Memoria del proyecto

Plataforma de monitorización remota para la atención domiciliaria

Trabajo de Fin de Grado

Ingeniería Informática

Julio de 2023

Autor

Germán Francés Tostado

Tutor/a

David Cruz García

Gabriel Villarrubia González

Certificado de los tutores

D. Gabriel Villarrubia González, profesor del Departamento de Informática y Automática de la Universidad de Salamanca y D. David Cruz García, personal investigador de la Universidad de Salamanca,

HACE/N CONSTAR:

Que el trabajo titulado “Plataforma de monitorización remota para la atención domiciliaria”, que se presenta, ha sido realizado por D. Germán Francés Tostado, con DNI 70940996A y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado en Ingeniería Informática en esta Universidad.

Salamanca, 03 de Julio de 2023

CRUZ GARCIA, DAVID (FIRMA)
Firmado digitalmente por CRUZ GARCIA, DAVID (FIRMA)
Fecha: 2023.07.04 12:09:53 +02'00'

D. David Cruz García

VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Firmado digitalmente por VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Fecha: 2023.07.04 11:51:20 +02'00'

D. Gabriel Villarrubia González

Resumen

En los últimos años, las crisis sanitarias han demostrado cómo hospitales desbordados pueden paralizar el mundo. El objetivo de este proyecto es proporcionar atención médica de calidad a distancia mediante el uso de una tableta médica que se conecta a sensores asequibles y fáciles de usar, lo que permite a los pacientes tener consultas precisas desde la comodidad de sus hogares.

Esto permitiría a los hospitales centrarse en los casos prioritarios, reduciendo las listas de espera y liberando recursos para las consultas presenciales. Además, el proyecto sería especialmente beneficioso para los pueblos pequeños sin centros de salud cercanos, donde las personas mayores con problemas de salud a menudo se ven obligadas a recorrer largas distancias para recibir atención médica.

La plataforma de atención médica a domicilio busca la teleasistencia médico-paciente para realizar chequeos básicos a distancia.

Los pueblos de España sufren la falta de médicos, aproximadamente 28% (4500) de los profesionales que trabajan en el mundo rural se jubilarán en los próximos 5 años. No sólo eso, sino que el mundo rural necesita cada vez más nuevos recursos tecnológicos para facilitar las tareas de los profesionales. Debido al pasado Covid-19 la cantidad de recursos sanitarios se ha reducido exponencialmente, poniendo en peligro la salud de los pacientes y la seguridad del personal médico. Con el sistema de medición portátil enfocado principalmente a entornos rurales con escasa asistencia médica y personas dependientes, será posible realizar un pre-diagnóstico sin necesidad de desplazarse a un centro médico, a varios kilómetros de los núcleos rurales.

Palabras clave: Teleasistencia - Asistencia sanitaria inteligente - Telemedicina - Plataforma de monitorización - Asistencia domiciliaria

Summary

In recent years, healthcare crisis have demonstrated how overwhelmed hospitals can bring the world to a standstill. The objective of this project is to provide quality medical care remotely through the use of a medical tablet that connects to easy-to-use affordable sensors, allowing patients to have precise consultations from the comfort of their homes.

This would enable hospitals to focus on priority cases, reducing waiting lists and freeing up resources for face-to-face consultations. Additionally, the project would be especially beneficial for smaller towns without nearby health centers, where elderly individuals with fragile health conditions are often forced to travel long distances for medical care.

The home medical care platform is looking for doctor-patient telecare to perform basic check-ups remotely.

The villages of Spain are suffering from a lack of doctors, approximately 28% (4500) of the professionals working in the rural world will retire in the next 5 years. Not only that, but the rural world is increasingly in need of new technological resources to facilitate the tasks of the professionals. Due to the past Covid-19 the amount of healthcare resources has been exponentially reduced, endangering the health of patients and the safety of medical staff. With the portable measurement system focused mainly on rural environments with little medical assistance and dependent people, it will be possible to do a pre-diagnosis without the need to travel to a medical centre, several kilometers away from the rural centres.

Keywords: Telecare · Smart Healthcare · Telemedicine · Monitoring Platform · Homecare

Tabla de contenidos

Índice de figuras

Índice de tablas

1.Introducción

En los últimos años, las crisis sanitarias han dejado en evidencia la vulnerabilidad de los sistemas de atención médica en todo el mundo. La falta de recursos, la sobrecarga de los hospitales y la dificultad de acceso a la atención médica se han convertido en desafíos apremiantes. Ante este panorama, el desarrollo de soluciones tecnológicas innovadoras se ha convertido en una prioridad para abordar estos problemas de manera efectiva.

Este trabajo se centra en una solución prometedora que busca transformar la atención médica y superar las barreras geográficas y de recursos en los entornos rurales: el uso de una tableta médica conectada a sensores asequibles. Combinada con la plataforma de atención médica a distancia permite a los pacientes recibir consultas precisas desde la comodidad de sus hogares, realizando chequeos básicos y pre-diagnósticos sin necesidad de desplazarse a un centro médico, liberando así a los hospitales para que se enfoquen en los casos más urgentes y reduciendo las listas de espera.

En el contexto específico de los pueblos de España, donde la falta de médicos y los recursos sanitarios limitados plantean serios desafíos, esta solución tecnológica puede marcar la diferencia. Con aproximadamente el 28% de los profesionales médicos rurales en vías de jubilación en los próximos cinco años, es necesario encontrar alternativas innovadoras que permitan ampliar el acceso a la atención médica y mejorar la calidad de vida de las comunidades rurales.

A través de este trabajo, se explorarán los beneficios y desafíos asociados con la implementación de esta plataforma de atención médica a distancia en áreas rurales y en situaciones de emergencia sanitaria. Asimismo, se analizará cómo esta tecnología puede contribuir a mejorar el acceso a la atención médica en comunidades remotas y facilitar la vida de las personas mayores y personas con problemas de salud crónicos que a menudo enfrentan dificultades para recibir atención médica cercana.

A través de un enfoque multidisciplinario que combina la tecnología, la medicina y la gestión de la salud, este trabajo pretende arrojar luz sobre la importancia de adoptar soluciones innovadoras en el campo de la atención médica y cómo estas pueden transformar la forma en que se brinda atención médica en áreas rurales.

En última instancia, este proyecto busca contribuir a la construcción de un sistema de salud más resiliente, capaz de adaptarse a escenarios cambiantes y de mejorar la calidad de vida de los habitantes de áreas rurales, al tiempo que garantiza una atención médica precisa, eficiente y accesible para todos.

El presente documento recogerá los aspectos más relevantes del desarrollo del proyecto siguiendo la estructura enumerada a continuación:

- **Estado del arte**: Se investigarán productos ya existentes que busquen tener la misma finalidad, de manera que sean competencia directa o alternativas similares.

- **Objetivos**: Se especificarán los objetivos que pretende conseguir tanto el proyecto como el desarrollo del mismo

- **Conceptos teóricos**: Se explicarán varios conceptos relacionados con el sistema para facilitar la comprensión del trabajo.

- **Técnicas y herramientas**: Se detallarán las técnicas y herramientas empleadas durante el desarrollo.

- **Aspectos relevantes**: Se recogerán los aspectos más interesantes en el desarrollo del sistema.

- **Limitaciones del proyecto**: Se discutirán las limitaciones encontradas durante la realización del trabajo.

- **Líneas de trabajo futuras y conclusiones**: Para compensar las limitaciones anteriores, se propondrán unas posibles mejoras futuras y se expondrá la conclusión sobre el sistema final.

- **Referencias bibliográficas**

Esta memoria irá complementada de seis anexos que profundizarán de manera más específica en cada una de las fases:

- **Anexo I: Planificación temporal**: Recogerá todo lo relacionado a la planificación temporal y estimación de recursos.

- **Anexo II: Especificación de requisitos**: Recogerá la especificación de requisitos software del sistema a desarrollar.

- **Anexo III: Análisis de requisitos**: Recogerá la documentación pertinente a la etapa de análisis de los requisitos del sistema.

- **Anexo IV: Diseño del sistema software**: Recogerá la documentación pertinente a la fase de diseño del sistema.

- **Anexo V: Manual técnico**: Explicará la aplicación desde el punto de vista técnico para ofrecer ayuda al programador mediante descripciones del código.

- **Anexo VI: Manual de usuario**: Recogerá el funcionamiento y la forma de interactuar con el sistema a un usuario sin conocimiento previo de la plataforma

2.Estado del arte

Antes de comenzar ningún tipo de modelado de la aplicación, se ha llevado a cabo una investigación del estado del arte y trabajos relacionados para partir de una base.

Respecto a la medición de constantes vitales, en los últimos años ha surgido un auge de dispositivos IoT con ese fin, mayormente pulseras. Los trabajos ya existentes han sido comercializados en el mercado, pero debido a la gran variedad y extensión de estos productos, es imposible desarrollar una aplicación que funcione para todos, algunos se centran en enfermedades crónicas como por ejemplo enfermedades cardíacas, diabetes, incluso dispositivos que ayudan a la detección temprana de Alzheimer usando algoritmos de deformación dinámica del tiempo.

Uno de los ejemplos de estos dispositivos es el WISE (Wearable IoT-cloud-baSed hEalth monitoring system) que adopta un número de sensores interconectados para observar la condición médica del paciente.

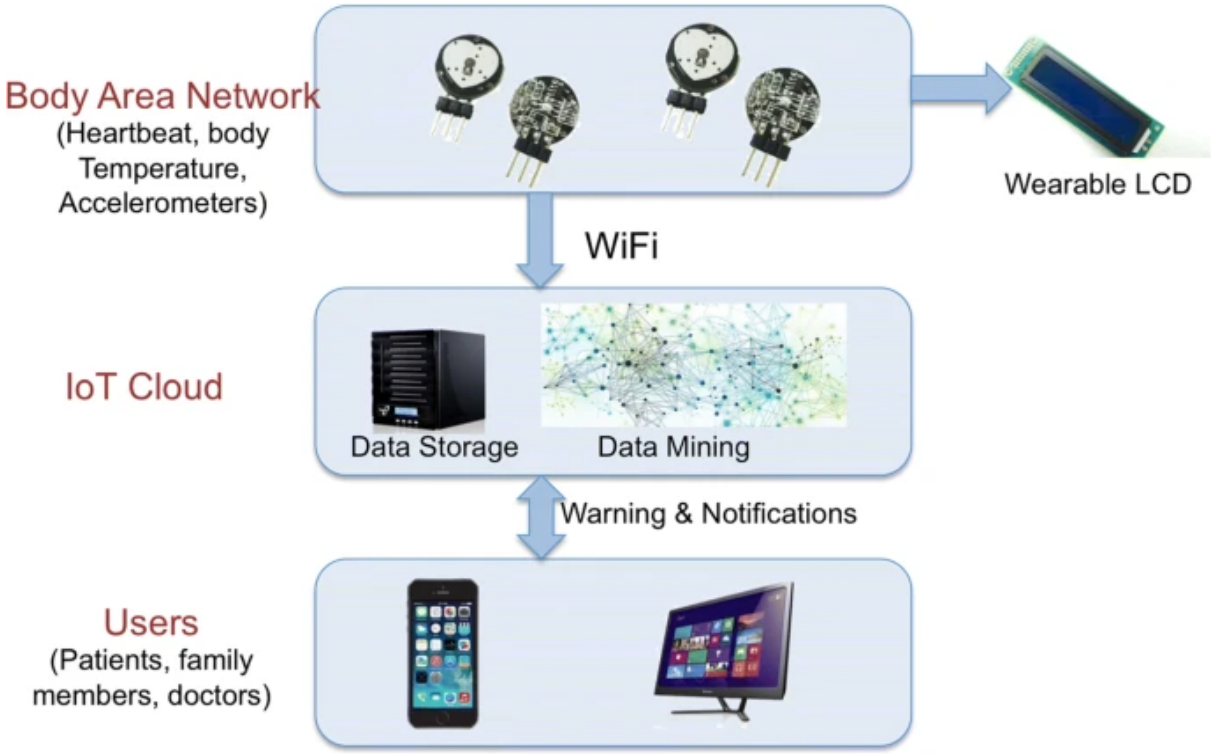


Figura 1: Framework del sistema WISE

Con una interfaz en tiempo real los usuarios podrán ver las lecturas de los dispositivos como se ve en la siguiente figura:

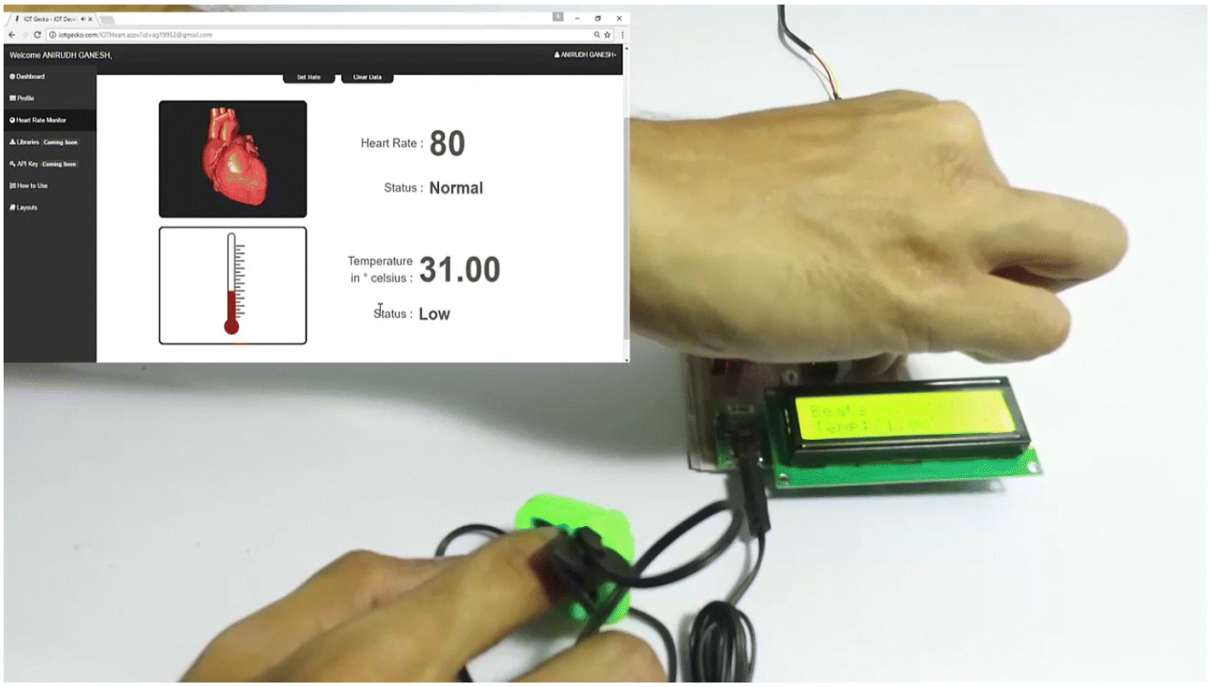


Figura 2: Lectura en tiempo real del sistema WISE

Pese a haber tantos dispositivos IoT de medición la mayoría se han centrado en datos anecdóticos más que en asistencia médica real. En este lugar es donde entran los monitores de pacientes.

Los monitores de pacientes son sistemas médicos que monitorizan la salud de un paciente sobre un periodo de tiempo. Generalmente son usados en hospitales, clínicas y ambulancias, mostrando las constantes vitales como la presión de sangre, frecuencia cardíaca, temperatura…

El problema de estos dispositivos es su elevado precio y su alta barrera de entrada, ya que requieren de usuarios formados.

Figura 3: Ejemplo Patient Monitor

Por lo tanto el proyecto llevado a cabo en este trabajo de fin de grado, consiste en encontrar el punto medio entre estas dos tecnologías. Un dispositivo que no tenga un elevado coste y que sea de un uso sencillo e intuitivo para personas comunes en vez de un personal especializado, en adición de poder utilizar los valores leídos para una monitorización remota del sanitario que corresponda.

3. Objetivos

En este apartado se expondrán los objetivos que ha de cumplir el sistema para ser funcional. Se detallarán tanto los objetivos del sistema como los objetivos personales que se pretenden alcanzar durante la realización del proyecto.

3.1 Objetivos personales

El objetivo principal personalmente viene dado por la motivación personal de crear una posible solución a un problema observado en mi entorno, al ser un núcleo urbano que año tras año ha ido perdiendo personal y medios sanitarios, por lo que algunas consultas de familiares tenían que llevarse a cabo presencialmente en otra ciudad con más medios, en este caso Salamanca.

Este hecho no sería un problema de por sí, pero las consultas no duraban más de unos minutos ya que eran chequeos rutinarios, de tal manera que se perdía más tiempo en los desplazamientos que en la propia consulta. Un problema acrecentado porque las personas que más lo sufren son nuestros familiares mayores, al ser las personas con más consultas médicas.

Siendo el trabajo de fin de grado el único trabajo de la carrera donde el estudiante puede dar rienda suelta, era el momento perfecto para aprender tecnologías nuevas, en este caso desarrollo web, a la vez que se podía buscar un grado de satisfacción personal.

3.2 Objetivos del sistema

Una vez tenido en mente el esquema general del proyecto a desarrollar, se definirán los objetivos de diseño, desarrollo e implementación que se deberán cumplir.

El objetivo principal del sistema es ofrecer una plataforma web a la que puedan acceder médicos y pacientes, consultar datos medidos y llevar a cabo una consulta telemática vía videollamadas.

Los objetivos que deberá cumplir el sistema son, en particular:

- **Gestión de usuarios**: El sistema deberá manejar altas, bajas y modificaciones de usuarios, además de manejar los diferentes roles

- **Medición de datos en tiempo real**: El sistema deberá ser capaz de gestionar la medición de datos de los sensores en tiempo real y mostrar sus valores en la plataforma.

- **Consultas telemáticas por videoconferencia**: El sistema deberá ser capaz de gestionar consultas médicas de manera telemática a través de videoconferencias.

- **Almacenamiento y visualización de datos**: El sistema deberá ser capaz de almacenar un histórico de valores medidos por los sensores y permitir a los usuarios ver los correspondientes. También guardará historiales médicos que podrán ser dados de alta y descargados.

4. Conceptos teóricos

4.1 API REST

Una **API REST** (Application Programming Interface, por sus siglas en inglés, Representational State Transfer) es un conjunto de reglas y convenciones que permite a dos aplicaciones o sistemas intercambiar datos y funcionalidades de manera estandarizada a través de internet.

La API REST se basa en los principios del protocolo **HTTP** y utiliza los métodos y códigos de estado definidos en este protocolo para realizar operaciones sobre los recursos. Los recursos en una API REST son elementos de información, como datos, archivos o funcionalidades, que pueden ser accedidos y manipulados a través de la **API**.

La comunicación entre el cliente y el servidor en una API REST se realiza a través de solicitudes HTTP a unos puntos finales denominados **Endpoints**, donde el cliente envía una solicitud al servidor utilizando los métodos estándar de HTTP, como **GET, POST, PUT, DELETE**, entre otros, y el servidor responde con un código de estado y los datos solicitados, generalmente en formato **JSON** (JavaScript Object Notation).

Una característica importante de las API REST es que son "stateless" o sin estado, lo que significa que cada solicitud realizada por el cliente contiene toda la información necesaria para que el servidor comprenda y responda adecuadamente, sin depender de un estado de sesión previo. Esto permite una mayor escalabilidad y simplifica el desarrollo y mantenimiento de las aplicaciones que utilizan la API REST.

En resumen, una API REST es un conjunto de reglas y estándares que permite a las aplicaciones comunicarse y compartir datos y funcionalidades de manera uniforme a través de internet, utilizando el protocolo HTTP y siguiendo principios como la transferencia de estado representacional y la arquitectura cliente-servidor. Esto facilita la integración de diferentes sistemas y la creación de aplicaciones distribuidas.

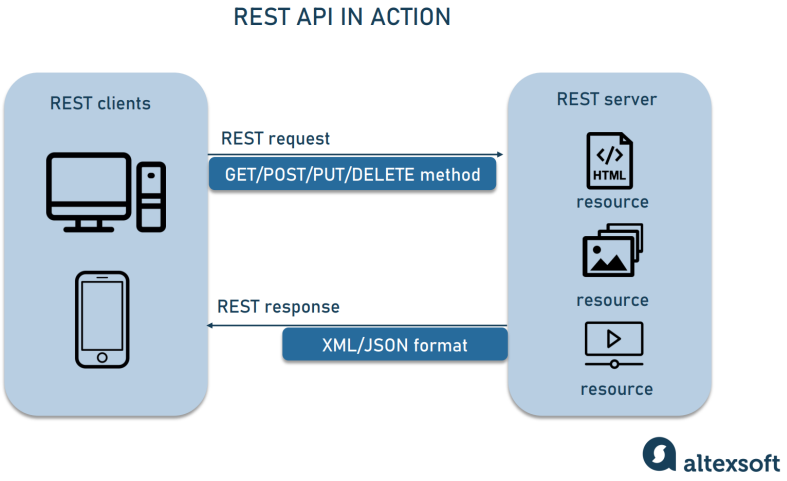


Figura 4: API REST

4.2 Herramientas CASE

Las herramientas CASE (Computer-Aided Software Engineering) o Ingeniería de Software Asistida por Computadora, son un conjunto de aplicaciones informáticas diseñadas para ayudar en el proceso de desarrollo de software. Estas herramientas proporcionan funcionalidades y soporte automatizado para diversas actividades relacionadas con la ingeniería de software.

Estas herramientas están diseñadas para mejorar la productividad y la calidad del desarrollo de software, al proporcionar un entorno integrado y automatizado que facilita las tareas de diseño, implementación y mantenimiento del software. Ayudan a reducir errores, acelerar el proceso de desarrollo y facilitar la colaboración entre los miembros del equipo de desarrollo.

4.3 Tablet médica (Patient Monitor)

El eje principal del trabajo será la tablet médica. Una tablet usual con Android, pero que viene con varios conectores de todo tipo de sensores integrados. A ella se podrán conectar desde termómetros a lectores de orina o electrodos.

Figura 5: Conectores Tablet médica

La tablet también contará con cámara frontal y altavoces para las videoconferencias, y se podrá apoyar en la mesa para que no sea pesada al usuario.

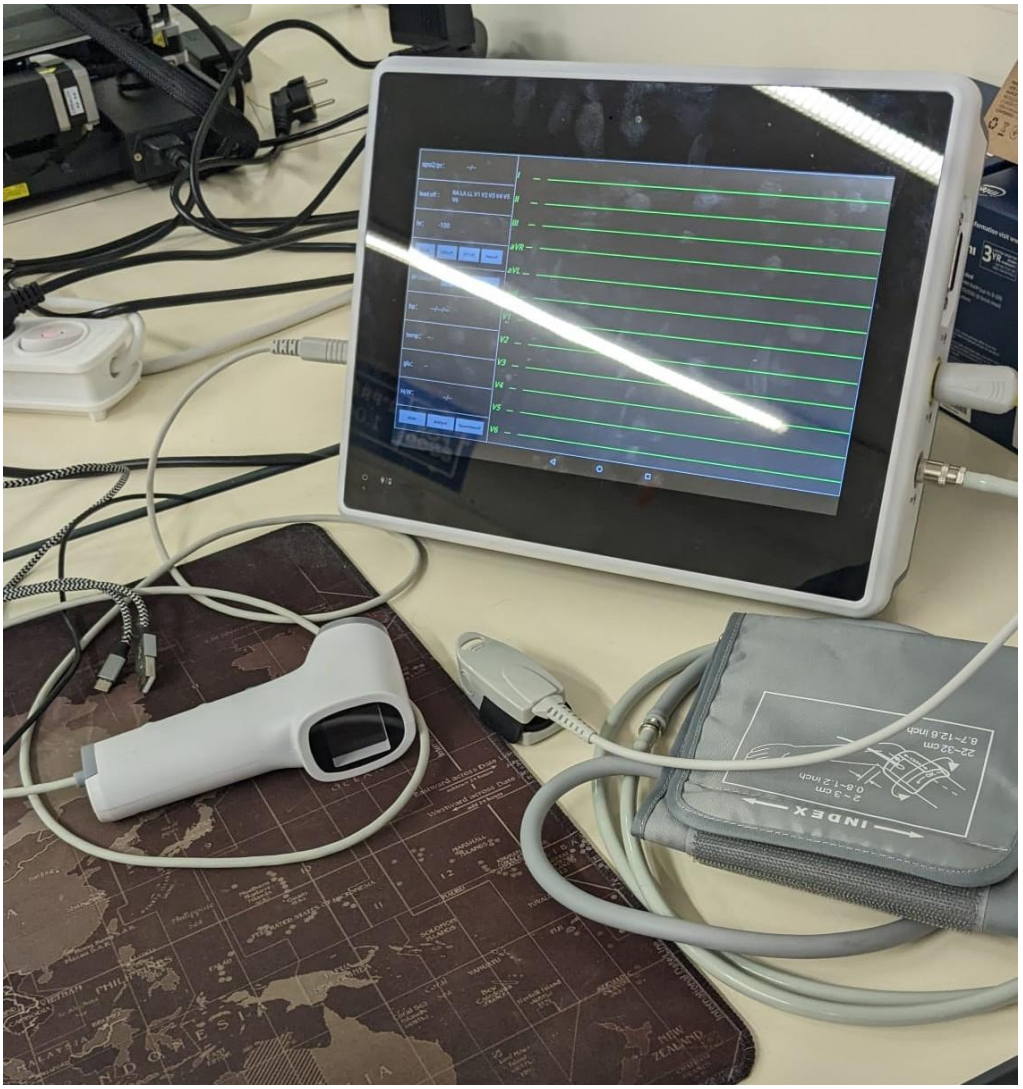


Figura 6: Tablet médica con sensores conectados

5. Técnicas y herramientas

En este apartado se detallarán las técnicas y herramientas de desarrollo empleadas durante el proyecto, incluyendo bibliotecas, módulos y programas software.

5.1 JavaScript

JavaScript es un lenguaje de programación elegido para la aplicación ya que es un lenguaje utilizado principalmente para desarrollar aplicaciones web interactivas. .

Algunas de sus características clave son:

- **Lenguaje interpretado**: JavaScript es un lenguaje interpretado, lo que significa que no necesita ser compilado antes de ser ejecutado. Los navegadores web interpretan directamente el código JavaScript y lo ejecutan en tiempo real, siendo idóneo para el lado del cliente en el servidor.

- **Orientado a objetos**: JavaScript es un lenguaje orientado a objetos, lo que permite crear y manipular objetos que representan entidades o conceptos del mundo real. Proporciona características como encapsulación, herencia y polimorfismo.

- **Tipado dinámico**: JavaScript es un lenguaje de tipado dinámico, lo que significa que las variables no están asociadas a un tipo de dato específico. Las variables pueden contener diferentes tipos de datos en diferentes momentos durante la ejecución del programa.

- **Ampliamente utilizado en desarrollo web**: JavaScript es el lenguaje de programación principal para el desarrollo de aplicaciones web interactivas. Permite la manipulación y modificación del contenido de una página web, la interacción con el usuario y la comunicación con servidores a través de solicitudes.

- **Event-driven**: JavaScript es un lenguaje basado en eventos, lo que significa que puede responder a acciones del usuario o eventos generados por el sistema. Puede capturar eventos como hacer clic en un elemento, mover el mouse o enviar un formulario, y ejecutar funciones específicas en respuesta a esos eventos.

- **Multiplataforma**: JavaScript se ejecuta en la mayoría de los navegadores web modernos en diferentes sistemas operativos, lo que lo convierte en un lenguaje multiplataforma.

- **Facilidad de integración**: JavaScript se puede integrar fácilmente con HTML y CSS, lo que permite la manipulación y modificación dinámica del contenido y la apariencia de una página web. Esto brinda una gran flexibilidad y poder para crear experiencias interactivas en el navegador.

Estas características hacen de JavaScript un lenguaje versátil y potente para el desarrollo web. Se combinará con la siguiente herramienta para ofrecer una plataforma web completa.

5.2 HTML y CSS

HTML (HyperText Markup Language) es el lenguaje de marcado utilizado para crear la estructura y el contenido de una página web. Con HTML, se definen los elementos y etiquetas que representan los diferentes elementos de una página, como encabezados, párrafos, imágenes, enlaces y formularios. Estas etiquetas permiten organizar y presentar la información de manera estructurada, y se pueden utilizar para agregar enlaces, imágenes, videos y otros elementos multimedia a una página web.

CSS (Cascading Style Sheets) es un lenguaje de estilo utilizado para controlar la apariencia y el diseño de una página web. Con CSS, se puede especificar cómo se debe mostrar cada elemento HTML, como el color, la fuente, el tamaño, el espaciado, la posición y otros aspectos visuales. CSS permite separar el contenido y la estructura de una página web de su presentación visual, lo que facilita la creación de estilos consistentes y la aplicación de cambios en el diseño de manera eficiente.

En resumen, HTML se ocupa de la estructura y el contenido de una página web, mientras que CSS se encarga de definir cómo se verá esa estructura y contenido en términos de colores, fuentes, tamaños y diseños. Juntos, HTML y CSS son las herramientas fundamentales para crear y dar formato a páginas web visualmente atractivas y funcionales básicas. Para añadir funcionalidades más complejas, será combinado con el anterior mencionado JavaScript, de manera que con HTML, CSS y JavaScript, tendremos: La página web, el estilo de la página web, y la funcionalidad.

5.3 NodeJS

Node.js es un entorno de ejecución de JavaScript del lado del servidor que permite a los desarrolladores utilizar JavaScript tanto en el navegador como en el servidor. A diferencia de JavaScript en el navegador, Node.js se basa en el motor de JavaScript V8 de Google Chrome y proporciona un entorno que permite ejecutar código JavaScript de forma eficiente en el servidor.

La funcionalidad de Node.js se ve potenciada por su ecosistema de paquetes, conocido como npm (Node Package Manager). npm es uno de los repositorios de paquetes de código abierto más grandes y activos, que permite a los desarrolladores acceder a miles de módulos y paquetes preexistentes. Esto significa que los desarrolladores pueden aprovechar la funcionalidad ya creada por la comunidad para agregar características adicionales a sus aplicaciones de manera rápida y eficiente, ahorrando tiempo y esfuerzo en el desarrollo.

En resumen, Node.js es un entorno de ejecución de JavaScript del lado del servidor que ofrece eficiencia, escalabilidad y un modelo de programación orientado a eventos. Su

capacidad para manejar conexiones concurrentes, junto con su ecosistema de paquetes npm, lo convierte en una opción popular para aplicaciones web en tiempo real, servicios de red y servidores de aplicaciones.

5.4 React

React es una biblioteca de JavaScript desarrollada por Facebook que se utiliza para construir interfaces de usuario interactivas y reutilizables. Se enfoca en la creación de componentes de interfaz de usuario que se actualizan de manera eficiente cuando los datos cambian. React utiliza un enfoque de "renderizado virtual" que optimiza el rendimiento al actualizar solo las partes necesarias de la interfaz de usuario en lugar de toda la página. Además, React permite la creación de aplicaciones de una sola página (SPA) y se integra bien con otras bibliotecas o frameworks.

5.5 Express

Express es un framework web minimalista y flexible para Node.js. Proporciona una capa de abstracción sobre el servidor HTTP de Node.js, simplificando la creación de aplicaciones web y APIs. Express facilita la definición de rutas, el manejo de solicitudes y respuestas HTTP, el manejo de cookies y sesiones, y el procesamiento de datos recibidos del cliente mediante la configuración de middlewares para recibir/responder a las peticiones.

Será utilizado junto al servidor Node.js para la creación de rutas API del servidor backend.

5.6 Core UI

Core UI es una plantilla de React que se usará para tener una interfaz de usuario moderna, bonita y responsiva, ahorrando tiempo de desarrollo. La plantilla elegida ha sido la Free React Admin Template, ya que es una plantilla destinada a webs con visualización de estadísticas y administración. Siendo idónea para el proyecto.

Al ser una plantilla de React, contará con diferentes elementos llamados componentes que se usarán de manera sencilla durante la interfaz de usuario, pudiendo así modificarla en base a las necesidades que vayan surgiendo en la aplicación.

Cuenta con una licencia MIT, permitiendo el uso comercial, la modificación, y distribución del software, mientras se mencione al autor de la plantilla: Łukasz Holeczek.

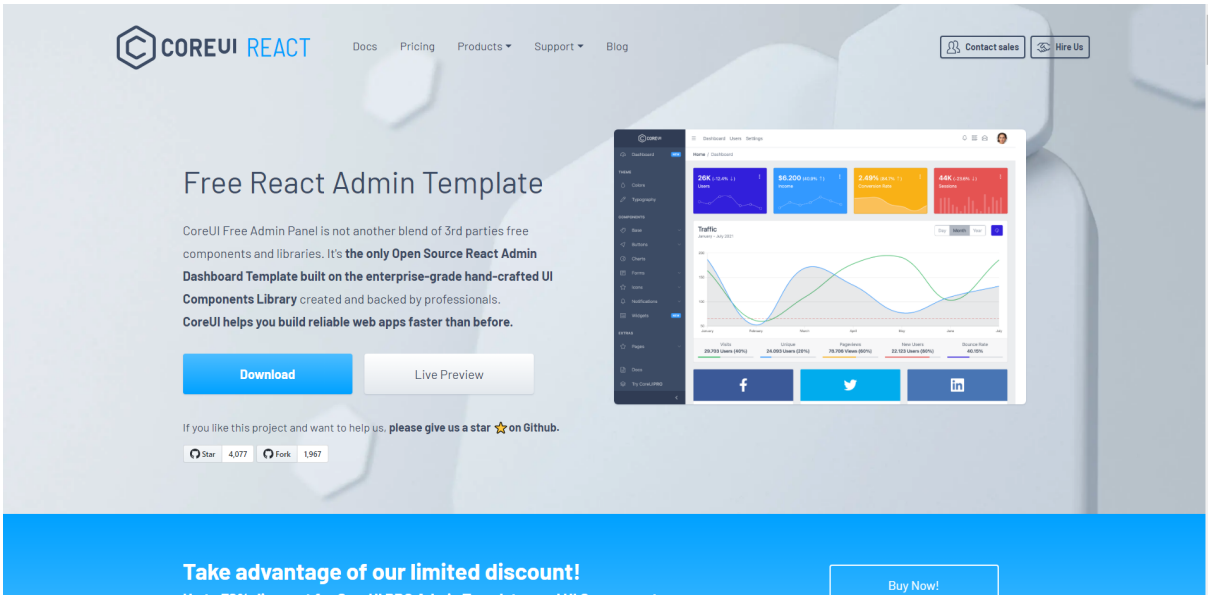


Figura 7: Plantilla Core UI

5.7 JSON

JSON (JavaScript Object Notation) es un formato de intercambio de datos ligero y fácil de leer utilizado para representar información estructurada. Se basa en la sintaxis de los objetos en JavaScript, lo que lo hace compatible con muchos lenguajes de programación.

JSON utiliza una combinación de pares clave-valor para organizar los datos en una estructura jerárquica. Los datos en JSON están representados principalmente por objetos (llaves y valores), matrices (listas de valores) y tipos de datos primitivos como cadenas de texto, números, booleanos y valores nulos.

En el sistema será utilizado como el formato por defecto para el intercambio de datos, ya que de esta manera es ampliamente utilizado en aplicaciones web y APIs, como alternativa a XML.

5.8 MySQL y MySQL Workbench

MySQL es un sistema de administración de bases de datos relacionales. Es un software de código abierto desarrollado por Oracle. MySQL utiliza múltiples tablas dentro de una base de datos, donde cada tabla tiene sus propios atributos y almacenará toda la información necesaria para el sistema, asegurando la persistencia de los datos.

MySQL Workbench es una herramienta visual para el empleo de MySQL, esta herramienta visual permitirá el diseño de nuevas bases de datos, administración de las existentes y consultas a las tablas guardadas.

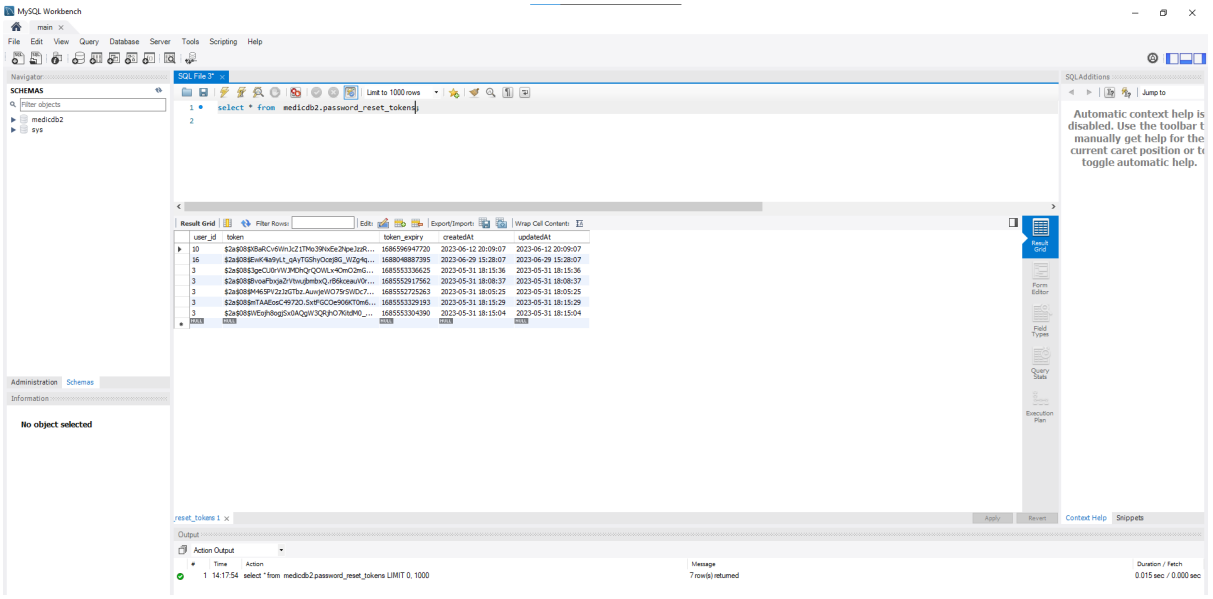


Figura 8: MySQL Workbench 8

5.9 Sequelize

Sequelize es un ORM (Object Relational Mapping) que permite a los usuarios llamar a funciones javascript para interactuar con la base de datos MySQL sin escribir consultas reales. Al estar disponible como módulo npm para Node.js es la opción perfecta para accesos de consultas, alta o baja de información de la base de datos desde el código de manera segura.

5.10 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft que se utiliza principalmente para programar y desarrollar aplicaciones de software. A parte de ser un editor de código, al lateral de la ventana tiene un navegador de ficheros donde se podrá buscar por los directorios, encontrando fácilmente palabras clave en proyectos grande, además de integrar una o más terminales. VSCode es altamente personalizable y extensible, soporta una gran cantidad de lenguajes y un ecosistema de extensiones creadas por la comunidad que tienen el fin de ayudar al desarrollador.

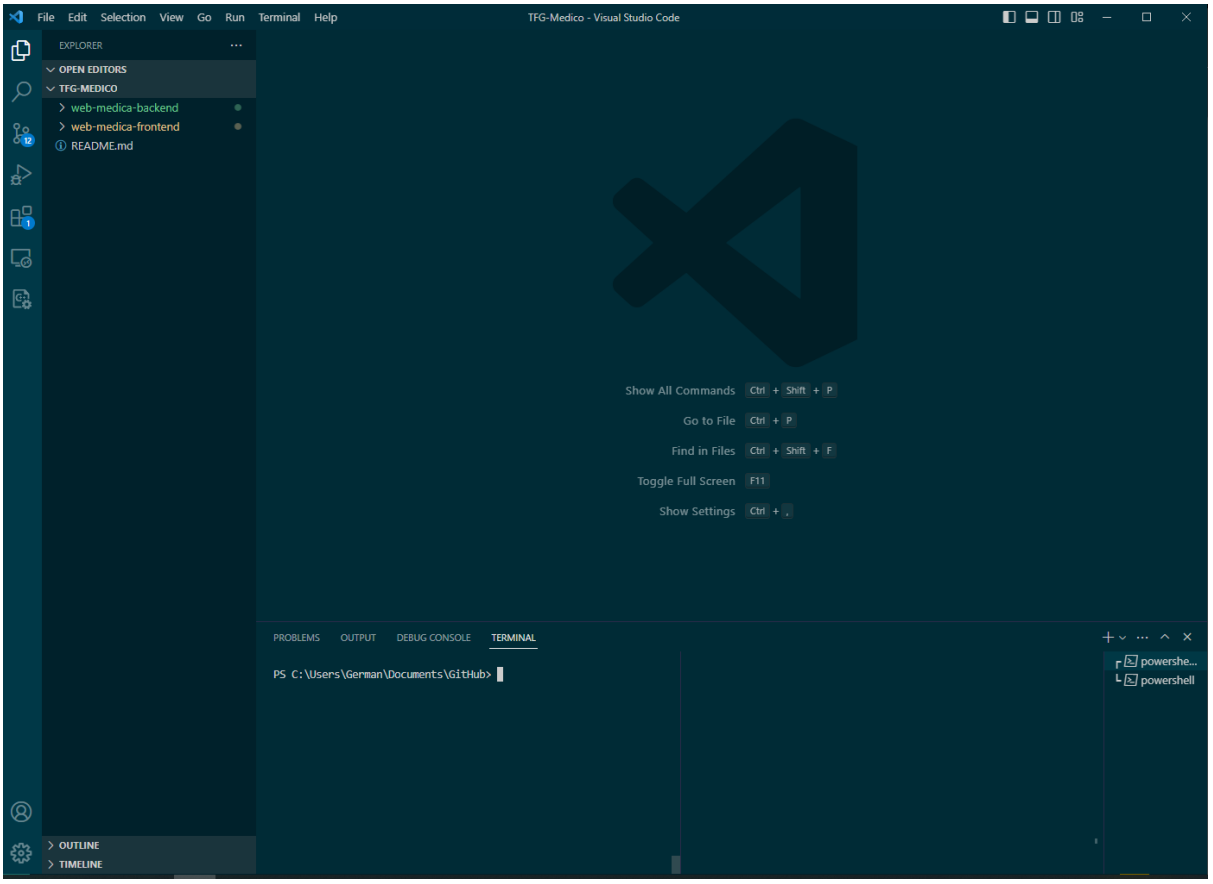


Figura 9: Visual Studio Code

5.11 Swagger y Swagger2pdf

Swagger es un módulo (NPM) de autodocumentación para APIs desarrolladas en Node.js. Usando un fichero de configuración **swagger.js** se le indicará las rutas de los endpoints que queremos documentar y la ruta del archivo de salida. Este archivo de salida será un **swagger_output.json** el cual tendrá información de qué tipo de petición recibe un endpoint (GET, POST, PUT, DELETE), los códigos de respuesta, el nombre de la ruta, y un campo de descripción que se puede rellenar manualmente.

Pese a que el formato .json sea legible a ojos del programador, lo mejor será exportarlo a un formato mucho más visual como pdf. Para llevar a cabo esta acción se ha hecho uso de la herramienta **Swagger2pdf** , se le pasará por argumentos el fichero json generado por swagger, y creará un fichero **api.pdf** en el cual estarán documentados de manera visual los endpoints del programa.

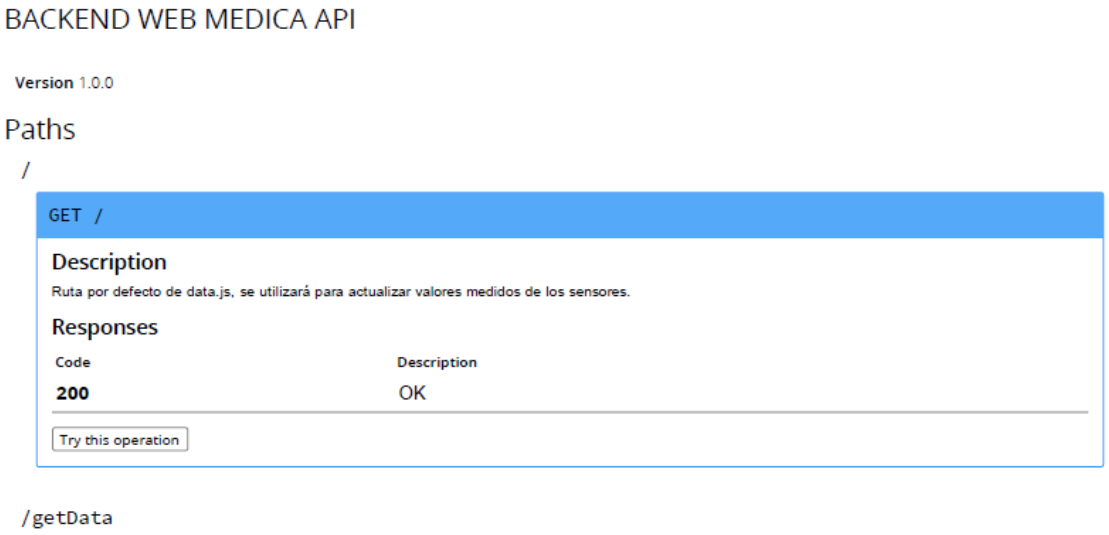


Figura 10: Fichero api.js con documentación de los endpoints

5.12 Android Studio

Android Studio es un entorno de desarrollo integrado (**IDE**) creado por Google, diseñado específicamente para facilitar la creación, desarrollo y depuración de aplicaciones móviles para dispositivos Android. Siendo la tablet médica un dispositivo Android, se ha usado este IDE para el desarrollo de la aplicación de la tablet. El entorno permite tanto el uso de Java como de Kotlin, pero se ha optado por usar Java ya que el kit de desarrollo (SDK) de la tablet venía en Java.

Android Studio permite ejecutar las aplicaciones creadas en entornos virtuales Android si no se dispone de un dispositivo Android conectado al ordenador vía usb en modo esclavo.

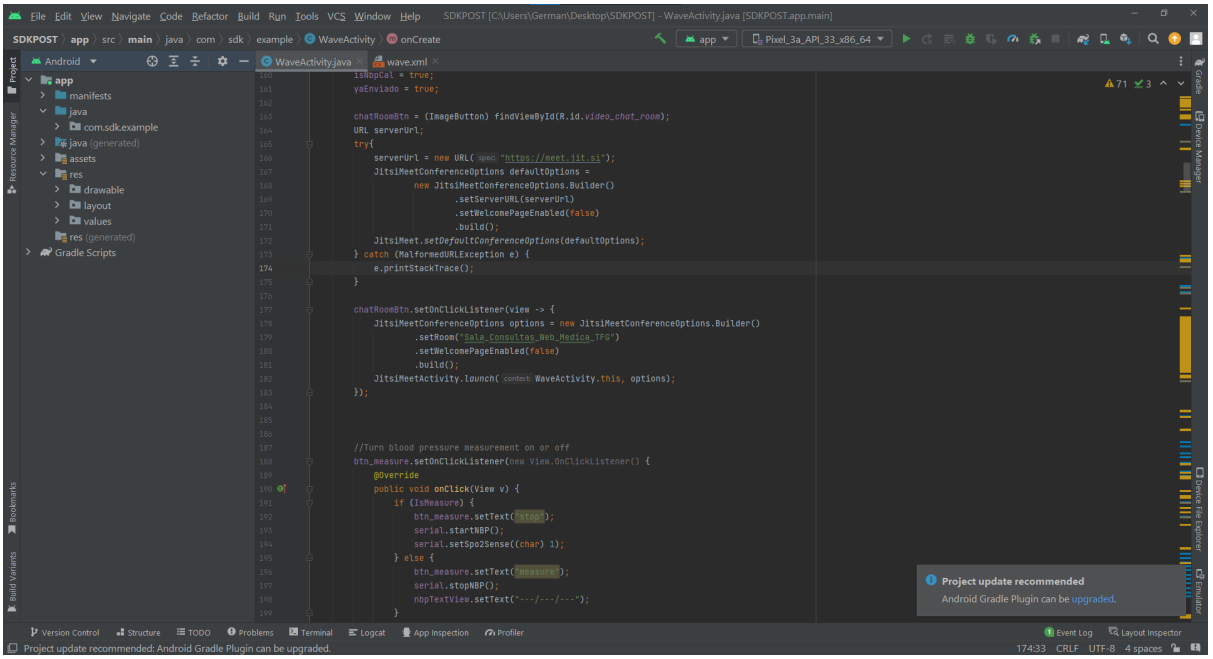


Figura 11: Android Studio

5.13 Git y GitKraken

Git es un software de control de versiones distribuido. Es una herramienta que fure creada pensando en la eficiencia de mantenimiento de diferentes versiones de aplicaciones en un repositorio de código.

GitKraken es una aplicación de escritorio que facilita mucho el uso de control de versiones mediante Git, pero haciendo uso de una interfaz de usuario amigable y de sencilla comprensión. Durante el desarrollo se ha usado la versión premium de GitKraken ya que venía incluida en el paquete de desarrollo de estudiantes de GitHub, al cual se puede acceder mediante la Universidad de Salamanca.

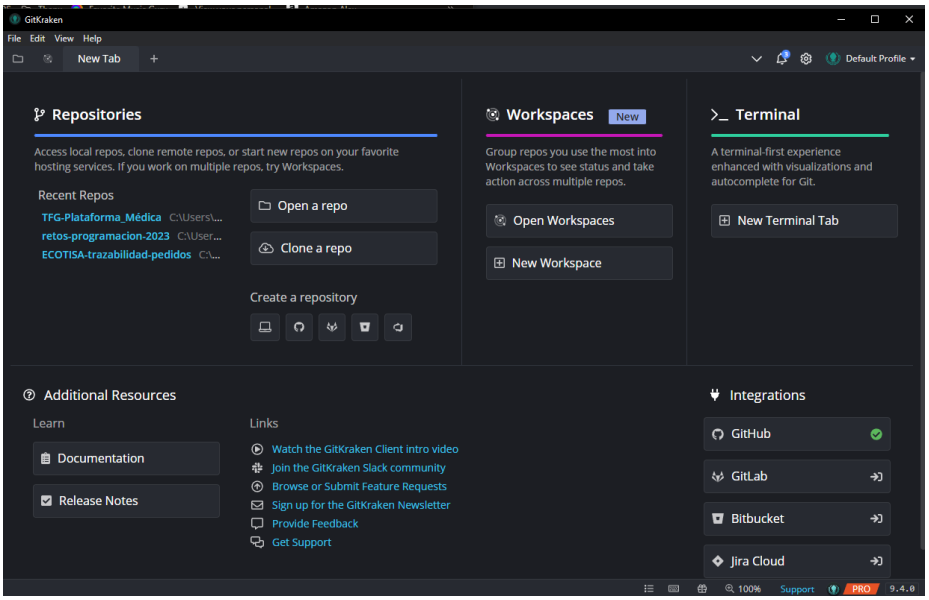


Figura 12: GitKraken

5.14 Visual Paradigm

Visual Paradigm es una herramienta CASE, centrada en el lenguaje unificado de modelado (**UML**) que permite el soporte del ciclo de vida completo del desarrollo de software: desde análisis y diseño, a través de la representación de todo tipo de diagramas UML.

Para el uso de esta herramienta se ha hecho uso de la licencia académica proporcionada por la Universidad de Salamanca.

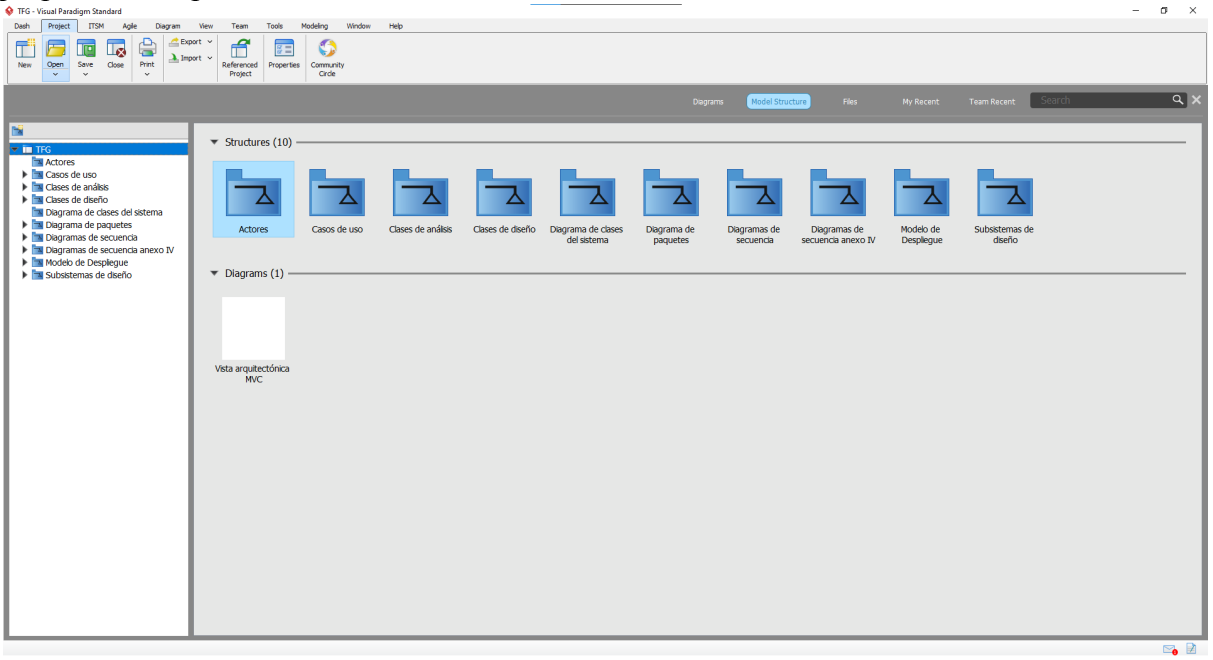


Figura 13: Visual Paradigm

5.15 EzEstimate

Es la herramienta utilizada para estimar la duración del proyecto en base a los puntos de casos de uso, complejidad, actores y los factores de complejidad técnica y de entorno.

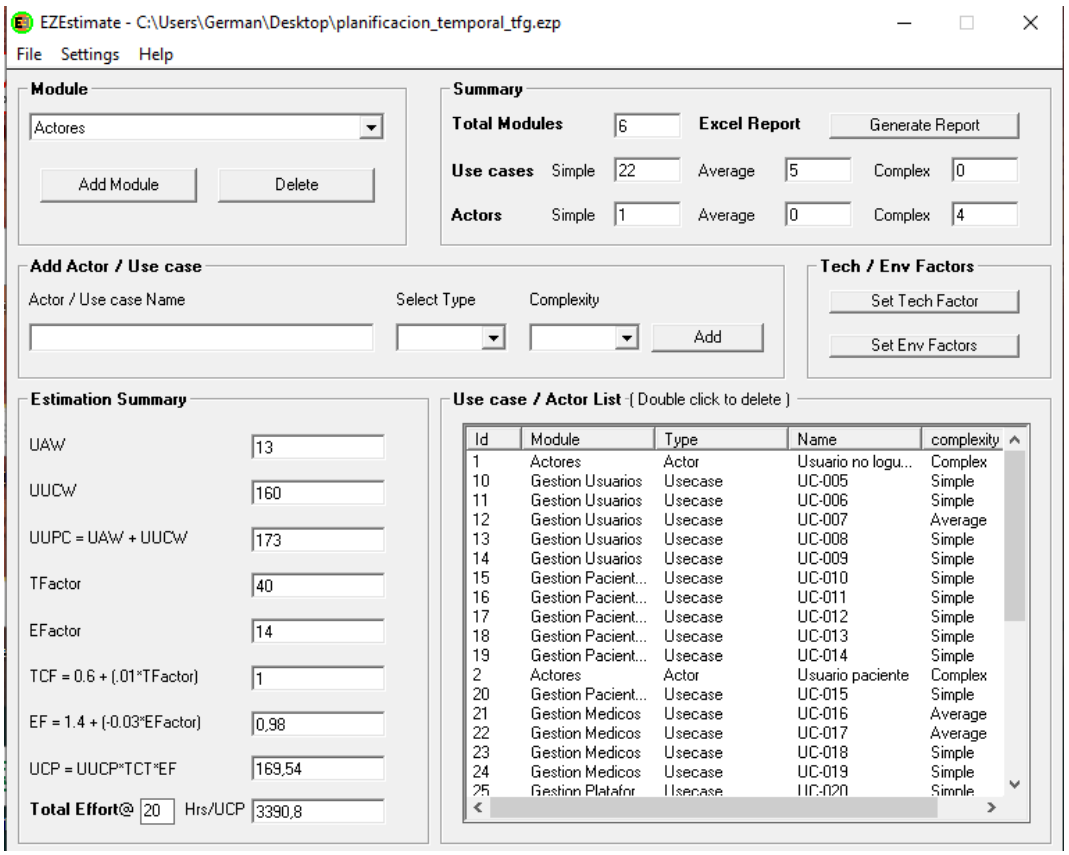


Figura 14: EZEstimate

5.16 Microsoft Project

Microsoft Project es un software de gestión de proyectos que sirve para la asignación de los recursos y tareas del proyecto, tratando de obtener un tiempo similar al estimado en la anterior herramienta, EZEstimate.

Además permite ver de manera visual el plan del proyecto mediante un diagrama de Gantt.

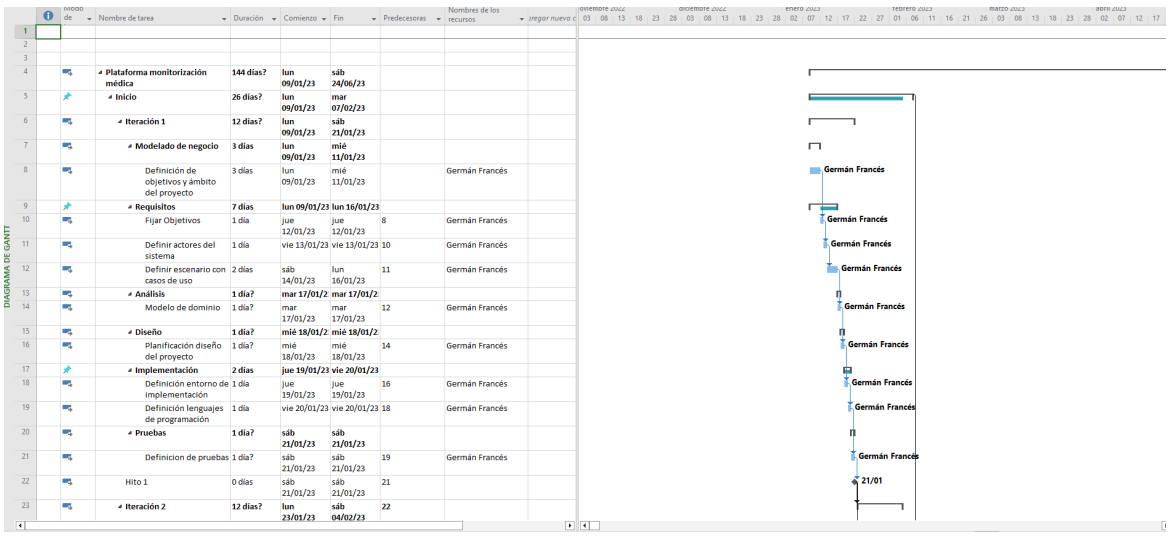


Figura 15: Microsoft Project

6. Aspectos relevantes del proyecto

En este apartado se explicarán los aspectos más destacados del desarrollo del proyecto, cada fase estará detallada en los anexos, desde la planificación temporal hasta la versión final de implementación.

6.1 Marco de trabajo

Como marco de trabajo se ha utilizado el Proceso Unificado, un marco de desarrollo de software dirigido por casos de uso que se caracteriza por su desarrollo iterativo e incremental y que está centrado en la arquitectura.

De esta manera permite que:

- Al estar dirigido por casos de uso, estos capturarán los requisitos funcionales, sus relaciones y el contenido en las iteraciones definidas.

- Al tener un desarrollo iterativo e incremental, en cada fase del PU, se harán varias iteraciones, cada iteración incrementará el desarrollo del sistema, eliminando errores y añadiendo mejoras y funcionalidades respecto a las iteraciones anteriores.

- Al estar centrado en la arquitectura, no existirá un modelo único que cubra todos los aspectos del sistema, sino que existirán múltiples modelos y vistas que definan la arquitectura del software. De manera que se partirá de una arquitectura inicial que se irá refinando con cada iteración.

Para ir más en profundidad, las cuatro fases del Proceso Unificado se detallarán a continuación, de nuevo, cada una de estas fases podrá tener varias iteraciones que irán mejorando el producto:

- **Inicio**: Fase inicial del proyecto donde se definirá el negocio, se presentará un modelo, plazos temporales y viabilidad.

- **Elaboración**: Se obtendrá una versión más detallada del proyecto, aparecerán nuevos requisitos y se ajustan las estimaciones.

- **Construcción**: Se desarrollará el sistema hasta llegar al prototipo que cumpla los requisitos mínimos.

- **Transición**: Fase final del proyecto donde el sistema debe estar preparado para despliegue y pruebas por el usuario final.

La siguiente figura ilustra las diferentes etapas del desarrollo de software y permite identificar la superposición de tareas a lo largo de las etapas, así como la carga de trabajo correspondiente en cada una de ellas.

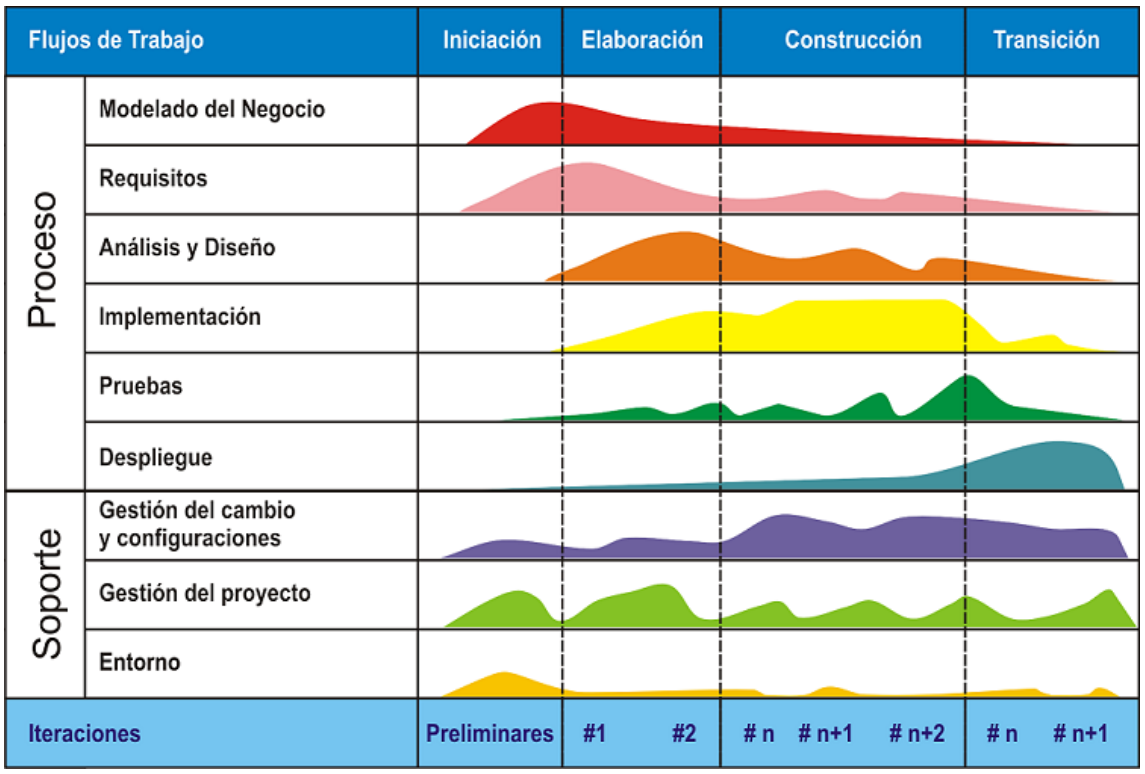


Figura 16: Fases del Proceso Unificado

6.2 Estimación temporal

En la primera etapa del proyecto, la tarea inicial fue llevar a cabo la estimación temporal del proyecto teniendo en cuenta los actores involucrados en el sistema, los casos de uso del proyecto y los factores de complejidad técnica y de entorno.

Este apartado está detallado en el "Anexo I: Plan de Proyecto"

Una vez recogidos los valores de todos los parámetros, se introdujeron en la herramienta EZEstimate para obtener la estimación temporal del proyecto.

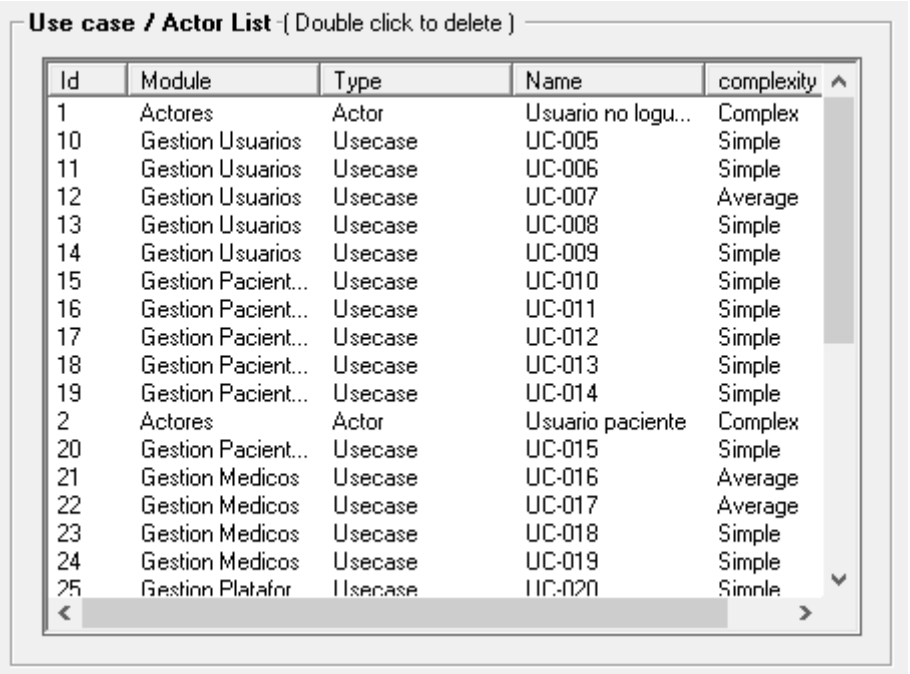


Figura 17: Casos de uso y Actores por módulos

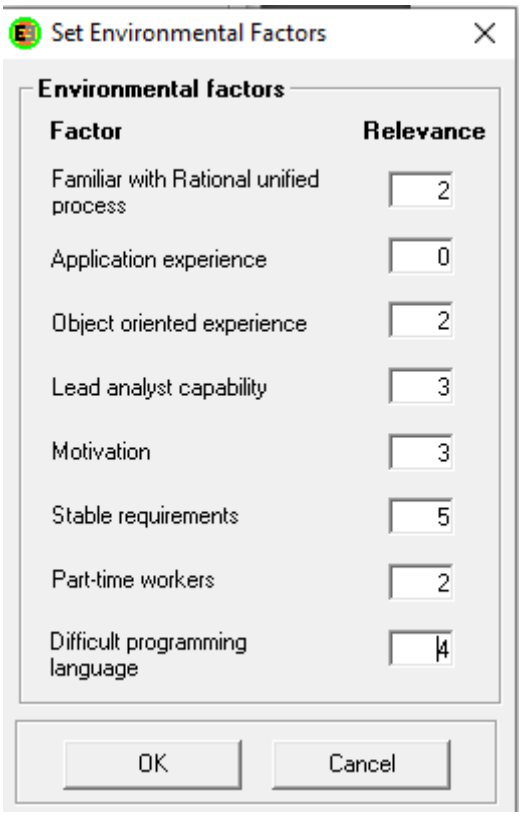


Figura 18: Factores de entorno

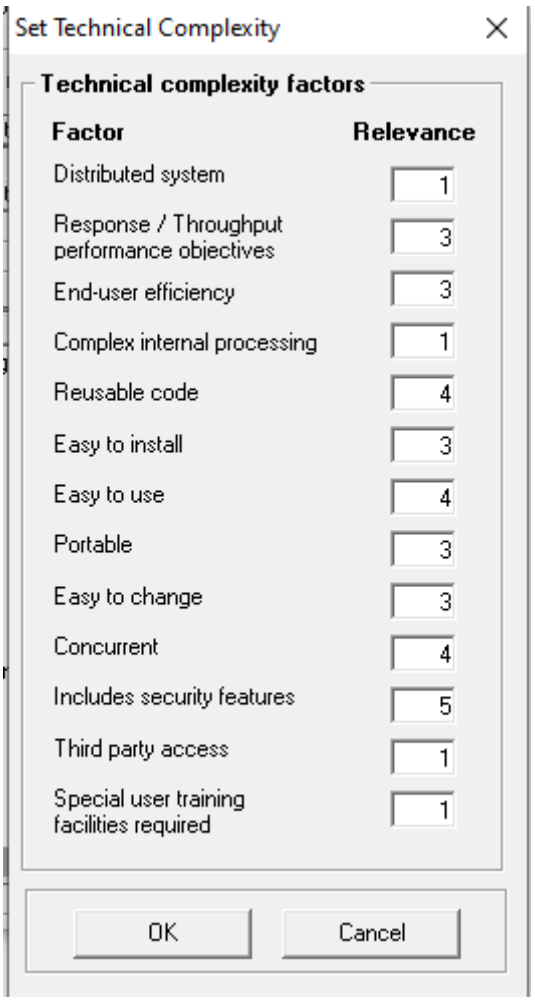


Figura 19: Factores de complejidad técnica

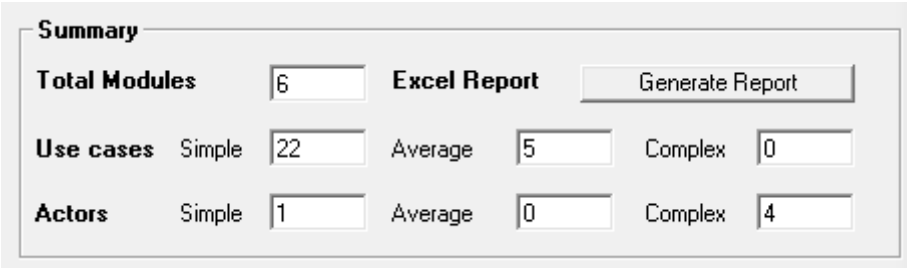


Figura 20: Resumen de EZEstimate

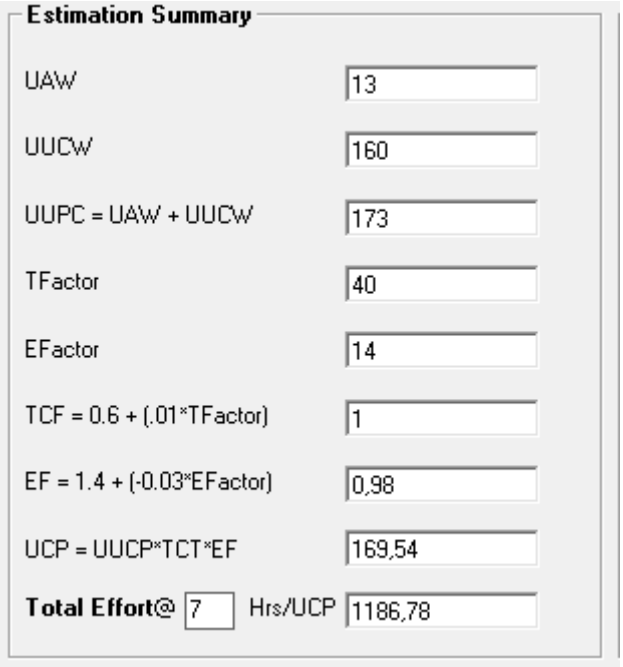


Figura 21: Resumen de la estimación

Se obtiene una estimación de **1186,78h.** Estimando que se va a trabajar unas **8 horas por día**, un único desarrollador, el proyecto duraría **148,34 días** de desarrollo.

6.3 Planificación temporal

Una vez conocida la estimación temporal, se podrá hacer la planificación para saber qué tareas desarrollar y en qué orden.

Para realizar la planificación se ha optado por utilizar la herramienta Microsoft Office Project. Se establecerán las fases y sus iteraciones según el modelo de Proceso Unificado, ya que cada fase podrá tener una serie de iteraciones donde se desarrollará el proyecto de manera incremental, mejorando en cada versión.

Estas fases, como se vieron en la Figura 16, serán:

- **Modelado de negocio**: Se realizará un análisis del proyecto a desarrollar, una investigación de viabilidad y búsqueda de recursos sobre sistemas similares o competencias.

- **Requisitos**: Se fijarán los requisitos y objetivos que tendrá el proyecto.

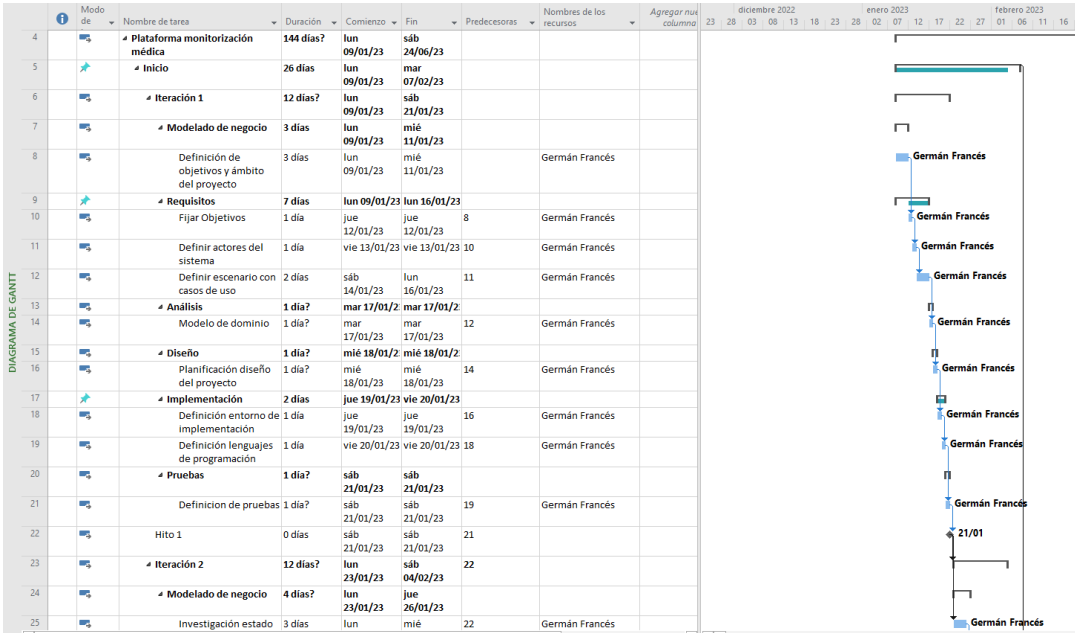
- **Análisis**: Se analizarán los requisitos establecidos en el punto anterior.

- **Diseño**: Se comenzará a plantear una implementación del sistema, dividiéndolo en componentes y determinando cómo funcionarán entre ellos.

- **Implementación**: Implementación del sistema y programación utilizando las herramientas necesarias.

- **Pruebas**: Realización de pruebas de todo el sistema verificando su correcto funcionamiento y obtener feedback sobre posibles mejoras.

Para obtener más información de este apartado, consultar el "Anexo I"



	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
4		**Plataforma monitorización médica**	**144 días?**	**lun 09/01/23**	**sáb 24/06/23**		
5		**Inicio**	**26 días**	**lun 09/01/23**	**mar 07/02/23**		
6		**Iteración 1**	**12 días?**	**lun 09/01/23**	**sáb 21/01/23**		
7		**Modelado de negocio**	**3 días**	**lun 09/01/23**	**mié 11/01/23**		
8		Definición de objetivos y ámbito del proyecto	3 días	lun 09/01/23	mié 11/01/23		Germán Francés
9		**Requisitos**	**7 días**	**lun 09/01/23**	**lun 16/01/23**		
10		Fijar Objetivos	1 día	jue 12/01/23	jue 12/01/23	8	Germán Francés
11		Definir actores del sistema	1 día	vie 13/01/23	vie 13/01/23	10	Germán Francés
12		Definir escenario con casos de uso	2 días	sáb 14/01/23	lun 16/01/23	11	Germán Francés
13		**Análisis**	**1 día?**	**mar 17/01/2**	**mar 17/01/2**		
14		Modelo de dominio	1 día?	mar 17/01/23	mar 17/01/23	12	Germán Francés
15		**Diseño**	**1 día?**	**mié 18/01/2**	**mié 18/01/2**		
16		Planificación diseño del proyecto	1 día?	mié 18/01/23	mié 18/01/23	14	Germán Francés
17		**Implementación**	**2 días**	**jue 19/01/23**	**vie 20/01/23**		
18		Definición entorno de implementación	1 día	jue 19/01/23	jue 19/01/23	16	Germán Francés
19		Definición lenguajes de programación	1 día	vie 20/01/23	vie 20/01/23	18	Germán Francés
20		**Pruebas**	**1 día?**	**sáb 21/01/23**	**sáb 21/01/23**		
21		Definicion de pruebas	1 día?	sáb 21/01/23	sáb 21/01/23	19	Germán Francés
22		Hito 1	0 días	sáb 21/01/23	sáb 21/01/23	21	
23		**Iteración 2**	**12 días?**	**lun 23/01/23**	**sáb 04/02/23**	**22**	
24		**Modelado de negocio**	**4 días?**	**lun 23/01/23**	**jue 26/01/23**		
25		Investigación estado	3 días	lun	mié	22	Germán Francés

Figura 22: Tareas Microsoft Project

Como se puede observar en las tareas iniciales, la planificación general del proyecto llevará a cabo unos 144 días, lo cual deja unos 4 días de holgura respecto a la estimación temporal calculada en el anterior apartado.

6.4 Especificación de requisitos

Una vez finalizada la estimación temporal y la planificación de tareas en base a ese tiempo, se puede pasar a la fase de especificación de requisitos donde se definirán los objetivos del sistema, requisitos de información, requisitos funcionales y requisitos no funcionales. Para ello, se utilizó la metodología Durán y Bernárdez para elicitación de requisitos de un sistema software.

Para obtener más información de este apartado, consultar el "Anexo II"

A continuación se hará un resumen de este documento citando algunos ejemplos.

6.4.1 Participantes

El proyecto cuenta con tres participantes, de los cuales dos son los tutores y uno es el alumno. Todos pertenecen a la misma organización, la Universidad de Salamanca (USAL).

- **Francés Tostado, Germán**: alumno y autor del proyecto

- **Cruz García, David**
- **Villarrubia González, Gabriel**

6.4.2 Objetivos del sistema

Se expondrán los objetivos que debe cumplir el sistema para satisfacer los requisitos planteados.

- **Gestión de Usuarios**
- **Medición de datos en tiempo real**
- **Consultas telemáticas por videoconferencia**
- **Almacenamiento y visualización de datos**

Los objetivos estarán detallados en el Anexo II, en tablas como la siguiente:

OBJ-0001	**Gestión de Usuarios**
Versión	1.0
Autores	Germán Francés Tostado
Fuentes	David Cruz García Gabriel Villarrubia González
Descripción	El sistema deberá gestionar los usuarios de plataforma (altas, bajas y modificaciones) y manejar los diferentes roles, por ejemplo: - Médicos: Tendrán acceso a la lista completa de pacientes, así como la visualización de estadísticas generales, y subida y bajada de informes médicos. - Pacientes: Tendrán acceso a sus propias mediciones y sus datos personales. - Admin: Tendrán acceso a la lista completa de médicos y podrá dar de alta nuevos médicos y de baja a médicos ya existentes
Subobjetivos	Ninguno
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 1: OBJ-0001 Gestión de Usuarios

6.4.3 Requisitos de información

Los requisitos de información indicarán qué datos debe almacenar el sistema

- **Información sobre usuarios**
- **Información sobre pacientes**
- **Información sobre roles de usuario**
- **Información sobre roles**
- **Información sobre tokens de recuperación de contraseña**
- **Información sobre historiales médicos**
- **Información sobre últimos datos medidos**

Igual que el apartado anterior, los requisitos de información estarán detallados en el Anexo II, en tablas como la siguiente:

IRQ – 0001	**Información sobre usuarios**
Versión	1.0
Autores	Germán Francés Tostado
Fuentes	David Cruz García Gabriel Villarrubia González
Dependencias	OBJ-0001 Gestión de usuarios
Descripción	El sistema deberá almacenar información sobre los usuarios registrados en el sistema
Datos específicos	- ID - DNI - Nombre - Apellidos - Email - Fecha de nacimiento - Número Seguridad Social - Dirección - Ciudad - País - Código postal - Contraseña
Importancia	Alta
Urgencia	Inmediato
Estado	Validado
Estabilidad	Alta
Comentarios	Sin comentarios

Tabla 2: IRQ-0001 Información sobre usuarios

6.4.4 Requisitos funcionales

Los requisitos funcionales indicarán los servicios que proporciona el sistema y cómo se comportará en cada situación que corresponda.

Durante la fase de dominio del problema se ha dividido el sistema en varios paquetes, y cada paquete contendrá casos de uso relacionados entre ellos.

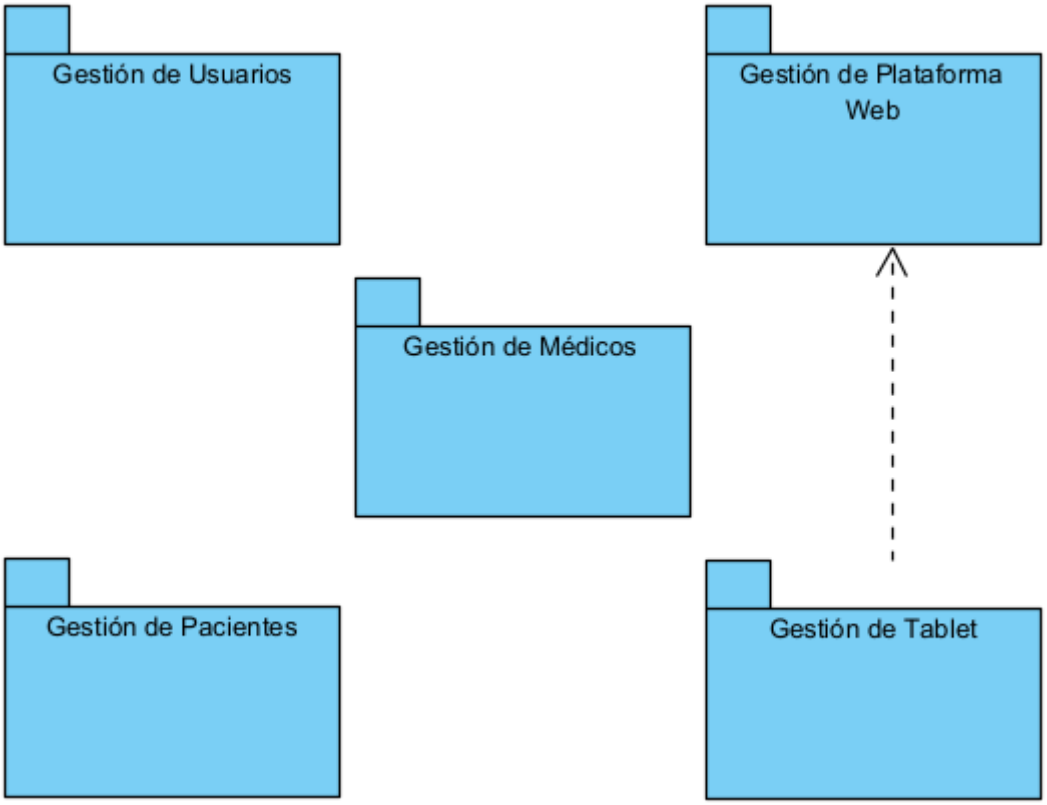


Figura 23: Diagrama de paquetes

Con el sistema divide en paquetes, se definirán los actores que interactuarán en el sistema, donde cada uno realizará funciones determinadas:

- **Usuario no logueado**: Usuario por defecto al conectarse a la plataforma, este usuario podrá registrarse, acceder a la plataforma o solicitar una recuperación de contraseña en caso de estar registrado y no acordarse de la misma.
- **Usuario logueado**: Usuario que ha accedido al sistema mediante usuario y contraseña. Podrá realizar determinadas acciones en base a su rol.
- **Usuario paciente**: Especialización del usuario logueado que hace referencia a un usuario con rol de paciente en la plataforma. Podrá ver sus propias mediciones, datos y acceder a la sala de consultas.
- **Usuario médico**: Especialización del usuario logueado que hace referencia a un usuario con rol de médico. Podrá ver las últimas mediciones del sistema, la lista de pacientes, los historiales médicos y acceder a la sala de consultas, aparte de editar sus propios datos.
- **Usuario admin**: Especialización del usuario logueado que hace referencia a un usuario con rol de admin. Este usuario será único y podrá gestionar los usuarios médicos.

- **Sistema**: Usuario que hace referencia al sistema. Realizará ciertas acciones automáticamente como actualización de datos y sincronización entre las plataformas web y tablet.

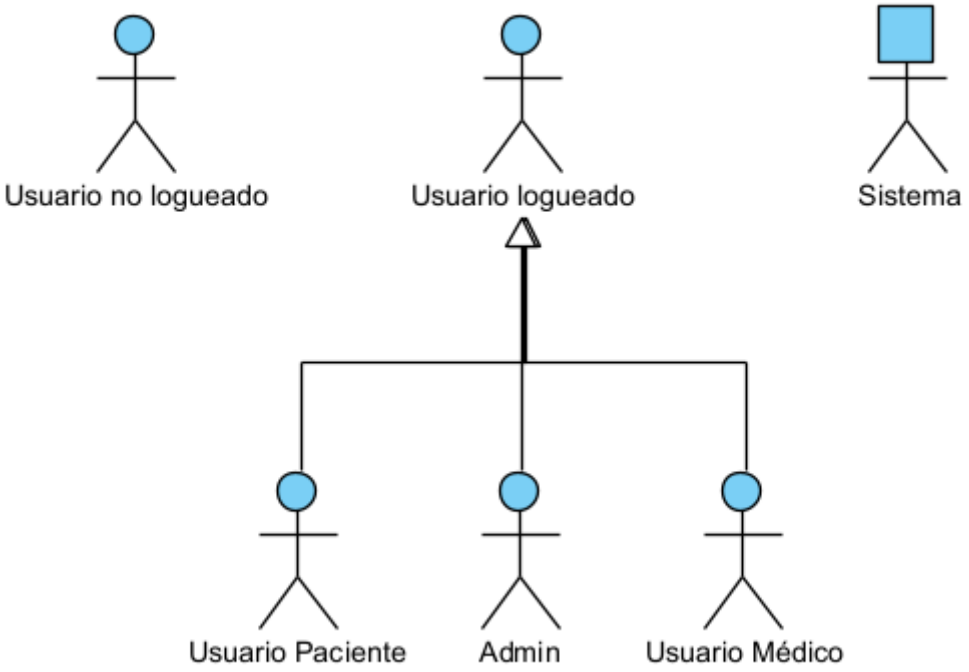


Figura 24: Jerarquía de actores

Como se mencionó anteriormente, los casos de uso se dividirán en los paquetes del sistema. A continuación se mostrarán los casos de uso del paquete Gestión de Usuarios, y una tabla de ejemplo de un caso de uso. El resto de casos de uso estarán detallados en el Anexo II.

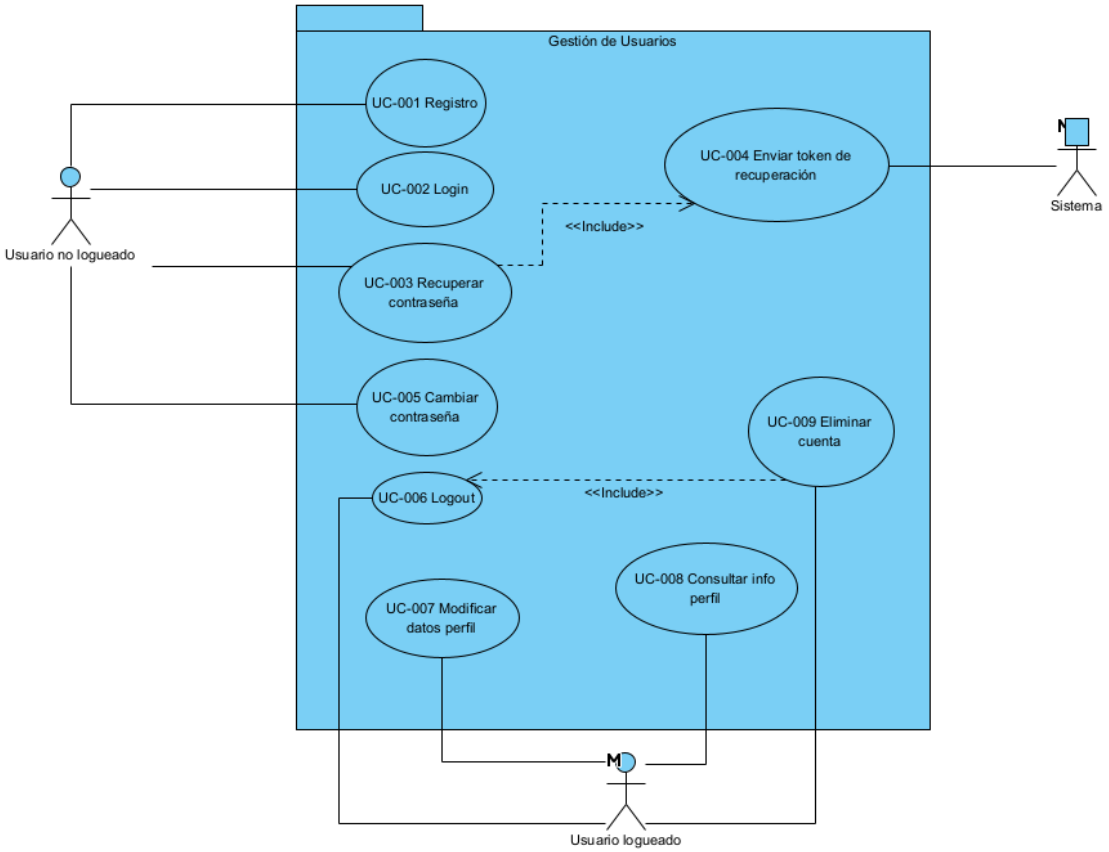


Figura 25: Diagrama de casos de uso Gestión de Usuarios

UC - 0001	**Registro**	
Versión	1.0	
Autores	Germán Francés Tostado	
Fuentes	David Cruz García Gabriel Villarrubia González	
Dependencias	-{OBJ-0001} Gestión de Usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el actor “ACT-001 Usuario no logueado” quiera registrarse en el sistema.	
Precondición	El actor “ACT-001 Usuario no logueado” no ha iniciado sesión en el sistema y no tiene una cuenta.	
Secuencia normal	**Paso**	**Acción**
	1	El actor “ACT-001 Usuario no logueado” solicita registrarse en el sistema
	2	El sistema muestra un formulario a rellenar con los datos del actor “ACT-001 Usuario no logueado”

	3	El actor “ACT-001 Usuario no logueado” rellena los datos.
	4	El sistema valida los datos introducidos.
	5	El sistema almacena los datos introducidos y redirige al actor “ACT-001 Usuario no logueado” a la página principal.
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
	4	Si los datos no son válidos, el sistema informa al usuario y se regresa al paso 2.
	5	Si el usuario o el email ya están dentro del sistema, o sucede algún error, el sistema informará del mismo y el caso de uso queda sin efecto.
Importancia	Alta	
Urgencia	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Sin comentarios	

Tabla 3: UC-0001 Registro

6.4.5 Requisitos no funcionales

Además de las funcionalidades indicadas con los requisitos anteriores, el sistema tendrá que disponer de servicios y funciones que permitan garantizar distintas características como pueden ser entre otras, rendimiento, seguridad, portabilidad.

A continuación se muestran los requisitos no funcionales necesarios para satisfacer estas características:

- **Disponibilidad**
- **Seguridad**
- **Protección de datos**
- **Portabilidad**
- **Concurrencia**
- **Eficiencia**
- **Facilidad de Uso**
- **Despliegue**

6.4.6 Matriz de Rastreabilidad

Una vez especificados todos los requisitos y los objetivos del sistema, se crea la matriz de rastreabilidad para ver la relación entre ellos.

6.5 Análisis de Requisitos

En la fase de análisis del proyecto se ha llevado a cabo un análisis de todos lso requisitos definidos en la fase anterior, para ello se ha terminado el modelo de dominio, realización de diagramas de secuencia de los casos de uso, clases de análisis y una vista de arquitectura del modelo de análisis.

Para más información sobre este apartado consultar el Anexo III.

6.5.1 Modelo de dominio

El modelo de dominio del sistema recogerá las clases que representan el mundo real y las relaciones que existen entre ellas, de manera que queden plasmadas las entidades que sea necesario gestionar y sus atributos.

El diagrama de clases del sistema desarrollado es el siguiente:

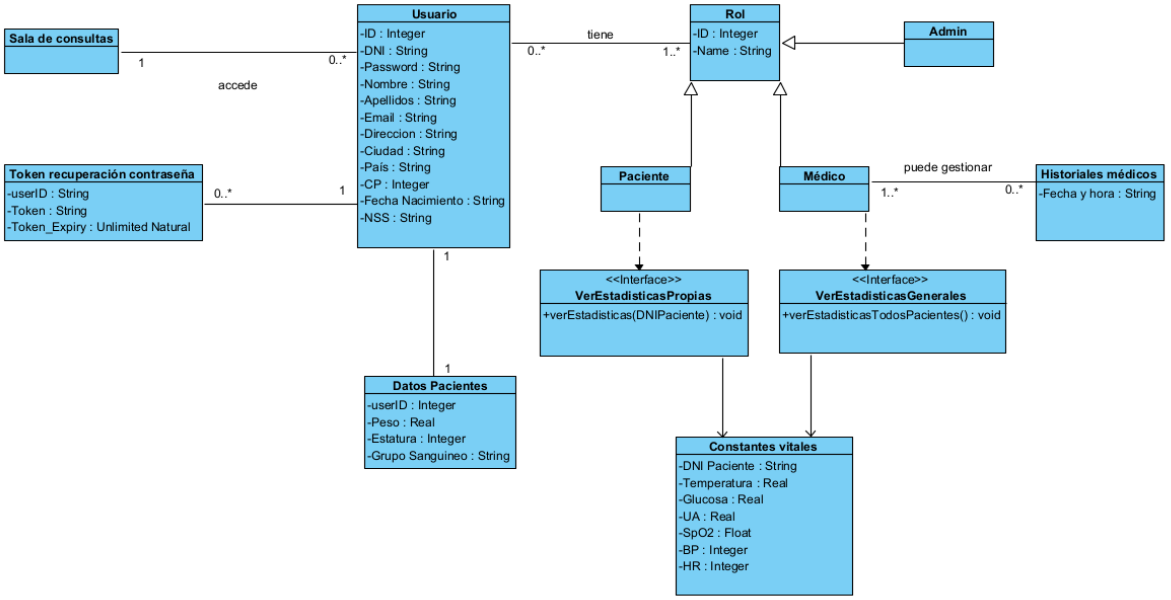


Figura 26: Diagrama de clases del sistema

6.5.2 Realización de diagramas de secuencia de casos de uso

Los diagramas de secuencia de cada caso de uso estarán divididos en los paquetes del sistema creados anteriormente. Estos diagramas nos permitirán tener una mejor representación de las transiciones entre los estados en los que se encuentra el sistema en cada momento.

Como ejemplo se mostrará uno de los diagramas de secuencia, el resto estarán en el Anexo III.

- UC-016 Añadir Médico

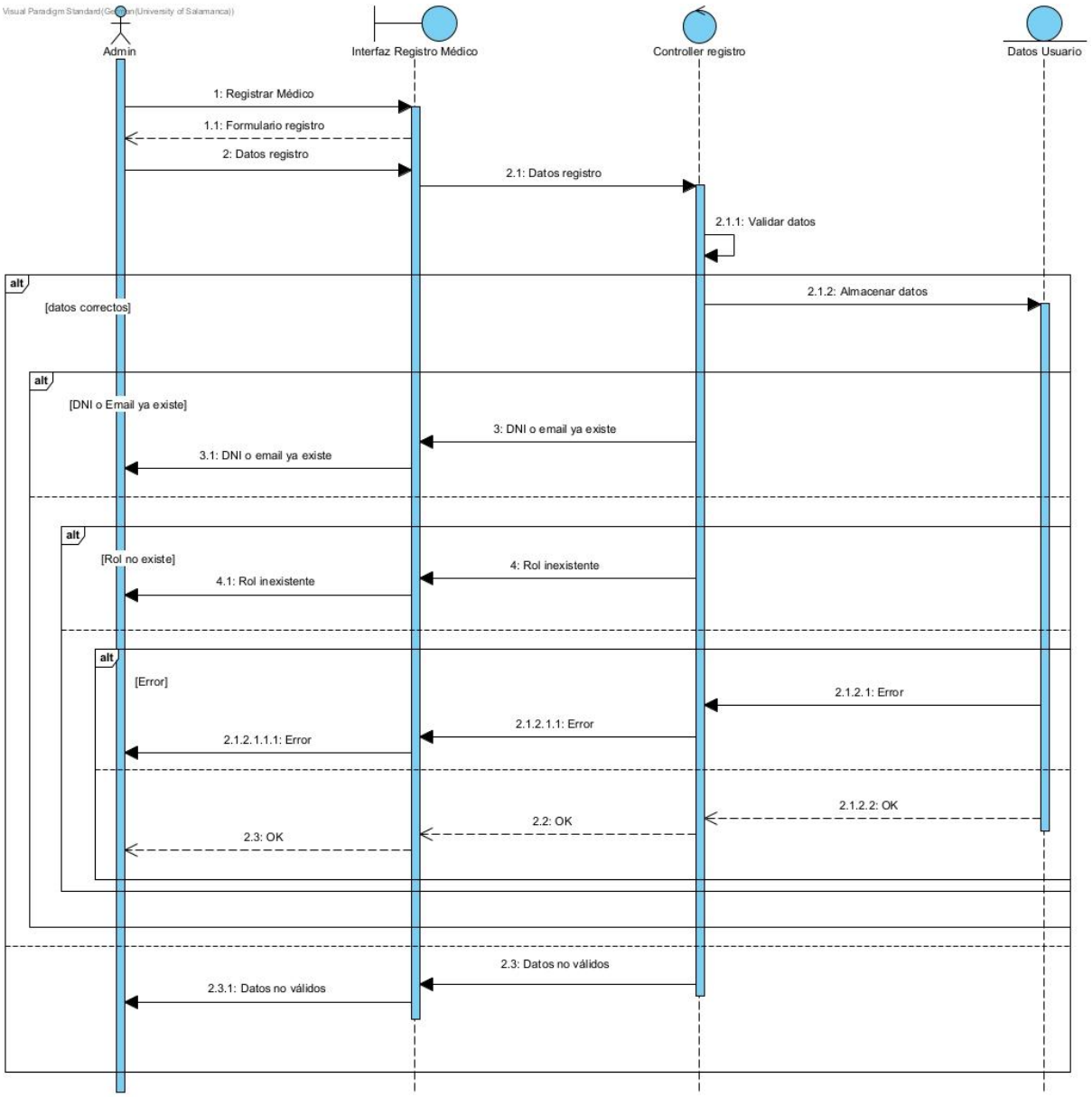


Figura 27: Diagrama de secuencia Añadir Médico

6.5.3 Clase de análisis

En el apartado de clases de análisis se han realizado diagramas de comunicación con el fin de permitir la visualización de la interacción y distribución de los componentes (interfaces, controladores y datos) del sistema presentados en los diagramas de secuencia previos.

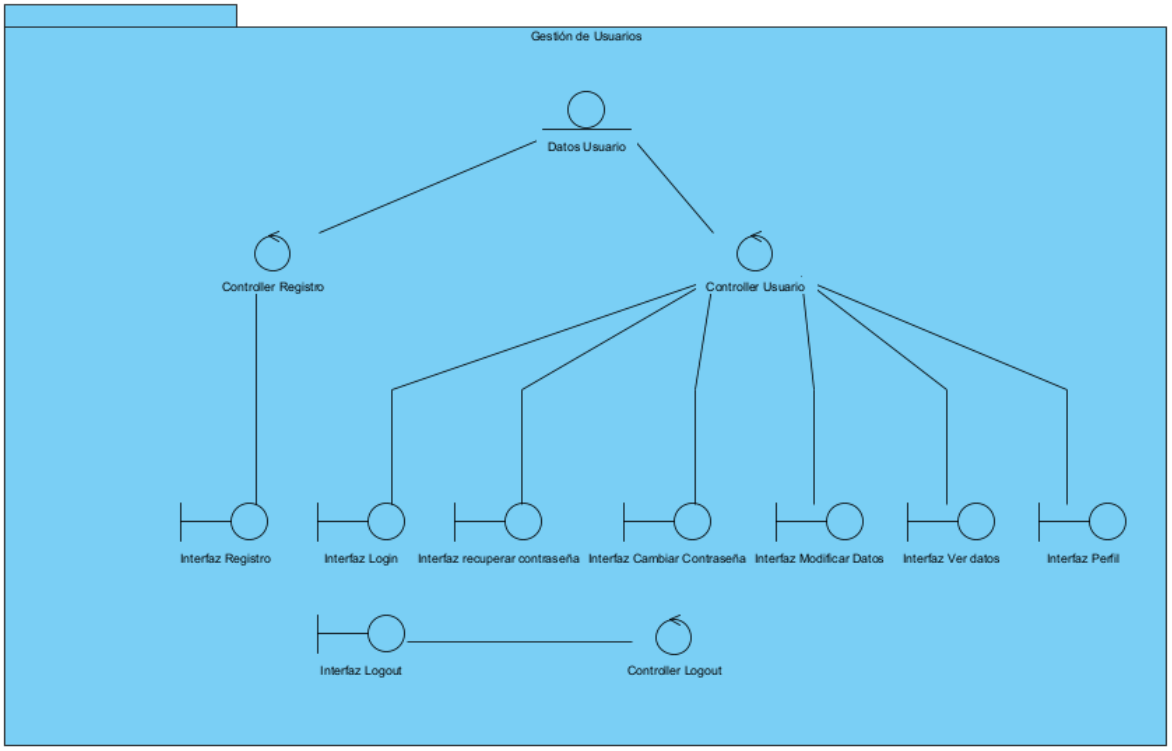


Figura 28: Diagrama de comunicación paquete Gestión de Usuarios

6.5.4 Vista arquitectónica del modelo de análisis

Se mostrará la vista completa de la arquitectura inicial que se ha obtenido en esta fase de análisis, de acuerdo al patrón arquitectónico Modelo-Vista-Controlador (MVC).

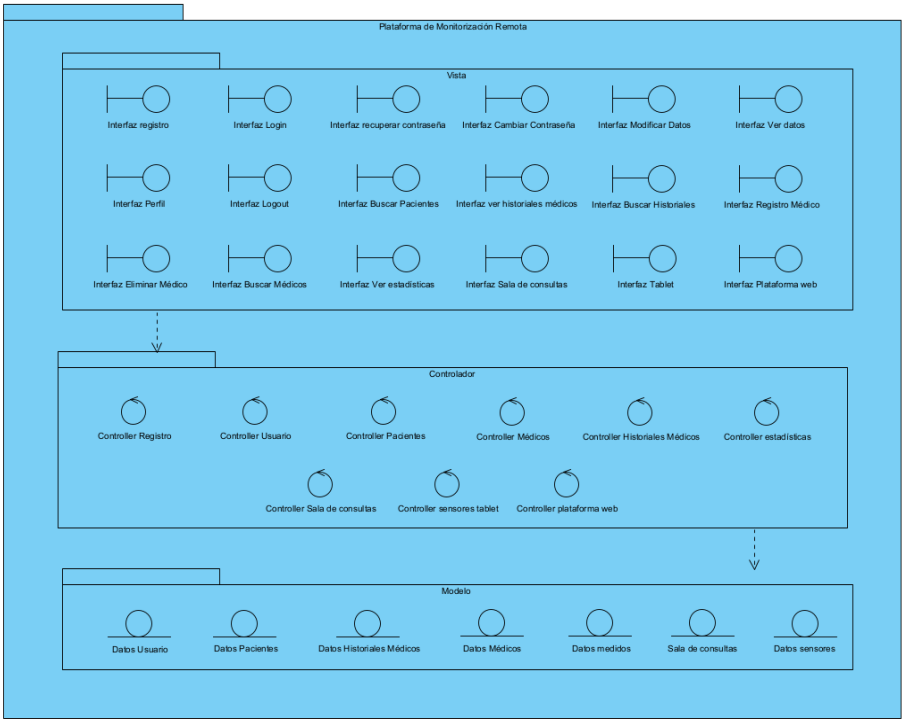


Figura 29: Vista arquitectónica

6.6 Diseño del sistema software

La fase de diseño se centrará en el dominio de la solución, aproximando más la aplicación a lo que sería una implementación real del sistema.

Para más información sobre este apartado consultar el Anexo IV.

6.6.1 Patrones arquitectónicos

- **Patrón API Gateway**

El patrón redirigirá las peticiones a la ruta que corresponda, tal que proporciona un único punto final para las aplicaciones de los clientes que necesiten comunicarse con los microservicios.

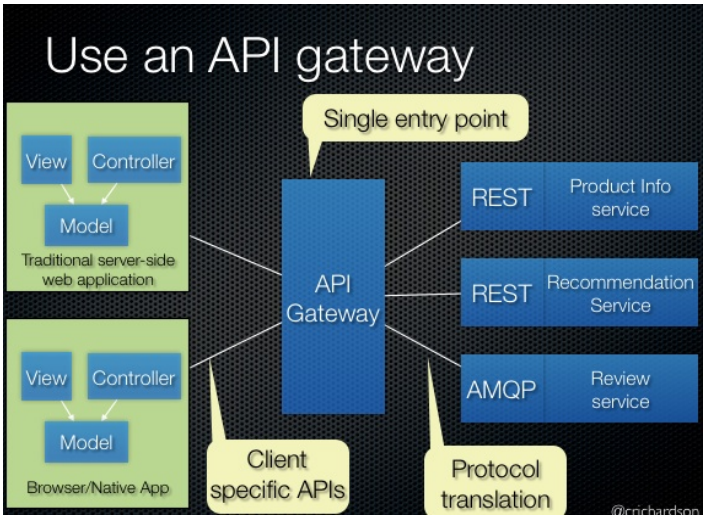


Figura 30: Patrón API Gateway

- **MVC**

El patrón MVC (Modelo-Vista-Controlador) es un patrón de arquitectura software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control. Esta arquitectura propone la separación de los componentes en tres grupos.

- **Modelo**: será la representación de los datos del sistema y la lógica de negocio.
- **Vista**: será la representación visual de los datos contenidos en el modelo al usuario.
- **Controlador**: Es el encargado de comunicar la vista con el modelo, gestionando el flujo de información.

- **Middleware**

El patrón Middleware (o Mediator) es el patrón que será utilizado para manejar la comunicación entre los componentes, especialmente las API del sistema. Hace posible que los componentes interactúen entre sí a través de un punto central: el mediador. En lugar de hablar directamente entre sí, el mediador recibe las peticiones y las envía

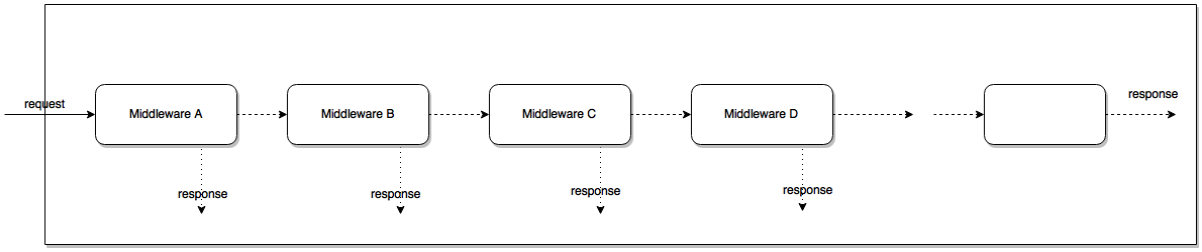


Figura 31: Patrón Middleware

- **Inheritance**

El patrón de diseño Inheritance (herencia) permitirá reducir el número de endpoints del sistema, estableciendo los métodos de las rutas de los endpoint bajo un subtipo, de manera que queden ordenados bajo una jerarquía.

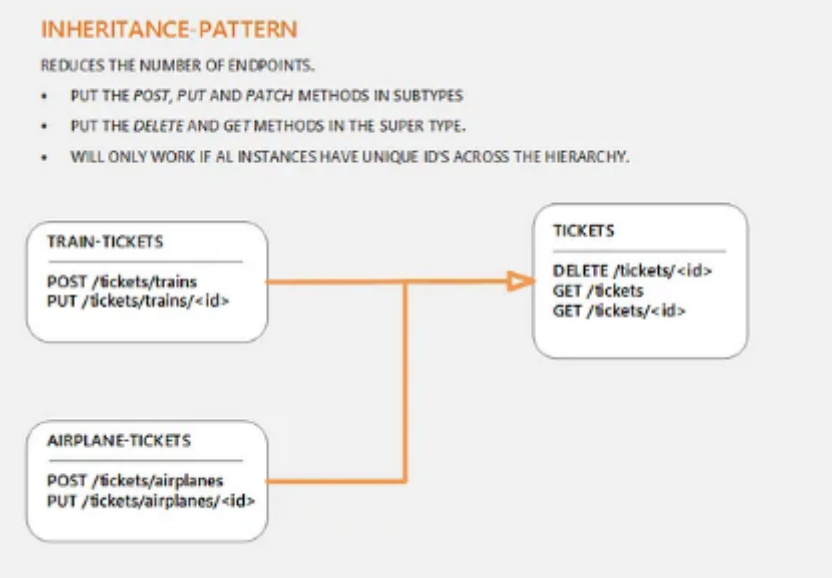


Figura 32: Patrón Inheritance

6.6.2 Subsistemas de diseño

Una vez definidos los patrones arquitectónicos del sistema, se realizará una división del proyecto en subpaquetes más pequeños y más específicos dentro de la estructura general de la aplicación.

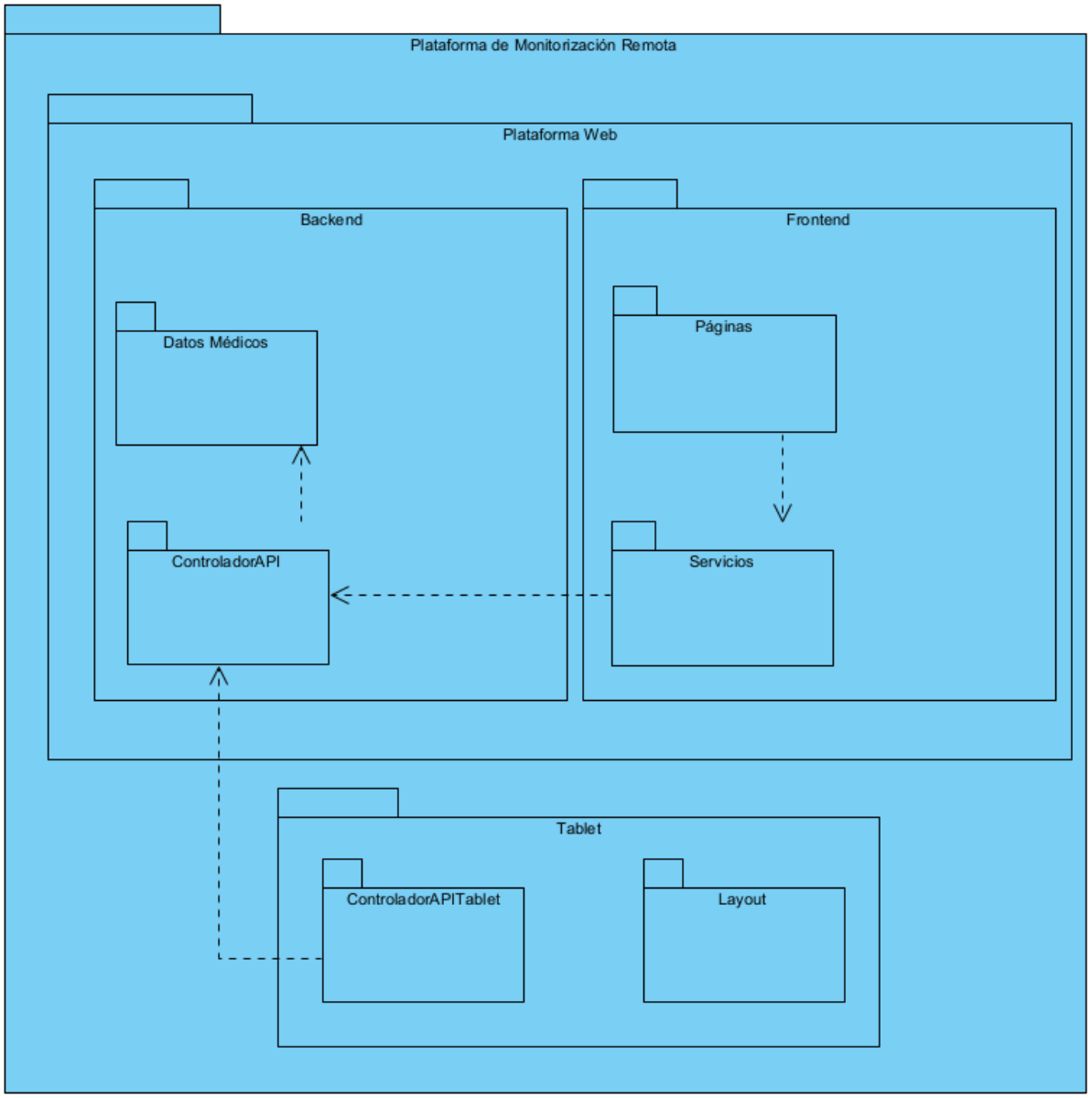


Figura 33: Subsistema de diseño

6.6.3 Clases de diseño

Este apartado especificará los contenidos del sistema, las clases de cada paquete, y los métodos de cada clase. A modo de ejemplo se expondrán el subpaquete **Controlador API** del servidor **Backend** y el subpaquete **Servicios** del servidor **Frontend**.

Para ver el resto de paquetes de manera más detallada, consultar el Anexo IV.

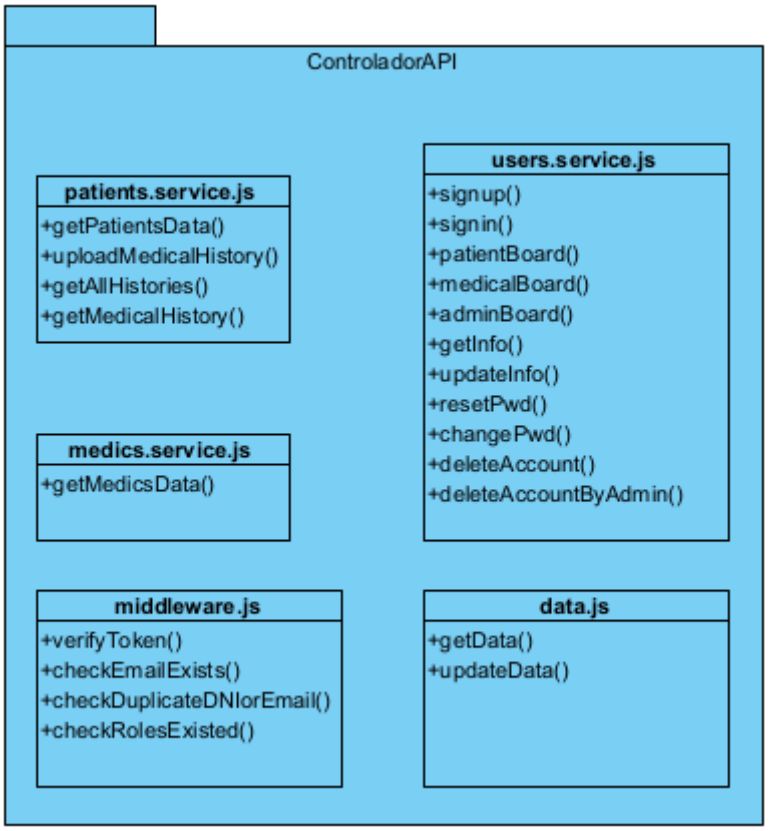


Figura 34: Subpaquete Controlador API

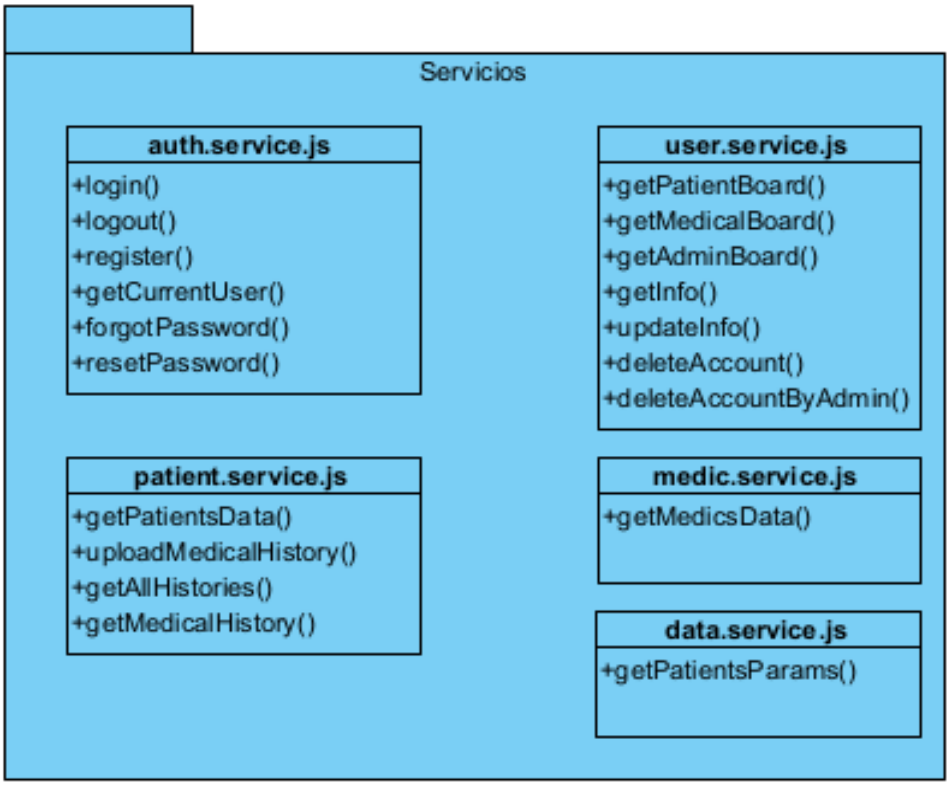


Figura 35: Subpaquete Servicios

6.6.4 Vista arquitectónica

Se mostrará la vista completa de la arquitectura inicial que se ha obtenido, de acuerdo al patrón arquitectónico Modelo-Vista-Controlador (MVC).

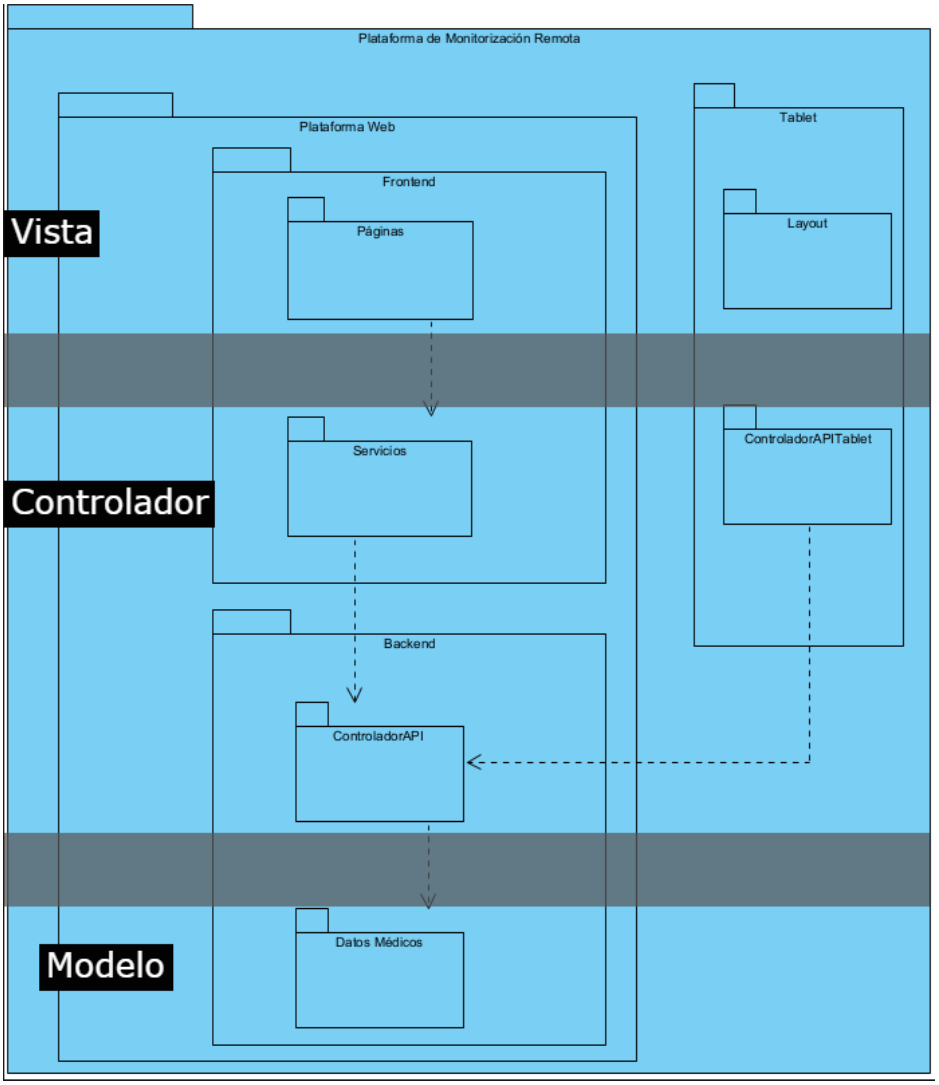


Figura 36: Vista arquitectónica del sistema en subpaquetes

6.6.5 Realización de casos de uso de diseño

Al igual que los diagramas de secuencia de los casos de uso especificados anteriormente. A continuación se mostrarán los diagramas de secuencia de diseño, indicando las clases de diseño creadas, se sigue la misma división de paquetes anterior.

Se adjuntará a modo de ejemplo el mismo diagrama que en el caso anterior, para más información consultar el Anexo IV.

- UC-016 Añadir Médico

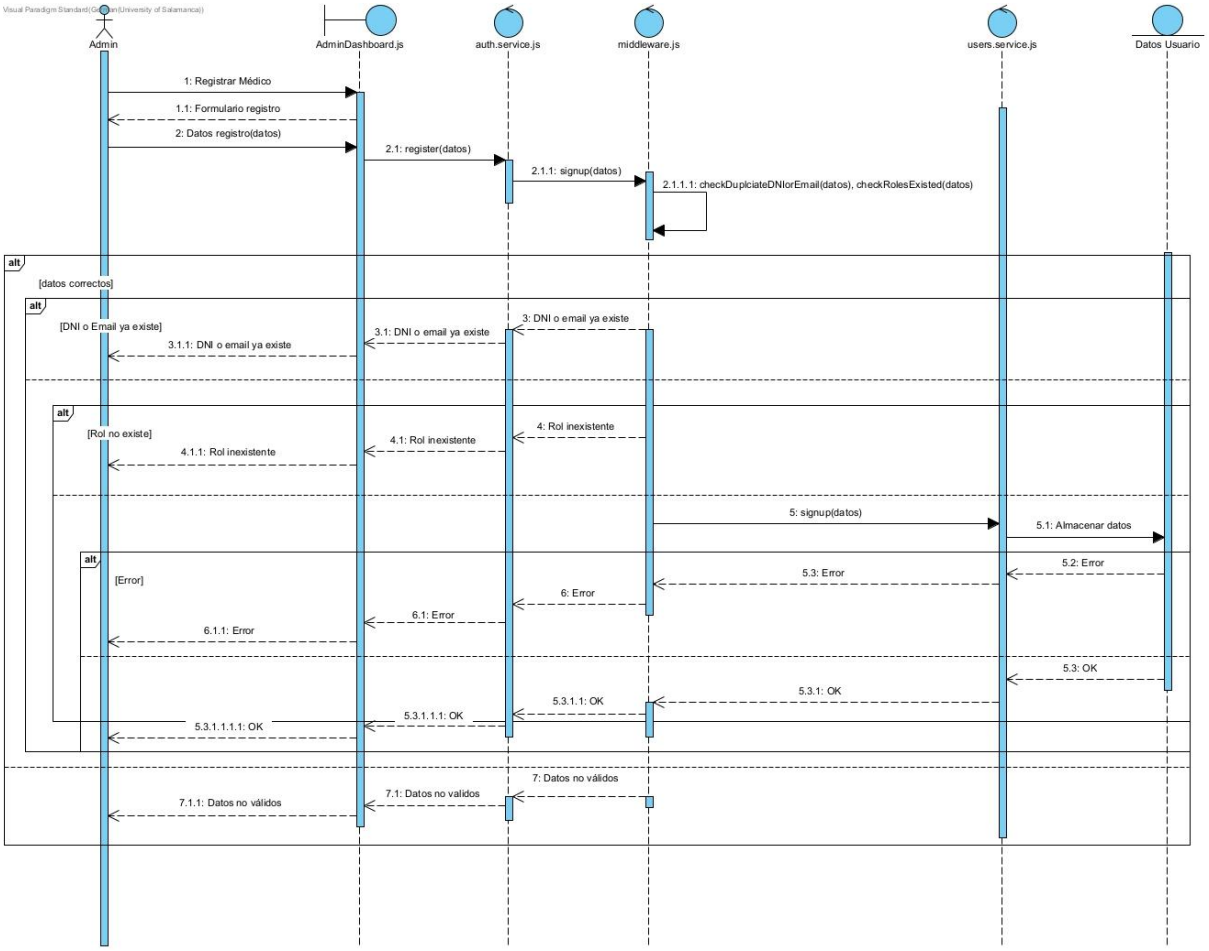


Figura 37: Diagrama de secuencia (diseño) Añadir Médico

6.6.6 Diseño de la base de datos

Para el correcto almacenamiento y gestión de los datos e información necesaria, se usará una base de datos MySQL con el siguiente diagrama Entidad-Relación.

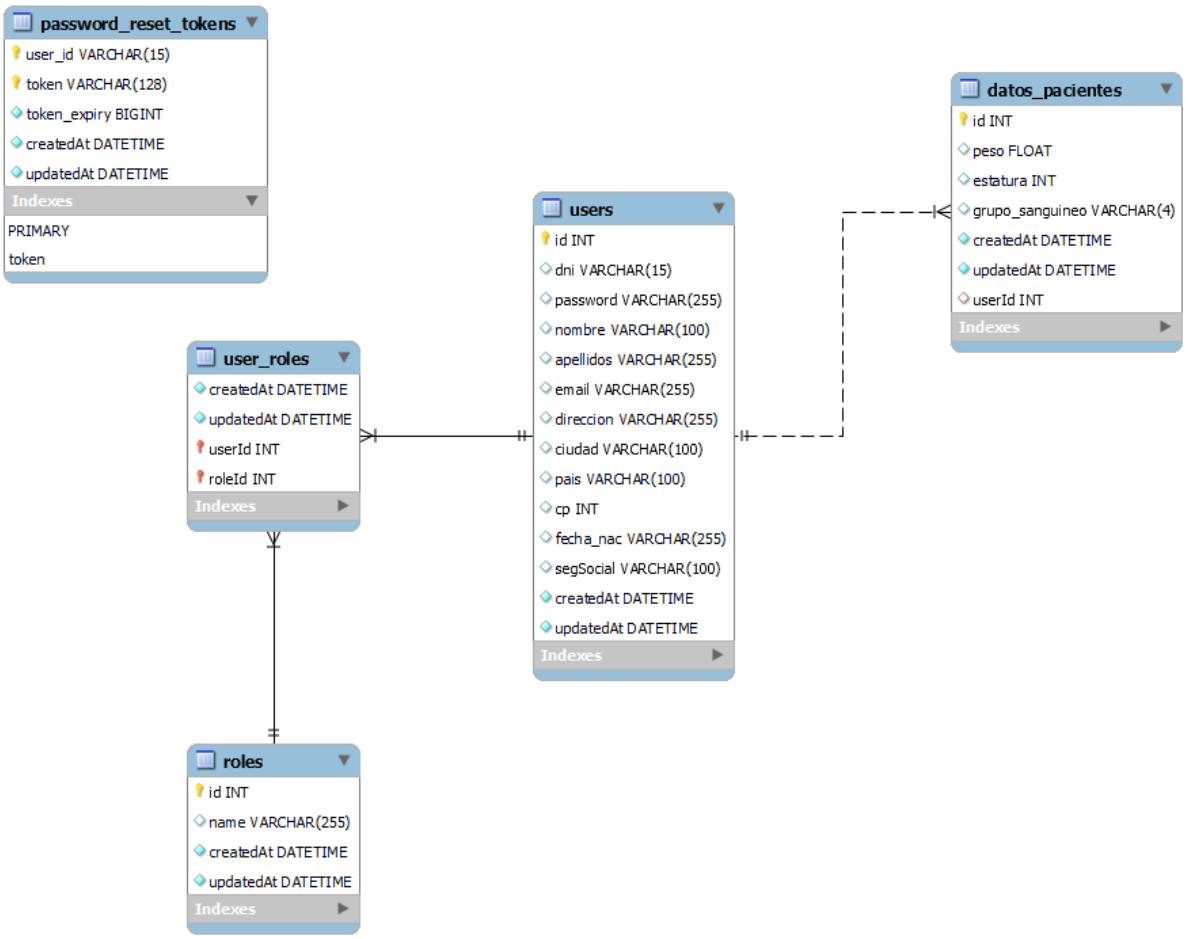


Figura 38: Diagrama Entidad-Relación de la base de datos

6.6.7 Modelo de despliegue

Se representará un modelo de despliegue utilizando un diagrama que muestre la arquitectura de ejecución del sistema incluyendo los sistemas hardware, y software que intervienen en el mismo.

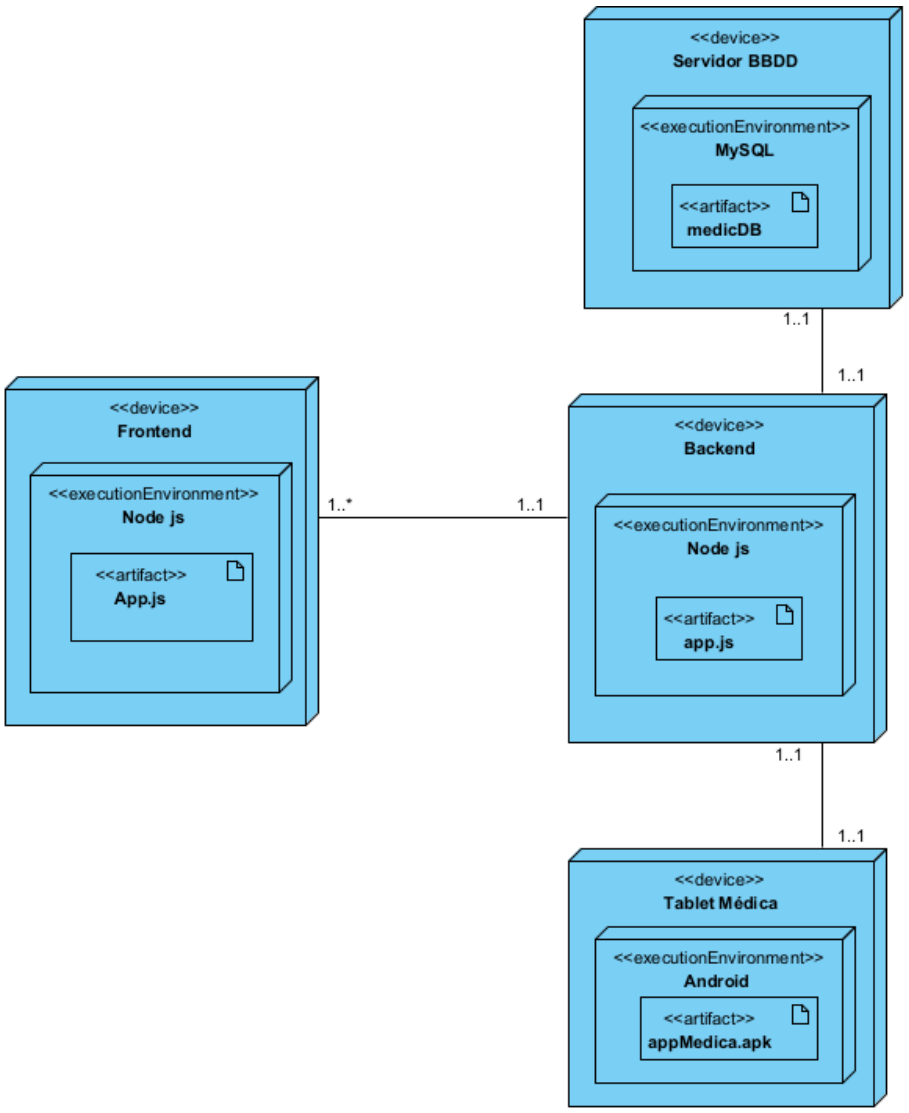


Figura 39: Diagrama de despliegue

6.7 Implementación

Durante la fase de implementación es cuando se ha realizado toda la codificación del sistema, partiendo de la base obtenida en la fase de diseño anterior, utilizando para ello las técnicas y herramientas mencionadas en el apartado 5 de este documento.

La implementación puede dividirse en 3 partes:

- **Backend**: Programación de la API REST a la que se comunican el resto de componentes (**Frontend** y **Tablet**) para llevar a cabo las funcionalidades del sistema. Es el servidor principal de la aplicación.
- **Frontend**: Programación del servidor web Frontend que hará de plataforma web a la que acceden los usuarios para realizar las diferentes acciones del sistema.
- **Tablet**: Programación de la aplicación de la Tablet médica que usarán los pacientes para enviar los datos medidos al servidor.

6.8 Pruebas

A lo largo de todo el proceso de desarrollo, tanto de los servidores como de la aplicación android, se han llevado a cabo numerosas pruebas con el fin de verificar el funcionamiento esperado del sistema en términos de programación.

Esta etapa ha sido crucial para detectar bugs y escenarios no contemplados en la aplicación, aportando nueva información para solucionar estos errores y acabar con un sistema sólido.

Finalmente una vez terminada la implementación del proyecto por separado, se puso en prueba todo el sistema global de manera constante para comprobar el correcto funcionamiento de todos los componentes.

6.9 Funcionalidad del sistema

En este apartado se mostrarán los aspectos más relevantes de la funcionalidad del sistema, en base a los paquetes de la aplicación que se han ido detallando a lo largo del proyecto.

Para obtener más información sobre las acciones que podrá realizar cada usuario, consultar el Anexo VI.

6.9.1 Gestión de usuarios

Al entrar a la aplicación, se mostrará una pantalla de inicio de sesión donde los usuarios podrán acceder con su cuenta o registrarse si no disponen de una.

Se verá una tarjeta central dividida en dos partes, a la izquierda tendrá el formulario de inicio de sesión, y a la derecha el botón de acceso a registro.

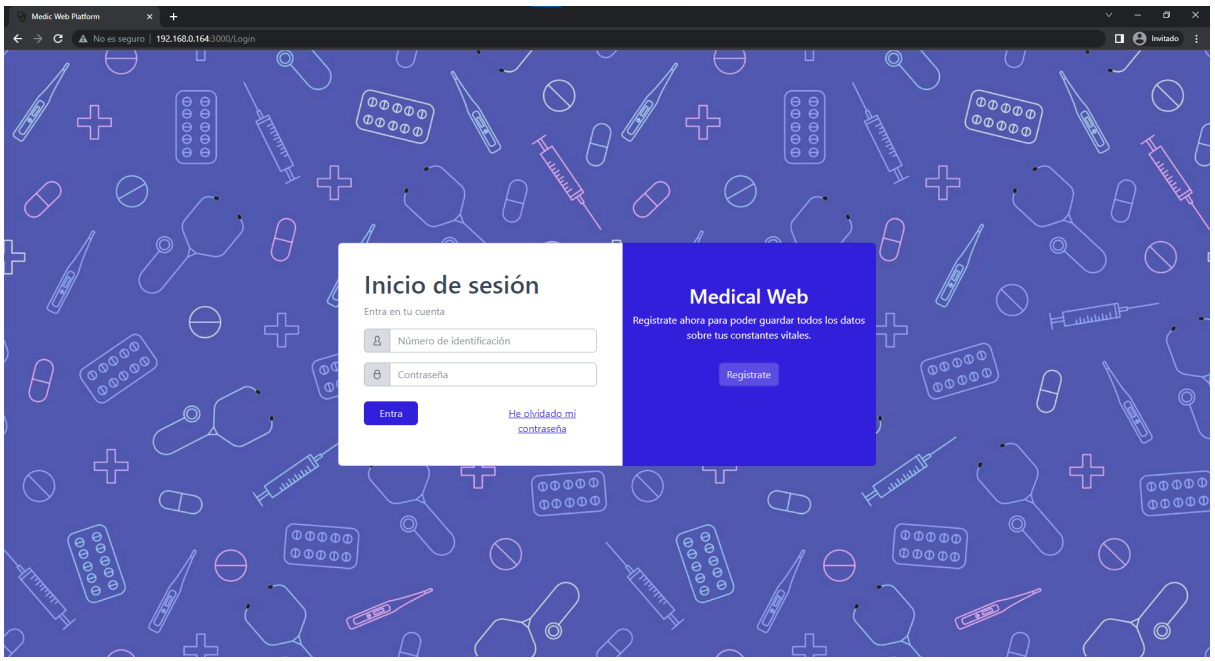


Figura 40: Inicio de sesión

Si en la pantalla anterior se aprieta el botón "**Registrate**", será redirigido a esta nueva pantalla.

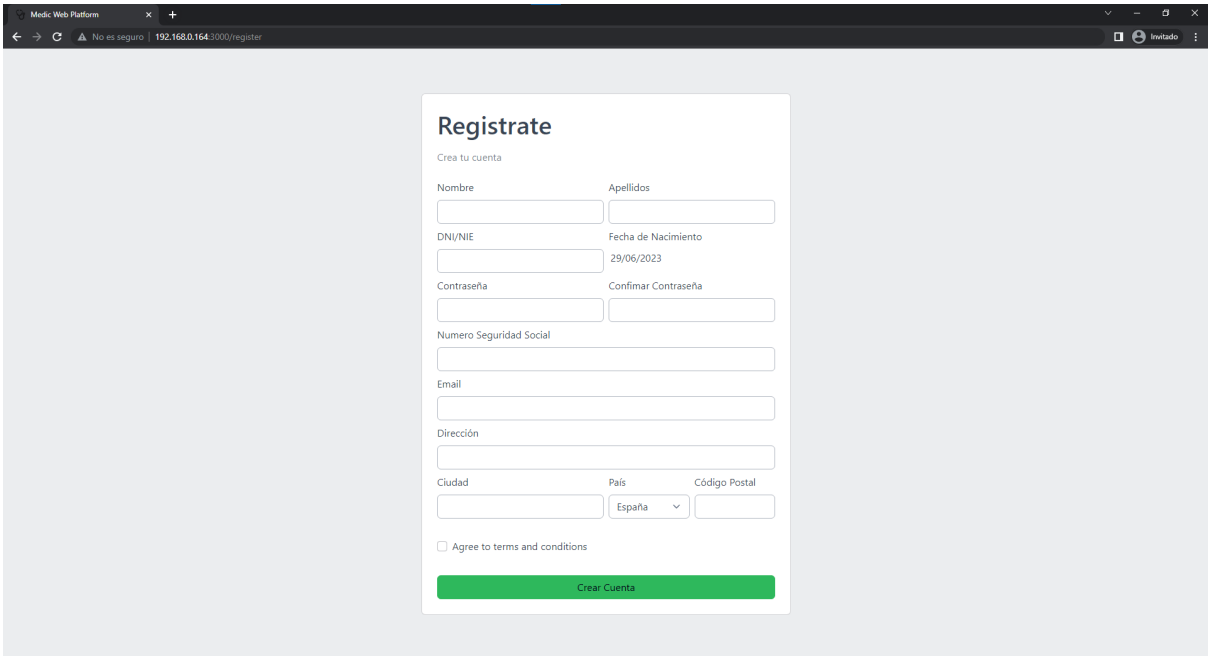


Figura 41: Registro

Si en la pantalla de inicio de sesión, el usuario ha clicado en "**He olvidado mi contraseña**", se mostrará un formulario con un campo donde el usuario deberá introducir el email asociado a la cuenta que desea recuperar.

Figura 42: Formulario de recuperación

Si el email es correcto y existe, se enviará un enlace de recuperación al mismo. El usuario tendrá que ir a su bandeja de entrada y acceder al enlace antes de que expire.

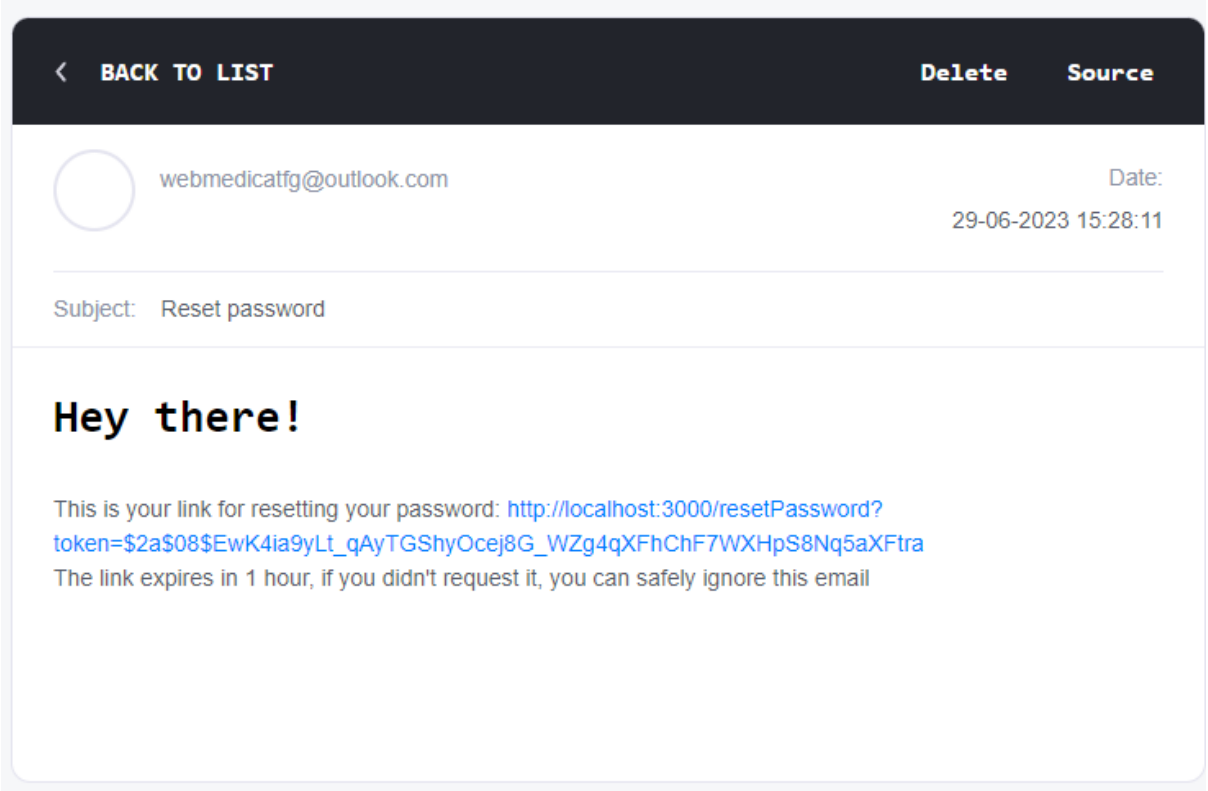


Figura 43: Email de recuperación

Al acceder al enlace, se le pedirá la nueva contraseña al usuario.

Figura 44: Nueva contraseña

Una vez el usuario haya iniciado sesión, se le mostrará una pantalla u otra en base a su rol.

Si es **paciente**:

Figura 45: Pantalla paciente normal.

En el caso de los pacientes, podrán acceder a la **Sala de consultas** y a los **Ajustes** de su perfil.

Figura 46: Barra de navegación Paciente

Si es **médico**:

Figura 47: Dashboard Médico

Con su barra de navegación especial asociada:

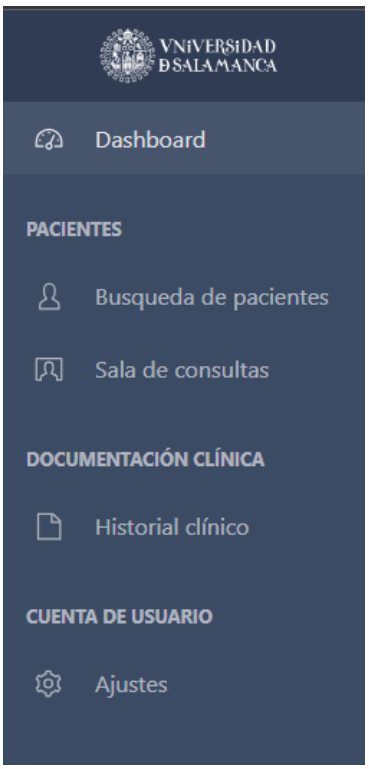


Figura 48: Barra de navegación Médico

O si por último, es **admin**:

Si accede a la plataforma un usuario con rol de Admin, verá dos botones principales, donde podrá dar de alta a nuevos usuarios médicos, o dar de baja a usuarios médicos ya existentes.

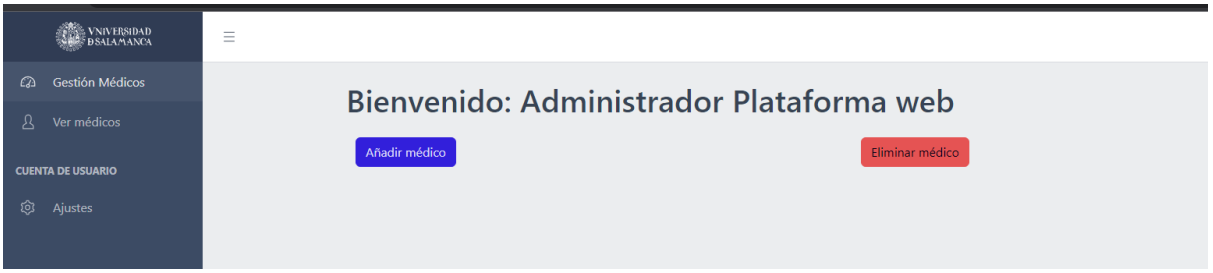


Figura 48: Dashboard Admin

6.9.2 Gestión de Pacientes

Los médicos podrán acceder a gestionar pacientes, buscando sus datos personales, historiales médicos y accediendo a la sala de consultas.

También podrán ver los últimos datos medidos, como se ve en la Figura 47.

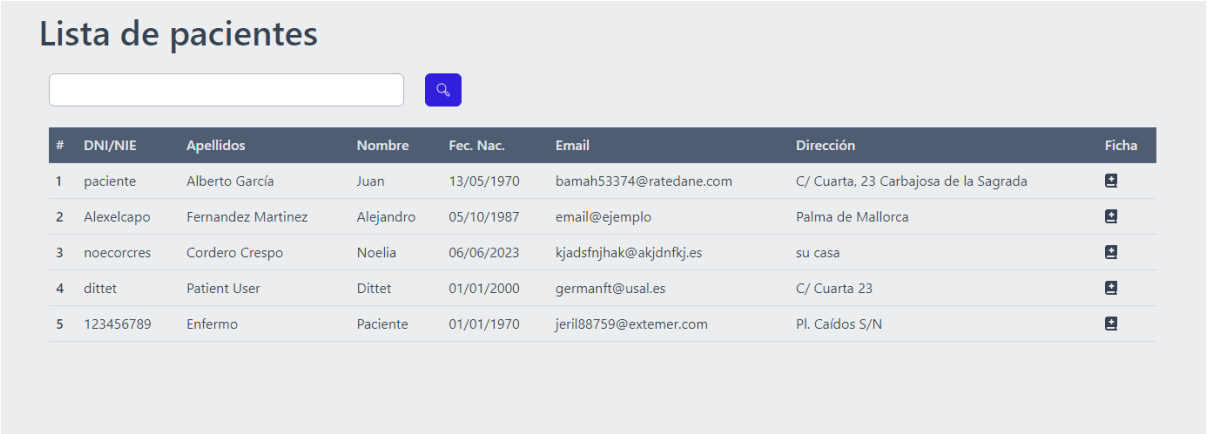


Figura 49: Lista de pacientes

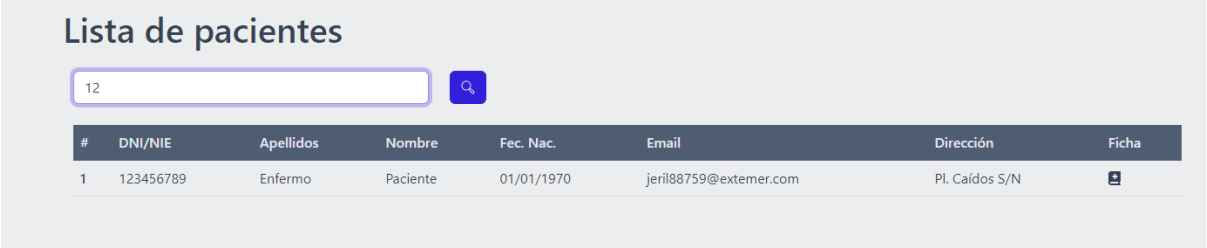


Figura 50: Lista de pacientes filtrada

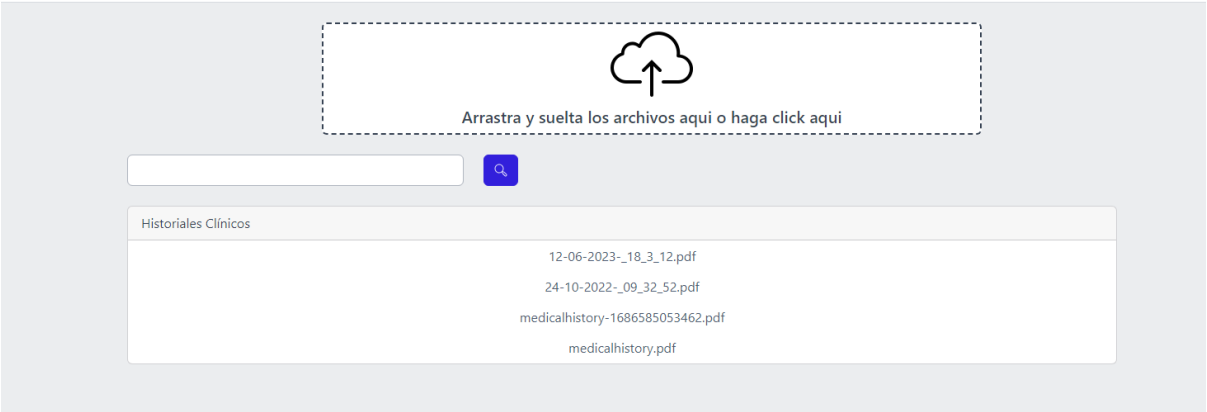
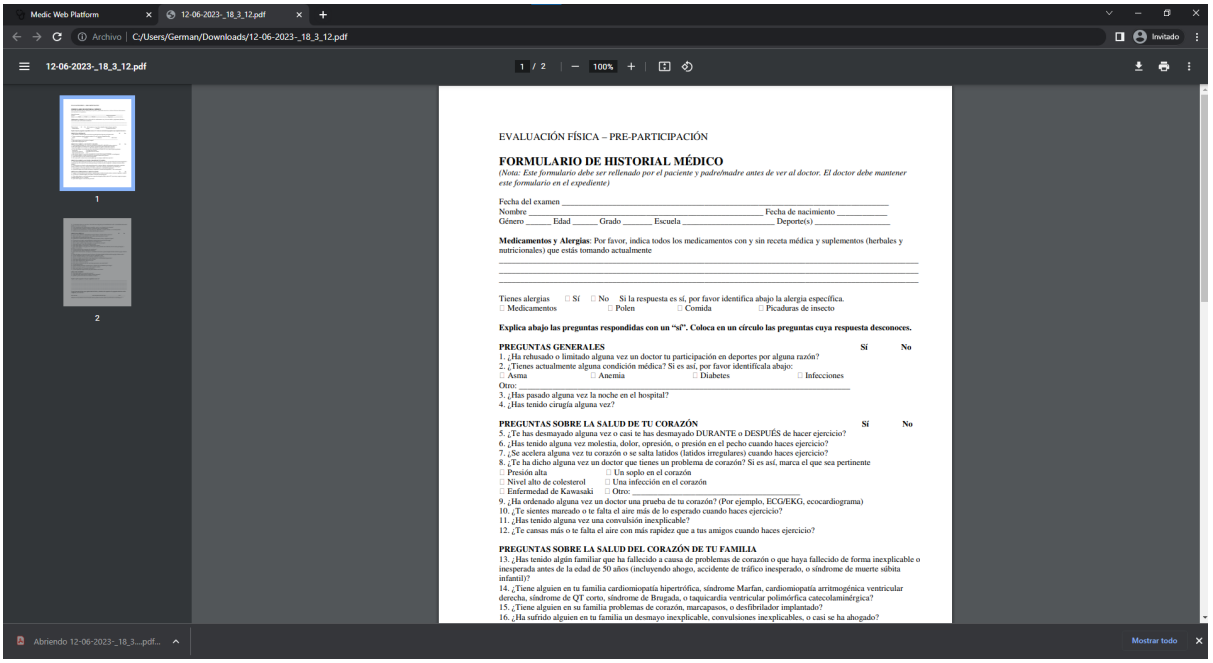


Figura 51: Historiales médicos



EVALUACIÓN FÍSICA – PRE-PARTICIPACIÓN

FORMULARIO DE HISTORIAL MÉDICO

(Nota: Este formulario debe ser rellenado por el paciente y padre/madre antes de ver al doctor. El doctor debe mantener este formulario en el expediente)

Fecha del examen _____

Nombre _____ Fecha de nacimiento _____

Género _____ Edad _____ Grado _____ Escuela _____ Deporte(s) _____

Medicamentos y Alergias: Por favor, indica todos los medicamentos con y sin receta médica y suplementos (herbales y nutricionales) que estás tomando actualmente

_____

_____

_____

Tienes alergias ☐ Sí ☐ No Si la respuesta es sí, por favor identifica abajo la alergia específica.

☐ Medicamentos ☐ Polen ☐ Comida ☐ Picaduras de insecto

Explica abajo las preguntas respondidas con un "sí". Coloca en un círculo las preguntas cuya respuesta desconoces.

PREGUNTAS GENERALES	**Sí**	**No**
1. ¿Ha rehusado o limitado alguna vez un doctor tu participación en deportes por alguna razón?		
2. ¿Tienes actualmente alguna condición médica? Si es así, por favor identifícala abajo:		
☐ Asma ☐ Anemia ☐ Diabetes ☐ Infecciones		
Otro: _____		
3. ¿Has pasado alguna vez la noche en el hospital?		
4. ¿Has tenido cirugía alguna vez?		

PREGUNTAS SOBRE LA SALUD DE TU CORAZÓN	**Sí**	**No**
5. ¿Te has desmayado alguna vez o casi te has desmayado DURANTE o DESPUÉS de hacer ejercicio?		
6. ¿Has tenido alguna vez molestia, dolor, opresión, o presión en el pecho cuando haces ejercicio?		
7. ¿Se acelera alguna vez tu corazón o se salta latidos (latidos irregulares) cuando haces ejercicio?		
8. ¿Te ha dicho alguna vez un doctor que tienes un problema de corazón? Si es así, marca el que sea pertinente		
☐ Presión alta ☐ Un soplo en el corazón		
☐ Nivel alto de colesterol ☐ Una infección en el corazón		
☐ Enfermedad de Kawasaki ☐ Otro: _____		
9. ¿Ha ordenado alguna vez un doctor una prueba de tu corazón? (Por ejemplo, ECG/EKG, ecocardiograma)		
10. ¿Te sientes mareado o te falta el aire más de lo esperado cuando haces ejercicio?		
11. ¿Has tenido alguna vez una convulsión inexplicable?		
12. ¿Te cansas más o te falta el aire con más rapidez que a tus amigos cuando haces ejercicio?		

PREGUNTAS SOBRE LA SALUD DEL CORAZÓN DE TU FAMILIA	**Sí**	**No**
13. ¿Has tenido algún familiar que ha fallecido a causa de problemas de corazón o que haya fallecido de forma inexplicable o inesperada antes de la edad de 50 años (incluyendo ahogo, accidente de tráfico inesperado, o síndrome de muerte súbita infantil)?		
14. ¿Tiene alguien en tu familia cardiomiopatía hipertrófica, síndrome Marfan, cardiomiopatía arritmogénica ventricular derecha, síndrome de QT corto, síndrome de Brugada, o taquicardia ventricular polimórfica catecolaminérgica?		
15. ¿Tiene alguien en su familia problemas de corazón, marcapasos, o desfibrilador implantado?		
16. ¿Ha sufrido alguien en tu familia un desmayo inexplicable, convulsiones inexplicables, o casi se ha ahogado?		

Figura 52: Historial médico descargado

6.9.3 Gestión de Médicos

De manera similar que los médicos con los pacientes, los administradores podrán acceder a gestionar médicos, buscando sus datos personales, dando de alta nuevos médicos, y dando de baja a médicos ya existentes, tal y como se ve en la figura 48.

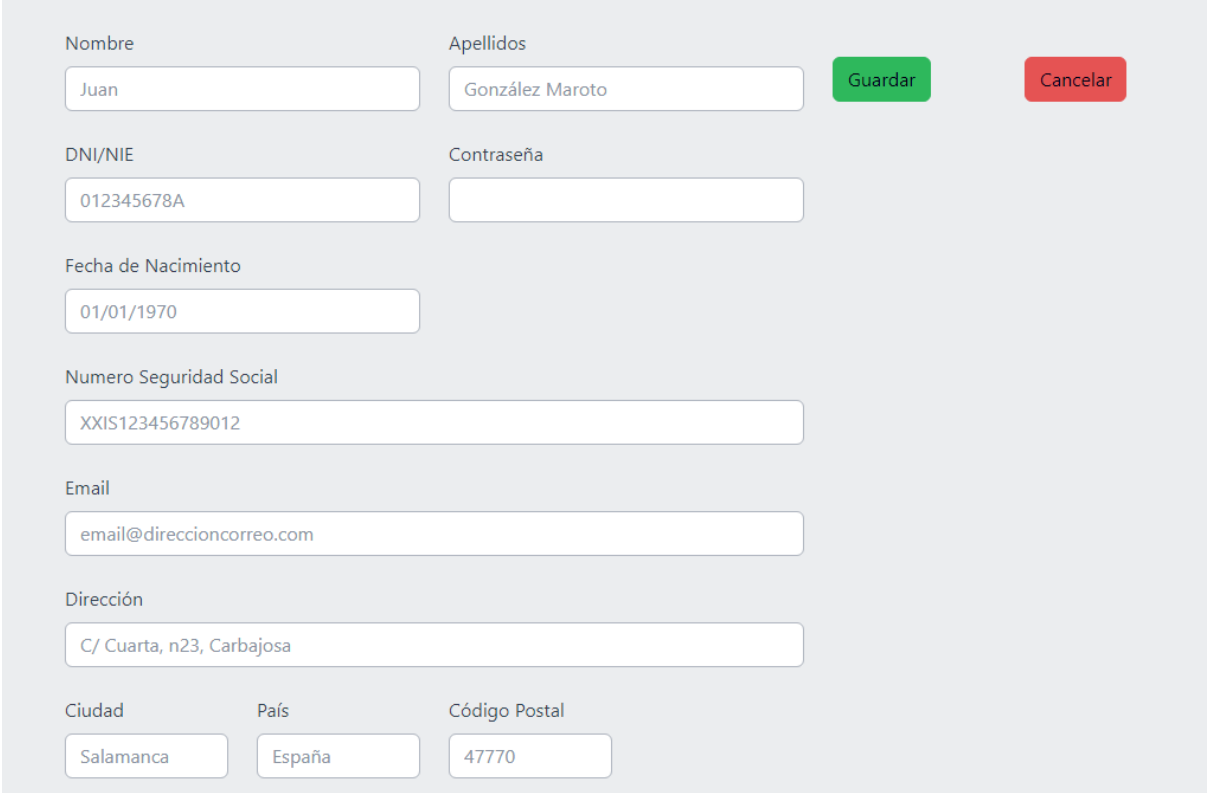


Figura 53: Alta nuevo médico

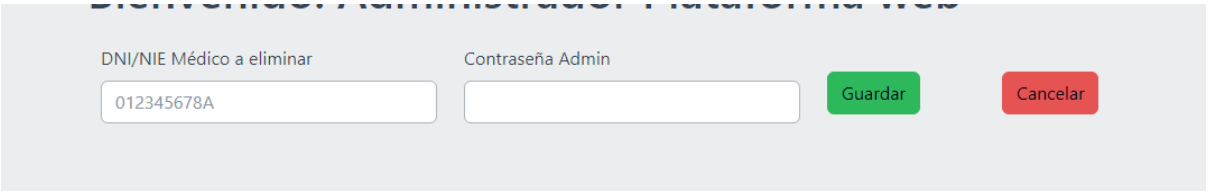


Figura 54: Eliminar Médico

6.9.4 Gestión de Plataforma Web

Los usuarios de la plataforma Web, podrán acceder a la sala de consultas mediante videoconferencia, y ver sus últimas mediciones, como se ve en las figuras 45 para pacientes, y 47 para médicos.

Tanto médicos como pacientes podrán acceder a la sala de consultas, que accederá a una sala de videollamada mediante Jitsi. El usuario podrá configurar sus dispositivos de entrada y salida, y el nombre con el que aparecerá en la llamada.

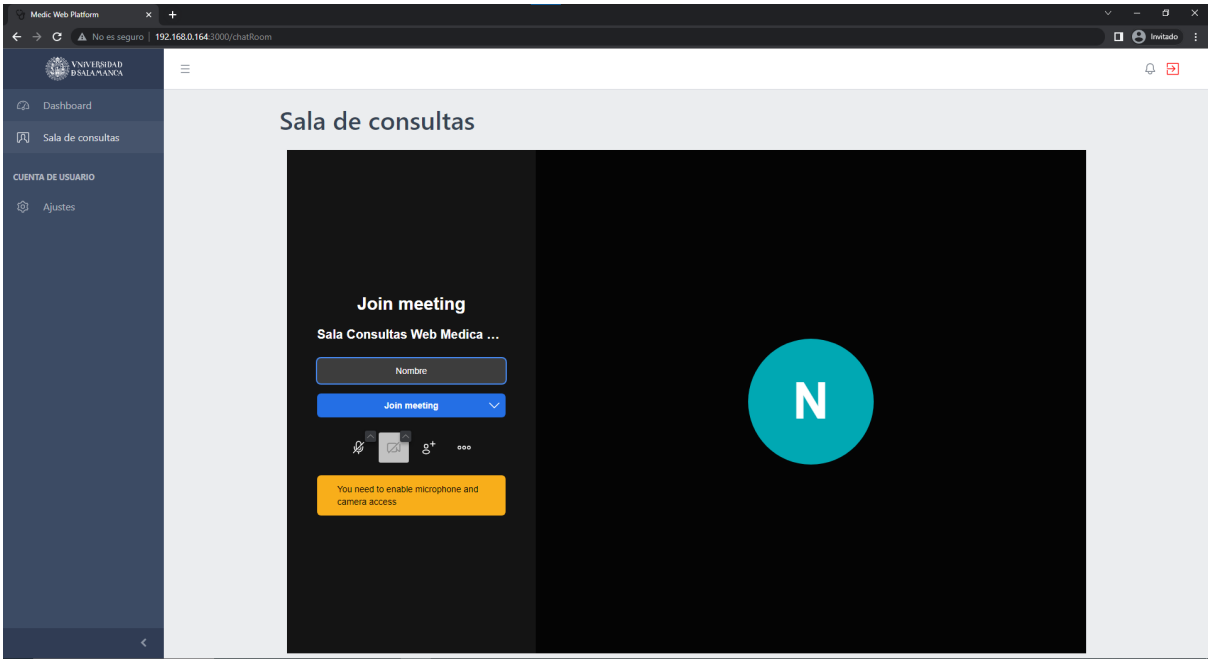


Figura 55: Sala de consultas

Además, todos los usuarios podrán acceder a sus datos personales y modificarlos, o incluso eliminar su cuenta, acción para la cual será necesario introducir la contraseña personal del usuario.

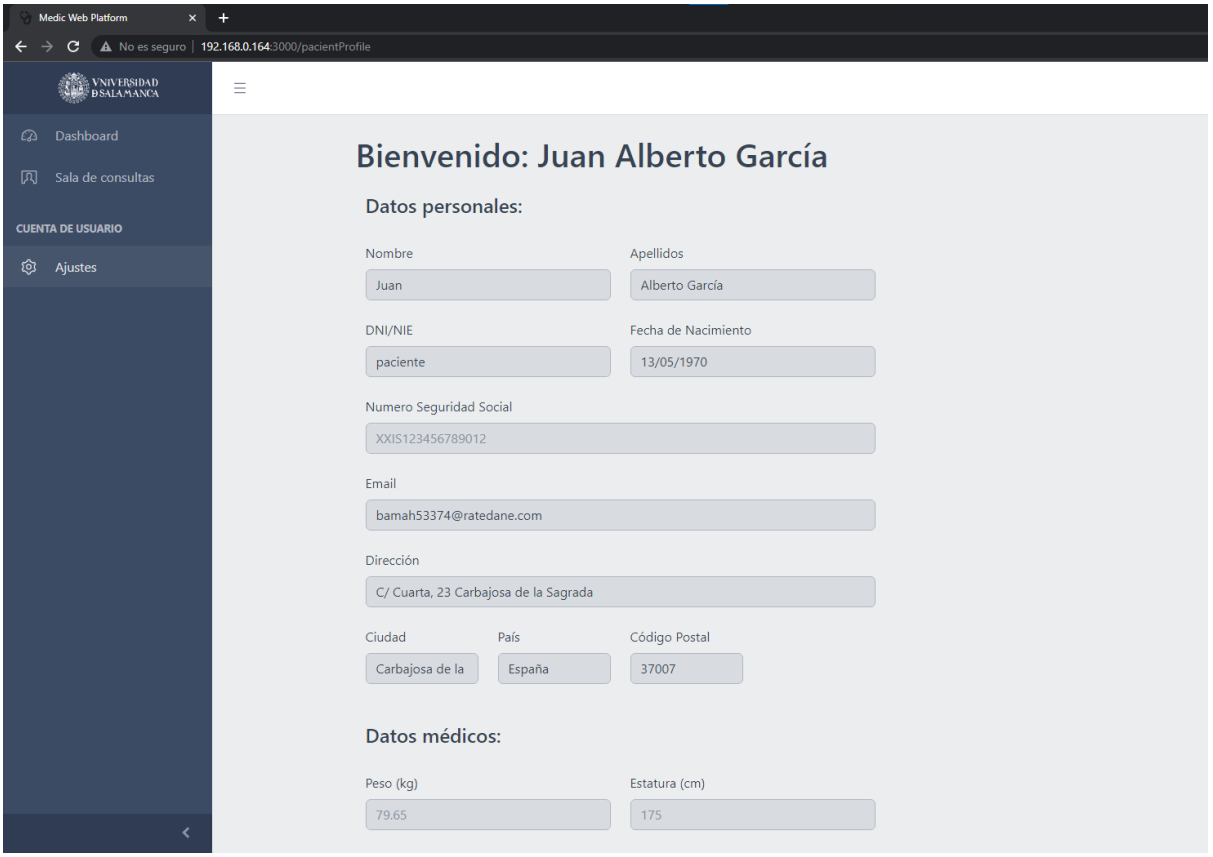


Figura 56: Datos personales

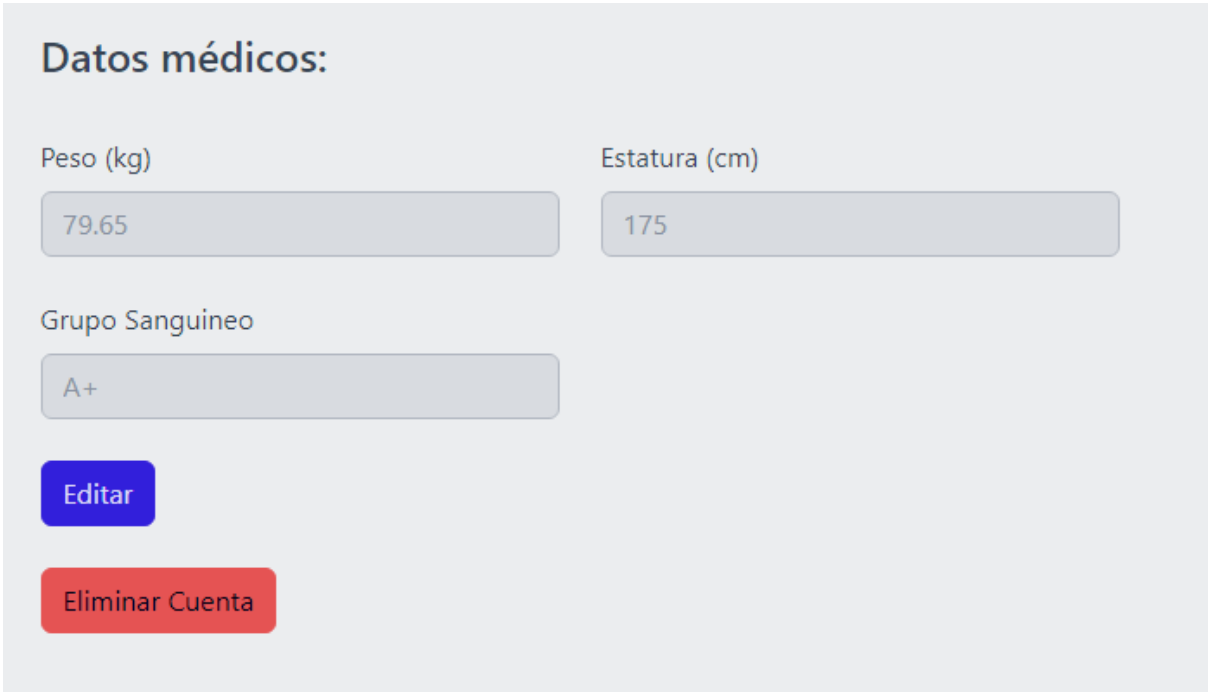


Figura 57: Datos personales 2

6.9.5 Gestión de Tablet

Nada más abrir la aplicación se mostrará la pantalla de inicio de sesión, donde se pedirán los credenciales de usuario y el botón de acceso.

En la parte inferior de la pantalla tendrán lugar logos autodescriptivos de los sensores más utilizados para ayudar a la comprensión a las personas mayores.

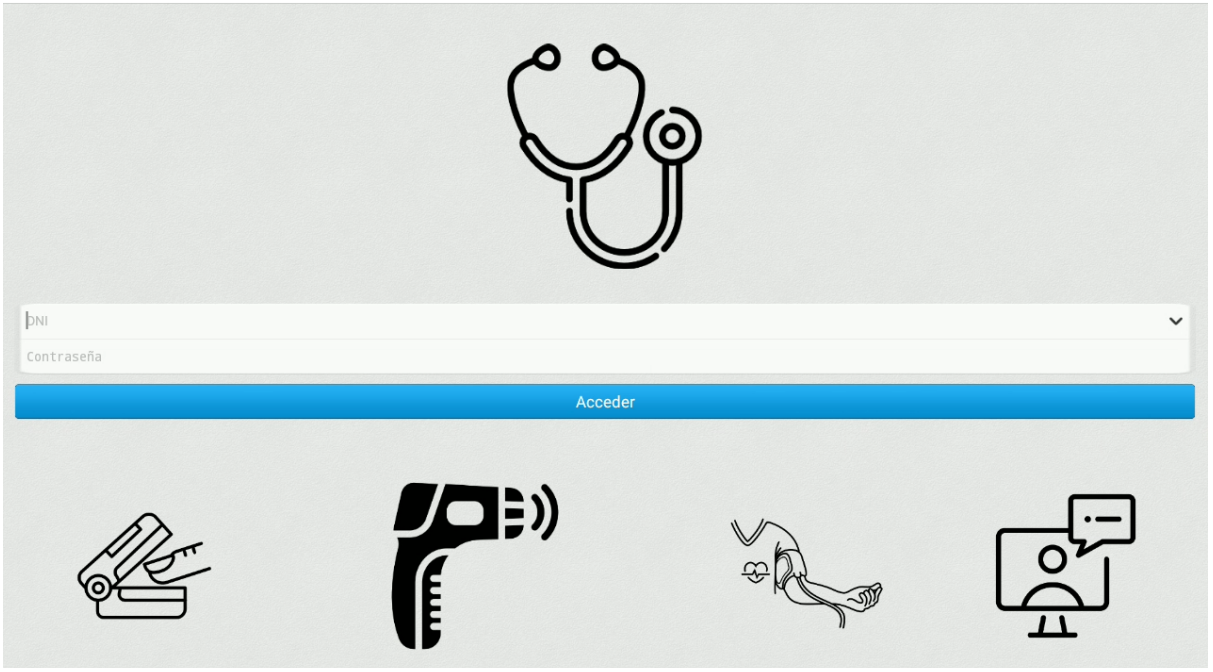


Figura 58: Login desde tablet

Al acceder se mostrará la pantalla principal del sistema. Serán bloques divididos por colores según el sensor utilizado. Inicialmente todos estarán a un valor neutro.

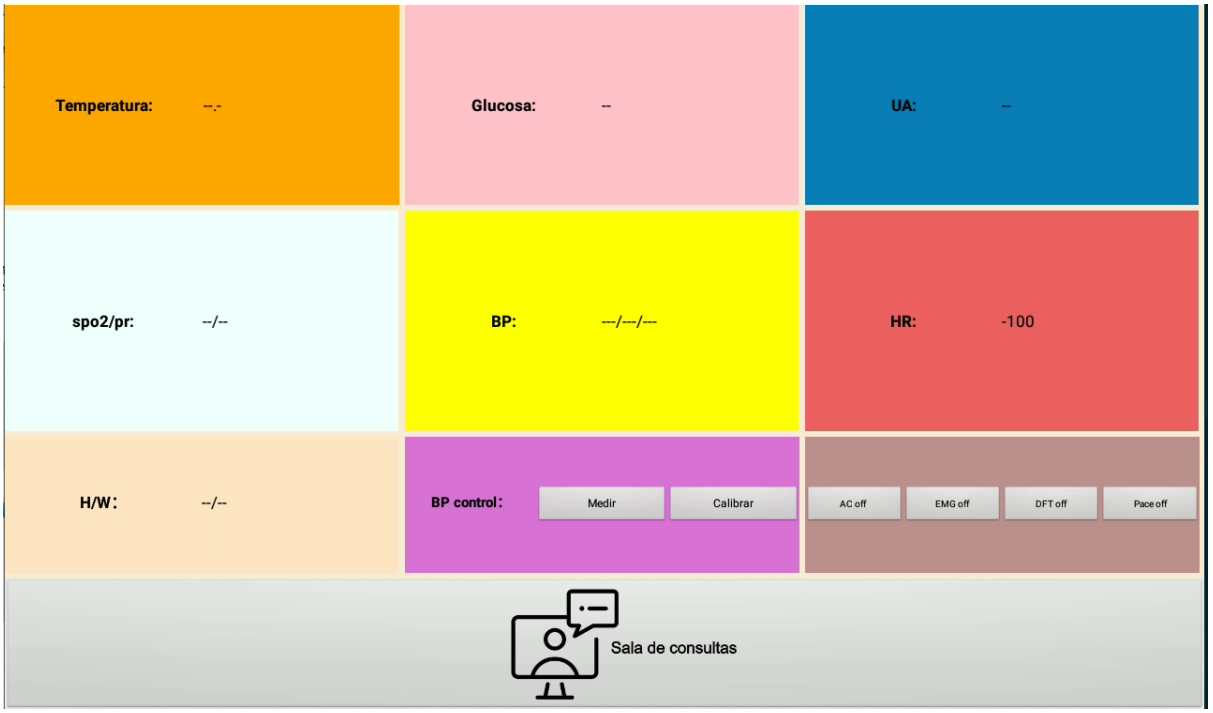


Figura 59: Pantalla principal tablet

También se podrá acceder a la sala de consultas desde el botón inferior de la pantalla.

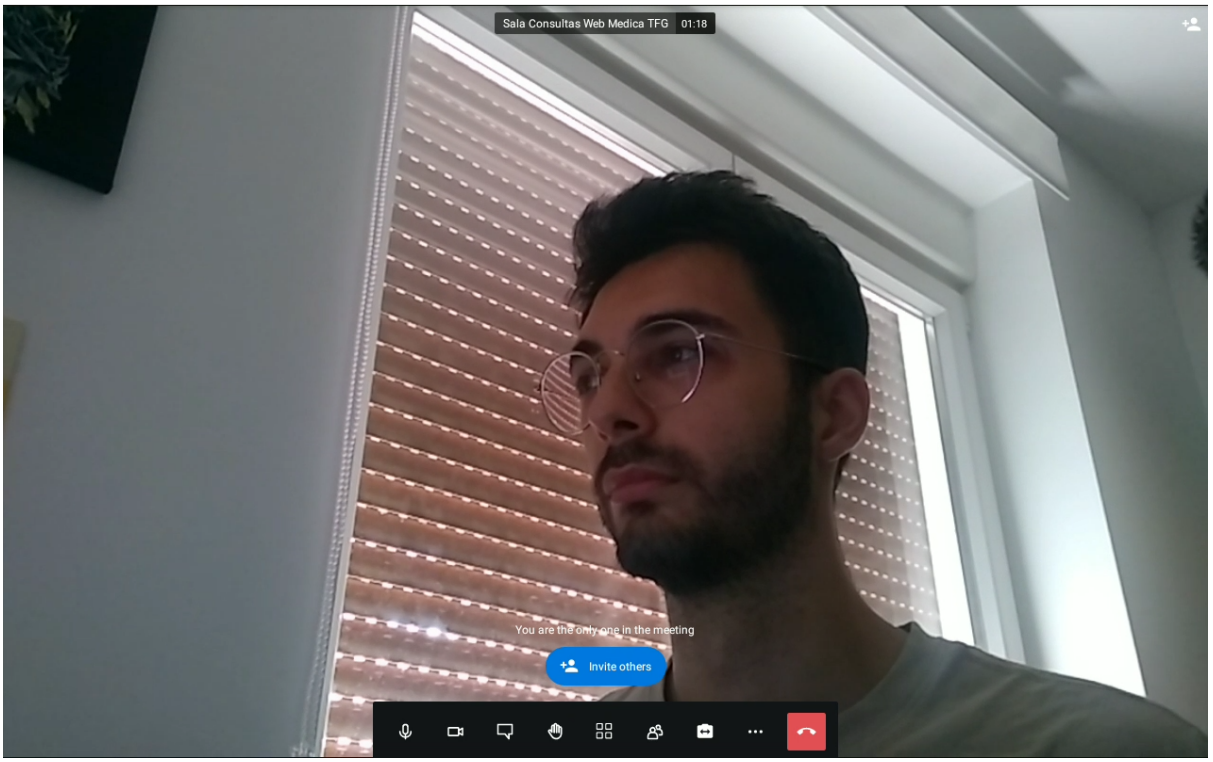


Figura 31: Sala de consultas desde la tablet

Para más detalles de las acciones que puede hacer cada usuario consultar el Anexo VI: Manual de usuario. Donde se explica todo con más información.

7. Conclusiones y líneas de trabajo futuras

En este apartado se recogen las conclusiones una vez finalizado el desarrollo del proyecto “Plataforma de monitorización remota para la atención domiciliaria”. Además se incluirá un apartado de líneas de trabajo futuras para mejorar el proyecto

Con el proyecto terminado, se puede concluir que se han ido cumpliendo todos los objetivos planteados al inicio del desarrollo, tanto los de sistema como los personales.

En lo personal, se ha aprendido mucho más a profundidad cómo funciona el desarrollo web por detrás y todo los procesos que se llevan a cabo para hacer que funcionen de manera que el usuario final no se de cuenta de todo lo que acarrea. El hecho de aprender nuevas tecnologías, actualizadas y que se usan en el mundo real ha sido un objetivo muy satisfactorio de cumplir, junto al objetivo de poner en marcha un prototipo con una nueva manera de ver las consultas sanitarias y que ofrece algo que en un futuro pueda ser útil para mucha gente. Al ser una primera versión, tiene limitaciones y mejoras que se pueden aplicar en líneas futuras, algunas se citarán más adelante.

Respecto a los objetivos del sistema se han logrado todos, la gestión de usuarios, como por ejemplo teniendo en cuenta los parámetros de seguridad de manera que un usuario con un rol específico no pueda acceder a páginas restringidas a otros roles, o el almacenaje de las contraseñas hasheadas, y todo el flujo por el que debe pasar un usuario y el programa para recuperar una contraseña de manera segura, sin que el sistema ofrezca más información de la necesaria para posibles ataques. También se tiene en estima el haber creado una base de datos más cercana al mundo real de lo visto en la carrera y las relaciones entre las tablas.

Se ha conseguido la medición de datos en tiempo real haciendo uso de sensores y el protocolo HTTP, y se ha aprendido, aunque de manera introductoria, desarrollo de aplicaciones Android.

Añadir que se ha investigado bastante sobre el protocolo WebRTC (Web Real Time Communications), un estándar que permite la comunicación de redes entre pares en tiempo real y el intercambio de datos multimedia en navegadores, pero que por cuestiones de eficiencia tanto temporal como de facilidad de uso para un usuario final, se ha acabado optando por el SDK de Jitsi. Logrando así acortar el tiempo de desarrollo al no tener que programar de cero conexiones de vídeo en tiempo real.

En lo relevante a mejoras futuras se podría implementar alguna de las siguientes propuestas, o cualquier otra que no se haya contemplado actualmente:

- **Añadir soporte a nuevos dispositivos de medición**: Ya sea con sensores conectados mediante puerto USB a la tablet, o haciendo uso de dispositivos IoT que estén integrados a la aplicación de la misma.

- **Sistema de envío de información sin conexión**: El sistema busca añadir conectividad sanitaria a pueblos remotos, pero hay veces que éstos no tienen conexión a internet óptima, por lo que lograr un sistema de envío de

información sin conexión mediante radiofrecuencia como por ejemplo con tecnología LoRA, podría ser una buena mejora.

- **Añadir servidor de videoconferencia dedicado**: Actualmente se usa una sala de videoconferencia proporcionada por Jitsi, podría crearse un servidor dedicado que hiciera de anfitrión para estas consultas de manera más segura.

- **Mejoras en la implementación**: Pese a que la aplicación se ha probado extensivamente, siempre puede haber mejoras de eficiencia o bugs que no se han detectado durante el desarrollo.

- **Reducir costes**: La tablet que hace de monitor de pacientes, aunque tenga un precio mucho más reducido que otros competidores, sigue siendo un coste alto, por lo que conseguir encontrar alternativas o reducir costes de estas sería óptimo.

Podrían ampliarse estas líneas de mejora haciendo un estudio en las necesidades de los usuarios centrándonos en médicos del mundo rural, de manera que puedan ofrecer un feedback útil para el desarrollo futuro del sistema.

8.Referencias

- McConnell, S. - “Desarrollo y gestión de proyectos informáticos ”.
  - Mc Graw Hill, 1997.

- Documentación Microsoft Project
  - https://support.microsoft.com/es-es/project

- Marco teórico de Desarrollo Software del Proceso Unificado
  - http://desarrollo-software-epis.blogspot.com/2010/02/marco-teorico-de-desarrollo-de-software.html

- Moreno García, María N. TEMA 3 Métodos de estimación y gestión del riesgo GESTIÓN DE PROYECTOS (GII-USAL)

- Moreno García, María N. TEMA 4 Planificación temporal de proyectos GESTIÓN DE PROYECTOS (GII-USAL)

- Pressman, R. S. - “Ingeniería del Software: Un Enfoque Práctico”. 7ª Edición. McGraw-Hill. 2010

- Amador Durán Toro, Beatriz Bernárdez Jiménez – “Metodología para la Elicitación de Requisitos de Sistemas Software”: (Accedido en 16-05-2023)
  - http://www.lsi.us.es/docs/informes/lsi-2000-10.pdf

- Visual Paradigm Manual: (Accedido en 20-05-2023)
  - https://www.visual-paradigm.com/support/documents/vpuserguide.jsp

- Moreno García, María N. TEMA 2 UML. Unified Modeling Language INGENIERÍA DEL SOFTWARE II

- Visual Paradigm Manual:
  - https://www.visual-paradigm.com/support/documents/vpuserguide.jsp

- Referencias sobre Patrones

  - https://medium.com/@patricksavalle/rest-api-design-as-a-craft-not-an-art-a3fd97ed3ef4

  - https://dzone.com/articles/understanding-middleware-pattern-in-expressjs

  - https://www.patterns.dev/posts/mediator-pattern

  - https://medium.com/design-microservices-architecture-with-patterns/api-gateway-pattern-8ed0ddfce9df

- Artículos para respaldar la falta de médicos:

    - https://www.lagacetadesalamanca.es/salamanca/la-vanguardista-formula-que-ya-usa-el-hospital-para-atender-a-enfermos-cronicos-del-medio-rural-LC13113448

    - https://www.lagacetadesalamanca.es/salamanca/la-urgencias-del-hospital-de-salamanca-atascadas-por-la-falta-de-camas-para-ingresos-AC13115621

    - INE-MSCBS, "Uso de servicios sanitarios (consulta médica, hospitalización, mamografía, citología, consulta ginecológica)," 2020.

- Papers de estado del arte:

    - J. Wan, M. A. A. H. Al-awlaqi, M. Li, M. O'Grady, X. Gu, J. Wang, and N. Cao, "Wearable iot enabled real-time health monitoring system," EURASIP Journal on Wireless Communications and Networking, vol. 2018, p. 298, Dec 2018

    - M. M. Islam, A. Rahaman, and M. R. Islam, "Development of smart healthcare monitoring system in iot environment," SN Computer Science, vol. 1, p. 185, May 2020.

    - K. T. Kadhim, A. M. Alsahlany, S. M. Wadi, and H. T. Kadhum, "An overview of patient's health status monitoring system based on internet of things (iot)," Wireless Personal Communications, vol. 114, pp. 2235–2262, Oct 2020.

    - O. S. Albahri, A. S. Albahri, K. I. Mohammed, A. A. Zaidan, B. B. Zaidan, M. Hashim, and O. H. Salman, "Systematic review of real-time remote health monitoring system in triage and priority-based sensor technology: Taxonomy, open challenges, motivation and recommendations," Journal of Medical Systems, vol. 42, p. 80, Mar 2018

- Fuentes utilizadas durante el desarrollo de la aplicación
    - https://sequelize.org/docs/v6/getting-started/

    - https://www.bezkoder.com/node-js-jwt-authentication-mysql/

    - https://jitsi.github.io/handbook/docs/dev-guide/dev-guide-react-sdk

    - https://github.com/coreui/coreui-react

    - https://medium.com/swlh/automatic-api-documentation-in-node-js-using-swagger-dd1ab3c78284

    - http://rajeevdotnet.blogspot.com/2019/02/export-swagger-api-document-to-pdf.html