ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DEL SOFTWARE

Curso Académico 2024/2025

Trabajo Fin de Grado

APLICACIÓN PARA LA GESTIÓN DE RECURSOS HOSPITALARIOS, ASIGNACIÓN INTELIGENTE DE CITAS Y RASTREO EN TIEMPO REAL DE PACIENTES

Autor: Antonio José Alanís Bernal

Director: Manuel Rubio Sánchez

Licencia

Resumen

Este Trabajo de Fin de Grado (TFG) presenta el desarrollo de un sistema informático orientado a la administración eficiente de recursos hospitalarios, con el objetivo de optimizar y automatizar la asignación de citas médicas, así como permitir el rastreo en tiempo real de los pacientes.

El objetivo principal es desarrollar una plataforma accesible, intuitiva y funcional que optimice la gestión de recursos hospitalarios, agilice los procesos clínicos, facilite la programación de citas —mediante algoritmos de programación lineal entera binaria— y reduzca los tiempos de espera, permitiendo además un seguimiento eficiente de los pacientes.

Para su desarrollo, se realizará un estudio preliminar del entorno hospitalario, identificando los principales cuellos de botella y oportunidades de mejora mediante soluciones digitales, con el fin de orientar el diseño de la aplicación hacia una mayor eficiencia y valor añadido.

El proyecto se desarrollará siguiendo un enfoque incremental y orientado a la mantenibilidad, incorporando tecnologías apropiadas que aseguren una construcción eficiente, un despliegue escalable y la posibilidad de evolución continua del sistema.

Se empleará React.js para la interfaz de usuario, Django y Django REST framework para la lógica de servidor y la exposición de servicios API, MySQL como sistema gestor de bases de datos relacional, Redis para la gestión de solicitudes en tiempo real, así como la infraestructura en la nube de Microsoft Azure y los servicios de seguridad de Cloudflare, garantizando disponibilidad y protección.

Palabras clave:

- Accesibilidad
- Gestión hospitalaria
- Asignación automática de citas
- Programación lineal
- Rastreo en tiempo real

Abstract

This Final Degree Project (TFG) presents the development of a computer system aimed at the efficient management of hospital resources, optimizing and automating the scheduling of medical appointments, and enabling real-time tracking of patients.

The main objective is to develop an accessible, intuitive, and functional platform that optimizes the management of hospital resources, streamlines clinical processes, facilitates appointment scheduling —using binary integer linear programming algorithms— and reduces waiting times, while also enabling efficient patient follow-up.

For its development, a preliminary study of the hospital environment will be conducted, identifying the main bottlenecks and opportunities for improvement through digital solutions, in order to guide the application design towards greater efficiency and added value.

The project will be developed following an incremental and maintainable approach, incorporating appropriate technologies to ensure efficient construction, scalable deployment, and the possibility of continuous system evolution.

React.js will be used for the user interface, Django and Django REST framework for server logic and API service exposure, MySQL as relational database management system, Redis for real-time request management, as well as Microsoft Azure cloud infrastructure and Cloudflare security services, ensuring availability and protection.

Keywords:

- Accessibility
- Hospital management
- Automated appointment scheduling
- Linear programming
- Real-time tracking

Índice de contenidos

Índice de Figuras

Índice de Tablas

Capítulo 1: Introducción

La creciente complejidad en la gestión de recursos hospitalarios y la necesidad de mejorar la eficiencia en la atención médica contrastan con el uso de sistemas informáticos anticuados, con interfaces poco intuitivas y con dependencia de plataformas específicas (Kuperman, 2003)

En respuesta a esta problemática, surge *FindMyPatient*, una aplicación web moderna y accesible que facilita la asignación inteligente de citas, el rastreo en tiempo real de pacientes y una gestión más eficiente de los recursos clínicos. Todo esto mediante una experiencia de usuario muy cuidada y eficiente.

A lo largo de este Trabajo Fin de Grado (TFG) se documenta el ciclo de vida completo del desarrollo de *FindMyPatient*. El trabajo abarca desde el análisis de requisitos hasta el diseño, implementación, pruebas y despliegue del sistema, siguiendo un enfoque iterativo e incremental.

El proyecto comienza con un estudio del estado del arte, que permite contextualizar el problema y justificar la necesidad de la solución propuesta. A partir de este análisis se define la idea de negocio, así como sus objetivos funcionales.

A continuación, se describen las tecnologías, lenguajes y *frameworks* utilizados para el desarrollo de la aplicación, destacando aquellas herramientas que han contribuido a mejorar la eficiencia, seguridad y escalabilidad del sistema.

Posteriormente, se detalla el proceso de desarrollo, incluyendo las decisiones de diseño, implementación progresiva, validaciones a través de pruebas continuas y el despliegue final de la aplicación.

Finalmente se presentan las conclusiones generales del proyecto, identificando los logros alcanzados, los principales desafíos técnicos y organizativos, y las lecciones aprendidas. También se proponen posibles líneas de mejora para futuras versiones del sistema.

1.1. Justificación

El aumento en la sofisticación en la gestión de recursos hospitalarios, sumado a la presión por brindar una atención médica eficiente y de calidad a un número cada vez mayor de pacientes, ha puesto en evidencia las limitaciones de los sistemas clínicos tradicionales.

Muchas de estas plataformas presentan interfaces anticuadas y poco intuitivas, dependen de tecnologías obsoletas que carecen de soporte y mantenimiento, y realizan tareas esenciales de forma muy primitiva. Esto dificulta la adaptación a las cambiantes demandas del entorno hospitalario actual.

Ante este panorama, se hace evidente la necesidad de una aplicación moderna que centralice y optimice todas estas tareas, proporcionando una solución intuitiva y accesible desde cualquier dispositivo, capaz de adaptarse a los diferentes perfiles de usuario dentro del hospital: desde el personal administrativo y los profesionales clínicos hasta los propios

pacientes. Es en este contexto donde nace este proyecto, aquí es donde surge *FindMyPatient*.

1.2. Objetivos

Antes de abordar el estado del arte y la posterior implementación, es fundamental definir el objetivo principal y los objetivos secundarios de este proyecto.

1.2.1. Objetivo principal

El objetivo principal de este TFG es crear un sistema de gestión hospitalaria moderno e inteligente, con una experiencia de usuario cuidada y eficiente, que facilite la administración de recursos, optimice la asignación de citas y permita el rastreo en tiempo real de pacientes que lo requieran.

1.2.2. Objetivos secundarios

Entre los objetivos secundarios se encuentran:

1. Realizar un análisis de los sistemas informáticos hospitalarios existentes en la actualidad.

2. Definir los requisitos y funcionalidades clave de la aplicación web, incluyendo la selección de *frameworks*, lenguajes de programación, herramientas de maquetación y sistemas de gestión de bases de datos, entre otros elementos técnicos relevantes.

3. Diseñar una arquitectura flexible y escalable, que permita futuras actualizaciones, mejoras y un mantenimiento eficiente.

4. Documentar todo el proceso de desarrollo, proporcionando una guía clara y estructurada sobre los componentes del sistema.

Capítulo 2: Marco teórico y tecnológico

2.1. Estado del Arte

La digitalización de la gestión hospitalaria ha experimentado un desarrollo acelerado en las dos últimas décadas, impulsada por la necesidad de optimizar recursos, reducir costes y mejorar la calidad asistencial. Sin embargo, persisten importantes retos en la integración de sistemas, la automatización de procesos y la capacidad de adaptación a contextos cambiantes (Organización Mundial de la Salud, 2021)

Los Sistemas de Información Hospitalaria (HIS, por sus siglas en inglés) constituyen la columna vertebral de la informatización en entornos sanitarios. Permiten la administración integrada de pacientes, recursos, agendas, historiales clínicos y facturación (Haux, 2006)

Soluciones ampliamente implantadas como Cerner Millennium a nivel internacional (Cerner Corporation, 2025), SAP Health & Meditech en EE. UU. y Europa (SAP Health, 2025), o Selene en el contexto español (Siemens Healthineers, 2025) han demostrado ser herramientas robustas para este propósito.

A pesar de su robustez y madurez, estos sistemas presentan limitaciones en cuanto a flexibilidad, experiencia de usuario y, especialmente, en la automatización avanzada de procesos como la asignación dinámica de citas o el rastreo en tiempo real de pacientes (Romero-Torres, 2021)

Por otro lado, en relación con la gestión de citas médicas, la mayoría de los sistemas actuales se basan en modelos secuenciales o asignación manual, lo que puede dar lugar a ineficiencias y largos tiempos de espera (Cardoen, 2010)

En los últimos años, la literatura científica ha puesto de manifiesto el potencial de la algoritmia para solucionar este problema (Gupta, 2008). Sin embargo, aún no se ha generalizado en la práctica hospitalaria, limitándose en muchas ocasiones a salidas comerciales que automatizan reservas, pero no optimizan recursos de forma inteligente (Zocdoc, 2025)

Finalmente, el rastreo en tiempo real de pacientes es una funcionalidad cada vez más demandada, especialmente en servicios críticos y quirúrgicos, aunque la implementación de estas tecnologías y los retos de integración siguen siendo barreras notables (Diario Médico, 2019)

En consecuencia, resulta evidente la necesidad de una aplicación moderna, accesible, con una experiencia de usuario cuidada y centrada en la optimización de recursos, la inteligencia en la asignación de citas y el rastreo eficiente de pacientes, como la propuesta en este TFG.

2.2. Identidad de Marca

Al tratarse de una aplicación orientada al entorno sanitario, es fundamental proyectar una identidad de marca clara, profesional y tecnológicamente avanzada. Para ello, se ha optado por un diseño minimalista y funcional, que transmita confianza, innovación y eficiencia desde el primer contacto visual.

El color predominante es el azul, una elección intencionada por su fuerte asociación con la salud, la tecnología y la seguridad (LIDERLOGO, 2025). El azul transmite serenidad y responsabilidad, características esenciales en el ámbito hospitalario, donde la precisión y la fiabilidad son clave.

El logotipo, por su parte, fusiona elementos simbólicos fácilmente reconocibles: una silueta que recuerda a una figura médica acompañada de un icono similar al de geolocalización. Esto comunica de forma visual e intuitiva el propósito de la aplicación: el rastreo en tiempo real de pacientes, así como la gestión eficiente de los recursos hospitalarios y la optimización en la gestión de citas.

Figura 1. Logotipo de la aplicación.

2.3. Tecnologías utilizadas

En esta sección se detallan los lenguajes, *frameworks* y herramientas empleadas en el desarrollo de la aplicación web. Se incluyen tanto tecnologías esenciales para el funcionamiento del sistema como aquellas que han servido de apoyo en la gestión y el control de versiones. Cada herramienta es analizada en función de su funcionalidad, utilidad y contribución al proyecto.

2.3.1. Herramientas

Durante el desarrollo de la aplicación web, se han empleado diversas herramientas clave que han facilitado y optimizado el proceso. A continuación, se describen brevemente cada una de ellas y su función principal en el contexto del proyecto.

GitHub

GitHub es una plataforma online que permite almacenar y gestionar proyectos de software utilizando Git, tanto en modalidad pública como privada. Esta herramienta fomenta el trabajo en equipo y facilita la colaboración entre desarrolladores de todo el mundo, permitiendo compartir y mejorar proyectos de manera sencilla (GitHub, 2025)

A lo largo de la carrera, esta plataforma ha sido utilizada frecuentemente, motivo por el cual se eligió para alojar el repositorio público del código fuente del software desarrollado.

Adicionalmente, GitHub permite implementar flujos de trabajo automatizados (workflows), lo que facilitó la integración y entrega continua de la aplicación web.

Enlace al repositorio de GitHub de este proyecto.

Trello

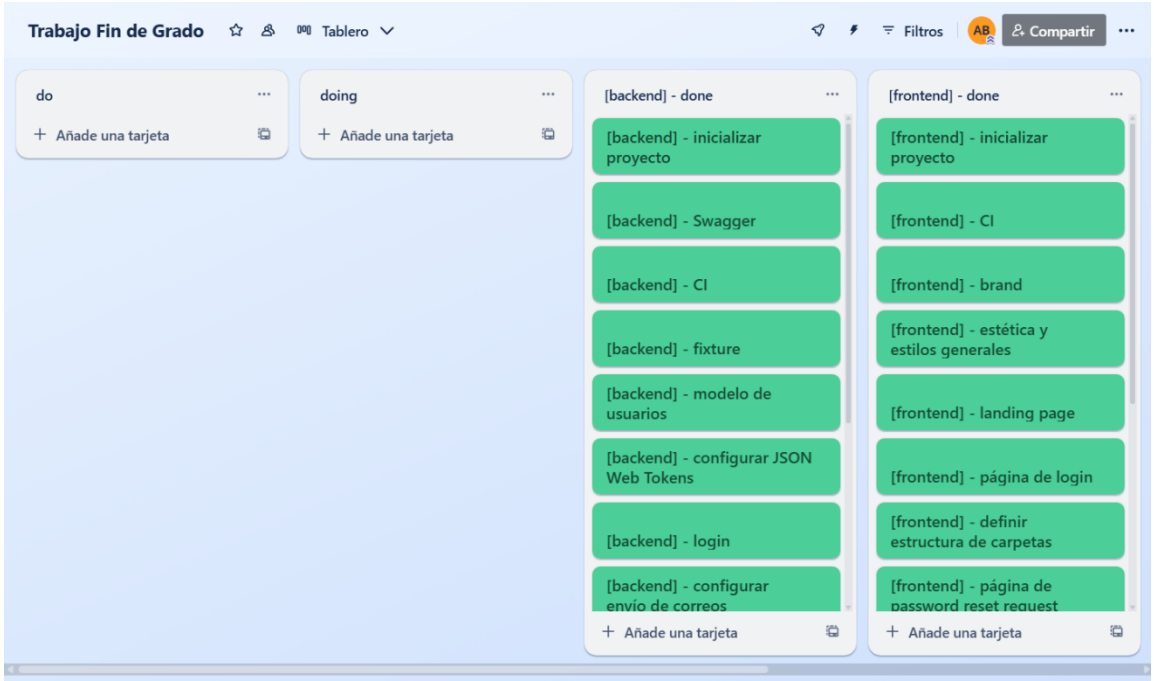

Figura 2. Captura de pantalla del tablero de Trello empleado.

Trello es una aplicación de gestión de proyectos basada en Kanban. Usa tableros, listas y tarjetas para organizar tareas, facilitando el seguimiento del progreso y la colaboración. Ayuda a visualizar actividades, controlar la carga de trabajo y optimizar procesos (Atlassian, 2025)

En este TFG se ha utilizado Trello para planificar y organizar el trabajo a lo largo de los diferentes incrementos del producto, con columnas como “do”, “doing”, “[backend] - done”, “[frontend] - done”, entre otras.

Visual Studio Code

Visual Studio Code es un editor de código ampliamente reconocido por su versatilidad y alto nivel de personalización, contando con un extenso ecosistema de extensiones que facilitan la gestión de proyectos de todo tipo (Microsoft, 2025)

Su compatibilidad con múltiples lenguajes de programación y la disponibilidad de numerosas extensiones lo convierten en una opción idónea para este proyecto.

PyCharm

PyCharm es un entorno de desarrollo integrado (IDE) especialmente diseñado para Python, valorado por su potencia y por las múltiples herramientas que ofrece para facilitar la programación en este lenguaje (Jetbrains, 2025).

Destaca por sus funciones avanzadas de autocompletado, refactorización y depuración, lo que resulta especialmente útil en tareas relacionadas con Python, facilitando el proceso de depuración frente a otros editores.

Docker

Docker es una plataforma que permite crear, desplegar y gestionar aplicaciones mediante contenedores, lo que facilita la portabilidad y consistencia entre distintos entornos (Docker, 2025)

Durante el desarrollo, resultó útil para lanzar y gestionar bases de datos de forma rápida y aislada, así como para generar imágenes de despliegue del sistema.

MySQL

MySQL es un sistema de gestión de bases de datos relacional ampliamente utilizado en el desarrollo de sistemas informáticos (Oracle, 2025)

Por su fiabilidad y rendimiento, se seleccionó para almacenar la información estructurada del proyecto. Su capacidad para gestionar relaciones complejas resulta especialmente adecuada en el ámbito hospitalario, donde las entidades están altamente interconectadas.

phpMyAdmin

phpMyAdmin es una herramienta web que facilita la administración de bases de datos MySQL mediante una interfaz gráfica intuitiva (phpMyAdmin, 2025). Facilitó gestionar y supervisar las bases de datos del sistema forma ágil y sencilla.

Redis

Redis es una base de datos en memoria orientada a clave-valor, destacada por su rapidez en el manejo de datos en tiempo real (Redis, 2025). En este TFG, se empleó para gestionar y consultar rápidamente la ubicación de los pacientes, facilitando su seguimiento ágil.

OpenStreetMap

OpenStreetMap es un proyecto colaborativo que proporciona mapas libres y editables a nivel global (OpenStreetMap, 2025). Su integración permitió mostrar en el cliente la ubicación en tiempo real de los pacientes, facilitando la visualización geográfica precisa y accesible de los datos.

Mailtrap

Mailtrap es un servicio diseñado para simular un servidor de correo electrónico en entornos de desarrollo y pruebas (Mailtrap, 2025). Gracias a esta herramienta, fue posible enviar, revisar y depurar correos electrónicos generados por la aplicación sin riesgo de que lleguen a destinatarios reales, facilitando así el desarrollo y la verificación de funcionalidades relacionadas con el envío de emails.

Gmail SMTP

Gmail SMTP es el servicio de correo saliente proporcionado por Google, utilizado para enviar emails desde aplicaciones y servicios de manera segura (Google, 2025). Así, se posibilitó el envío de mensajes a través de una infraestructura real.

PlantUML

PlantUML es una herramienta que permite crear diagramas de forma sencilla mediante un "código" (PlantUML, 2024). Se utilizó para diseñar diagramas de arquitectura y relaciones.

draw.io

draw.io es una herramienta online que permite crear cualquier tipo de diagrama de manera visual e intuitiva (draw.io, 2025). Resultó especialmente útil para diseñar diagramas de flujo y navegación.

ESLint

ESLint es un analizador estático para código JavaScript y TypeScript (OpenJS, 2025). Sirve para asegurar la calidad y consistencia del código del cliente, facilitando la detección temprana de errores y la aplicación de buenas prácticas.

Black

Black es un formateador automático de código para Python que aplica convenciones de estilo de forma estricta y uniforme (PSF, 2025). Se recurre a él para mantener el código Python limpio y estandarizado durante el proceso de desarrollo.

Microsoft Azure

Microsoft Azure es la plataforma en la nube de Microsoft que ofrece servicios de cómputo, bases de datos, redes y almacenamiento bajo un modelo *as-a-service* (Microsoft, 2025)

En este TFG se empleó para desplegar en producción los contenedores generados con Docker y alojar las bases de datos MySQL y Redis.

Name.com

Name.com es un registrador de dominios y proveedor de DNS que facilita la administración de nombres de dominio y certificados SSL (Name.com, 2025)

Se utilizó para registrar el dominio público del proyecto y gestionar los registros DNS necesarios, asegurando una resolución fiable del nombre de la aplicación en el entorno de producción.

Cloudflare

Cloudflare es una red de distribución de contenidos (CDN) y plataforma de seguridad que actúa como proxy inverso, ofreciendo caché estática, protección frente a ataques DDoS, gestión de certificados TLS y soporte integral para conexiones seguras HTTPS (Cloudflare, 2025)

Su integración permitió reducir la latencia global y proteger la aplicación contra amenazas comunes, garantizando además el acceso cifrado y seguro a través de HTTPS sin necesidad de modificar la infraestructura subyacente.

2.3.2. Lenguajes

En este apartado se describen los lenguajes y tecnologías fundamentales utilizados en el desarrollo de la aplicación web, incluyendo tanto lenguajes de programación como de estilos.

Python

Python es un lenguaje de programación versátil y de alto nivel, ampliamente utilizado en el desarrollo de aplicaciones web, análisis de datos y automatización (PSF, 2025).

En este proyecto, constituye la base de la lógica del servidor, al ofrecer eficiencia en la manipulación de datos y una amplia variedad de bibliotecas especializadas.

JavaScript

JavaScript es el principal lenguaje de programación para el desarrollo del lado del cliente en aplicaciones web (ECMA, 2025). Su uso dota de interactividad y dinamismo a la interfaz, facilitando la comunicación con el servidor y la actualización de contenidos en tiempo real.

CSS

CSS es el lenguaje estándar para definir la presentación y el diseño visual de páginas web (W3C, 2025). Proporciona estilo y coherencia a la interfaz y adaptar la aplicación a distintos dispositivos y resoluciones.

2.3.3. Frameworks y bibliotecas usadas

En este apartado se describen los *frameworks* y las bibliotecas utilizadas durante el desarrollo del software. Para este TFG, un *framework* se entiende como un conjunto de bibliotecas y estructuras que ofrecen una base sólida para crear aplicaciones de manera segura y eficiente.

Django

Django es un *framework* web de alto nivel para Python que fomenta el desarrollo rápido y limpio (DSF, 2025)

Aunque la curva de aprendizaje inicial fue exigente, permitió ahorrar mucho tiempo gracias a sus numerosas funcionalidades integradas y la escasa necesidad de escribir código repetitivo.

Django REST Framework

Django REST Framework es una potente extensión de Django para construir APIs RESTful de manera rápida, flexible y segura (DSF, 2025). Su integración facilitó la creación de endpoints robustos y personalizables, simplificando la gestión de serialización de datos, autenticación y permisos.

PuLP

PuLP es una biblioteca de Python orientada a la resolución de problemas de optimización lineal (PSF, 2025).

Fue esencial para implementar el algoritmo de asignación inteligente de citas, minimizando el tiempo de espera de los pacientes.

Se optó por ella debido a su sintaxis sencilla, integración con Python y no requerir instalar software adicional externo.

React.js

React.js es una biblioteca de JavaScript para la construcción de interfaces de usuario interactivas y escalables (Meta, 2025). Facilitó el desarrollo de un cliente dinámico y eficiente, asegurando una experiencia fluida para el usuario.

Bootstrap

Bootstrap es un *framework* de código abierto para el diseño responsivo y la maquetación de interfaces web (Bootstrap, 2025), con una base de componentes predefinidos personalizables.

Su uso aceleró el desarrollo visual, permitiendo adaptar y personalizar componentes para dotar a la interfaz de una identidad propia y garantizar compatibilidad con distintos dispositivos.

Django Test Runner

Django Test Runner es el sistema de ejecución de pruebas integrado en Django, que facilita la validación automática del comportamiento del sistema (DSF, 2025)

Se empleó tanto para pruebas de integración como para algunos tests unitarios del servidor, contribuyendo a comprobar la robustez y fiabilidad del sistema a lo largo del desarrollo.

Vitest

Vitest es un *framework* de testing moderno y rápido para aplicaciones JavaScript y TypeScript (Vitest, 2025). Se realizaron pruebas unitarias y de componentes en el cliente, posibilitando la detección temprana de errores y asegurando la calidad en la interfaz.

Capítulo 3: Ciclo de vida de la aplicación

En este capítulo se examinará el ciclo de vida de la aplicación, abarcando desde su concepción inicial hasta su despliegue en producción. Se presentará las metodologías empleadas durante el desarrollo y se detallarán los artefactos generados a lo largo de todo el proceso, tales como el Análisis de Requisitos, el Diseño, la Implementación, las Pruebas y el Despliegue.

3.1. Metodologías empleadas

Para el desarrollo de la aplicación se adoptó una metodología iterativa e incremental, concretamente basada en los principios de Scrum (Sommerville, 2020).

El proceso se organizaba en ciclos de aproximadamente mes y medio, al final de los cuales se organizaban reuniones para presentar y entregar un nuevo incremento del producto. Durante estas reuniones, se identificaban nuevas necesidades surgidas del uso del incremento y se definían funcionalidades adicionales para el siguiente ciclo de desarrollo.

La gestión de tareas y el seguimiento del avance se realizaban mediante Trello, lo que permitía visualizar en todo momento el estado de cada funcionalidad o tarea.

En cuanto al control de versiones, se seguía la estrategia de ramas de Gitflow (Atlassian, 2025). Las nuevas funcionalidades se desarrollaban en ramas independientes, que se integraban mediante Pull Requests. Para los mensajes de commit, aplicábamos la convención Conventional Commits (Conventional Commits, 2019), facilitando la trazabilidad y claridad del historial de cambios.

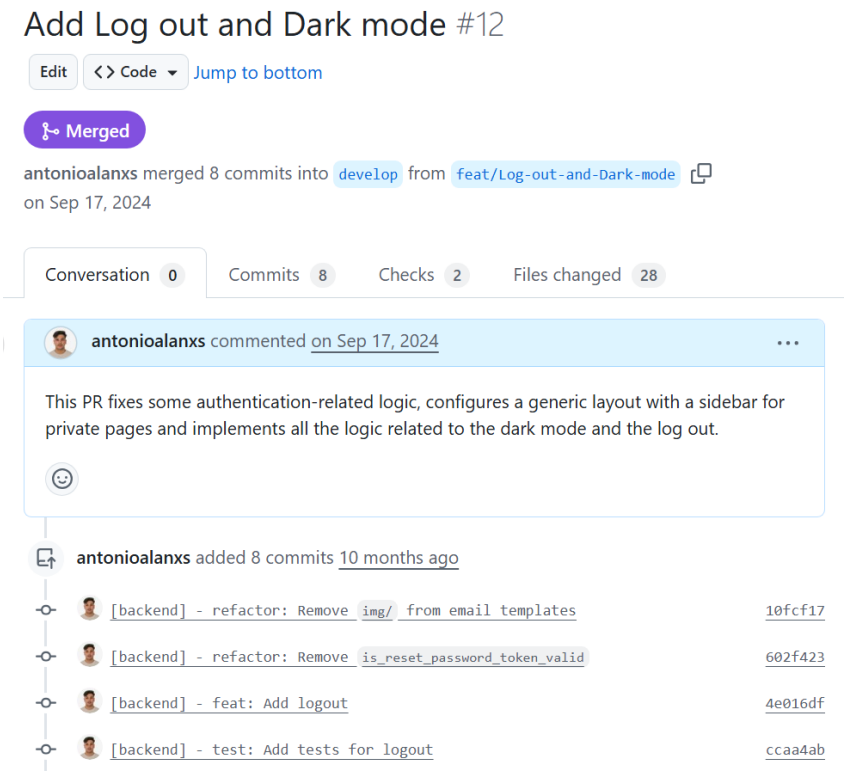

Figura 3. Captura de pantalla de la estrategia seguida en GitHub.

3.2. Análisis de requisitos

En esta sección se describen los requisitos identificados durante el desarrollo de la aplicación, incluyendo los requisitos funcionales, los atributos de calidad y los tipos de usuario. Todos estos aspectos se presentan de forma global, abarcando las distintas versiones de la aplicación.

3.2.1. Tipos de Usuario

En la aplicación existen cuatro tipos de usuario:

Tipo de Usuario	Descripción
Anónimo	Usuario no registrado en la aplicación, sin acceso a la mayoría de las funcionalidades.
Paciente	Usuario registrado con permisos orientados al uso de las funcionalidades destinadas a pacientes.
Doctor	Usuario registrado con acceso a las funcionalidades propias del rol médico.
Administrador	Usuario registrado, con control total sobre la plataforma y todas sus entidades, que dispone del conjunto de permisos más amplio y completo.

Tabla 1. Tipos de Usuario.

3.2.2. Requisitos funcionales

En esta sección se presentan los requisitos funcionales que debe cumplir el sistema, organizados según los distintos tipos de usuario. La siguiente tabla ofrece una visión general de qué funcionalidades están disponibles para cada perfil.

	Anónimo	Paciente	Doctor	Administrador
Recibir email de registro en el sistema	✓			
Olvido de contraseña		✓	✓	✓
Cambiar contraseña		✓	✓	✓
Iniciar sesión		✓	✓	✓
Cerrar sesión		✓	✓	✓
Editar perfil		✓	✓	✓
Compartir ubicación en vivo		✓		
Visualizar ubicación en vivo			✓	✓
Solicitar cita médica		✓	✓	✓
Recibir email de confirmación de cita médica		✓		
Consultar citas médicas		✓	✓	✓
Consultar cita médica			✓	✓
Consultar calendario de citas médicas			✓	
Cambiar estado de cita			✓	✓

Añadir observaciones a cita			✓	✓
Registrar paciente			✓	✓
Consultar pacientes			✓	✓
Visualizar ficha de paciente			✓	✓
Editar ficha de paciente			✓	✓
Eliminar paciente			✓	✓
Consultar tratamientos médicos		✓	✓	✓
Mandar tratamiento médico			✓	
Consultar pruebas médicas		✓	✓	✓
Mandar prueba médica			✓	
Registrar doctor			✓	✓
Consultar doctores				✓
Visualizar ficha de doctor				✓
Editar ficha de doctor				✓
Eliminar doctor				✓
Crear especialidad médica				✓
Consultar especialidades médicas				✓
Visualizar ficha de especialidad médica				✓
Editar ficha de especialidad médica				✓
Eliminar especialidad médica				✓
Registrar sala				✓
Consultar salas				✓
Visualizar ficha de sala				✓
Editar ficha de sala				✓
Eliminar sala				✓
Registrar administrador				✓
Consultar administradores				✓
Visualizar administrador				✓
Editar administrador				✓
Eliminar administrador				✓
Crear grupo de permisos				✓
Consultar grupos de permisos				✓
Consultar grupo de permisos				✓
Editar grupo de permisos				✓
Eliminar grupo de permisos				✓
Exportar base de datos			✓	✓

Tabla 2. Requisitos funcionales generales.

Con base en la información resumida en la tabla, se procede al desglose y descripción detallada de cada uno de los requisitos funcionales del sistema.

3.2.2.1. Usuario Anónimo

RF-UAN-001 - Recibir email de registro en el sistema

Nombre	Recibir email de registro en el sistema
Identificador	RF-UAN-001
Descripción	Un usuario anónimo recibirá un correo electrónico con sus credenciales de acceso cuando sea registrado en el sistema por un administrador o un doctor.

Tabla 3. RF-UAN-001 - Recibir email de registro en el sistema.

3.2.2.2. Usuario Registrado

RF-UR-001 - Olvido de contraseña

Nombre	Olvido de contraseña
Identificador	RF-UR-001
Descripción	Un usuario registrado que haya olvidado su contraseña podrá solicitar el envío de un correo electrónico con un enlace para restablecerla, el cual lo redirigirá a una página donde podrá establecer una nueva contraseña.

Tabla 4. RF-UR-001 - Olvido de contraseña.

RF-UR-002 - Iniciar sesión

Nombre	Iniciar sesión
Identificador	RF-UR-002
Descripción	Un usuario registrado podrá iniciar sesión en la plataforma mediante sus credenciales (NIF y contraseña).

Tabla 5. RF-UR-002 - Iniciar sesión.

RF-UR-003 - Editar información personal

Nombre	Editar información personal
Identificador	RF-UR-003
Descripción	Un usuario registrado podrá editar y actualizar su información personal de la plataforma, asegurando que sus datos estén siempre correctos.

Tabla 6. RF-UR-003 - Editar información personal.

RF-UR-004 - Editar información relacionada con rol de usuario

Nombre	Editar información relacionada con rol de usuario
Identificador	RF-UR-004
Descripción	Un usuario registrado podrá editar y actualizar la información relacionada con su rol de la plataforma, asegurando que estos datos estén siempre correctos.

Tabla 7. RF-UR-004 - Editar información relacionada con rol de usuario.

RF-UR-005 - Cambiar contraseña

Nombre	Cambiar contraseña
Identificador	RF-UR-005
Descripción	Un usuario registrado podrá cambiar su contraseña desde la plataforma una vez haya iniciado sesión. Para completar el cambio, deberá ingresar su contraseña actual; si es correcta, podrá establecer una nueva, de lo contrario no se permitirá el cambio.

Tabla 8. RF-UR-005 - Cambiar contraseña.

RF-UR-006 - Cerrar sesión

Nombre	Cerrar sesión
Identificador	RF-UR-006
Descripción	Un usuario registrado podrá cerrar sesión en la plataforma en cualquier momento para finalizar su acceso y proteger su información.

Tabla 9. RF-UR-006 - Cerrar sesión.

3.2.2.3. Paciente

RF-URP-001 - Compartir ubicación en tiempo real

Nombre	Compartir ubicación en tiempo real
Identificador	RF-URP-001
Descripción	Un paciente podrá compartir su ubicación en tiempo real desde la plataforma. Para ello, se le solicitarán los permisos de localización necesarios y, una vez concedidos, su ubicación podrá ser consultada por el personal médico autorizado.

Tabla 10. RF-URP-001 - Compartir ubicación en tiempo real.

RF-URP-002 - Solicitar cita médica

Nombre	Solicitar cita médica
Identificador	RF-URP-002
Descripción	Un paciente podrá solicitar una cita médica con su doctor de cabecera a través de la plataforma. A partir de ahí, será un doctor o un administrador quien lo derive a un especialista.

Tabla 11. RF-URP-002 - Solicitar cita médica.

RF-URP-003 - Recibir email de confirmación de cita médica

Nombre	Recibir email de confirmación de cita médica
Identificador	RF-URP-003
Descripción	Un paciente recibirá un correo electrónico de confirmación al agendar una cita médica con los detalles de la misma.

Tabla 12. RF-URP-003 - Recibir email de confirmación de cita médica.

RF-URP-004 - Consultar mis citas médicas

Nombre	Consultar mis citas médicas
Identificador	RF-URP-004
Descripción	Un paciente podrá ver el listado de todas sus citas médicas registradas en la plataforma.

Tabla 13. RF-URP-004 - Consultar mis citas médicas.

RF-URP-005 - Consultar mis tratamientos médicos

Nombre	Consultar mis tratamientos médicos
Identificador	RF-URP-005
Descripción	Un paciente podrá consultar el listado de todos sus tratamientos médicos registrados en la plataforma.

Tabla 14. RF-URP-005 - Consultar mis tratamientos médicos.

RF-URP-006 - Consultar mis pruebas médicas

Nombre	Consultar mis pruebas médicas
Identificador	RF-URP-006
Descripción	Un paciente podrá consultar el listado de todas sus pruebas médicas registradas en la plataforma.

Tabla 15. RF-URP-006 - Consultar mis pruebas médicas.

3.2.2.4. Doctor

RF-URD-001 - Solicitar cita médica con especialidad para paciente asignado

Nombre	Solicitar cita médica con especialidad para paciente asignado
Identificador	RF-URD-001
Descripción	Un doctor podrá derivar a su paciente a una cita médica con especialidad.

Tabla 16. RF-URD-001 - Solicitar cita médica con especialidad para paciente asignado.

RF-URD-002 - Consultar citas médicas asignadas

Nombre	Consultar citas médicas asignadas
Identificador	RF-URD-002
Descripción	Un doctor podrá ver el listado de todas las citas médicas en las que está involucrado y que están registradas en la plataforma.

Tabla 17. RF-URD-002 - Consultar citas médicas asignadas.

RF-URD-003 - Consultar calendario

Nombre	Consultar calendario
Identificador	RF-URD-003
Descripción	Un doctor podrá ver su calendario con las citas médicas programadas y otros eventos relevantes, directamente desde la plataforma.

Tabla 18. RF-URD-003 - Consultar calendario.

RF-URD-004 - Cambiar el estado de una cita asignada

Nombre	Cambiar el estado de una cita asignada
Identificador	RF-URD-004
Descripción	Un doctor podrá cambiar el estado de una cita médica que tenga asignada, actualizándola según corresponda (por ejemplo: programada, en progreso, cancelada)

Tabla 19. RF-URD-004 - Cambiar el estado de una cita asignada.

RF-URD-005 - Añadir observaciones a una cita asignada

Nombre	Añadir observaciones a una cita asignada
Identificador	RF-URD-005
Descripción	Un doctor podrá añadir observaciones o notas a una cita médica que haya atendido, registrando información relevante sobre la consulta directamente en la plataforma.

Tabla 20. RF-URD-005 - Añadir observaciones a una cita asignada.

RF-URD-006 - Consultar pacientes asignados

Nombre	Consultar pacientes asignados
Identificador	RF-URD-006
Descripción	Un doctor podrá ver un listado de todos los pacientes que tiene asignados en la plataforma.

Tabla 21. RF-URD-006 - Consultar pacientes asignados.

RF-URD-007 - Registrar paciente

Nombre	Registrar paciente
Identificador	RF-URD-007
Descripción	Un doctor podrá registrar nuevos pacientes en la plataforma, ingresando todos sus datos requeridos y asignándolos a sí mismo o a otro doctor disponible.

Tabla 22. RF-URD-007 - Registrar paciente.

RF-URD-008 - Eliminar paciente asignado

Nombre	Eliminar paciente asignado
Identificador	RF-URD-008
Descripción	Un doctor podrá eliminar de la plataforma a un paciente que tenga asignado.

Tabla 23. RF-URD-008 - Eliminar paciente asignado.

RF-URD-009 - Visualizar ubicación en tiempo real de paciente asignado

Nombre	Visualizar ubicación en tiempo real de paciente asignado
Identificador	RF-URD-009
Descripción	Un doctor podrá visualizar en la plataforma la ubicación en tiempo real de los pacientes que tiene asignados, siempre y cuando el paciente haya otorgado su consentimiento para compartir dicha información.

Tabla 24. RF-URD-009 - Visualizar ubicación en tiempo real de paciente asignado.

RF-URD-010 - Visualizar ficha de paciente asignado

Nombre	Visualizar ficha de paciente asignado
Identificador	RF-URD-010
Descripción	Un doctor podrá visualizar en detalle la ficha de un paciente que tiene asignado, la cual contiene su información personal, dirección, tratamientos médicos y resultados de pruebas médicas registrados en la plataforma.

Tabla 25. RF-URD-010 - Visualizar ficha de paciente asignado.

RF-URD-011 - Editar ficha de paciente asignado

Nombre	Editar ficha de paciente asignado
Identificador	RF-URD-011
Descripción	Un doctor podrá editar y actualizar la información de un paciente que tenga asignado en la plataforma, asegurando que estos datos estén siempre correctos.

Tabla 26. RF-URD-011 - Editar ficha de paciente asignado.

RF-URD-012 - Mandar tratamiento médico a paciente

Nombre	Mandar tratamiento médico a paciente
Identificador	RF-URD-012
Descripción	Un doctor podrá enviar tratamientos médicos a los pacientes que atienda en sus citas, registrando la información correspondiente en la plataforma para que el paciente pueda consultarla posteriormente.

Tabla 27. RF-URD-012 - Mandar tratamiento médico a paciente.

RF-URD-013 - Mandar prueba médica a paciente

Nombre	Mandar prueba médica a paciente
Identificador	RF-URD-013
Descripción	Un doctor podrá enviar pruebas médicas a los pacientes que atienda en sus citas, registrando la información correspondiente en la plataforma para que el paciente pueda consultarla posteriormente.

Tabla 28. RF-URD-013 - Mandar prueba médica a paciente.

RF-URD-014 - Exportar base de datos

Nombre	Exportar base de datos
Identificador	RF-URD-014
Descripción	Un doctor podrá exportar la información de la base de datos de la plataforma relacionada únicamente con su actividad y los pacientes que tiene asignados, con el fin de realizar estadística o investigación.

Tabla 29. RF-URD-014 - Exportar base de datos.

3.2.2.5. Administrador

RF-URAD-001 - Solicitar cita médica con especialidad para cualquier paciente

Nombre	Solicitar cita médica con especialidad para cualquier paciente
Identificador	RF-URAD-001
Descripción	Un administrador podrá derivar a cualquier paciente a una cita médica con especialidad.

Tabla 30. RF-URAD-001 - Solicitar cita médica con especialidad para cualquier paciente.

RF-URAD-002 - Consultar todas las citas del sistema

Nombre	Consultar todas las citas del sistema
Identificador	RF-URAD-002
Descripción	Un administrador podrá ver el listado de todas las citas médicas registradas en la plataforma.

Tabla 31. RF-URAD-002 - Consultar todas las citas del sistema.

RF-URAD-003 - Cambiar el estado de cualquier cita

Nombre	Cambiar el estado de cualquier cita
Identificador	RF-URAD-003
Descripción	Un administrador podrá cambiar el estado de cualquier cita médica del sistema, actualizándola según corresponda (por ejemplo: programada, en progreso, cancelada)

Tabla 32. RF-URAD-003 - Cambiar el estado de cualquier cita.

RF-URAD-004 - Añadir observaciones a cualquier cita

Nombre	Añadir observaciones a cualquier cita
Identificador	RF-URAD-004
Descripción	Un administrador podrá añadir observaciones o notas a cualquier cita médica del sistema, registrando información relevante sobre la consulta directamente en la plataforma.

Tabla 33. RF-URAD-004 - Añadir observaciones a cualquier cita.

RF-URAD-005 - Consultar grupos de permisos

Nombre	Consultar grupos de permisos
Identificador	RF-URAD-005
Descripción	Un administrador podrá ver el listado de todos los grupos de permisos del sistema.

Tabla 34. RF-URAD-005 - Consultar grupos de permisos.

RF-URAD-006 - Crear grupo de permisos

Nombre	Crear grupo de permisos
Identificador	RF-URAD-006
Descripción	Un administrador podrá registrar un nuevo grupo de permisos en la plataforma.

Tabla 35. RF-URAD-006 - Crear grupo de permisos.

RF-URAD-007 - Visualizar grupo de permisos

Nombre	Visualizar grupo de permisos
Identificador	RF-URAD-007
Descripción	Un administrador podrá visualizar en detalle un grupo de permisos.

Tabla 36. RF-URAD-007 - Visualizar grupo de permisos.

RF-URAD-008 - Editar grupo de permisos

Nombre	Editar grupo de permisos
Identificador	RF-URAD-008
Descripción	Un administrador podrá editar y actualizar la información de un grupo de permisos de la plataforma, asegurando que estos datos estén siempre correctos.

Tabla 37. RF-URAD-008 - Editar grupo de permisos.

RF-URAD-009 - Eliminar grupo de permisos

Nombre	Eliminar grupo de permisos
Identificador	RF-URAD-009
Descripción	Un administrador podrá eliminar un grupo de permisos de la plataforma.

Tabla 38. RF-URAD-009 - Eliminar grupo de permisos.

RF-URAD-010 - Consultar administradores

Nombre	Consultar administradores
Identificador	RF-URAD-010
Descripción	Un administrador podrá ver un listado de todos los administradores registrados en la plataforma.

Tabla 39. RF-URAD-010 - Consultar administradores.

RF-URAD-011 - Registrar administrador

Nombre	Registrar administrador
Identificador	RF-URAD-011
Descripción	Un administrador podrá registrar nuevos administradores en la plataforma ingresando todos sus datos requeridos.

Tabla 40. RF-URAD-011 - Registrar administrador.

RF-URAD-012 - Visualizar administrador

Nombre	Visualizar administrador
Identificador	RF-URAD-012
Descripción	Un administrador podrá visualizar en detalle a otro.

Tabla 41. RF-URAD-012 - Visualizar administrador.

RF-URAD-013 - Editar administrador

Nombre	Editar administrador
Identificador	RF-URAD-013
Descripción	Un administrador podrá editar y actualizar la información de otro administrador de la plataforma, asegurando que estos datos estén siempre correctos.

Tabla 42. RF-URAD-013 - Editar administrador.

RF-URAD-014 - Eliminar administrador

Nombre	Eliminar administrador
Identificador	RF-URAD-014
Descripción	Un administrador podrá eliminar a otro de la plataforma.

Tabla 43. RF-URAD-014 - Eliminar administrador.

RF-URAD-015 - Consultar doctores

Nombre	Consultar administradores
Identificador	RF-URAD-015
Descripción	Un administrador podrá ver un listado de todos los doctores registrados en la plataforma.

Tabla 44. RF-URAD-015 - Consultar doctores.

RF-URAD-016 - Registrar doctor

Nombre	Registrar doctor
Identificador	RF-URAD-016
Descripción	Un administrador podrá registrar nuevos doctores en la plataforma ingresando todos sus datos requeridos.

Tabla 45. RF-URAD-016 - Registrar doctor.

RF-URAD-017 - Visualizar ficha de doctor

Nombre	Visualizar ficha de doctor
Identificador	RF-URAD-017
Descripción	Un administrador podrá visualizar en detalle la ficha de cualquier doctor del sistema, la cual contiene su información personal, de rol, y sus pacientes asignados.

Tabla 46. RF-URAD-017 - Visualizar ficha de doctor.

RF-URAD-018 - Editar ficha de doctor

Nombre	Editar doctor
Identificador	RF-URAD-018
Descripción	Un administrador podrá editar y actualizar la información de un doctor de la plataforma, asegurando que estos datos estén siempre correctos.

Tabla 47. RF-URAD-018 - Editar doctor.

RF-URAD-019 - Eliminar doctor

Nombre	Eliminar doctor
Identificador	RF-URAD-019
Descripción	Un administrador podrá eliminar a un doctor de la plataforma.

Tabla 48. RF-URAD-019 - Eliminar doctor.

RF-URAD-020 - Consultar todos los pacientes del sistema

Nombre	Consultar todos los pacientes del sistema
Identificador	RF-URAD-020
Descripción	Un administrador podrá ver un listado de todos los pacientes registrados en la plataforma.

Tabla 49. RF-URAD-020 - Consultar todos los pacientes del sistema.

RF-URAD-021 - Registrar paciente

Nombre	Registrar paciente
Identificador	RF-URAD-021
Descripción	Un administrador podrá registrar nuevos pacientes en la plataforma, ingresando todos sus datos requeridos y asignándolos a un doctor disponible.

Tabla 50. RF-URAD-021 - Registrar paciente.

RF-URAD-022 - Eliminar cualquier paciente

Nombre	Eliminar paciente
Identificador	RF-URAD-022
Descripción	Un administrador podrá eliminar de la plataforma a cualquier paciente.

Tabla 51. RF-URAD-022 - Eliminar cualquier paciente.

RF-URAD-023 - Visualizar ubicación en tiempo real de cualquier paciente

Nombre	Visualizar ubicación en tiempo real de cualquier paciente
Identificador	RF-URAD-023
Descripción	Un administrador podrá visualizar en la plataforma la ubicación en tiempo real de cualquier paciente del sistema, siempre y cuando el paciente haya otorgado su consentimiento para compartir dicha información.

Tabla 52. RF-URAD-023 - Visualizar ubicación en tiempo real de cualquier paciente.

RF-URAD-024 - Visualizar ficha de cualquier paciente

Nombre	Visualizar ficha de cualquier paciente
Identificador	RF-URAD-024
Descripción	Un administrador podrá visualizar en detalle la ficha de cualquier paciente del sistema, la cual contiene su información personal, dirección, tratamientos médicos y resultados de pruebas médicas registrados en la plataforma.

Tabla 53. RF-URAD-024 - Visualizar ficha de cualquier paciente.

RF-URAD-025 - Editar ficha de cualquier paciente

Nombre	Editar ficha de cualquier paciente
Identificador	RF-URAD-025
Descripción	Un administrador podrá editar y actualizar la información de cualquier paciente de la plataforma, asegurando que estos datos estén siempre correctos.

Tabla 54. RF-URAD-025 - Editar ficha de cualquier paciente.

RF-URAD-026 - Mandar tratamiento médico a paciente

Nombre	Mandar tratamiento médico a paciente
Identificador	RF-URAD-026
Descripción	Un administrador, con acceso a todas las citas registradas en el sistema, tendrá la facultad de mandar tratamientos médicos a los pacientes asociados a dichas citas, registrando la información correspondiente en la plataforma para que el paciente pueda consultarla posteriormente.

Tabla 55. RF-URAD-026 - Mandar tratamiento médico a paciente.

RF-URAD-027 - Mandar prueba médica a paciente

Nombre	Mandar prueba médica a paciente
Identificador	RF-URAD-027
Descripción	Un administrador, con acceso a todas las citas registradas en el sistema, tendrá la facultad de mandar pruebas médicas a los pacientes asociados a dichas citas, registrando la información correspondiente en la plataforma para que el paciente pueda consultarla posteriormente.

Tabla 56. RF-URAD-027 - Mandar prueba médica a paciente.

RF-URAD-028 - Consultar salas

Nombre	Consultar salas
Identificador	RF-URAD-028
Descripción	Un administrador podrá ver un listado de todas las salas del centro que están registradas en la plataforma.

Tabla 57. RF-URAD-028 - Consultar salas.

RF-URAD-029 - Registrar sala

Nombre	Registrar sala
Identificador	RF-URAD-029
Descripción	Un administrador podrá registrar nuevas salas pertenecientes al centro en la plataforma, ingresando toda la información requerida.

Tabla 58. RF-URAD-029 - Registrar sala.

RF-URAD-030 - Visualizar sala

Nombre	Visualizar sala
Identificador	RF-URAD-030
Descripción	Un administrador podrá visualizar en detalle la ficha de cualquier sala perteneciente al centro y registrada en la plataforma.

Tabla 59. RF-URAD-030 - Visualizar sala.

RF-URAD-031 - Editar sala

Nombre	Editar sala
Identificador	RF-URAD-031
Descripción	Un administrador podrá editar y actualizar la información de una sala perteneciente al centro y registrada en la plataforma, asegurando que estos datos estén siempre correctos.

Tabla 60. RF-URAD-031 - Editar sala.

RF-URAD-032 - Eliminar sala

Nombre	Eliminar sala
Identificador	RF-URAD-032
Descripción	Un administrador podrá eliminar una sala perteneciente al centro de la plataforma.

Tabla 61. RF-URAD-032 - Eliminar sala.

RF-URAD-031 - Consultar especialidades médicas

Nombre	Consultar especialidades médicas
Identificador	RF-URAD-031
Descripción	Un administrador podrá ver el listado de todas las especialidades médicas del sistema.

Tabla 62. RF-URAD-031 - Consultar especialidades médicas.

RF-URAD-032 - Registrar especialidad médica

Nombre	Registrar especialidad médica
Identificador	RF-URAD-032
Descripción	Un administrador podrá registrar una nueva especialidad médica en la plataforma.

Tabla 63. RF-URAD-032 - Registrar especialidad médica.

RF-URAD-033 - Visualizar ficha de especialidad médica

Nombre	Visualizar especialidad médica
Identificador	RF-URAD-033
Descripción	Un administrador podrá visualizar en detalle la ficha de una especialidad médica, la cual contiene la información que la define y los listados de doctores y salas asociadas a ella.

Tabla 64. RF-URAD-033 - Visualizar ficha de especialidad médica.

RF-URAD-034 - Editar especialidad médica

Nombre	Editar especialidad médica
Identificador	RF-URAD-034
Descripción	Un administrador podrá editar y actualizar la información de una especialidad médica de la plataforma, asegurando que estos datos estén siempre correctos.

Tabla 65. RF-URAD-034 - Editar especialidad médica.

RF-URAD-035 - Eliminar especialidad médica

Nombre	Eliminar especialidad médica
Identificador	RF-URD-035
Descripción	Un administrador podrá eliminar una especialidad médica de la plataforma.

Tabla 66. RF-URAD-035 - Eliminar especialidad médica.

3.2.3. Atributos de calidad

En esta sección se listarán y describirán el conjunto de atributos de calidad que ha de cumplir la aplicación web.

QA-001 - Soporte para la mayoría de navegadores web

Nombre	Soporte para la mayoría de navegadores web
Identificador	QA-001
Descripción	La aplicación debe ser desarrollada de manera que sea soportada por la gran mayoría de los navegadores web modernos.
Tipo	Obligatorio

Tabla 67. QA-001 - Soporte para la mayoría de navegadores web.

QA-002 - Diseño responsive

Nombre	Diseño responsive
Identificador	QA-002
Descripción	La aplicación debe adaptarse a cualquier tamaño de pantalla para poder ejecutarse correctamente en todo tipo de dispositivos, desde monitores grandes hasta teléfonos móviles.
Tipo	Obligatorio

Tabla 68. QA-002 - Diseño responsive.

QA-003 - Interfaz de usuario minimalista y user-friendly

Nombre	Interfaz de usuario minimalista y user-friendly
Identificador	QA-003
Descripción	La interfaz aplicación debe ser minimalista y fácil de usar, para asegurar una experiencia de usuario agradable y sencilla.
Tipo	Obligatorio

Tabla 69. QA-003 - Interfaz de usuario minimalista y user-friendly.

QA-004 - Navegación hacia atrás en todas las pantallas

Nombre	Navegación hacia atrás en todas las pantallas
Identificador	QA-004
Descripción	El usuario debe poder navegar hacia atrás desde cualquier pantalla de la aplicación, ya sea mediante un botón específico en la interfaz o utilizando las funcionalidades nativas del navegador o dispositivo, contribuyendo así a una experiencia de usuario fluida y satisfactoria.
Tipo	Obligatorio

Tabla 70. QA-004 - Navegación hacia atrás en todas las pantallas.

QA-005 - Accesibilidad y Usabilidad

Nombre	Accesibilidad y Usabilidad
Identificador	QA-005
Descripción	La aplicación debe seguir la mayoría de los principios de usabilidad y accesibilidad, para que el mayor número posible de usuarios pueda utilizar el software sin inconvenientes.
Tipo	Obligatorio

Tabla 71. QA-005 - Accesibilidad y Usabilidad.

QA-006 - Envío de emails

Nombre	Envío de emails
Identificador	QA-006
Descripción	La aplicación debe permitir el envío de correos electrónicos a los usuarios, ya sea para notificaciones, confirmaciones u otras comunicaciones relevantes, garantizando que el proceso sea fiable y seguro.
Tipo	Obligatorio

Tabla 72. QA-006 - Envío de emails.

QA-007 - Maquetación de emails

Nombre	Maquetación de emails
Identificador	QA-007
Descripción	La aplicación debe generar emails con un diseño claro y consistente con la identidad visual, asegurando que sean legibles en cualquier dispositivo y contribuyan a una buena experiencia de usuario.
Tipo	Obligatorio

Tabla 73. QA-007 - Maquetación de emails.

QA-008 - Escalabilidad

Nombre	Escalabilidad
Identificador	QA-008
Descripción	La aplicación debe ser construida para gestionar eficientemente el crecimiento en la base de usuarios y en el volumen de datos.
Tipo	Obligatorio

Tabla 74. QA-008 – Escalabilidad.

QA-009 - Disponibilidad

Nombre	Disponibilidad
Identificador	QA-009
Descripción	La aplicación debe ser construida para garantizar su disponibilidad y acceso a los usuarios en todo momento, evitando interrupciones en el servicio y asegurando una experiencia de usuario continua.
Tipo	Obligatorio

Tabla 75. QA-009 – Disponibilidad.

QA-010 - Autenticación JWT

Nombre	Autenticación JWT
Identificador	QA-010
Descripción	La aplicación debe implementar autenticación utilizando JSON Web Tokens (JWT), asegurando la protección de los recursos y la identificación fiable de los usuarios durante el acceso al sistema.
Tipo	Obligatorio

Tabla 76. QA-010 - Autenticación JWT.

QA-011 - Renovación automática de la sesión

Nombre	Renovación automática de la sesión
Identificador	QA-011
Descripción	La aplicación debe permitir que el cliente renueve el JWT Access Token automáticamente utilizando un Refresh Token válido, siempre que este no haya expirado, para mantener la sesión activa sin requerir una nueva autenticación del usuario.
Tipo	Obligatorio

Tabla 77. QA-011 - Renovación automática de la sesión.

QA-012 - Búsqueda y paginación de todos los recursos

Nombre	Búsqueda y paginación de todos los recursos
Identificador	QA-012
Descripción	La aplicación debe implementar búsqueda y paginación en todos los recursos, permitiendo a los usuarios filtrar y navegar grandes volúmenes de datos de forma eficiente, mejorando así la experiencia de usuario.
Tipo	Obligatorio

Tabla 78. QA-012 - Búsqueda y paginación de todos los recursos.

QA-013 - Base de datos SQL

Nombre	Base de datos SQL
Identificador	QA-013
Descripción	La aplicación debe utilizar una base de datos SQL para gestionar las relaciones complejas entre las distintas entidades del ámbito hospitalario.
Tipo	Obligatorio

Tabla 79. QA-013 - Base de datos SQL.

QA-014 - Exportación de datos en formato Excel (.xlsx) o JSON

Nombre	Exportación de datos en formato Excel (.xlsx) o JSON
Identificador	QA-014
Descripción	La aplicación debe permitir la exportación de los datos almacenados en la base de datos en formato Excel (.xlsx) o JSON, brindando al usuario la opción de elegir el formato durante el proceso.
Tipo	Obligatorio

Tabla 80. QA-014 - Exportación de datos en formato Excel (.xlsx) o JSON.

QA-015 - Seguridad y protección de la información confidencial de los usuarios

Nombre	Seguridad y protección de la información confidencial de los usuarios
Identificador	QA-015
Descripción	Dado que la aplicación maneja datos sensibles como direcciones, correos electrónicos, NIF, números de la seguridad social, entre otros, es fundamental garantizar la seguridad de la plataforma y la protección adecuada de la información personal de todos los usuarios.
Tipo	Obligatorio

Tabla 81. QA-015 - Seguridad y protección de la información confidencial de los usuarios.

3.3. Diseño y Arquitectura

En esta sección se presentará el diseño de las diferentes áreas que componen la aplicación web, así como la arquitectura general del proyecto. Se abordarán aspectos relacionados con la navegación, la gestión de bases de datos y la estructura arquitectónica en su conjunto.

3.3.1. Arquitectura

Al desarrollar la aplicación utilizando *frameworks* como React.js en el *frontend* y Django (con Django REST Framework) en el *backend*, se adopta una arquitectura de tipo Cliente-Servidor.

Esta estructura se basa en la separación lógica entre dos componentes: por un lado, el cliente (React.js) que consume servicios y presenta la interfaz al usuario, y, por otro lado, el servidor (Django) que se encarga de procesar la lógica de negocio, gestionar los datos y exponer una API que el *frontend* pueda utilizar (Fowler, Patterns of enterprise application architecture, 2012)

Uno de los enfoques que se ha utilizado en este contexto es el de *API for frontend*, que consiste en diseñar la API pensando específicamente en las necesidades de la interfaz de usuario. Este patrón permite crear una comunicación eficiente y centrada en el producto entre el *frontend* y el *backend*, mejorando la experiencia de desarrollo y la claridad de responsabilidades entre ambas capas (Roy, 2023)

Entre las ventajas de esta arquitectura destacan la facilidad para administrar datos y controlar la seguridad desde el servidor, así como la posibilidad de escalar el sistema o mantener de forma independiente tanto el *frontend* como el *backend*. Sin embargo, también implica ciertas limitaciones, como la dependencia total de la red para que el sistema funcione correctamente y el riesgo de fallos o ataques si el servidor no está disponible o seguro (Fielding)

La Figura 4 muestra la arquitectura resultante usando Django, Django REST Framework, React.js y el modelo Cliente-Servidor:

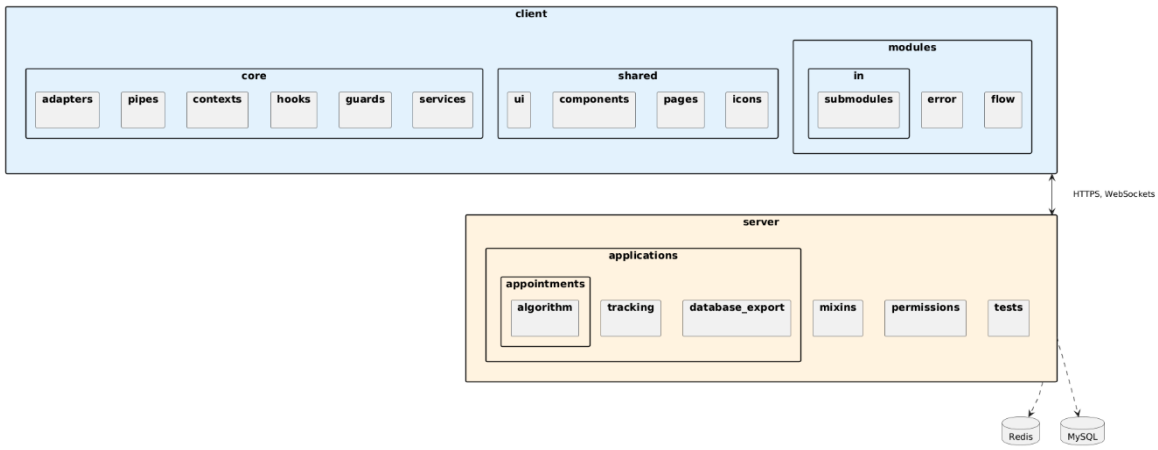

Figura 4. Arquitectura de la aplicación.

En el paquete *client*, la estructura se divide en tres bloques principales. El paquete *core* reúne la lógica central utilizada en todo el *frontend*, como hooks personalizados, servicios para llamadas a la API o manejo de almacenamiento local, contextos globales, guardas de rutas, y utilidades como adapters y pipes para transformar datos provenientes del *backend*. *shared* contiene recursos compartidos en toda la interfaz: estilos CSS y Sass (*ui*), páginas, componentes e íconos. Finalmente, *modules* organiza las funcionalidades específicas por secciones, incluyendo flujos de navegación, manejo de errores y submódulos internos relacionados con distintas áreas.

En el paquete *server* se concentra toda la lógica de negocio de la aplicación. Esto incluye la gestión avanzada de permisos, el acceso y manipulación de bases de datos MySQL y Redis, así como la interacción con las diversas entidades del sistema. Además, aquí se implementan funcionalidades clave como la asignación inteligente de citas, el seguimiento en tiempo real de pacientes y la exportación de datos. El paquete *applications* agrupa los módulos principales de funcionalidad, mientras que *mixins* y *permissions* encapsulan lógica auxiliar que facilitan la extensión y mantenimiento del sistema. Finalmente, en *tests* se ubican las pruebas de integración que aseguran el correcto funcionamiento del sistema en su conjunto.

La comunicación entre *client* y *server* se realiza mediante HTTPS para las operaciones estándar a través de la API REST, mientras que el rastreo de pacientes se apoya en WebSockets para lograr actualizaciones en tiempo real.

Aunque el diagrama refleja la arquitectura general del sistema, se han omitido muchos elementos para mantener la simplicidad visual, tanto del cliente como del servidor.

3.3.2. Navegación

A continuación, se muestra la navegación completa del sistema y la conexión entre las distintas pantallas, organizada en diagramas separados según el rol del usuario. Cada tipo de usuario (anónimo, paciente, doctor o administrador) tiene acceso a un conjunto específico de páginas, por lo que se han generado diagramas individuales que representan las vistas disponibles para cada uno.

La Figura 5 presenta la navegación del usuario anónimo. Al no estar autenticado, su acceso se restringe exclusivamente a las páginas de inicio de sesión, recuperación de contraseña y restablecimiento de contraseña. En consecuencia, permanece en la entrada del sistema sin posibilidad de interactuar con funcionalidades internas.

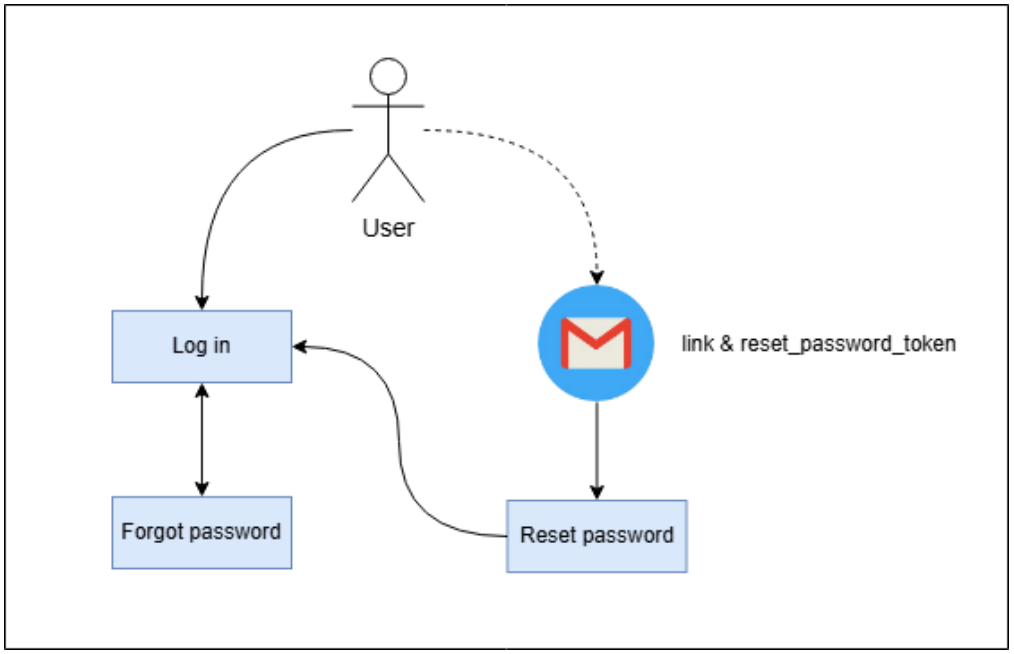

Figura 5. Mapa de Navegación del Usuario Anónimo.

La Figura 6 muestra la navegación del paciente. Este puede acceder a la página de inicio, solicitar una cita, consultar sus citas programadas, revisar sus pruebas médicas y tratamientos, así como gestionar su perfil. Desde este último, tendrá acceso a editar su información, cambiar su contraseña y cerrar sesión.

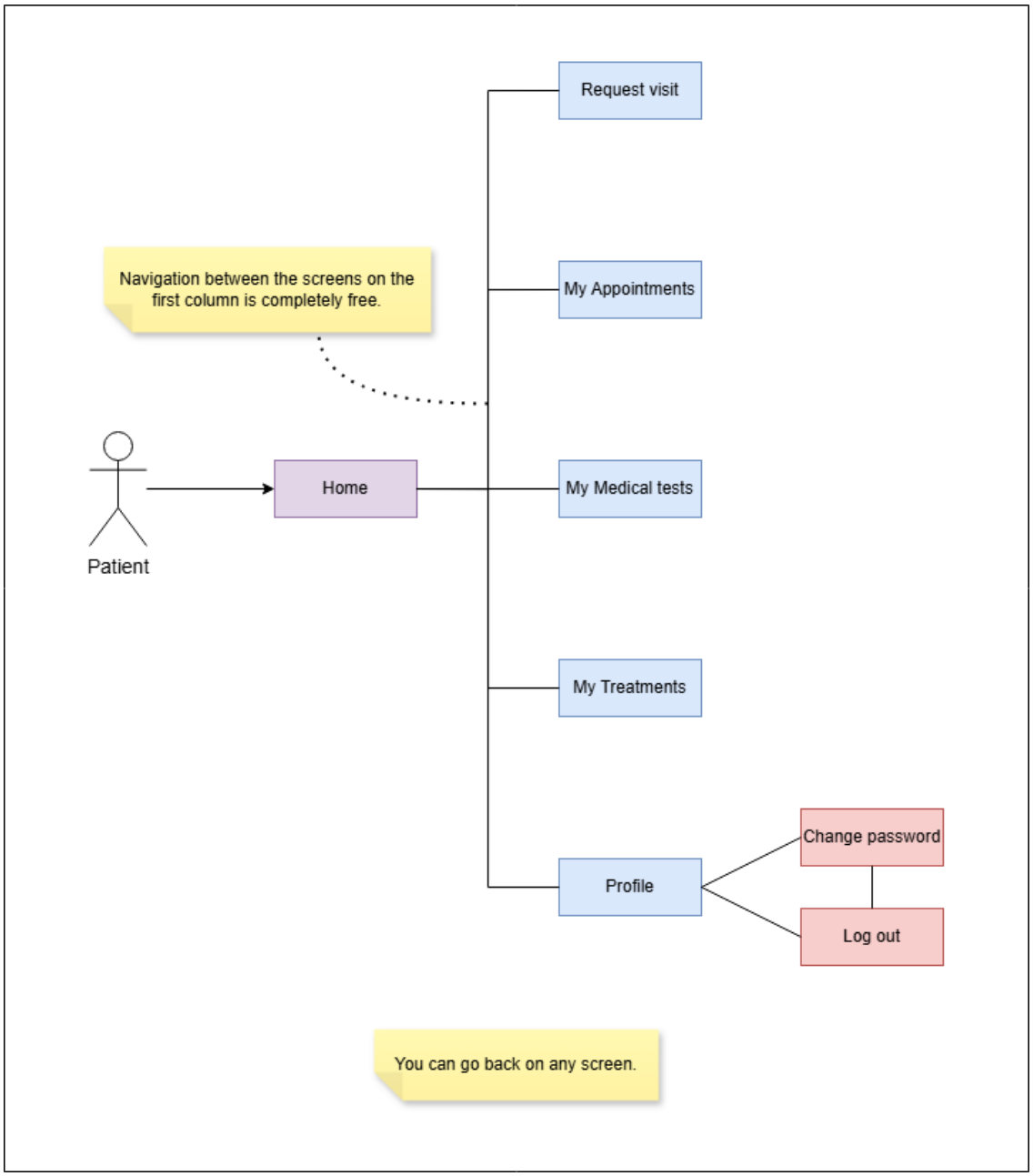

Figura 6. Mapa de Navegación del Paciente.

La Figura 7 muestra la navegación del doctor. Este puede acceder a la página de inicio, solicitar citas con especialidad para sus pacientes y consultar el listado de pacientes asignados. Desde allí, podrá ver su ubicación en tiempo real y acceder a su ficha médica (información personal, historial clínico, tratamientos y pruebas), así como editar sus datos. También dispone de un listado de citas en las que participa, con acceso al detalle y modificación, además de un calendario para visualizarlas de forma más clara. Adicionalmente, puede acceder a la exportación de la base de datos. Por último, puede gestionar su perfil, donde puede editar su información, cambiar la contraseña o cerrar sesión.

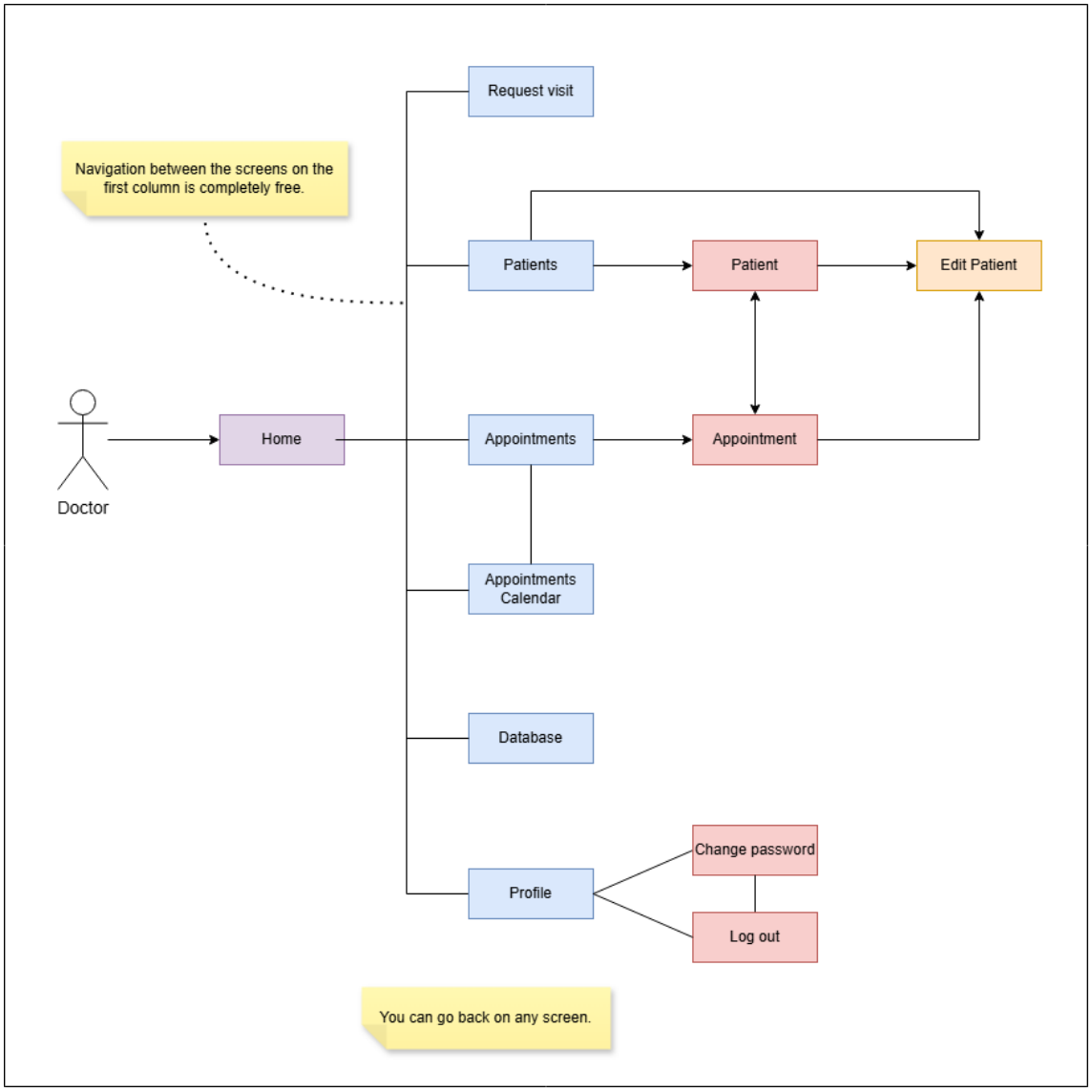

Figura 7. Mapa de Navegación del Doctor.

La Figura 8 muestra la navegación del administrador. Este tiene acceso a la página de inicio, así como a los listados de todas las entidades del sistema, con la posibilidad de visualizar sus detalles, editarlas o eliminarlas. También puede consultar la ubicación en tiempo real de todos los pacientes y acceder a sus fichas clínicas (información personal, historial clínico, tratamientos y pruebas), con opción de editarlas o eliminarlas, así como al listado completo de citas, con acceso a sus detalles y la posibilidad de modificarlas. Igualmente puede acceder a la exportación de la base de datos. Por último, puede gestionar su perfil, donde puede editar su información, cambiar la contraseña o cerrar sesión.

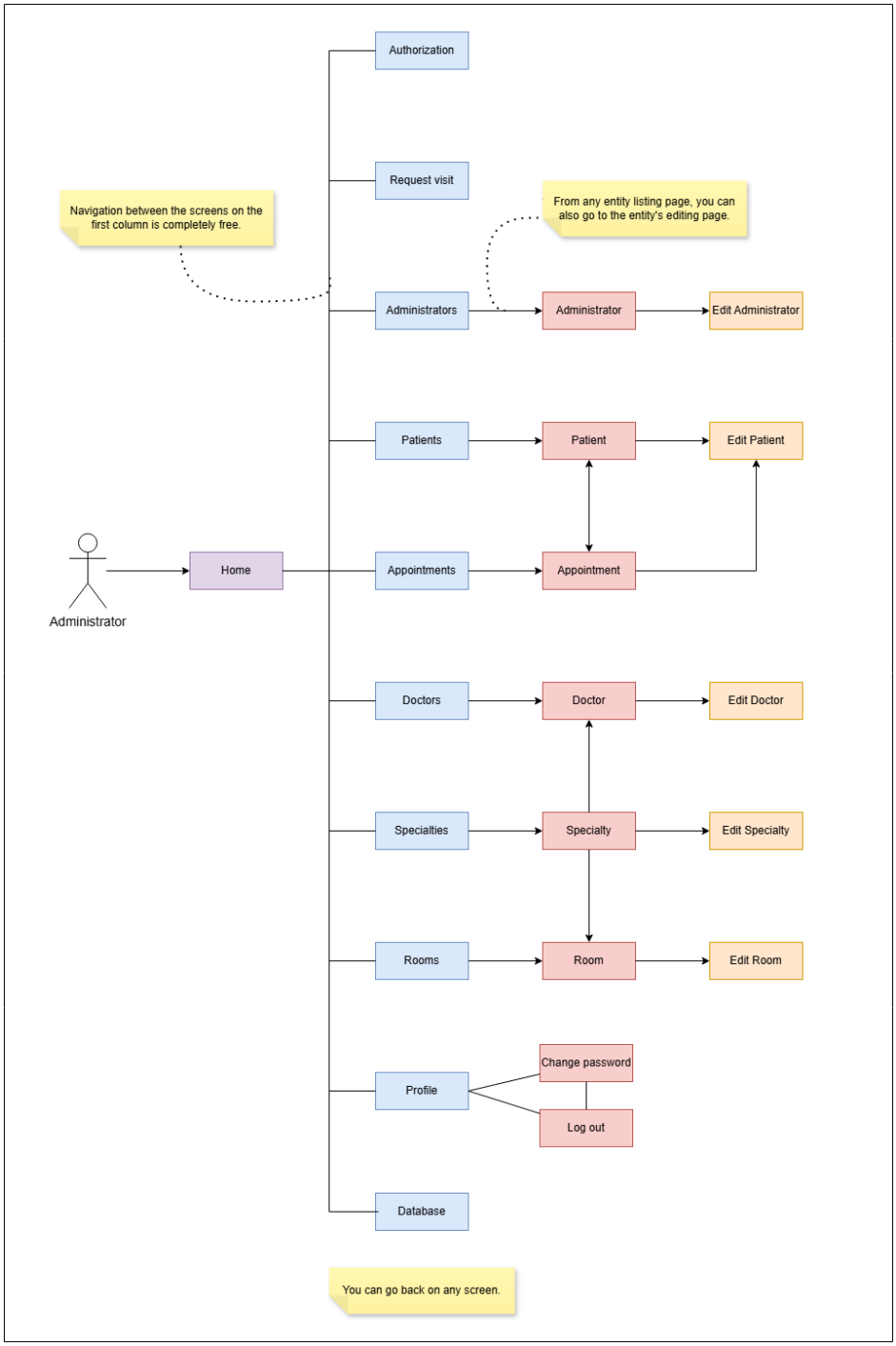

Figura 8. Mapa de Navegación del Administrador.

Finalmente, la Figura 9 muestra la navegación hacia páginas especiales. Si un recurso no puede cargarse en el momento, el usuario es redirigido a la página *Error404*. Si el usuario no está autenticado o no cuenta con los permisos necesarios, se le redirige a la página *Error403*. En todos los demás casos, se le permite acceder a la página solicitada.

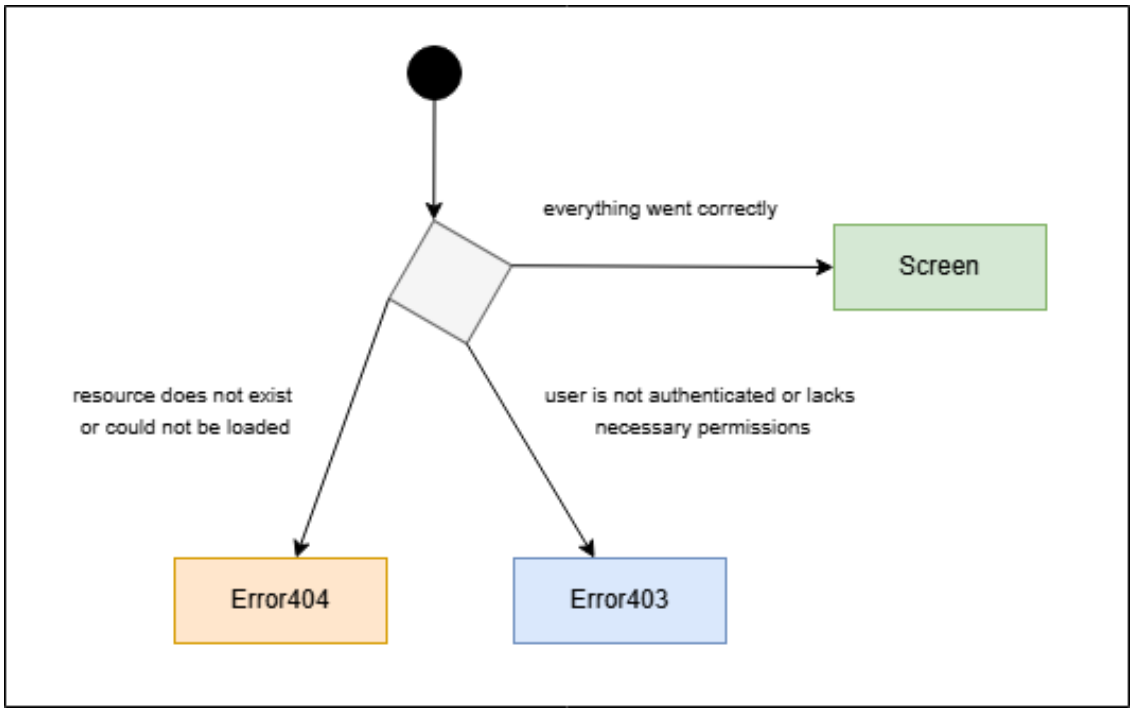

Figura 9. Acceso a páginas restringidas.

3.3.2. Base de datos

Debido a la naturaleza del entorno donde se desarrolla la aplicación, el ámbito hospitalario, se ha identificado un elevado número de entidades y relaciones entre ellas. Por ello, se ha optado por la utilización de MySQL como sistema de gestión de bases de datos relacional, ya que ofrece eficiencia y fiabilidad en el manejo de estructuras complejas.

La Figura 10 muestra todas las entidades del sistema y sus relaciones, donde:

- Un *Doctor*, especializado en *MedicalSpecialty*, tiene asignados *Patient* (médico de cabecera) y puede atender múltiples *Appointment*.
- Un *Patient* vive en un *Address* y puede solicitar múltiples *Appointment*.
- Un *Appointment*, que se da en un *Schedule*, involucra a un *Patient*, a un *Doctor* junto a su *MedicalSpecialty* y a la *Room* donde se celebra.
- En un *Appointment*, al *Patient* se le pueden mandar *Treatment* y requerir *MedicalTest*.
- *Administrator* es una entidad separada que gestiona y controla todo el sistema.
- La entidad *Schedule* se ha omitido para mantener la simplicidad visual.
- Las entidades *Doctor*, *Patient*, y *Administrator* heredan de la entidad *User*, que se ha omitido para mantener la simplicidad visual.

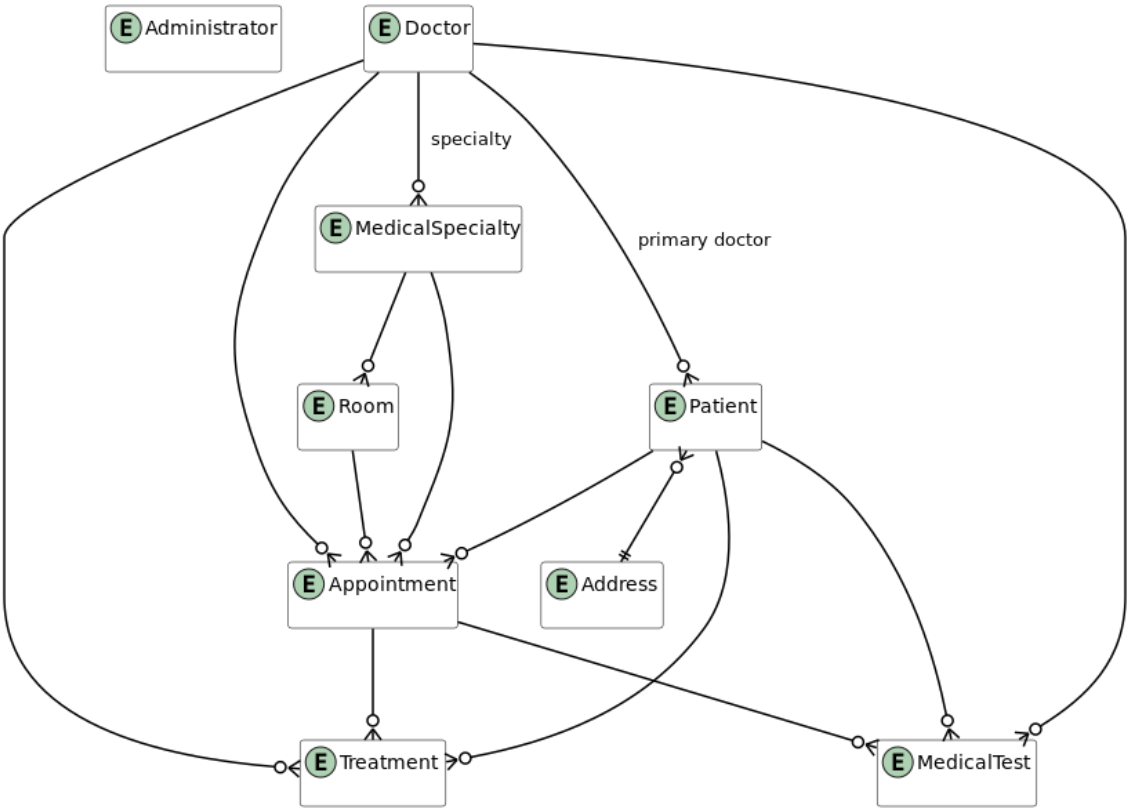

Figura 10. Diagrama E / R del sistema.

La Figura 11 muestra el diagrama correspondiente a la base de datos, acompañado de una descripción de los atributos de cada entidad.

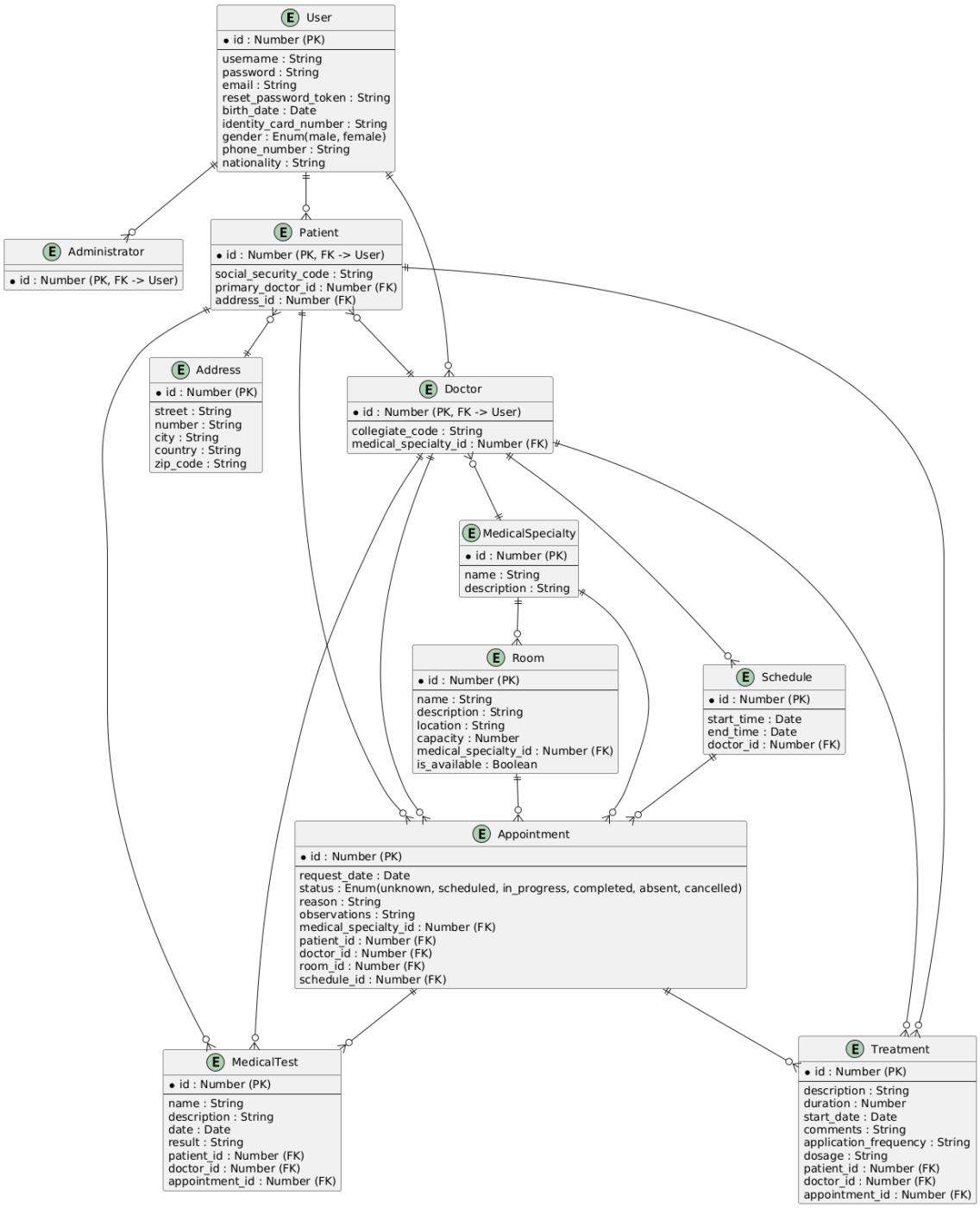

Figura 11. Diagrama Relacional de la Base de datos.

Complementariamente, se ha incorporado Redis para el rastreo de pacientes en tiempo real, dada su capacidad para gestionar información dinámica con alto rendimiento. Este aspecto se abordará con mayor detalle en la siguiente sección.

3.4. Implementación

Esta sección presenta los aspectos más significativos de la implementación detectados durante el desarrollo de la aplicación. Para este TFG, se detallarán mecanismos como el rastreo de pacientes en tiempo real, el algoritmo inteligente para la asignación de citas y la aplicación de criterios de usabilidad y accesibilidad.

3.4.1. Rastreo en tiempo real de pacientes

El rastreo en tiempo real de pacientes requiere una comunicación constante y fluida entre cliente y servidor. Sin embargo, este tipo de interacción no puede lograrse de forma eficiente mediante peticiones tradicionales sobre el protocolo HTTPS.

Las conexiones HTTP/HTTPS siguen un modelo de solicitud-respuesta, lo que obliga al cliente a realizar peticiones periódicas (polling) para detectar cambios. Esto introduce latencia y una carga innecesaria sobre el servidor y la red (Wang, 2013)

Para resolver esta limitación, se ha adoptado el uso de WebSockets, un protocolo que permite establecer un canal de comunicación persistente y en tiempo real entre cliente y servidor (Ably Realtime Ltd., 2022)

La Figura 12 muestra cómo se ha implementado esta solución en la aplicación. El servidor, utilizando Redis en modo *Publisher/Subscriber* (Redis, 2025), establece canales *patient/{id}/doctor/{id}* para cada par paciente-doctor. Ambos clientes, identificados por sus respectivos *ids*, se conectan al canal; el paciente publica su latitud y longitud aproximadamente cada 3 segundos, mientras que el doctor se suscribe para recibir estas actualizaciones en tiempo real.

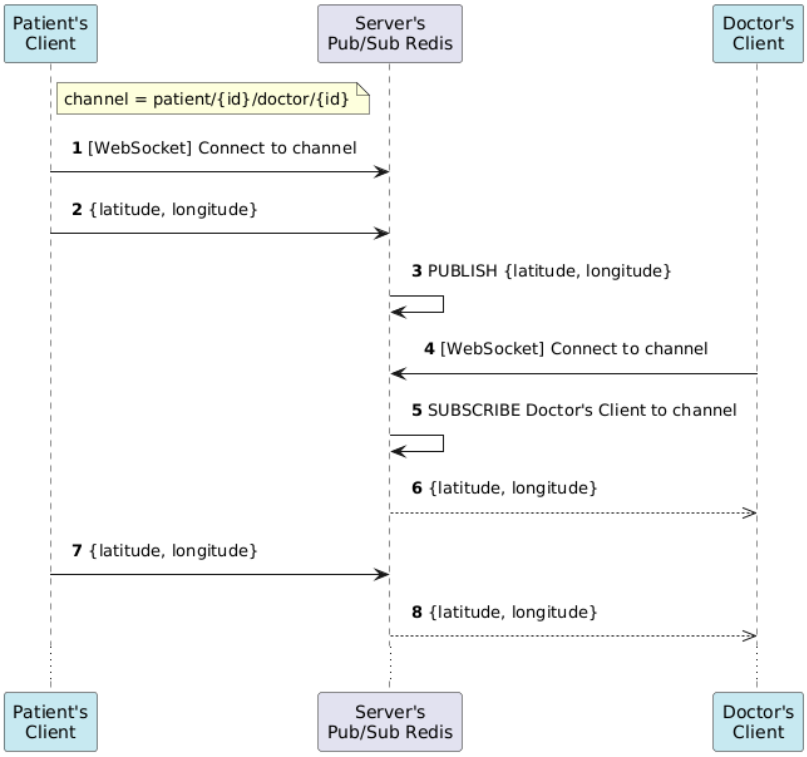

Figura 12. Diagrama de secuencia de Rastreo de pacientes en tiempo real.

En el cliente, esta lógica se desglosa en el hook useLocation.js para el paciente y en el componente MapServer.jsx para el doctor. En el servidor, se implementa en los archivos correspondientes a la aplicación tracking.

3.4.2. Algoritmo de asignación inteligente de citas

Se ha utilizado un enfoque de programación lineal entera binaria para resolver el problema de asignación de citas médicas. Este modelo es adecuado por la naturaleza discreta del problema (Vanderbei, 2020), donde las variables de decisión indican si se asigna (1) o no (0) una cita a un paciente en un horario específico. Además, permite manejar de forma rigurosa las múltiples restricciones implicadas, como la disponibilidad de recursos, las características de la cita y las preferencias de los pacientes.

3.4.2.1. Objetivos

El objetivo del modelo es minimizar el tiempo de espera de los pacientes y optimizar la utilización de los recursos hospitalarios, respetando los siguientes criterios:

- Disponibilidad de médicos, salas de consulta y franjas horarias dentro del rango permitido.
- Preferencias de los pacientes por turno de mañana, tarde o sin preferencia.
- Exclusividad horaria de médicos y salas.

3.4.2.2. Modelo matemático

Conjuntos utilizados

- $\mathcal{C}$: Conjunto de todas las citas médicas registradas en el sistema.
- $\mathcal{C}_{\text{programadas}} \subseteq \mathcal{C}$: Subconjunto de citas que ya se encuentran previamente agendadas y no pueden ser modificadas.
- $\mathcal{M}$: Conjunto de todos los médicos del sistema.
- $\mathcal{M}_{\text{disponibles}} \subseteq \mathcal{D}$: Subconjunto de médicos que están disponibles para atender nuevas citas.
- $\mathcal{S}$: Conjunto de todas las salas para alojar citas del sistema.
- $\mathcal{S}_{\text{especialidad}} \subseteq \mathcal{S}$: Subconjunto de salas asociadas a una especialidad médica específica, destinadas exclusivamente a alojar citas correspondientes a dicha especialidad.
- $\mathcal{D}$: Conjunto de días posibles para agendar citas médicas.
- $\mathcal{H}$: Conjunto de horas posibles para agendar citas médicas.
- $\mathcal{H}_{\text{preferidas}} \subseteq \mathcal{H}$: Subconjunto de horas preferidas por el paciente, definidas como prioritarias para agendar su cita.

Variable de decisión

$$X_{c,m,d,h,s} = \begin{cases} 1 & \text{si la cita } c \text{ se asigna al médico } m, \text{ en el día } d, \text{ a la hora } h, \text{ en la sala } s, \\ 0 & \text{en otro caso.} \end{cases}$$

Función objetivo

Minimizar el tiempo de espera del paciente.

$$min \sum_{c \in \mathcal{C}} \sum_{m \in \mathcal{M}} \sum_{d \in \mathcal{D}} \sum_{h \in \mathcal{H}} \sum_{s \in \mathcal{S}} (TiempoEspera(d,h)) \cdot X_{c,m,d,h,s}$$

Donde *TiempoEspera(d, h)* representa el tiempo entre la solicitud y el horario de la cita agendada para la cita c, el medico m, el día d, la hora h y la sala s.

Restricciones

1. **Naturaleza de las variables**.

   Las variables deben ser números enteros binarios.

   $$X_{c,d,t,h,s} \in \{0,1\} \quad \forall c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, h \in \mathcal{H}, s \in \mathcal{S}$$

2. **Las citas que ya están confirmadas deben mantenerse como tales**.

   $$X_{c,m,d,h,s} = 1 \quad \forall (c,m,d,h,s) \in \mathcal{C}_{programadas}$$

3. **Cada cita debe ser asignada exactamente una vez**.

   Una misma cita no puede estar agendada simultáneamente con más de un médico, en más de una fecha ni en más de una sala diferentes.

   $$\sum_{m \in \mathcal{M}_{disponibles}} \sum_{d \in \mathcal{D}} \sum_{h \in \mathcal{H}} \sum_{s \in \mathcal{S}} X_{c,m,d,h,s} = 1 \quad \forall c \in \mathcal{C}$$

4. **Un médico no puede atender más de una cita al mismo tiempo**.

$$\sum_{c \in \mathcal{C}} X_{c,m,d,h,s} \leq 1 \quad \forall m \in \mathcal{M}_{disponibles}, \quad \forall d \in \mathcal{D}, \quad \forall h \in \mathcal{H}, \quad \forall s \in \mathcal{S}$$

5. **Cada cita, de acuerdo con su especialidad médica, solo puede ser alojada en salas que correspondan a dicha especialidad**.

   $$\sum_{c \in \mathcal{C}} \sum_{m \in \mathcal{M}_{\text{disponibles}}} \sum_{d \in \mathcal{D}} \sum_{h \in \mathcal{H}} X_{c,m,d,h,s} = 0 \quad \forall s \in \mathcal{S} \backslash \mathcal{S}_{\text{especialidad}}$$

6. **Una sala no puede alojar más de una cita al mismo tiempo**.

   $$\sum_{c \in \mathcal{C}} \sum_{m \in \mathcal{M}_{\text{disponibles}}} X_{c,m,d,h,s} \leq 1 \quad \forall d \in \mathcal{D}, \quad \forall h \in \mathcal{H}, \quad \forall s \in \mathcal{S}$$

7. **Si el paciente selecciona un horario preferido, la cita debe agendarse en dicho horario**.

$$\sum_{m \in \mathcal{M}_{\text{disponibles}}} \sum_{d \in \mathcal{D}} \sum_{\substack{h \in \mathcal{H} \\ h \notin \mathcal{H}_{\text{preferidas}}}} \sum_{s \in \mathcal{S}} X_{c_{\text{nueva}},m,d,h,s} = 0$$

Donde c_{nueva} es la cita que se quiere agendar en el momento.

En el servidor, el modelo matemático se implementa con la biblioteca PuLP en el archivo algorithm.py de la aplicación appointments.

3.4.3. Usabilidad

En esta sección se detallan los principios de usabilidad que cumple el sistema.

La usabilidad hace referencia al grado en que un sistema software permite a los usuarios alcanzar sus objetivos de manera efectiva, eficiente y satisfactoria, facilitando así la realización de las tareas para las que fue diseñado (Nielsen Norman Group, 2025)

Se abordarán todos los principios de usabilidad, presentando un ejemplo representativo para cada uno. Aunque solo se muestre un ejemplo por principio para mantener la claridad visual, la aplicación está repleta de múltiples casos que los ejemplifican.

3.4.3.1. Eficiencia

Este eje se refiere a la rapidez con la que los usuarios pueden realizar tareas dentro del sistema.

La Figura 13 muestra que la aplicación cumple con este principio, ya que, por ejemplo, si se desea acceder a la ficha clínica de un paciente específico dentro de un listado extenso, se puede utilizar la función de búsqueda para localizarlo y así acceder rápidamente a su información.

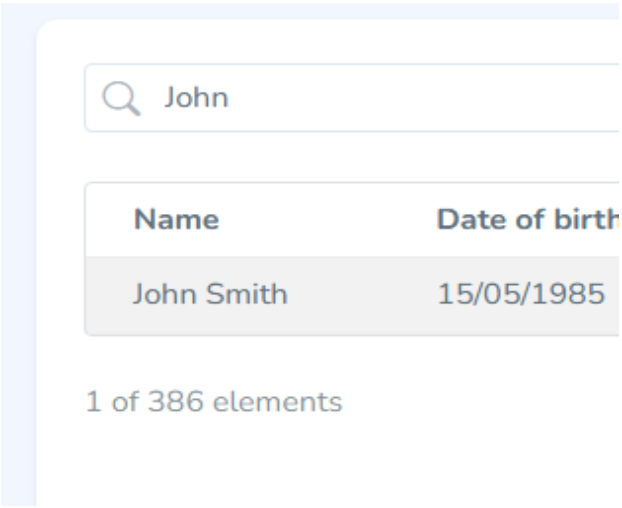

Figura 13. Ejemplo de Usabilidad – Eficiencia.

3.4.3.2. Facilidad de aprendizaje

Este eje se refiere a lo fácil que resulta para un nuevo usuario aprender a utilizar el sistema y comprender su funcionamiento desde los primeros usos.

La Figura 14 muestra que la aplicación cumple con este principio, ya que presenta una interfaz clara y sencilla. Además del menú lateral, que es intuitivo y fácil de navegar, el diseño en general utiliza íconos y textos explicativos que guían al usuario en el uso de las distintas funcionalidades.

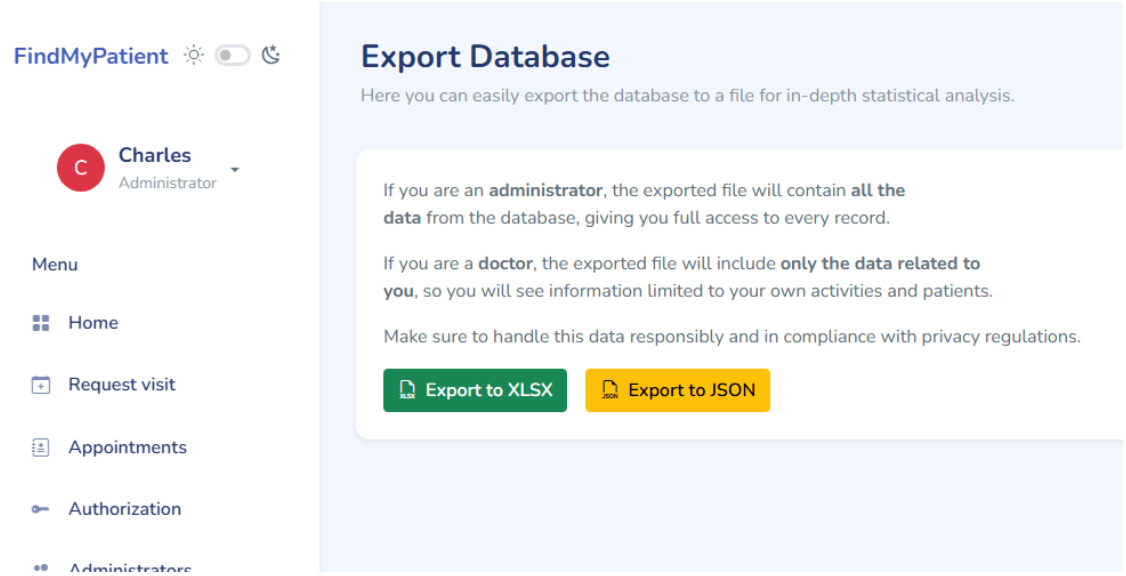

Figura 14. Ejemplo de Usabilidad - Facilidad de aprendizaje.

3.4.3.3. Memorabilidad

Este eje se refiere a la facilidad con la que un usuario puede recordar cómo usar un sistema después de haberlo dejado por un tiempo.

Las Figuras 15 y 16 muestran que la aplicación cumple con este principio gracias a una navegación clara, coherente y poco profunda. Las acciones principales están agrupadas en una barra visible y consistente en todas las pantallas, y se utilizan *tooltips* para orientar al usuario. Todo esto facilita que el usuario recuerde cómo usar la interfaz, incluso tras un tiempo sin utilizarla.

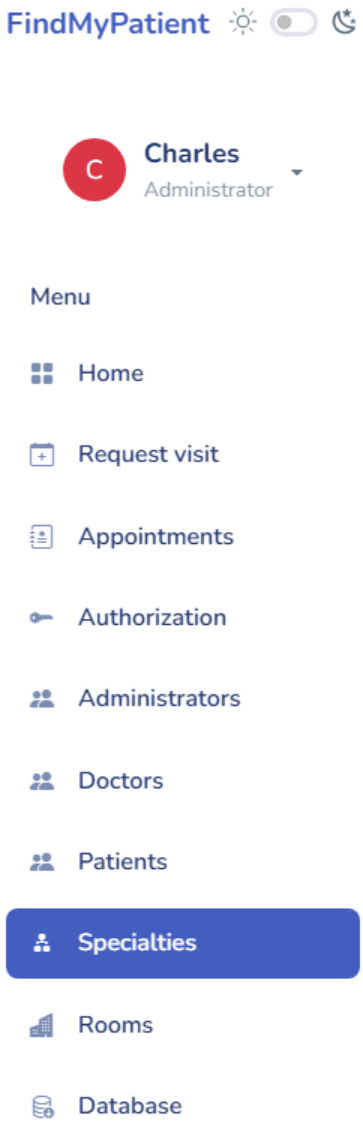

Figura 15. Ejemplo de Usabilidad – Memorabilidad.

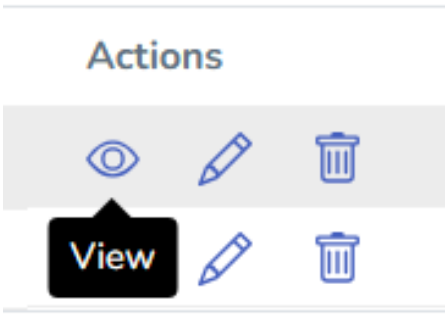

Figura 16. Ejemplo de Usabilidad – Memorabilidad (2).

3.4.3.4. Prevención de errores

Este eje se refiere a minimizar la ocurrencia de errores durante el uso del sistema, anticipándose a ellos y ofreciendo mecanismos para evitarlos.

Las Figuras 17 y 18 muestran que la aplicación cumple con este principio al solicitar confirmación antes de borrar o actualizar recursos, y al informar al usuario si un campo de un formulario ha sido rellenado de forma incorrecta, evitando así errores comunes.

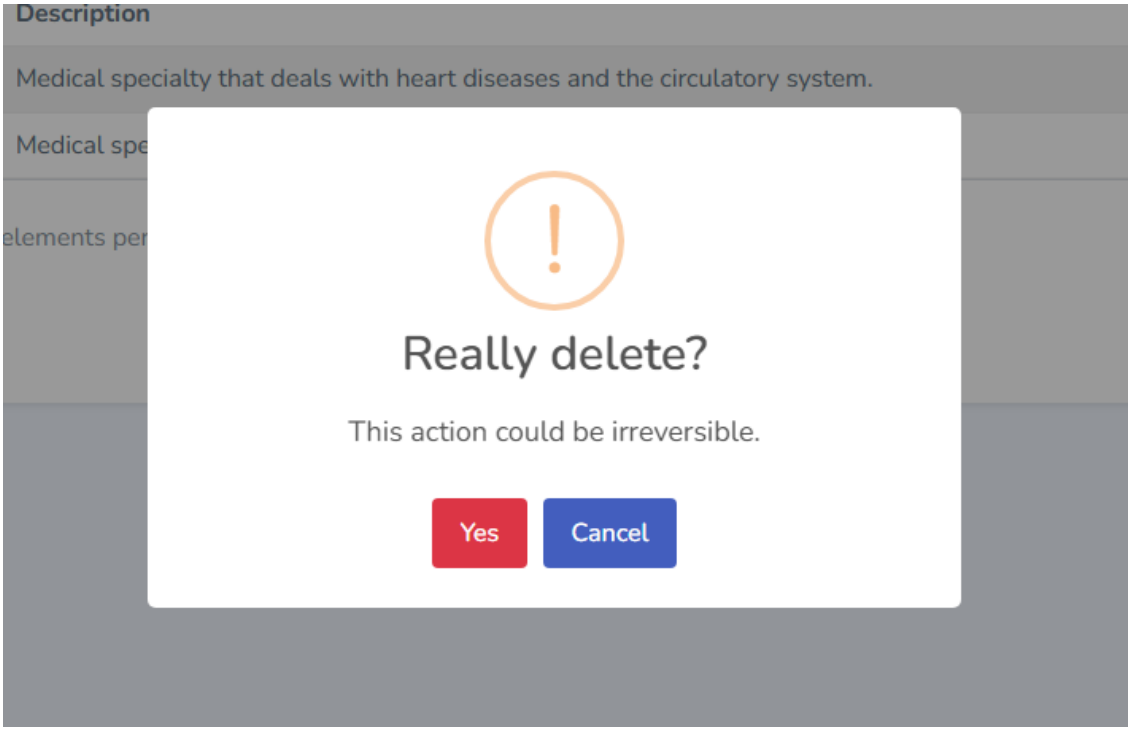

Figura 17. Ejemplo de Usabilidad - Prevención de errores.

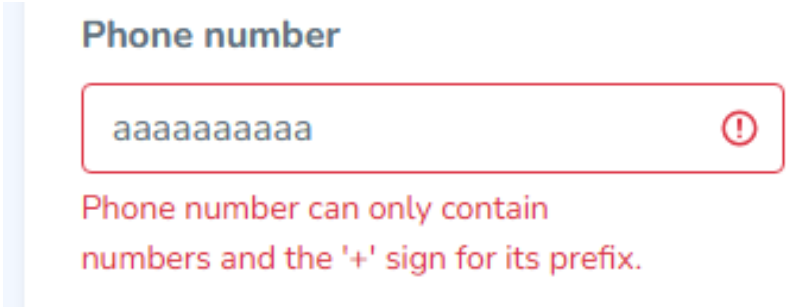

Figura 18. Ejemplo de Usabilidad - Prevención de errores (2).

3.4.3.5. Satisfacción

Este eje se refiere a la experiencia agradable y positiva que siente el usuario al interactuar con el sistema.

La Figura 19 muestra que la aplicación cumple con este principio, gracias a un diseño visual atractivo, una interfaz intuitiva, una respuesta rápida a las acciones del usuario y un diseño responsive que se adapta adecuadamente a diferentes dispositivos.

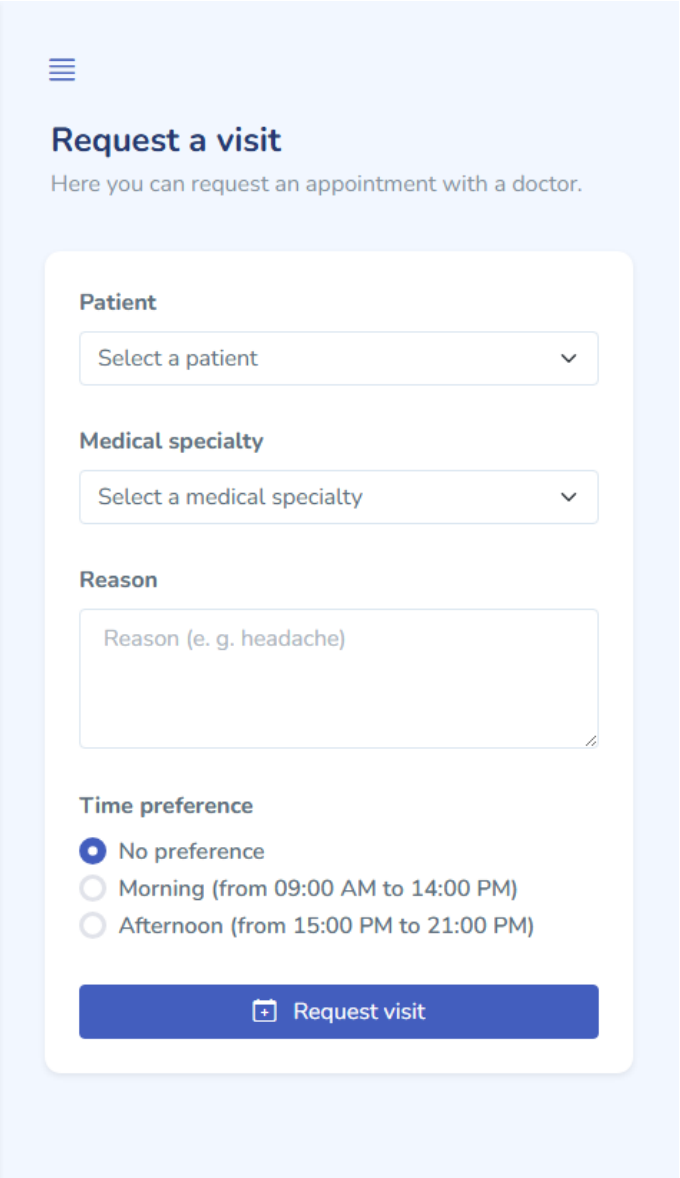

Figura 19. Ejemplo de Usabilidad – Satisfacción.

3.4.4. Accesibilidad

En esta sección se describen los principios de accesibilidad que cumple la aplicación web.

La accesibilidad se refiere al grado en que cualquier persona, independientemente de sus capacidades físicas o cognitivas, puede utilizar y comprender el sistema para llevar a cabo tareas específicas (Revilla Muñoz, 2018)

Se abordarán todos los principios de accesibilidad, presentando un ejemplo representativo para cada uno. Aunque solo se muestre un ejemplo por principio para mantener la claridad visual, la aplicación está repleta de múltiples casos que los ejemplifican.

3.4.4.1. Perceptible

Este eje se refiere a que la información del sistema debe ser presentada de forma que todos los usuarios puedan percibirla.

La aplicación cumple con este principio al utilizar texto como elemento principal para describir acciones y componentes, complementado por iconos y colores que brindan contexto adicional (véase la Figura 14). Esto permite que la mayoría de usuarios, sin discapacidades visuales graves, puedan acceder a la información de forma clara.

3.4.4.2. Operable

Este eje se refiere a que los usuarios puedan interactuar con todos los elementos del sistema, independientemente del método de entrada utilizado.

La aplicación cumple parcialmente con este principio, ya que puede utilizarse mediante teclado y ratón tradicionales, así como a través de pantallas táctiles o teclados virtuales (véase la Figura 19). Sin embargo, no ofrece compatibilidad con tecnologías de entrada no convencionales como comandos de voz, por lo que usuarios con discapacidades motrices severas podrían encontrar barreras.

3.4.4.3. Comprensible

Este eje se refiere a que la información y la interfaz del sistema sean fáciles de entender.

La aplicación cumple con este principio al incorporar elementos como barras de carga, mensajes emergentes (pop-ups) y notificaciones (toasts) que informan claramente sobre el estado del sistema o el resultado de las acciones (véase las Figuras 17, 20 y 21)

Sin embargo, el sistema solo está disponible en idioma inglés, lo cual podría limitar su comprensión para personas que no dominen este idioma.

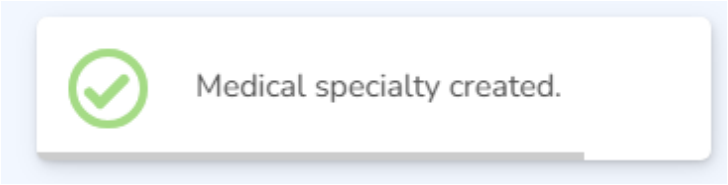

Figura 20. Ejemplo de Accesibilidad - Comprensible.

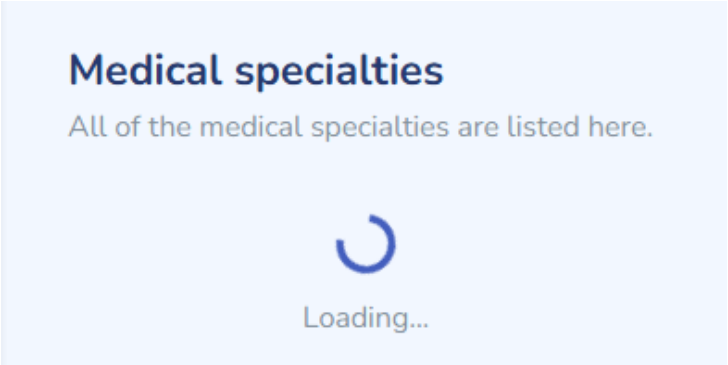

Figura 21. Ejemplo de Accesibilidad - Comprensible (2).

3.4.4.4. Robusto

Este eje se centra en avaluar la compatibilidad del sistema con todo tipo de tecnologías.

La aplicación cumple con este principio, ya que es accesible desde los principales navegadores modernos y se adapta correctamente a distintos tamaños de pantalla (véase las Figuras 17, 19 y 21). Esto permite su uso en computadoras, tablets y teléfonos inteligentes.

3.5. Pruebas

En esta sección se detallan las pruebas realizadas en la aplicación, las cuales incluyen pruebas unitarias, de componentes y de integración.

Para las pruebas unitarias y de componentes, se ha utilizado el *framework* Vitest. Para las pruebas de integración, se ha utilizado el sistema de ejecución de pruebas integrado en Django, Django Test Runner.

3.5.1. Pruebas Unitarias y Pruebas de Componente

Se llevaron a cabo pruebas unitarias y de componente en todas las páginas y componentes del cliente. Por ejemplo, en el caso de la *Sidebar*, se verificó lo siguiente:

- Se renderiza correctamente.
- Se renderiza correctamente con las opciones de navegación según el rol del usuario.
- Resalta la opción activa de acuerdo con la ruta actual.
- Responda correctamente en versión móvil, con apertura y cierre del menú lateral.
- Responda correctamente en versión tablet, con apertura y cierre del menú lateral.
- Responda correctamente en versión de escritorio.

En total, se ejecutaron 52 tests unitarios y de componentes.

3.5.2. Pruebas de Integración

Se llevaron a cabo pruebas de integración en la totalidad de los endpoints del servidor, comprobando tanto su correcto funcionamiento como el cumplimiento de los distintos niveles de permisos definidos.

Por ejemplo, para una operación tan básica como obtener un usuario, se contemplaron los siguientes escenarios:

- Usuario autenticado con rol administrador obtiene un usuario existente exitosamente.
- Usuario autenticado con rol administrador intenta obtener un usuario inexistente y recibe un error.
- Usuario autenticado con rol paciente intenta obtener información de otro paciente y se le deniega el acceso.
- Usuario autenticado con rol administrador accede correctamente a los datos de un usuario.

- Usuario no autenticado intenta acceder a un usuario y recibe un error de autorización.
- Usuario autenticado con rol doctor accede a un paciente que le fue asignado correctamente.
- Usuario autenticado con rol doctor intenta acceder a un paciente que no le fue asignado y se le deniega el acceso.
- Usuario autenticado accede a su propio perfil correctamente.
- Usuario autenticado intenta acceder al perfil de otro usuario y se le deniega el acceso.

En total, se ejecutaron 285 tests de integración.

3.6. Despliegue

En esta sección se explicarán los pasos realizados para el despliegue de la aplicación. Se detallarán los distintos workflows creados mediante GitHub Actions para este propósito.

3.6.1. Integración Continua

La integración continua implica que cada modificación en el código del proyecto se somete a pruebas automáticas para asegurar que se incorpora correctamente al sistema sin causar errores *(Fowler, Continuous Integration, 2020)*

Se ha creado el archivo ci-cd.yml, el cual prepara el entorno y ejecuta las pruebas correspondientes cada vez que se realiza un commit en la rama develop, al crear una Pull Request y al hacer merge de una Pull Request hacia las ramas develop o main.

```yaml
name: CI/CD Pipeline

on:
  push:
    branches:
      - develop
      - main
  pull_request:
    branches:
      - develop
      - main
```

Figura 22. ci-cd.yml – Previas condiciones de ejecución.

```yaml
- name: Install dependencies
  run: |
    python -m pip install --upgrade pip
    pip install -r requirements.txt
  working-directory: backend

- name: Apply database migrations
  run: |
    python manage.py makemigrations
    python manage.py migrate
  working-directory: backend

- name: Run backend tests
  run: python manage.py test
  working-directory: backend
```

Figura 23. ci-cd.yml – Ejecución de tests del servidor.

```yaml
- name: Install dependencies
  run: npm install --legacy-peer-deps
  working-directory: frontend

- name: Run frontend tests
  run: npm run test
  working-directory: frontend
```

Figura 24. ci-cd.yml – Ejecución de tests del cliente.

3.6.2. Entrega Continua

La entrega continua implica que los cambios en el código se preparan automáticamente para su liberación a producción, garantizando que el software esté siempre en un estado desplegable y que las actualizaciones puedan entregarse de manera rápida y confiable (Humble, 2010)

En este caso, se actualizan las imágenes de Docker publicadas en Docker Hub. El archivo ci-cd.yml incluye un job que, utilizando las credenciales de la plataforma, ejecuta el script build_and_push_image.sh para realizar dicha actualización, siempre que los jobs *backend-ci* y *frontend-ci* hayan finalizado correctamente y se haya hecho un push a la rama main.

```yaml
build-and-push-images-to-docker-hub:
  name: Build and Push Docker Images to Docker Hub
  runs-on: ubuntu-latest
  needs: [backend-ci, frontend-ci]
  if: github.ref == 'refs/heads/main'

  steps:
    - name: Get .env file from secrets
      run: echo "${{ secrets.ENVIRONMENT_FILE }}" > .env

    - name: Run script
      run: |
        ./build_and_push_image.sh \
            backend.dockerfile \
            ${{ secrets.DOCKER_HUB_USERNAME }}/findmypatient-backend:latest \
            ${{ secrets.DOCKER_HUB_USERNAME }} ${{ secrets.DOCKER_HUB_PASSWORD }}

        ./build_and_push_image.sh \
            frontend.dockerfile \
            ${{ secrets.DOCKER_HUB_USERNAME }}/findmypatient-frontend:latest \
            ${{ secrets.DOCKER_HUB_USERNAME }} ${{ secrets.DOCKER_HUB_PASSWORD }}
```

Figura 25. ci-cd.yml – Actualizar imágenes de Docker publicadas en Docker Hub.

El fichero .env se extrae previamente de los secretos de GitHub para que las imágenes de Docker puedan utilizar estas variables durante su ejecución.

Las imágenes de Docker están disponibles en la siguiente página.

Por otro lado, el *workflow* ci-cd.yml incluye un job encargado de desplegar estas imágenes en los servicios en la nube de Microsoft Azure, específicamente en dos Azure Container Instance. Este despliegue se realiza mediante el uso de claves federadas asociadas al grupo de recursos al que pertenecen los contenedores, siempre que los jobs *backend-ci*, *frontend-ci* y *build-and-push-images-to-docker-hub* hayan finalizado correctamente y se haya hecho un push a la rama main.

```yaml
deploy-to-azure:
  name: Deploy to Azure
  runs-on: ubuntu-latest
  needs: [backend-ci, frontend-ci, build-and-push-images-to-docker-hub]
  if: github.ref == 'refs/heads/main'
```

Figura 26. ci-cd.yml - Despligue de imágenes en Microsoft Azure.

```yaml
- name: Azure CLI login
  uses: azure/login@v2
  with:
    client-id: ${{ secrets.AZURE_CLIENT_ID }}
    tenant-id: ${{ secrets.AZURE_TENANT_ID }}
    subscription-id: ${{ secrets.AZURE_SUBSCRIPTION_ID }}
```

Figura 27. ci-cd.yml - Despligue de imágenes en Microsoft Azure (2).

```yaml
- name: Deploy Docker images to Azure Container Instances
  run: |
    az container create \
      --resource-group ${{ secrets.AZURE_RESOURCE_GROUP }} \
      --name findmypatient-backend \
      --image ${{ secrets.DOCKER_HUB_USERNAME }}/findmypatient-backend:latest \
      --registry-login-server index.docker.io \
      --registry-username ${{ secrets.DOCKER_HUB_USERNAME }} \
      --registry-password ${{ secrets.DOCKER_HUB_PASSWORD }} \
      --dns-name-label backend-findmypatient \
      --ports 80 \
      --os-type Linux \
      --cpu 1 \
      --memory 1.5

    az container create \
      --resource-group ${{ secrets.AZURE_RESOURCE_GROUP }} \
      --name findmypatient-frontend \
      --image ${{ secrets.DOCKER_HUB_USERNAME }}/findmypatient-frontend:latest \
      --registry-login-server index.docker.io \
      --registry-username ${{ secrets.DOCKER_HUB_USERNAME }} \
      --registry-password ${{ secrets.DOCKER_HUB_PASSWORD }} \
      --dns-name-label frontend-findmypatient \
      --ports 80 \
      --os-type Linux \
      --cpu 1 \
      --memory 1.5
```

Figura 28. ci-cd.yml - Despligue de imágenes en Microsoft Azure (3).

Este job descarga las imágenes Docker actualizadas desde Docker Hub para crear y ejecutar los contenedores necesarios, utilizando máquinas con configuración optimizada para minimizar recursos, aprovechando la suscripción estudiantil proporcionada por la universidad.

El contenedor del backend se conecta a las bases de datos desplegadas en Azure, específicamente a un Azure MySQL Flexible Server y a un Azure Redis Cache.

Tras el despliegue, el frontend está disponible en frontend-findmypatient.spaincentral.azurecontainer.io y el backend en backend-findmypatient.spaincentral.azurecontainer.io.

Cloudflare actúa como proxy inverso, redirigiendo los subdominios a los dominios personalizados frontend.findmypatient.software y backend.findmypatient.software, registrados sin coste a través de name.com gracias al GitHub Student Developer Pack proporcionado por la universidad.

Además, Cloudflare gestiona automáticamente los certificados SSL y el cifrado TLS de extremo a extremo, asegurando que todo el tráfico se sirva mediante HTTPS sin necesidad de configuración adicional, lo que mejora tanto la seguridad como el rendimiento de la aplicación.

Capítulo 4: Conclusiones y trabajos futuros

En este capítulo se presentarán las conclusiones del proyecto, haciendo énfasis en cumplimiento de los objetivos planteados, las dificultades encontradas a lo largo del desarrollo y las posibles mejoras que se implementarán en la aplicación web en el futuro.

4.1. Objetivos logrados

Durante la realización de este TFG se planteó un objetivo principal centrado en el desarrollo de la aplicación web, acompañado de varios objetivos secundarios que sirvieron como guía para alcanzar la meta principal.

El objetivo principal de este proyecto era crear un sistema de gestión hospitalaria accesible, moderno e inteligente, con una experiencia de usuario cuidada y eficiente, que facilitase la administración de recursos, optimizase la asignación de citas y permitiese el rastreo en tiempo real de pacientes que lo requiriesen. Este objetivo se ha alcanzado satisfactoriamente, ya que la plataforma desarrollada cumple con todos estos requisitos y puede ser utilizada desde cualquier dispositivo.

En cuanto a los objetivos secundarios, el primero de ellos consistía en realizar un análisis de los sistemas informáticos hospitalarios implantados hasta la fecha. Este análisis, que ha quedado reflejado en este documento y en las decisiones tomadas respecto al diseño y funcionamiento de la plataforma, se ha llevado a cabo satisfactoriamente.

El segundo propósito se centró en identificar los requisitos, funcionalidades clave y tecnologías para implementar en la aplicación. Esto ha sido posible gracias, en gran parte, al estudio sobre los sistemas informáticos hospitalarios actuales, realizado en el primer objetivo.

Asimismo, se abordó el diseño de una arquitectura flexible y escalable, lo que, gracias a las tecnologías y herramientas empleadas, ha facilitado el mantenimiento y la evolución futura del sistema.

Por último, al realizar este TFG, se ha elaborado una documentación clara y concisa sobre el proceso de desarrollo abarcando tanto las funcionalidades de la aplicación como la estructura interna del código y la relación entre sus distintos componentes.

4.2. Problemas afrontados

Este desarrollo se ha enfrentado a cinco dificultades principales que han influido de manera significativa en el tiempo de desarrollo.

4.2.1. Tecnologías utilizadas

La mayoría de las tecnologías empleadas eran nuevas para el desarrollador, puesto que no se abordaron o no se profundizaron en ellas durante mi paso por el grado universitario.

El empleo de herramientas como React.js, Django y Django REST Framework, con las que no estaba familiarizado antes de comenzar el proyecto, supuso una curva de aprendizaje que ralentizó bastante el desarrollo.

La selección de estas tecnologías se basó en su proyección a futuro, ya que se trata de tecnologías modernas que cuentan con un desarrollo y soporte continuo. Por ello, el esfuerzo invertido en su implementación garantiza una calidad y sostenibilidad a largo plazo.

4.2.2. Maquetación de la interfaz

La maquetación de la interfaz ha supuesto un reto considerable, en parte debido a la elevada autoexigencia impuesta en esta área, ya que es uno de los aspectos del proyecto que más me entusiasman.

Me propuse crear una interfaz adaptada al ámbito sanitario, coherente y minimalista, junto con una experiencia de usuario muy cuidada, adaptada a los tres roles de usuario diferentes (administrador, doctor y paciente) considerando las necesidades y posibles problemas que pudieran surgir para cada uno.

Aunque empleé una librería de componentes para agilizar el desarrollo, estos solo sirvieron como base, ya que fue necesario personalizar ampliamente el CSS interno de los mismos para lograr la estética deseada.

Asimismo, se ha puesto especial atención en el diseño responsive, garantizando que la interfaz se adapte correctamente y sea funcional en cualquier tipo de dispositivo.

4.2.3. Ubicación en tiempo real

La implementación del rastreo tiempo real de pacientes también ha sido un reto técnico. He tenido que enfrentarme a tecnologías y conceptos completamente nuevos para mí.

Por un lado, nunca antes había trabajado con WebSockets, por lo que tuve que aprender desde cero cómo funcionan los canales de comunicación en tiempo real, cómo se establecen y mantienen las conexiones entre el servidor y los clientes, así como su integración con las tecnologías empleadas en el sistema.

Además, el uso de Redis como sistema de gestión de canales y almacenamiento intermedio también supuso un aprendizaje adicional. Fue necesario familiarizarme con su instalación, configuración e integración para garantizar un buen rendimiento en la transmisión de datos en tiempo real.

Por último, tampoco tenía experiencia previa en la incorporación de mapas interactivos en aplicaciones web. La necesidad de mostrar la localización de los usuarios en tiempo real en el cliente implicó aprender a trabajar con librerías específicas de mapas para React, gestionar la actualización dinámica de los marcadores y resolver problemas relacionados con la sincronización de la información geográfica.

4.2.4. Algoritmo de asignación inteligente de citas

El algoritmo de asignación de citas representó un reto novedoso y exigente dentro del desarrollo del proyecto. Si bien durante mi formación universitaria hemos resuelto numerosos problemas de optimización lineal en bastantes asignaturas, nunca antes había afrontado el reto de definir completamente un problema real y, además, hacerlo dentro de un marco tan específico como la programación lineal entera binaria.

Diseñar este algoritmo requirió un análisis exhaustivo del sistema de citas, para traducir sus necesidades y requisitos en una formulación matemática precisa y eficiente. Este proceso implicó revisar y cuestionar cada aspecto del problema.

Otra dificultad fue recopilar y limpiar todos los datos necesarios para el algoritmo, ya que provenían de distintas fuentes y formatos. La selección y validación debían ser muy cuidadosas, pues cualquier error podía afectar el resultado, por lo que fue imprescindible aplicar procesos estrictos de filtrado y control de calidad.

Finalmente, quedaba aprender a utilizar librerías especializadas en resolver este tipo de problemas en Python, como PuLP, así como integrar el algoritmo resultante en el servidor.

4.2.5. Testing

El apartado de pruebas supuso un desafío considerable, ya que tuve que formarme en los distintos tipos de pruebas, así como en prácticas de integración continua, dado que estos temas no se trataron en profundidad en la carrera.

Este proceso resultó especialmente laborioso debido a la gran cantidad de funcionalidades y a la compleja gestión de permisos de la aplicación, lo que me llevó a probar meticulosamente cada caso posible. Como resultado, la aplicación cuenta con una batería de más de 325 tests en total, que cubren exhaustivamente los diferentes escenarios.

Esta robusta cobertura de pruebas ha proporcionado una red de seguridad fundamental, facilitando la integración y la entrega continua de nuevas funcionalidades con garantías de calidad y fiabilidad.

4.3. Trabajos futuros

El producto desarrollado en este TFG tendrá como finalidad ser utilizado como una aplicación de gestión de recursos hospitalarios, asignación inteligente de citas y rastreo de pacientes en tiempo real. Por lo tanto, será imprescindible actualizar y mantener el software de manera continua a medida que las necesidades del entorno hospitalario se expandan.

Entre las posibles líneas de evolución se encuentra la adaptación de la aplicación a una Progressive Web Application (PWA) (Mozilla, 2025) utilizando herramientas como Capacitor (Capacitor, 2025), lo que permitirá a los pacientes utilizar la plataforma cómodamente desde sus dispositivos móviles (sin acceder a la web) y explotar aún más la funcionalidad de localización en tiempo real.

De igual manera, será importante definir e implementar políticas claras y flexibles para la gestión y el consentimiento en la compartición de la ubicación por parte de los usuarios, garantizando así la privacidad y el cumplimiento normativo.

Por otro lado, otra mejora relevante es la ampliación del algoritmo de programación lineal entera binaria de asignación de citas, incorporando nuevas casuísticas y restricciones, como la minimización de los traslados del doctor entre distintas salas, lo que contribuiría a una gestión más eficiente de los recursos.

Además, la adaptación de la aplicación a diferentes idiomas y contextos culturales permitiría adaptar la aplicación a distintos idiomas y contextos culturales, facilitando su uso en entornos diversos.

También se plantea la necesidad de incrementar la cantidad y calidad de las pruebas del sistema. Actualmente, las pruebas se centran en la verificación funcional, aceptación y sanidad de la aplicación, por lo que resultará necesario añadir pruebas de carga, estabilidad y estrés. De este modo, se podrá asegurar la robustez y fiabilidad del servicio, incluso en situaciones de alta demanda o frente a posibles ataques.

Por último, se irán añadiendo funcionalidades dependiendo de las necesidades futuras del entorno hospitalario.

Bibliografía

Ably Realtime Ltd. (2022). *The WebSocket handbook*.

Atlassian. (2025). *Flujo de trabajo de Gitflow | Atlassian Git Tutorial*. Obtenido de https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow

Atlassian. (2025). *Transformación de tu flujo de trabajo con Trello*. Obtenido de https://www.atlassian.com/es/software/trello

Bootstrap. (2025). *Bootstrap · The most popular HTML, CSS, and JS library in the world*. Obtenido de https://getbootstrap.com/

Capacitor. (2025). *Using Capacitor with React*. Obtenido de https://capacitorjs.com/solution/react

Cardoen, B. D. (2010). Operating room planning and scheduling: A literature review. *European Journal of Operational Research*, 201(3), 921-932.

Cerner Corporation. (2025). *Cerner Millennium*. Obtenido de https://www.cerner.com/solutions/electronic-health-record-ehr

Cloudflare. (2025). *Conecta, protege y desarrolla en cualquier parte | Cloudflare*. Obtenido de https://www.cloudflare.com/es-es/

Conventional Commits. (2019). *Conventional Commits*. Obtenido de https://www.conventionalcommits.org/en/v1.0.0/

Diario Médico. (2019). Trazabilidad de pacientes quirúrgicos mediante RFID en el Hospital Universitario de Salamanca. *Diario Médico*.

Docker. (2025). *Docker: Accelerated Container Application Development*. Obtenido de https://www.docker.com/

draw.io. (2025). *draw.io*. Obtenido de https://app.diagrams.net/

DSF. (2025). *Django REST framework*. Obtenido de https://www.django-rest-framework.org/

DSF. (2025). *The web framework for perfectionists with deadlines | Django*. Obtenido de https://www.djangoproject.com/

ECMA. (2025). *ECMAScript® 2026 Language Specification*. Obtenido de https://tc39.es/ecma262/

Fielding, R. T. (s.f.). *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine.

Fowler, M. (2012). *Patterns of enterprise application architecture*. Addison-Wesley.

Fowler, M. (2020). *Continuous Integration*. Obtenido de https://martinfowler.com/articles/continuousIntegration.html

GitHub. (2025). *GitHub*. Obtenido de https://github.com/

Google. (2025). *Añadir Gmail a otro cliente de correo - Ayuda de Gmail*. Obtenido de https://support.google.com/mail/answer/7126229?hl=es

Gupta, D. &. (2008). Appointment scheduling in health care: Challenges and opportunities. *IIE Transactions*, 40(9), 800-819.

Haux, R. (2006). *Health information systems – past, present, future*.

Humble, J. &. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional.

Jetbrains. (2025). *PyCharm: The only Python IDE you need*. Obtenido de https://www.jetbrains.com/es-es/pycharm/

Kuperman, G. J. (2003). Computer physician order entry: benefits, costs, and issues. *Journal of the American Medical Informatics Association*, 10(4), 345–356.

LIDERLOGO. (2025). *Significado Del Color Azul | Liderlogo: Agencia De Diseño*. Obtenido de https://www.liderlogo.es/diseno/significado-del-color-azul/

Mailtrap. (2025). *Mailtrap: Plataforma de envío de emails*. Obtenido de https://mailtrap.io/es/

Meta. (2025). *React*. Obtenido de https://es.react.dev/

Microsoft. (2025). *Servicios de informática en la nube | Microsoft Azure*. Obtenido de https://azure.microsoft.com/es-es

Microsoft. (2025). *Visual Studio Code - Code Editing. Redefined*. Obtenido de https://code.visualstudio.com/

Mozilla. (2025). *Aplicaciones Web Progresivas | MDN*. Obtenido de https://developer.mozilla.org/es/docs/Web/Progressive_web_apps

Name.com. (2025). *Name.com*. Obtenido de https://www.name.com/es-la

Nielsen Norman Group. (2025). *Usability 101 (Video) - NN/g*. Obtenido de https://www-nngroup-com.translate.goog/videos/usability-101/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=rq#:~:text=Usability%20is%20defined%20by%205,memorability%2C%20errors%2C%20and%20satisfaction.

OpenJS. (2025). *ESLint*. Obtenido de https://eslint.org/

OpenStreetMap. (2025). *OpenStreetMap Wiki*. Obtenido de https://wiki.openstreetmap.org/

Oracle. (2025). *MySQL*. Obtenido de https://www.mysql.com/

Organización Mundial de la Salud. (2021). *Global strategy on digital health 2020–2025*. Obtenido de https://www.who.int/publications/i/item/9789240020924

phpMyAdmin. (2025). *phpMyAdmin*. Obtenido de https://www.phpmyadmin.net/

PlantUML. (2024). *https://plantuml.com/es/*. Obtenido de https://plantuml.com/es/

PSF. (2025). *Optimization with PuLP — PuLP documentation*. Obtenido de https://coin-or.github.io/pulp/

PSF. (2025). *psf/black: The uncompromising Python code formatter*. Obtenido de https://github.com/psf/black

PSF. (2025). *Welcome to Python.org*. Obtenido de https://www.python.org/

Redis. (2025). *Redis - La plataforma de datos en tiempo real*. Obtenido de https://redis.io/es/

Redis. (2025). *Redis Documentation - Pub/Sub*. Obtenido de https://redis.io/docs/interact/pubsub/

Revilla Muñoz, O. &. (2018). *Accesibilidad web. WCAG 2.1 de forma sencilla*. Obtenido de https://olgacarreras.blogspot.com/2018/11/libro-accesibilidad-web-wcag-21-de.html

Romero-Torres, A. e. (2021). Challenges in Hospital Information Systems: An Overview. *Journal of Healthcare Engineering*.

Roy, T. (2023). Designing APIs for Frontend Applications. *Modern Web Practices Journal*.

SAP Health. (2025). *SAP for Healthcare*. Obtenido de https://www.sap.com/industries/healthcare.html

Siemens Healthineers. (2025). *Selene HIS*. Obtenido de https://www.siemens-healthineers.com/es-es/healthcare-it/clinical-it/selene-his

Sommerville, I. (2020). *Ingeniería de software (10ª ed.)*. Pearson.

Vanderbei, R. J. (2020). En *Linear programming: Foundations and extensions (5th ed.)*. Springer.

Vitest. (2025). *Vitest | Next Generation testing framework*. Obtenido de https://vitest.dev/

W3C. (2025). *Cascading Style Sheets*. Obtenido de https://www.w3.org/Style/CSS/Overview.en.html

Wang, Q. X. (2013). WebSocket-based real-time communication system. *International Conference on Computer Sciences and Applications*.

Zocdoc. (2025). *Zocdoc*. Obtenido de https://www.zocdoc.com/

Anexos

Anexo I – Listado de Siglas

API – Application Programming Interface

CSS – Cascading Style Sheets

HTTPS – HyperText Transfer Protocol Secure

JSON – JavaScript Object Notation

SQL – Structured Query Language

TFG – Trabajo de Fin de Grado

HIS – Hospital Information System

FIFO – First In First Out

PWA – Progressive Web Application

IDE – Integrated Development Environment

Sass – Syntactically Awesome Stylesheets

PSF – Python Software Foundation

W3C – World Wide Web Consortium

ECMA – ECMAScript

SMTP – Simple Mail Transfer Protocool

DSF – Django Software Foundation

NIF – Número de identificación fiscal

JSON – JavaScript Object Notation

JWT – JSON Web Tokens

JS – JavaScript

ID, **id** – Identificador

SSL – Secure Sockets Layer

TLS – Transport Layer Security

DDoS – Distributed Denial of Service

DNS – Domain Name System

SSL – Secure Sockets Layer

Anexo II – Diseño y Demostración de la aplicación web

Puedes visualizar el diseño y la demostración de la aplicación web en el siguiente video.