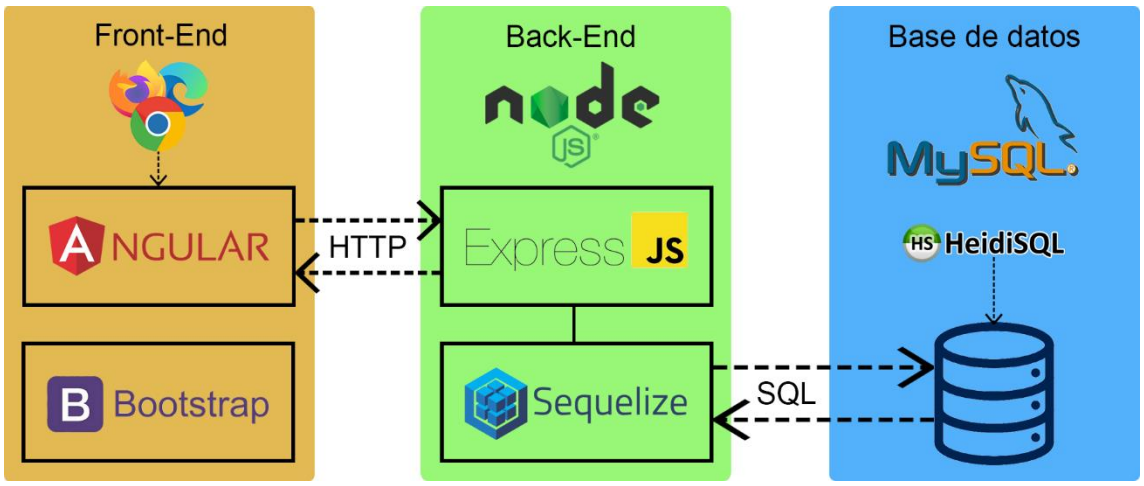
- **Equipo de desarrollo**: Está compuesto por un único desarrollador quién se encargará del desarrollo de las historias de usuario, además de un tutor quien desempeñará el papel de asesor y evaluará la amplitud del proyecto.
- **Equipo de diseño**: El equipo de UI/UX, coincide con el equipo de desarrollo. Velará por la usabilidad y experiencia del usuario diseñando la interfaz de la aplicación web.
- ***Product owner***: Encargado de validar el funcionamiento de la aplicación y proponer nuevas funcionalidades. Representa a los usuarios.
- **Usuarios**: Usuarios finales de la aplicación, perfiles de inversionistas inmobiliarios.

En el caso de que la aplicación se comercializase, sería necesario ampliar los involucrados:

- **Equipo de marketing:** Será responsable de promocionar la aplicación con el objetivo de atraer a nuevos usuarios e impulsar la adopción de la plataforma.
- **Equipo de sistemas:** Se encargará de administrar la infraestructura tecnológica necesaria para mantener la aplicación web en funcionamiento. Alojamiento de servidores, la seguridad de la información, la gestión de bases de datos, etc.
- **Equipo de finanzas:** Se ocupará de la gestión financiera del proyecto para garantizar la sostenibilidad económica y el éxito financiero a largo plazo de la aplicación.
- **Equipo de soporte:** Proporcionará asistencia técnica y atención al cliente a los usuarios de la aplicación para que tengan una experiencia positiva.

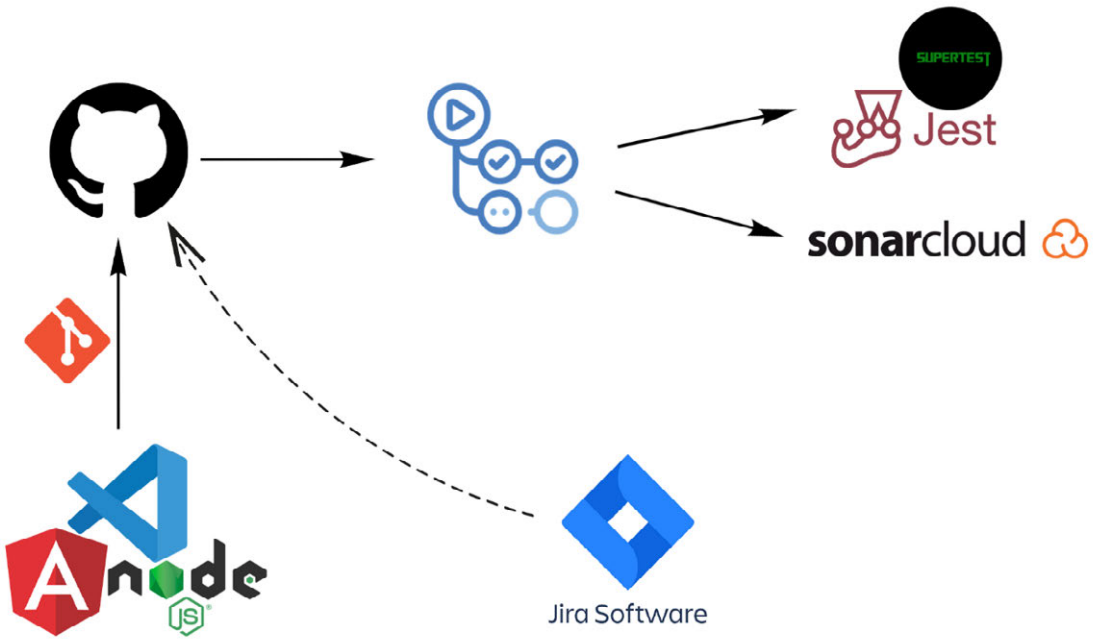
2.6. Haz ver la solución

En esta sección, el equipo presenta una visión inicial de la solución, incluyendo la arquitectura, tecnologías y herramientas seleccionadas para el desarrollo e integración de la aplicación.



En el diagrama anterior se muestra el *stack* tecnológico de la aplicación dividido en tres partes:

- **Front-End:** El *framework* elegido es Angular, se utiliza la versión 16, es decir, la penúltima en el momento de realización del proyecto. Adicionalmente, se incorpora la biblioteca de Bootstrap para facilitar el diseño de la aplicación web. Se utiliza Visual Studio Code como IDE para el desarrollo.
- **Back-End:** Se elige Node.js como entorno por la gran cantidad de recursos y opciones disponibles y su creciente demanda en el mundo laboral. En concreto, se utiliza Express como *framework* para el desarrollo del Back-End como una API REST con la que se comunicará la parte Front-End. Para interactuar con la base de datos, se propone utilizar Sequelize como ORM. Como IDE también se utiliza Visual Studio Code.
- **Base de Datos:** Se elige MySQL para la persistencia de datos en una base de datos relacional. Como gestor de base de datos se utiliza HeidiSQL.



Para la gestión de versiones se utilizará Git, creando dos repositorios de código, uno para el Front-End y otro para el Back-End. Como repositorio remoto se elige GitHub, la versión SaaS (*Software as a Service*).

En cuanto a la gestión del proyecto se realiza mediante Jira Software, herramienta que se conecta con los dos repositorios de código de GitHub para mayor trazabilidad.

En cuanto a la integración continua se realiza mediante *workflows* de GitHub Actions, uno para cada repositorio. En el flujo de integración continua se llevan a cabo varias tareas: la construcción del proyecto; ejecución de pruebas, en el caso del Back-End unitarios y de integración realizados con el *framework* Jest y la biblioteca Supertest; y análisis estático de código, mediante SonarCloud, servicio de análisis de código basado en la nube.

2.7. Riesgos e incertidumbres

Con esta dinámica se debe detectar todo aquello que pueda suponer un riesgo para el proyecto, es decir, aquellas preocupaciones que nos pueden mantener despiertos durante la noche (*keep us at night*). De esta manera, identificando esos riesgos e incertidumbres, hace más sencillo manejarlos y ser conscientes de ellos durante el desarrollo del proyecto.

Estos riesgos se pueden dividir en dos en función del control que se puede tener sobre ellos. Por tanto, serán riesgos sobre los que podemos influir, si tenemos cierto control para acometerlos y solventar los problemas que presenten; mientras que, si no hay ese control, serán riesgos sobre los que no podemos influir, y por tanto no debe preocuparnos en exceso.

2.7.1. Riesgos en los que podemos influir

- **Dominio de las tecnologías y herramientas:** Se trata del principal riesgo de este proyecto al tratarse de un proyecto final de máster. Para mitigarlo se realiza una investigación sobre las tecnologías con menos conocimiento.
- **Tiempo:** Existe un tiempo limitado para este proyecto, además de que el equipo de desarrollo tiene otra ocupación al margen de este proyecto. Será necesario organizar muy bien semana a semana, todo lo que se quiere realizar para no perder el poco tiempo del que se dispone.
- **Estimaciones:** Muy relacionado con los dos riesgos anteriores. Al carecer de un gran dominio del *stack* tecnológico y tiempo limitado, las malas estimaciones de las historias de usuario pueden perjudicar el desarrollo. Para mitigarlo, en la medida de lo posible, se tendrá que ser cauto con las primeras estimaciones, a la par que un aprendizaje continuo de las estimaciones según avanza el proyecto.
- **Diseño de la arquitectura:** Un mal diseño de la arquitectura puede comprometer todo el proyecto. Por tanto, hay que presentarle especial atención y consultar al tutor, más experimentado, en la medida de lo posible.
- **Calidad del *software*:** Si no se le da la importancia que merece durante todo el proyecto, el código puede degenerar rápidamente.

2.7.2. Riesgos en los que no podemos influir

- **Problemas en las tecnologías y herramientas utilizadas**: Debidos a fallos, limitaciones o incompatibilidades.

- **Enfermedad o baja de cualquier tipo:** En cualquier equipo existen bajas, pero en este caso que solo existe un desarrollador, aun toma más importancia.

2.8. Planificación global

Para comenzar un proyecto es crucial realizar una planificación del mismo. Esta sección se centra en proporcionar una estimación de alto nivel del tiempo y esfuerzo que puede requerir el proyecto. Aunque no se busca una precisión exacta, se pretende obtener una aproximación para que todos los involucrados tengan una idea de la magnitud de lo que se desea construir. Las épicas que se incluyen son las descritas en el apartado 2.4 con la etiqueta "en alcance" o "por decidir", entendiendo que las calificadas como "fuera de alcance" están más allá de esta planificación global.

En la siguiente tabla se reparten 1000 unidades de valor (UV) entre las épicas para priorizar los distintos módulos. Además, se realiza una estimación asignando puntos de esfuerzo (PE), para determinar el tiempo aproximado que se tardaría en realizar cada una de las épicas, teniendo en cuenta que 1 PE corresponde con 3 horas de trabajo.

Épica	UV	PE
Gestión de usuarios	310	14
Gestión de propiedades	280	18
Gestión de operaciones	140	8
Generación de balances	130	12
Gestión de documentos	70	10
Ayuda en declaración de la renta	60	10
Filtrado y búsqueda avanzada	10	8

Además, para el cálculo de esfuerzo, es necesario tener en cuenta los puntos de esfuerzo de que se van a dedicar a hitos que no aportan unidades de valor.

Hito	PE
Estudio y preparación previa	8
Gestión del proyecto (Incluye reuniones, correcciones, preparación de *sprints*, etc.)	6
Desarrollo de historias técnicas	6
Documentación	20

Sumando todos los puntos de esfuerzo se obtiene un total de 118 PE, con la equivalencia de 3 horas por punto de esfuerzo, se traduce en 360 horas.

Se calcula que se van a dedicar 25 horas semanales. A excepción del *sprint* 0 que constará de una semana para abordar las historias técnicas, los *sprints* serán de dos semanas, por tanto 50 horas de dedicación. Entonces el proyecto concluirá en el *sprint* 6,7 (7). En la siguiente tabla se especifica las fechas con más detalle.

Sprint	Fecha de inicio	Fecha de fin
0	1 de abril de 2024	7 de abril de 2024
1	8 de abril de 2024	21 de abril de 2024

2	22 de abril de 2024	5 de mayo de 2024
3	6 de mayo de 2024	19 de mayo de 2024
4	20 de mayo de 2024	2 de junio de 2024
5	3 de junio de 2024	16 de junio de 2024
Revisión y documentación	17 de junio de 2024	**25 de junio de 2024**
ENTREGA TFM		
6	1 de julio de 2024	14 de julio de 2024
7	15 de julio de 2024	28 de julio de 2024
FIN DEL PROYECTO		

Hacer hincapié en que el proyecto se lleva a cabo con una metodología ágil y por tanto esto se trata de una estimación, no un compromiso. Las funcionalidades iniciales pueden variar, variando la estimación dada. El único compromiso existente es la fecha de entrega del TFM.

2.9. Factores de ejecución

En este apartado se trata de priorizar los factores que se consideran clave en la ejecución del proyecto, con el objetivo de, tener claro cuáles son las opciones en el caso de que no todo salga como se desea, es decir, con máxima calidad, cumpliendo con todo el alcance y en con el tiempo y coste establecidos.

Prioridad	Factores clave	Razón
1	Calidad	La calidad del código es algo innegociable es un proyecto *software*. El código debe ser de calidad en favor de su mantenibilidad, escalabilidad, eficiencia, etc.
2	Tiempo	Existe una fecha fija para la entrega del proyecto que debe cumplirse, se convierte también en algo imprescindible.
3	Experiencia del usuario	La experiencia del usuario debe ser buena, simple e intuitiva, pero teniendo en cuenta que la funcionalidad está por encima.
4	Seguridad	La seguridad en caso de comercialización debería ser irrenunciable al tratar datos comprometidos, pero en el ámbito de un proyecto universitario se puede ser más flexible.
5	Alcance	No es algo clave, se podrá prescindir o añadir funcionalidades.
6	Presupuesto	No existe un presupuesto definido, ni expectativas de comercialización.

2.10. Costes

En el último apartado del *inception deck* se pretende calcular los costos y gastos necesarios para llevar a cabo el proyecto. Para ello se hace una planificación detallada de los recursos necesarios, incluyendo los recursos humanos, que son las horas invertidas por el personal que participa en la realización del trabajo, y los costos materiales, que incluyen los gastos físicos. Además, hay que tener en cuenta los gastos por licencias y tecnologías. No se tienen en cuenta los gastos secundarios, como internet o luz, ya que se tendrían independientemente de este proyecto.

2.10.1. Recursos humanos

Puesto asumido	Salario anual	Salario por hora	Horas (totales)	Total
Desarrollador	40.000€	17,86€	300	5.358,00€
Digital Product Manager	80.000€	35,72€	24	857,28€
				6.215,28€

2.10.2. Costos materiales

Producto	Precio	Tiempo amortización (años)	Amortización al mes	Meses	Coste final
MSI GS65 Stealth Thin 8RE	1599€	6	22,21€	3	66,63€
PC Tutor	1000€	6	13,89 €	3	41,67€
Hardware (Auriculares, monitores, ratón, teclado, etc.)	600€	5	10,00€	3	30,00€
					138,30€

2.10.3. Licencias

Tecnología/Herramienta	Licencia	Coste
Node.js	Código abierto	0,00€
Angular	Código abierto. Mantenido por Google	0,00€
GitHub	Se utiliza la cuenta gratuita, que además dispone de ciertos minutos gratis para CI	0,00€
Microsoft Office 365	Licencia Office 365 Education (UPM)	0,00€
Jira	Plan Free con limitaciones	0,00€
MySQL	Licencia GPL	0,00€
Visual Studio Code	Editor gratuito de Microsoft	0,00€
		0,00€

2.10.4. Resumen

Descripción del gasto	Total
Recursos humanos	6.215,28€
Costes materiales	138,30€
Licencias	0,00€
	6353,58€

Por ende, se estima que la realización del proyecto supone un coste final de **SEIS MIL TRESCEINTOS CINCUENTA Y TRES CON CINCUENTA Y OCHO EUROS** (6353,58€).

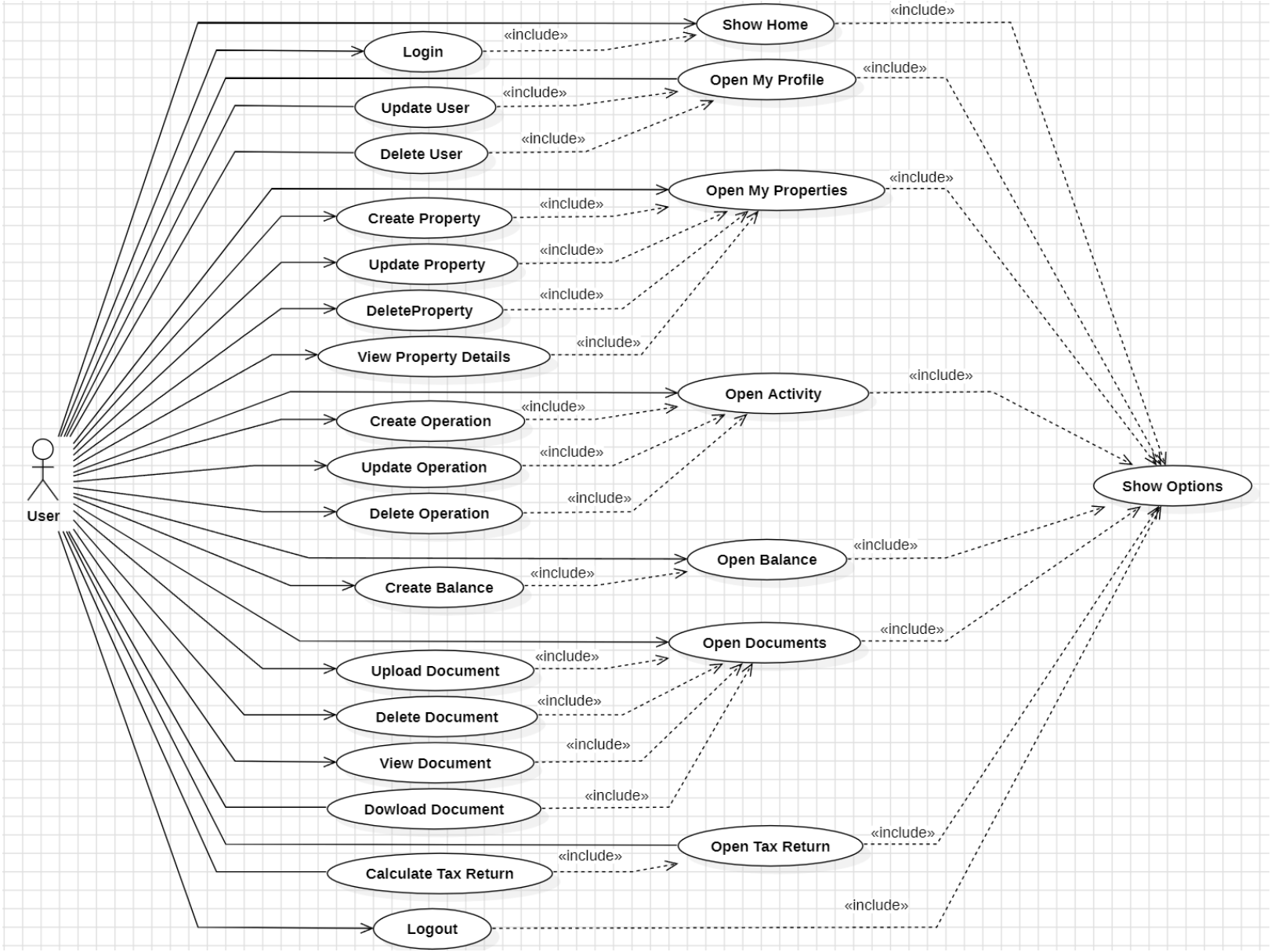
3. PRODUCT BACKLOG

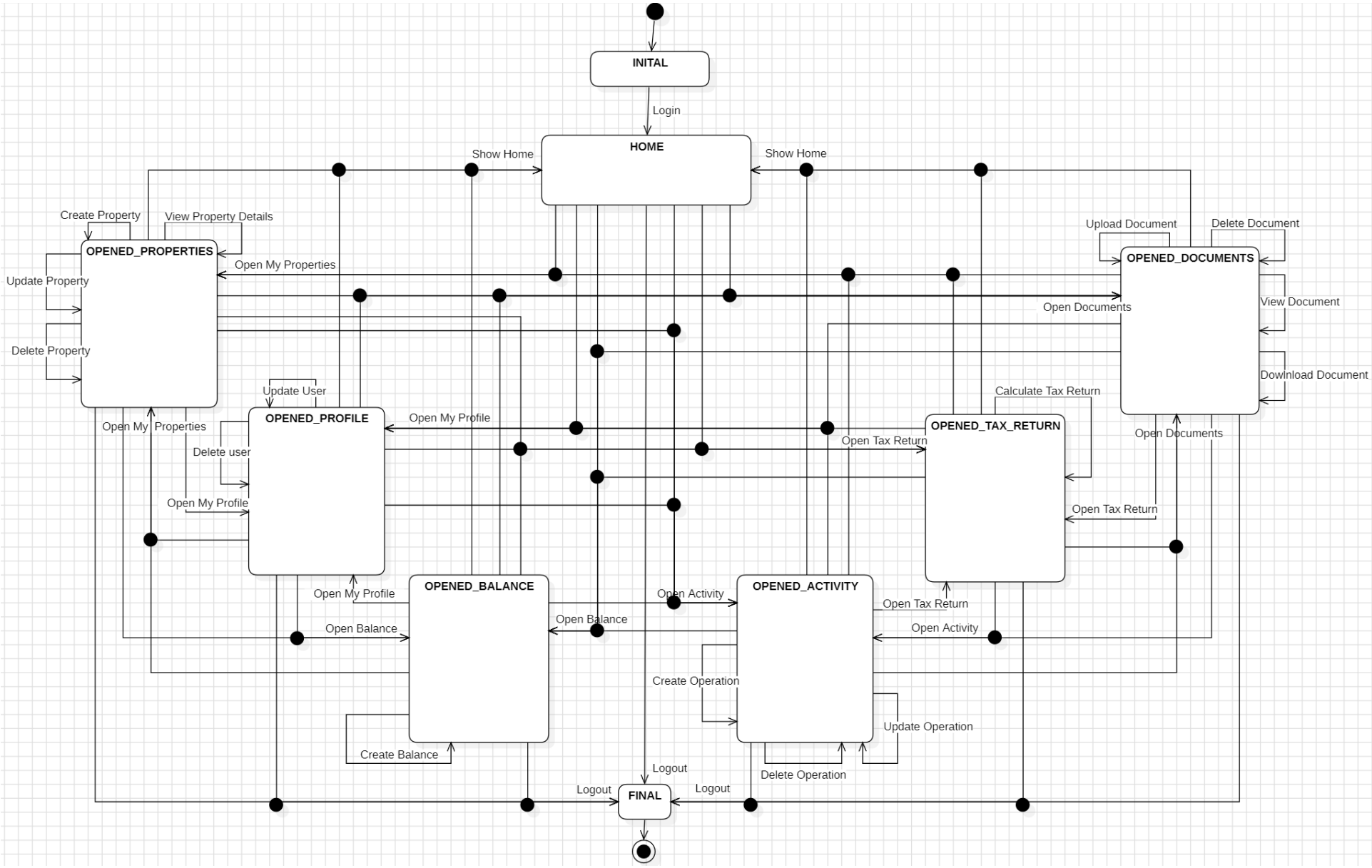
En este tercer capítulo se desea obtener el *product backlog*, es decir, una lista priorizada de tareas o requisitos que deben completarse para desarrollar el producto. Para lograr esto, primero se identifican los casos de uso en un diagrama de casos de uso, donde se muestran las distintas interacciones entre los usuarios y el sistema. Cada caso de uso representa un conjunto de acciones que los usuarios pueden realizar, y cada actor representa a los usuarios u otros sistemas que interactúan con el sistema.

El diagrama de casos de uso es esencial para comprender los requisitos funcionales desde la perspectiva del usuario y garantizar que todas las necesidades sean capturadas de manera exhaustiva. Además, se realiza el diagrama de contexto de casos de uso, que se trata de un diagrama de estados donde se muestra cómo los casos de uso se relacionan con otros casos de uso a través de estados intermedios. Este diagrama proporciona una visión clara de las transiciones entre diferentes estados del sistema y cómo las acciones de los usuarios influyen en estos estados.

Una vez que los casos de uso han sido identificados y representados en estos diagramas, se procede a descomponer cada caso de uso en historias de usuario. Las historias de usuario se crean siguiendo el formato "Como [actor], quiero [funcionalidad] para [beneficio]", lo que asegura que cada historia esté centrada en el usuario y su valor. Este enfoque permite que las necesidades y expectativas de los usuarios se traduzcan en tareas específicas y priorizadas que el equipo de desarrollo debe abordar.

A continuación, se presentan los diagramas de casos de uso y el diagrama de contexto con casos de uso, seguidos de la tabla con el *product backlog* detallado.


User
Show Home
Login
Open My Profile
Update User
Delete User
Open My Properties
Create Property
Update Property
DeleteProperty
View Property Details
Open Activity
Create Operation
Update Operation
Delete Operation
Open Balance
Create Balance
Open Documents
Upload Document
Delete Document
View Document
Dowload Document
Open Tax Return
Calculate Tax Return
Logout
Show Options
«include»


INITAL
Login
HOME
Show Home
Show Home
Create Property
View Property Details
OPENED_PROPERTIES
Update Property
Delete Property
Open My Properties
Upload Document
Delete Document
OPENED_DOCUMENTS
Open Documents
View Document
Download Document
Update User
OPENED_PROFILE
Open My Properties
Delete user
Open My Profile
Open My Profile
Calculate Tax Return
OPENED_TAX_RETURN
Open Tax Return
Open Documents
Open Tax Return
Open My Profile
OPENED_BALANCE
Open Balance
Open Balance
Open Activity
OPENED_ACTIVITY
Open Tax Return
Open Activity
Create Operation
Update Operation
Delete Operation
Create Balance
Logout
Logout
Logout
FINAL

Grupo	ID	Título	Descripción	Prioridad
Requisitos no funcionales previos (Épica TFM-4)	TFM-3	Configuración del proyecto Front-End	**Como** desarrollador, **quiero** disponer del proyecto de Angular creado, configurado y ligado a un repositorio de código de GitHub, **para** comenzar con el desarrollo Front-End.	Muy alta
	TFM-5	Añadir flujo de integración continua para proyecto Front-End	**Como** desarrollador, **quiero** añadir integración continua con GitHub Actions al proyecto de Angular, **para** automatizar la construcción, pruebas y análisis de código con SonarCloud.	Muy alta
	TFM-6	Configuración del proyecto Back-End	**Como** desarrollador, **quiero** disponer del proyecto de Node.js creado, configurado y ligado a un repositorio de código de GitHub, **para** comenzar con el desarrollo Back-End.	Muy alta
	TFM-7	Añadir flujo de integración continua para proyecto Back-End	**Como** desarrollador, **quiero** añadir integración continua con GitHub Actions al proyecto de Node.js, **para** automatizar la construcción, pruebas y análisis de código con SonarCloud.	Muy alta
	TFM-8	Configuración de la base de datos	**Como** desarrollador, **quiero** configurar una base de datos MySQL y conectarla con el proyecto de Node.js, **para** persistir datos del proyecto.	Muy alta
	TFM-9	Configuración del entorno de pruebas	**Como** desarrollador, **quiero** configurar el entorno de pruebas unitarias y de integración con Jest, SuperTest, **para** verificar el correcto funcionamiento del código.	Muy alta
Gestión de usuarios (Épica TFM-10)	TFM-11	Registro de usuario	**Como** cliente a utilizar la aplicación, **quiero** dar de alta un nuevo usuario, **para** poder acceder a las funcionalidades de la aplicación.	Muy alta

	TFM-12	Autenticación de usuario	**Como** usuario dado de alta, **quiero** iniciar y cerrar sesión en la aplicación, **para** acceder a mis datos personales, interactuar con las funcionalidades y garantizar la privacidad y seguridad de mi cuenta	Muy alta
	TFM-13	Consultar "Mi perfil"	**Como** usuario autenticado, **quiero** tener una funcionalidad de "mi perfil", **para** consultar los datos de mi usuario y tener acceso a funcionalidades de edición o borrado de mi usuario	Alta
	TFM-14	Editar usuario	**Como** usuario en "mi Perfil", **quiero** poder editar mi usuario, **para** actualizar o corregir mis datos.	Alta
	TFM-15	Eliminar usuario	**Como** usuario en "mi Perfil", **quiero** poder eliminar mi usuario, **para** dejar de ser usuario de la aplicación.	Alta
	TFM-16	Diseño de la página de inicio	**Como** desarrollador, **quiero** que la aplicación tenga un "*home*" accesible con o sin autenticación, **para** que mejorar la interfaz y que exista una buena experiencia de usuario	Media
Gestión de propiedades (Épica TFM-23)	TFM-18	Abrir "Mis propiedades"	**Como** usuario autenticado, **quiero** tener una funcionalidad de "mis propiedades" accesible desde la cabecera, **para** listar mis propiedades y tener acceso a funcionalidades de visualización, creación, edición y borrado de propiedades	Alta
	TFM-19	Incorporación de mapa con la disposición de las propiedades	**Como** usuario en "mis propiedades", **quiero** tener un mapa donde se disponen las propiedades, **para** situarlas con facilidad.	Media
	TFM-20	Crear propiedad	**Como** usuario en "mis propiedades", **quiero** poder crear nuevas propiedades, **para** su gestión y utilización del resto de funcionalidades	Alta

	TFM-21	Editar propiedad	**Como** usuario en "mis propiedades", **quiero** poder editar propiedades, **para** actualizar y corregir datos de estas.	Alta
	TFM-22	Eliminar propiedad	**Como** usuario en "mis propiedades", **quiero** poder eliminar propiedades, **para** dejar de operar con ellas dentro de la aplicación web.	Alta
Gestión de operaciones (Épica TFM-29)	TFM-25	Abrir "actividad" (operaciones)	**Como** usuario autenticado, **quiero** tener una funcionalidad de actividad **para** listar las operaciones de una propiedad y tener acceso a funcionalidades de creación, edición y borrado operaciones (ingresos y gastos)	Alta
	TFM-26	Crear operación	**Como** usuario en "actividad", **quiero** poder crear nuevos ingresos y gatos, **para** registrar cualquier transacción financiera asociada a una de mis propiedades	Alta
	TFM-27	Editar operación	**Como** usuario en "actividad", **quiero** poder editar operaciones, **para** actualizar información o realizar ajustes necesarios en las transacciones registradas.	Media
	TFM-28	Eliminar operación	**Como** usuario en "actividad", **quiero** poder eliminar operaciones, **para** para eliminar registros incorrectos o duplicados y mantener la integridad de la información financiera de la propiedad.	Alta
Balances (Épica TFM-30)	TFM-31	Recogida y filtrado de datos para realizar balances	**Como** usuario, **quiero** poder seleccionar una de mis propiedades y un rango de fechas, **para** poder hacer cálculos con las operaciones filtradas	Media
	TFM-32	Generación de gráficos	**Como** usuario, **quiero** visualizar en gráficos las finanzas de mis propiedades, **para** estudiar su rentabilidad	Media
Gestión de documentos (Épica	TFM-34	Subida de documentos	**Como** usuario, **quiero** subir ficheros a la aplicación, **para** tener recogidos los distintos documentos en un solo lugar	Media

TFM-33)	TFM-35	Listar, ver y descargar documentos	**Como** usuario, **quiero** listar mis documentos subidos y poder verlos y descargarlos, **para** tener la información del documento a disposición	Media
	TFM-36	Eliminar documentos	**Como** usuario, **quiero** poder eliminar documentos subidos, **para** que se deje de estar entre los documentos subidos a la aplicación	Media
Ayuda en la declaración de la renta (Épica TFM-37)	TFM-38	Recogida y filtrado de datos para realizar los cálculos de la renta	**Como** usuario, **quiero** poder seleccionar una de mis propiedades, año fiscal, y demás datos necesarios **para** poder hacer cálculos relacionados con la renta	Media
	TFM-39	Presentación de ayuda fiscal para realizar la declaración de la renta	**Como** usuario, **quiero** visualizar la ayuda fiscal **para** realizar la declaración de la renta de una forma más sencilla	Media

4. DESARROLLO DEL SISTEMA

4.1. *Sprint* 0

El *sprint* 0 es el único que consta de una semana de duración. El objetivo es la puesta en marcha el proyecto, abordando todas aquellas historias técnicas necesarias para montar el ecosistema *software*.

4.1.1. *Sprint backlog*

A continuación, se muestra todas las historias técnicas incluidas en el *sprint* 0, de acuerdo con su priorización en la planificación del *sprint.*

TFM-3 Configuración del proyecto Front-End					
Descripción	**Como** desarrollador **Quiero** disponer del proyecto de Angular creado, configurado y ligado a un repositorio de código de GitHub **Para** comenzar con el desarrollo Front-End				
Prioridad	Muy alta	**Puntos de historia**	3	**Tiempo consumido**	4h
Criterios de aceptación	**Dado** un proyecto *software* **Cuando** se desea realizar la implementación **Entonces** se dispone de un proyecto Front-End con el *framework* de Angular creado y configurado				
Tareas					
Nombre de la tarea					**Tipo**
Crear proyecto básico de Angular					Front-End
Crear repositorio de GitHub y relacionarlo con Jira					DevOps
Configurar estructura inicial proyecto de Angular					Front-End

TFM-5 Añadir flujo de integración continua para Front-End					
Descripción	**Como** desarrollador **Quiero** añadir integración continua con GitHub Actions al proyecto de Angular **Para** automatizar la construcción, pruebas y análisis de código con SonarCloud				
Prioridad	Muy alta	**Puntos de historia**	2	**Tiempo consumido**	2h
Criterios de aceptación	**Dado** un proyecto Front-End con el *framework* Angular y con su repositorio de código en GitHub **Cuando** se realiza un *push* sobre la rama develop **Entonces** se lanza un flujo de integración continua que instala el proyecto, lo construye, y se realiza el análisis estático de código con SonarCloud				

Tareas	
Nombre de la tarea	**Tipo**
Configuración GitHub Actions y SonarCloud	DevOps
Creación del *workflow* para la integración continua	DevOps

TFM-6 Configuración del proyecto Back-End					
Descripción	**Como** desarrollador **Quiero** disponer del proyecto de Node.js creado, configurado y ligado a un repositorio de código de GitHub **Para** comenzar con el desarrollo Back-End				
Prioridad	Muy alta	**Puntos de historia**	3	**Tiempo consumido**	4h
Criterios de aceptación	**Dado** un proyecto *software* **Cuando** se desea realizar la implementación **Entonces** se dispone de un proyecto Front-End con el *framework* de Angular creado y configurado				

Tareas	
Nombre de la tarea	**Tipo**
Crear proyecto básico de Node.js	Back-End
Crear repositorio de GitHub y relacionarlo con Jira	DevOps
Configurar estructura inicial proyecto de Node.js con Express	Back-End

TFM-7 Añadir flujo de integración continua para Back-End					
Descripción	**Como** desarrollador **Quiero** añadir integración continua con GitHub Actions al proyecto de Node.js **Para** automatizar la construcción, pruebas y análisis de código con SonarCloud				
Prioridad	Muy alta	**Puntos de historia**	2	**Tiempo consumido**	1h
Criterios de aceptación	**Dado** un proyecto Back-End en Node.js con el *framework* Angular y con su repositorio de código en GitHub **Cuando** se realiza un *push* sobre la rama develop **Entonces** se lanza un flujo de integración continua que instala el proyecto, lo construye, pasa los test unitarios y de integración, y se realiza el análisis estático de código con SonarCloud				

Tareas	
Nombre de la tarea	**Tipo**
Creación del *workflow* para la integración continua	DevOps

TFM-8 Configuración de la base de datos					
Descripción	**Como** desarrollador **Quiero** configurar una base de datos MySQL y conectarla con el proyecto de Node.js **Para** persistir datos del proyecto				
Prioridad	Muy alta	**Puntos de historia**	3	**Tiempo consumido**	5h
Criterios de aceptación	**Dado** un proyecto Back-End en Node.js con el *framework* Express **Cuando** se desean persistir datos mediante el ORM Sequelize **Entonces** existe una base de datos MySQL creada y conectada con el Back-End				

Tareas	
Nombre de la tarea	**Tipo**
Instalación local MySQL y HeidiSQL	Despliegue
Configuración ORM Sequelize en Back-End	BBDD
Conexión con BBDD y persistencia de datos	BBDD

TFM-9 Configuración del entorno de pruebas					
Descripción	**Como** desarrollador **Quiero** configurar el entorno de pruebas unitarias y de integración con Jest, SuperTest **Para** verificar el correcto funcionamiento del código				
Prioridad	Muy alta	**Puntos de historia**	5	**Tiempo consumido**	4h
Criterios de aceptación	**Dado** un proyecto Back-End en Node.js con el *framework* Angular **Cuando** se desea probar una funcionalidad de forma unitaria **Entonces** el entorno de pruebas con el *framework* Jest está configurado				
	Dado un proyecto Back-End en Node.js con el *framework* Angular **Cuando** se desea realizar un test de integración **Entonces** el entorno de pruebas con el *framework* Jest y la biblioteca SuperTest está configurado				
	Dado un proyecto Back-End en Node.js con el *framework* Angular **Cuando** se dispara el flujo de integración continua **Entonces** se lanzan los test unitarios y de integración con el flujo de GitHub Actions				

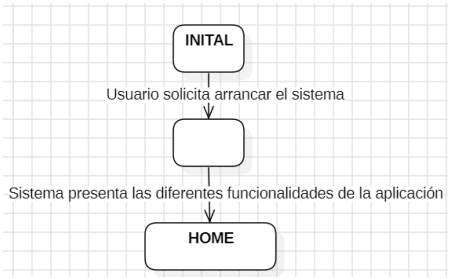
Tareas	
Nombre de la tarea	**Tipo**
Instalación de Jest y SuperTest	Despliegue
Creación de test unitarios "*mockeando*"	Pruebas
Creación de test de integración	Pruebas
Configurar el flujo de integración continua del Back-End para incluir las pruebas.	Pruebas/DevOps

4.1.2. Diseño de la arquitectura

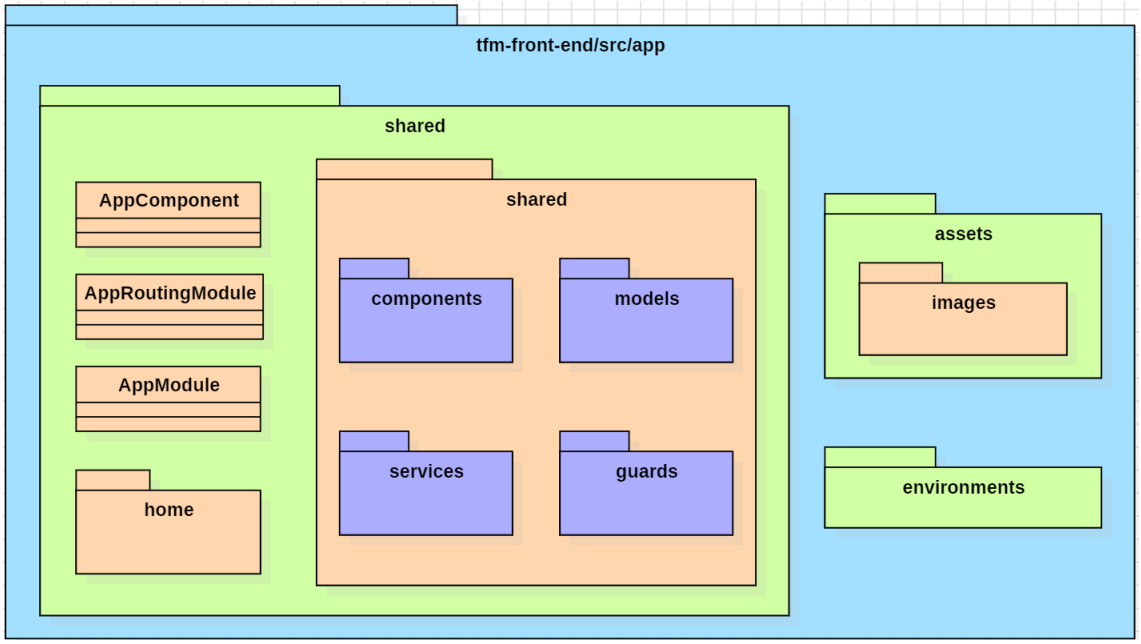
En este apartado se detalla el diseño y la arquitectura de la aplicación, tanto de la parte frontal, como la trasera de la aplicación.

4.1.2.1. Diagramas de casos de uso

Ver opciones/cabecera (ShowOptions):



4.1.2.2.a. Diagrama de paquetes (Front)



En el proyecto del Front-End, dentro de la carpeta **src** donde se presenta el diagrama, se encuentra la carpeta principal **app**, contenedora del corazón de la aplicación; la carpeta **assets**, que contendrá los recursos estáticos de la aplicación; y la carpeta **environments**, donde guardar archivos de configuración específicos del entorno.

Dentro de la carpeta **app** están los 3 archivos principales de la aplicación:

AppComponent: Componente principal de la aplicación, donde se define la estructura básica de la aplicación y se incluyen otros componentes.

AppModule: Este es el módulo principal de la aplicación. Los módulos en Angular son una forma de organizar características relacionadas en bloques cohesivos.

AppRoutingModule: Este módulo contiene la configuración de enrutamiento de la aplicación. Define cómo la aplicación responde a diferentes URL y qué componentes se deben renderizar en respuesta.

Al mismo nivel que estos ficheros está la carpeta **home**, que simplemente es un componente específico para la página de inicio de la aplicación. De la misma forma se irán añadiendo componentes principales de la aplicación (login, register…)

También a este nivel se tiene la carpeta **shared**, que albergará los elementos que se utilizan en varias partes de la aplicación:

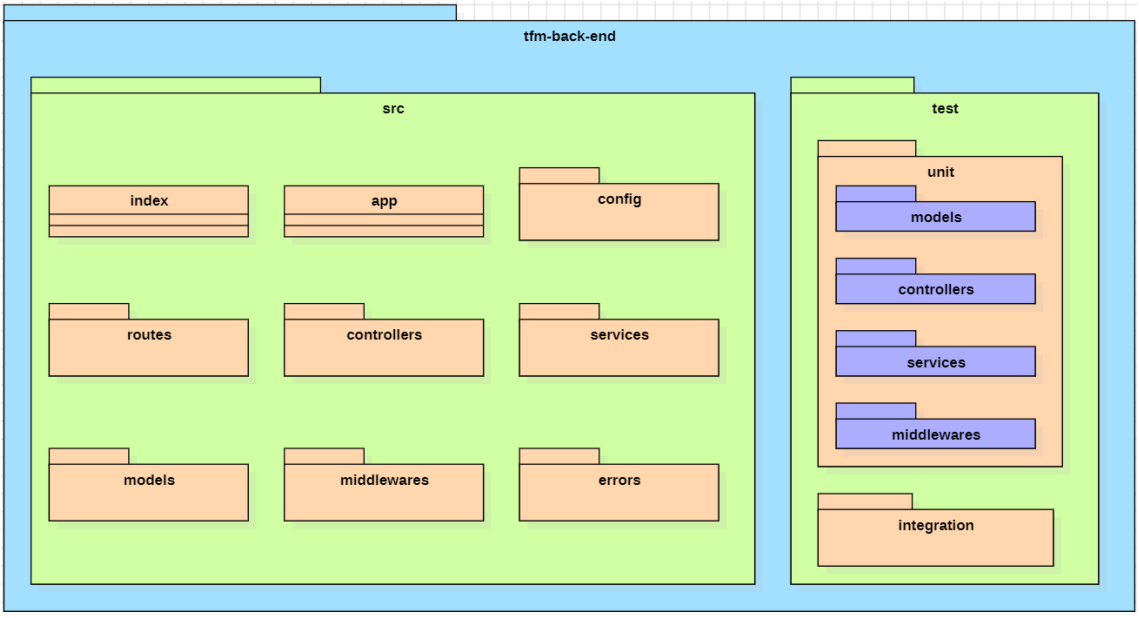
components: Esta subcarpeta tendrá componentes de interfaz de usuario reutilizables que se pueden utilizar en varias partes de la aplicación, como podrá ser un dialogo de confirmación o rechazo de una acción.

models: Esta subcarpeta almacenará los modelos de datos que definen la forma y estructura de los datos utilizados en la aplicación.

services: Donde se guardarán los servicios que manejan la lógica empresarial. Los servicios pueden ser inyectados en componentes y otros servicios, lo que permite compartir código y datos.

guards: Donde se tendrán los guardias que se utilizan para proteger rutas y gestionar permisos.

4.1.2.2.b. Diagrama de paquetes (Back)



El proyecto del Back-End se compone de dos carpetas principales: **src** donde se aloja el código fuente, y **test** donde está el código de pruebas.

En **src** se encuentran los dos archivos principales: **app** donde se configura la aplicación de Express e **index** para arrancar la aplicación. La separación de estos archivos es en favor de la modularidad, ya que por ejemplo desde los test de integración, se necesitará

levantar una aplicación de Express pero con otra configuración distinta específica para pruebas.

Dentro de src también se tienen las siguientes carpetas:

Primero, **config** para almacenar las configuraciones, como la configuración de la conexión con la base de datos.

La siguiente carpeta **routes**, donde se almacenas los archivos que definen cómo la aplicación responde a las diferentes solicitudes HTTP en varias rutas.

Posteriormente, la carpeta **controllers** albergará todos los controladores necesarios para el proyecto. Estos métodos de controlador obtienen la solicitud de las rutas y las convierten en respuestas HTTP con el uso de cualquier *middleware* según sea necesario.

Por tanto, la carpeta **middlewares** contendrá aquellos *middlewares* necesarios para la aplicación, por ejemplo, para la autenticación, el manejo de errores, etc.

Después, en la carpeta **models** tendrá los modelos de datos necesarios para la aplicación. En el caso de este proyecto al utilizar el ORM Sequelize, cada modelo representa una tabla en la base de datos y define los campos y métodos para interactuar con esa tabla. Sequelize se encargará de traducir estas interacciones a consultas SQL que se ejecutan en la base de datos.

A continuación, la carpeta **services** albergará toda la lógica de negocio. Utilizando los servicios para mantener el código de los controladores limpio y modular, ya que los controladores pueden llamar a los servicios cuando necesitan realizar una funcionalidad específica. Esto permite que la lógica de negocio se mantenga separada de la lógica de manejo de solicitudes HTTP, lo que facilita la prueba y el mantenimiento del código.

Por último, en la carpeta **errors** se almacenará archivos que definen clases de error personalizadas para manejar errores específicos de la aplicación de una manera más estructurada y coherente.

En la parte de pruebas, carpeta **test**, por un lado, está la carpeta **unit** para almacenar los test unitarios de las capas **models**, **controllers**, **services** y **middlewares**. Y, por otro lado, la carpeta **integration** donde se situarán los test que prueban la integración entre las distintas capas.

4.1.2.3. Modelo de datos

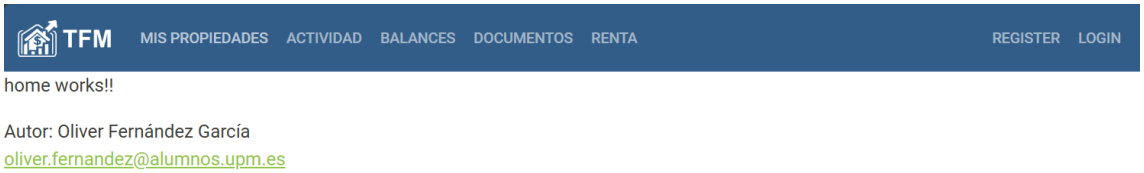
En este *sprint* 0, como el objetivo es montar el ecosistema del proyecto, se incluye el modelo de User simplificado, es decir, con menos atributos y funciones de lo que tendrá el modelo de User final, el cual se presentará posteriormente. Así, se consigue centralizar los esfuerzos en montar la arquitectura, interconectar las distintas capas, conectar con la base de datos, y configurar el entorno de pruebas con un modelo más sencillo.

User (Simplified)
+user_id: UUID (PK) +username: String (NOT NULL, UNIQUE) +password: String (NOT NULL) +email: String (NOT NULL)

Además, a los atributos definidos, Sequelize ORM añade createdAt y updatedAt, con la fecha de registro y de actualización, respectivamente. Y también distintas funciones de instancia (save, update…), de modelo (create, findAll, findByApk…), asociaciones, validaciones, etc. Como esto sucederá con todos los modelos, a partir de ahora no se volverá a especificar en este documento.

4.1.3. Interfaz

En este *sprint* como parte de la estructura, se añade a la interfaz de usuario la cabecera con las distintas opciones, la página principal de la aplicación aún por desarrollar, y el pie.



4.1.4. *Endpoints*

De nuevo, como el objetivo del *sprint* es abordar las historias técnicas y dejar todo funcionando, los *endpoints* creados (para listar usuarios y crearlos) se especificarán más adelante.

4.1.5. Pruebas unitarias y de integración

Por el momento se configura el entorno de pruebas, creando la estructura para los test unitarios y de integración. Se incorporan las bibliotecas para pruebas y se crean test para los *endpoints* creados que servirán como ejemplo para los siguientes *sprints*.

4.1.6. *Burn-down chart*

A continuación, se muestra el gráfico de pendiente (*burn-down chart*) para visualizar y analizar el progreso de los puntos de historia del *sprint* en relación con el tiempo.

De manera muy visual puede observarse como a mitad del *sprint* se produce una pausa en la entrega de tareas, esto se debe a varios factores:

Problema en la cuenta de GitHub para conectar con aplicaciones de terceros, que será comentado en la revisión del *sprint*.

Las dos últimas tareas coinciden con la configuración de la base de datos y la configuración del entorno de pruebas, que son las que más tiempo han consumido del *sprint*.

El fin de semana se dispone de más tiempo para dedicar al proyecto.

En conclusión, por el momento en este *sprint* 0 no preocupa una pendiente tan inclinada a final del *sprint*. Por un lado, ha existido un problema ajeno al desarrollo y por el otro lado, se espera que en los próximos *sprints* también haya más entrega los fines de semana por mayor disponibilidad de tiempo, afectando así la forma del gráfico.

4.1.7. *Sprint review*

En la revisión se realiza una evaluación del *sprint*, es decir, se realiza una verificación para asegurar que todas las historias de usuario, que se establecieron al inicio del *sprint*, han sido completadas y recibido la aprobación del cliente.

Es esencial tener en cuenta que las historias técnicas desarrolladas en este *sprint* no son sometidas a revisión por parte del cliente. No obstante, invertir tiempo en estas tareas técnicas preliminares acelerará las entregas del producto en el futuro.

Las tablas que siguen a continuación proporcionan un desglose detallado de las historias de usuario que se planificaron para este *sprint*.

Historia	Criterios de aceptación	Entregada/ Aceptada
TFM-3 Configuración del proyecto Front-End	**Como** desarrollador **Quiero** disponer del proyecto de Angular creado, configurado y ligado a un repositorio de código de GitHub **Para** comenzar con el desarrollo Front-End	Sí / Sí
TFM-5 Añadir flujo de integración continua para proyecto Front-End	**Dado** un proyecto Front-End con el *framework* Angular y con su repositorio de código en GitHub **Cuando** se realiza un *push* sobre la rama develop **Entonces** se lanza un flujo de integración continua que instala el proyecto, lo construye, y se realiza el análisis estático de código con SonarCloud	Sí / Sí
TFM-6 Configuración del proyecto Back-End	**Dado** un proyecto *software* **Cuando** se desea realizar la implementación **Entonces** se dispone de un proyecto Front-End con el *framework* de Angular creado y configurado	Sí / Sí
TFM-7 Añadir flujo de integración continua para proyecto Back-End	**Dado** un proyecto *software* **Cuando** se desea realizar la implementación **Entonces** se dispone de un proyecto Front-End con el *framework* de Angular creado y configurado	Sí / No
TFM-8 Configuración de la base de datos	**Dado** un proyecto Back-End en Node.js con el *framework* Express **Cuando** se desean persistir datos mediante el ORM Sequelize **Entonces** existe una base de datos MySQL creada y conectada con el Back-End	Sí / Sí
TFM-9 Configuración del entorno de pruebas	**Dado** un proyecto Back-End en Node.js con el *framework* Angular **Cuando** se desea probar una funcionalidad de forma unitaria **Entonces** el entorno de pruebas con el *framework* Jest está configurado	Sí / Sí
	Dado un proyecto Back-End en Node.js con el *framework* Angular **Cuando** se desea realizar un test de integración **Entonces** el entorno de pruebas con el *framework* Jest y la biblioteca SuperTest está configurado	
	Dado un proyecto Back-End en Node.js con el framework Angular **Cuando** se dispara el flujo de integración	

	continua **Entonces** se lanzan los test unitarios y de integración con el flujo de GitHub Actions	

Más allá de las complicaciones al arrancar un proyecto de *software* por compatibilidad de versiones, dificultades en instalaciones o errores que se pueden considerar habituales y que han sido solventados sobre la marcha, la principal barrera durante este *sprint* ha sido un problema ajeno al proyecto debido a una de las herramientas de terceros utilizada, es decir, un problema sobre el que no se puede influir (véase sección 2.7. Riesgos e incertidumbres).

Una vez creados los repositorios en GitHub (para el Front-End y el Back-End), ligados con Jira Software y funcionando la trazabilidad. También con la integración continua del Front-End funcionando, lo que quiere decir que la historia "TFM-5 Añadir flujo de integración continua para proyecto Front-End" estaba entregada. Todo eso conlleva la configuración de GitHub Actions y también de SonarCloud, pero aparece el error de que la cuenta de GitHub utilizada se marca con el *flag* de *spam* lo que impide autorizarse con aplicaciones de terceros:

"You are marked as spam, and therefore cannot authorize a third party application"

A efectos prácticos, esto significa que deja de funcionar la integración continua con GitHub Actions, los *workflows* se encolan, pero jamás arrancan. No se puede acceder al SonarCloud ya configurado. La conexión con Jira también deja de funcionar.

Esto ha retrasado mucho la entrega de las historias de este *sprint* en busca de una solución, que no ha sido encontrada. Como reportan otros usuarios con el mismo problema, no hay una solución aparente, más allá de contactar con el equipo de GitHub y que ellos lo solucionen, algo que puede demorarse durante semanas. Por tanto, este problema se espera que no solo afecte a este primer *sprint* y siga consumiendo tiempo de los venideros.

4.1.8. Retrospectiva

En esta ceremonia final se reflexiona sobre el *sprint* 0 que acaba de concluir con el objetivo de mejorar continuamente.

Se discute lo que funcionó bien y lo que podría mejorarse. Se identifican las áreas de éxito para continuar con ellas y se señalan los desafíos para abordarlos en el próximo *sprint*. Esta es una oportunidad para que el equipo aprenda de sus experiencias y haga ajustes en su enfoque para mejorar la eficiencia y la efectividad.

	¿Qué ha pasado?	**Acciones**
¿Qué fue bien?	Estimación acertada	-
	Superación de dificultades técnicas con gran soltura y eficacia	-

¿Qué se puede mejorar?	Durante el desarrollo se tiene muchas historias en progreso sin llegar a terminar ninguna	A partir de ahora tratar de cerrar historias y no acumular tantas en progreso
¿Qué fue mal?	Problema con la cuenta de GitHub	Contacto con el equipo de soporte de GiHub
		Si no se soluciona, habrá que crear una nueva cuenta y reconfigurar todo

4.2. *Sprint* 1

El *sprint* 1 ya cuenta con una duración de dos semanas, siendo esta la duración establecida desde un comienzo ya que el *sprint* 0 se trataba de una excepción. Como objetivo se tiene la gestión y autenticación de usuarios para acceder a las futuras funcionalidades de la aplicación.

4.2.1. *Sprint backlog*

En las siguientes tablas, se detallas todas las historias de usuario incluidas en el *sprint* 1 de acuerdo con su priorización en la planificación del *sprint.* La última (TFM-17), se trata de una historia técnica para acometer el error que estaba imposibilitando el uso de Jira, GitHub Actions, SonarCloud, etc. Pese a categorizarse con una prioridad muy alta, no se tiene en cuenta en la planificación ya que esta historia entra en mitad del *sprint* al no tener una solución del supuesto problema por parte del soporte de GitHub.

TFM-11 Registro de usuario					
Descripción	**Como** cliente a utilizar la aplicación **Quiero** dar de alta un nuevo usuario **Para** poder acceder a las funcionalidades de la aplicación.				
Prioridad	Muy alta	**Puntos de historia**	8	**Tiempo consumido**	8h
Criterios de aceptación	**Dado** un cliente de la aplicación **Cuando** quiere darse de alta como usuario **Entonces** dispone de un formulario de registro				
	Dado un cliente de la aplicación **Cuando** completa el formulario de registro con su username, email, contraseña dos veces OK **Entonces** se da de alta su usuario y se le redirige al login para que se autentique				
	Dado un cliente de la aplicación **Cuando** completa alguno de los campos del formulario de registro incorrectamente **Entonces** se muestra el error correspondiente y se permite modificar los datos del formulario para corregirlos y completar el registro				

Tareas	
Nombre de la tarea	**Tipo**
Crear modelo de usuario y controlador (servicios/rutas…) para leer y crear un usuario	Back-End
Formulario de registro y conexión con Back-End	Front-End

TFM-12 Autenticación de usuarios	
Descripción	**Como** usuario dado de alta **Quiero** iniciar y cerrar sesión en la aplicación **Para** acceder a mis datos personales, interactuar con las funcionalidades y garantizar la privacidad y seguridad de mi cuenta

Prioridad	Muy alta	**Puntos de historia**	13	**Tiempo consumido**	11h
Criterios de aceptación	**Dado** un usuario **Cuando** intenta acceder a una funcionalidad de la aplicación a excepción de visualizar el home **Entonces** no tiene permisos y se le redirige al formulario login para que se autentique				
	Dado un cliente de la aplicación **Cuando** completa el formulario de login con su username y contraseña OK **Entonces** está autenticado en la aplicación y se le redirige al inicio				
	Dado un cliente de la aplicación **Cuando** completa alguno de los campos del formulario de login incorrectamente **Entonces** se muestra error en el inicio de sesión y se le permite seguir intentándolo				
	Dado un usuario logeado **Cuando** pulsa en el botón para cerrar sesión en la cabecera **Entonces** cierra sesión de su cuenta y se le redirige al login				

Tareas	
Nombre de la tarea	**Tipo**
Endpoint para el login de usuarios	Back-End
Middleware para comprobar la autenticación del usuario	Back-End
Formulario de login y conexión con Back-End	Front-End
Funcionalidad de cierre de sesión y establecer guardias, interceptores, etc.	Front-End

TFM-13 Consultar "Mi perfil"					
Descripción	**Como** usuario autenticado **Quiero** tener una funcionalidad de "mi perfil" **Para** consultar los datos de mi usuario y tener acceso a funcionalidades de edición o borrado de mi usuario				
Prioridad	Alta	**Puntos de historia**	3	**Tiempo consumido**	5h
Criterios de aceptación	**Dado** un usuario logeado **Cuando** pulsa en "Mi perfil" situado en la cabecera **Entonces** se muestra los datos de su perfil teniendo opciones para editarlos y para borrar su usuario				

Tareas	
Nombre de la tarea	**Tipo**
Middleware para realizar la validación de usuarios en los *endpoints* (en este caso leer usuario)	Back-End
Diseño de la página "mi perfil" adaptado a incorporar TFM-14 y TFM-15 y conexión con Back-End para leer el perfil	Front-End

TFM-14 Editar usuario					
Descripción	**Como** usuario en "mi Perfil" **Quiero** poder editar mi usuario **Para** actualizar o corregir mis datos.				
Prioridad	Alta	**Puntos de historia**	5	**Tiempo consumido**	4h
Criterios de aceptación	**Dado** un usuario en "mi perfil" **Cuando** pulsa el botón de editar **Entonces** puede editar todos los campos menos el nombre de usuario (y contraseña), y guardar o cancelar los cambios realizados.				

Tareas	
Nombre de la tarea	**Tipo**
Endpoint para editar un usuario	Back-End
Formulario de edición de usuario con validaciones y conexión con Back-End	Front-End

TFM-15 Eliminar usuario					
Descripción	**Como** usuario en "mi Perfil" **Quiero** poder eliminar mi usuario **Para** dejar de ser usuario de la aplicación.				
Prioridad	Alta	**Puntos de historia**	3	**Tiempo consumido**	4h
Criterios de aceptación	**Dado** un usuario en "Mi perfil" **Cuando** pulsa en eliminar y confirma que desea eliminar su cuenta **Entonces** su usuario se elimina				

Tareas	
Nombre de la tarea	**Tipo**
Endpoint para eliminar un usuario	Back-End
Componente de confirmación de acción reutilizable	Front-End
Funcionalidad de borrado de usuario y conexión con Back-End	Front-End

TFM-16 Diseño de la página de inicio					
Descripción	**Como** desarrollador **Quiero** que la aplicación tenga un "*home*" accesible con o sin autenticación **Para** que mejorar la interfaz y que exista una buena experiencia de usuario				
Prioridad	Media	**Puntos de historia**	3	**Tiempo consumido**	4h
Criterios de aceptación	N/A				

Tareas	
Nombre de la tarea	**Tipo**
Realizar la página de inicio de la aplicación	Front-End

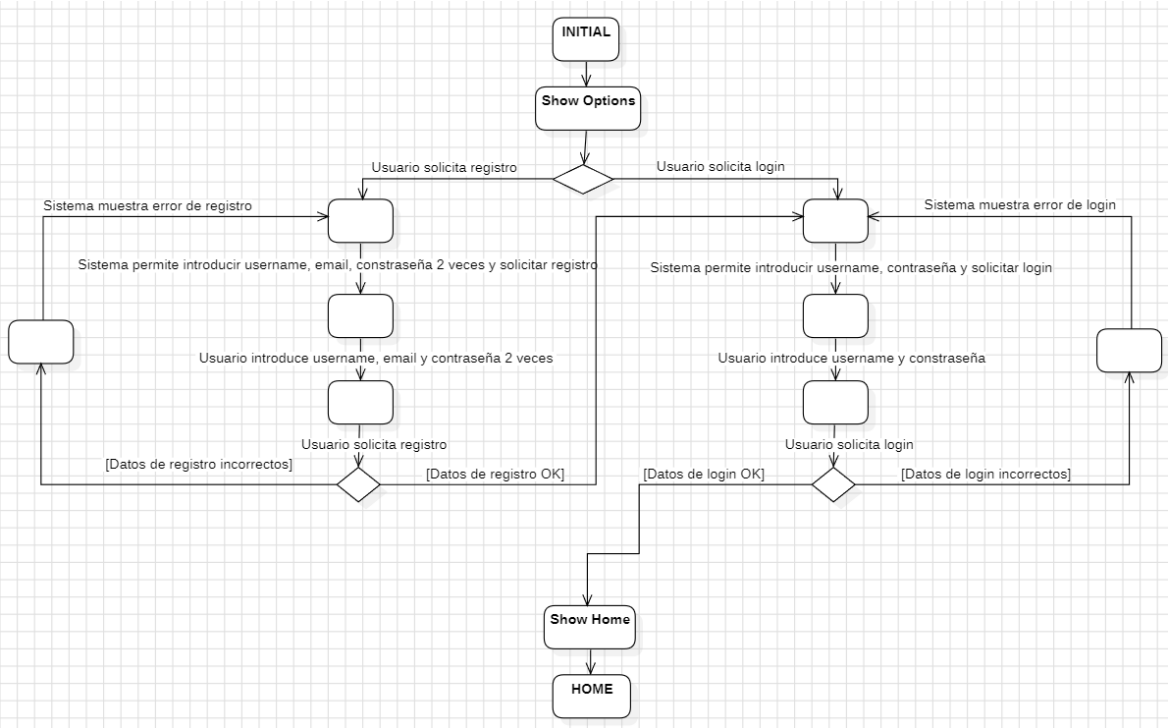
TFM-17 (Error) Crear nueva cuenta de GitHub y enlazar todas las herramientas del ecosistema					
Descripción	**Como** desarrollador **Quiero** tener un ecosistema *software* con todas las herramientas enlazadas funcionando **Para** un desarrollo de calidad apropiado				
Prioridad	Muy Alta	**Puntos de historia**	3	**Tiempo consumido**	5h
Criterios de aceptación	N/A				

Tareas	
Nombre de la tarea	**Tipo**
Creación de nueva cuenta de GitHub y de los repositorios de código	DevOps
Arreglar integración continua con GitHub Actions de Front-End y Back-End	DevOps
Arreglar conexión con SonarCloud	DevOps
Arreglar conexión y trazabilidad con Jira Software	DevOps

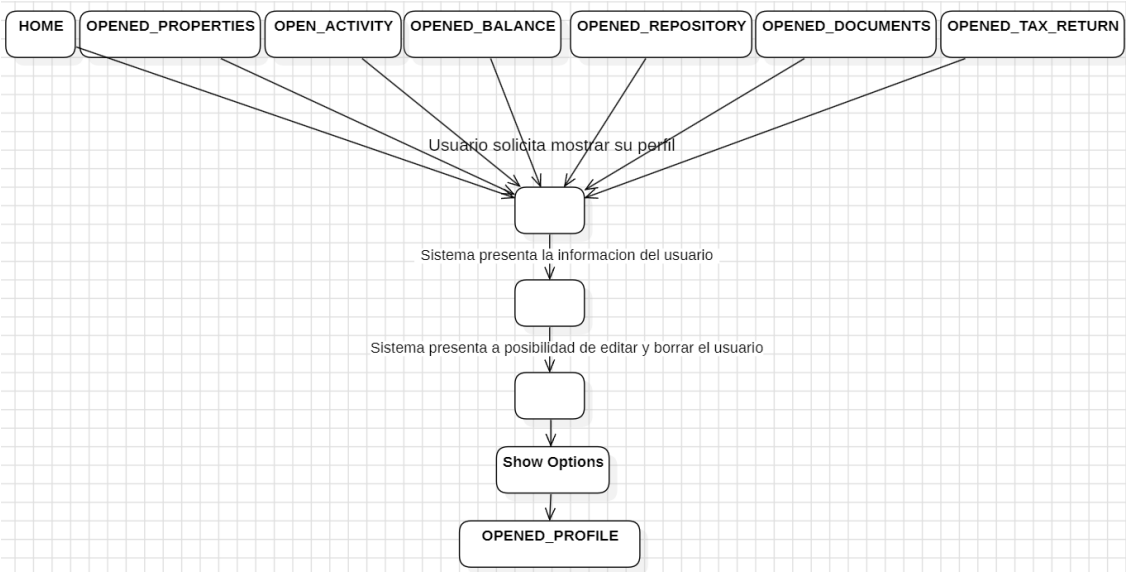
4.2.2. Diseño de la arquitectura

4.2.2.1. Diagramas de casos de uso

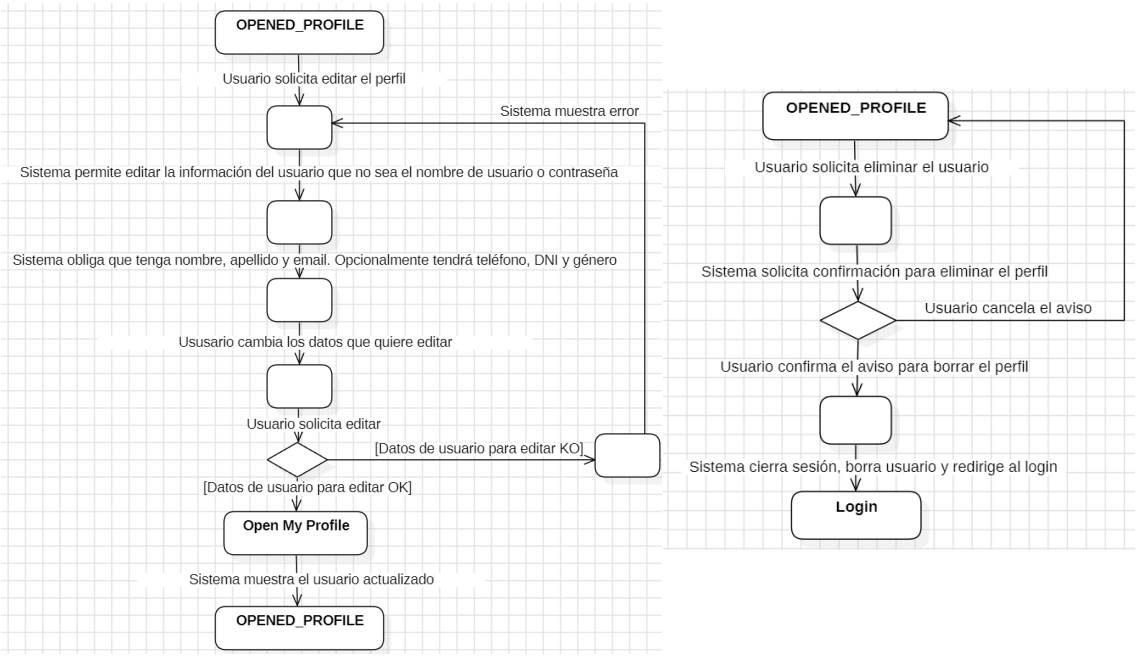
Login:



Abrir mi perfil (OpenMyProfile):

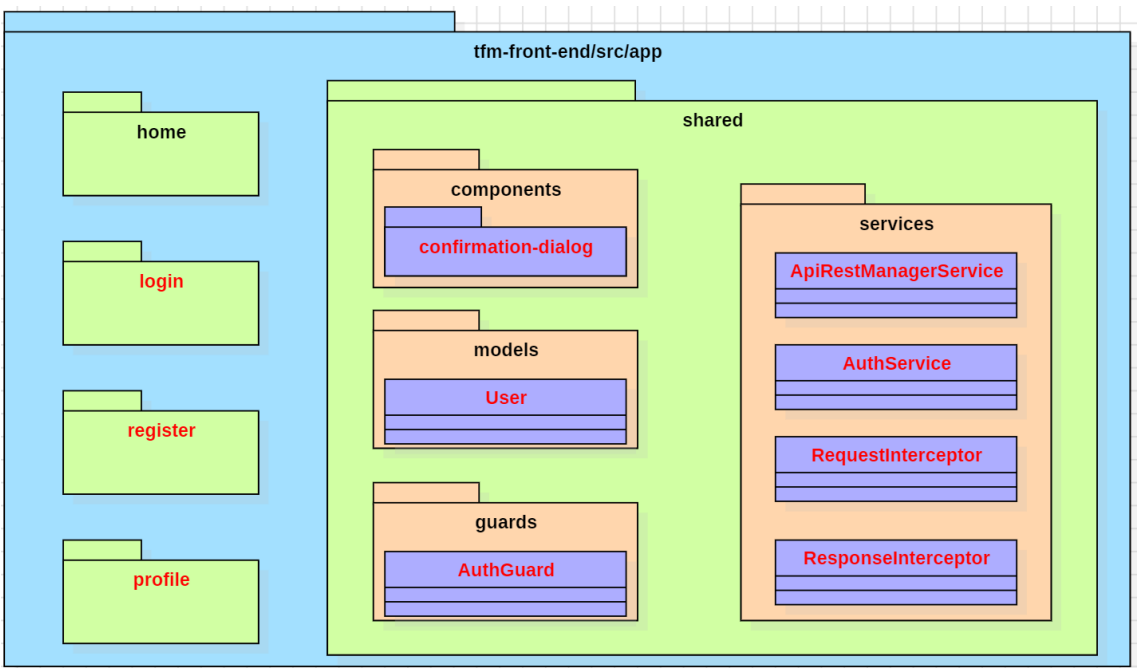


Editar perfil (UpdateProfile) y borrar perfil (DeleteProfile):



4.2.2.2.a. Diagrama de paquetes (Front)

Para el Front-End se hace foco en la carpeta app, dejando fuera la carpeta assets ya descrita anteriormente (véase 4.1.2.2.a.), igualmente se omite AppComponent, AppModule y AppRoutingModule. Con texto rojo se indica aquellos elementos nuevos añadidos durante este *sprint*.



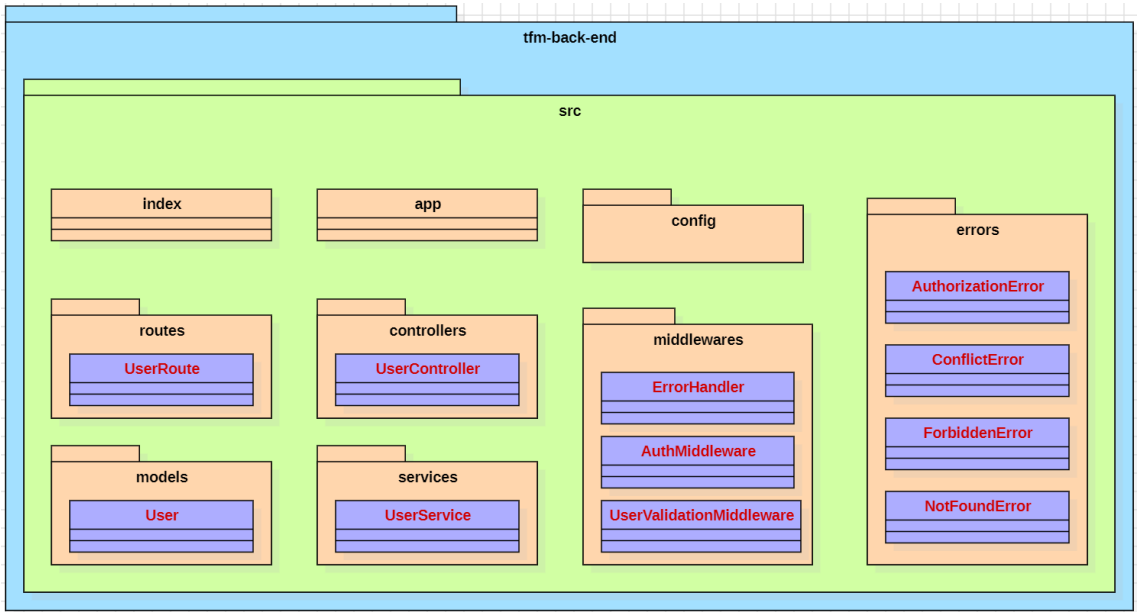
El componente **home** estaba creado y se termina de realizar, por su parte, se crea los componentes **login**, **register** y **profile** que contienen la funcionalidad que su nombre indica.

Por otro lado, en los elementos compartidos de la aplicación (**shared**), se crea el **modelo** de usuario (**user**). En **components** se tiene un diálogo (**ConfirmationDialogComponent**) para confirmar o cancelar una acción. Y en **guards**, un "guardián" para proteger las rutas de forma que si el usuario no está autenticado solo pueda acceder a las rutas de inicio de sesión y registro, y viceversa, cuando esté autenticado no pueda acceder a esas rutas y sí al resto. La página de inicio siempre es accesible.

Por último, en los **servicios** se tiene: el servicio **AuthService**, para todo el manejo de tokens y sesiones; los servicio **RequestInterceptor** y **ResponseInterceptor**, para capturar las peticiones salientes o entrantes y poner o extraer el token de las cabeceras; y el servicio **ApiRestManagerService**, para la comunicación con el Back-End.

4.2.2.2.b. Diagrama de paquetes (Back)

Para el Back-End se hace foco en la carpeta src, dejando fuera la carpeta test ya que al final replica la estructura del código fuente, para probar las distintas capas (test unitarios) y la integración de todas ellas (test de integración).



En las **rutas** se tiene **UserRoute** donde se define cómo la aplicación responde a las diferentes solicitudes HTTP en la ruta de usuario. Aunque no se especifica, en el archivo de **index** todas las rutas de la aplicación son importadas y unidas para permitir que todas las rutas estén disponibles en el momento que la aplicación importe este fichero.

Dentro de los **modelos** se tiene el **User** definido en la siguiente sección y que representa una un modelo de datos que representa una tabla en base de datos. Similar a lo que sucede en las rutas, en el archivo **index** se conecta con Sequelize ORM utilizando la configuración (carpeta **config**).

Por otra parte, existe tanto un **controlador** (**UserController**) y un **servicio** (**UserService**) para el usuario, el primero para procesar las solicitudes de las rutas en respuestas HTTP y el segundo para contener la lógica de negocio y mantener un controlador limpio y modular.

Los **middlewares** son utilizados para tareas como la autenticación (**AuthMiddleware**), validación de usuarios para control accesos prohibidos (**UserValidationMiddleware**) y manejo de errores (**ErrorHandler**) dentro de la aplicación.

Por último, en la carpera **errors** se incluyen varios tipos de errores para manejarlos de una forma estructurada, para errores de validación de datos se reutiliza el error de validación proporcionado por Sequelize.

4.2.2.3. Modelo de datos

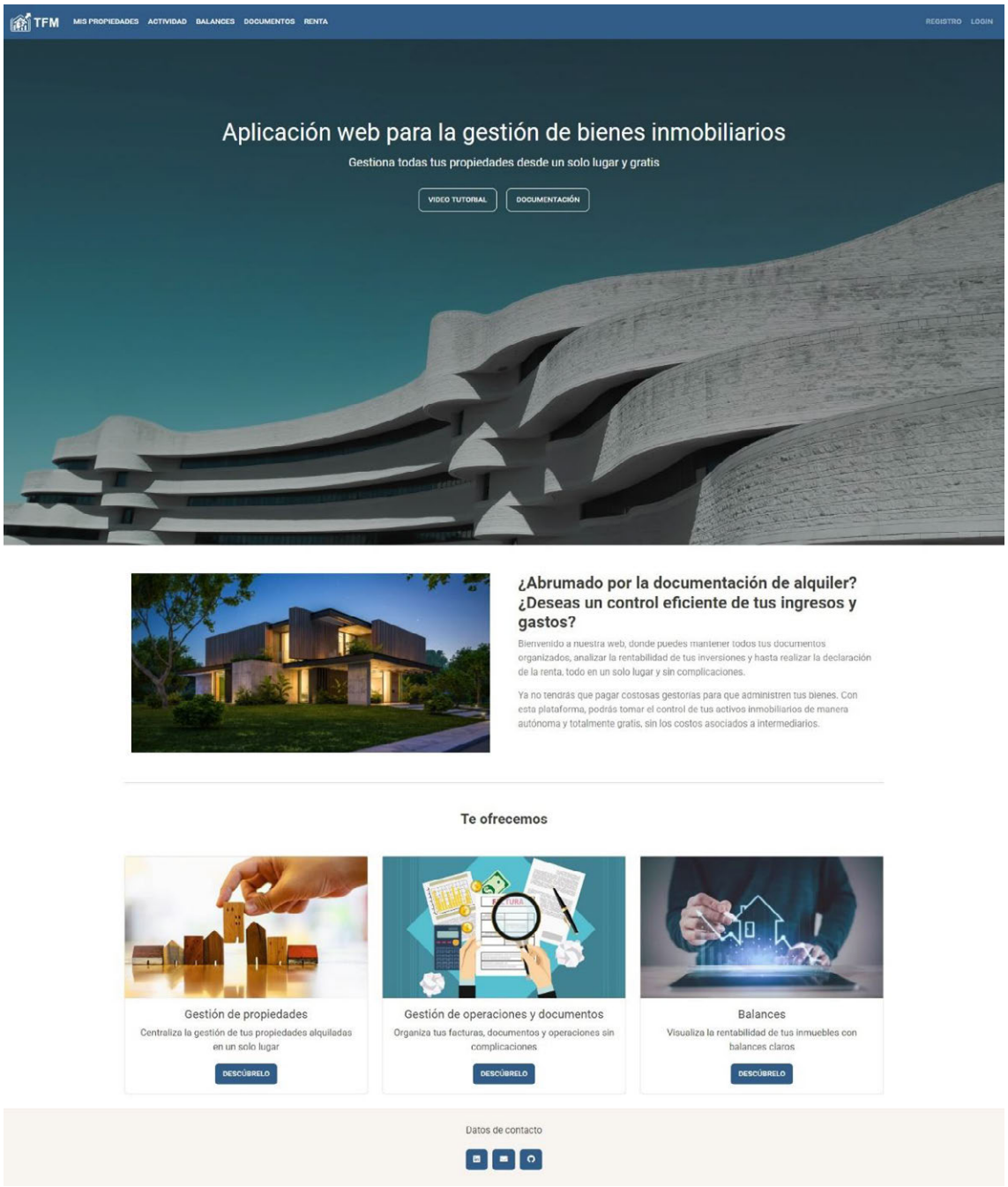
En base de datos se tiene el modelo User, con más atributos que en el *sprint* 0, siendo este el modelo con los atributos definitivos a falta de incluir las futuras asociaciones con otros modelos. En la imagen se muestran los diferentes atributos además de especificar si es primary key (PK), único (UNIQUE) o si es obligatorio (NOT NULL).

User
+user_id: UUID (PK) +name: String (NOT NULL) +lastName: String (NOT NULL) +username: String (NOT NULL, UNIQUE) +password: String (NOT NULL) +email: String (NOT NULL) +phoneNumber: String +dni: String (UNIQUE) +gender: 'male' | 'female'

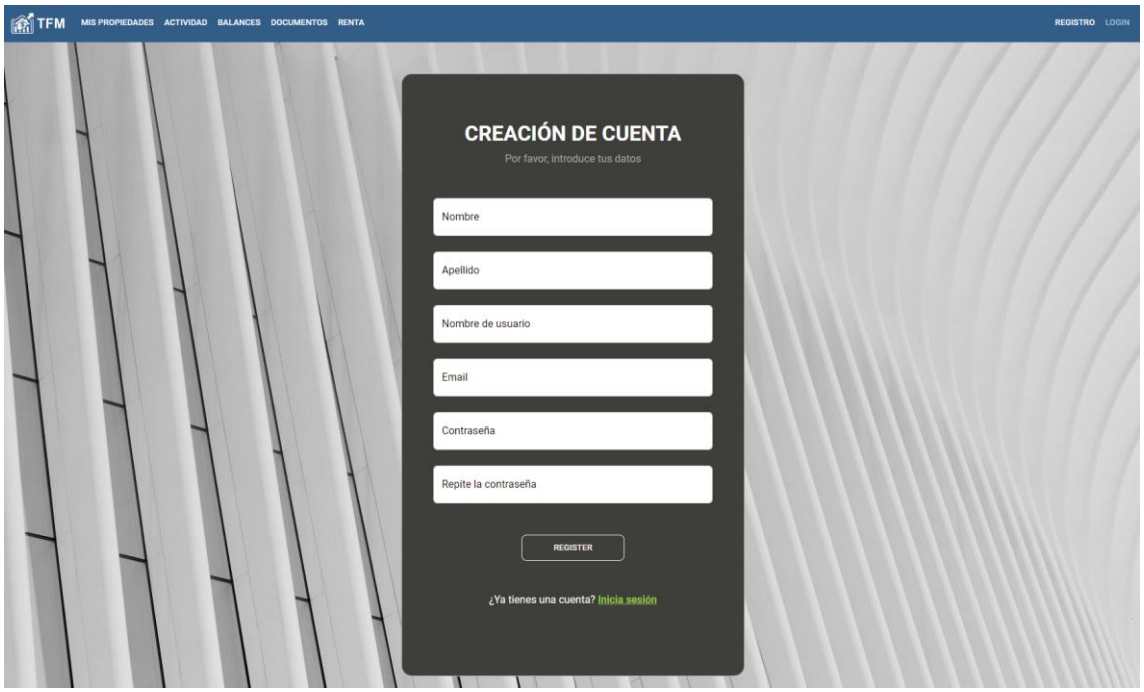
4.2.3. Interfaz

Interfaz de usuario a fecha de finalización del *sprint* 1.

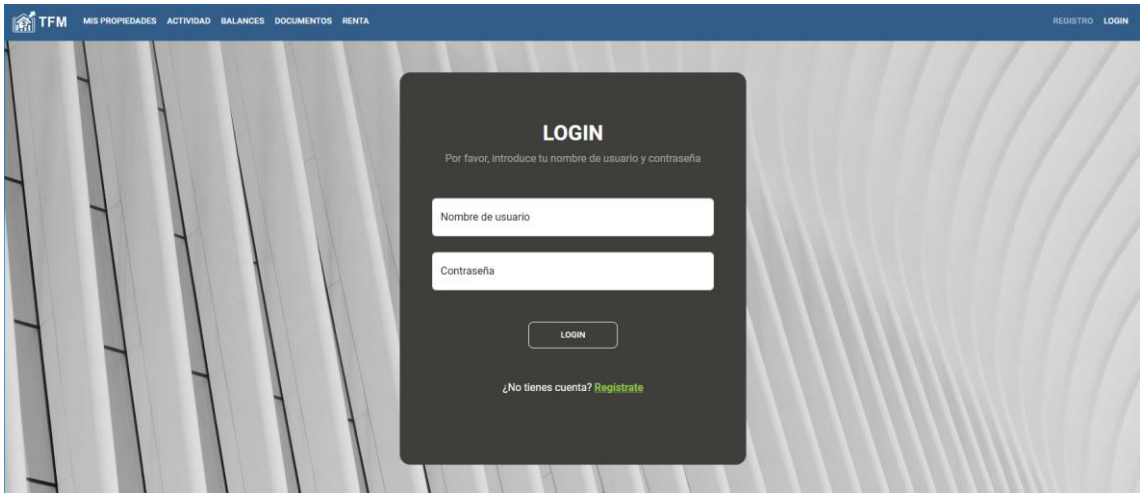
Página de inicio:



Registro:

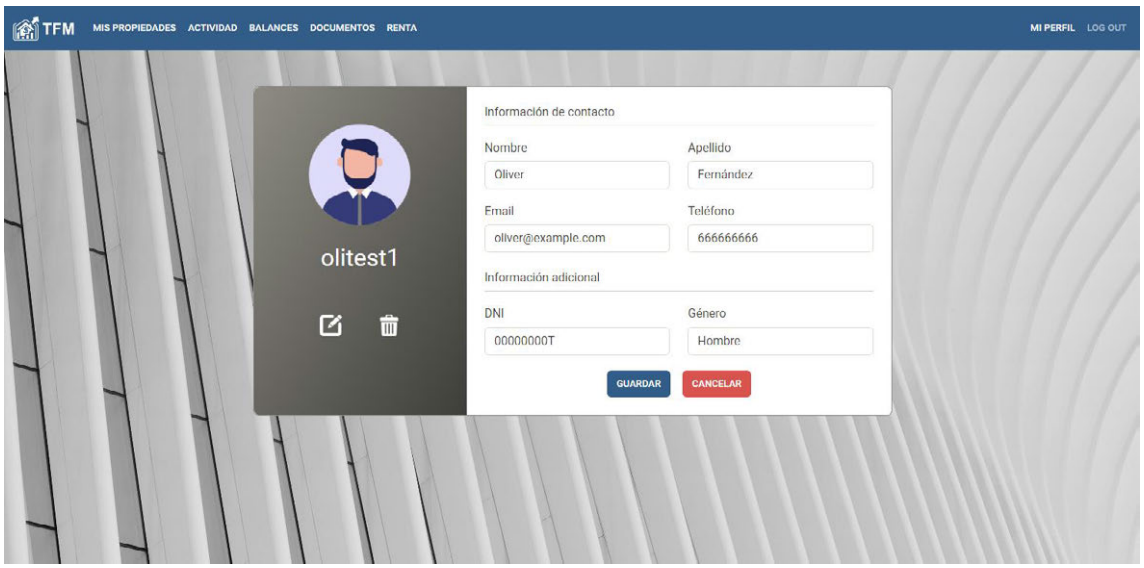


Login:



Mi perfil:

En formato edición, por lo que los distintos campos son editables y se visualiza los botones de “guardar” y “cancelar”. Sin este formato simplemente se visualizan los campos del perfil sin ser editables. Y la opción de borrar abre un dialogo de confirmación.



4.2.4. *Endpoints*

Durante este *sprint* los nuevos *endpoints* están enfocados en cubrir la autenticación y gestión de usuario coincidiendo con el objetivo del *sprint*. La autenticación se realiza mediante JSON Web Token (JWT), que por un lado en su carga útil (*payload*), contiene información del usuario, como el user_id o el nombre de usuario además de la fecha de creación y de expiración del token, y, por otro lado, la cabecera y la firma como manda el estándar RFC-7519.

Los *endpoints* que permiten la autenticación del usuario son los siguientes:

REGISTRO			
Endpoint			
Método	**Ruta**	**Descripción**	
POST	/users/register	Da de alta un nuevo usuario en el sistema	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
username*	String	Body	Nombre de usuario único
name*	String	Body	Nombre del usuario
lastName*	String	Body	Apellido del usuario
password*	String	Body	Contraseña del usuario
email*	String	Body	Dirección de correo electrónico del usuario
phoneNumber	String	Body	Número de teléfono del usuario
dni	String	Body	Documento de identidad del usuario
gender	‘male’ | ‘female’	Body	Género del usuario
Respuestas			
Código	**Mensaje**	**Descripción**	
200	OK	El usuario se ha creado correctamente	
400	BAD REQUEST	Algún parámetro del cuerpo es incorrecto	
409	CONFLICT	Ya existe un usuario con ese nombre de usuario	

LOGIN			
Endpoint			
Método	**Ruta**	**Descripción**	
GET	/users/login	Autentica al usuario y se genera un *token* asociado	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
username*	String	Query	Nombre de usuario para el login
password*	String	Query	Contraseña del usuario en texto claro
Respuestas			
Código	**Mensaje**	**Descripción**	
200	OK	Usuario autenticado, se recibe en la cabecera Authorization el token generado	
400	BAD REQUEST	Nombre de usuario o contraseña no especificado	
401	UNAUTHORIZED	Nombre de usuario o contraseña incorrecto	

En la gestión de usuarios no existe un leer todos los usuarios porque a priori no se necesita al no existir un rol de administrador, por tanto, se tiene:

LEER USUARIO			
Endpoint			
Método	**Ruta**	**Descripción**	
GET	/users/{username}	Se devuelve la información del usuario	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
username*	String	Path	Nombre de usuario para leer
Respuestas			
Código	**Mensaje**	**Descripción**	
200	OK	Se recibe la información del usuario	
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido	
403	FORBIDDEN	Acceso denegado	
404	NOT FOUND	No existe un usuario con ese nombre de usuario	

BORRAR USUARIO			
Endpoint			
Método	**Ruta**	**Descripción**	
DELETE	/users/{username}	Se borra el usuario en base de datos	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
username*	String	Path	Nombre de usuario para borrar
Respuestas			
Código	**Mensaje**	**Descripción**	
200	OK	El usuario se ha borrado satisfactoriamente	
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido	
403	FORBIDDEN	Acceso denegado	
404	NOT FOUND	No existe un usuario con ese nombre de usuario	

EDITAR USUARIO			
Endpoint			
Método	**Ruta**	**Descripción**	
PUT	/users/{username}	Actualizar los datos del usuario	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
username	String	Path	Nombre de usuario a editar
name*	String	Body	Nuevo nombre del usuario
lastName*	String	Body	Nuevo apellido del usuario
email*	String	Body	Nueva dirección de correo electrónico del usuario
phoneNumber	String	Body	Nuevo número de teléfono del usuario
dni	String	Body	Nuevo documento de identidad del usuario
gender	String	Body	Nuevo género del usuario, solo acepta "male" o "female"
Respuestas			
Código	**Mensaje**	**Descripción**	
200	OK	El usuario se ha actualizado correctamente	
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido	
403	FORBIDDEN	Acceso denegado	
404	NOT FOUND	No existe un usuario con ese nombre de usuario	

4.2.5. Pruebas unitarias y de integración

```
PASS test/integration/user.spec.js
  Users integration test
    √ Read user KO - User not found (54 ms)
    √ Create user OK (162 ms)
    √ Create user KO - Username already exists (16 ms)
    √ Read user OK (14 ms)
    √ Create user KO - Bad request (130 ms)
    √ Create user KO - Bad request no password (18 ms)
    √ Login user OK (129 ms)
    √ Login user KO - No password (15 ms)
    √ Login user KO - User not found (17 ms)
    √ Login user KO - Incorrect password (142 ms)
    √ Update user OK (32 ms)
    √ Update user KO - User not found (17 ms)
    √ Delete user OK (19 ms)
    √ Delete user KO - User not found (14 ms)

PASS test/unit/controllers/user.controller.spec.js
  User Controller
    √ Read user OK (23 ms)
    √ Read user KO - User not found (15 ms)
    √ Create user OK (21 ms)
    √ Create user KO - Username already exists (16 ms)
    √ Create user KO - No password (13 ms)
    √ Create user KO - Bad Request no password (16 ms)
    √ Create user KO - Bad Request wrong email format (12 ms)
    √ Login user OK (15 ms)
    √ Login user KO - No username (13 ms)
    √ Login user KO - User not found (12 ms)
    √ Login user KO - Incorrect password (14 ms)
    √ Update user OK (12 ms)
    √ Update user KO - User not found (11 ms)
    √ Delete user OK (12 ms)
    √ Delete user KO - User not found (11 ms)

PASS test/unit/services/user.service.spec.js
  User Service
    √ readUser OK (3 ms)
    √ readUser OK - But user not found (1 ms)
    √ createUser OK (1 ms)
    √ isCorrectPassword OK (1 ms)
    √ generateToken OK (2 ms)
    √ updateUser OK (1 ms)
    √ deleteUser OK (1 ms)
    √ deleteUser KO - User not found (1 ms)

PASS test/unit/models/user.model.spec.js
  User Model
    √ Find User by username (6 ms)
    √ Create user (3 ms)
    √ Create user KO - missing required field (1 ms)
    √ Update user (1 ms)
    √ Delete user (4 ms)

PASS test/unit/middlewares/auth.spec.js
  Auth Middleware
    √ Auth KO - Token not provided (32 ms)
    √ Auth KO - Token is invalid (13 ms)
    √ Auth KO - Token has expired (20 ms)
    √ Auth OK (25 ms)

PASS test/unit/middlewares/userValidation.spec.js
  User Validation middleware
    √ Requested username is the same as the authenticated username (3 ms)
    √ Requested username is not the same as the authenticated username (8 ms)

Test Suites: 6 passed, 6 total
Tests:       48 passed, 48 total
Snapshots:   0 total
Time:        5.168 s
```

4.2.5.1. Cobertura de código

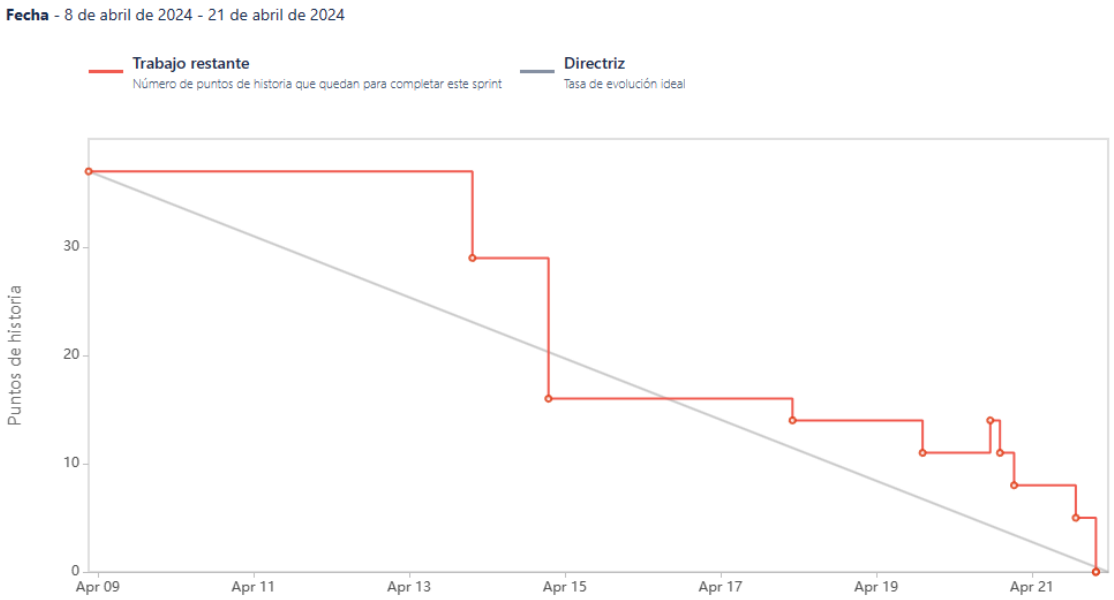
All files

96.75% Statements 179/185 **74%** Branches 37/50 **86.36%** Functions 19/22 **96.75%** Lines 179/185

File	Statements		Branches		Functions		Lines	
src	93.75%	15/16	100%	0/0	100%	1/1	93.75%	15/16
src/config	90.9%	10/11	62.5%	10/16	100%	0/0	90.9%	10/11
src/controllers	100%	59/59	100%	18/18	100%	5/5	100%	59/59
src/errors	100%	16/16	100%	0/0	100%	4/4	100%	16/16
src/middlewares	97.05%	33/34	90%	9/10	100%	3/3	97.05%	33/34
src/models	72.72%	8/11	0%	0/6	0%	0/3	72.72%	8/11
src/routes	100%	16/16	100%	0/0	100%	0/0	100%	16/16
src/services	100%	22/22	100%	0/0	100%	6/6	100%	22/22

4.2.6. *Burn-down chart*

A continuación, se presenta el gráfico de pendiente que comenzó con un total de 32 puntos de historia.



En él se puede observar como la primera semana se realizan las dos historias con mayor peso correspondientes al registro y la autenticación llegando al comienzo de la segunda semana incluso por debajo de la línea ideal.

En la segunda semana se continua con la entrega de historias, esta vez de un peso menor, por lo que se pueden observar más entregas, pero un bajada más lenta y progresiva. Algo a destacar es que al final del *sprint* existe un pendiente ascendente, ya que con el *sprint* iniciado se incorpora la historia técnica TFM-17, que corresponde a la corrección de los problemas con la integración de las diferentes herramientas y GitHub y que se comentará más en profundidad en la siguiente sección.

4.2.7. *Sprint review*

A continuación, se muestra la tabla con las historias planificadas para este *sprint*:

Historia	Criterios de aceptación	Entregada/ Aceptada
TFM-11 Registro de usuario	**Dado** un cliente de la aplicación **Cuando** quiere darse de alta como usuario **Entonces** dispone de un formulario de registro	Sí / Sí
	Dado un cliente de la aplicación **Cuando** completa el formulario de registro con su username, email, contraseña dos veces OK **Entonces** se da de alta su usuario y se le redirige al login para que se autentique	
	Dado un cliente de la aplicación **Cuando** completa alguno de los campos del formulario de registro incorrectamente **Entonces** se muestra el error correspondiente y se permite modificar los datos del formulario para corregirlos y completar el registro	
TFM-12 Autenticación de usuarios	**Dado** un usuario **Cuando** intenta acceder a una funcionalidad de la aplicación a excepción de visualizar el home **Entonces** no tiene permisos y se le redirige al formulario login para que se autentique	Sí / Sí
	Dado un cliente de la aplicación **Cuando** completa el formulario de login con su username y contraseña OK **Entonces** está autenticado en la aplicación y se le redirige al inicio	
	Dado un cliente de la aplicación **Cuando** completa alguno de los campos del formulario de login incorrectamente **Entonces** se muestra error en el inicio de sesión y se le permite seguir intentándolo	
	Dado un usuario logeado **Cuando** pulsa en el botón para cerrar sesión en la cabecera **Entonces** cierra sesión de su cuenta y se le redirige al login	
TFM-13 Consultar "Mi perfil"	**Dado** un usuario logeado **Cuando** pulsa en "Mi perfil" situado en la cabecera **Entonces** se muestra los datos de su perfil teniendo opciones para editarlos y para borrar su usuario	Sí / Sí

TFM-14 Editar usuario	**Dado** un usuario en "mi perfil" **Cuando** pulsa el botón de editar **Entonces** puede editar todos los campos menos el nombre de usuario (y contraseña), y guardar o cancelar los cambios realizados.	Sí / Sí
TFM-15 Eliminar usuario	**Dado** un usuario en "Mi perfil" **Cuando** pulsa en eliminar y confirma que desea eliminar su cuenta **Entonces** su usuario se elimina	Sí / Sí
TFM-16 Diseño de la página de inicio	N/A	Sí / Sí

Por otro lado, está la historia técnica TFM-17 para solucionar los problemas con la cuenta de GitHub que ya venían arrastrados del anterior *sprint* y que impedían utilizar GitHub con GitHub Actions, Jira, SonarCloud, etc.

Como tras casi dos semanas, el soporte de GitHub no resolvió el problema, se opta por crear una nueva cuenta y de nuevo realizar toda la configuración del ecosistema con la nueva cuenta. En esto se incluye la creación de los repositorios en la nube para el Back-End y Front-End, conexión de los repositorios con Jira Software y configuración de los flujos de integración continua.

Se ha realizado correctamente y a fin del *sprint* 1 el ecosistema está perfectamente montado y funcionando. El inconveniente es que está tarea se ha incorporado a mitad del *sprint* y no en la planificación del mismo al estar esperando una solución del equipo de GitHub, por tanto, esto ha supuesto un esfuerzo extra para acometerla y aun así entregar el resto de historias planificadas.

4.2.8. Retrospectiva

Por último, para finalizar este *sprint* 1 se realiza la ceremonia de retrospectiva para analizar cómo ha ido y posibles mejoras para los siguientes *sprints*.

	¿Qué ha pasado?	**Acciones**
¿Qué fue bien?	Optar por una autenticación basada en token JWT, por su conocimiento previo para la implementación, así como todas las ventajas que tiene de seguridad, compatibilidad, estado, etc.	-

	Buen desglose de las historias de usuario en tareas más concretas y abordables para mayor sencillez a la hora de acometerlas	-
¿Qué se puede mejorar?	Inclusión de la tarea TFM-17 en mitad del *sprint*	Pese a tratarse de un problema de una de las herramientas de terceros utilizada, y que la solución dependía del soporte que ellos brindasen, se debería de haber tenido en cuenta dentro de la planificación del *sprint* para no ir tan apurado con la entrega del resto de historias.
¿Qué fue mal?	-	-

4.3. *Sprint 2*

El objetivo del tercer *sprint,* número 2 por comenzar en el *sprint* 0, es la gestión de propiedades por parte del usuario de la aplicación. Por tanto, las historias que se van a realizar durante este *sprint* 2 están englobadas dentro de la épica TFM-23 Gestión de Propiedades.

4.3.1. *Sprint backlog*

En la planificación del *sprint* se incluyen las siguientes historias de usuario:

TFM-18 Abrir "Mis propiedades"					
Descripción	**Como** usuario autenticado **Quiero** tener una funcionalidad de "mis propiedades" accesible desde la cabecera **Para** listar mis propiedades y tener acceso a funcionalidades de visualización, creación, edición y borrado de propiedades				
Prioridad	Alta	**Puntos de historia**	13	**Tiempo consumido**	15h
Criterios de aceptación	**Dado** un usuario logeado **Cuando** pulsa en "Mis Propiedades" situado en la cabecera **Entonces** se muestra un listado con las propiedades del usuario				
	Dado un usuario en "Mis propiedades" **Cuando** pulsa en un elemento de la lista de propiedades **Entonces** se muestran los detalles de esa propiedad				

Tareas	
Nombre de la tarea	**Tipo**
Crear los diferentes modelos de propiedades y asociarlos al modelo de usuario	BBDD
Listar propiedades de usuario	Back-End
Diseño interfaz "Mis propiedades", conexión con Back-End para listar las propiedades del usuario	Front-End
Leer una propiedad	Back-End
Ver los detalles de una propiedad del listado	Front-End

TFM-20 Crear propiedad					
Descripción	**Como** usuario en "mis propiedades" **Quiero** poder crear nuevas propiedades **Para** su gestión y utilización del resto de funcionalidades				
Prioridad	Alta	**Puntos de historia**	8	**Tiempo consumido**	10h
Criterios de aceptación	**Dado** un usuario en "Mis propiedades" **Cuando** pulsa crear una nueva propiedad **Entonces** puede elegir el tipo de propiedad a crear				

Tareas	
Nombre de la tarea	**Tipo**

Creación de propiedad teniendo en cuenta que puede ser de distintos tipos (casa, piso, garaje…)	Back-End
Interfaz para la creación de propiedades modular para que el usuario pueda crear los distintos tipos de propiedades desde un solo formulario	Front-End

TFM-21 Editar propiedad					
Descripción	**Como** usuario en "mis propiedades" **Quiero** poder editar propiedades **Para** actualizar y corregir datos de estas				
Prioridad	Alta	**Puntos de historia**	8	**Tiempo consumido**	0h (No comenzada)
Criterios de aceptación	**Dado** un usuario en "Mis propiedades" **Cuando** pulsa en editar una de sus propiedades en el listado **Entonces** puede editar la información de dicha propiedad				

Tareas	
Nombre de la tarea	**Tipo**
Edición de propiedad teniendo en cuenta que puede ser de distintos tipos (casa, piso, garaje…)	Back-End
Interfaz para la edición de propiedades modular para que el usuario pueda actualizar los distintos tipos de propiedades desde un solo formulario	Front-End

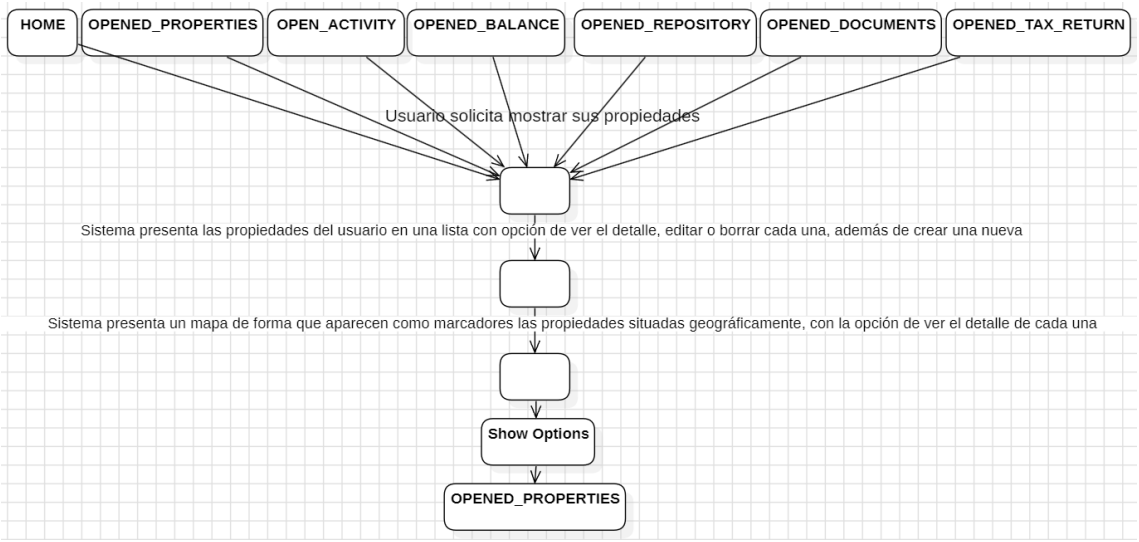
TFM-22 Eliminar propiedad					
Descripción	**Como** usuario en "mis propiedades" **Quiero** poder eliminar propiedades **Para** dejar de operar con ellas dentro de la aplicación web				
Prioridad	Alta	**Puntos de historia**	3	**Tiempo consumido**	3h
Criterios de aceptación	**Dado** un usuario en "Mis propiedades" **Cuando** pulsa borrar sobre una de las propiedades del listado **Entonces** confirma que quiere borrar y se eliminar la propiedad				

Tareas	
Nombre de la tarea	**Tipo**
Endpoint para borrar una propiedad	Back-End
Funcionalidad de borrado de propiedad con dialogo de confirmación para el frontal	Front-End

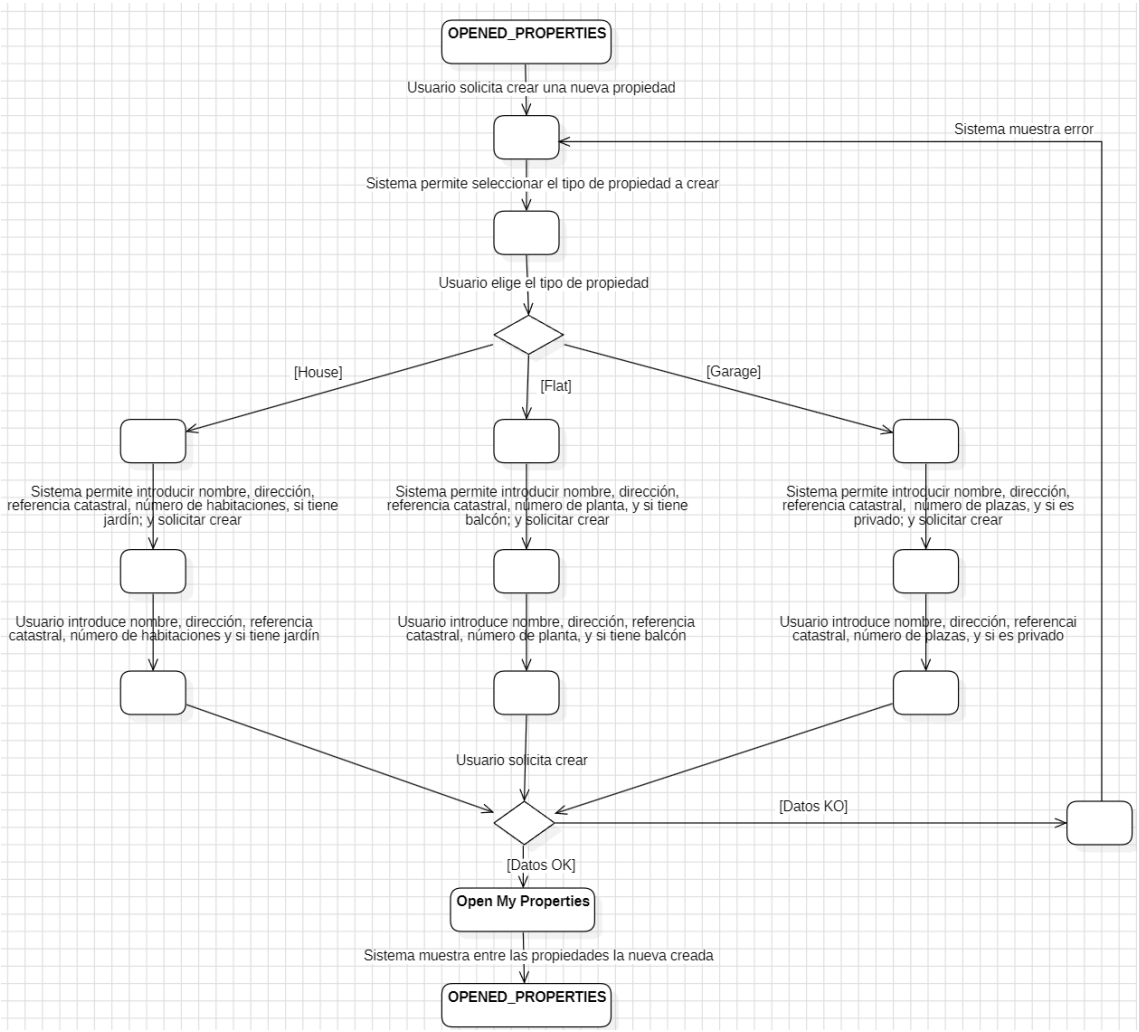
4.3.2. Diseño de la arquitectura

4.3.2.1. Diagramas de casos de uso

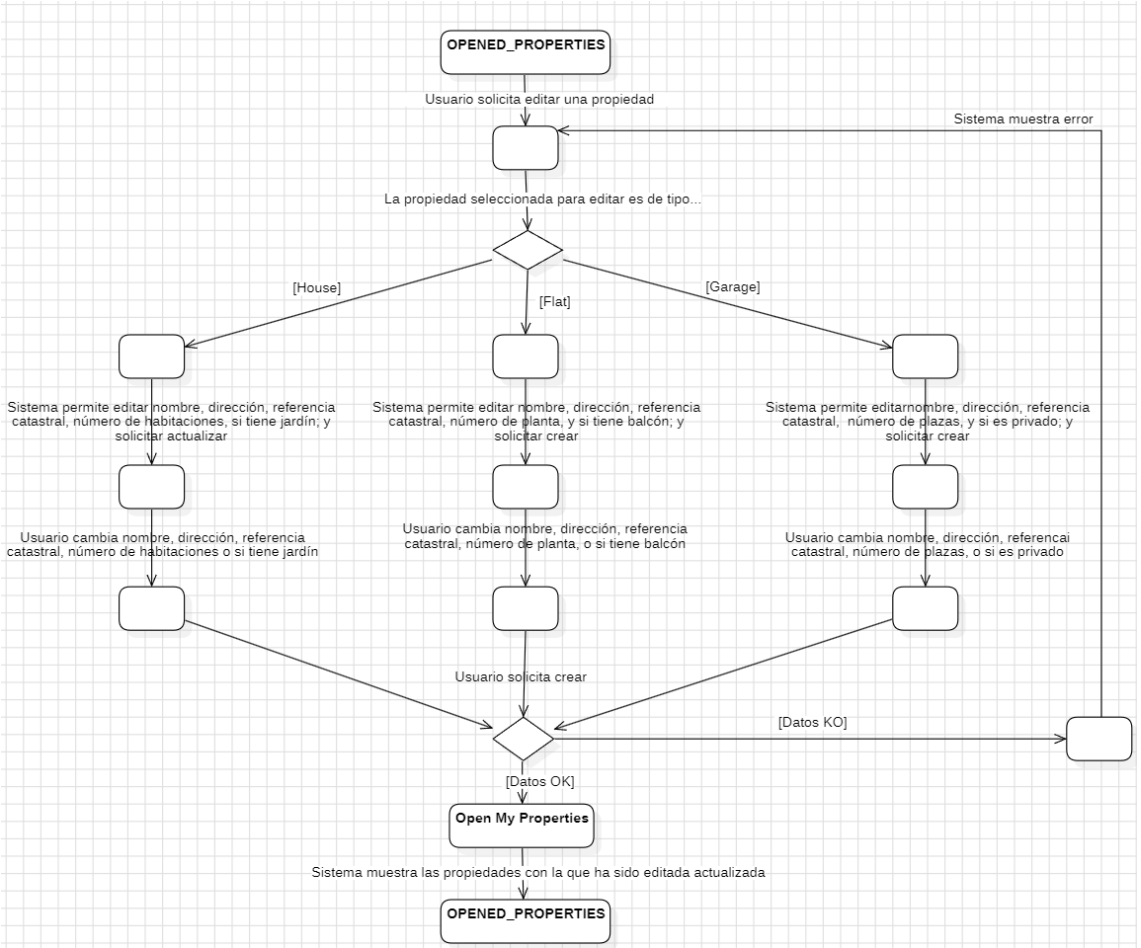
Abrir mis propiedades (OpenMyProperties):



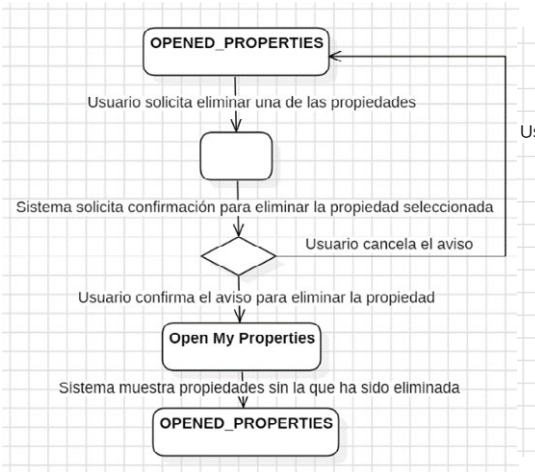
Crear propiedad (CreateProperty):

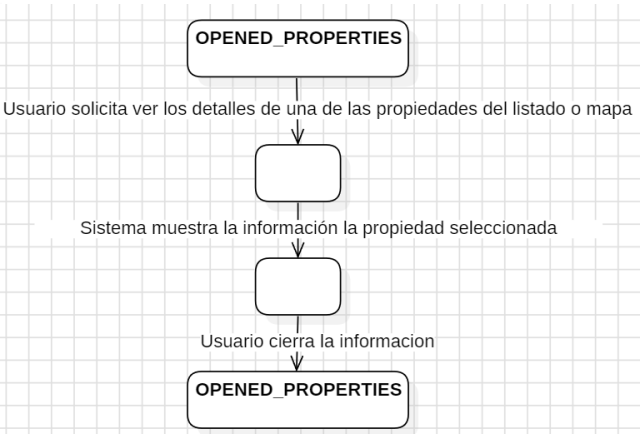


Editar propiedad (UpdateProperty):

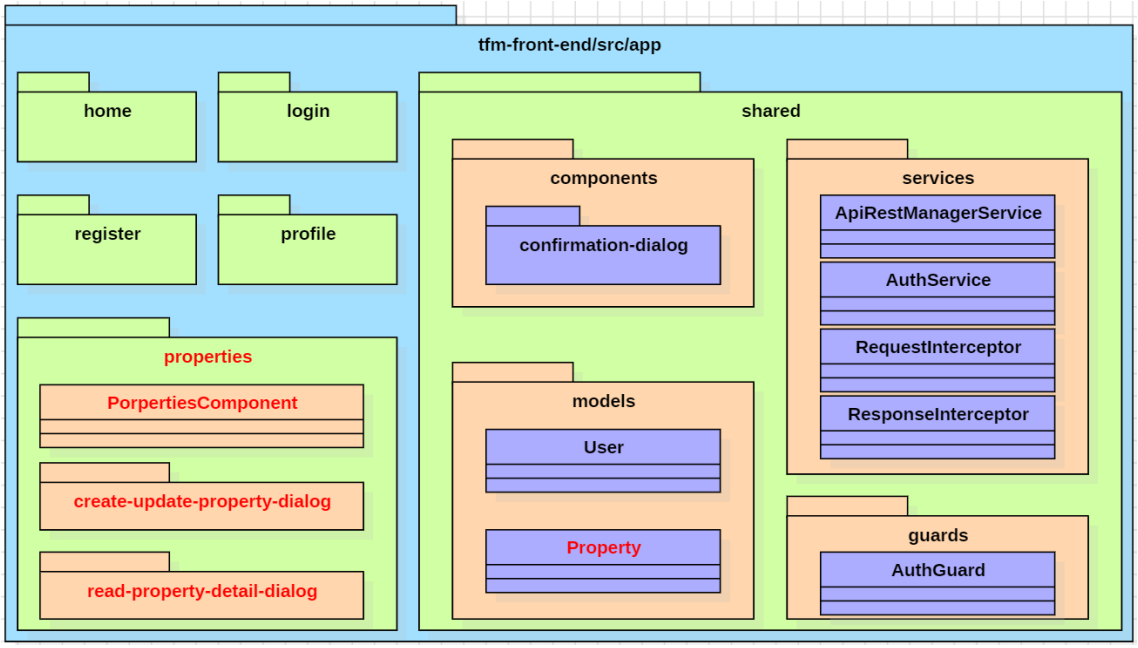


Eliminar propiedad (DeleteProperty) y ver detalles (ViewPropertyDetails):





4.3.2.2.a. Diagrama de paquetes (Front)

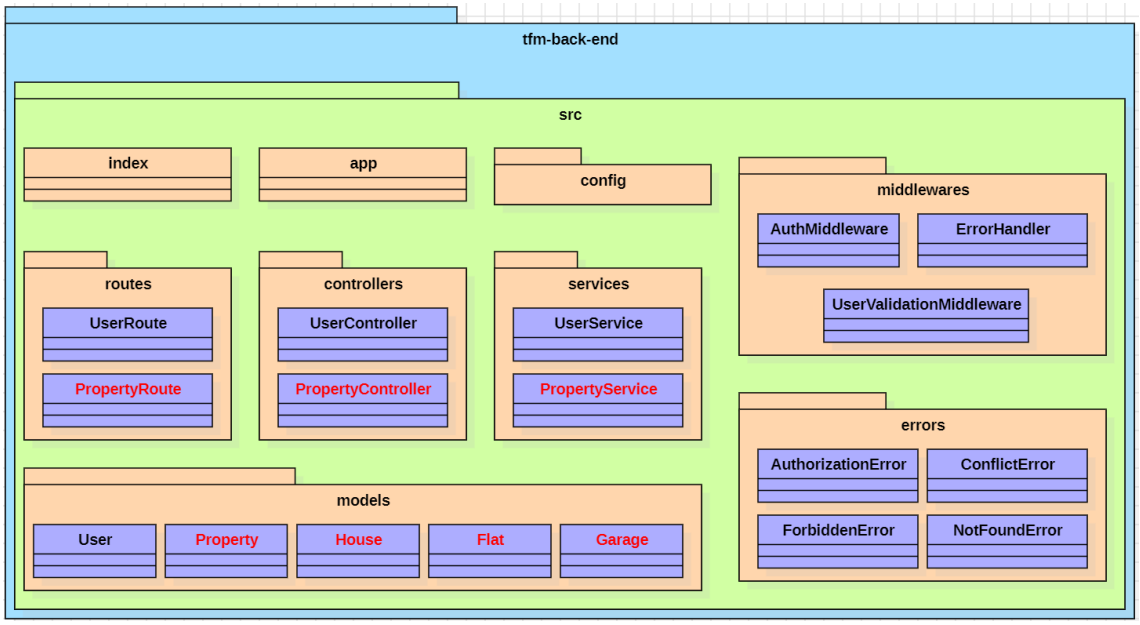


Se añade el modelo de **Propiedad**, en el mismo fichero se encuentran los distinto tipos de propiedad como casa, piso o garaje.

Dentro de la carpeta **properties** está toda la interfaz para la gestión de propiedades en la aplicación:

Por un lado, se tiene el componente **PropertiesComponent** (con su hoja de estilos y plantilla HTML) que es aquel que se invoca cuando se pulsa en "Mis propiedades". En este componente se listan todas las propiedades, y se puede acceder a las siguientes funcionalidades: leer los detalles de una propiedad, que utilizará el componente **ReadPropertyDetailDialogComponent** para abrir un *pop up* con los detalles; crear y editar propiedades, para lo que se utiliza el componente **CreateUpdatePropertyDialogComponent** que por el momento solo tiene la funcionalidad de crear ya que no ha dado tiempo a realizar la edición y será parte del siguiente *sprint*; eliminar propiedades; y en el próximo *sprint* también se incorporará el mapa.

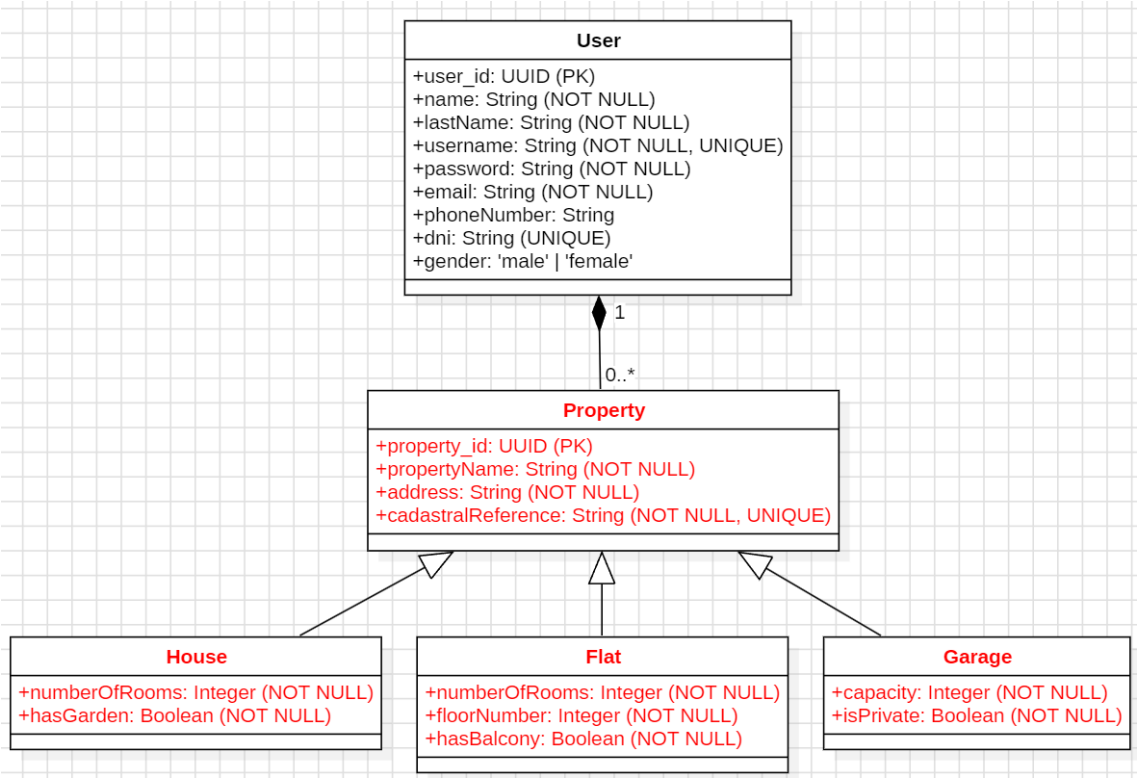
4.3.2.2.b. Diagrama de paquetes (Back)



Durante este *sprint* se crean los nuevos modelos de Sequelize para las propiedades (casa, piso y garaje) donde cada uno representa una nueva tabla en base de datos, así como las relaciones con los modelos ya existentes, en la siguiente sección se habla más acerca del modelo de datos.

Igualmente a lo que sucedía con la gestión de usuarios, se crea: **PropertyRoute**, para responder a las peticiones HTTP relacionadas con las propiedades; **PropertyController**, para procesar dichas peticiones; y **PropertyService**, con la lógica de negocio para la gestión de propiedades.

4.3.2.3. Modelo de datos



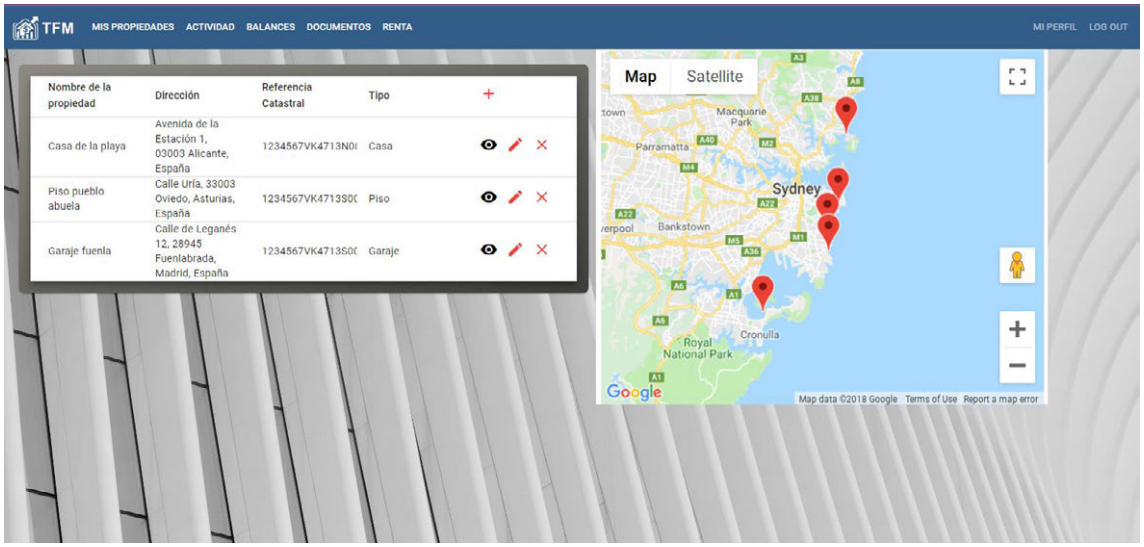
Se establece una relación de composición entre el modelo ya existente de Usuario y el recién creado de Propiedad, con una multiplicidad de 1 a *n*, es decir, un usuario puede tener *n* propiedades, mientras que una propiedad solo puede pertenecer a un usuario.

Los modelos Casa, Piso y Garaje heredan de Propiedad. La herencia en base de datos relacionales como MySQL no se implementa de manera nativa, como sucede en los sistemas orientados a objetos, por tanto, se simula utilizando "*Table per Type*" (Tabla por tipo). De esta forma se definen los cuatro modelos: Propiedad, Casa, Piso y Garaje, siendo la relación entre Casa/Piso/Garaje de uno a uno con Propiedad, es decir, se mantiene una tabla separada para cada subclase (Casa, Piso, Garaje) con una relación de clave foránea que las une a una tabla de superclase común (Propiedad).

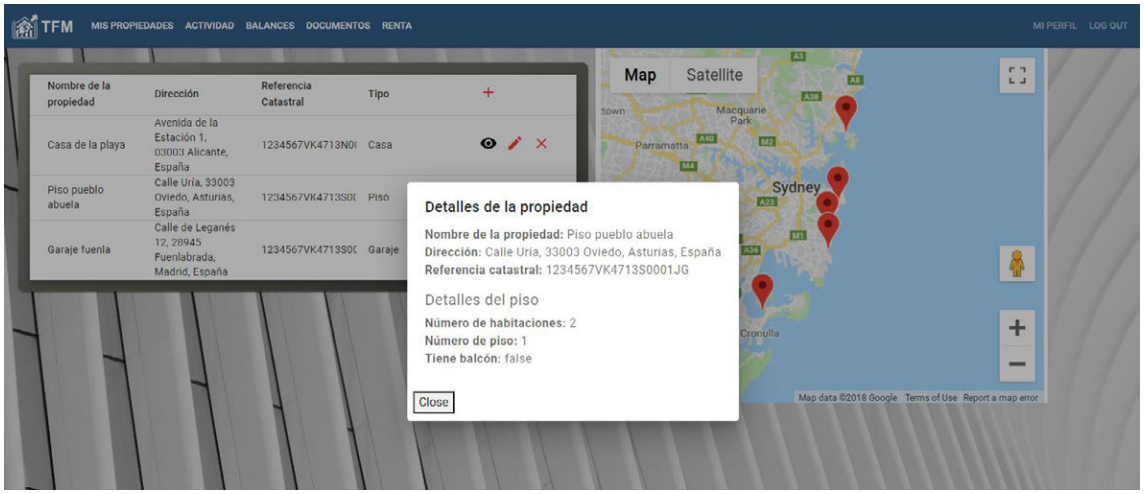
4.3.3. Interfaz

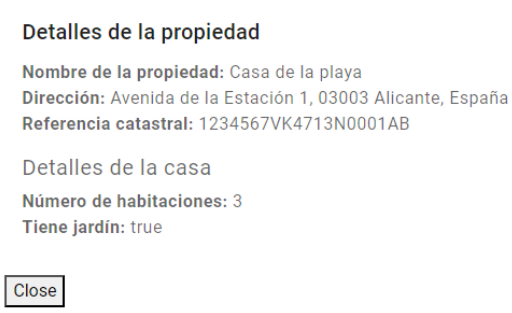
Mis propiedades:

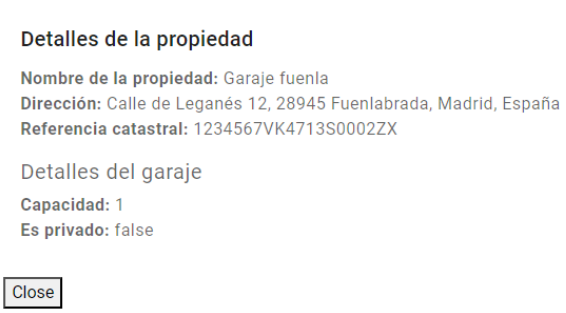
Se listan las propiedades del usuario y se dispone de botones para crear, leer, editar y borrar. La imagen del mapa simplemente guarda hueco en la interfaz para cuando se coloque el mapa de verdad.



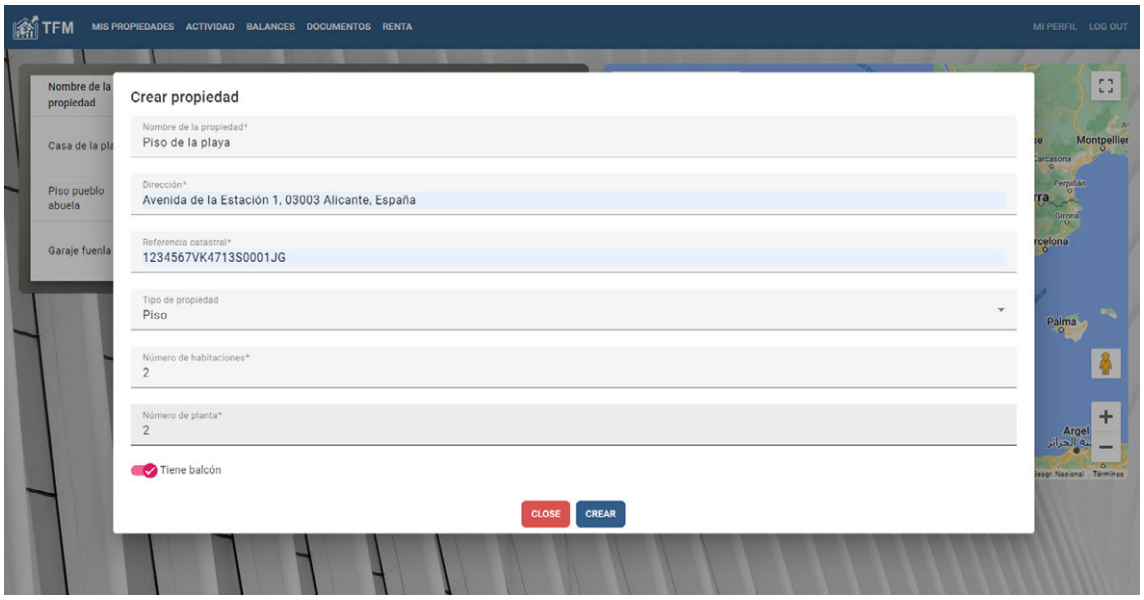
Leer detalles de una propiedad:

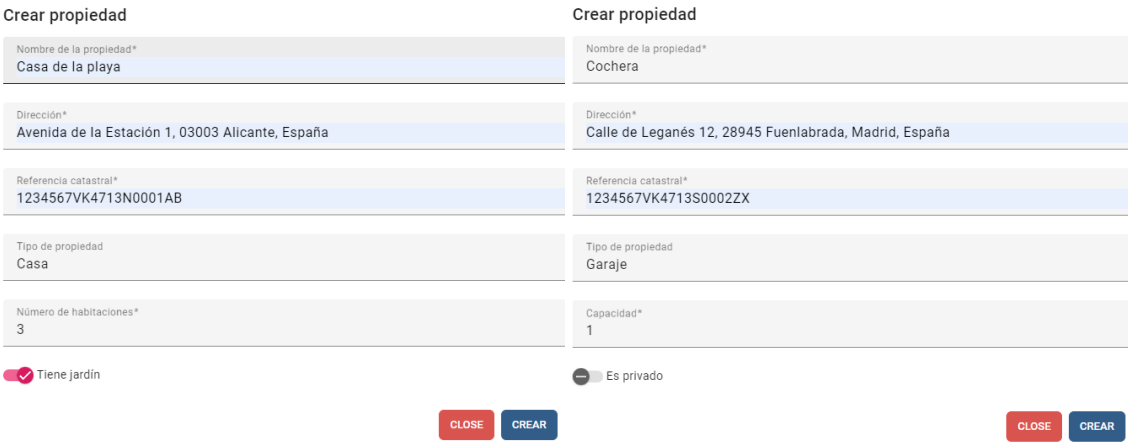




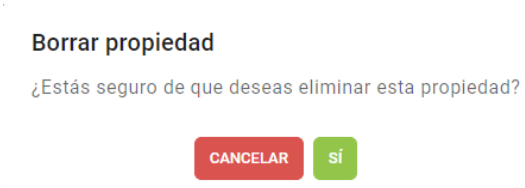


Crear propiedad:





Borrar propiedad:



4.3.4. *Endpoints*

Por supuesto, de acorde al objetivo del *sprint* de realizar la gestión de propiedades, los distintos *endpoints* añadidos son para cumplir dicho propósito, habiendo añadido los siguientes *endpoints*.

LISTAR PROPIEDADES			
Endpoint			
Método	**Ruta**		**Descripción**
GET	/properties		Lista las propiedades del usuario autenticado
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
-	-	-	El usuario autenticado se obtiene mediante el token
Respuestas			
Código	**Mensaje**		**Descripción**
200	OK		Devuelve la lista de propiedades asociadas al usuario autenticado
401	UNAUTHORIZED		Token no proporcionado, expirado o inválido

LEER PROPIEDAD			
Endpoint			
Método	**Ruta**		**Descripción**
GET	/properties/{property_id}		Lee la información de la propiedad seleccionada
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
property_id*	String	Path	ID de la propiedad a leer
Respuestas			
Código	**Mensaje**		**Descripción**
200	OK		Devuelve la información de la propiedad
401	UNAUTHORIZED		Token no proporcionado, expirado o inválido
403	FORBIDDEN		Acceso denegado, la propiedad solicitada no pertenece al usuario autenticado
404	NOT FOUND		No existe una propiedad con ese ID

CREAR PROPIEDAD			
Endpoint			
Método	**Ruta**		**Descripción**
POST	/properties		Se crea una propiedad para el usuario autenticado
Parámetros			
Nombre	**Tipo**	**Ubi.**	**Descripción**
propertyName*	String	Body	Nombre de la propiedad que a crear
address*	String	Body	Dirección de la propiedad a crear
cadastralReference	String	Body	Referencia catastral de la

*			propiedad a crear
houseDetails	Object `"houseDetails": { "numberOfRooms": 2, "hasGarden": false }`	Body	Si la propiedad es una casa, los detalles de esta. Contiene: numberOfRooms*: Integer y hasGarden*: Boolean.
flatDetails	Object `"flatDetails": { "numberOfRooms": 2, "floorNumber": 1, "hasBalcony": true }`	Body	Si la propiedad es un piso, los detalles de este. Contiene: numberOfRooms*: Integer y floorNumber*: Integer y hasBalcony*: String.
garageDetails	Object `"garageDetails": { "capacity": "1", "isPrivate": "true" }`	Body	Si la propiedad es un garaje, los detalles de este. Contiene: capacity*: Integer y isPrivate*: Boolean.

Respuestas		
Código	**Mensaje**	**Descripción**
201	OK	Propiedad creada correctamente
400	BAD REQUEST	Algún parámetro del cuerpo es incorrecto
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido

BORRAR PROPIEDAD			
Endpoint			
Método	**Ruta**		**Descripción**
DELETE	/properties/{property_id}		Borrar la propiedad seleccionada
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
property_id*	String	Path	ID de la propiedad a borrar
Respuestas			
Código	**Mensaje**		**Descripción**
200	OK		Propiedad borrada, devuelve mensaje de confirmación
401	UNAUTHORIZED		Token no proporcionado, expirado o inválido
403	FORBIDDEN		Acceso denegado, la propiedad que se quiere borrar no pertenece al usuario autenticado
404	NOT FOUND		No existe una propiedad con ese ID

El *endpoint* de editar propiedad se detallará cuando se realice durante el siguiente *sprint*.

4.3.5. Pruebas unitarias y de integración

```
PASS test/integration/user.spec.js
  Users integration test
    √ Read user KO - User not found (58 ms)
    √ Create user OK (205 ms)
    √ Create user KO - Username already exists (19 ms)
    √ Read user OK (18 ms)
    √ Create user KO - Bad request (157 ms)
    √ Create user KO - Bad request no password (17 ms)
    √ Login user OK (142 ms)
    √ Login user KO - No password (13 ms)
    √ Login user KO - User not found (17 ms)
    √ Login user KO - Incorrect password (158 ms)
    √ Update user OK (33 ms)
    √ Update user KO - User not found (17 ms)
    √ Delete user OK (20 ms)
    √ Delete user KO - User not found (15 ms)

PASS test/unit/controllers/user.controller.spec.js
  User Controller
    √ Read user OK (29 ms)
    √ Read user KO - User not found (20 ms)
    √ Create user OK (31 ms)
    √ Create user KO - Username already exists (22 ms)
    √ Create user KO - No password (20 ms)
    √ Create user KO - Bad Request no password (22 ms)
    √ Create user KO - Bad Request wrong email format (16 ms)
    √ Login user OK (18 ms)
    √ Login user KO - No username (16 ms)
    √ Login user KO - User not found (18 ms)
    √ Login user KO - Incorrect password (16 ms)
    √ Update user OK (15 ms)
    √ Update user KO - User not found (14 ms)
    √ Delete user OK (15 ms)
    √ Delete user KO - User not found (14 ms)

PASS test/integration/property.spec.js
  Property integration test
    √ Read properties of user OK - No properties (40 ms)
    √ Create property OK (53 ms)
    √ Create property KO - Bad request (16 ms)
    √ Read properties of user OK (18 ms)
    √ Read property OK (30 ms)
    √ Read property KO - Property not found (24 ms)
    √ Delete property OK (30 ms)
    √ Delete property KO - Property not found (19 ms)

PASS test/unit/controllers/property.controller.spec.js
  Property Controller
    √ Read properties of user OK (31 ms)
    √ Read properties of user KO (16 ms)
    √ Read property OK (13 ms)
    √ Read property KO - Property not found (13 ms)
    √ Read property KO - Forbidden (14 ms)
    √ Create property OK (29 ms)
    √ Create property KO - Bad request (21 ms)
    √ Delete property KO - Forbidden (14 ms)
    √ Delete property OK (14 ms)
    √ Delete property KO - Property not found (14 ms)

PASS test/unit/services/user.service.spec.js
  User Service
    √ readUser OK (5 ms)
    √ readUser OK - But user not found (3 ms)
    √ createUser OK (1 ms)
    √ isCorrectPassword OK (1 ms)
    √ generateToken OK (3 ms)
    √ updateUser OK (1 ms)
    √ deleteUser OK (1 ms)
    √ deleteUser KO - User not found (1 ms)

PASS test/unit/models/user.model.spec.js
  User Model
    √ Find User by username (11 ms)
    √ Create user (3 ms)
    √ Create user KO - missing required field (2 ms)
    √ Update user (1 ms)
    √ Delete user (2 ms)

PASS test/unit/middlewares/auth.spec.js
  Auth Middleware
    √ Auth KO - Token not provided (34 ms)
    √ Auth KO - Token is invalid (15 ms)
    √ Auth KO - Token has expired (26 ms)
    √ Auth OK (17 ms)

PASS test/unit/models/property.model.spec.js
  Property Model
    √ Find all properties (8 ms)
    √ Find Property by property_id (2 ms)
    √ Create property (3 ms)
    √ Create property missing parameter (1 ms)
    √ Create property with missing required field (1 ms)
    √ Delete property (2 ms)

PASS test/unit/middlewares/userValidation.spec.js
  User Validation middleware
    √ Requested username is the same as the authenticated username (3 ms)
    √ Requested username is not the same as the authenticated username (9 ms)

PASS test/unit/services/property.service.spec.js
  Property Service
    √ Return properties by user id (9 ms)
    √ Read property by id (2 ms)
    √ Create property House (2 ms)
    √ Create property Flat (2 ms)
    √ Create property Garage (2 ms)
    √ Delete property by id (1 ms)
    √ Delete property by id - Property not found (2 ms)
    √ Validate property ownership (2 ms)
    √ Validate property ownership - Property not found (19 ms)
    √ Validate property ownership - User is not the owner (1 ms)

Test Suites: 10 passed, 10 total
Tests:       82 passed, 82 total
Snapshots:   0 total
Time:        10.222 s
```

4.3.5.1. Cobertura de código

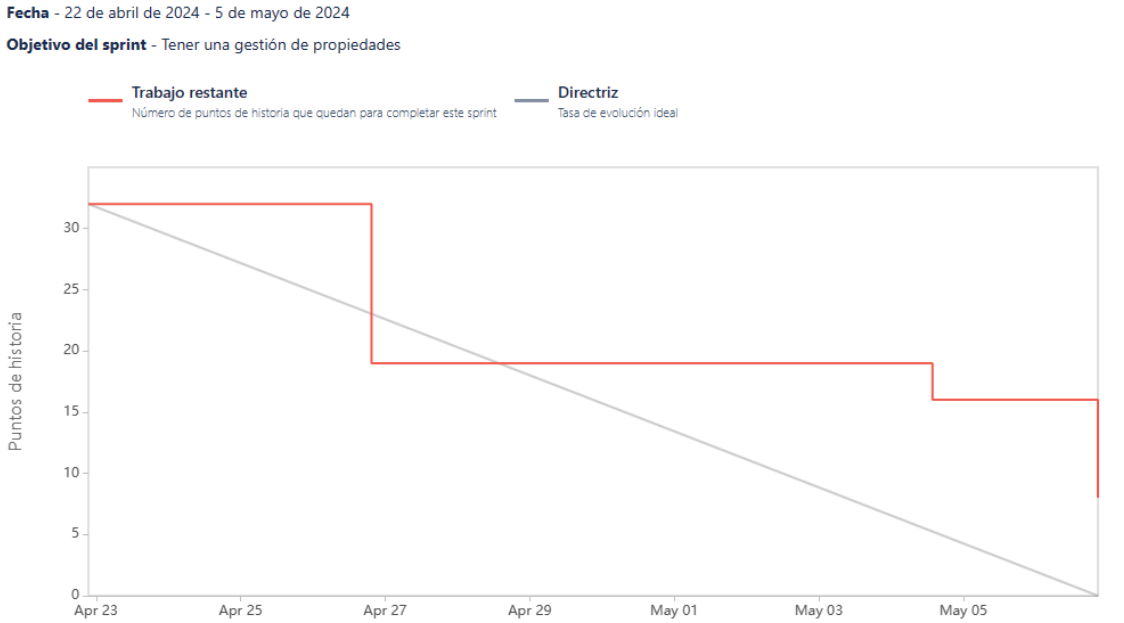
All files

98.16% Statements 321/327 **82.6%** Branches 76/92 **91.66%** Functions 33/36 **98.14%** Lines 318/324

File	Statements		Branches		Functions		Lines	
src	93.75%	15/16	100%	0/0	100%	1/1	93.75%	15/16
src/config	90.9%	10/11	62.5%	10/16	100%	0/0	90.9%	10/11
src/controllers	100%	117/117	100%	34/34	100%	10/10	100%	117/117
src/errors	100%	16/16	100%	0/0	100%	4/4	100%	16/16
src/middlewares	97.05%	33/34	90%	9/10	100%	3/3	97.05%	33/34
src/models	92.3%	36/39	25%	3/12	50%	3/6	92.3%	36/39
src/routes	100%	29/29	100%	0/0	100%	0/0	100%	29/29
src/services	100%	65/65	100%	20/20	100%	12/12	100%	62/62

4.3.6. *Burn-down chart*

A continuación, se presenta el gráfico de pendiente que comenzó con un total de 32 puntos de historia.



Las historias que incluyen la creación de modelos, lectura y creación de propiedades han supuesto un gran esfuerzo al tratar con modelos con herencia, y querer realizar todo con la mayor modularidad posible, esto hace que el gráfico durante *sprint* se aleje de la diagonal ideal.

Esta complejidad hace que no se hayan entregado todas las historias planificadas para el *sprint*, quedando los 8 puntos de historia correspondientes a "TFM-21 Editar propiedad" que pasa para el *backlog* del siguiente *sprint*.

4.3.7. *Sprint review*

En la siguiente tabla se presentan las historias que se han incluido en el *backlog* del *sprint* 2 durante la planificación y se revisa si ha sido entregada y si cumple los criterios de aceptación definidos.

Historia	Criterios de aceptación	Entregada/ Aceptada
TFM-18 Abrir "Mis propiedades"	**Dado** un usuario logeado **Cuando** pulsa en "Mis Propiedades" situado en la cabecera **Entonces** se muestra un listado con las propiedades del usuario	Sí / Sí
	Dado un usuario en "Mis propiedades" **Cuando** pulsa en un elemento de la lista de propiedades **Entonces** se muestran los detalles de esa propiedad	
TFM-20 Crear propiedad	**Dado** un usuario en "Mis propiedades" **Cuando** pulsa crear una nueva propiedad **Entonces** puede elegir el tipo de propiedad a crear	Sí / Sí
TFM-21 Editar propiedad	**Dado** un usuario en "Mis propiedades" **Cuando** pulsa en editar una de sus propiedades en el listado **Entonces** puede editar la información de dicha propiedad	No / -
TFM-22 Eliminar propiedad	**Dado** un usuario en "Mis propiedades" **Cuando** pulsa borrar sobre una de las propiedades del listado **Entonces** confirma que quiere borrar y se eliminar la propiedad	Sí / Sí

Como se ve en la tabla tres de las cuatro historias que se habían planificado para este *sprint* se entregan y además se aceptan de acuerdo a los criterios de aceptación definidos para cada una de las historias. Por otro lado, existe una cuarta historia de usuario que ni siquiera ha sido empezada, por lo tanto, tampoco se entrega y pasa de forma íntegra al *sprint* 3.

Las mayores dificultades que se han encontrado durante el desarrollo de este *sprint* ha sido la modelización de base de datos. Pese a tener el esquema claro antes de comenzar, la implementación ha supuesto complicaciones. La falta de experiencia con las relaciones en Sequelize ORM ha demorado la entrega de las historias de este *sprint*. Como parte positiva, una vez superada esta barrera, se prevé que se facilite las futuras incorporaciones.

La parte frontal aunque también ha supuesto esfuerzo, finalmente se ha conseguido el objetivo de que el usuario pueda operar con los distintos tipos de propiedad de forma cómoda y con un estilo adaptado al resto de la web. De cara al desarrollo la unión de parte trasera y frontal resultante hace que añadir nuevos tipos de propiedad en el futuro sea muy sencillo.

4.3.8. Retrospectiva

Por último, con la retrospectiva se puede estudiar que ha sucedido y que acciones se han tomado o se pueden tomar de cara a la mejora continua.

	¿Qué ha pasado?	**Acciones**
¿Qué fue bien?	Todas las historias entregadas se hacen con la calidad precisa. Pese a que no se entreguen todas, se asume que esto es algo común en el desarrollo de un proyecto *software* y se tiene en cuenta en las previsiones iniciales.	-
¿Qué se puede mejorar?	Las relaciones en Sequelize ORM, junto a un modelo de herencia no implantable de forma nativa, demoran el desarrollo.	Poner atención extra en situaciones en las que por motivos de falta de experiencia en cierto campo pueda demorar la entrega, y hacer un estudio previo para mejor estimación, planificación y ejecución.
¿Qué fue mal?	Estimación. De los 32 puntos con los que se comienza el *sprint* se entregan 24, esto quiere decir que un 25% de los puntos de historia no son entregados.	Pese a que hasta ahora las estimaciones han sido acertadas, en este *sprint* se han desviado. Se debe aprender y tener en cuenta a la hora de estimar posibles dificultades que se pueden encontrar como ha sido el caso.

4.4. *Sprint* 3

Durante este *sprint* 3 los objetivos son diversos. Por un lado, está el objetivo de terminar con las historias de la épica TFM-23 Gestión de propiedades, tanto con la edición de propiedades del usuario, como con la incorporación de un mapa donde se sitúen todas las propiedades del usuario. Por el otro lado, comenzar con las tareas de leer y crear operaciones pertenecientes a la épica TFM-29 Gestión de operaciones.

4.4.1. *Sprint backlog*

Como se ha comentado en la sección anterior, para este *sprint* se incluye la historia TFM-21, no completada durante el *sprint* 2, quedando el *sprint backlog* de la siguiente manera:

TFM-21 Editar propiedad					
Descripción	**Como** usuario en "mis propiedades" **Quiero** poder editar propiedades **Para** actualizar y corregir datos de estas				
Prioridad	Alta	**Puntos de historia**	8	**Tiempo consumido**	5h
Criterios de aceptación	**Dado** un usuario en "Mis propiedades" **Cuando** pulsa en editar una de sus propiedades en el listado **Entonces** puede editar la información de dicha propiedad				
Tareas					
Nombre de la tarea					**Tipo**
Edición de propiedad teniendo en cuenta que puede ser de distintos tipos (casa, piso, garaje…)					Back-End
Interfaz para la edición de propiedades modular para que el usuario pueda actualizar los distintos tipos de propiedades desde un solo formulario					Front-End

TFM-19 Incorporación de mapa con la disposición de las propiedades					
Descripción	**Como** usuario en "mis propiedades" **Quiero** tener un mapa donde se disponen las propiedades **Para** situarlas con facilidad				
Prioridad	Media	**Puntos de historia**	8	**Tiempo consumido**	10h
Criterios de aceptación	**Dado** un usuario autenticado **Cuando** está en "mis propiedades" **Entonces** se muestra un mapa donde contiene marcadores con sus propiedades				
	Dado un usuario en "mis propiedades" **Cuando** pulsa uno de los marcadores del mapa **Entonces** se abre los detalles de esa propiedad				
Tareas					

Nombre de la tarea	Tipo
Incorporación de Google Maps en Angular (Formación, incorporación de servicio de Google Cloud, visualización de mapa…)	Front-End
Creación de servicio que transforme direcciones en coordenadas	Front-End
Añadir marcadores en el mapa con las propiedades y que puedan ser pulsables para abrir los detalles	Front-End

TFM-25 Abrir "actividad" (operaciones)					
Descripción	**Como** usuario autenticado **Quiero** tener una funcionalidad de actividad **Para** listar las operaciones de una propiedad y tener acceso a funcionalidades de creación, edición y borrado operaciones (ingresos y gastos)				
Prioridad	Alta	**Puntos de historia**	13	**Tiempo consumido**	10h
Criterios de aceptación	**Dado** un usuario autenticado **Cuando** accede a actividad **Entonces** puede seleccionar entre una de sus propiedades para consultar la actividad de esta				
	Dado un usuario en "Actividad" y habiendo seleccionado entre una de sus propiedades **Cuando** pulsa en añadir gasto/ingreso **Entonces** puede rellenar el formulario para crear una nueva operación				
Tareas					
Nombre de la tarea					**Tipo**
Crear y asociar el nuevo modelo de Operación					BBDD
Leer operaciones de una propiedad					Back-End
Diseño interfaz "Actividad", conexión con Back-End para listar las operaciones una vez seleccionada una de las propiedades					Front-End

TFM-26 Crear operación					
Descripción	**Como** usuario en "actividad" **Quiero** poder crear nuevos ingresos y gatos **Para** registrar cualquier transacción financiera asociada a una de mis propiedades				
Prioridad	Alta	**Puntos de historia**	8	**Tiempo consumido**	7h
Criterios de aceptación	**Dado** un usuario autenticado **Cuando** accede a actividad **Entonces** puede seleccionar entre una de sus propiedades para consultar la actividad de esta				
Tareas					

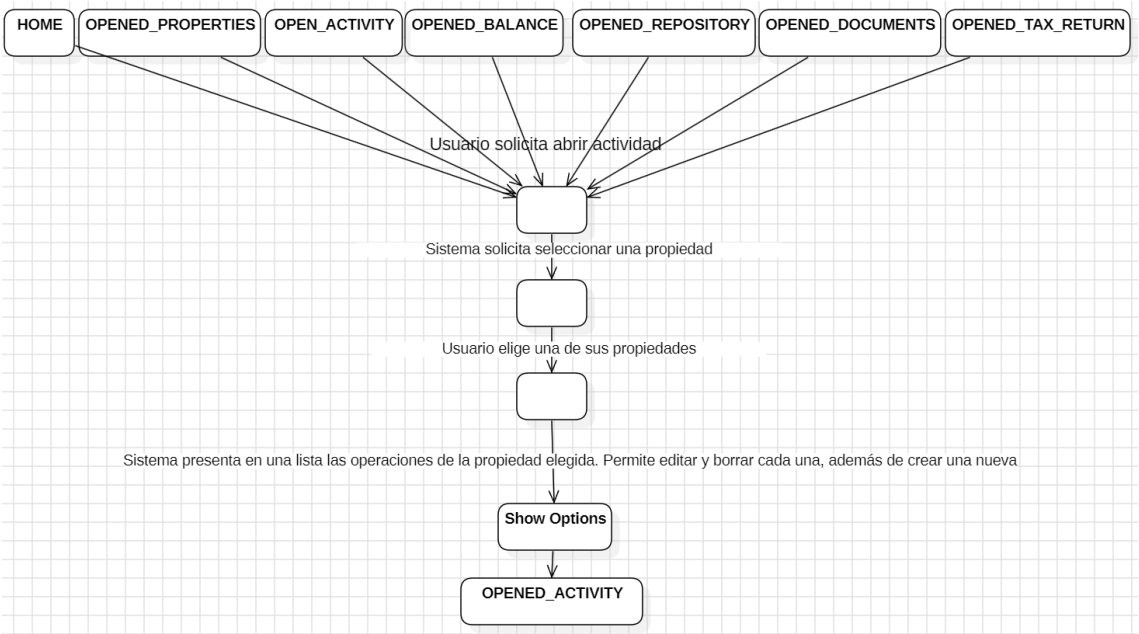
Nombre de la tarea	Tipo
Creación de una operación para una propiedad	Back-End
Interfaz y conexión con Back-End para la creación de oepraciones	Front-End

4.4.2. Diseño de la arquitectura

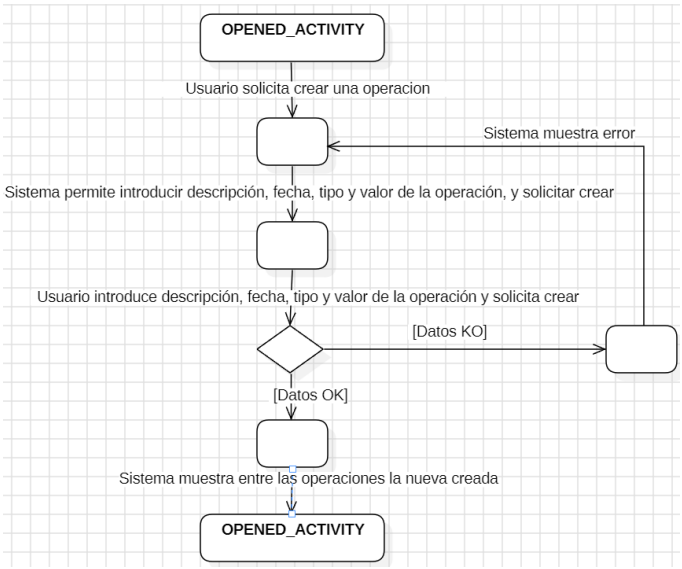
4.4.2.1. Diagramas de casos de uso

Véase diagramas de caso de uso “editar propiedad” y “abrir mis propiedades” en la sección 4.3.2.1, ya que el primero finalmente se realiza en este *sprint*, y el mapa está incluido el segundo.

Abrir actividad (OpenActivity):

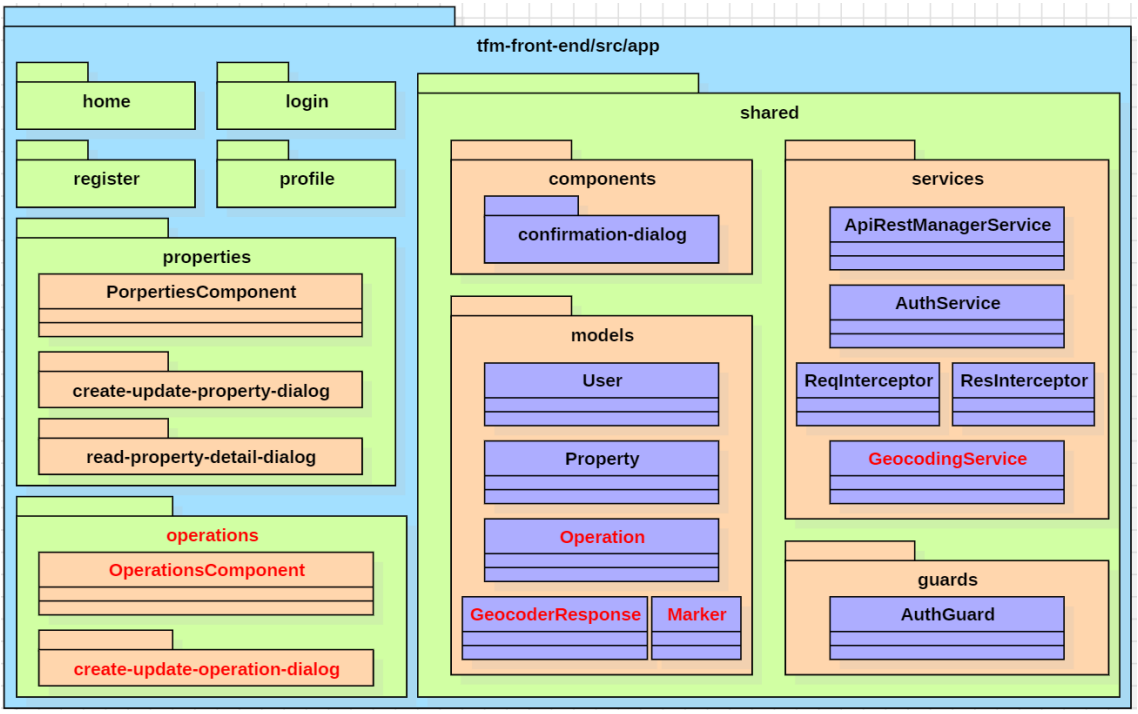


Crear operación (CreateOperation):



4.4.2.2.a. Diagrama de paquetes (Front)

En este diagrama de paquetes del Front-End se especifica en rojo aquello que ha sido añadido durante el *sprint*, pese a que en algunas historias se trabaja sobre componentes creados anteriormente como es la edición de propiedades.

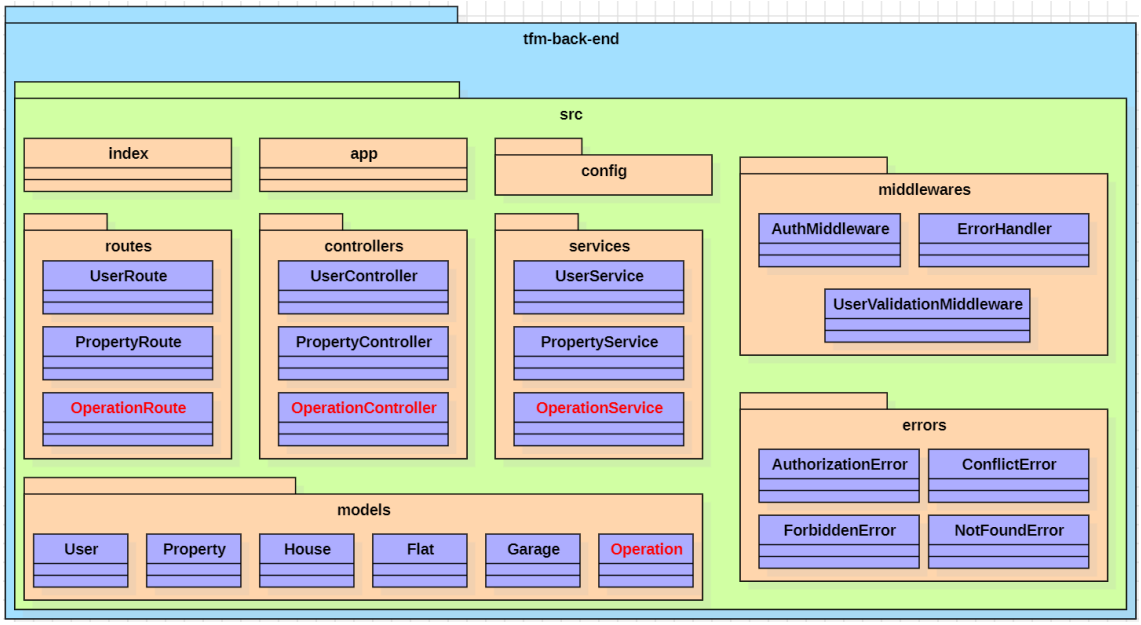


Por tanto, para finalizar con las historias de gestión de propiedades, es decir, con la edición de las mismas y la incorporación de Google Maps, el desarrollo de la interfaz se trabaja sobre los componentes ya comenzados durante el anterior *sprint* **PropertiesComponent** y **CreateUpdatePropertyDialogComponent.**

Adicionalmente, la visualización del mapa donde localizar las propiedades, hace necesaria la creación del servicio **GeocodingService**, donde se utiliza la API de Geocoding de Google Maps para trasformar direcciones en coordenadas. Muy relacionado con este servicio está el nuevo modelo **GeocoderResponse**, que, como su nombre indica, es el modelo de respuesta que se espera de la API. También relacionado con esta nueva funcionalidad, se crea el modelo **Marker** donde se define el formato de los marcadores que se sitúan en el mapa.

Para la gestión de operaciones se crea el modelo **Operation**, en la carpeta **operations** se sitúa aquello relacionado con la interfaz del manejo de operaciones. Cuando se pulsa en "Actividad" se invoca el componente **OperationsComponent**, donde se escoge una propiedad y se listan sus operaciones, aparte de poder acceder a la funcionalidad de creado, edición y borrado de estas. **CreateUpdateOperationDialogComponent** es el dialogo que contiene el formulario de creación y edición de operaciones.

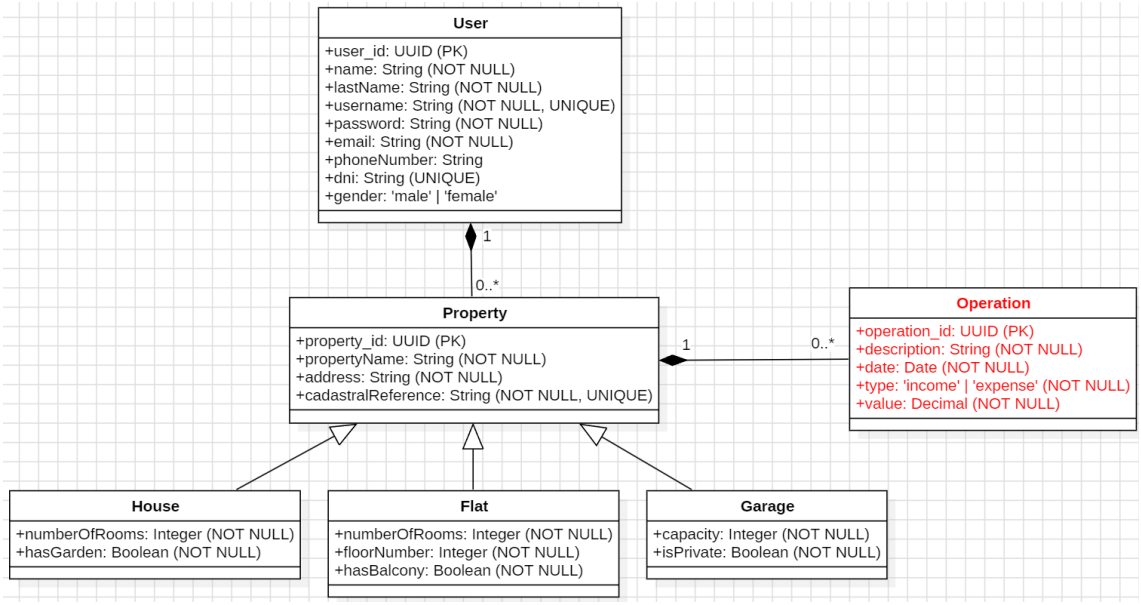
4.4.2.2.b. Diagrama de paquetes (Back)



Para la edición de propiedades se añade la ruta, controlador y servicio necesario. Mientras que, para el mapa, no se incorpora nada nuevo ya que solo usa la petición de listar propiedades ya existente.

Para la gestión de operaciones se añade el modelo, ruta, controlador y servicio igual que ya sucedía para los usuarios o las propiedades.

4.4.2.3. Modelo de datos

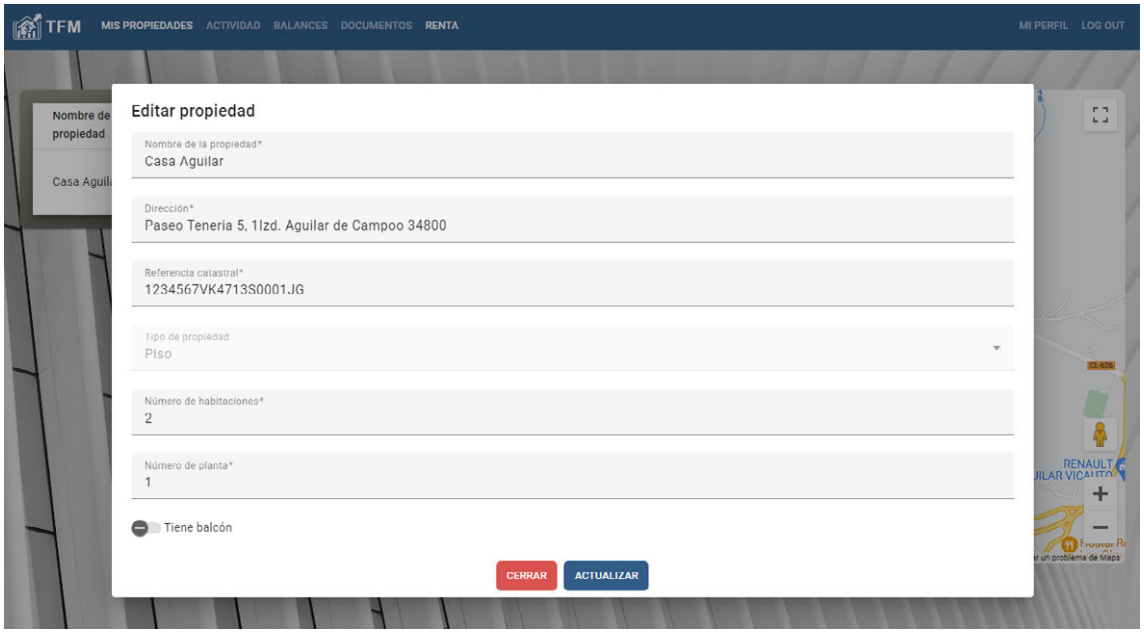


Se incorpora el nuevo modelo de datos Operación, con una relación de composición entre el modelo de Propiedad y Operación, con multiplicidad 1 a *n*. Por tanto, un usuario puede tener n propiedades, y a su vez cada propiedad puede tener n operaciones, mientras que cada operación solo puede pertenecer a una propiedad.

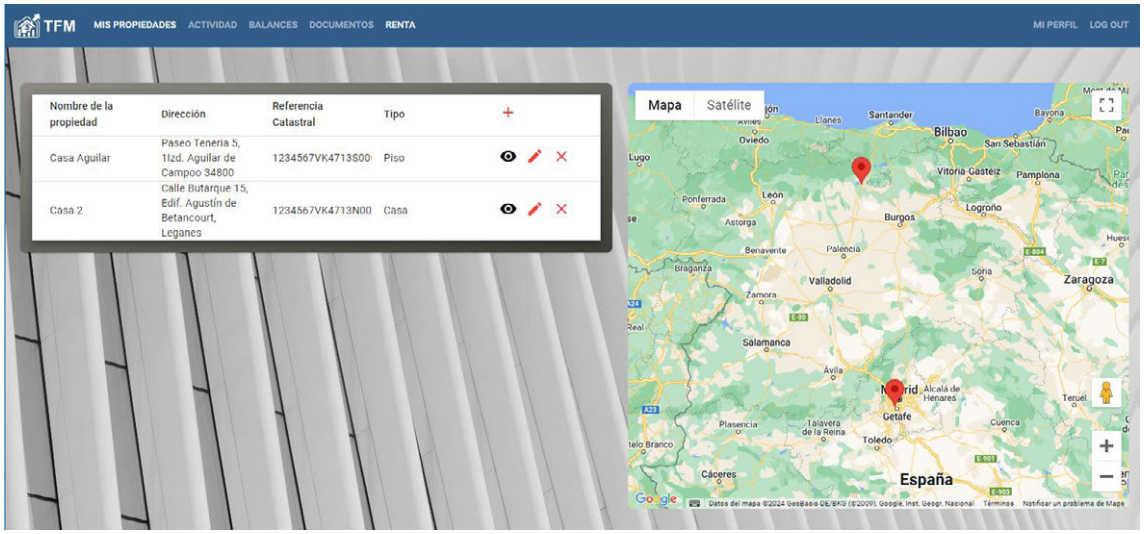
Este modelo se escoge así, de forma que cuando se utilicen las operaciones en el futuro, por ejemplo, para generar balances o para realizar la renta, el usuario este obligado a seleccionar una de sus propiedades para posteriormente poder operar con las operaciones que la componen.

4.4.3. Interfaz

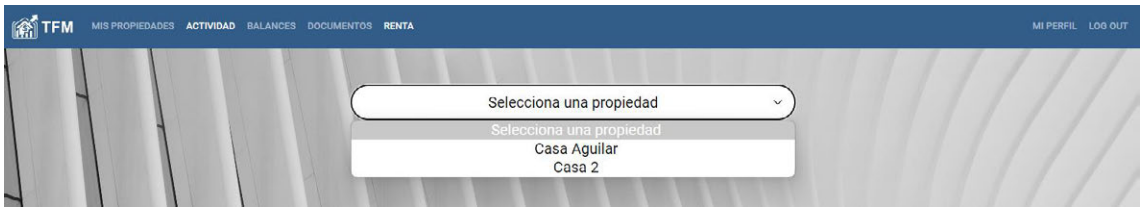
Editar propiedad:



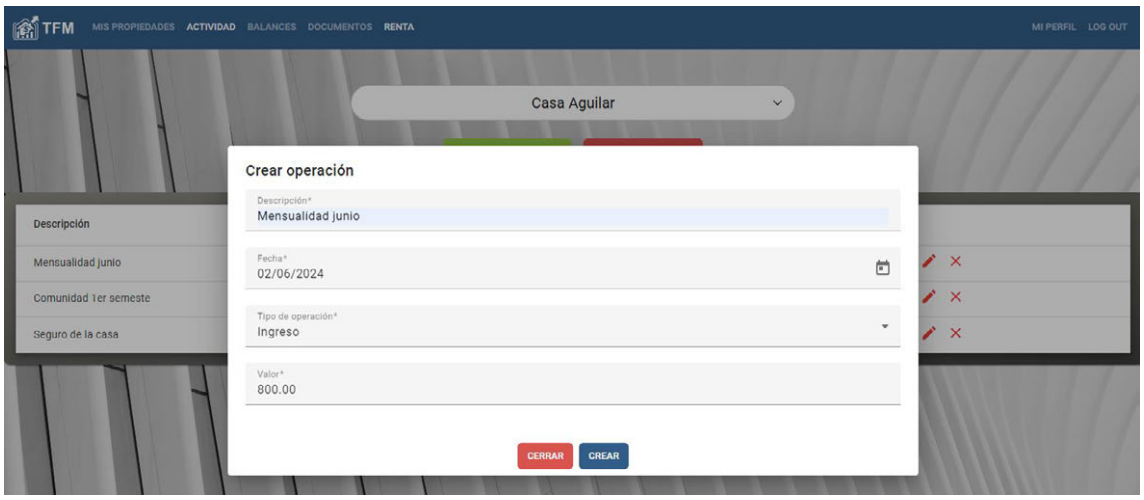
Mapa. Se ajusta automáticamente, para que siempre se vea todas las propiedades, en este caso la mitad norte de la península. Si se pulsa en los marcadores del mapa, se muestran los detalles de la propiedad (véase 4.3.3)



Actividad:



Crear operación:



4.4.4. *Endpoints*

Para la realización de las historias de usuario de este *sprint* se han añadido los siguientes *endpoints*. Se recuerda que para el mapa se utiliza en *endpoint* de listar propiedades (véase 4.3.4).

EDITAR PROPIEDAD			
Endpoint			
Método	**Ruta**	**Descripción**	
PUT	/properties/{property_id}	Se edita la propiedad indicada	
Parámetros			
Nombre	**Tipo**	**Ubi.**	**Descripción**
property_id	String	Path	ID de la propiedad a editar
propertyName*	String	Body	Nuevo nombre

address*	String	Body	Nueva dirección
cadastralReference*	String	Body	Nueva referencia catastral
houseDetails	Object `"houseDetails": { "numberOfRooms": 2, "hasGarden": false }`	Body	Nuevos detalles de la propiedad si es una casa. Contiene: numberOfRooms*: Integer y hasGarden*: Boolean.
flatDetails	Object `"flatDetails": { "numberOfRooms": 2, "floorNumber": 1, "hasBalcony": true }`	Body	Nuevos detalles de la propiedad si es un piso. Contiene: numberOfRooms*: Integer y floorNumber*: Integer y hasBalcony*: String.
garageDetails	Object `"garageDetails": { "capacity": "1", "isPrivate": "true" }`	Body	Nuevos detalles de la propiedad si es un garaje. Contiene: capacity*: Integer y isPrivate*: Boolean.

Respuestas		
Código	**Mensaje**	**Descripción**
200	OK	Propiedad editada correctamente
400	BAD REQUEST	Algún parámetro del cuerpo es incorrecto
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido
403	FORBIDDEN	Acceso denegado, la propiedad solicitada no pertenece al usuario autenticado
404	NOT FOUND	No existe una propiedad con ese ID

LISTAR OPERACIONES		
Endpoint		
Método	**Ruta**	**Descripción**
GET	/operations/property/{property_id}	Lista las operaciones de una propiedad

Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
property_id	String	Path	La propiedad de la cual listar sus operaciones

Respuestas		
Código	**Mensaje**	**Descripción**
200	OK	Devuelve la lista de operaciones de la propiedad
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido
403	FORBIDDEN	Acceso denegado, la propiedad sobre la que listar operaciones no pertenece al usuario autenticado
404	NOT FOUND	No existe una propiedad con ese ID

LEER OPERACIÓN			
Endpoint			
Método	**Ruta**		**Descripción**
GET	/operations/{operation_id}		Lee la información de la operación seleccionada
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
oepration_id*	String	Path	ID de la operación a leer
Respuestas			
Código	**Mensaje**		**Descripción**
200	OK		Devuelve la información de la operación
401	UNAUTHORIZED		Token no proporcionado, expirado o inválido
403	FORBIDDEN		Acceso denegado, la propiedad asociada a la operación no pertenece al usuario autenticado
404	NOT FOUND		No existe una operación con ese ID

CREAR OPERACIÓN			
Endpoint			
Método	**Ruta**	**Descripción**	
POST	/operations	Crea una operación para una propiedad dada en el cuerpo	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
property_id*	String	Body	ID de la propiedad sobre la que crear la operación
description*	String	Body	Descripción de la operación a crear
date*	String (Date)	Body	Fecha de la operación (aaaa-mm-dd)
type*	‘income’ | ‘expense’	Body	Tipo de operación (ingreso o gasto)
value*	Decimal	Body	Cantidad de dinero de la operación
Respuestas			
Código	**Mensaje**		**Descripción**
201	OK		Devuelve la operación creada
400	BAD REQUEST		Algún parámetro del cuerpo es incorrecto
401	UNAUTHORIZED		Token no proporcionado, expirado o inválido
403	FORBIDDEN		Acceso denegado, la propiedad sobre la que crear la operación no pertenece al usuario

4.4.5. Pruebas unitarias y de integración

```
PASS test/integration/user.spec.js
  Users integration test
    √ Read user KO - User not found (54 ms)
    √ Create user OK (195 ms)
    √ Create user KO - Username already exists (18 ms)
    √ Read user OK (17 ms)
    √ Create user KO - Bad request (137 ms)
    √ Create user KO - Bad request no password (16 ms)
    √ Login user OK (141 ms)
    √ Login user KO - No password (12 ms)
    √ Login user KO - User not found (16 ms)
    √ Login user KO - Incorrect password (145 ms)
    √ Update user OK (35 ms)
    √ Update user KO - User not found (15 ms)
    √ Delete user OK (18 ms)
    √ Delete user KO - User not found (16 ms)

PASS test/integration/property.spec.js
  Property integration test
    √ Read properties of user OK - No properties (25 ms)
    √ Create property OK (39 ms)
    √ Create property KO - Bad request (14 ms)
    √ Read properties of user OK (16 ms)
    √ Read property OK (26 ms)
    √ Read property KO - Property not found (16 ms)
    √ Update property OK (35 ms)
    √ Delete property OK (20 ms)
    √ Delete property KO - Property not found (16 ms)

PASS test/unit/controllers/operation.controller.spec.js
  Operation Controller
    √ Read operations of property OK (34 ms)
    √ Read operations of property KO - Property not found (15 ms)
    √ Read operations of property KO - Forbidden (15 ms)
    √ Read operation OK (15 ms)
    √ Read operation KO - Operation not found (14 ms)
    √ Read operation KO - Forbidden (20 ms)
    √ Create operation OK (23 ms)
    √ Create operation KO - Validation error (16 ms)
    √ Create operation KO - Property not found (14 ms)
    √ Create operation KO - Forbidden (18 ms)

PASS test/unit/controllers/user.controller.spec.js
  User Controller
    √ Read user OK (32 ms)
    √ Read user KO - User not found (14 ms)
    √ Create user OK (24 ms)
    √ Create user KO - Username already exists (14 ms)
    √ Create user KO - No password (13 ms)
    √ Create user KO - Bad Request no password (14 ms)
    √ Create user KO - Bad Request wrong email format (14 ms)
    √ Login user OK (15 ms)
    √ Login user KO - No username (14 ms)
    √ Login user KO - User not found (14 ms)
    √ Login user KO - Incorrect password (14 ms)
    √ Update user OK (15 ms)
    √ Update user KO - User not found (12 ms)
    √ Delete user OK (13 ms)
    √ Delete user KO - User not found (14 ms)

PASS test/unit/controllers/property.controller.spec.js
  Property Controller
    √ Read properties of user OK (35 ms)
    √ Read properties of user KO (18 ms)
    √ Read property OK (17 ms)
    √ Read property KO - Property not found (12 ms)
    √ Read property KO - Forbidden (18 ms)
    √ Create property OK (29 ms)
    √ Create property KO - Bad request (18 ms)
    √ Edit property OK (20 ms)
    √ Edit property KO - Property not found (23 ms)
    √ Edit property KO - Forbidden (19 ms)
    √ Delete property KO - Forbidden (21 ms)
    √ Delete property OK (20 ms)
    √ Delete property KO - Property not found (12 ms)

PASS test/integration/operation.spec.js
  Operation integration test
    √ Read operations of property OK - No operations (28 ms)
    √ Create operation OK (31 ms)
    √ Create operation KO - Property not found (18 ms)
    √ Create operation KO - Validation Error (16 ms)
    √ Read operations of property OK - One operation (20 ms)
    √ Read operation OK (22 ms)
    √ Read operations KO - Property not found (17 ms)
    √ Read operation KO - Not Found (16 ms)

PASS test/unit/services/user.service.spec.js
  User Service
    √ readUser OK (4 ms)
    √ readUser OK - But user not found (1 ms)
    √ createUser OK (1 ms)
    √ isCorrectPassword OK (1 ms)
    √ generateToken OK (3 ms)
    √ updateUser OK (1 ms)
    √ deleteUser OK (1 ms)
    √ deleteUser KO - User not found (1 ms)

PASS test/unit/models/property.model.spec.js
  Property Model
    √ Find all properties (10 ms)
    √ Find Property by property_id (2 ms)
    √ Create property (4 ms)
    √ Create property missing parameter (2 ms)
    √ Create property with missing required field (2 ms)
    √ Update property (1 ms)
    √ Delete property (1 ms)

PASS test/unit/models/user.model.spec.js
  User Model
    √ Find User by username (9 ms)
    √ Create user (3 ms)
    √ Create user KO - missing required field (1 ms)
    √ Update user (1 ms)
    √ Delete user (2 ms)

PASS test/unit/middlewares/auth.spec.js
  Auth Middleware
    √ Auth KO - Token not provided (32 ms)
    √ Auth KO - Token is invalid (14 ms)
    √ Auth KO - Token has expired (18 ms)
    √ Auth OK (15 ms)

PASS test/unit/services/operation.service.spec.js
  Property Service
    √ readOperationsByPropertyId (4 ms)
    √ readOperationsByPropertyIdAndDateRange (2 ms)
    √ readOperation (1 ms)
    √ createOperation (2 ms)

PASS test/unit/services/property.service.spec.js
  Property Service
    √ Return properties by user id (5 ms)
    √ Read property by id (2 ms)
    √ Create property House (2 ms)
    √ Create property Flat (2 ms)
    √ Create property Garage (2 ms)
    √ Update property House (1 ms)
    √ Update property Flat (1 ms)
    √ Update property Garage (1 ms)
    √ Delete property by id (2 ms)
    √ Delete property by id - Property not found (2 ms)
    √ Validate property ownership (2 ms)
    √ Validate property ownership - Property not found (30 ms)
    √ Validate property ownership - User is not the owner (2 ms)

PASS test/unit/models/operation.model.spec.js
  Operation Model
    √ Find all operations by property id (6 ms)
    √ Find Operation by operation_id (1 ms)
    √ Create operation (3 ms)
    √ Create operation with missing required field (1 ms)

PASS test/unit/middlewares/userValidation.spec.js
  User Validation middleware
    √ Requested username is the same as the authenticated username (3 ms)
    √ Requested username is not the same as the authenticated username (8 ms)

Test Suites: 14 passed, 14 total
Tests:       116 passed, 116 total
Snapshots:   0 total
Time:        12.458 s
```

4.4.5.1. Cobertura de código

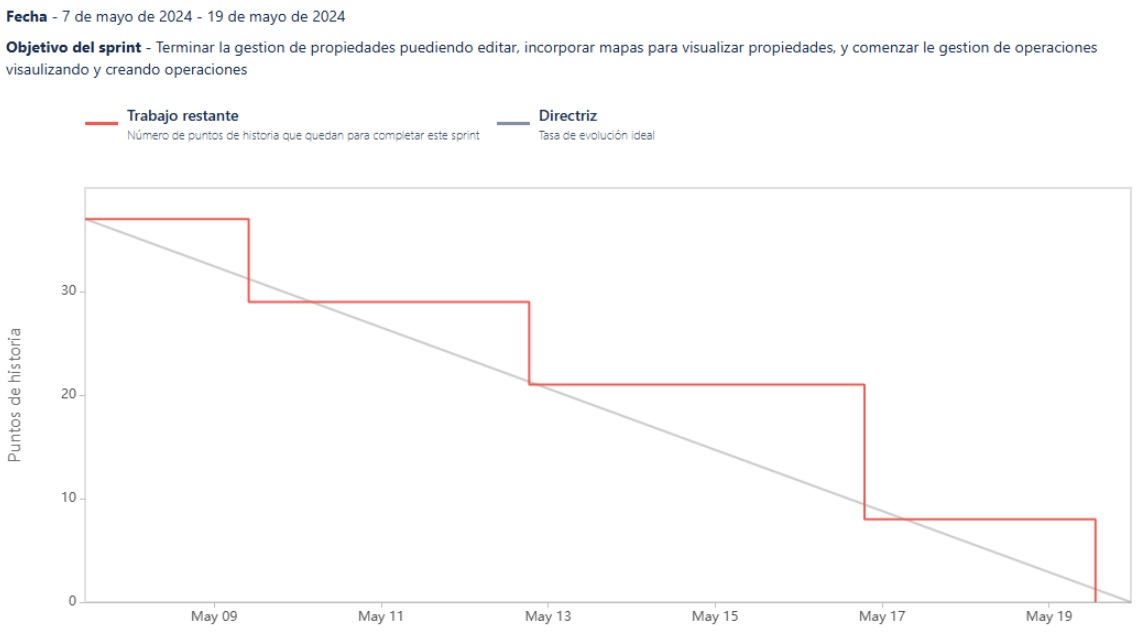
All files

97.92% Statements 378/386 **84.31%** Branches 86/102 **92.85%** Functions 39/42 **97.91%** Lines 375/383

File	Statements		Branches		Functions		Lines	
src	93.75%	15/16	100%	0/0	100%	1/1	93.75%	15/16
src/config	90.9%	10/11	62.5%	10/16	100%	0/0	90.9%	10/11
src/controllers	98.68%	150/152	100%	44/44	100%	13/13	98.68%	150/152
src/errors	100%	16/16	100%	0/0	100%	4/4	100%	16/16
src/middlewares	97.05%	33/34	90%	9/10	100%	3/3	97.05%	33/34
src/models	93.33%	42/45	25%	3/12	50%	3/6	93.33%	42/45
src/routes	100%	39/39	100%	0/0	100%	0/0	100%	39/39
src/services	100%	73/73	100%	20/20	100%	15/15	100%	70/70

4.4.6. *Burn-down chart*

En la siguiente imagen se puede observar el gráfico de pendiente que corresponde a las dos semanas de duración del *sprint* 3.



En el gráfico se puede observar como el *sprint* comienza con 32 puntos de historia, de los cuales se entregan todos al final del *sprint*. Se puede observar como la entrega durante estas dos semanas ha sido la que más se acerca a la pendiente ideal, habiendo una entrega de historias periódica y constante.

4.4.7. *Sprint review*

En la revisión se repasa las historias de usuario que se planificaron para este *sprint*, si se han entregado al final del mismo, y si se acepta esta entrega de acuerdo con los criterios de aceptación definidos en un principio.

Historia	Criterios de aceptación	Entregada/ Aceptada
TFM-21 Editar propiedad	**Dado** un usuario en "Mis propiedades" **Cuando** pulsa en editar una de sus propiedades en el listado **Entonces** puede editar la información de dicha propiedad	Sí / Sí
TFM-19 Incorporación de mapa con la disposición de las propiedades	**Dado** un usuario autenticado **Cuando** está en "mis propiedades" **Entonces** se muestra un mapa donde contiene marcadores con sus propiedades	Sí / Sí
	Dado un usuario en "mis propiedades" **Cuando** pulsa uno de los marcadores del mapa **Entonces** se abre los detalles de esa propiedad	
TFM-25 Abrir "actividad" (operaciones)	**Dado** un usuario autenticado **Cuando** accede a actividad **Entonces** puede seleccionar entre una de sus propiedades para consultar la actividad de esta	Sí / Sí
	Dado un usuario en "Actividad" y habiendo seleccionado entre una de sus propiedades **Cuando** pulsa en añadir gasto/ingreso **Entonces** puede rellenar el formulario para crear una nueva operación	
TFM-26 Crear operación	**Dado** un usuario autenticado **Cuando** accede a actividad **Entonces** puede seleccionar entre una de sus propiedades para consultar la actividad de esta	Sí / Sí

Las historias de usuario relacionadas con la gestión de propiedades u operaciones se realizan en un tiempo cercano a la estimación, y no se presentan complicaciones extra al estar familiarizado con este tipo de tareas durante el proyecto.

Por su parte, la incorporación de Google Maps es la funcionalidad que más incertidumbre presentaba para este *sprint*. Aprendiendo de la retrospectiva del *sprint* 2, a la hora de estimar se tuvo en cuenta la necesidad de formación y búsqueda de información para llevarla a cabo. Finalmente, con todo lo necesario para acometerla, se desarrolla cumpliendo los criterios de aceptación definidos y en el tiempo estimado.

4.4.8. Retrospectiva

Para finalizar con el *sprint* 3, en la retrospectiva se analiza que ha sucedido durante el sprint para tomar acciones en caso de que fuese necesario.

	¿Qué ha pasado?	Acciones
¿Qué fue bien?	Buena estimación de las historias, tanto en las que se tenía más experiencia por similitud con otras, como en las nuevas de Google Maps.	
	Historias con tareas y criterios de aceptación claros que facilita su desarrollo	
¿Qué se puede mejorar?	Múltiples objetivos durante el *sprint*	Por motivos de fechas en este *sprint* coinciden tareas muy distintas entre gestión de propiedades, incorporación de mapa, manejo de operaciones. Esto hace que durante el desarrollo se pierda un poco el foco principal del *sprint*, el cual estaba más claro en los anteriores.
¿Qué fue mal?	-	-

4.5. *Sprint* 4

El *sprint* 4, y penúltimo antes de la entrega del proyecto fin de máster, abarca la finalización de las historias de gestión de operaciones (épica TFM-29), la épica TFM-30 Balances y el comienzo de la gestión de documentos (épica TFM-33) con la subida de estos.

4.5.1. *Sprint backlog*

Para completar los objetivos del *sprint* 4 la planificación ha dado como resultado la siguiente pila de trabajo.

TFM-27 Editar operación					
Descripción	**Como** usuario en "actividad" **Quiero** poder editar operaciones **Para** actualizar información o realizar ajustes necesarios en las transacciones registradas				
Prioridad	Media	**Puntos de historia**	5	**Tiempo consumido**	5h
Criterios de aceptación	**Dado** un usuario en "actividad" y habiendo seleccionado una propiedad **Cuando** pulsa editar en una de las operaciones del listado **Entonces** puede actualizar la información de la operación				
Tareas					
Nombre de la tarea					**Tipo**
Edición de operación					Back-End
Interfaz para la edición de operación reutilizando el dialogo de creación, y conexión con el Back-End					Front-End

TFM-28 Eliminar operación					
Descripción	**Como** usuario en "actividad" **Quiero** poder eliminar operaciones **Para** para eliminar registros incorrectos o duplicados y mantener la integridad de la información financiera de la propiedad				
Prioridad	Alta	**Puntos de historia**	3	**Tiempo consumido**	2h
Criterios de aceptación	**Dado** un usuario en "actividad" y habiendo seleccionado una propiedad **Cuando** pulsa borrar en una de las operaciones del listado y confirma el aviso **Entonces** se borra la operación				
Tareas					
Nombre de la tarea					**Tipo**
Endpoint para borrar una operación					Back-End
Funcionalidad de borrado de operación con dialogo de confirmación para el frontal					Front-End

TFM-31 Recogida y filtrado de datos para realizar balances					
Descripción	**Como** usuario **Quiero** poder seleccionar una de mis propiedades y un rango de fechas **Para** poder hacer cálculos con las operaciones filtradas				
Prioridad	Media	**Puntos de historia**	8	**Tiempo consumido**	10h
Criterios de aceptación	**Dado** un usuario en "Balances" **Cuando** selecciono una propiedad, un rango de fechas predefinido o personalizado y un intervalo de tiempo, y genero el balance **Entonces** se realiza un filtrado de operaciones y las operaciones necesarias para generar un balance				

Tareas	
Nombre de la tarea	**Tipo**
Recogida de datos del usuario para el cálculo del balance	Front-End
Filtro de las operaciones según los datos introducidos y operaciones para generar el balance	Back-End

TFM-32 Generación de gráficos					
Descripción	**Como** usuario **Quiero** visualizar en gráficos las finanzas de mis propiedades **Para** estudiar su rentabilidad				
Prioridad	Media	**Puntos de historia**	8	**Tiempo consumido**	6h
Criterios de aceptación	**Dado** un usuario en "Balances" **Cuando** ha generado un balance **Entonces** se visualiza en un gráfico de barras con las rentabilidades en el rango de fechas y con intervalo seleccionado				

Tareas	
Nombre de la tarea	**Tipo**
Incorporación de librería chart.js, y componente para realizar gráficos de barras	Front-End
Componente "balances" y definición de contrato con Back-End	Front-End
Adaptar el balance calculado al contrato definido, para representar balances en gráficos de barras	Back-End

TFM-34 Subida de documentos					
Descripción	**Como** usuario **Quiero** subir ficheros a la aplicación **Para** tener recogidos los distintos documentos en un solo lugar				
Prioridad	Alta	**Puntos de historia**	3	**Tiempo consumido**	3h

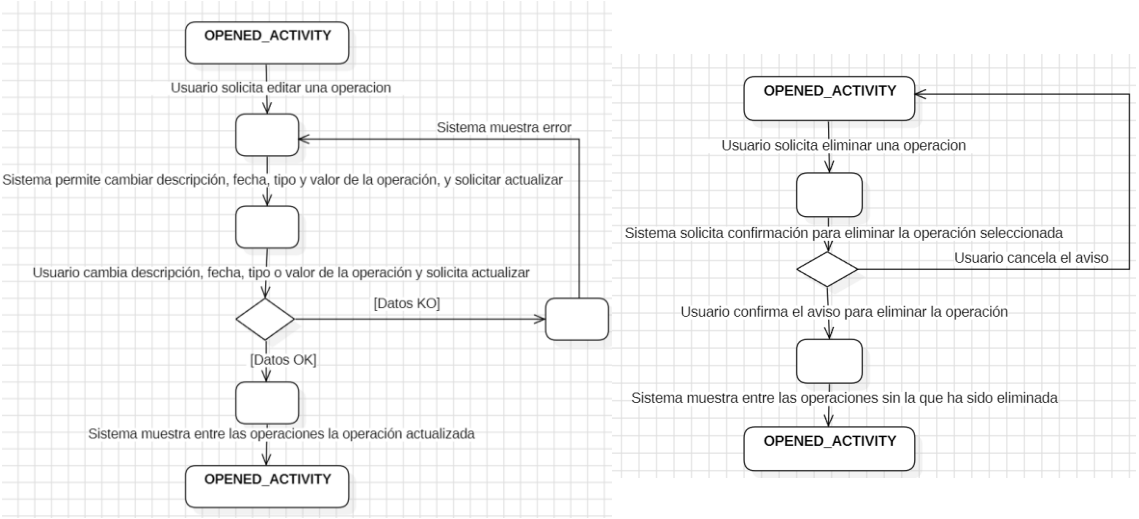
Criterios de aceptación	**Dado** un usuario en “Documentos” **Cuando** selecciono y subo un fichero con formato pdf, word, jpeg o png **Entonces** se almacena el documento en la aplicación

Tareas	
Nombre de la tarea	**Tipo**
Controlador, ruta, servicio, modelo para la subida de documentos	Back-End
Creación componente “documentos” e interfaz para seleccionar y subir un documento	Front-End

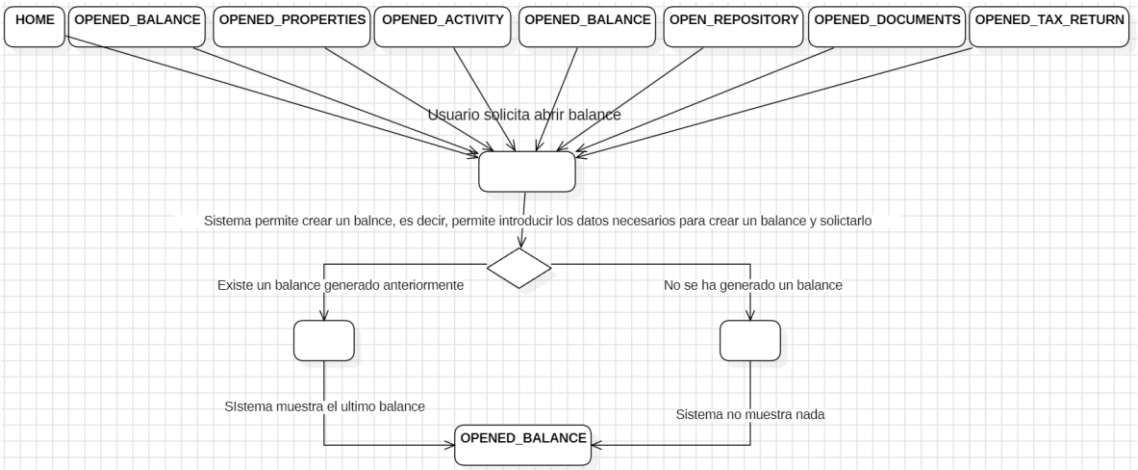
4.5.2. Diseño de la arquitectura

4.5.2.1. Diagramas de casos de uso

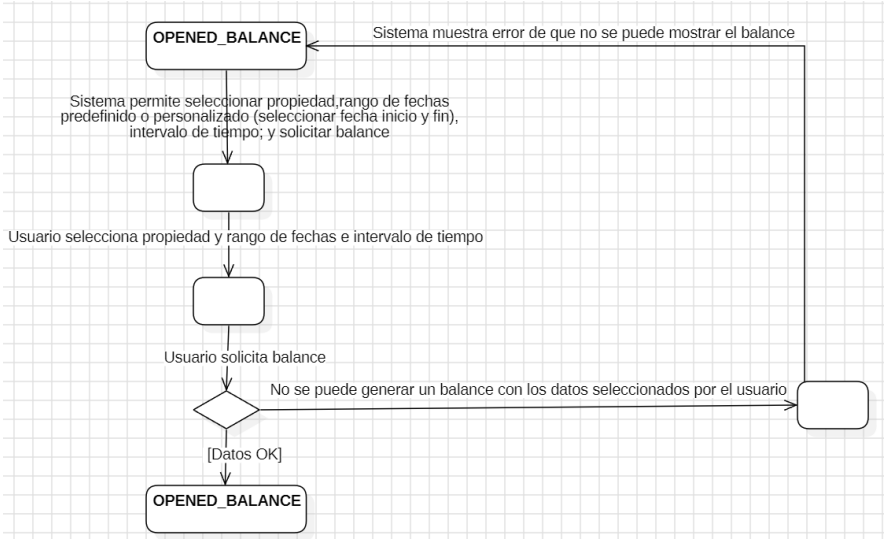
Editar operación (UpdateOperation) y eliminar operación (DeleteOperation):



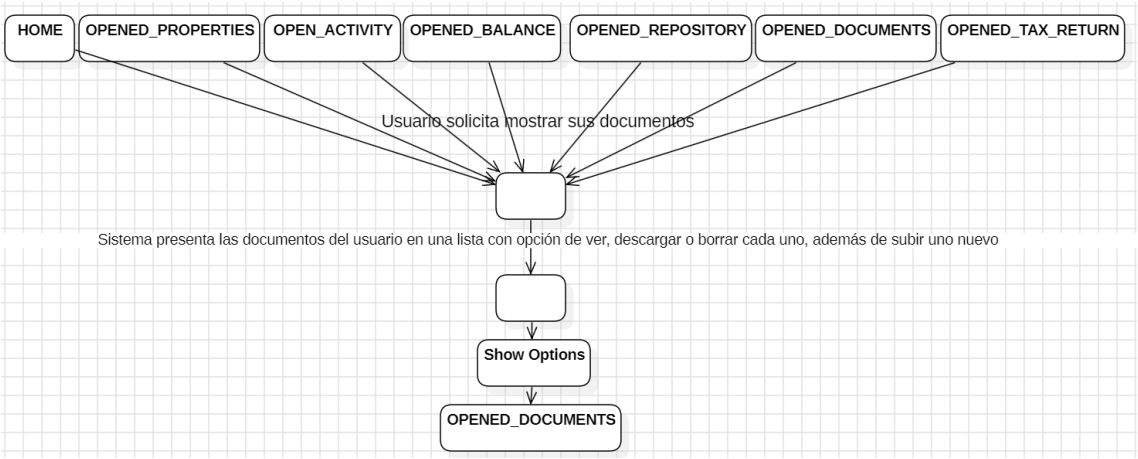
Abrir balance (OpenBalance):



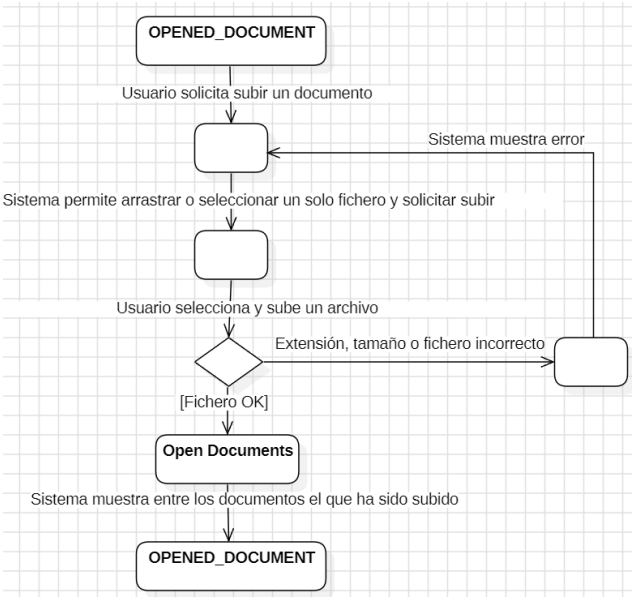
Crear balance (CreateBalance):



Abrir documentos (OpenDocuments):

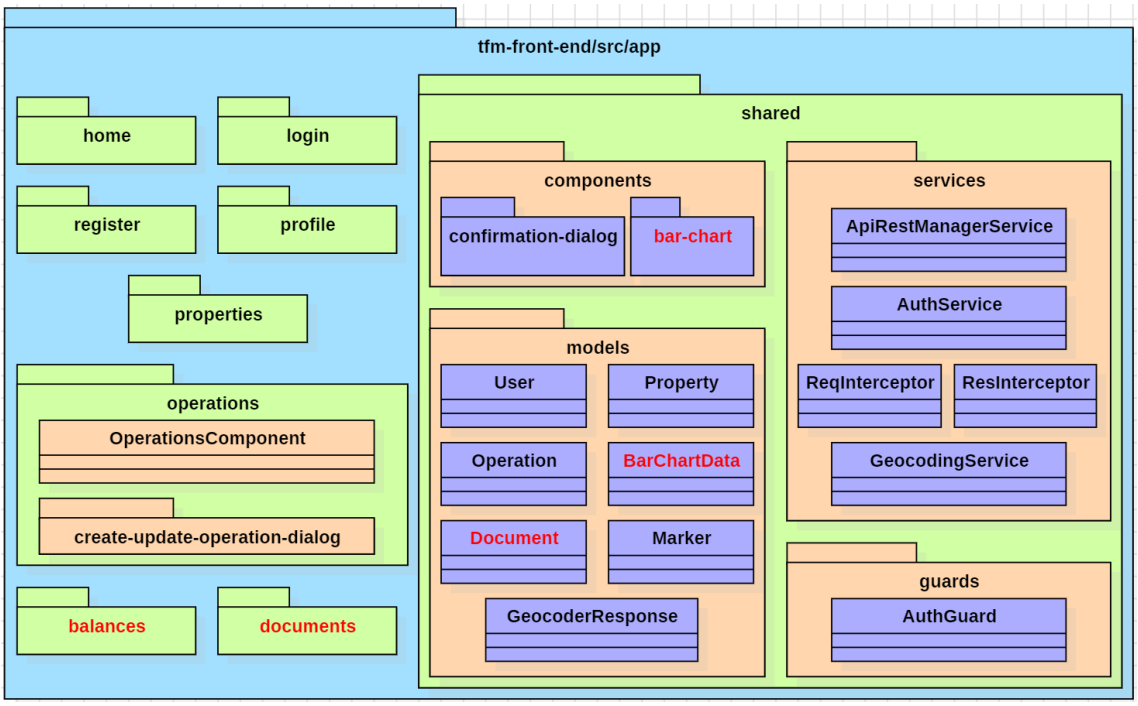


Subir documento (UploadDocument):



4.5.2.2.a. Diagrama de paquetes (Front)

Con texto rojo se presenta aquello añadido durante este *sprint*. Por su parte, para finalizar la gestión de operaciones, tanto con la edición como borrado de estas, se sigue trabajando sobre los componentes ya creados en el anterior *sprint* para crear y listar operaciones.

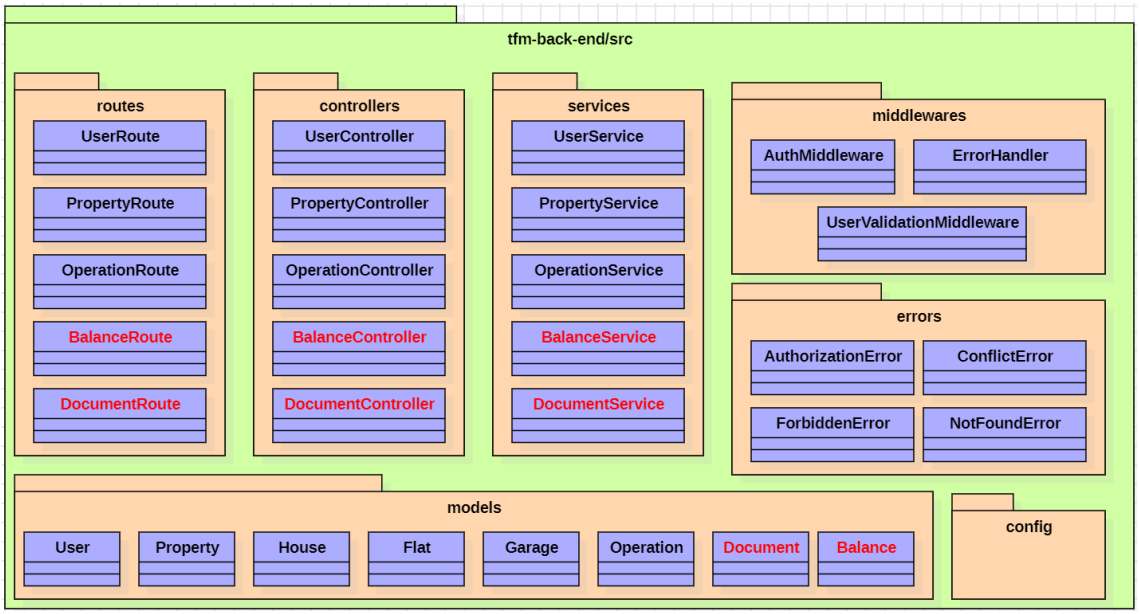


Para la funcionalidad de **balances**, accesible desde la cabecera, se agrupa en la carpeta con ese nombre. Por tanto, la recogida de datos por parte del usuario (rango de fechas, intervalo de tiempo, etc.) y la presentación del gráfico generado se realiza en **BalanceComponent**.

Para la generación de un gráfico de barras, en la carpeta **shared,** se crea el componente **BarChartComponent**, que recibirá como entrada los datos necesarios para crear el gráfico de barras (representado por el nuevo modelo **BarCharData**), y además una variable "*booleana"* para mostrar, o no, un botón bajo el gráfico para exportarlo como imagen.

La interfaz para la subida de documentos, y las posteriores historias de usuario relacionadas con la gestión de documentos, se recoge dentro de la carpeta **documents**, como también la utilización del modelo **Document**.

4.5.2.2.b. Diagrama de paquetes (Back)

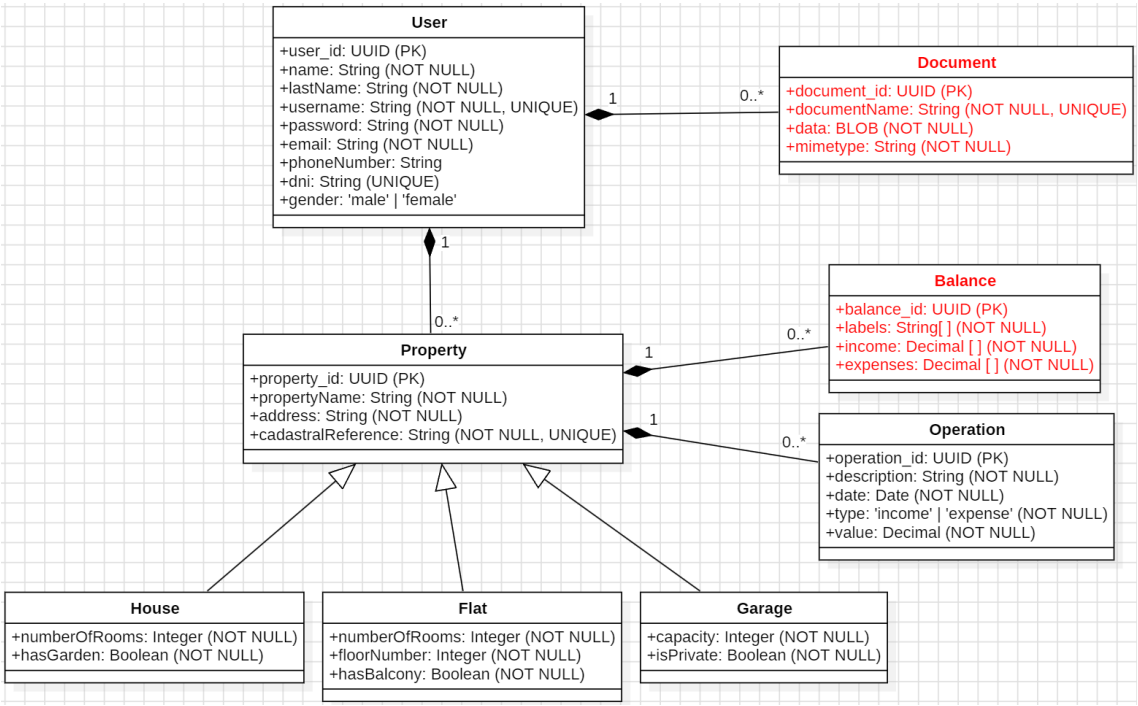


Para completar la edición y borrado de operaciones se añaden las rutas correspondientes en **OperationRoute** y que se verán en detalle en la sección 4.5.2.4. *Endpoints*, así como los controladores y servicios necesarios.

Para manejar balances y documentos, como siempre, se crean sus modelos **Balance** y **Document**, respectivamente. Además, será necesario responder a las peticiones HTTP mediante **BalanceRoute** y **DocumentRoute**; procesar estas peticiones con **BalanceController** y **DocumentController**; y realizar la lógica de negocio desde **BalanceService** y **DocumentService**.

Como para el cálculo de balances es necesario las operaciones desde BalanceService es indispensable el uso de los servicios de operaciones (OperationService).

4.5.2.3. Modelo de datos

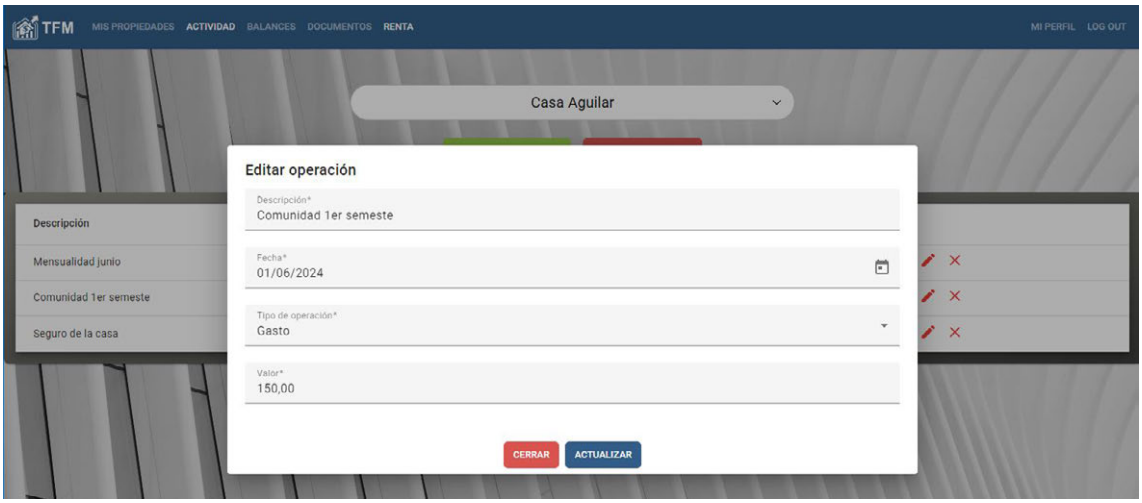


Se decide guardar los balances a la hora de crearlos, pese a que no existe ninguna funcionalidad que se prevea acceder a un histórico de balances, simplemente a modo de registro de los balances generados por cada usuario. Al igual que sucedía con el modelo Operación, el usuario puede crear *n* balances para una propiedad, perteneciendo cada balance a una sola propiedad, aquella que seleccione el usuario junto al resto de datos necesarios para generar el balance.

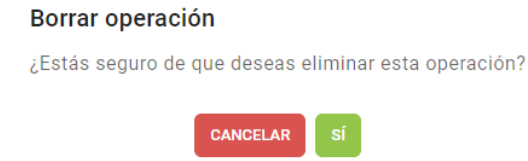
Para almacenar los documentos que sube el usuario a la aplicación se crea el modelo Documento. En el atributo "data", de tipo BLOB (acrónimo de *Binay Large OBject*), se almacena grandes cantidades de datos binarios, es decir, los documentos. La multiplicidad es 1 a *n*, es decir, un usuario puede tener *n* documentos, mientras que un documento solo puede pertenecer a un usuario.

4.5.3. Interfaz

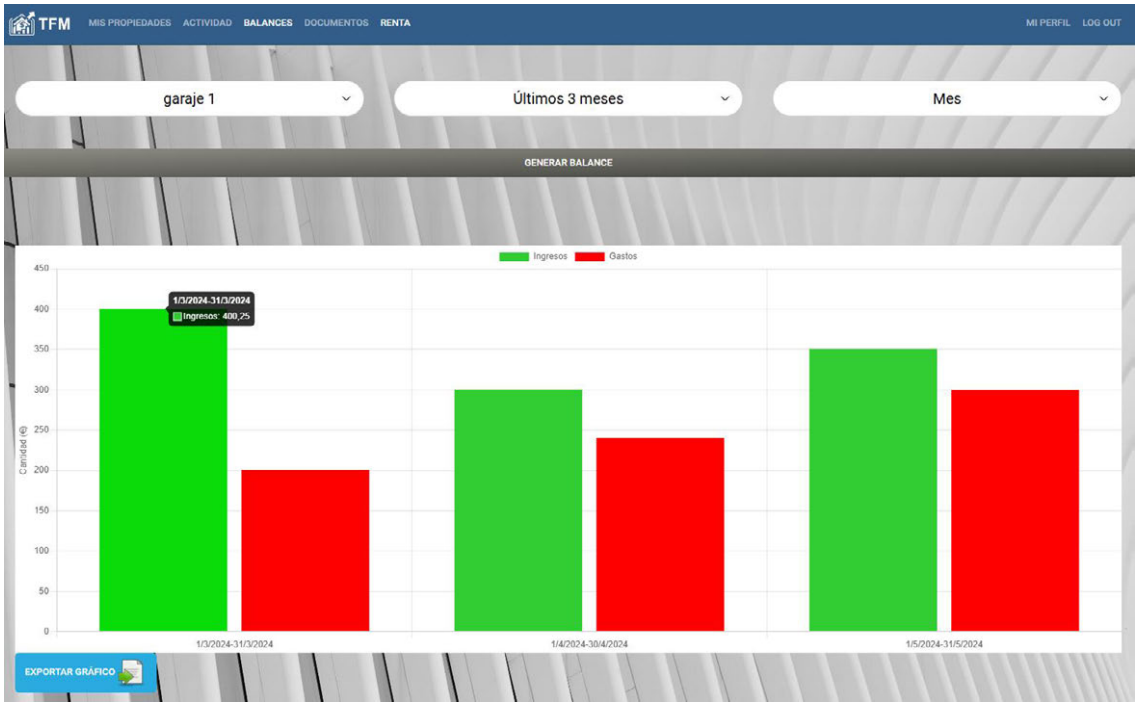
Editar operación:

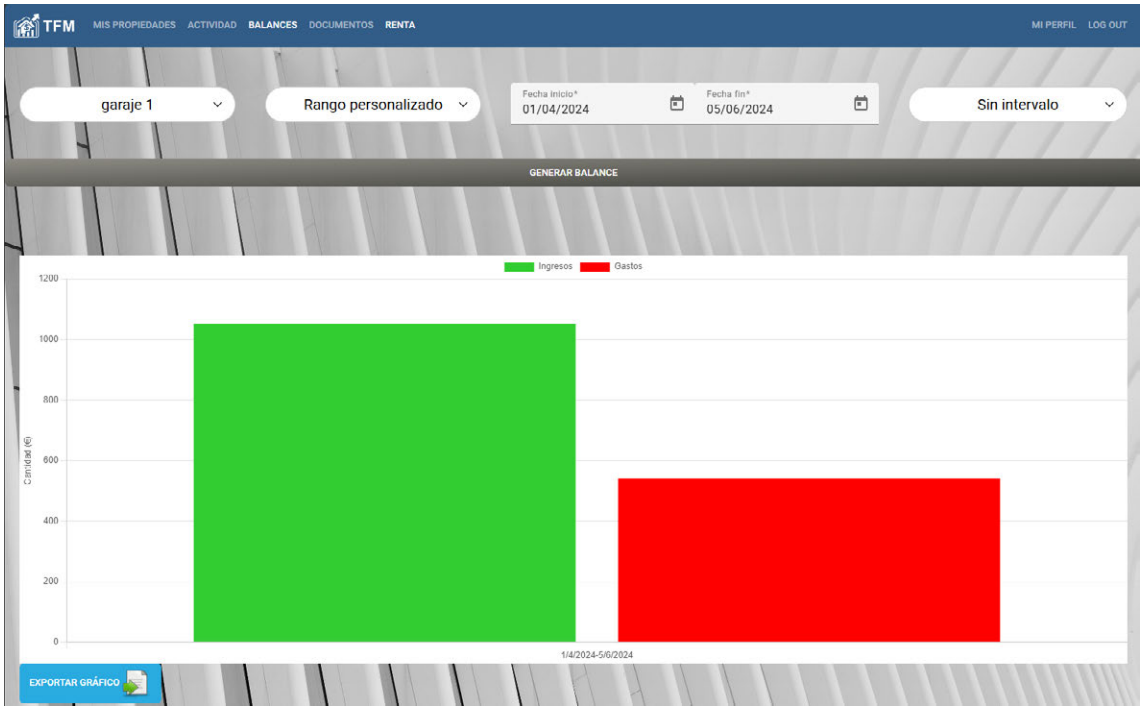


Borrar operación:

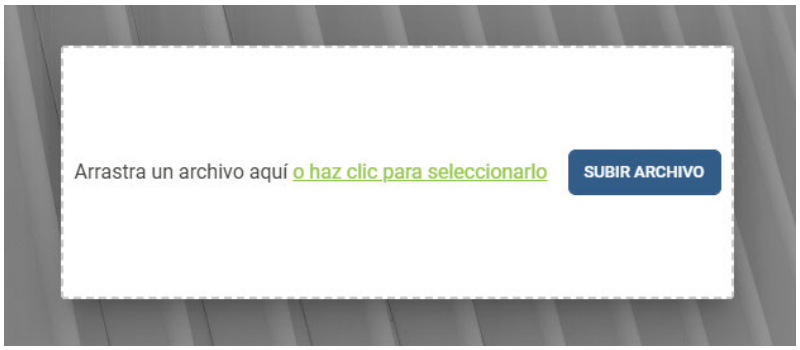


Balances:





Subir documento:



4.5.4. *Endpoints*

Para este *sprint* se tienen *endpoints* relacionados con la gestión de operaciones:

EDITAR OPERACIÓN			
Endpoint			
Método	**Ruta**		**Descripción**
PUT	/operations/{operation_id}		Editar la operación seleccionada
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
operation_id*	String	Path	ID de la operación a editar
description*	String	Body	Nueva descripción de la operación
date*	String (Date)	Body	Nueva fecha de la operación (aaaa-mm-dd)
type*	'income' | 'expense'	Body	Nuevo tipo de operación (ingreso o gasto)
value*	Decimal	Body	Nueva cantidad de dinero de la operación
Respuestas			
Código	**Mensaje**		**Descripción**
200	OK		Devuelve la operación editada
400	BAD REQUEST		Algún parámetro del cuerpo es incorrecto
401	UNAUTHORIZED		Token no proporcionado, expirado o inválido
403	FORBIDDEN		Acceso denegado, la propiedad asociada a la operación no pertenece al usuario
404	NOT FOUND		La operación no se ha encontrado

BORRAR OPERACIÓN			
Endpoint			
Método	**Ruta**		**Descripción**
DELETE	/operations/{operation_id}		Borrar la operación seleccionada
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
operation_id*	String	Path	ID de la operación a borrar
Respuestas			
Código	**Mensaje**		**Descripción**
200	OK		Operación borrada, devuelve mensaje de

		confirmación
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido
403	FORBIDDEN	Acceso denegado, la propiedad asociada a la operación no pertenece al usuario
404	NOT FOUND	No existe una propiedad con ese ID

Para la creación de balances se tiene el siguiente *endpoint*:

CREAR BALANCE			
Endpoint			
Método	**Ruta**	**Descripción**	
POST	/balances	Genera los datos necesarios para crear un balance y representarlo en un gráfico desde el frontal	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
property_id*	String	Body	ID de la propiedad seleccionada para crear el balance
dateRange*	Object `"dateRange": { "startDate": "2024-04-01", "endDate": "2024-04-30" }`	Body	Rango de fechas del balance, compuesto por una fecha de inicio (startDate*: String) y una fecha fin (endDate*: String)
timeInterval*	Integer	Body	Intervalo de tiempo en meses. 0: sin intervalo, 1: mensual, 3: trimestral, 12: anual, etc.
Respuestas			
Código	**Mensaje**	**Descripción**	
201	OK	Devuelve el balance creado	
400	BAD REQUEST	Algún parámetro del cuerpo es incorrecto	
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido	
403	FORBIDDEN	Acceso denegado, la propiedad seleccionada para crear el balance no pertenece al usuario	
404	NOT FOUND	La propiedad seleccionada para crear el balance no existe	

Y, por último, para subir documentos se ha creado el siguiente *endpoint*, que será el primero de los *endpoints* para la gestión de documentos.

SUBIR DOCUMENTO			
Endpoint			
Método	**Ruta**	**Descripción**	
POST	/documents	Sube el documento y se guarda en base de datos	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
Content-Type*	String	Header	Cabecera con el valor “multipart/form-data”
file*	File	Body (form-data)	Fichero adjunto en el cuerpo con formato FormData, con la clave “file”

Respuestas		
Código	**Mensaje**	**Descripción**
201	OK	Devuelve el balance creado
400	BAD REQUEST	Algún parámetro del cuerpo es incorrecto
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido
403	FORBIDDEN	Acceso denegado, la propiedad seleccionada para crear el balance no pertenece al usuario
404	NOT FOUND	La propiedad seleccionada para crear el balance no existe
409	CONFLICT	Ya existe un documento con ese nombre de usuario

4.5.5. Pruebas unitarias y de integración

```
PASS test/integration/user.spec.js
  Users integration test
    √ Read user KO - User not found (76 ms)
    √ Create user OK (228 ms)
    √ Create user KO - Username already exists (19 ms)
    √ Read user OK (23 ms)
    √ Create user KO - Bad request (174 ms)
    √ Create user KO - Bad request no password (15 ms)
    √ Login user OK (150 ms)
    √ Login user KO - No password (19 ms)
    √ Login user KO - User not found (28 ms)
    √ Login user KO - Incorrect password (137 ms)
    √ Update user OK (23 ms)
    √ Read user KO - User not found (76 ms)
    √ Create user OK (228 ms)
    √ Create user KO - Username already exists (19 ms)
    √ Read user OK (23 ms)
    √ Create user KO - Bad request (174 ms)
    √ Create user KO - Bad request no password (15 ms)
    √ Login user OK (150 ms)
    √ Login user KO - No password (19 ms)
    √ Login user KO - User not found (28 ms)
    √ Login user KO - Incorrect password (137 ms)
    √ Update user OK (23 ms)
    √ Update user KO - User not found (17 ms)
    √ Delete user OK (18 ms)
    √ Delete user KO - User not found (16 ms)

PASS test/integration/operation.spec.js
  Operation integration test
    √ Read operations of property OK - No operations (29 ms)
    √ Create operation OK (32 ms)
    √ Create operation KO - Property not found (23 ms)
    √ Create operation KO - Validation Error (26 ms)
    √ Update operation OK (32 ms)
    √ Update operation KO - Operation not found (20 ms)
    √ Read operations of property OK - One operation (32 ms)
    √ Read operation OK (22 ms)
    √ Read operations KO - Property not found (18 ms)
    √ Read operation KO - Not Found (18 ms)
    √ Delete operation OK (26 ms)
    √ Delete operation KO - Not Found (16 ms)

PASS test/unit/controllers/operation.controller.spec.js
  Operation Controller
    √ Read operations of property OK (32 ms)
    √ Read operations of property KO - Property not found (15 ms)
    √ Read operations of property KO - Forbidden (16 ms)
    √ Read operation OK (44 ms)
    √ Read operation KO - Operation not found (47 ms)
    √ Read operation KO - Forbidden (37 ms)
    √ Create operation OK (48 ms)
    √ Create operation KO - Validation error (18 ms)
    √ Create operation KO - Property not found (19 ms)
    √ Create operation KO - Forbidden (17 ms)
    √ Update operation OK (15 ms)
    √ Update operation KO - Operation not found (15 ms)
    √ Update operation KO - Forbidden (16 ms)
    √ Delete operation OK (14 ms)
    √ Delete operation KO - Operation not found (17 ms)
    √ Delete operation KO - Forbidden (18 ms)

PASS test/integration/document.spec.js
  Document integration test
    √ Upload document OK (63 ms)
    √ Upload document KO - Unsupported file type (31 ms)
    √ Upload document KO - Missing file (18 ms)

PASS test/unit/controllers/user.controller.spec.js
  User Controller
    √ Read user OK (32 ms)
    √ Read user KO - User not found (13 ms)
    √ Create user OK (26 ms)
    √ Create user KO - Username already exists (16 ms)
    √ Create user KO - No password (15 ms)
    √ Create user KO - Bad Request no password (16 ms)
    √ Create user KO - Bad Request wrong email format (19 ms)
    √ Login user OK (16 ms)
    √ Login user KO - No username (15 ms)
    √ Login user KO - User not found (13 ms)
    √ Login user KO - Incorrect password (13 ms)
    √ Update user OK (15 ms)
    √ Update user KO - User not found (14 ms)
    √ Delete user OK (12 ms)
    √ Delete user KO - User not found (14 ms)

PASS test/unit/controllers/balance.controller.spec.js
  Balance Controller
    √ Create balance OK (37 ms)
    √ Create balance KO - Missing required fields (18 ms)
    √ Create balance KO - Invalid timeInterval (17 ms)
    √ Create balance KO - Forbidden (20 ms)
    √ Create balance KO - Property not found (23 ms)

PASS test/unit/controllers/property.controller.spec.js
  Property Controller
    √ Read properties of user OK (26 ms)
    √ Read properties of user KO (13 ms)
    √ Read property OK (14 ms)
    √ Read property KO - Property not found (13 ms)
    √ Read property KO - Forbidden (14 ms)
    √ Create property OK (24 ms)
    √ Create property KO - Bad request (14 ms)
    √ Edit property OK (15 ms)
    √ Edit property KO - Property not found (18 ms)
    √ Edit property KO - Forbidden (15 ms)
    √ Delete property KO - Forbidden (13 ms)
    √ Delete property OK (13 ms)
    √ Delete property KO - Property not found (12 ms)
```

```
PASS test/integration/property.spec.js
  Property integration test
    √ Read properties of user OK - No properties (21 ms)
    √ Create property OK (37 ms)
    √ Create property KO - Bad request (14 ms)
    √ Read properties of user OK (17 ms)
    √ Read property OK (23 ms)
    √ Read property KO - Property not found (18 ms)
    √ Update property OK (34 ms)
    √ Delete property OK (20 ms)
    √ Delete property KO - Property not found (16 ms)

PASS test/integration/balance.spec.js
  Balance integration test
    √ Create Balance OK (62 ms)
    √ Create Balance KO - Property not found (22 ms)
    √ Create Balance KO - Validation Error (16 ms)
    √ Create Balance KO - Validation Error timeInterval (17 ms)

PASS test/unit/controllers/document.controller.spec.js
  Document Controller
    √ Upload document OK (41 ms)
    √ Upload document KO - Unsupported file type (24 ms)
    √ Upload document KO - Missing file (14 ms)
    √ Upload document KO - File with existing document name (21 ms)

PASS test/unit/services/document.service.spec.js
  Document Service
    √ createDocument OK (12 ms)
    √ createDocument with missing fields (2 ms)
    √ createDocument with invalid data (4 ms)

PASS test/unit/services/balance.service.spec.js
  Balance Service
    √ calculateBalanceForInterval OK - Interval 0 (6 ms)
    √ calculateBalanceForInterval OK - With interval (3 ms)

PASS test/unit/models/balance.model.spec.js
  Balance Model
    √ Save balance (13 ms)

PASS test/unit/services/user.service.spec.js
  User Service
    √ readUser OK (4 ms)
    √ createUser OK (1 ms)
    √ createUser OK (1 ms)
    √ isCorrectPassword OK (1 ms)
    √ generateToken OK (2 ms)
    √ updateUser OK (1 ms)
    √ deleteUser OK (1 ms)
    √ deleteUser KO - User not found (1 ms)

PASS test/unit/models/property.model.spec.js
  Property Model
    √ Find all properties (7 ms)
    √ Find Property by property_id (2 ms)
    √ Create property (3 ms)
    √ Create property missing parameter (1 ms)
    √ Create property with missing required field (2 ms)
    √ Update property (1 ms)
    √ Delete property (2 ms)

PASS test/unit/models/document.model.spec.js
  Document Model
    √ Create document (9 ms)
    √ Create document missing parameter (2 ms)

PASS test/unit/models/user.model.spec.js
  User Model
    √ Find User by username (9 ms)
    √ Create user (2 ms)
    √ Create user KO - missing required field (4 ms)
    √ Update user (1 ms)
    √ Delete user (1 ms)

PASS test/unit/middlewares/auth.spec.js
  Auth Middleware
    √ Auth KO - Token not provided (32 ms)
    √ Auth KO - Token is invalid (16 ms)
    √ Auth KO - Token has expired (17 ms)
    √ Auth OK (17 ms)

PASS test/unit/services/operation.service.spec.js
  Property Service
    √ readOperationsByPropertyId (5 ms)
    √ readOperationsByPropertyIdAndDateRange (1 ms)
    √ readOperation (1 ms)
    √ createOperation (1 ms)
    √ updateOperation (1 ms)
    √ deleteOperation (2 ms)

PASS test/unit/services/property.service.spec.js
  Property Service
    √ Return properties by user id (5 ms)
    √ Read property by id (2 ms)
    √ Create property House (2 ms)
    √ Create property Flat (2 ms)
    √ Create property Garage (2 ms)
    √ Update property House (2 ms)
    √ Update property Flat (2 ms)
    √ Update property Garage (1 ms)
    √ Delete property by id (4 ms)
    √ Delete property by id - Property not found (1 ms)
    √ Validate property ownership (1 ms)
    √ Validate property ownership - Property not found (18 ms)
    √ Validate property ownership - User is not the owner (2 ms)

PASS test/unit/middlewares/userValidation.spec.js
  User Validation middleware
    √ Requested username is the same as the authenticated username (3 m
    √ Requested username is not the same as the authenticated username

Test Suites: 22 passed, 22 total
Tests:       154 passed, 154 total
Snapshots:   0 total
Time:        19.607 s
```

4.5.5.1. Cobertura de código

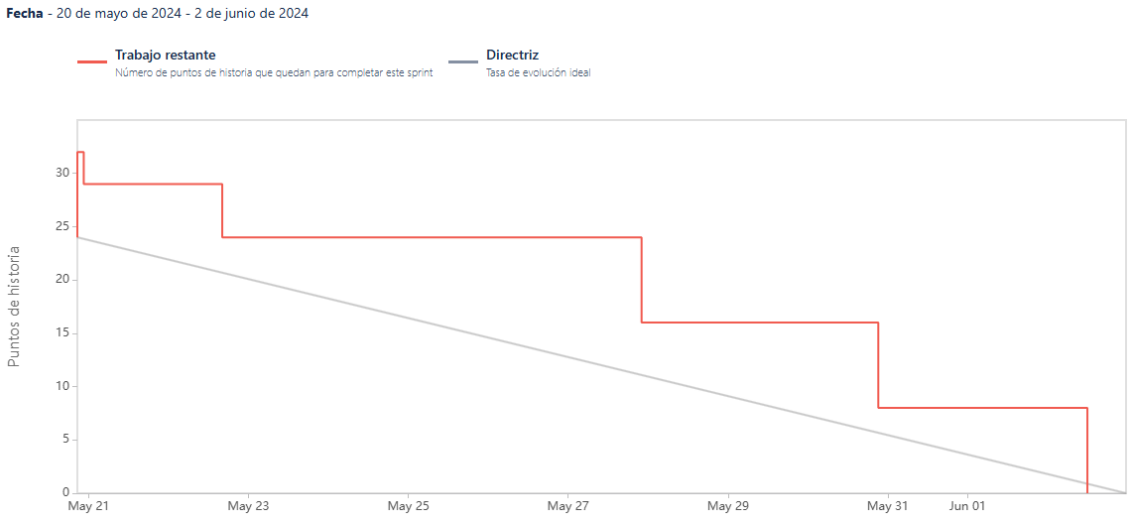
All files

98.84% Statements 513/519 **84.68%** Branches 94/111 **95.16%** Functions 59/62 **98.83%** Lines 510/516

File	Statements		Branches		Functions		Lines	
src	93.75%	15/16	100%	0/0	100%	1/1	93.75%	15/16
src/config	90.9%	10/11	62.5%	10/16	100%	0/0	90.9%	10/11
src/controllers	100%	183/183	100%	41/41	100%	19/19	100%	183/183
src/errors	100%	16/16	100%	0/0	100%	4/4	100%	16/16
src/middlewares	100%	35/35	100%	10/10	100%	3/3	100%	35/35
src/models	94.11%	64/68	28.57%	4/14	76.92%	10/13	94.11%	64/68
src/routes	100%	56/56	100%	0/0	100%	0/0	100%	56/56
src/services	100%	134/134	96.66%	29/30	100%	22/22	100%	131/131

4.5.6. *Burn-down chart*

El gráfico de pendiente, por un error en la definición de la hora de comienzo del *sprint*, hace que se refleje como que el *sprint* comienza con 24 puntos de historia, y se añaden 8 puntos de historia con el *sprint* iniciado. Realmente lo que sucede es que el *sprint* comienza con 32 puntos de historia, simplemente que una historia de 8 puntos se mueve del *product backlog* al *sprint backlog* en una hora posterior a la configurada para el inicio del *sprint.* Por tanto, la línea ideal debería tener más pendiente.



Con esa pendiente ideal con mayor inclinación, la entrega real de historias se acerca notablemente, hasta entregar todo lo planificado durante este *sprint* cumpliendo con el 100% del compromiso.

4.5.7. *Sprint review*

En esta penúltima *review*, como en las anteriores, se estudia las historias de usuario que han sido planificadas para este *sprint* y se califican como entregas y/o aceptadas si cumplen los criterios de aceptación predefinidos.

Historia	Criterios de aceptación	Entregada/ Aceptada
TFM-27 Editar operación	**Dado** un usuario en "actividad" y habiendo seleccionado una propiedad **Cuando** pulsa editar en una de las operaciones del listado **Entonces** puede actualizar la información de la operación	Sí / Sí
TFM-28 Eliminar operación	**Dado** un usuario en "actividad" y habiendo seleccionado una propiedad **Cuando** pulsa borrar en una de las operaciones del listado y confirma el aviso **Entonces** se borra la operación	Sí / Sí
TFM-31 Recogida y filtrado de datos para realizar balances	**Dado** un usuario en "Balances" **Cuando** selecciono una propiedad, un rango de fechas predefinido o personalizado y un intervalo de tiempo, y genero el balance **Entonces** se realiza un filtrado de operaciones y las operaciones necesarias para generar un balance	Sí / Sí
TFM-32 Generación de gráficos	**Dado** un usuario en "Balances" **Cuando** ha generado un balance **Entonces** se visualiza en un gráfico de barras con las rentabilidades en el rango de fechas y con intervalo seleccionado	Sí / Sí
TFM-34 Subida de documentos	**Dado** un usuario en "Documentos" **Cuando** selecciono y subo un fichero con formato pdf, word, jpeg o png **Entonces** se almacena el documento en la aplicación	Sí / Sí

Las historias de usuario relativas a la gestión de operaciones no suponen ninguna dificultad, es algo que ya se ha realizado durante el proyecto así que se desarrolla, prueba y entrega sin mayor complicación.

La dificultad de este *sprint* reside en la creación de balances. Primero se estudian distintas librerías para representar gráficos, más en concreto, gráficos de barras como se define en los criterios de aceptación. Finalmente se escoge **chart.js**, y tras conseguir con datos de pruebas representar el gráfico de barras con el formato deseado, se define el contrato a seguir entre el Front-End y Back-End.

Con ese contrato sobre la mesa, y de una forma muy simplificada, ya solo queda recoger los datos necesarios desde el Front, enviarlos al Back para hacer las operaciones que corresponden y devolver los datos tal y como los necesita el Front, donde se representará el gráfico.

La subida de documentos no es tan trivial como puede ser la creación de operaciones, al tratar ficheros, finalmente se implementa utilizando la librería **multer**, un middleware

para Node.js que se utiliza para manejar datos de formularios multipart/form-data, en este caso la subida de archivos.

4.5.8. Retrospectiva

En la ceremonia de retrospectiva del *sprint* 4 se identifica lo siguiente.

	¿Qué ha pasado?	**Acciones**
¿Qué fue bien?	Estimaciones de tiempo correctas para un *sprint*, a priori, con dificultad	
	En relación con lo anterior, entrega constante muy cerca de la pendiente ideal, algo que se refleja en el *burn-down chart.*	
¿Qué se puede mejorar?	Llegado el momento de realizar las tareas relacionadas con balances y documentos, se pierde mucho tiempo analizando posibles librerías en vez de ir directo al desarrollo	Pese a que este estudio se incluye en la estimación, sería más adecuado tener bien definido que librerías se van a utilizar antes de comenzar el desarrollo de la historia
¿Qué fue mal?	-	-

4.6. *Sprint* 5

En este último *sprint* se tienen los siguientes objetivos: terminar la épica TFM-33 Gestión de documentos, para poder listar, ver, descargar y eliminar documentos, tras acometer la subida de documentos en el anterior *sprint;* realizar la épica TFM-37 Ayuda en la declaración de la renta, con las dos historias de usuario que la componen.

4.6.1. *Sprint backlog*

El *sprint backlog* resultantes contiene las siguientes historias de usuario:

TFM-35 Listar, ver y descargar documentos					
Descripción	**Como** usuario **Quiero** listar mis documentos subidos y poder verlos y descargarlos **Para** tener la información del documento a disposición				
Prioridad	Media	**Puntos de historia**	8	**Tiempo consumido**	9h
Criterios de aceptación	**Dado** un usuario logueado **Cuando** pulsa en "Documentos" accesible desde la cabecera **Entonces** se muestra un listado con los archivos subidos ordenados con los más nuevos primero				
	Dado un usuario en "Documentos" **Cuando** pulsa para visualizar uno de los documentos del listado **Entonces** se abre en una nueva pestaña visualizando el documento (si el documento no tiene formato word)				
	Dado un usuario en "Documentos" **Cuando** pulsa para descargar uno de los documentos del listado **Entonces** se descarga el documento en la carpeta de descargas por defecto del navegador				

Tareas	
Nombre de la tarea	**Tipo**
Controlador, ruta, servicio para listar los documentos del usuario, y para leer un documento	Back-End
Interfaz para visualizar el listado de documentos, así como desarrollo de las opciones de ver y descargar los documentos	Front-End

TFM-36 Eliminar documentos					
Descripción	**Como** usuario **Quiero** poder eliminar documentos subidos **Para** que se deje de estar entre los documentos subidos a la aplicación				
Prioridad	Alta	**Puntos de historia**	3	**Tiempo consumido**	3h
Criterios de aceptación	**Dado** un usuario en "Documentos" **Cuando** pulsa borrar en uno de los documentos del listado y confirma el aviso **Entonces** se borra el documento				

	Dado un usuario en "Documentos" **Cuando** pulsa borrar todos los documentos y confirma el aviso **Entonces** se borran todos los documentos del usuario

Tareas	
Nombre de la tarea	**Tipo**
Endpoint para eliminar un documento y para eliminar todos los documentos del usuario	Back-End
Funcionalidad de borrado de un documento y de todos los documentos del usuario con dialogo de confirmación para el frontal	Front-End

TFM-38 Recogida y filtrado de datos para realizar los cálculos de la renta					
Descripción	**Como** usuario **Quiero** poder seleccionar una de mis propiedades, año fiscal, y demás datos necesarios **Para** poder hacer cálculos relacionados con la renta				
Prioridad	Media	**Puntos de historia**	13	**Tiempo consumido**	15h
Criterios de aceptación	**Dado** un usuario en "Renta" **Cuando** selecciono una propiedad que tiene registrado valor catastral, valor de construcción, importe de adquisición y gastos de adquisición **Entonces** es una propiedad válida para el cálculo de la renta				
	Dado un usuario en "Renta" y que ha seleccionado una propiedad válida **Cuando** se introduce el año fiscal, número de días que la propiedad ha sido rentada y, opcionalmente, importe en mejoras de años anterior y del ejercicio actual; y se envía **Entonces** se realizan los cálculos para calcular la renta imponible, gastos deducibles y amortización				

Tareas	
Nombre de la tarea	**Tipo**
Recogida de datos del usuario para el cálculo de la renta	Front-End
Filtro de operaciones para cálculo de renta imponible y gastos deducibles, y cálculo de amortización	Back-End

TFM-39 Presentación de ayuda fiscal para realizar la declaración de la renta					
Descripción	**Como** usuario **Quiero** visualizar la ayuda fiscal **Para** realizar la declaración de la renta de una forma más sencilla				
Prioridad	Media	**Puntos de historia**	8	**Tiempo consumido**	3h

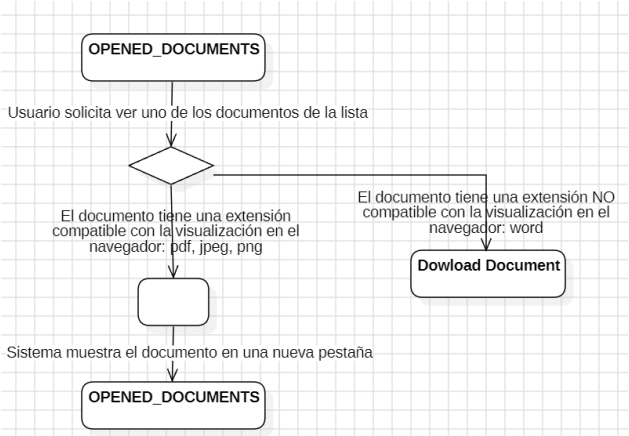
Criterios de aceptación	**Dado** un usuario en "Renta" **Cuando** ha solicitado la ayuda fiscal **Entonces** se presenta la renta imponible, gastos deducibles y amortización para la propiedad y años fiscal seleccionado

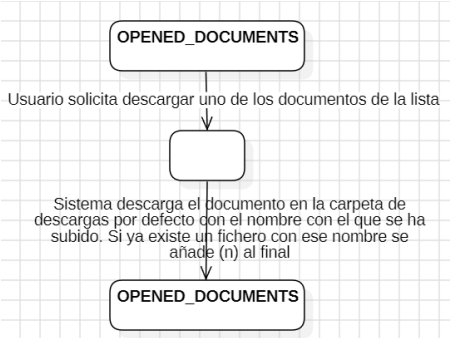
Tareas	
Nombre de la tarea	**Tipo**
Contrato con Back-End para mostrar la renta imponible, gastos deducibles y amortización	Front-End
Adaptar la renta calculada al contrato definido, para representar la ayuda fiscal en el frontal	Back-End

4.6.2. Diseño de la arquitectura

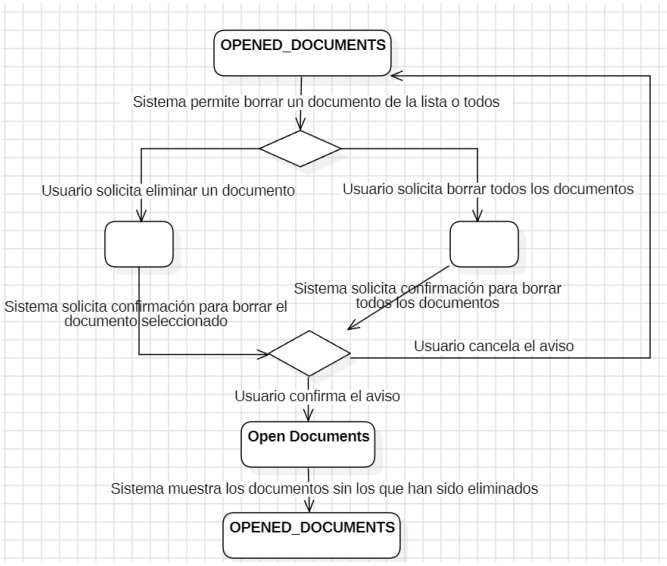
4.6.2.1. Diagramas de casos de uso

Ver documento (ViewDocument) y descargar documento (Download Document):

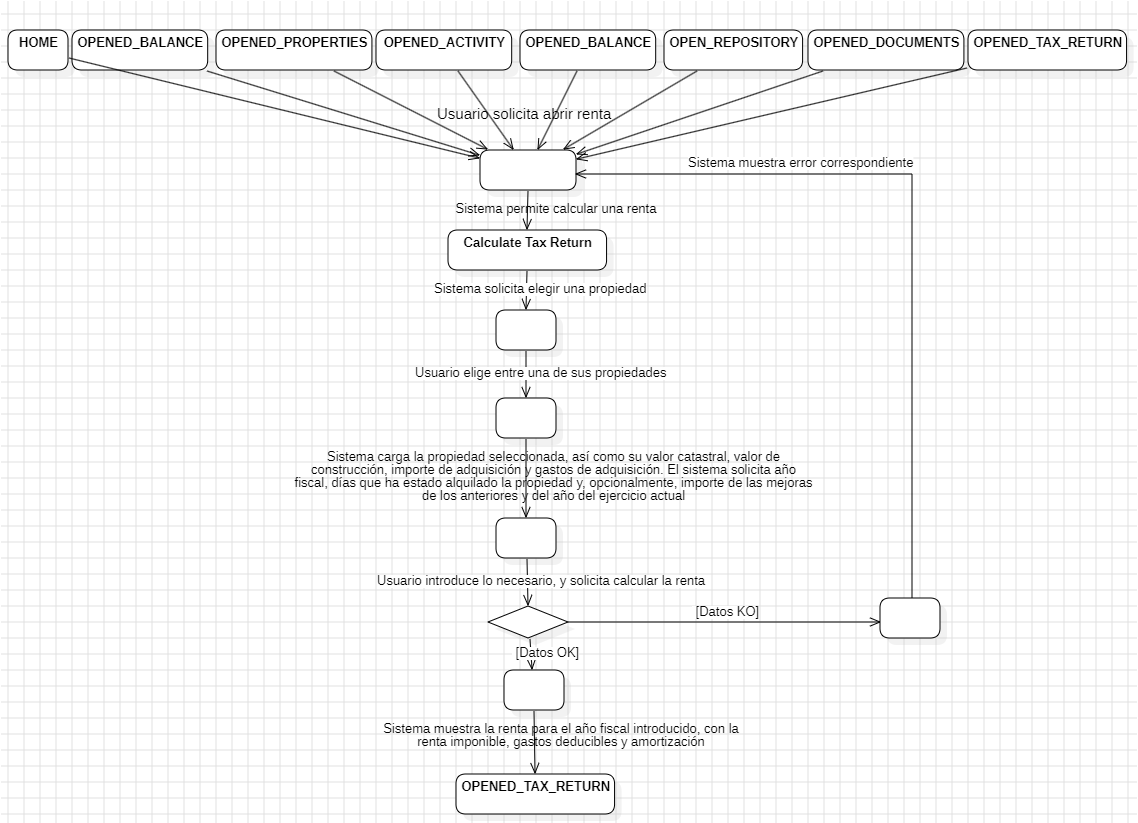




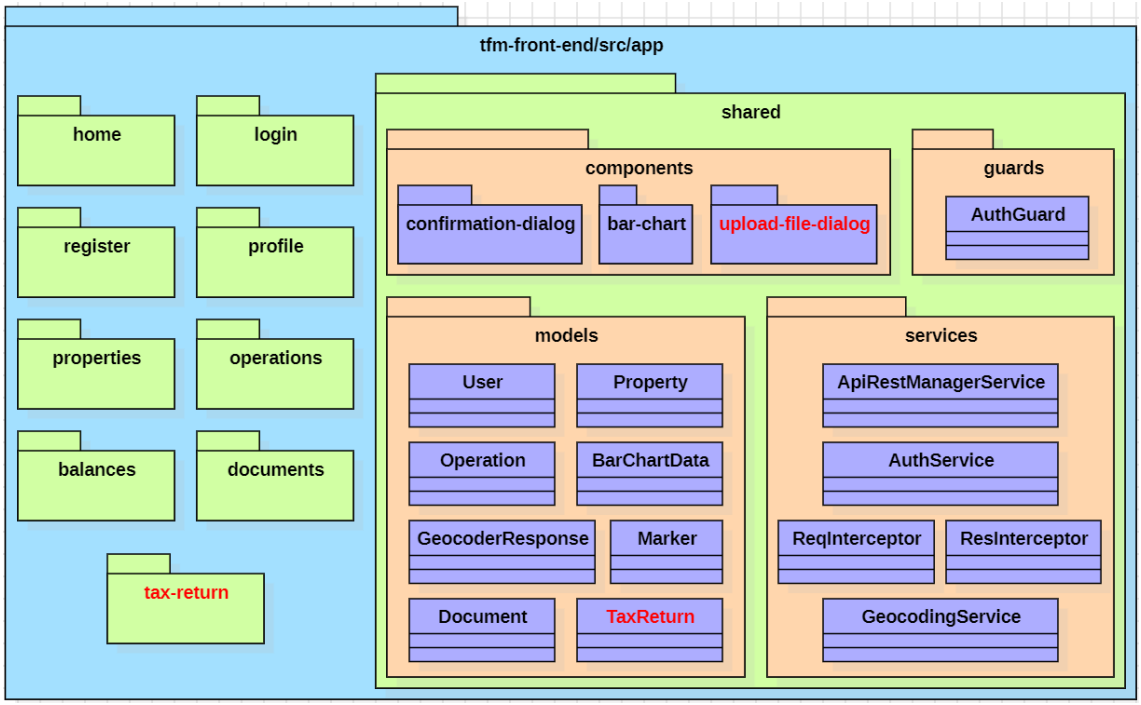
Borrar documento (DeleteDocument)



Abrir renta (OpenTaxReturn) y calcular renta (CalculateTaxReturn):



4.6.2.2.a. Diagrama de paquetes (Front)

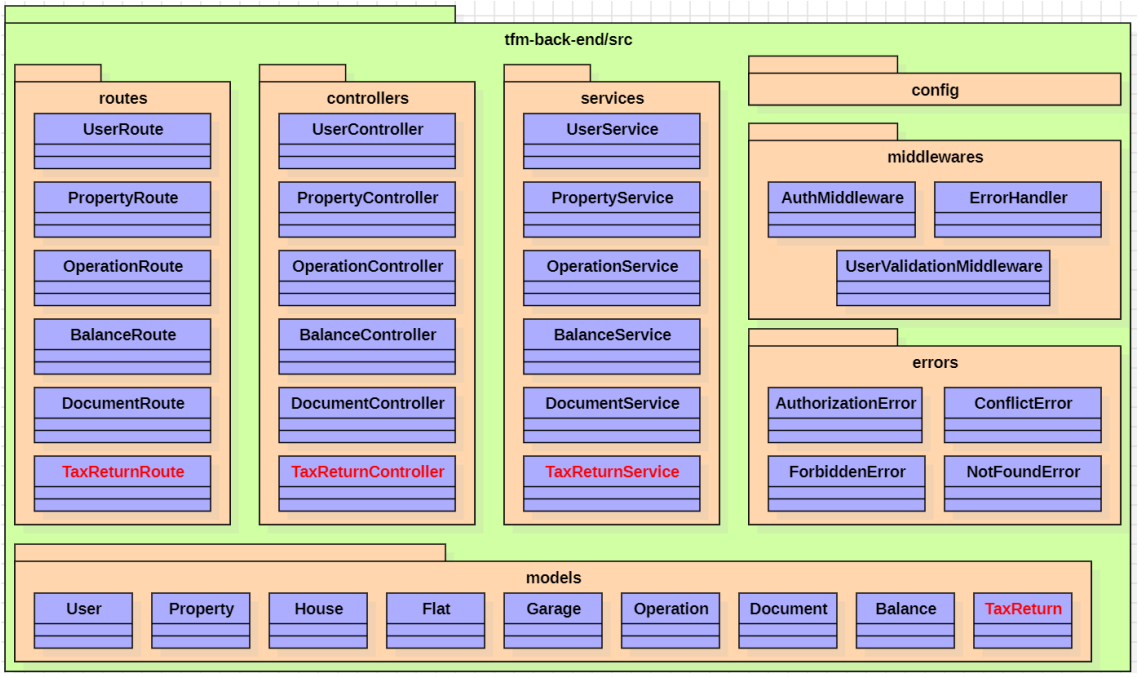


Para terminar con la gestión de documentos, en la parte frontal de la aplicación se sigue trabajando sobre el paquete de **documents**, donde durante este *sprint* se incluyen al componente **DocumentsComponent** las funcionalidades de listar los documentos, la posibilidad de leerlos en una nueva pestaña y/o descargarlos, así como borrarlos. Por

otro lado, se separa la funcionalidad de subir documento el componente compartido **UploadFileDialogComponent** para que pueda ser reutilizable.

En cuanto al cálculo de la declaración de la renta, se crea el modelo **TaxReturn** y el componente **TaxReturnComponent** donde se encuentra la interfaz de la ruta /tax-return para recoger la parte frontal de la épica TFM-37 Ayuda en la declaración de la renta.

4.6.2.2.b. Diagrama de paquetes (Back)

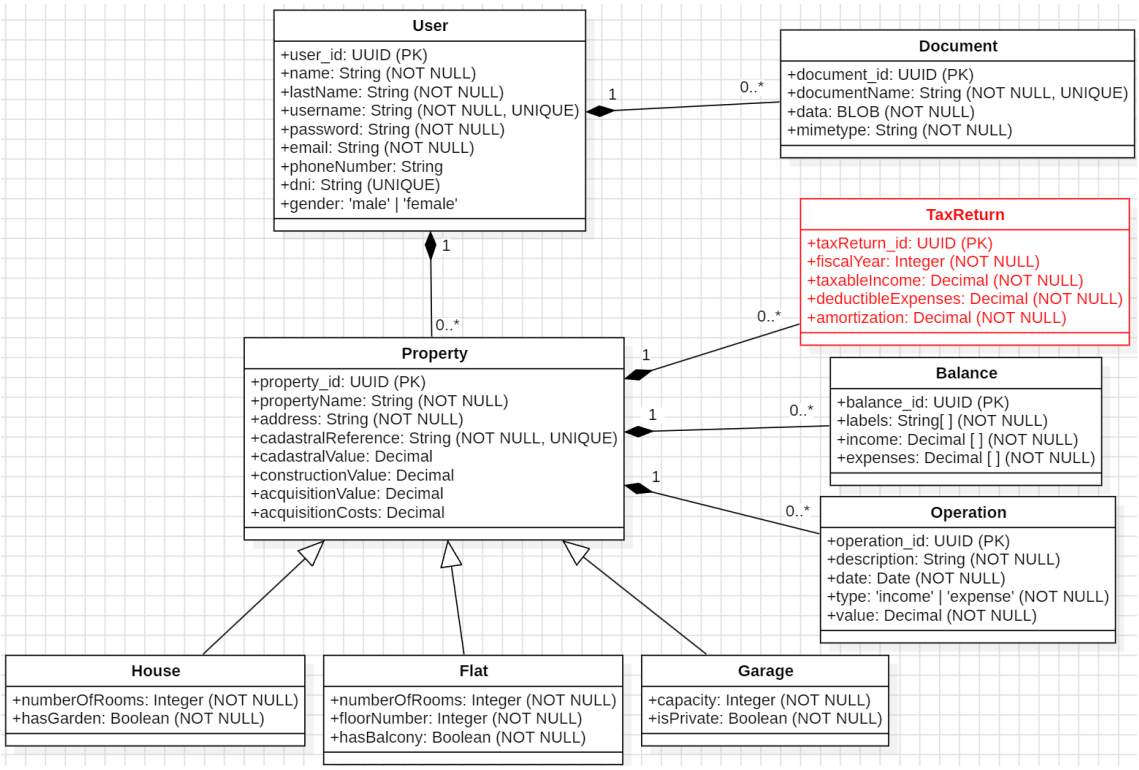


En la parte trasera sucede lo mismo que en le frontal, en el *sprint* 5 se sigue trabajando las rutas, controladores y servicios de documentos, añadiendo los *endpoints* correspondientes.

En rojo: **TaxReturnRoute**, con la ruta correspondiente al POST de /tax-return; **TaxReturnController** con el controlador asociado para esta ruta para calcular la declaración de la renta; **TaxReturnService** donde se recoge la lógica de negocio para calcular la renta imponible, gastos deducible y amortización, para lo que también se hace uso de otros servicios.

4.6.2.3. Modelo de datos

El siguiente diagrama muestra el modelo de base de datos del *sprint* 5 y que presenta dos cambios con respecto al anterior.



Por un lado, se añaden cuatro atributos al modelo Propiedad:

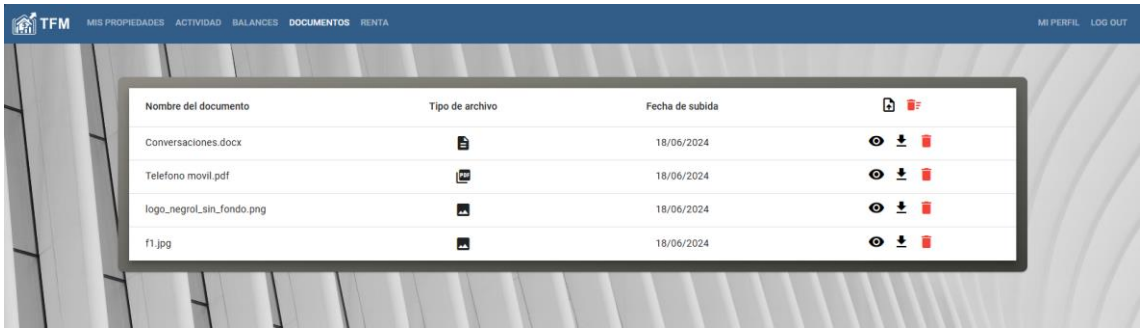
- cadastralValue
- constructionValue
- acquisitionValue
- acquisitionCosts

Todos ellos pueden ser nulos, es decir, no son obligatorios, y son de tipo Decimal, ya que representan un importe. Más adelante cuando se comente los cambios realizados en los *endpoints* afectados se comentará más acerca de estos nuevos atributos.

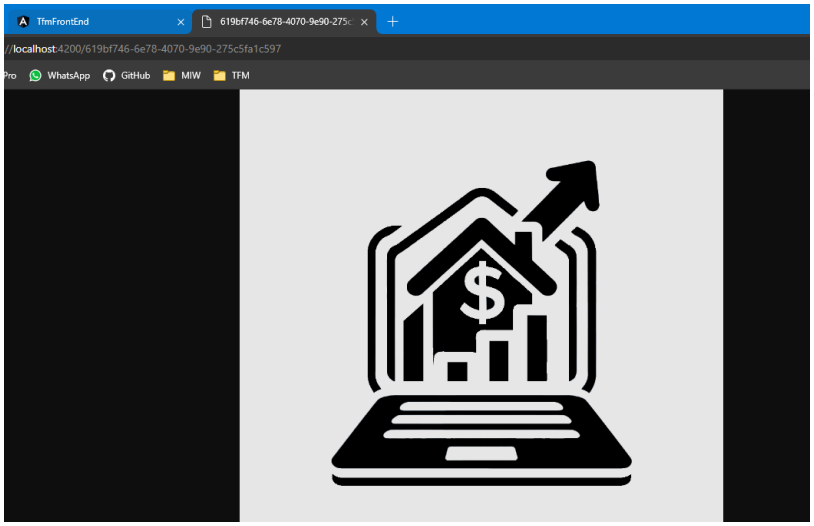
Por otro lado, el nuevo modelo de DeclaraciónRenta (TaxReturn), muy similar al modelo de Balance, ya que se guarda la renta a la hora de calcular la ayuda presentada al usuario a modo de histórico, no se prevé utilizar el registro de declaraciones de la renta guardadas.

4.6.3. Interfaz

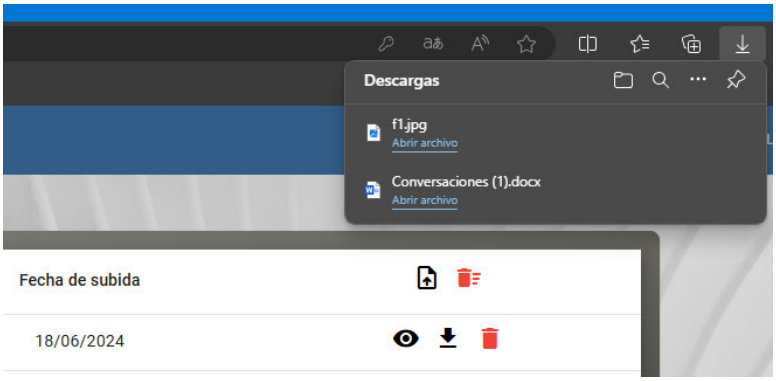
Listar documentos:



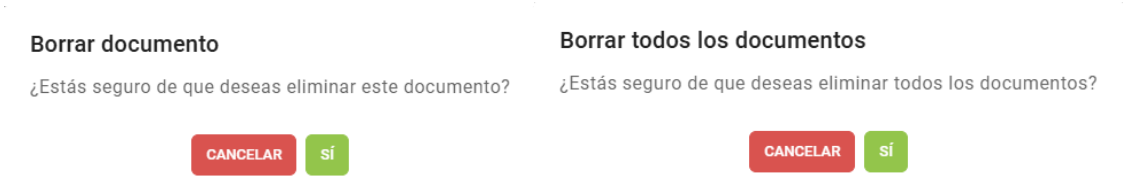
Ver documento:



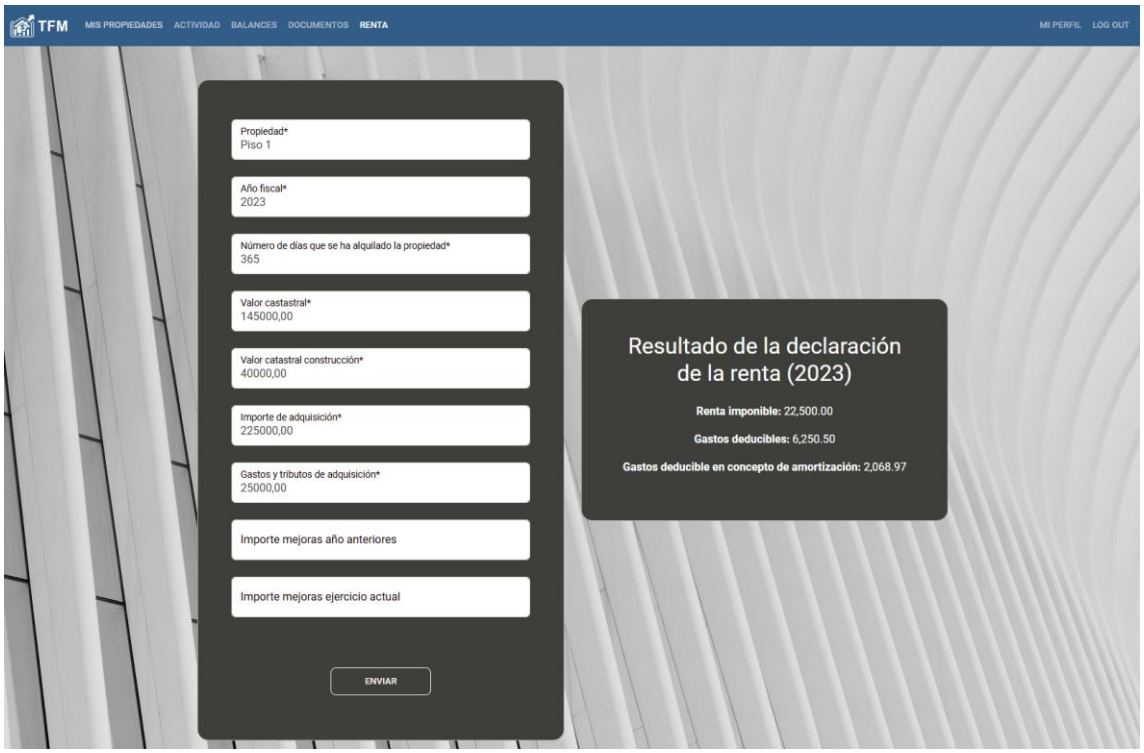
Descargar documento:



Borrar documentos:



Ayuda en la declaración de la renta:



4.6.4. *Endpoints*

Los *endpoints* para listar y leer documentos, así como para borrar uno o todos los documentos del usuario son los 4 siguientes

LISTAR DOCUMENTOS			
Endpoint			
Método	**Ruta**	**Descripción**	
GET	/documents	Lista los documentos del usuario autenticado	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
-	-	-	El usuario autenticado se obtiene mediante el token
Respuestas			
Código	**Mensaje**	**Descripción**	
200	OK	Devuelve la lista de docuemntos asociadas al usuario autenticado	
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido	

LEER DOCUMENTO			
Endpoint			
Método	**Ruta**	**Descripción**	
GET	/documents/{document_id}	Lee la información del documento seleccionado	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
document_id*	String	Path	ID del documento a leer

Respuestas		
Código	**Mensaje**	**Descripción**
200	OK	Devuelve la información del documento
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido
403	FORBIDDEN	Acceso denegado, el documento solicitado no pertenece al usuario autenticado
404	NOT FOUND	No existe un documento con ese ID

BORRAR DOCUMENTO			
Endpoint			
Método	**Ruta**	**Descripción**	
DELETE	/documents/{document_id}	Borrar el documento seleccionado	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
document_id*	String	Path	ID del documento a borrar
Respuestas			
Código	**Mensaje**	**Descripción**	
200	OK	Documento borrado, devuelve mensaje de confirmación	
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido	
403	FORBIDDEN	Acceso denegado, el documento solicitado no pertenece al usuario autenticado	
404	NOT FOUND	No existe un documento con ese ID	

BORRAR DOCUMENTO			
Endpoint			
Método	**Ruta**	**Descripción**	
DELETE	/documents	Borra todos los documentos del usuario autenticado	
Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
-	-	-	El usuario autenticado se obtiene mediante el token
Respuestas			
Código	**Mensaje**	**Descripción**	
200	OK	Documentos borrados, devuelve mensaje de confirmación	
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido	

Para realizar la declaración de la renta se crea el siguiente *endpoint*:

CALCULAR DECLARACION DE LA RENTA		
Endpoint		
Método	**Ruta**	**Descripción**
POST	/tax-return	Genera los datos necesarios para crear un balance y representarlo en un gráfico desde el frontal
Parámetros		

Nombre	Tipo	Ubicación	Descripción
property_id*	String	Body	ID de la propiedad seleccionada sobre la que hacer la declaración
fiscalYear*	Integer	Body	Año fiscal de la renta (Ej. 2023)
numberOfDaysRented*	Integer	Body	Número de días que la propiedad seleccionada ha sido rentada en el año fiscal indicado (Ej. 365)
previousYearsImprovements	Decimal	Body	Importe de las mejoras de la propiedad que se han realizado en años anteriores
currentYearImprovements	Decimal	Body	Importe de las mejoras de la propiedad que se han realizado durante el año del ejercicio

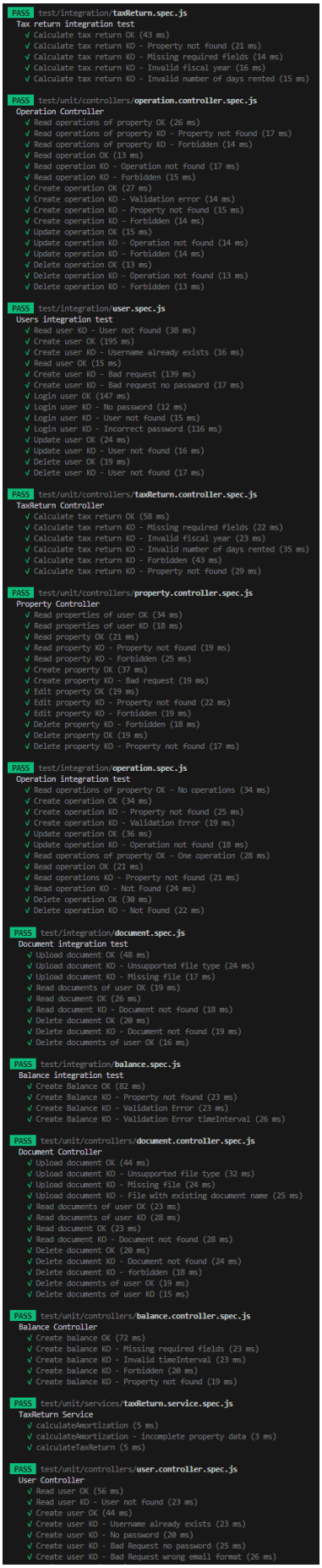
Respuestas		
Código	**Mensaje**	**Descripción**
201	OK	Devuelve el balance creado
400	BAD REQUEST	Algún parámetro del cuerpo es incorrecto
401	UNAUTHORIZED	Token no proporcionado, expirado o inválido
403	FORBIDDEN	Acceso denegado, la propiedad seleccionada para la declaración de la renta no pertenece al usuario
404	NOT FOUND	La propiedad seleccionada para la declaración de la renta no existe

A parte de la creación se los anteriores *endpoints*, se modifica la creación de propiedad (4.3.2.4) y edición de propiedad (4.4.2.4), endpoints a los cuales se añade estos cuatro parámetros:

Parámetros			
Nombre	**Tipo**	**Ubicación**	**Descripción**
cadastralValue	Decimal	Body	Valor catastral total
constructionValue	Decimal	Body	Valor catastral excluido el valor del suelo (valor de construcción)
acquisitionValue	Decimal	Body	Valor de adquisición del inmueble que figura en la escritura
acquisitionCosts	Decimal	Body	Gastos de adquisición pudiendo sumar los gastos como notaría, registro, impuestos (IVA o ITP), gastos de agencia, etc.

Hay que tener en cuenta que, pese a que son opcionales, serán obligatorios si se quiere calcular la declaración de la renta de esa propiedad ya que se utilizan para el cálculo de la amortización del inmueble.

4.6.5. Pruebas unitarias y de integración



```
PASS test/integration/taxReturn.spec.js
  Tax return integration test
    √ Calculate tax return OK (43 ms)
    √ Calculate tax return KO - Property not found (21 ms)
    √ Calculate tax return KO - Missing required fields (14 ms)
    √ Calculate tax return KO - Invalid fiscal year (16 ms)
    √ Calculate tax return KO - Invalid number of days rented (15 ms)

PASS test/unit/controllers/operation.controller.spec.js
  Operation Controller
    √ Read operations of property OK (26 ms)
    √ Read operations of property KO - Property not found (17 ms)
    √ Read operations of property KO - Forbidden (14 ms)
    √ Read operation OK (13 ms)
    √ Read operation KO - Operation not found (17 ms)
    √ Read operation KO - Forbidden (15 ms)
    √ Create operation OK (27 ms)
    √ Create operation KO - Validation error (14 ms)
    √ Create operation KO - Property not found (15 ms)
    √ Create operation KO - Forbidden (14 ms)
    √ Update operation OK (15 ms)
    √ Update operation KO - Operation not found (14 ms)
    √ Update operation KO - Forbidden (14 ms)
    √ Delete operation OK (13 ms)
    √ Delete operation KO - Operation not found (13 ms)
    √ Delete operation KO - Forbidden (13 ms)

PASS test/integration/user.spec.js
  Users integration test
    √ Read user KO - User not found (38 ms)
    √ Create user OK (195 ms)
    √ Create user KO - Username already exists (16 ms)
    √ Read user OK (15 ms)
    √ Create user KO - Bad request (139 ms)
    √ Create user KO - Bad request no password (17 ms)
    √ Login user OK (147 ms)
    √ Login user KO - No password (12 ms)
    √ Login user KO - User not found (15 ms)
    √ Login user KO - Incorrect password (116 ms)
    √ Update user OK (24 ms)
    √ Update user KO - User not found (16 ms)
    √ Delete user OK (19 ms)
    √ Delete user KO - User not found (17 ms)

PASS test/unit/controllers/taxReturn.controller.spec.js
  TaxReturn Controller
    √ Calculate tax return OK (58 ms)
    √ Calculate tax return KO - Missing required fields (22 ms)
    √ Calculate tax return KO - Invalid fiscal year (23 ms)
    √ Calculate tax return KO - Invalid number of days rented (35 ms)
    √ Calculate tax return KO - Forbidden (43 ms)
    √ Calculate tax return KO - Property not found (29 ms)

PASS test/unit/controllers/property.controller.spec.js
  Property Controller
    √ Read properties of user OK (34 ms)
    √ Read properties of user KO (18 ms)
    √ Read property OK (21 ms)
    √ Read property KO - Property not found (19 ms)
    √ Read property KO - Forbidden (25 ms)
    √ Create property OK (37 ms)
    √ Create property KO - Bad request (19 ms)
    √ Edit property OK (19 ms)
    √ Edit property KO - Property not found (22 ms)
    √ Edit property KO - Forbidden (19 ms)
    √ Delete property KO - Forbidden (18 ms)
    √ Delete property OK (19 ms)
    √ Delete property KO - Property not found (17 ms)

PASS test/integration/operation.spec.js
  Operation integration test
    √ Read operations of property OK - No operations (34 ms)
    √ Create operation OK (34 ms)
    √ Create operation KO - Property not found (25 ms)
    √ Create operation KO - Validation Error (19 ms)
    √ Update operation OK (36 ms)
    √ Update operation KO - Operation not found (18 ms)
    √ Read operations of property OK - One operation (28 ms)
    √ Read operation OK (21 ms)
    √ Read operations KO - Property not found (21 ms)
    √ Read operation KO - Not Found (24 ms)
    √ Delete operation OK (30 ms)
    √ Delete operation KO - Not Found (22 ms)

PASS test/integration/document.spec.js
  Document integration test
    √ Upload document OK (48 ms)
    √ Upload document KO - Unsupported file type (24 ms)
    √ Upload document KO - Missing file (17 ms)
    √ Read documents of user OK (19 ms)
    √ Read document OK (26 ms)
    √ Read document KO - Document not found (18 ms)
    √ Delete document OK (20 ms)
    √ Delete document KO - Document not found (19 ms)
    √ Delete documents of user OK (16 ms)

PASS test/integration/balance.spec.js
  Balance integration test
    √ Create Balance OK (82 ms)
    √ Create Balance KO - Property not found (23 ms)
    √ Create Balance KO - Validation Error (23 ms)
    √ Create Balance KO - Validation Error timeInterval (26 ms)

PASS test/unit/controllers/document.controller.spec.js
  Document Controller
    √ Upload document OK (44 ms)
    √ Upload document KO - Unsupported file type (32 ms)
    √ Upload document KO - Missing file (24 ms)
    √ Upload document KO - File with existing document name (25 ms)
    √ Read documents of user OK (23 ms)
    √ Read documents of user KO (28 ms)
    √ Read document OK (23 ms)
    √ Read document KO - Document not found (28 ms)
    √ Delete document OK (20 ms)
    √ Delete document KO - Document not found (24 ms)
    √ Delete document KO - Forbidden (18 ms)
    √ Delete documents of user OK (19 ms)
    √ Delete documents of user KO (15 ms)

PASS test/unit/controllers/balance.controller.spec.js
  Balance Controller
    √ Create balance OK (72 ms)
    √ Create balance KO - Missing required fields (23 ms)
    √ Create balance KO - Invalid timeInterval (23 ms)
    √ Create balance KO - Forbidden (20 ms)
    √ Create balance KO - Property not found (19 ms)

PASS test/unit/services/taxReturn.service.spec.js
  TaxReturn Service
    √ calculateAmortization (5 ms)
    √ calculateAmortization - incomplete property data (3 ms)
    √ calculateTaxReturn (5 ms)

PASS test/unit/controllers/user.controller.spec.js
  User Controller
    √ Read user OK (56 ms)
    √ Read user KO - User not found (23 ms)
    √ Create user OK (44 ms)
    √ Create user KO - Username already exists (23 ms)
    √ Create user KO - No password (20 ms)
    √ Create user KO - Bad Request no password (25 ms)
    √ Create user KO - Bad Request wrong email format (26 ms)
    √ Login user OK (25 ms)
    √ Login user KO - No username (28 ms)
    √ Login user KO - User not found (24 ms)
    √ Login user KO - Incorrect password (29 ms)
    √ Update user OK (39 ms)
    √ Update user KO - User not found (34 ms)
    √ Delete user OK (22 ms)
    √ Delete user KO - User not found (35 ms)

PASS test/integration/property.spec.js
  Property integration test
    √ Read properties of user OK - No properties (30 ms)
    √ Create property OK (48 ms)
    √ Create property KO - Bad request (18 ms)
    √ Read properties of user OK (23 ms)
    √ Read property OK (31 ms)
    √ Read property KO - Property not found (35 ms)
    √ Update property OK (45 ms)
    √ Delete property OK (29 ms)
    √ Delete property KO - Property not found (22 ms)

PASS test/unit/services/balance.service.spec.js
  Balance Service
    √ calculateBalanceForInterval OK - Interval 0 (8 ms)
    √ calculateBalanceForInterval OK - With interval (4 ms)

PASS test/unit/services/document.service.spec.js
  Document Service
    √ createDocument OK (14 ms)
    √ createDocument with missing fields (3 ms)
    √ createDocument with invalid data (4 ms)
    √ readDocumentByDocumentName (5 ms)
    √ readDocumentByDocumentId (6 ms)
    √ readDocumentsByUserId (16 ms)
    √ deleteDocumentByDocumentId (5 ms)
    √ deleteDocumentsByUserId (4 ms)
    √ validateDocumentOwnership (2 ms)
    √ validateDocumentOwnership with non-existing document (2 ms)
    √ validateDocumentOwnership with non-owner user (2 ms)

PASS test/unit/models/user.model.spec.js
  User Model
    √ Find User by username (12 ms)
    √ Create user (4 ms)
    √ Create user KO - missing required field (2 ms)
    √ Update user (1 ms)
    √ Delete user (2 ms)

PASS test/unit/services/user.service.spec.js
  User Service
    √ readUser OK (4 ms)
    √ readUser OK - But user not found (2 ms)
    √ createUser OK (1 ms)
    √ isCorrectPassword OK (1 ms)
    √ generateToken OK (2 ms)
    √ updateUser OK (1 ms)
    √ deleteUser OK (1 ms)
    √ deleteUser KO - User not found (1 ms)

PASS test/unit/models/balance.model.spec.js
  Balance Model
    √ Save balance (14 ms)

PASS test/unit/models/document.model.spec.js
  Document Model
    √ Create document (7 ms)
    √ Create document missing parameter (1 ms)
    √ Read document by documentName (2 ms)
    √ Read document by documentId (1 ms)
    √ Read documents by userId (2 ms)
    √ Delete document by documentId (4 ms)
    √ Delete documents by userId (2 ms)

PASS test/unit/models/property.model.spec.js
  Property Model
    √ Find all properties (9 ms)
    √ Find Property by property_id (2 ms)
    √ Create property (3 ms)
    √ Create property missing parameter (2 ms)
    √ Create property with missing required field (3 ms)
    √ Update property (1 ms)
    √ Delete property (5 ms)

PASS test/unit/models/taxReturn.model.spec.js
  Tax Return Model
    √ Save tax return (9 ms)

PASS test/unit/middlewares/auth.spec.js
  Auth Middleware
    √ Auth KO - Token not provided (37 ms)
    √ Auth KO - Token is invalid (15 ms)
    √ Auth KO - Token has expired (20 ms)
    √ Auth OK (19 ms)

PASS test/unit/services/operation.service.spec.js
  Property Service
    √ readOperationsByPropertyId (4 ms)
    √ readOperationsByPropertyIdAndDateRange (2 ms)
    √ readOperation (2 ms)
    √ createOperation (2 ms)
    √ updateOperation (1 ms)
    √ deleteOperation (2 ms)

PASS test/unit/models/operation.model.spec.js
  Operation Model
    √ Find all operations by property id (14 ms)
    √ Find Operation by operation_id (2 ms)
    √ Create operation (4 ms)
    √ Create operation with missing required field (2 ms)
    √ Update operation (3 ms)
    √ Delete operation (2 ms)

PASS test/unit/services/property.service.spec.js
  Property Service
    √ Return properties by user id (7 ms)
    √ Read property by id (1 ms)
    √ Create property House (2 ms)
    √ Create property Flat (3 ms)
    √ Create property Garage (2 ms)
    √ Update property House (1 ms)
    √ Update property Flat (1 ms)
    √ Update property Garage (2 ms)
    √ Delete property by id (6 ms)
    √ Delete property by id - Property not found (2 ms)
    √ Validate property ownership (2 ms)
    √ Validate property ownership - Property not found (18 ms)
    √ Validate property ownership - User is not the owner (2 ms)

PASS test/unit/middlewares/userValidation.spec.js
  User Validation middleware
    √ Requested username is the same as the authenticated username (4
    √ Requested username is not the same as the authenticated username

Test Suites: 26 passed, 26 total
Tests:       197 passed, 197 total
Snapshots:   0 total
Time:        31.157 s
```

4.6.5.1. Cobertura de código

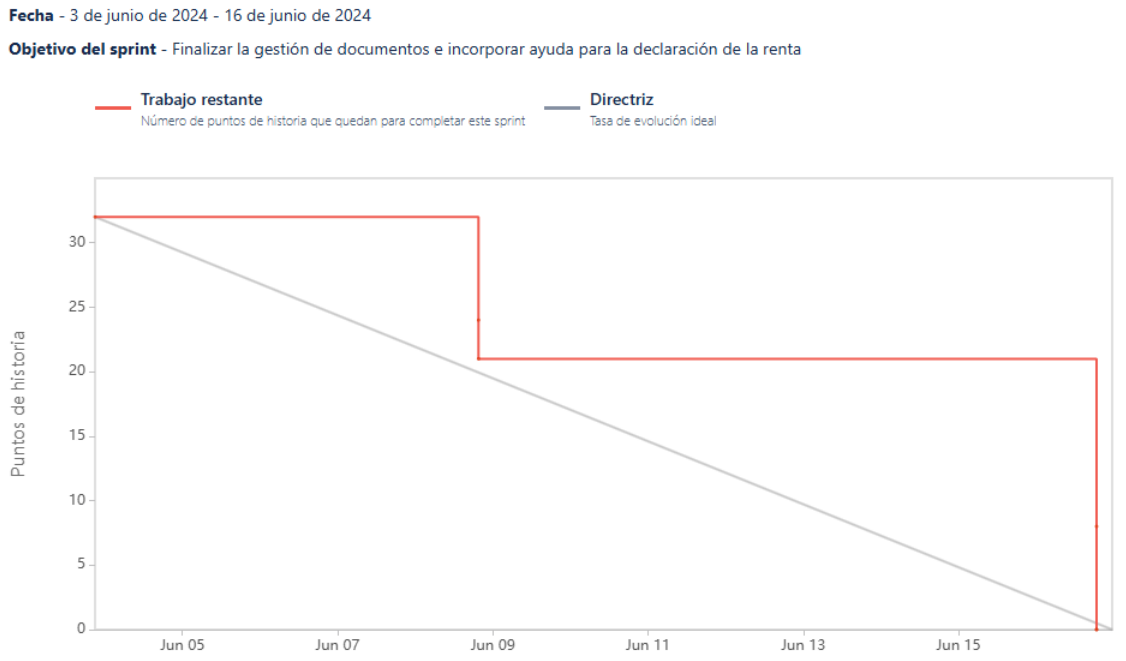
All files

99.04% Statements 622/628 **86.66%** Branches 117/135 **96.1%** Functions 74/77 **99.04%** Lines 619/625

File		Statements		Branches		Functions		Lines	
src		93.75%	15/16	100%	0/0	100%	1/1	93.75%	15/16
src/config		90.9%	10/11	62.5%	10/16	100%	0/0	90.9%	10/11
src/controllers		100%	228/228	100%	51/51	100%	26/26	100%	228/228
src/errors		100%	16/16	100%	0/0	100%	4/4	100%	16/16
src/middlewares		100%	34/34	100%	10/10	100%	3/3	100%	34/34
src/models		94.59%	70/74	31.25%	5/16	76.92%	10/13	94.59%	70/74
src/routes		100%	68/68	100%	0/0	100%	0/0	100%	68/68
src/services		100%	181/181	97.61%	41/42	100%	30/30	100%	178/178

4.6.6. *Burn-down chart*

La planificación del *sprint* comienza con 32 puntos de historia y a final del *sprint* se logra entregar todo.



Mirando en detalle el gráfico de pendiente puede observarse dos escalones. El primero corresponde con la entrega de las dos primeras historias, que al ser funcionalidades de gestión de documentos terminan de probarse y entregarse al mismo tiempo. Sucede los mismo con las dos historias de la épica TFM-37 Ayuda en la declaración de la renta.

4.6.7. *Sprint review*

En la siguiente tabla se muestran las cuatro historias de usuario incluidas en la planificación del *sprint*, junto a sus criterios de aceptación para verificar su aceptación, como también si han sido entregadas.

Historia	Criterios de aceptación	Entregada/ Aceptada
TFM-35 Listar, ver y descargar documentos	**Dado** un usuario logueado **Cuando** pulsa en "Documentos" accesible desde la cabecera **Entonces** se muestra un listado con los archivos subidos ordenados con los más nuevos primero	Sí / Sí
	Dado un usuario en "Documentos" **Cuando** pulsa para visualizar uno de los documentos del listado **Entonces** se abre en una nueva pestaña visualizando el documento (si el documento no tiene formato word)	
	Dado un usuario en "Documentos" **Cuando** pulsa para descargar uno de los documentos del listado **Entonces** se descarga el documento en la carpeta de descargas por defecto del navegador	
TFM-36 Eliminar documentos	**Dado** un usuario en "Documentos" **Cuando** pulsa borrar en uno de los documentos del listado y confirma el aviso **Entonces** se borra el documento	Sí / Sí
	Dado un usuario en "Documentos" **Cuando** pulsa borrar todos los documentos y confirma el aviso **Entonces** se borran todos los documentos del usuario	
TFM-38 Recogida y filtrado de datos para realizar los cálculos de la renta	**Dado** un usuario en "Renta" **Cuando** selecciono una propiedad que tiene registrado valor catastral, valor de construcción, importe de adquisición y gastos de adquisición **Entonces** es una propiedad válida para el cálculo de la renta	Sí / Sí
	Dado un usuario en "Renta" y que ha seleccionado una propiedad válida **Cuando** se introduce el año fiscal, número de días que la propiedad ha sido rentada y, opcionalmente, importe en mejoras de años anterior y del ejercicio actual; y se envía **Entonces** se realizan los cálculos para calcular la renta imponible, gastos deducibles y amortización	
TFM-39 Presentación de ayuda fiscal para realizar la declaración de la renta	**Dado** un usuario en "Renta" **Cuando** ha solicitado la ayuda fiscal **Entonces** se presenta la renta imponible, gastos deducibles y amortización para la propiedad y años fiscal seleccionado	Sí / Sí

Como ayuda para la declaración de la renta de los inmuebles del usuario se presenta la renta imponible y los gastos deducibles que se obtienen gracias a los ingresos y gastos introducidos en la aplicación, utilizando los del año fiscal del ejercicio que se quiere realizar.

También se presentan los gastos deducibles en concepto de amortización, es decir, se entiende que los bienes inmuebles pierden valor por funcionamiento, uso, disfrute u obsolescencia, por tanto, la Agencia Tributaria permite deducirse de estos gastos referentes a la deprecación efectiva del inmueble (Normativa: Arts. 23.1 b) Ley IRPF y 13 h) y 14 Reglamento).

En la funcionalidad desarrollada no se está teniendo en cuenta situaciones especiales, por ejemplo, el límite de la amortización acumulada de los inmuebles, y que dependerá si los bienes han sido adquiridos a título oneroso o a título lucrativo. Como trabajo futuro podría incluirse esas situaciones específicas no contempladas en el desarrollo de este *sprint*.

4.6.8. Retrospectiva

Como retrospectiva del último *sprint* puede extraerse lo siguiente.

	¿Qué ha pasado?	**Acciones**
¿Qué fue bien?	A pesar de que el gráfico de pendiente presenta dos escalones muy pronunciados porque que solo representa la entrega de tareas, en el *sprint* se va desarrollando de forma continua.	
¿Qué se puede mejorar?	Desconocimiento de ciertos aspectos y cálculos fiscales, por tanto, no se tiene claro que datos se necesitan para realizar la ayuda en la renta	Un estudio previo de todos los aspectos fiscales necesarios hubiese ayudado tanto el diseño como el desarrollo del *sprint*, en concreto de la épica TFM-37
¿Qué fue mal?	-	-

5. CONCLUSIONES Y LÍNEAS FUTURAS

5.1. Conclusiones

La idea de este proyecto nace con el objetivo principal de ofrecer un nuevo servicio web en el mercado inmobiliario. Esto implica el diseño y desarrollo de una aplicación web dirigida a los propietarios de inmuebles, y que desean gestionar sus propiedades alquiladas de forma activa, sin intermediarios como gestores.

Para cumplir ese propósito, se han definido varias funcionalidades de acuerdo con las necesidades del usuario objetivo. La principal de ellas es la gestión de propiedades, ya que las demás funcionalidades giran en torno a esta. Se ha implementa con éxito la gestión de diferentes tipos de inmuebles (casas, pisos, garajes) de forma modular, permitiendo la incorporación de nuevos tipos de bienes inmuebles sin complicaciones. Además, la adición de un mapa que no estaba en la idea original, mejora significativamente la experiencia del usuario, permitiendo la visualización geográfica de las propiedades.

Igualmente, la gestión de operaciones asociadas a las propiedades se ha resuelto satisfactoriamente, permitiendo al usuario registrar los ingresos y gastos derivados de sus alquileres. Sobre las propiedades y operaciones, se apoyan funcionalidades como la generación de balances para estudiar la rentabilidad mediante gráficos y el cálculo de la renta como ayuda al usuario para a declaración de impuesto. Ambas funcionalidades se han entregan correctamente, incluso cuando la incorporación de la última estaba en duda en la planificación global, lo que demuestra que el ritmo de desarrollo ha sido igual o superior a lo esperado.

Aunque quizás no tan llamativas, se planificaron funcionalidades esenciales para esta aplicación web como la gestión de usuarios, proporcionando autenticación y seguridad para una web que contiene datos sensibles de los usuarios, y la gestión de documentos, necesaria debido a la gran cantidad de trámites burocráticos, facturas y un largo etcétera relaciona con los inmuebles. Con el desarrollo exitoso de todas estas utilidades, se da por alcanzado satisfactoriamente el objetivo principal del proyecto.

No tan orientados a la funcionalidad de la aplicación, sino más a la parte técnica de esta, se definieron una serie de objetivos específicos. La creación de una parte servidora en forma de API REST con Node.js se cumple de manera efectiva, al tener una base de Node.js con el *framework* Express. Además, se ha creado un buen entorno de pruebas, que “testea” meticulosamente el Back-End con test unitarios y de integración, e incluso se han realizado pruebas de API con Postman, algo que no ha sido mostrado en este documento.

En la parte frontal, a diferencia de la parte servidora, el conocimiento era mucho menor y se limitaba a lo visto durante el máster. A pesar de eso, se han obtenido buenos resultados con un Front-End desarrollado con Angular que se conecta con la API creada. Igualmente, la parte de persistencia de la aplicación era un área sin mucho dominio, y, quizás, una de las que más dificultades ha presentado. El uso de Sequelize ORM no ha resultado tan fácil como se esperaba, y si hubiese que comenzar de nuevo el proyecto, quizás se replantearía realizarlo directamente con SQL.

Todo el proyecto se ha desarrollado dentro de un ecosistema de software que, a pesar de los problemas iniciales con la conexión de GitHub con servicios de terceros, ha resultado ser un entorno de trabajo muy eficiente. Se estableció una conexión entre los repositorios de GitHub y Jira para integrar la gestión del proyecto con las actividades de desarrollo. Además, se implementó un flujo de integración continua en GitHub Actions para cada repositorio, lo cual ha facilitado notablemente la detección temprana de errores mediante la automatización de pruebas. Este enfoque también ha contribuido a mantener altos estándares de calidad y seguridad, gracias al análisis estático del código.

En general, los resultados de la aplicación web se consideran exitosos. Los objetivos han sido alcanzados dentro del tiempo previsto. Las complicaciones propias de un proyecto software se han resuelto con soltura y eficacia. Además, los conocimientos y aptitudes adquiridos durante el máster han sido aplicados, demostrados y consolidados con este trabajo final.

5.2. Líneas futuras

A continuación, se presenta las posibles líneas futuras de desarrollo y mejora de la aplicación.

- **Pruebas en el frontal**. Mientras que la parte del servidor tiene una cobertura de prueba casi del 100% mediante pruebas unitarias, de integración y de API, el Front-End carece de pruebas más allá de pruebas manuales (situadas en el pico de la pirámide de Cohn). Dado que en el máster no se abordaron las pruebas en el Front-End, una de las líneas futuras será investigar y estudiar este tipo de pruebas e incorporarlas en el proyecto de Angular.
- **Despliegue**. Durante el proyecto se ha trabajado con los proyectos de Back-End y Front-End en un entorno local. Como trabajo futuro, queda pendiente realizar el despliegue de estos en una plataforma especializada en la nube. Mas allá de un despliegue manual, el objetivo sería incorporar al flujo de integración continua existente en GitHub Actions una fase de despliegue continuo (CI/CD). Esto implica automatizar el proceso de entrega de software, asegurando que cada cambio validado pase por pruebas automatizadas y sea desplegado automáticamente. Se contemplaría también la implementación de monitoreo continuo y estrategias de *rollback* automático para mantener la estabilidad y disponibilidad del sistema después de cada despliegue.
- **Filtro y búsqueda Avanzada**. En la lista de "noes" realizada en el *inception deck* (ver sección 2.4), en la categoría "por decidir" se incluía el filtro y búsqueda avanzada. Este aspecto se refiere al filtrado de propiedades, operaciones, documentos, basado en criterios como fecha, nombre y otros criterios relevantes. Esta funcionalidad se considera esencial, ya que facilita la búsqueda y organización de elementos en los diferentes listados de la aplicación. También se contemplaría la capacidad de ordenar por columnas en las tablas, mejorando así la experiencia de usuario al interactuar con grandes conjuntos de datos.
- **Nuevas funcionalidades**. En la categoría de "fuera de alcance" del *inception deck* existen nuevas líneas para ampliar el proyecto tales como la creación de nuevos roles, comparativa de rentabilidades, exportación de datos, chat en tiempo real con experto, etc. Recibir retroalimentación de los usuarios permitirá identificar sus necesidades y definir las nuevas áreas de desarrollo futuro.

BIBLIOGRAFÍA

Ortiz, O. (2024, 19 junio). El precio de la vivienda supera su techo histórico a pesar la nueva ley. *Qué!* Recuperado de https://www.que.es/2024/06/19/precio-vivienda-techo-historico/

Blanco, I. (2024, 18 junio). La escasez de vivienda castiga a la generación Z: «Para los jóvenes comprar una casa es una utopía, no vivirán como sus abuelos». *20minutos*. Recuperado de https://www.20minutos.es/noticia/5521617/0/escasez-vivienda-castiga-generacion-z-para-los-jovenes-comprar-una-casa-es-una-utopia-no-viviran-como-sus-abuelos/

Laoyan, S. (2024, 8 febrero). Agile Manifesto: Qué son las metodologías ágiles. *Asana*. Recuperado de https://asana.com/es/resources/agile-methodology

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifesto for Agile Software Development. *Agile Alliance*. https://agilemanifesto.org/

Martins, J. (2024, 15 febrero). Scrum: Conceptos clave y cómo se aplica en la gestión de proyectos. *Asana*. Recuperado de https://asana.com/es/resources/what-is-scrum

Alonso, A. (2021, 7 diciembre). Recorrido por las 10 dinámicas de Agile Inception. *Medium*. Recuperado de https://adrianalonsodev.medium.com/recorrido-por-las-10-din%C3%A1micas-de-agile-inception-d6ebbd1af2ef

Angular. (s. f.). Angular Documentation. Recuperado de https://v16.angular.io/docs

Angular Components. (s. f.). Angular material. Recuperado de https://material.angular.io/

Bootstrap. (s. f.). Bootstrap documentation. Recuperado de https://getbootstrap.com/docs/5.3/

Park, T. (s. f.). Bootswatch: Sandstone. *Bootswatch.com.* Recuperado de https://bootswatch.com/sandstone/

MDBootstrap.com. (2023). Bootstrap Profile - free examples, templates & tutorial. Recuperado de https://mdbootstrap.com/docs/standard/extended/profiles/

MDBootstrap.com. (2023). Free Bootstrap starter templates. Recuperado de https://mdbootstrap.com/freebies/

nodejs.org. (s. f.). Node.js - Introduction to Node.js. Recuperado de https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

Diego.Coder. (2024, 17 enero). Debugging de aplicaciones Node.js con VSCode. *Medium*. Recuperado de https://medium.com/@diego.coder/debugging-de-aplicaciones-node-js-con-vscode-5f02e5d2e900

Manandhar, G. (2024, 4 junio). Organizing your Express.js project structure for better productivity. *LogRocket*. Recuperado de https://blog.logrocket.com/organizing-express-js-project-structure-better-productivity/

Fanizzi, R. (2021, 12 diciembre). Registro & Login con nodejs (Jwt, bcrypt), express, y MySQL (Backend). *Medium*. Recuperado de https://medium.com/@ricardo.fanizzi/registro-login-con-node-js-express-y-mysql-backend-7eea7440837a

Sequelize. (2024, 20 junio). Sequelize - Getting started. Recuperado de https://sequelize.org/docs/v6/getting-started/

Patil, S. (2022, 23 agosto). Chart.js Tutorial – How to Make Bar and Line Charts in Angular. Recuperado de https://www.freecodecamp.org/news/how-to-make-bar-and-line-charts-using-chartjs-in-angular/

Manojkumar. (2023, 11 agosto). Integrating Google Maps in Angular 16. *Medium*. Recuperado de https://medium.com/yavar/integrating-google-maps-in-angular16-6990a2e69a28

Borkar, N. (2021, 14 diciembre). Google Maps in Angular with Marker, Info window and Marker Clustering. *Medium*. Recuperado de https://medium.com/@neetishborkar/google-maps-in-angular-with-marker-info-window-and-marker-clustering-65778e961c51

Jest. (2024, 16 enero). Jest - Getting started. Recuperado de https://jestjs.io/docs/getting-started

Orie, C. (2019, 8 noviembre). Testing NodeJs/Express API with Jest and Supertest. Recuperado de https://dev.to/nedsoft/testing-nodejs-express-api-with-jest-and-supertest-1km6

Hämmerle, F. (2020, 23 abril). Using dotenv with Jest. *Medium*. Recuperado de https://lusbuab.medium.com/using-dotenv-with-jest-7e735b34e55f

Mijacobs. (2023, 5 octubre). ¿Qué es Git? - Azure DevOps. Recuperado de https://learn.microsoft.com/es-es/devops/develop/git/what-is-git

Fernández, Y. (2019, 30 octubre). Qué es Github y qué es lo que le ofrece a los desarrolladores. *Xataka*. Recuperado de https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores

GitHub. (s. f.). Learn GitHub Actions. Recuperado de https://docs.github.com/en/actions/learn-github-actions

Abakumov, S. (2023, 11 noviembre). How to make MySQL work in your GitHub Actions. Recuperado de https://ovirium.com/blog/how-to-make-mysql-work-in-your-github-actions/

SonarSource. (s. f.). SonarCloud Documentation. Recuperado de https://docs.sonarsource.com/sonarcloud/