UNIVERSIDAD DE VALLADOLID
E.T.S.I DE TELECOMUNICACIÓN

TRABAJO FINAL DE GRADO
GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Plataforma de hábitos saludables para personas con problemas de salud mental

Autor: Álvaro Vaquero Giménez
Tutor: Alfonso Bahillo Martínez

28 de junio de 2023

TÍTULO:	Plataforma de hábitos saludables para personas con problemas de salud mental
AUTOR:	Álvaro Vaquero Giménez
TUTOR:	Alfonso Bahillo Martínez
DEPARTAMENTO:	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTE:	D. Rubén M. Lorenzo Toledo
VOCAL:	D. Ramón J. Durán Barroso
SECRETARIO:	D. Alfonso Bahillo Martínez
SUPLENTE:	D. Juan C. Aguado Manzano
SUPLENTE:	D.ª Patricia Fernández Del Reguero
FECHA:	5 de julio de 2023
CALIFICACIÓN:	

Resumen

El proyecto se centra en el desarrollo de una aplicación web dirigida a profesionales sanitarios, diseñada para registrar y observar la evolución de diferentes parámetros fisiológicos de pacientes con problemas de salud mental, como el peso, perímetro abdominal y tensión arterial, entre otros. Además, la aplicación permite la prescripción de una dieta semanal (desayuno, comida, merienda y cena) y una rutina de ejercicios personalizada, así como recoger automáticamente el feedback de los pacientes sobre las dietas y rutinas prescritas.

La aplicación web forma parte de un proyecto más amplio que incluye una aplicación móvil dirigida a los pacientes. La aplicación móvil proporciona a los pacientes acceso a la información asignada, como las comidas y los ejercicios, desde sus dispositivos móviles. Además, permite a los pacientes valorar las comidas y ejercicios recibidos, y a los profesionales poder ver dichas valoraciones en la aplicación web, mejorando así la comunicación y el seguimiento del tratamiento.

En cuanto a las plataformas utilizadas, el proyecto se compone de dos elementos principales: una aplicación web o frontend que utiliza la tecnología Angular para la interfaz de usuario, y un backend que se desgrana en una base de datos MongoDB, elegida por su flexibilidad, y una API desarrollada en NestJS.

Este proyecto está destinado a ser evaluado y validado para su posterior explotación a nivel regional y nacional en el Servicio de Psiquiatría y Salud Mental del Complejo Público Asistencial de Zamora, donde se atiende a personas con problemas de salud mental. La aplicación web proporcionará a los profesionales sanitarios una herramienta personalizada para el seguimiento de los parámetros fisiológicos de los pacientes y la prescripción de dietas y rutinas de ejercicio físico,

además del seguimiento sobre el feedback de los pacientes en relación a las dietas y rutinas prescritas.

PALABRAS CLAVE: Aplicación Web, Angular, NestJS, MongoDB, salud mental, nutrición, ejercicio.

Abstract

The project focuses on the development of a web application aimed at healthcare professionals, designed to record and observe the evolution of different physiological parameters of patients with mental health issues, such as weight, abdominal circumference, and blood pressure, among others. Additionally, the application allows for the prescription of a weekly diet plan (breakfast, lunch, snack, and dinner) and a personalized exercise routine, as well as automatically collecting feedback from patients regarding the prescribed diets and routines.

The web application is part of a broader project that includes a mobile application targeted at the patients. The mobile application provides patients with access to assigned information, such as meals and exercises, from their mobile devices. Furthermore, it allows patients to rate the received meals and exercises, while professionals can view these ratings in the web application, thereby improving communication and treatment monitoring.

Regarding the platforms used, the project consists of two main elements: a web application or frontend that utilizes Angular technology for the user interface, and a backend that consists of a MongoDB database chosen for its flexibility, along with an API developed in NestJS.

This project is intended to be evaluated and validated for subsequent regional and national implementation in the Psychiatry and Mental Health Service of the Zamora Public Assistance Complex, where individuals with mental health problems are treated. The web application will provide healthcare professionals with a personalized tool for monitoring patients' physiological parameters and prescribing

diets and exercise routines, in addition to tracking patient feedback regarding the prescribed diets and routines.

KEYWORDS: Web Application, Angular, NestJS, MongoDB, mental health, nutrition, exercise.

Índice general

Índice de figuras **xiii**

Índice de tablas **xvii**

1 Introducción **1**
1.1 Motivación 1
1.2 Introducción al proyecto 2
1.3 Objetivos 3
1.4 Fases de desarrollo 3
1.5 Estructura de la memoria 4

2 Estado del Arte **7**
2.1 Tecnologías utilizadas 7
2.1.1 Angular 8
2.1.2 NestJS 10
2.1.3 MongoDB 10
2.2 Herramientas utilizadas 12

3 Análisis y Diseño **17**
3.1 Descripción de los actores 18
3.2 Glosario de términos 19
3.3 Requisitos funcionales 20
3.4 Casos de uso 21
3.4.1 Gestión de pacientes 22
3.4.2 Gestión de profesionales 23
3.4.3 Gestión de recetas 24
3.4.4 Gestión de rutinas 25

3.4.5 Gestión de dietas . . . 25
3.4.6 Gestión de Comidas . . . 27
3.4.7 Gestión de Ejercicios . . . 28
3.4.8 Gestión de Consultas . . . 29
3.4.9 Gestión de Autenticación . . . 30
3.5 Modelo de datos . . . 32
3.6 Diccionario de datos . . . 33
3.6.1 Employee . . . 34
3.6.2 Patient . . . 34
3.6.3 Recipe . . . 35
3.6.4 Routine . . . 37
3.6.5 Diet . . . 37
3.6.6 Food . . . 38
3.6.7 Move . . . 40
3.6.8 Consult . . . 41
3.6.9 Attachment . . . 42
3.7 Arquitectura del sistema . . . 43

4 Desarrollo **45**
4.1 Desarrollo de la API . . . 45
4.1.1 Estructura de directorios . . . 45
4.1.2 Peticiones HTTP . . . 48
4.2 Desarrollo de la aplicación web . . . 55
4.2.1 Estructura de directorios . . . 55
4.2.2 Navegación . . . 58

5 Manual de Usuario **63**
5.1 Pantalla de login . . . 64
5.2 Pantalla de pacientes . . . 66
5.3 Pantalla de profesionales . . . 67
5.4 Pantalla de recetas . . . 68
5.5 Pantalla de rutinas . . . 70
5.6 Pantalla de dietas . . . 70
5.7 Pantalla del paciente . . . 73
5.7.1 Página de gráficas . . . 73
5.7.2 Página de consultas . . . 74
5.7.3 Página de comidas . . . 75

5.7.4 Página de ejercicios 77
5.8 Pantalla de configuración 79

6 Manual del Desarrollador **81**
6.1 Entorno de producción 81
6.1.1 Aplicación web 82
6.1.2 API 83
6.2 Variables de entorno 84

7 Conclusiones **85**
7.1 Resultados finales 85
7.2 Líneas futuras 86
7.3 Valoración personal 86

Índice de código fuente **89**

Bibliografía **91**

ÍNDICE DE FIGURAS

Figura 1.1 Imagotipo de Food&Move 3

Figura 2.1 Icono de Angular 9
Figura 2.2 Icono de NestJS 10
Figura 2.3 Icono de MongoDB 11
Figura 2.4 Icono de Visual Studio Code 12
Figura 2.5 Icono de Postman 12
Figura 2.6 Icono de Git 13
Figura 2.7 Icono de GitHub 13
Figura 2.8 Icono de OpenVPN 13
Figura 2.9 Icono de PuTTY 14
Figura 2.10 Icono de StarUML 14
Figura 2.11 Icono de Overleaf 14
Figura 2.12 Icono de MongoDB Atlas 15

Figura 3.1 Diagrama de los actores 19
Figura 3.2 Diagrama de CU de gestión de pacientes 22
Figura 3.3 Diagrama de CU de gestión de profesionales 23
Figura 3.4 Diagrama de CU de gestión de recetas 24
Figura 3.5 Diagrama de CU de gestión de rutinas 25
Figura 3.6 Diagrama de CU de gestión de dietas 26
Figura 3.7 Diagrama de CU de gestión de comidas 27
Figura 3.8 Diagrama de CU de gestión de ejercicios 28
Figura 3.9 Diagrama de CU de gestión de consultas 29
Figura 3.10 Diagrama de CU de gestión de autenticación 30
Figura 3.11 Modelo de datos 32

Figura 3.12 Arquitectura del sistema 43

Figura 4.1 Árbol de directorios de la API 46
Figura 4.2 Módulos de la API 48
Figura 4.3 Módulo Patients 48
Figura 4.4 Árbol de directorios de la app 56
Figura 4.5 Directorio app 57
Figura 4.6 Navegación de la aplicación 59
Figura 4.7 Módulo de enrutamiento principal 60
Figura 4.8 Guardian AuthGuard 60

Figura 5.1 Pantalla de login 64
Figura 5.2 Pantalla de recuperar contraseña 65
Figura 5.3 Pantalla de cambiar contraseña 65
Figura 5.4 Pantalla de pacientes 66
Figura 5.5 Pantalla de agregar/editar un paciente 67
Figura 5.6 Pantalla de profesionales 67
Figura 5.7 Pantalla de agregar/editar un profesional 68
Figura 5.8 Pantalla de recetas 68
Figura 5.9 Pantalla de agregar/editar una receta 69
Figura 5.10 Ventana de importar pdf 69
Figura 5.11 Pantalla de rutinas 70
Figura 5.12 Pantalla de agregar/editar una rutina 70
Figura 5.13 Pantalla de dietas 71
Figura 5.14 Pantalla de editar una dieta 71
Figura 5.15 Pantalla de agregar comida 72
Figura 5.16 Ventana de importar receta 72
Figura 5.17 Página de las gráficas de un paciente 73
Figura 5.18 Página de las consultas de un paciente 74
Figura 5.19 Página de agregar/editar una consulta 74
Figura 5.20 Página de comidas de un paciente 75
Figura 5.21 Ventana de importación de dieta 76
Figura 5.22 Página de agregar/editar comida de un paciente 76
Figura 5.23 Página de ejercicios de un paciente 77
Figura 5.24 Página de agregar/editar un ejercicio de un paciente 78
Figura 5.25 Ventana de importación de rutina 78
Figura 5.26 Ventana del profesional registrado 79

Figura 5.27 Edición del perfil del usuario 79
Figura 5.28 Cambio de contraseña del usuario 79

Figura 6.1 Configuración de NGINX 82
Figura 6.2 Subir a producción la aplicación web 83
Figura 6.3 Subir a producción la API 83
Figura 6.4 Estado de pm2 84
Figura 6.5 Dirección de la API 84
Figura 6.6 Dirección de la base de datos 84

ÍNDICE DE TABLAS

Tabla 3.2 Glosario 20
Tabla 3.4 Tabla diccionario de datos de 'Employee' 34
Tabla 3.6 Tabla diccionario de datos de 'Patient' 35
Tabla 3.8 Tabla diccionario de datos de 'Recipe' 36
Tabla 3.10 Estructura de los ingredientes 37
Tabla 3.12 Tabla diccionario de datos de 'Routine' 37
Tabla 3.14 Tabla diccionario de datos de 'Diet' 38
Tabla 3.16 Tabla diccionario de datos de 'Food' 39
Tabla 3.18 Tabla diccionario de datos de 'Move' 40
Tabla 3.20 Tabla diccionario de datos de 'Consult' 42
Tabla 3.22 Tabla diccionario de datos de 'Attachment' 42

Tabla 4.2 Peticiones HTTP del módulo 'Employees' 49
Tabla 4.4 Peticiones HTTP del módulo 'Patients' 50
Tabla 4.6 Peticiones HTTP del módulo 'Recipes' 51
Tabla 4.8 Peticiones HTTP del módulo 'Routines' 51
Tabla 4.10 Peticiones HTTP del módulo 'Foods' 52
Tabla 4.12 Peticiones HTTP del módulo 'Moves' 53
Tabla 4.14 Peticiones HTTP del módulo 'Consults' 54
Tabla 4.16 Peticiones HTTP del módulo 'Attachments' 54
Tabla 4.18 Peticiones HTTP del módulo 'Auth' 55
Tabla 4.20 Peticiones HTTP del módulo 'Files' 55

CAPÍTULO 1

INTRODUCCIÓN

1.1 Motivación 1
1.2 Introducción al proyecto 2
1.3 Objetivos 3
1.4 Fases de desarrollo 3
1.5 Estructura de la memoria 4

1.1 Motivación

La salud mental (SM) es uno de los pilares de la calidad de vida de los ciudadanos. Es sabido que tras el COVID19, la SM de los europeos se ha visto comprometida constituyéndose en una epidemia que ha incrementado las tasas de suicidio, aumento de la demanda de los servicios de SM y reducción de la atención a las personas con enfermedad mental grave y prolongada (EMGP), si bien ha posibilitado un mayor desarrollo de las nuevas tecnologías en salud y de la atención a distancia.

Igualmente, uno de los colectivos más perjudicados en términos de accesibilidad y equidad de atención sanitaria son las personas que viven en el medio rural, en el que las poblaciones son pequeñas con pocos servicios, no buenas comunicaciones y por tanto, con muchas limitaciones para acceder a las prestaciones de SM, con incremento de suicidios, abandono de tratamiento y de falta de continuidad de cuidados en EMGP

y a una baja accesibilidad al tratamiento, especialmente entre las personas mayores (alta prevalencia en el territorio).

Ante esta situación, las actuaciones habituales ha sido el que los cuidadores familiares asumieran el cuidado sin apenas apoyo y con poca continuidad de cuidados y en casos extremos el desplazamiento e institucionalización de las personas con EMGP.

1.2 Introducción al proyecto

El proyecto, cuya marca comercial es "Food&Move", se centra en el ámbito de los hábitos saludables, como una alimentación equilibrada y la práctica habitual de ejercicio físico, con el objetivo de ayudar a las personas con problemas de salud mental a manejar y reducir los síntomas asociados a unos malos hábitos saludables.

El proyecto está divido en dos herramientas: una aplicación web destinada a los profesionales, y una aplicación móvil destinada a los pacientes.

En cuanto a la aplicación web, que será en lo que nos centraremos en este TFG, se encarga de permitir a los profesionales hacer un seguimiento semanal de ciertos parámetros fisiológicos del paciente, como son: peso, perímetro abdominal, tensión arterial, entre otros. Así mismo, la aplicación les permite registrar distintas dietas y ejercicios de cada paciente en base a la evolución temporal de esos parámetros fisiológicos.

Por otro lado, la aplicación móvil permite a los pacientes hacer un seguimiento de la dieta semanal, con ejemplos prácticos a través de vídeos para la elaboración de las comidas, y las rutinas de ejercicios que han sido asignados por el profesional. El paciente podrá evaluar al final del día su grado de satisfacción y cumplimiento tanto de la dieta como del ejercicio. Los profesionales podrán ver dicha valoración en la aplicación web.

En resumen, esta herramienta intenta brindar apoyo tanto a los profesionales en la gestión del seguimiento de las dietas y ejercicios de sus pacientes, como a los propios pacientes para que puedan seguir su tratamiento más fácilmente.

Figura 1.1: Imagotipo de Food&Move

1.3 Objetivos

Mi trabajo en este proyecto consiste en desarrollar una primera versión de la aplicación web, donde los profesionales sanitarios pueden interactuar con el sistema, y la parte del backend, constituida por una API y una base de datos.

El objetivo principal es crear una base fuerte y consistente del proyecto donde los profesionales puedan asignar dietas semanales y rutinas de ejercicios personalizados a cada paciente y puedan hacer un seguimiento de ciertos parámetros fisiológicos.

Posteriormente, ya fuera del objetivo de este trabajo, el proyecto completo, tanto la aplicación web como la aplicación móvil, será evaluado y validado en el Servicio de Psiquiatría y Salud Mental del Complejo Público Asistencial de Zamora para su posterior explotación a nivel regional y nacional.

1.4 Fases de desarrollo

Durante el desarrollo de este trabajo se ha llevado a cabo las siguientes fases:

- **Estudio:** Al inicio del proyecto tuve un par de reuniones con el tutor y el centro especializado de Zamora, donde pude entender en qué consistía el proyecto y sacar los principales objetivos del mismo. En esta fase pude organizar las distintas fases del proyecto y decidir que tecnologías y herramientas iba a utilizar.
- **Análisis y diseño**: A través de esas reuniones puede sacar el conjunto de requisitos y funcionalidades necesarias para el sistema. A partir de esto, pude analizar los actores que intervienen en el sistema, las relaciones existentes entre cada funcionalidad y diseñar la estructura que presenta la base de datos. También pude diseñar cómo sería la navegación de la aplicación y la interfaz de cada una de las páginas.

- **Desarrollo**: En esta fase pude implementar el código de la parte de la aplicación web y la programación de la API con el conjunto de peticiones HTTP y la lógica correspondiente a cada petición.
- **Pruebas**: Esta fase es paralela a la fase de desarrollo, ya que cada vez que iba creando una nueva funcionalidad, realizaba pruebas sobre esta para poder encontrar posibles errores y posteriormente corregirla.
- **Implementación**: Una vez terminado la fase de desarrollo, se continuó el proyecto alojando las aplicaciones en un servidor en un entorno de producción. De esta manera se pudo comprobar el correcto funcionamiento entre las dos partes del proyecto, la aplicación web y la aplicación móvil.
- **Documentación**: Esta última fase consiste en el desarrollo de este documento, con la cual ayudará al mantenimiento y creación de nuevas versiones del proyecto.

1.5 Estructura de la memoria

Para entender más fácilmente cómo se ha llevado a cabo el desarrollo del proyecto, he optado por estructurar este documento de la siguiente manera:

En este primer capítulo se ha realizado una motivación e introducción del proyecto, explicando brevemente de qué trata y qué se espera obtener de la aplicación.

El segundo capítulo contiene el estado del arte, donde presenta las distintas tecnologías que han sido elegidas para desarrollar el proyecto, así como las distintas herramientas que se han utilizado.

En el tercer capítulo recoge el análisis y diseño que se ha llevado a cabo al inicio del proyecto para el desarrollo del mismo. En este capítulo se presentan varios apartados como son los requisitos funcionales del sistema, los casos de uso (CU), el modelo de datos, entre otros.

El cuarto capítulo contiene el desarrollo del proyecto, tanto de la aplicación web como la API. En este se verán puntos como la estructuración de directorios, las peticiones HTTP desarrolladas en la API y la navegación de la aplicación web.

En el quinto capítulo se muestran las distintas páginas e interfaces que presenta la aplicación. En este se va detallando las distintas funcionalidades que presenta cada una de las páginas y unas breves instrucciones para llevarlas a cabo.

El sexto capítulo expone el proceso de cómo llevar el proyecto a un entorno de producción. Este muestra unas instrucciones de cómo alojar en el servidor tanto la aplicación web como la API.

El séptimo y último capítulo contiene las conclusiones donde se expone los resultados finales del proyecto, un listado con distintos puntos de mejora para desarrollar en versiones posteriores y donde doy una opinión personal sobre el trabajo.

CAPÍTULO 2

ESTADO DEL ARTE

2.1 Tecnologías utilizadas 7
2.1.1 Angular 8
2.1.2 NestJS 10
2.1.3 MongoDB 10
2.2 Herramientas utilizadas 12

En este capítulo, nos centraremos en las tecnologías seleccionadas para el desarrollo del proyecto, así como las herramientas utilizadas para hacerlo posible. Analizaremos en detalle estas tecnologías y exploraremos las herramientas que se han empleado en cada etapa del proyecto.

2.1 Tecnologías utilizadas

En este proyecto, se identifican tres componentes principales: una aplicación web para interactuar con el usuario, una base de datos para almacenar los datos y una API para facilitar la comunicación entre la página web y la base de datos. En esta sección, se analizarán las diferentes tecnologías utilizadas en cada uno de estos componentes y se realizará una comparación con otras opciones disponibles, con el objetivo de explicar la elección de estas tecnologías en lugar de otras alternativas.

2.1.1 Angular

Antes de comprender el papel que desempeña angular en nuestro proyecto, es esencial comprender la estructura de una plataforma web.

Cuando un usuario accede a una dirección web, se realiza una solicitud al servidor que contiene la plataforma a visitar. Esta interacción entre los usuarios y los servidores requiere el uso de diferentes tecnologías en ambos extremos: el lado del usuario, conocido como "frontend", y el lado del servidor, conocido como "backend".

En el caso de este proyecto, Angular [1] se encuentra en el lado del frontend. Es un framework desarrollado y mantenido por Google que se utiliza para crear aplicaciones web en el lado del cliente. Angular se basa en TypeScript, que es un superconjunto de JavaScript, el lenguaje principal utilizado en el desarrollo web.

Angular, antes conocido como AngularJS, se inició en 2010. AngularJS ofrecía controladores y directivas para dividir el código JavaScript y gestionar la interfaz gráfica de usuario y las solicitudes al servidor por separado. En su segunda versión, el equipo de desarrollo decidió simplificar el nombre del framework a Angular.

Uno de los principales objetivos de Angular es reforzar la estructura del Modelo-Vista-Controlador (MVC) [19] en el desarrollo web y las Single Page Applications (SPA). El MVC es una arquitectura que separa el código por sus distintas responsabilidades, donde la vista es la interfaz gráfica de usuario y cambia en función del controlador que gestiona su contenido. Las Single Page Applications son páginas web que cargan un único archivo HTML en el cliente y cuyo contenido se actualiza dinámicamente a medida que el usuario interactúa con ella, lo que conlleva a un mejor tiempo de reacción.

Se ha optado por elegir Angular debido a las numerosas ventajas que nos aporta a nuestro proyecto. Algunas de estas ventajas son:

- Facilita la estructuración y organización del código, mejorando el mantenimiento y escalabilidad de la aplicación.
- Permite la generación de componentes reutilizables.
- Cuenta con una comunidad activa y un amplio ecosistema de herramientas y bibliotecas.

- Aporta una experiencia de usuario fluida y rápida al construir aplicaciones de una sola página.

Figura 2.1: Icono de Angular

Aparte de Angular, existen diversas tecnologías disponibles para el desarrollo web. A continuación, se presenta alguna de ellas:

React

React [15] es una librería de JavaScript creada por Facebook en 2011, aunque no fue declarada de código abierto hasta 2013. Esta librería ha sido creada con el objetivo de facilitar la construcción de interfaces de usuario y aprovechar su funcionalidad en el desarrollo de aplicaciones de una sola página (SPA).

React es bastante flexible con ventajas como el DOM virtual y la adaptabilidad con otras librerías y herramientas. Por otro lado, Angular es una solución más completa.

Vue.js

Vue.js [18] es un framework de JavaScript para la creación de interfaces de usuario y aplicaciones de una sola página (SPA). Creado por Evan You en 2014. La curva de aprendizaje de esta tecnología es la más sencilla de los frameworks más populares: React, Vue y Angular.

Esta tecnología, aparte de ser sencilla, es bastante rápida. Sin embargo, Angular es mucho más robusto.

2.1.2 NestJS

NestJS [9] es un framework progresivo de NodeJS desarrollado en TypeScript, aunque también permite desarrollar aplicaciones en código JavaScript, diseñado para facilitar el desarrollo de aplicaciones backend, aportando una buena estructura y metodología inicial.

NodeJS es un programa ideal para desarrollar aplicaciones web aportando un muy buen rendimiento y alta escalabilidad, sin embargo, el desarrollo del mismo puede ser una tarea complicada si no se tiene una buena metodología y una buena estructura. NestJS soluciona este problema aportando una arquitecutra clara, facilitando el desarrollo y mantenimiento de las aplicaciones.

NestJS es la tecnología que se ha optado para desarrollar la API de nuestro proyecto debido, además de la ventaja mencionada anteriormente, a que usar el mismo lenguaje de programación que en el frontend (TypeScript) hace que la curva de aprendizaje del mismo sea más sencilla.

Figura 2.2: Icono de NestJS

Existen otras tecnologías que sirven como alternativa para desarrollar la parte del backend como LoopBack, Sails.js, Express.js, entre otros. Sin embargo, se ha optado por NestJS por su comunidad activa, su sintaxis parecida a Angular y su extensa documentación de fácil comprensión.

2.1.3 MongoDB

MongoDB [8] es una base de datos NoSQL orientado a documentos [4]. Para comprender mejor esto, vamos a ver el significado de estos dos conceptos.

NoSQL (Not Only SQL) se refiere a una base de datos no relacional. A diferencia de las bases de datos relacionales, que se basan en tablas, los sistemas NoSQL utilizan diferentes modelos para almacenar y recuperar información [17].

Algunas de las principales características de las bases de datos NoSQL son:

- Permite almacenar datos estructurados, no estructurados y semiestructurados, lo que aporta una gran flexibilidad.
- Utiliza una escalabilidad horizontal, lo que le permite manejar gran volúmenes de datos distribuyendo la carga en varios servidores, aportando un alto rendimiento.
- Está diseñado para ser altamente disponible, lo que permite mantener el acceso a los datos aunque haya algún fallo en algún servidor individual.

MongoDB está orientado a documentos, esto quiere decir que almacena los datos en forma de documentos. Estos suelen ser de tipo JSON (JavaScript Object Notation) que no presentan un equema concreto, lo que brinda una gran versatilidad. Otra ventaja de este tipo de bases de datos es que nos aporta una gran capacidad de consulta, pudiendo realizar consultas complejas y de forma rápida.

Se ha elegido MongoDB como base de datos debido a todas estas ventajas y a su fácil implementación en la API (NestJS).

Figura 2.3: Icono de MongoDB

2.2 Herramientas utilizadas

En esta sección se presentará las distintas herramientas que se ha utilizado durante el desarrollo del proyecto.

Visual Studio Code

Visual Studio Code [5] es un editor de código utilizado para el desarrollo tanto de la aplicación web como la API.

Figura 2.4: Icono de Visual Studio Code

Postman

Postman [14] es una herramienta de desarrollo de API que permite probar, documentar y realizar solicitudes a servicios web. Esta herramienta me ha ayudado para realizar las distintas peticiones HTTP a la API desarrollada y comprobar su correcto funcionamiento.

Figura 2.5: Icono de Postman

Git

Git [6] es un sistema de control de versiones. Utilizado para el seguimiento y gestión de los cambios del proyecto.

Figura 2.6: Icono de Git

GitHub

GitHub [7] es una plataforma de alojamiento y gestión de repositorios de código fuente basada en el sistema de control de versiones Git. En ella se encuentra alojado los repositorios git tanto de la aplicación web como la API.

Figura 2.7: Icono de GitHub

OpenVPN

OpenVPN [11] es una plataforma que permite establecer conexiones seguras y privadas a través de redes no seguras. Esto es necesario para poder conectar con el servidor donde se encunetran alojados la aplicación web y la API en la fase de implementación.

Figura 2.8: Icono de OpenVPN

PuTTY

PuTTY es un programa que funciona como un cliente de terminal y emulador de terminal. Este nos permite conectarnos de forma remota a través de SSH al servidor donde se encuentran alojados los componentes de nuestro proyecto.

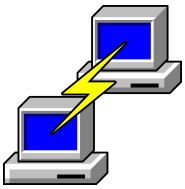

Figura 2.9: Icono de PuTTY

StarUML

StarUML [16] es una herramienta de modelado de software que permite crear diagramas UML para el diseño y la visualización de sistemas y aplicaciones. Este me ha ayudado en el desarrollo de los diagramas de CU y el modelo de datos para poder documentar este proyecto.

Figura 2.10: Icono de StarUML

Overleaf

Overleaf [12] es la plataforma en línea encargada de la edición de documentos LaTeX. Empleado para la creación de este documento.

Figura 2.11: Icono de Overleaf

MongoDB Atlas

MongoDB Atlas [3] es un servicio de base de datos en la nube que ofrece MongoDB. Aquí es donde está alojado la base de datos del proyecto.

Figura 2.12: Icono de MongoDB Atlas

CAPÍTULO 3

ANÁLISIS Y DISEÑO

3.1 Descripción de los actores 18
3.2 Glosario de términos 19
3.3 Requisitos funcionales 20
3.4 Casos de uso 21
3.4.1 Gestión de pacientes 22
3.4.2 Gestión de profesionales 23
3.4.3 Gestión de recetas 24
3.4.4 Gestión de rutinas 25
3.4.5 Gestión de dietas 25
3.4.6 Gestión de Comidas 27
3.4.7 Gestión de Ejercicios 28
3.4.8 Gestión de Consultas 29
3.4.9 Gestión de Autenticación 30
3.5 Modelo de datos 32
3.6 Diccionario de datos 33
3.6.1 Employee 34
3.6.2 Patient 34
3.6.3 Recipe 35
3.6.4 Routine 37
3.6.5 Diet 37
3.6.6 Food 38

3.6.7 Move 40
3.6.8 Consult 41
3.6.9 Attachment 42
3.7 Arquitectura del sistema 43

En el presente capítulo se presenta la fase del análisis y el diseño del sistema, una etapa importante para el desarrollo de cualquier proyecto.

Comenzaremos por identificar quiénes son los actores o usuarios que intervienen en la aplicación.

Posteriormente, se presentará un pequeño glosario que recopila un conjunto de términos que serán utilizados a lo largo del documento.

A continuación, nos adentraremos en los distintos requisitos y funcionalidades (CU) que debería presentar el proyecto.

En la siguiente sección, se abordará el modelo y diccionario de datos, donde se mostrará la estructura de datos que utiliza el sistema.

Por último, se presentará la arquitectura del sistema, donde se especificará la relación de los distintos componentes que lo conforman.

3.1 Descripción de los actores

Antes de ver las funcionalidades o requisitos que debe cumplir el sistema, considero importante conocer los distintos actores que interactúan con la aplicación.

Esta aplicación web está destinada para el uso exclusivo de los profesionales sanitarios, por tanto, son ellos quienes consideramos los actores del sistema.

Dentro de los usuarios, podemos encontrar dos grupos: profesionales y administradores. Estos últimos tendrán acceso a todas las funcionalidades del sistema.

Por lo tanto, podemos encontrar los siguientes actores:

- **Usuario no autenticado:** Usuario que aún no ha iniciado sesión.

- **Profesional:** Usuario registrado en la aplicación con funcionalidades específicas dentro del sistema.
- **Administrador:** Usuario registrado en la aplicación con privilegios dentro del sistema.

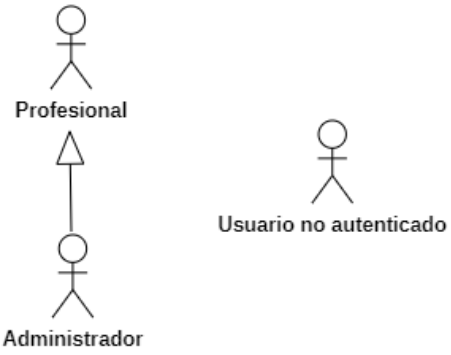

Figura 3.1: Diagrama de los actores

3.2 Glosario de términos

En este apartado, se representa un glosario con un conjunto de definiciones de los términos que se van a utilizar durante el diseño y desarrollo de este proyecto.

Esto se puede observar en una tabla con tres campos. El primer campo hace referencia al término al que queremos hacer referencia, el segundo campo ofrece una breve descripción de dicho término, y el último campo proporciona el nombre utilizado tanto en la base de datos como en la programación, siendo esta último columna en inglés debido al enfoque en el uso de dicho idioma en el ámbito de la programación.

Término	Definición e Información	Programación
Profesional	Enfermero/a encargado de administrar a sus pacientes	Employee
Administrador	Profesionales con ciertos privilegios. Estos pueden administrar el resto de enfermeros	Admin
Paciente	Persona que recibe atención por parte de los profesionales	Patient

Término	Definición e Información	Programación
Receta	Plato de comida previamente establecido que puede ser importado en los menús individuales de cada paciente	Recipe
Rutina	Ejercicio o actividad física previamente establecido que puede ser importado en los registros de ejercicios individuales de cada paciente	Routine
Dieta semanal	Selección de platos previamente establecidos que pueden ser importados en el menú semanal de cada paciente	Diet
Comida	Plato de comida que se asigna para un día concreto en el menú de cada paciente	Food
Ejercicio	Actividad física que se asigna para un día concreto en los registros de ejercicios de cada paciente	Move
Consulta	Seguimiento médico de los distintos parámetros fisiológicos en una fecha concreta	Consult

Tabla 3.2: Glosario

3.3 Requisitos funcionales

Los requisitos funcionales son un conjunto de características requeridas por el sistema que expresa una capacidad de acción del mismo. En esta sección, presentaremos un listado con los principales requisitos que demanda nuestro sistema:

- El sistema debe de permitir a los administradores crear, modificar y eliminar los distintos profesionales, así como asignarles un rol (administrador o profesional).
- Los profesionales deben poder crear, modificar y eliminar sus propios pacientes.

- Los administradores deben poder gestionar a sus propios pacientes y al del resto de profesionales.
- Los profesionales deben poder iniciar sesión con su email y contraseña, así como poder cambiar de contraseña en el caso de olvido.
- Los profesionales deben poder registrar consultas agregando los distintos parámetros fisiológicos medidos. Estos son: masa, índice de masa corporal, perímetro abdominal, tensión arterial, triglicéridos séricos, colesterol bueno (HDL), colesterol malo (LDL), hemoglobina glicosilada y glucosa en plasma.
- El sistema debe representar la evolución temporal de cada uno de los parámetros fisiológicos medidos en las consultas de cada paciente.
- Los profesionales deben tener la capacidad de asignar comidas (desayuno, comida, merienda y cena) semanales, con opción a editar comidas puntuales, a sus pacientes.
- Los profesionales deben poder asignar ejercicios físicos diarios a sus pacientes.
- Los profesionales deben poder visualizar las valoraciones de las comidas y rutinas de ejercicios hechas por cada paciente.
- Los profesionales deben tener la capacidad de poder modificar sus datos personales, así como poder cambiar su contraseña.

3.4 Casos de uso

En este apartado vamos a identificar y describir los distintos CU que permiten a los actores interactuar con el sistema.

Al haber bastantes CU, he optado en dividir esta sección en varios escenarios:

- Gestión de pacientes [3.4.1]
- Gestión de profesionales [3.4.2]
- Gestión de recetas [3.4.3]
- Gestión de rutinas [3.4.4]
- Gestión de dietas [3.4.5]

- Gestión de comidas [3.4.6]
- Gestión de ejercicios [3.4.7]
- Gestión de consultas [3.4.8]
- Gestión de autenticación [3.4.9]

A continuación, exploraremos estos escenarios y nos adentraremos en detalle en los distintos casos de uso que los componen.

3.4.1 Gestión de pacientes

A cada paciente se le es asignado un profesional. Por lo tanto, el sistema tiene que permitir a los profesionales poder gestionar a todos sus pacientes. Este escenario está constituido por varios CU, como son:

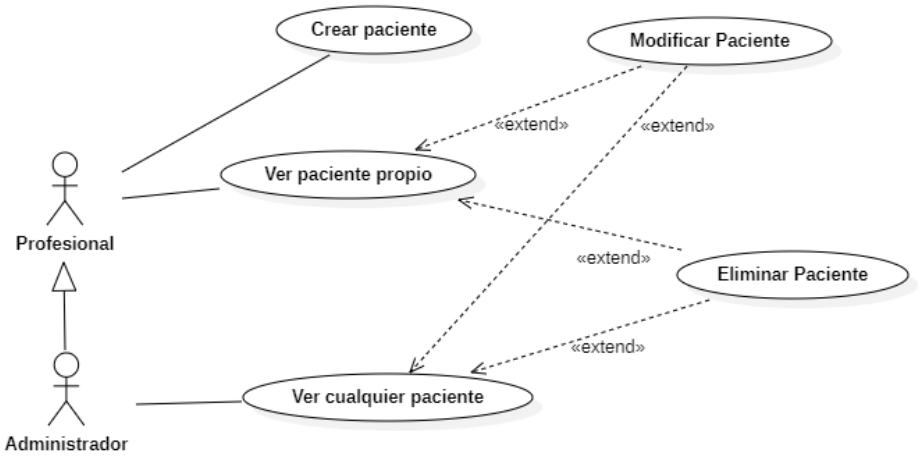

Figura 3.2: Diagrama de CU de gestión de pacientes

- **Crear paciente:** Los profesionales podrán crear sus propios pacientes. Para ello, el profesional debe rellenar un formulario con los distintos campos del paciente: nombre, apellido, email, teléfono, altura, fecha de nacimiento, contraseña y foto de perfil. Siendo el nombre, teléfono y contraseña obligatorios.

  La contraseña se podrá generar de forma aleatoria.

- **Modificar paciente:** Una vez creado el paciente, el profesional podrá modificar cualquier dato del mismo. El profesional podrá visualizar un formulario autocompletado con los distintos datos del paciente a modificar.
- **Eliminar paciente:** Los profesionales podrán eliminar a los pacientes. Para asegurar que no ha pulsado el botón de eliminar sin querer, se mostrará una ventana de confirmación. Una vez eliminado dicho paciente, se eliminará todas las consultas, comidas y ejercicios que se le fue asignado a dicho paciente.
- **Ver paciente propio:** Los profesionales podrán visualizar a todos sus pacientes. Son a ellos a los que pueden modificar y eliminar.
- **Ver cualquier paciente:** Los profesionales que son asignados como administradores podrán visualizar a todos los pacientes aparte de los suyos. Por consecuencia, podrán modificar y eliminar a cualquier paciente.

3.4.2 Gestión de profesionales

El sistema debe permitir gestionar los profesionales que van a tener acceso a la aplicación y tener uso de ella. Los encargados de esta tarea son aquellos profesionales que son asignados como administradores.

Los casos de uso de dicho escenario son los siguientes:

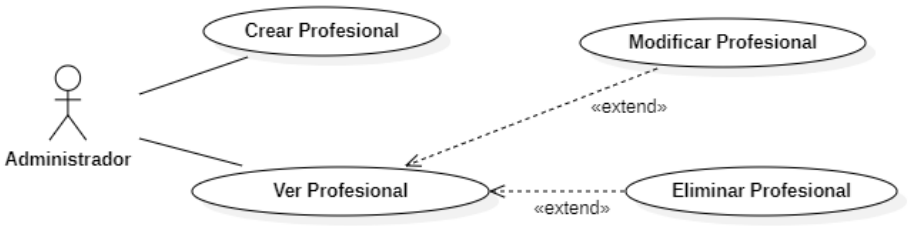

Figura 3.3: Diagrama de CU de gestión de profesionales

- **Crear profesional:** Los administradores podrán crear nuevos profesionales. Para ello, deben de rellenar un formulario con los distintos campos del profesional: nombre, apellido, email, teléfono, foto de perfil y si es o no administrador. Siendo el nombre y email obligatorios.

Una vez rellenado el formulario, le llegará un email al profesional creado con una contraseña aleatoria con la que podrá acceder a la aplicación. Posteriormente, este podrá cambiarla en la pantalla de configuración.

- **Modificar profesional:** Una vez creado los profesionales, los administradores podrán modificar cual dato de dichos profesionales.
- **Eliminar profesional:** Los administradores tendrán la capacidad de eliminar a cualquier profesional del sistema. Para aseguarse de que no ha pulsado el botón de eliminar sin querer, se mostrará una ventana de confirmación. Una vez eliminado el profesional, este no podrá acceder a la aplicación.
- **Ver profesional:** Los administradores podrán visualizar a todos los profesionales registrados en el sistema.

3.4.3 Gestión de recetas

Uno de los principales objetivos de la aplicación es asignar platos de comida a cada paciente. Considerando que cada paciente debe tener un desayuno, almuerzo, merienda y cena todos los días, sería una tarea tediosa tener que crear manualmente todos los platos cada vez que se desea asignar uno. Por lo tanto, se ha decidido implementar recetas predefinidas por los profesionales, de manera que solo sea necesario importarlas para asignar el plato correspondiente al paciente. Es por ello que existe un escenario dedicado a la gestión de recetas. Los casos de uso son los siguientes:

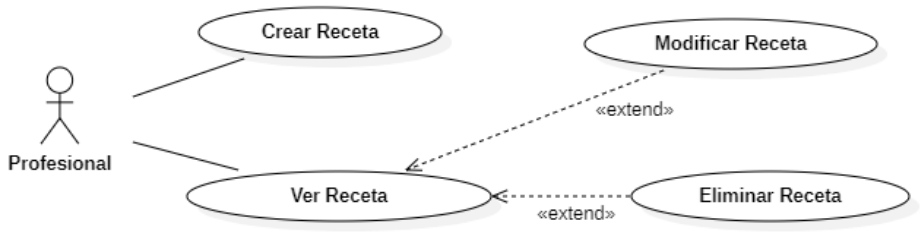

Figura 3.4: Diagrama de CU de gestión de recetas

- **Crear receta:** Los profesionales podrán crear nuevas recetas.
- **Modificar receta:** Todas las recetas creadas podrán ser modificadas por los profesionales.
- **Eliminar receta:** Los profesionales podrán eliminar cualquier receta.

- **Ver receta:** Los profesionales podrán visualizar todas las recetas registradas en el sistema.

3.4.4 Gestión de rutinas

Otra funcionalidad principal de la aplicación es la asignación de ejercicios físicos a cada paciente. Al igual que pasa con las comidas, asignar ejercicios manualmente a diario se hace una tarea muy tediosa. Por lo tanto, se ha implementado la creación de rutinas predefinidas por los profesionales para ser importados a los pacientes cuado se quiera asignar dicho ejercicio. El sistema debe permitir gestionar todas las rutinas. Este escenario está formados pos los siguientes CU:

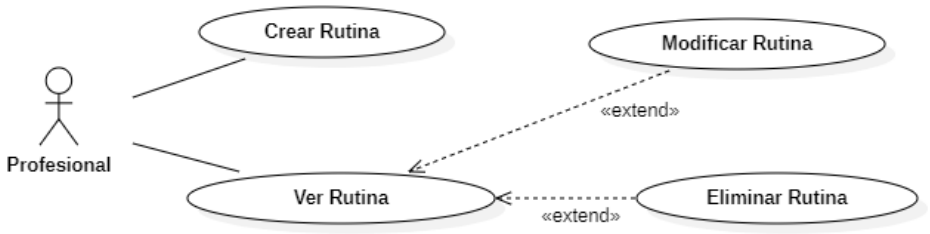

Figura 3.5: Diagrama de CU de gestión de rutinas

- **Crear rutina:** Los profesionales podrán crear nuevas rutinas.
- **Modificar rutina:** Todas las rutinas creadas podrán ser modificadas por los profesionales.
- **Eliminar rutina:** Los profesionales podrán eliminar cualquier rutina.
- **Ver rutina:** Los profesionales podrán visualizar todas las rutinas registradas en el sistema.

3.4.5 Gestión de dietas

Como ya hemos mencionado anteriormente, semanalmente se tiene que asignar bastantes platos de comida a cada paciente. Aún con la existencia de las recetas, esto es una tarea lenta.

Con el fin de simplificar el proceso, se ha implementado la creación de dietas semanales prescritas por los profesionales. Estas dietas consisten en un conjunto de platos planificados para una semana completa. De esta manera, los profesionales tienen la opción de crear un menú que abarque una semana entera, y luego importarlo en los menús individuales de los pacientes para una semana específica. De esta manera, se facilita la asignación de los platos adecuados a cada paciente a lo largo de la semana.

Algo a remarcar es que una vez se ha importado una dieta semanal es posible editar comidas puntuales dentro de la dieta de forma personalizada según los gustos y/o alergias del paciente.

Los CU que forman la gestión de estas diestas semanales son:

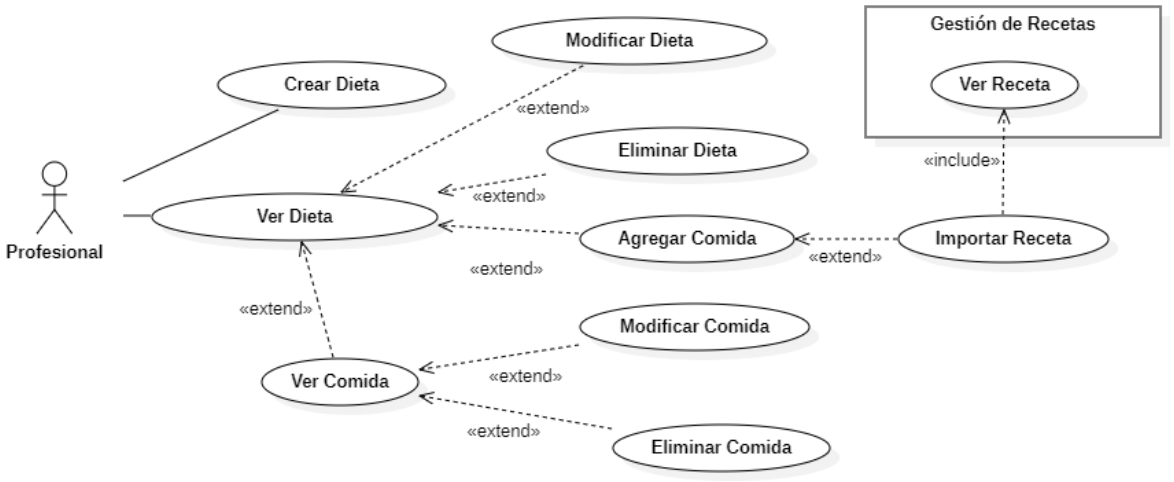

Figura 3.6: Diagrama de CU de gestión de dietas

- **Crear dieta:** Los profesionales podrán crear nuevas dietas.
- **Modificar dieta:** Todas las dietas creadas podrán ser modificadas por los profesionales.
- **Eliminar dieta:** Los profesionales podrán eliminar cualquier dieta.
- **Ver dieta:** Los profesionales podrán visualizar todas las dietas registradas en el sistema.
- **Agregar comida:** Los profesionales podrán agregar un plato de comida en una determinada dieta. En esta parte, se le permite al profesional poder importar ya una receta prescrita, en vez de crear un nuevo plato manualmente.

- **Ver comida:** Los profesionales podrán visualizar todos los platos de comida que forman una determinada dieta, y si se desea, modificarlos o eliminarlos.

3.4.6 Gestión de Comidas

Como ya se he mencionado en apartados anteriores, una de las principales funcionalidades que pretende esta aplicación es asignar un conjunto de platos de comida a los distintos pacientes. Es por ello que el sistema permite a los profesionales poder gestionar esto. Los CU son:

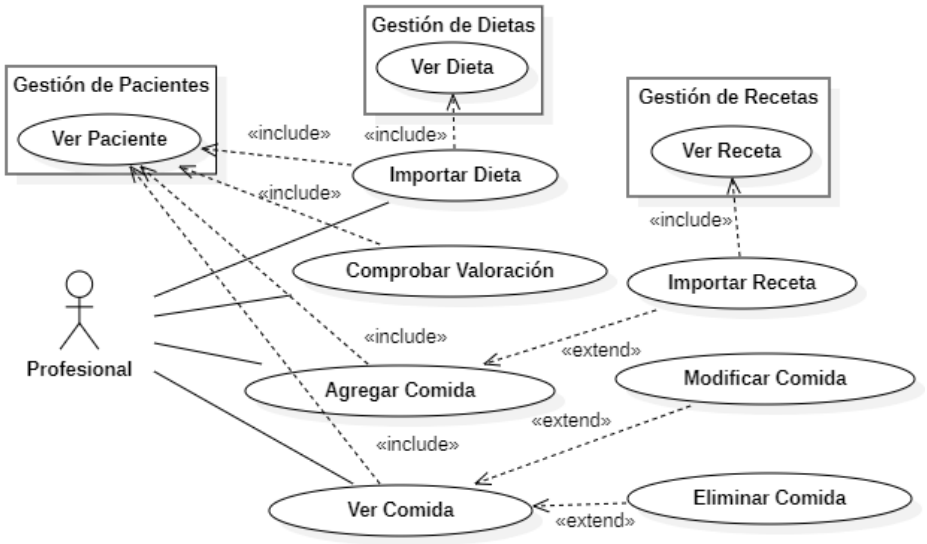

Figura 3.7: Diagrama de CU de gestión de comidas

- **Agregar comida:** Los profesionales podrán crear un plato de comida para un paciente y día concreto. Este también da la posibilidad de importar una receta ya prescrita.
- **Modificar comida:** Los profesionales podrán modificar todas las comidas que han sido asignadas a un determinado paciente.
- **Eliminar comida:** Los profesionales podrán eliminar las comidas asignadas a un determinado paciente.
- **Ver comida:** Los profesionales podrán visualizar todas las comidas que han sido asignadas a un determinado paciente.

- **Importar dieta:** Los profesionales podrán importar una dieta semanal ya prescrita por los profesionales. De esta forma se agregará el conjunto de platos de dicha dieta a la semana seleccionada. Esta importación no eliminará los platos que estaban asignados en dicha semana.
- **Comprobar valoración:** Los profesionales podrán visualizar las valoraciones que aportan los pacientes en cada comida que se les ha asignado, en caso de que este tenga valoración.

3.4.7 Gestión de Ejercicios

Al igual que las comidas, la asignación de ejercios físicos a los pacientes forma una de las principales funcionalidades de la aplicación. El sistema permitirá a los profesionales poder gestionar los ejercicios que son asignados a cada paciente. Los principales CU son:

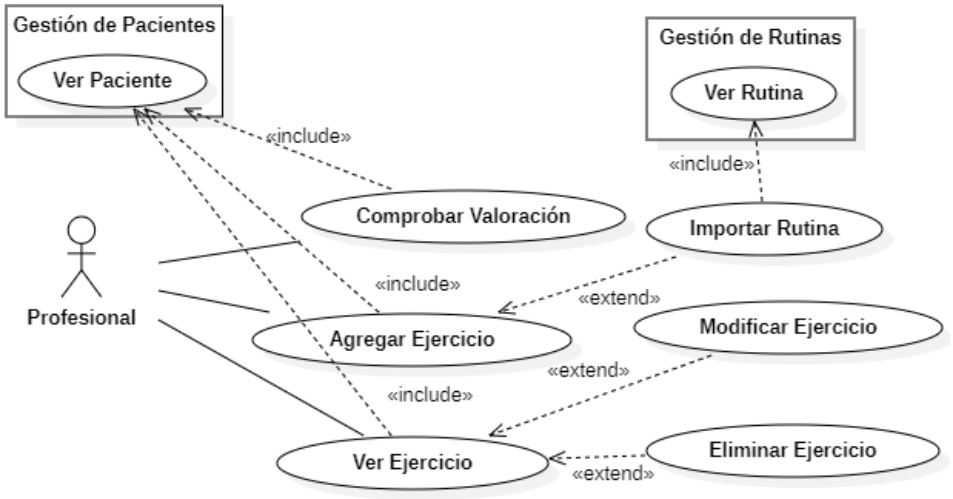

Figura 3.8: Diagrama de CU de gestión de ejercicios

- **Agregar ejercicio:** Los profesionales podrán crear rutinas de ejercicios para un paciente y día concreto. Este también da la posibilidad de importar una rutina ya prescrita.
- **Modificar ejercicio:** Los profesionales podrán modificar todos los ejercicios que han sido asignados a un determinado paciente.
- **Eliminar ejercicio:** Los profesionales podrán eliminar los ejercicios asignados a un determinado paciente.

- **Ver ejercicio:** Los profesionales podrán visualizar todos los ejercicios que han sido asignados a un determinado paciente.
- **Comprobar valoración:** Los profesionales podrán visualizar las valoraciones que aportan los pacientes en cada ejercicio que se les ha asignado, en caso de que este tenga valoración.

3.4.8 Gestión de Consultas

Otra de las principales funcionalidades es la monitorización y registro de los parámetros fisiológicos de cada paciente. Estos son registrados como consultas. A continuación se remarcan algunos de los CU relacionados con la gestión de consultas:

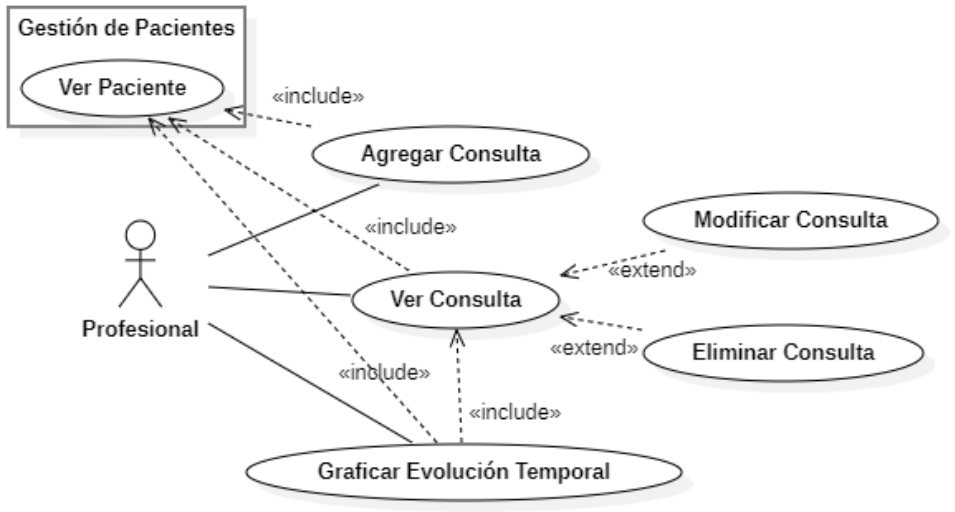

Figura 3.9: Diagrama de CU de gestión de consultas

- **Agregar consulta:** Los profesionales podrán registrar los parámetros fisiólogicos de cada paciente de un determinado día. Estos parámetros son: masa, índice de masa corporal, perímetro abdominal, tensión arterial, triglicéridos séricos, colesterol bueno (HDL), colesterol malo (LDL), hemoglobina glicosilada y glucosa en plasma.
- **Modificar consulta:** Los profesionales podrán modificar todos las consultas de cada paciente.
- **Eliminar consulta:** Los profesionales podrán eliminar las consultas de cada paciente.

- **Ver consulta:** Los profesionales podrán visualizar las consultas de cada paciente.
- **Graficar evolución temporal:** Los profesionales podrán visualizar la evolución temporal de cada parámetro fisiológico registrado en las consultas en un determinado rango de tiempo. Este rango es modificable por el profesional.

3.4.9 Gestión de Autenticación

Una vez registrado los profesionales en el sistema, estos deben de tener la capacidad de poder acceder a ella. Es por ello que la aplicación debe permitir gestionar el tema de la autenticación, así como poder gestionar las claves de acceso. A continuación, se muestra los distintos CU de este escenario:

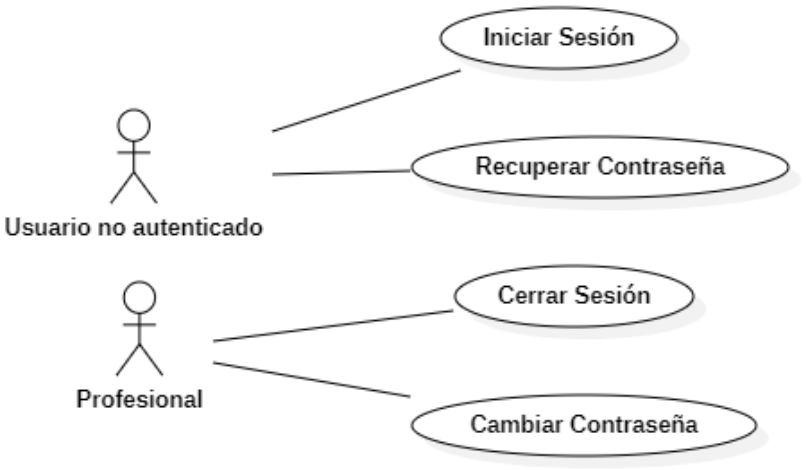

Figura 3.10: Diagrama de CU de gestión de autenticación

- **Iniciar sesión:** El sistema permitirá iniciar sesión a todos los usuarios a través de su email y contraseña. Este comprobará que el usuario existe, que es un profesional y que la contraseña coincide. En caso contrario, se mostrará un mensaje indicándole el motivo de por qué no tiene acceso.
- **Recuperar contraseña:** En caso de que a algún profesional se le olvide la contraseña, el sistema permitirá recuperarla. Este debe indicar el email de su cuenta, el sistema comprobará que existe el usuario y enviará un enlace a dicho email. Ese enlace le redirigirá a una página donde podrá introducir la nueva contraseña para acceder al sistema.

- **Cerrar Sesión:** Los profesionales podrán cerrar la sesión iniciada. Una vez realizado esta acción, le redirigirá a la página de login.
- **Cambiar contraseña:** Los profesionales podrán cambiar su contraseña. Para ello deben introducir la actual contraseña e introducir la nueva contraseña. El sistema validará si la contraseña cumple los requisitos y si la contraseña actual introducida coincide con la real.

3.5 Modelo de datos

Con los requisitos y CU que hemos identificado anteriormente podemos sacar la estructura con la que se guardará los datos en la BBDD.

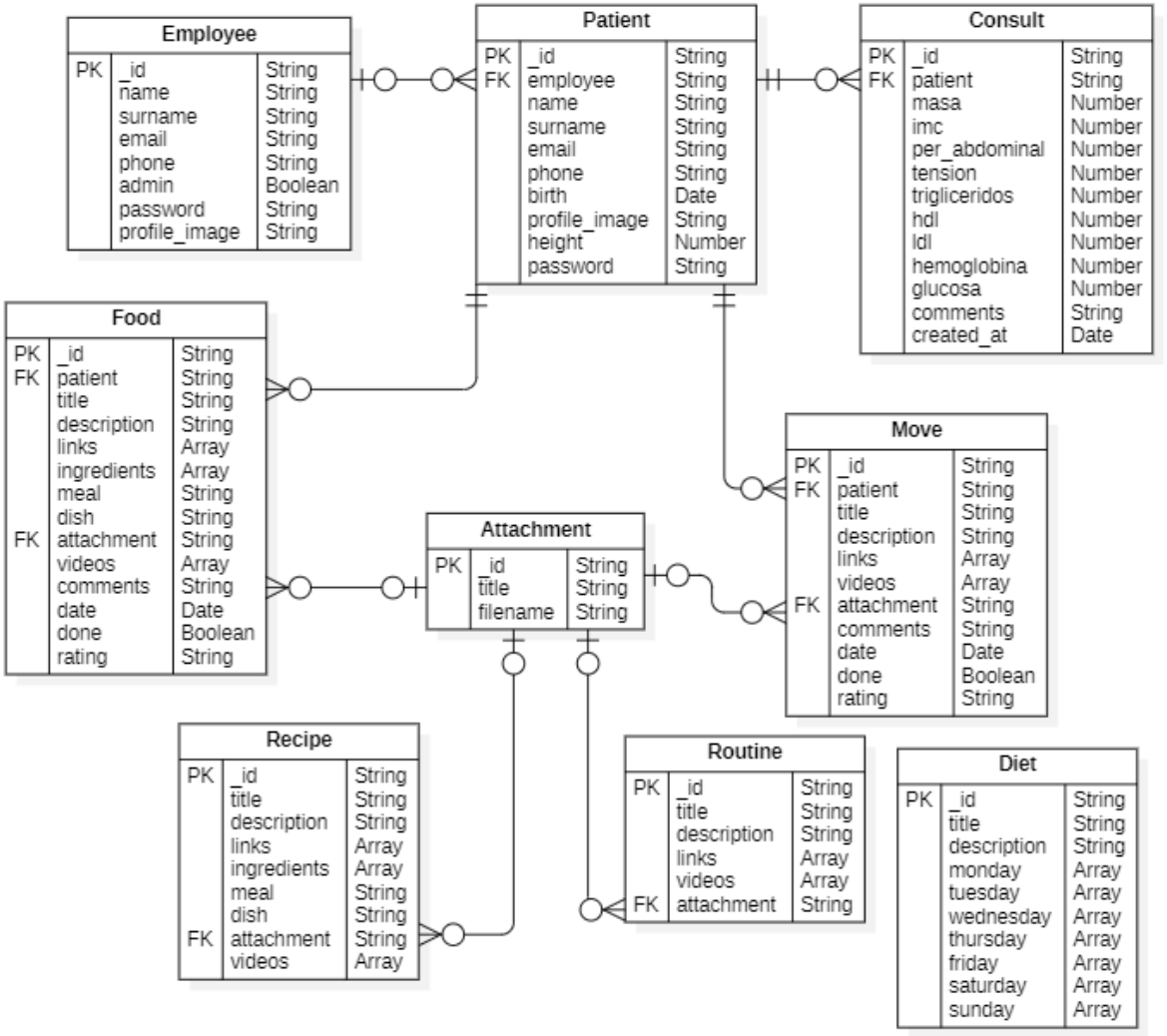

Figura 3.11: Modelo de datos

El diagrama queda reducido a las siguientes entidades:

- **Employee:** En esta entidad se almacena todos los datos referentes a un profesional.
- **Patient:** En esta entidad se almacena todos los datos referentes a cada paciente. Este tiene una relación uno a mucho con la entidad *'Employee'*. Cada profesional puede tener varios pacientes, pero cada paciente solo puede tener asignado un profesional.

- **Recipe:** Almacenan los datos referentes a las recetas que son creados por los profesionales para una próxima importación. Estos pueden tener un archivo adjunto asociado.
- **Routine:** Esta entidad guarda la información relacionada con las rutinas que son creadas por los profesionales para una próxima importación. Estos pueden tener un archivo adjunto asociado.
- **Diet:** En esta entidad se registran los datos acerca de las dietas semanales creadas por los profesionales para una próxima importación en una semana de un paciente en concreto.
- **Food:** Almacena los datos referentes a las comidas que se le asignan a un paciente. Este posee una relación uno a mucho con la entidad *'Patient'*. Los pacientes pueden tener asignadas varias comidas, pero a cada comida se le asigna un solo paciente. Estos pueden tener un archivo adjunto asociado.
- **Move:** Esta entidad almacena los datos referentes a los ejercicios que se le asignan a un paciente. Al igual que las comidas, estos solo se le asigna un paciente, pero los pacientes pueden tener asignados varios ejercicios. Relación uno a muchos con la entidad *'Patient'*. Estos pueden tener un archivo adjunto asociado.
- **Consult:** En esta entidad se almacenan los parámetros fisiológicos medidos de un paciente en un determinado día. Esta consulta está asignada a un solo paciente, sin embargo, los pacientes pueden tener asignadas varias consultas, por lo que la entidad *'Consult'* tiene una relación uno a muchos con la entidad *'Patient'*.
- **Attachment:** Los registros en esta entidad contienen información sobre los archivos adjuntos que son asignados a las entidades: *'Food'*, *'Move'*, *'Recipe'* y *'Routine'*. Este tiene una relación uno a muchos con las entidades mencionadas porque cada archivo adjunto puede ser asignado en varios elementos, pero cada elemento solo puede tener asociado un archivo adjunto.

3.6 Diccionario de datos

A continuación, presentamos una descripción más detallada de cada una de las entidades en la base de datos.

3.6.1 Employee

Atributo	Descripción	Dominio	Único	Null	Notas
_id	Identificador único del profesional	String	Sí	No	Clave primaria
name	Nombre del profesional	String	No	No	
surname	Apellidos del profesional	String	No	Sí	
email	Email del profesional	String	Sí	No	
phone	Teléfono del profesional	String	No	Sí	
admin	Es o no el profesional administrador	Boolean	No	No	
profile_image	Foto de perfil del profesional	String	No	Sí	
password	Contraseña del profesional	String	No	No	

Tabla 3.4: Tabla diccionario de datos de 'Employee'

El profesional se deberá identificar en la aplicación web con su email y contraseña, por lo tanto, estos campos son obligatorios y el email debe ser único.

El atributo *'profile_image'*, en caso de tener una foto de perfil, guarda el nombre de dicha imagen.

3.6.2 Patient

Atributo	Descripción	Dominio	Único	Null	Notas
_id	Identificador único del paciente	String	Sí	No	Clave primaria
employee	Identificador del profesional	String	No	No	Clave foránea
name	Nombre del paciente	String	No	No	
surname	Apellidos del paciente	String	No	Sí	
email	Email del paciente	String	No	Sí	
phone	Teléfono del paciente	String	Sí	No	
birth	Fecha de nacimiento del paciente	Date	No	Sí	
profile_image	Foto de perfil del paciente	String	No	Sí	
height	Altura del paciente	Number	No	Sí	
password	Contraseña del paciente	String	No	No	

Tabla 3.6: Tabla diccionario de datos de 'Patient'

Los pacientes, a diferencia de los profesionales, se identificarán en la aplicación móvil con su número de teléfono y contraseña. Por lo tanto, es necesario que estos campos sean obligatorios y, en el caso del teléfono, único.

La altura, si bien no se considera estrictamente un dato personal del paciente, sino más bien un parámetro fisiológico, se ha decidido incluirlo en esta entidad debido a que normalmente no experimenta cambios.

Al igual que los profesionales, el atributo *'profile_image'* guarda el nombre de la imagen de su foto de perfil, en caso de que dicho paciente tenga.

3.6.3 Recipe

Atributo	Descripción	Dominio	Único	Null	Notas
_id	Identificador único de la receta	String	Sí	No	Clave primaria
title	Título de la receta	String	No	No	
description	Descripción de la receta	String	No	Sí	
links	Conjunto de enlaces	Array	No	No	
ingredients	Conjunto de ingredientes que forman la receta	Array	No	No	
meal	Desayuno, almuerzo, merienda o cena	String	No	No	
dish	Principal, primer plato, segundo plato o postre	String	No	No	
attachment	Identificador del archivo adjunto	String	No	Sí	Clave foránea
videos	Conjunto de enlaces de vídeos de youtube	Array	No	Sí	

Tabla 3.8: Tabla diccionario de datos de 'Recipe'

Estos almacenan comidas prescritas para posteriormente ser importado en los menús individuales de cada paciente.

Los atributos *'meal'* y *'dish'* indican que tipo de plato es dicha receta. Estos pueden ser: plato principal del desayuno, primer plato del almuerzo, segundo plato del almuerzo, postre del almuerzo, plato principal de la merienda, plato principal de la cena y postre de la cena.

El atributo *'ingredients'* es un array donde almacena un conjunto de objetos. Cada uno de estos objetos representa un ingrediente de la receta, cuya estructura es la siguiente:

Atributo	Descripción	Dominio	Único	Null	Notas
name	Nombre del ingrediente	String	No	No	

Atributo	Descripción	Dominio	Único	Null	Notas
quantity	Cantidad del ingrediente a utilizar	Number	No	Sí	
unit	Unidad de medida a utilizar	String	No	Sí	
isChecked	Indica si el paciente ya tiene o no dicho ingrediente	Boolean	No	No	

Tabla 3.10: Estructura de los ingredientes

El atributo *'isChecked'* es para una funcionalidad presente en la aplicación móvil, no para la aplicación web. Este permite a los pacientes hacer un control sobre la lista de la compra.

3.6.4 Routine

Atributo	Descripción	Dominio	Único	Null	Notas
_id	Identificador único de la rutina	String	Sí	No	Clave primaria
title	Título de la rutina	String	No	No	
description	Descripción de la rutina	String	No	Sí	
links	Conjunto de enlaces	Array	No	No	
videos	Conjunto de enlaces de vídeos de youtube	Array	No	Sí	
attachment	Identificador del archivo adjunto	String	No	Sí	Clave foránea

Tabla 3.12: Tabla diccionario de datos de 'Routine'

3.6.5 Diet

Atributo	Descripción	Dominio	Único	Null	Notas
_id	Identificador único de la dieta	String	Sí	No	Clave primaria
title	Título de la dieta	String	No	No	
description	Descripción de la dieta	String	No	Sí	
monday	Conjunto de comidas asignadas al lunes	Array	No	No	
tuesday	Conjunto de comidas asignadas al martes	Array	No	No	
wednesday	Conjunto de comidas asignadas al miércoles	Array	No	No	
thursday	Conjunto de comidas asignadas al jueves	Array	No	No	
friday	Conjunto de comidas asignadas al viernes	Array	No	No	
saturday	Conjunto de comidas asignadas al sábado	Array	No	No	
sunday	Conjunto de comidas asignadas al domingo	Array	No	No	

Tabla 3.14: Tabla diccionario de datos de 'Diet'

Las dietas, como ya vimos con anterioridad, sirven para importar un conjunto de platos en una semana concreta de un determinado paciente. Estos platos son guardados en los atributos *'monday'*, *'tuesday'*, *'wednesday'*, etc., en función del día de la semana que se le asigna dicho plato de comida.

Todos estos atributos guardan un conjunto de objetos con la misma estructura que la entidad *'Recipe'* que vimos en el apartado 3.6.3.

3.6.6 Food

Atributo	Descripción	Dominio	Único	Null	Notas
_id	Identificador único de la comida	String	Sí	No	Clave primaria
patient	Identificador del paciente	String	No	No	Clave foránea
title	Título de la comida	String	No	No	
description	Descripción de la comida	String	No	Sí	
links	Conjunto de enlaces	Array	No	No	
ingredients	Conjunto de ingredientes que forman la comida	Array	No	No	
meal	Desayuno, almuerzo, merienda o cena	String	No	No	
dish	Principal, primer plato, segundo plato o postre	String	No	No	
attachment	Identificador del archivo adjunto	String	No	Sí	Clave foránea
videos	Conjunto de enlaces de vídeos de youtube	Array	No	Sí	
comments	Comentarios relacionados con la comida	String	No	Sí	
date	Fecha que se le asigna a la comida	Date	No	No	
done	Indica si el paciente ha realizado o no dicha comida	Boolean	No	No	
rating	Valoración del paciente: glad, normal, sad	String	No	Sí	

Tabla 3.16: Tabla diccionario de datos de 'Food'

Esta entidad almacena los datos referentes a los platos de comida asignados a cada paciente. Como podemos observar, este tiene atributos que ya hemos visto en la entidad 'Recipe', apartado 3.6.3. Esto es lo que nos permite importar una receta en una comida.

Aparte, tenemos atributos que hace referencia al paciente que le es asignado (*'patient'*), la fecha de asignación (*'date'*), unos comentarios (*'comments'*), si ha realizado dicha comida el paciente o no (*'done'*) y una valoración del paciente (*'rating'*).

3.6.7 Move

Atributo	Descripción	Dominio	Único	Null	Notas
_id	Identificador único del ejercicio	String	Sí	No	Clave primaria
patient	Identificador del paciente	String	No	No	Clave foránea
title	Título del ejercicio	String	No	No	
description	Descripción del ejercicio	String	No	Sí	
links	Conjunto de enlaces	Array	No	No	
videos	Conjunto de enlaces de vídeos de youtube	Array	No	Sí	
attachment	Identificador del archivo adjunto	String	No	Sí	Clave foránea
comments	Comentarios relacionados con el ejercicio	String	No	Sí	
date	Fecha que se le asigna a el ejercicio	Date	No	No	
done	Indica si el paciente ha realizado o no dicho ejercicio	Boolean	No	No	
rating	Valoración del paciente: glad, normal, sad	String	No	Sí	

Tabla 3.18: Tabla diccionario de datos de 'Move'

Esta entidad almacena los datos referentes a los ejercicios asignados a cada paciente. Como podemos observar, este tiene atributos que ya hemos visto en la entidad 'Routine', apartado 3.6.4. Esto es lo que nos permite importar una rutina en un ejercicio.

Aparte, tenemos atributos que hace referencia al paciente que le es asignado (*'patient'*), la fecha de asignación (*'date'*), unos comentarios (*'comments'*), si ha realizado dicho ejercicio el paciente o no (*'done'*) y una valoración del paciente (*'rating'*).

3.6.8 Consult

Atributo	Descripción	Dominio	Único	Null	Notas
_id	Identificador único de la consulta	String	Sí	No	Clave primaria
patient	Identificador del paciente	String	No	No	Clave foránea
masa	Masa corporal	Number	No	Sí	
imc	Índice de masa corporal	Number	No	Sí	
per_abdominal	Perímetro abdominal	Number	No	Sí	
tension	Tensión arterial	Number	No	Sí	
trigliceridos	Triglicéridos séricos	Number	No	Sí	
hdl	Colesterol bueno	Number	No	Sí	
ldl	Colesterol malo	Number	No	Sí	
hemoglobina	Hemoglobina glicosilada	Number	No	Sí	
glucosa	Glucosa en plasma	Number	No	Sí	

Atributo	Descripción	Dominio	Único	Null	Notas
comments	Comentarios con respecto a la consulta	String	No	Sí	
created_at	Fecha de la consulta	Date	No	No	

Tabla 3.20: Tabla diccionario de datos de 'Consult'

Estos almacenan los parámetros fisiológicos de un paciente, medidos en un determinado día. Dichos parámetros no son obligatorios, por lo que la aplicación permite a los profesionales controlar solo los parámetros que a él le interesa.

Este tiene asignado un paciente (*'patient'*), una fecha (*'created_at'*) y unos comentarios (*'comments'*).

3.6.9 Attachment

Atributo	Descripción	Dominio	Único	Null	Notas
_id	Identificador único del archivo adjunto	String	Sí	No	Clave primaria
title	Título del archivo adjunto	String	Sí	No	
filename	Nombre del archivo almacenado	String	Sí	No	

Tabla 3.22: Tabla diccionario de datos de 'Attachment'

Como ya hemos visto en los anteriores apartados, varias entidades tienen asociado una archivo adjunto (*'attachment'*). Esto les permite asociar un pdf a cada comida y ejercicio y que el paciente pueda entender mejor las instrucciones del mismo.

Todos estos archivos son registrados en esta entidad.

3.7 Arquitectura del sistema

En esta sección, se describe la arquitectura del sistema implementado en el proyecto. Entender la arquitectura que se está utilizando es fundamental para comprender cómo opera el conjunto de componentes que conforman la aplicación.

El sistema se compone de tres elementos principales: una aplicación frontend, una API y una base de datos.

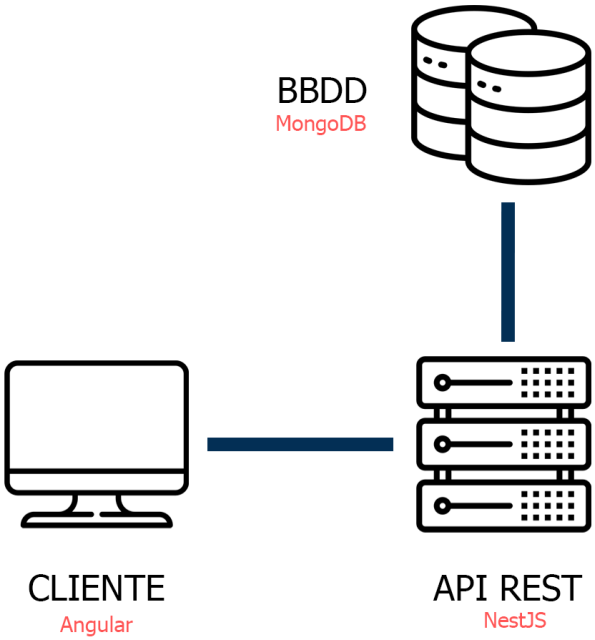

Figura 3.12: Arquitectura del sistema

La aplicación frontend es la interfaz con la que interactúan los usuarios con el sistema. A través de esta interfaz, los usuarios pueden acceder a diferentes funcinalidades, visualizar información relevante y realizar accciones específicas. La aplicación frontend se encarga de permitir la interacción del usuario con la API.

La API funciona como si fuese un puente entre el frontend y la base de datos. Proporciona un conjunto de peticiones HTTP que permite la comunicación entre la aplicación frontend y la base de datos. La API se encarga de recibir las solicitudes por parte de la aplicación frontend, procesarlas y enviar las respuestas correspondientes. Además, este realiza operaciones de autenticación para garantizar la seguridad e integridad de los datos.

La base de datos se encarga de almacenar la información necesaria para el funcionamiento del sistema.

CAPÍTULO 4

DESARROLLO

4.1 Desarrollo de la API 45
4.1.1 Estructura de directorios 45
4.1.2 Peticiones HTTP 48
4.2 Desarrollo de la aplicación web 55
4.2.1 Estructura de directorios 55
4.2.2 Navegación 58

En este capítulo exploraremos aspectos importantes dentro de la fase del desarrollo. Se presentará las estructuras de directorios que se han utilizado, tanto en la API como en la aplicación, las peticiones HTTP que nos aporta la API y el mapa de navegación de la aplicación.

4.1 Desarrollo de la API

4.1.1 Estructura de directorios

La organización de los archivos y carpetas desempeña un papel fundamental en la gestión y mantenimiento del proyecto. Una estructura bien definida facilita la navegación a través de los distintos componenetes.

En este apartado, se presenta la estructura de directorios adoptados en la API.

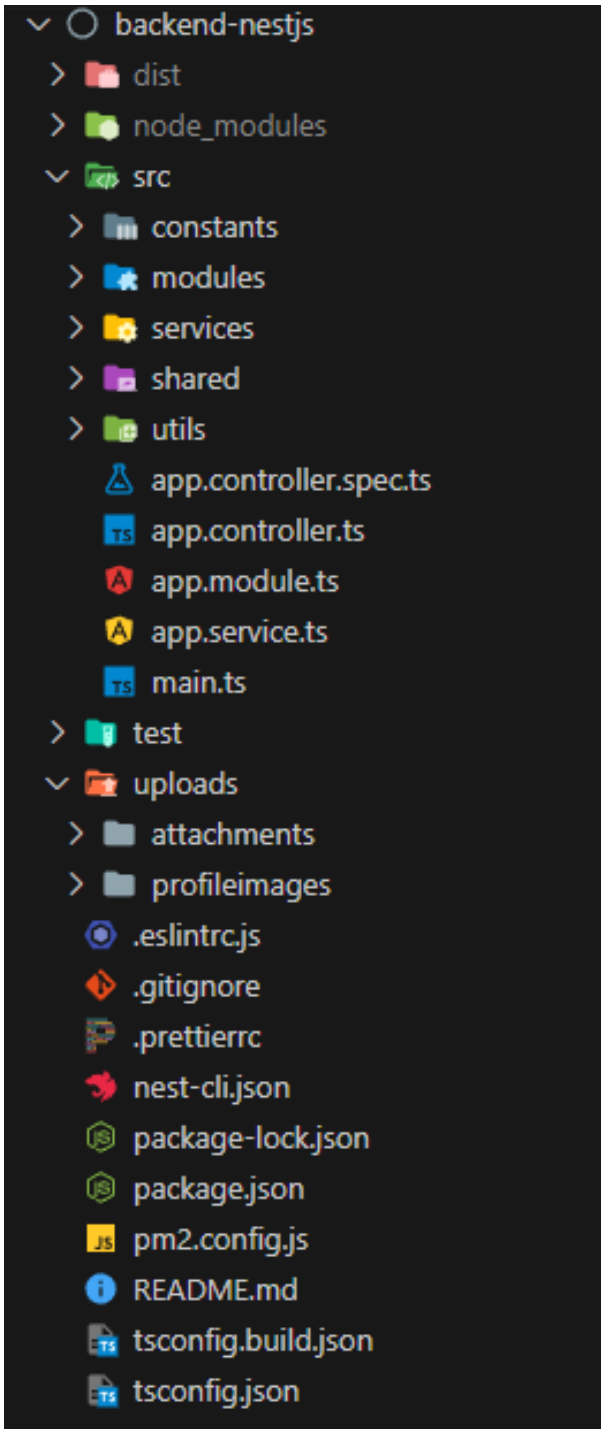

Figura 4.1: Árbol de directorios de la API

Como podemos ver en la figura 4.1, en la carpeta raíz destacan las siguientes carpetas:

- **src:** En ella se encuentra toda el código principal.
- **uploads:** En esta se almacenan las distintas fotos de perfil y archivos adjuntos subidos por los pacientes y profesionales.

El resto de directorios y archivos de la carpeta raíz se encargan de tareas como la configuración de typescript, del control de versiones, del testeo u otras tareas que no son de gran relevancia para nosotros.

Dentro de la carpeta **src** encontramos el módulo principal y el main del programa. También podemos encontrar las siguientes carpetas:

- **constants:** Contiene distintos archivos en la cual registran un conjunto de constantes que se irán utilizando a lo largo del código.
- **modules:** Contiene todos los módulos de la API. Más adelante veremos cuáles son.
- **services:** Contiene los servicios que son utilizados en más de un módulo.
- **shared:** Contiene el conjunto de guardianes, enumerados y dtos globales.
- **utils:** Contiene funciones de utilidad que pueden ser utilizados en diferentes partes de la aplicación.

En la API podemos encontrar los siguientes módulos:

Como podemos observar en la imagen 4.2, existe un módulo dedicado a cada entidad de la base de datos. Aparte de estos, tenemos otros que se encargan de tareas como la autentificación (*'auth'*), el envío de emails (*'mail'*) o la gestión de archivos (*'files'*).

Cada uno de estos módulos gestiona su controlador, donde aporta las distintas peticiones HTTP, y sus propios servicios, donde gestiona toda la lógica principal de cada petición que aporta el controlador. Así mismo, en estos se declaran los esquemáticos de la bbdd y los dtos (Data Transfer Object). Podemos verlo más claro con el siguiente ejemplo:

En la imagen 4.3 podemos ver los directorios del módulo de *'patients'*, donde tiene su modulo, controlador y servicio, junto a sus dtos y esquemático. El resto de módulos presenta una estructura similar.

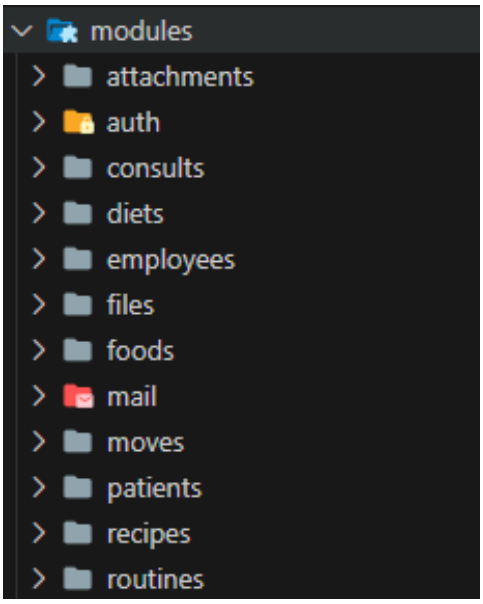

Figura 4.2: Módulos de la API

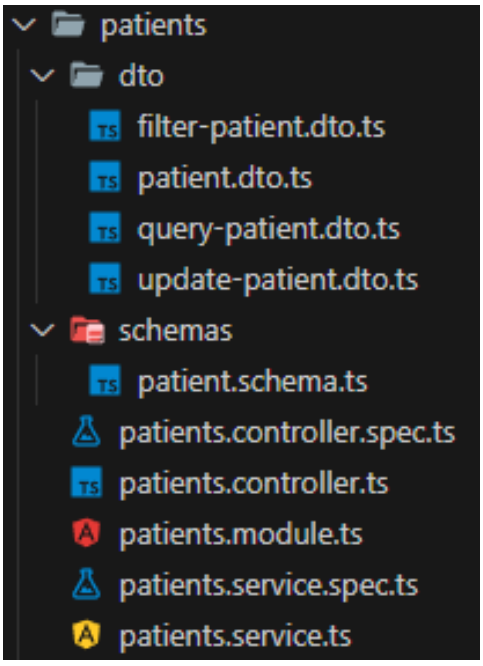

Figura 4.3: Módulo Patients

4.1.2 Peticiones HTTP

En esta sección, presentaremos una lista de distintas peticiones que se pueden realizar a la API desarrollada. Estas peticiones nos permitirán interactuar con los diversos datos almacenados en la base de datos y aprovechar al máximo su funcionalidad.

A través de estas peticiones, podremos acceder y manipular los datos almacenados, obteniendo la información relevante para nuestro proyecto y realizar acciones específicas según nuestras necesidades.

Se propocionará información detallada sobre cada petición, incluyendo la ruta de acceso, el método HTTP utilizado y una breve descripción de su propósito y funcionalidad. Estas peticiones se encuentran organizadas en función de su módulo correspondiente.

Employees

Método	Ruta	Descripción
/api/employees/		
GET	{id}	Obtiene un profesional específico
POST	lookUp	Obtiene un profesional que cumple con ciertos criterios de filtro
POST	filter	Obtiene un conjunto de profesionales que cumplen con ciertos criterios de filtro
POST	create	Crea un nuevo profesional
PATCH	update/{id}	Actualiza un profesional específico
DELETE	remove/{id}	Elimina un profesional específico
POST	upload/{id}	Asigna una foto de perfil a un profesional específico
DELETE	remove-profile-image/{id}	Elimina la foto de perfil de un profesional específico
POST	change-password/{id}	Cambia la contraseña de un profesional específico
POST	fogotPassword/{email}	Crea un enlace de recuperación de contraseña e envía un correo
POST	recoverPassword/	Cambia la contraseña en un proceso de recuperación de contraseña

Tabla 4.2: Peticiones HTTP del módulo 'Employees'

Patients

Método	Ruta	Descripción
/api/patients/		
GET	{id}	Obtiene un paciente específico
POST	lookUp	Obtiene un paciente que cumple con ciertos criterios de filtro
POST	filter	Obtiene un conjunto de pacientes que cumplen con ciertos criterios de filtro
POST	create	Crea un nuevo paciente
PATCH	update/{id}	Actualiza un paciente específico
DELETE	remove/{id}	Elimina un paciente específico
POST	upload/{id}	Asigna una foto de perfil a un paciente específico
DELETE	remove-profile-image/{id}	Elimina la foto de perfil de un paciente específico
GET	randomPassword	Genera una contraseña aleatoria

Tabla 4.4: Peticiones HTTP del módulo 'Patients'

Recipes

Método	Ruta	Descripción
/api/recipes/		
GET	{id}	Obtiene una receta específica

Método	Ruta	Descripción
POST	lookUp	Obtiene una receta que cumple con ciertos criterios de filtro
POST	filter	Obtiene un conjunto de recetas que cumplen con ciertos criterios de filtro
POST	create	Crea una nueva receta
PATCH	update/{id}	Actualiza una receta específica
DELETE	remove/{id}	Elimina una receta específica

Tabla 4.6: Peticiones HTTP del módulo 'Recipes'

Routines

Método	Ruta	Descripción
/api/routines/		
GET	{id}	Obtiene una rutina específica
POST	lookUp	Obtiene una rutina que cumple con ciertos criterios de filtro
POST	filter	Obtiene un conjunto de rutinas que cumplen con ciertos criterios de filtro
POST	create	Crea una nueva rutina
PATCH	update/{id}	Actualiza una rutina específica
DELETE	remove/{id}	Elimina una rutina específica

Tabla 4.8: Peticiones HTTP del módulo 'Routines'

Foods

Método	Ruta	Descripción
/api/foods/		
GET	{id}	Obtiene una comida específica
GET	findAll	Obtiene todas las comidas
POST	find	Obtiene un conjunto de comidas que cumplen con ciertos criterios de filtro
POST	create	Crea una nueva comida
PATCH	update/{id}	Actualiza una comida específica
DELETE	remove/{id}	Elimina una comida específica
POST	findByPatient/{id}	Obtiene un conjunto de comidas de un paciente en específico, delimitado por un rango de fechas determinado
POST	findIngredientes/{id}	Obtiene un conjunto de ingredientes de todas las comidas de un paciente en específico, cuyas fechas están delimitados por un rango determinado
GET	checkIngrediente/{id}/{name}	Marca o desmarca un ingrediente de una comida en específico
GET	importDiet/{dietId}/{patientId}/{date}	Importa una dieta a un paciente determinado en una semana concreta

Tabla 4.10: Peticiones HTTP del módulo 'Foods'

Moves

Método	Ruta	Descripción
/api/moves/		

Método	Ruta	Descripción
GET	{id}	Obtiene un ejercicio específica
GET	findAll	Obtiene todos los ejercicios
POST	find	Obtiene un conjunto de ejercicios que cumplen con ciertos criterios de filtro
POST	create	Crea un nuevo ejercicio
PATCH	update/{id}	Actualiza un ejercicio específico
DELETE	remove/{id}	Elimina un ejercicio específico
POST	findByPatient/{id}	Obtiene un conjunto de ejercicios asignados a un paciente específico dentro de un rango de fechas determinado

Tabla 4.12: Peticiones HTTP del módulo 'Moves'

Consults

Método	Ruta	Descripción
/api/consults/		
GET	{id}	Obtiene una consulta específica
POST	lookUp	Obtiene una consulta que cumple con ciertos criterios de filtro
POST	filter	Obtiene un conjunto de consultas que cumplen con ciertos criterios de filtro
POST	create	Crea una nueva consulta
PATCH	update/{id}	Actualiza una consulta específica
DELETE	remove/{id}	Elimina una consulta específica

Método	Ruta	Descripción
POST	getValues/{id}/{key}	Obtiene un conjunto de medidas de un parámetro específico, pertenecientes a un paciente determinado, dentro de un rango de fechas específico

Tabla 4.14: Peticiones HTTP del módulo 'Consults'

Attachments

Método	Ruta	Descripción
/api/attachments/		
GET	findOne/{id}	Obtiene la referencia de un archivo en específico
GET	findAll	Obtiene la referencia de todos los archivos
POST	create	Crea un nuevo archivo
PATCH	update/{id}	Actualiza un archivo específico
DELETE	remove/{id}	Elimina un archivo específico

Tabla 4.16: Peticiones HTTP del módulo 'Attachments'

Auth

Método	Ruta	Descripción
/api/auth/		
POST	loginPatient	Iniciar sesión para un paciente específico

Método	Ruta	Descripción
POST	loginEmployee	Iniciar Sesión para un profesional específico

Tabla 4.18: Peticiones HTTP del módulo 'Auth'

Files

Método	Ruta	Descripción
/api/files/		
GET	profile-image/{filename}	Obtiene la imagen de una foto de perfil
GET	attachment/{filename}	Obtiene el fichero de un archivo adjunto

Tabla 4.20: Peticiones HTTP del módulo 'Files'

4.2 Desarrollo de la aplicación web

4.2.1 Estructura de directorios

Al igual que en el desarrollo de la API, la estructuración de la aplicación frontend es fundamental para la gestión y mantenimiento del proyecto.

En este apartado se presenta la estructura de directorios de la aplicación frontend, detallando los diferentes directorios y su función en la organización de los archivos del proyecto.

Como podemos observar en la figura 4.4, en la carpeta raíz se encuentra la carpeta **src**, esta contiene la parte importante del programa. El resto de ficheros y carpetas se encargan de temas como la configuración de typescript, la configuración de angular, del control de versiones, entre otros.

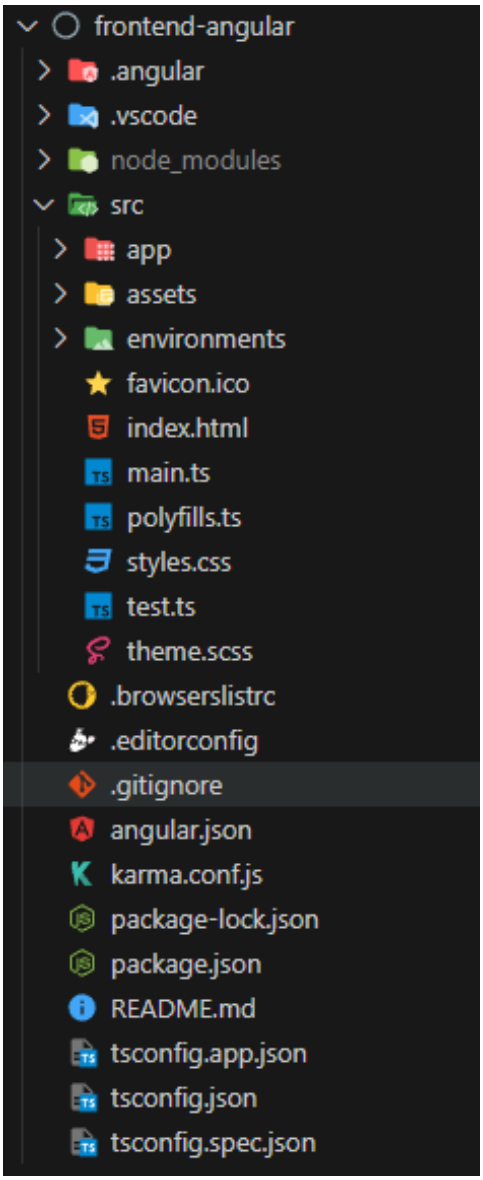

Figura 4.4: Árbol de directorios de la app

Dentro del directorio **src** observamos el main del programa, el index.html y la hoja de estilos styles.css. Aparte de estos, encontramos los siguientes directorios:

- **app:** Esta contine la gran parte del programa, los módulos, servicios, modelos, guardianes, entre otros. Este lo veremos con más detenimiento más adelante.
- **assets:** En este directorio se alamacena los distitos ficheros como imágenes, iconos u hojas de estilos globales.

- **environments:** Aquí se guardan los archivos que almacenan un conjunto de constantes que tiene un valor u otro en función en que entorno estamos ejecutando el programa (desarrollo o producción). Esto nos es de utilidad para indicar la dirección de la API, ya que no es el mismo cuando estamos desarrollandolo que cuando el proyecto está en producción.

Como ya hemos dicho, la carpeta **app** contiene la gran parte del programa. Este está constituido por los siguientes directorios:

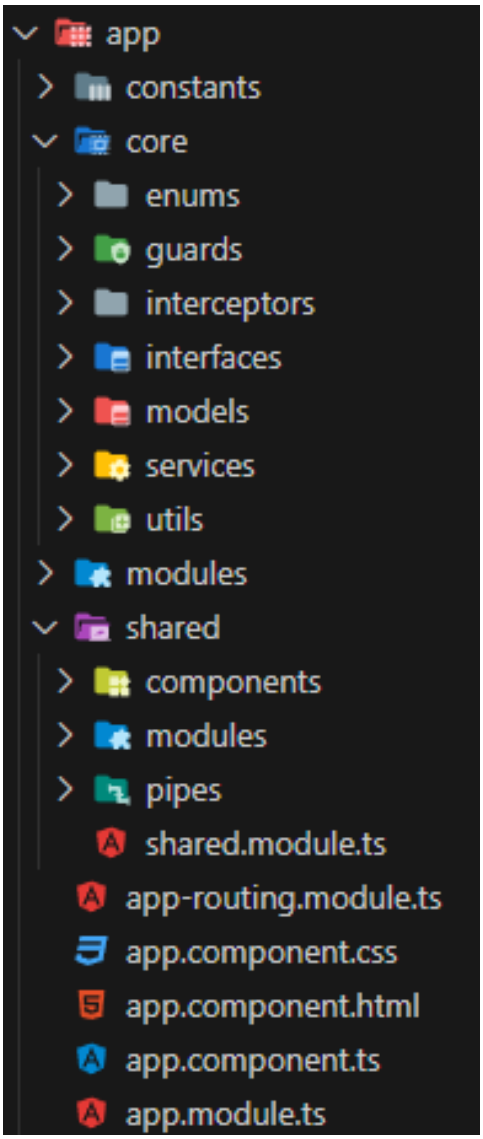

Figura 4.5: Directorio app

- **constants:** Constiene distintos archivos en la cual registran un conjunto de constantes que serán utilizados a lo largo del programa.

- **core:** Este almacenan archivos y componenetes globales. Este contiene directorios como:
  - **enums:** Conjunto de enumerados.
  - **guards:** Guardianes, con el fin de que cada usuario pueda acceder únicamente a las páginas que le corresponde.
  - **interceptors:** Interceptores, para interceptar solicitudes y respuestas HTTP.
  - **interfaces:** Conjunto de interfaces generales.
  - **models:** Conjunto de interfaces que hacen referencia a los modelos de datos.
  - **services:** Conjunto de servicios que interactúan con la API y otros servicios con otras funcionalidades globales, como el manejo del enrutamiento, de mostrar mensajes de error o de éxito, de mostrar una barra de carga, entre otros.
  - **utils:** Archivos que contienen un conjunto de funciones generales.
- **modules:** Este contiene los distintos módulos y páginas del programa.
- **shared:** Este guarda un conjunto de módulos y componentes que son reutilizados en las distintas páginas de la aplicación. También este almacena los distintos pipes, herramientas que nos permiten modificar o tranformar la información de cierta manera.

4.2.2 Navegación

Durante el desarrollo de la aplicación se ha tenido que implementar el enrutamiento de las diferentes páginas que componen la aplicación.

En la figura 4.6 se muestra las pantallas de la aplicación y se puede observar cómo se establece la navegación entre ellas.

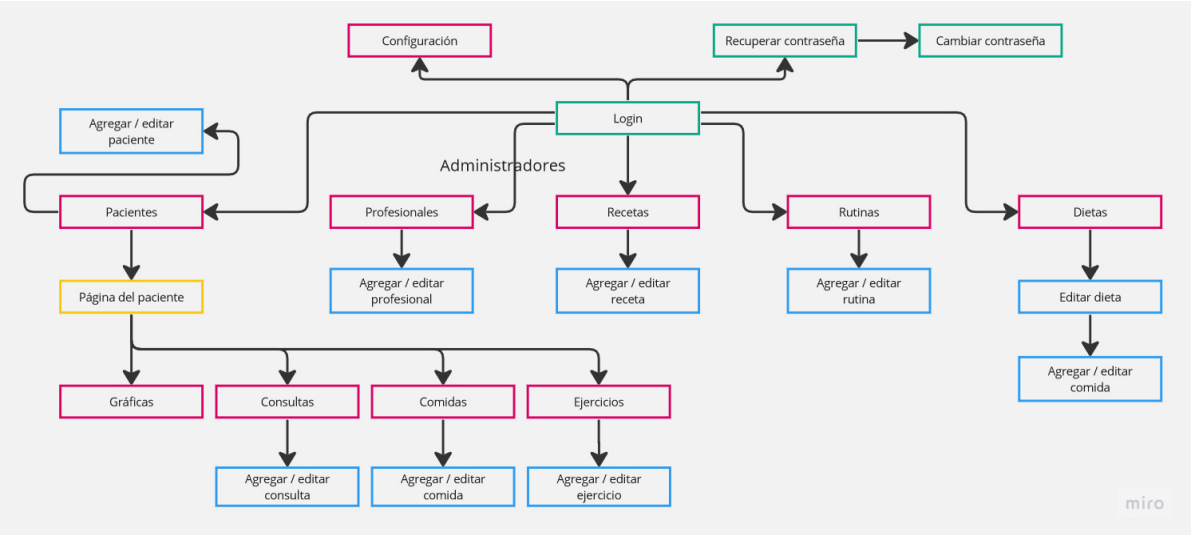

Figura 4.6: Navegación de la aplicación

Carga de páginas

La navegación en Angular se realiza gracias al módulo denominado *RouterModule*. En nuestro proyecto vamos a aprovechar la capacidad de carga de páginas bajo demanda que proporciona este componente.

La carga de páginas bajo demanda consiste en que el módulo que contiene una página se carga únicamente cuando el usuario acceda a ella. En caso de no utilizar esta técnica, el navegador carga todas las pantallas que componen la aplicación nada mas acceder a ella. Es por ello que utilizar una carga bajo demanda reduce notablemente la carga inicial de la aplicación.

Sin embargo, esto implica que cada vez que cambiemos de página va ha haber un pequeño tiempo de carga, que de la otra manera no sucedería ya que todas las páginas son cargadas al inicio. Aun así, no son tiempos muy largos y evitamos que al iniciar la aplicación haya tiempo de carga muy prolongado y se le haga demasido frustado al usuario.

Para implementar la navegación en Angular, cada módulo debe tener un módulo de enrutamiento que indique qué debe cargar cuando se acceda a determinadas rutas.

En la figura 4.7 podemos observar el módulo de enrutamiento principal de la aplicación.

```
const routes: Routes = [
  {path: '', component: HomePageComponent, loadChildren: () => import('@modules/home/home.module').then(x => x.HomeModule), canActivate: [AuthGuard]},
  {path: 'auth', loadChildren: () => import('./modules/auth/auth.module').then(x => x.AuthModule), canActivate: [AutoLoginGuard]},
  {path: '**', redirectTo: ''}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Figura 4.7: Módulo de enrutamiento principal

En el array de rutas llamado *Routes* se define qué módulo cargar para cada ruta. En nuestro caso está dividia en dos módulos:

- **Home:** Donde registra las distintas páginas y módulos principales de la apliación.
- **Auth:** Este guarda páginas relacionadas con el login, recuperación de contraseña, entre otros.

Estos están protegidos mediante unas guardias que nos permite dar acceso al usuario a estas páginas en función si ya tiene una sesión iniciada o no. En caso de que no lo esté, el sistema, independientemente de que URL introdudca, redirigirá al usuario a la página de login. En caso contrario, redirigirá al usuario a la página principal de la aplicación.

En la figura 4.8 podemos observar el guardian que impide el acceso a las páginas principales a los usuarios que no han iniciado sesión.

```
@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {

  constructor (
    private authService: AuthService,
    private routerService: RouterService
  ) {}

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
      const isLogin = this.authService.isLogin(); // Hay una sesión existente?
      if (isLogin) this.authService.setSession(); // Si ha iniciado sesión, mantenemos la sesión
      if (!isLogin) this.routerService.goToLogin(); // Si no, redirigimos a la página de login
      return isLogin;
  }

}
```

Figura 4.8: Guardian AuthGuard

En cada módulo creado en la aplicación existe un módulo de enrutamiento donde se define todas las hijas rutas de dicho módulo.

CAPÍTULO 5

MANUAL DE USUARIO

5.1 Pantalla de login 64

5.2 Pantalla de pacientes 66

5.3 Pantalla de profesionales 67

5.4 Pantalla de recetas 68

5.5 Pantalla de rutinas 70

5.6 Pantalla de dietas 70

5.7 Pantalla del paciente 73

5.7.1 Página de gráficas 73

5.7.2 Página de consultas 74

5.7.3 Página de comidas 75

5.7.4 Página de ejercicios 77

5.8 Pantalla de configuración 79

En este capítulo exploraremos las diversas pantallas que componen la aplicación web. Se mostrará una imagen de cada una de ellas para comprender cada una de sus partes.

5.1 Pantalla de login

La pantalla de login, figura 5.1, es donde permite a los distintos profesionales iniciar sesión a la aplicación. En esta, ellos deben introducir el email y contraseña de su cuenta y pulsar sobre el botón 'Iniciar Sesión'. El sistema se encargará de verificar dichas credenciales.

En caso de que el profesional haya olvidado su contraseña, cabe la posibilidad de recuperarla. Para ello, en la pantalla de login, pulsamos sobre el texto '¿Has olvidado la contraseña?', y este te redirirá a la pantalla de recuperar contraseña, figura 5.2. Una vez introducido el email, se enviará un correo con un determinado enlace. Dicho enlace te redirige a la pantalla de cambiar contraseña, figura 5.3. Aquí el profesional podrá registrar la nueva contraseña para loguearse en la aplicación.

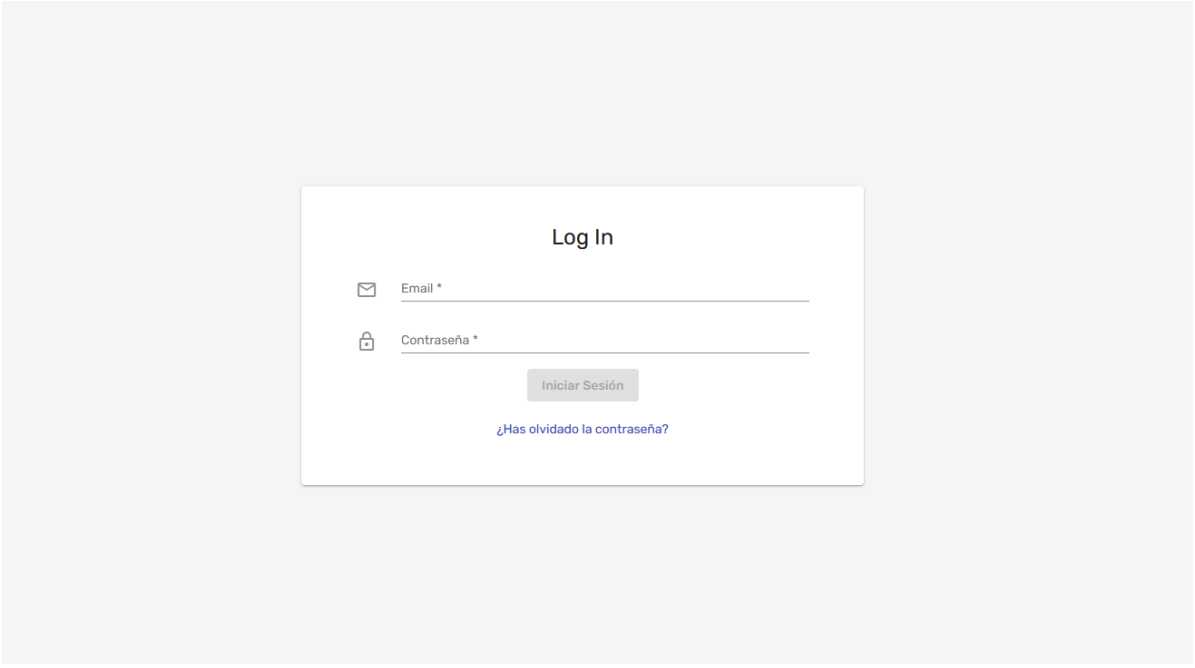

Figura 5.1: Pantalla de login

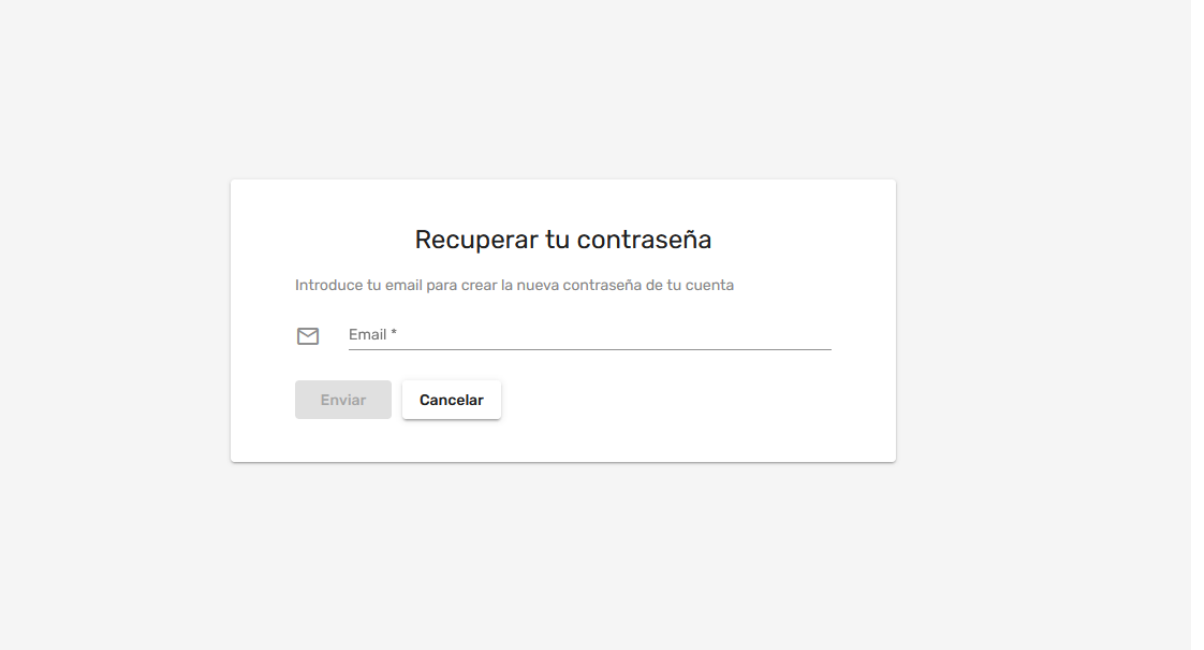

Figura 5.2: Pantalla de recuperar contraseña

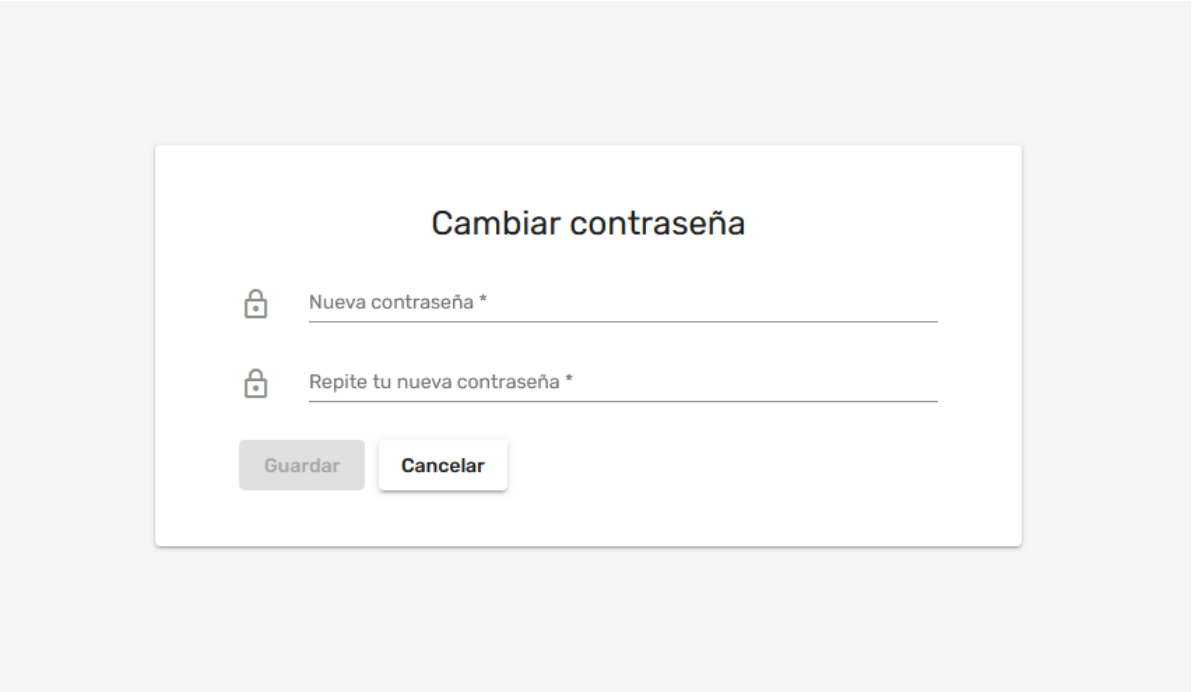

Figura 5.3: Pantalla de cambiar contraseña

5.2 Pantalla de pacientes

Nada más iniciar sesión los profesionales, podrán visualizar la pantalla de los pacientes, figura 5.4, donde estos podrán gestionar a todos sus pacientes. En esta se muestra un listado con todos los pacientes del profesional registrado. En caso de que el profesional sea administrador, aparecerán todos los pacientes registrados en el sistema. Los que no son suyos se marcarán con un fondo gris.

Aquí se puede visualizar la información de cada uno, así como poder eliminarlos o editarlos.

Pulsando sobre el botón 'Agregar Paciente' abrirá una nueva pantalla, semejante al de la figura 5.5, donde el profesional podrá rellenar el formulario para crear un nuevo paciente. Para la edición del paciente se ha utilizado la misma interfaz.

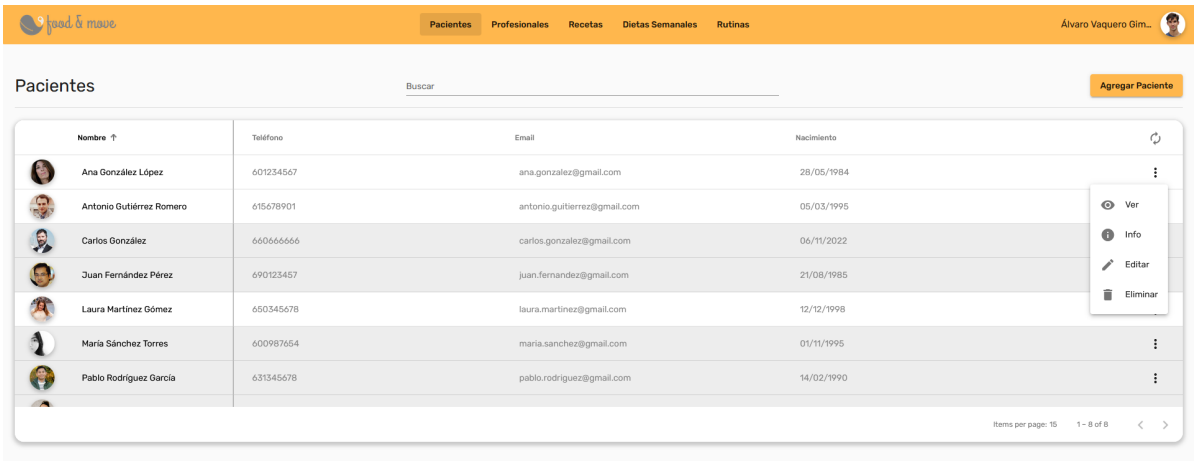

Figura 5.4: Pantalla de pacientes

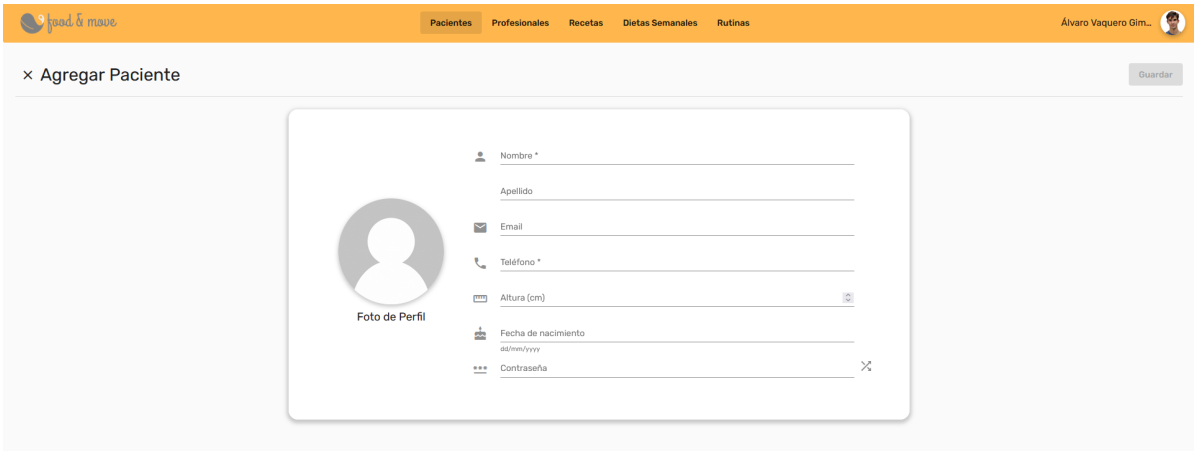

Figura 5.5: Pantalla de agregar/editar un paciente

5.3 Pantalla de profesionales

En esta pantalla los administradores podrán gestionar a todos los profesionales registrados en el sistema, así como ver su información, editarlos, eliminarlos o crear uno nuevo. Como podemos observar en la figura 5.6, la interfaz es muy parecida a la de la gestión de pacientes, figura 5.4.

Pulsando sobre el botón 'Agregar Profesional', nos dirigirá a la pantalla de agregar un nuevo profesional, figura 5.7. La pantalla de edición del profesional usa la misma interfaz.

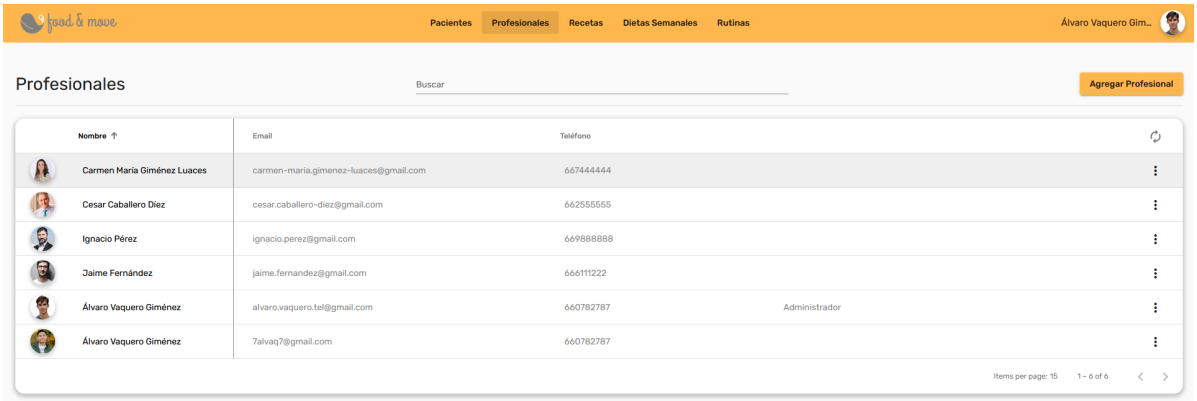

Figura 5.6: Pantalla de profesionales

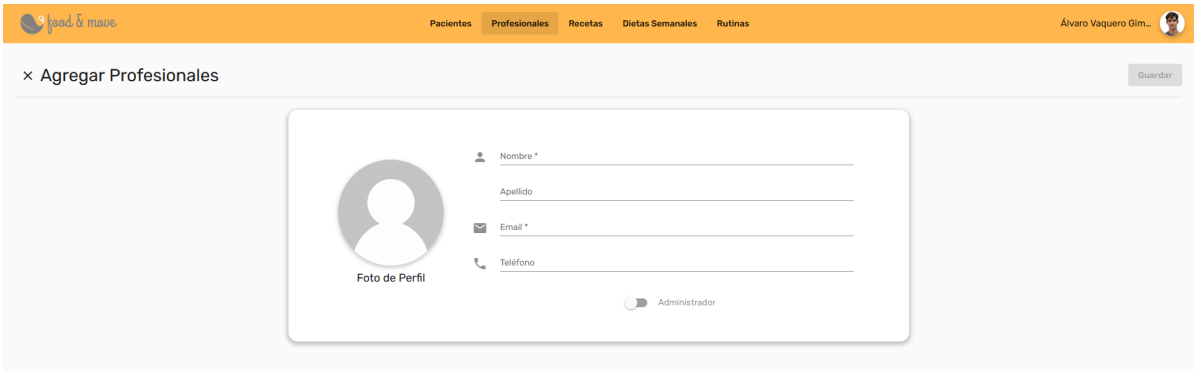

Figura 5.7: Pantalla de agregar/editar un profesional

5.4 Pantalla de recetas

En esta pantalla, figura 5.8, los profesionales podrán gestionar todas las recetas registradas en el sistema. Consta de un listado de todas las recetas. Pulsanso sobre los tres puntos de una determinada receta, nos da la opción de poder visualizar su información, editarla o eliminarla.

Pulsando sobre el botón 'Agregar Receta', nos dirigrirá a la pantalla de crear una nueva receta, con una interfaz semejante a la de la figura 5.9. La pantalla de edición de una receta utiliza la misma interfaz que esta.

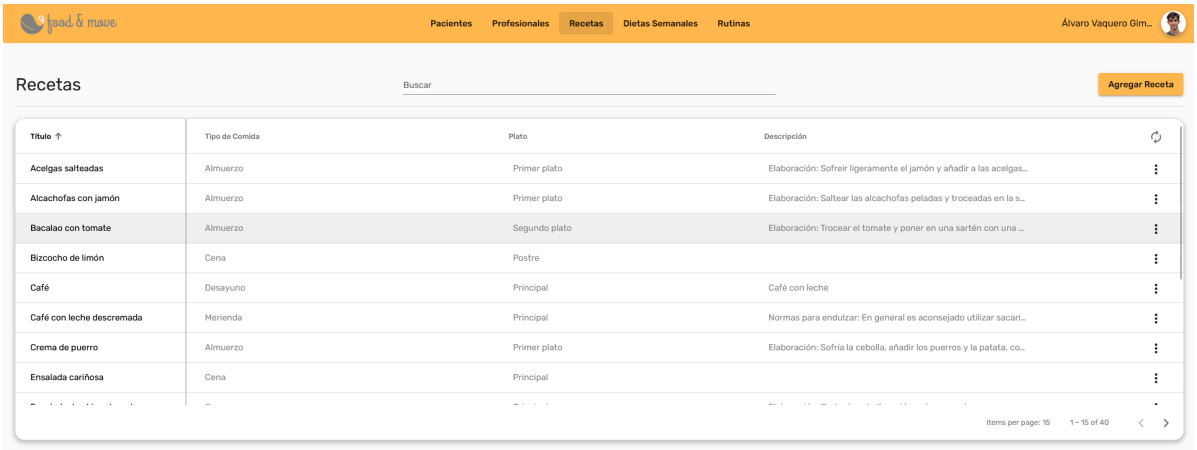

Figura 5.8: Pantalla de recetas

En el caso de crear o editar una receta y se quiere adjuntar un archivo a dicha receta, podemos pulsar sobre el botón 'Importar PDF'. Pulsando este, se abrirá ventana como la de la figura 5.10. Aquí podemos seleccionar un archivo ya existente o crear uno nuevo.

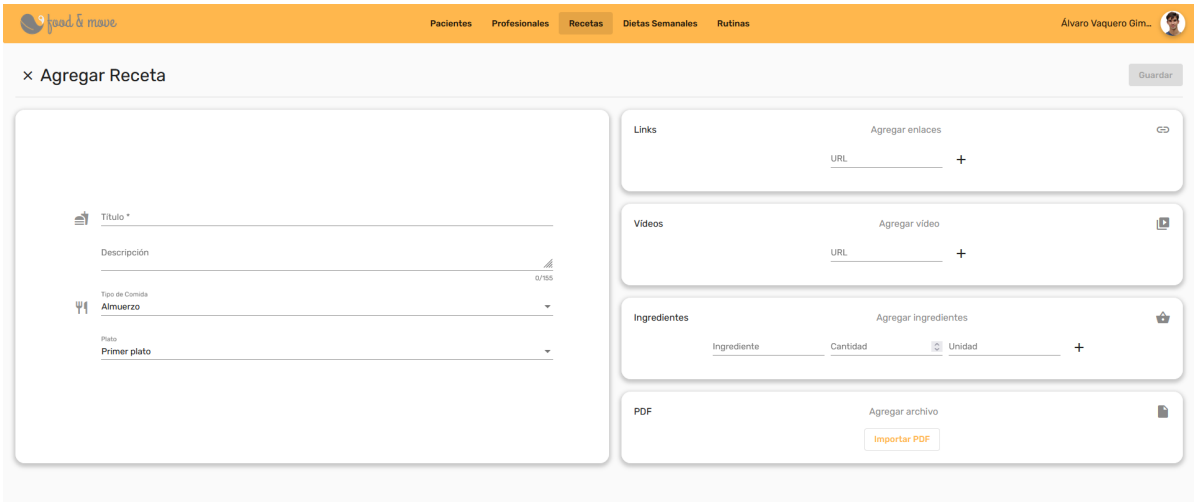

Figura 5.9: Pantalla de agregar/editar una receta

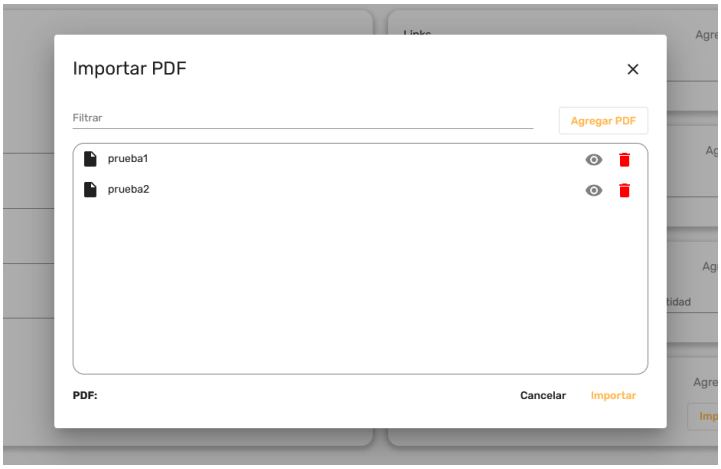

Figura 5.10: Ventana de importar pdf

5.5 Pantalla de rutinas

En esta pantalla, figura 5.11, los profesionales podrán gestionar todas las rutinas registradas en el sistema. Consta de un listado de todas las rutinas. Pulsanso sobre los tres puntos de una determinada rutina, nos da la opción de poder visualizar su información, editarla o eliminarla.

Pulsando sobre el botón 'Agregar Rutina', nos dirigrirá a la pantalla de crear una nueva rutina, con una interfaz semejante a la de la figura 5.12. La pantalla de edición de una rutina utiliza la misma interfaz que esta.

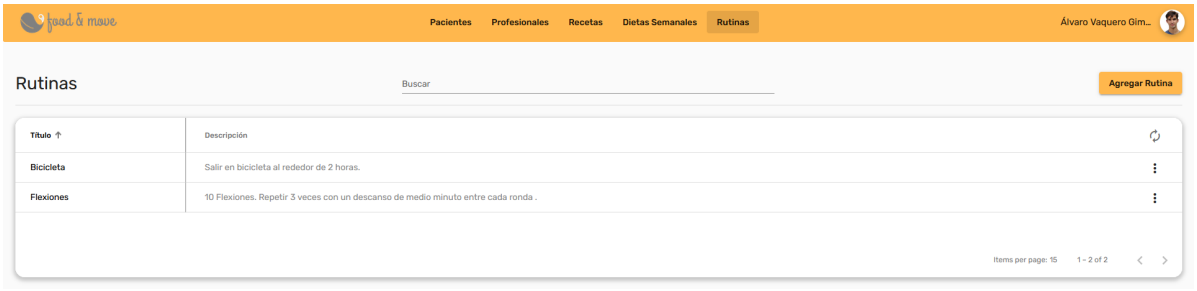

Figura 5.11: Pantalla de rutinas

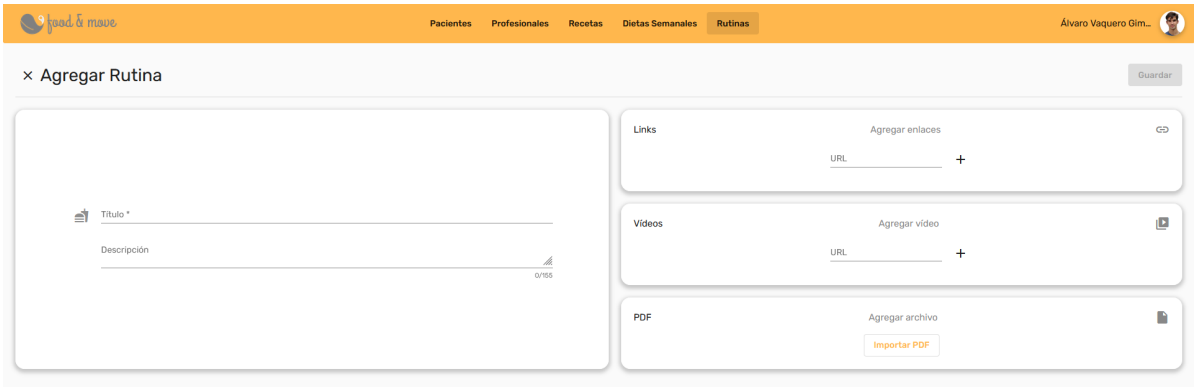

Figura 5.12: Pantalla de agregar/editar una rutina

5.6 Pantalla de dietas

En la esta pantalla, figura 5.13, los profesionales pueden gestionar todas las dietas. Aquí se listan todas las dietas y pulsando sobre los tres puntos en una dieta concreta, nos permite editarla o eliminarla.

Para poder crear una nueva dieta se debe pulsar sobre el botón 'Agregar Dieta Semanal'. Aparecerá una ventana donde el profesional debe introducir el título y descripción de la dieta.

Cuando seleccionamos la edición de una dieta, podemos observar una interfaz como la de la figura 5.14. En esta, podemos observar los platos que hay asignados en cada día de la semana. Para poder editar cada uno de estos o ver su información, solo basta con pulsar dicho plato. En el caso de eliminarlo, podemos pulsar sobre la cruz que encontramos a la derecha de cada plato de comida. Para agregar un nuevo plato, se puede pulsar sobre el icono situado arriba a la derecha del día que se quiere asignar.

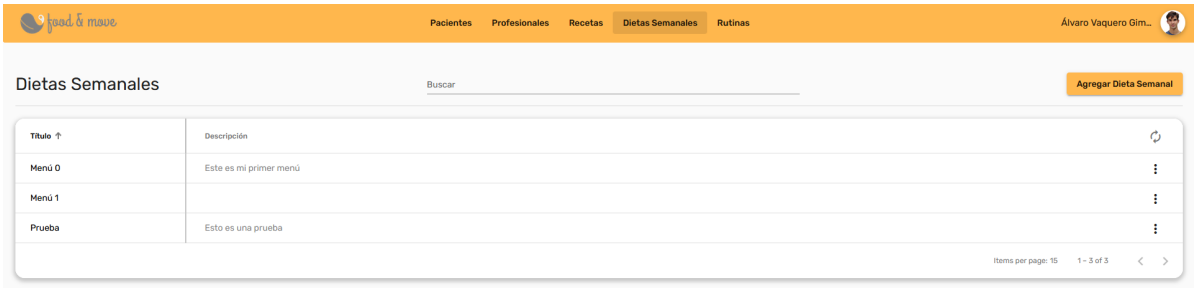

Figura 5.13: Pantalla de dietas

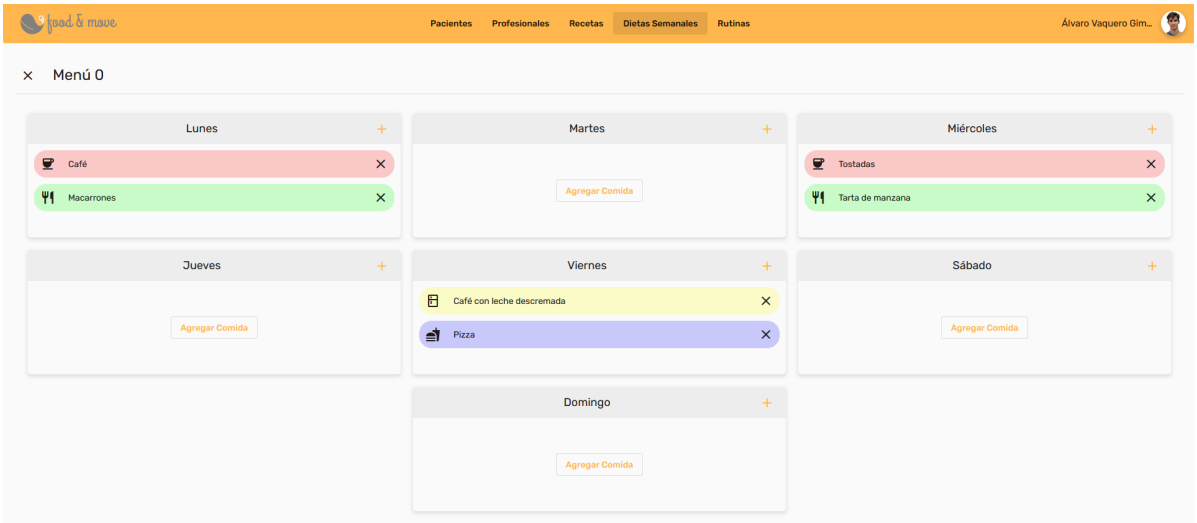

Figura 5.14: Pantalla de editar una dieta

Cuando agregamos o editamos un plato de comida podemos visualizar una interfaz como la de la figura 5.15. El profesional puede rellenar todos los campos o importar una dieta ya asignada pulsando sobre el botón 'Importar Receta'. Cuando pulsamos este, podemos visualizar una ventana como la de la figura 5.16, donde podemos buscar y seleccionar la receta deseada.

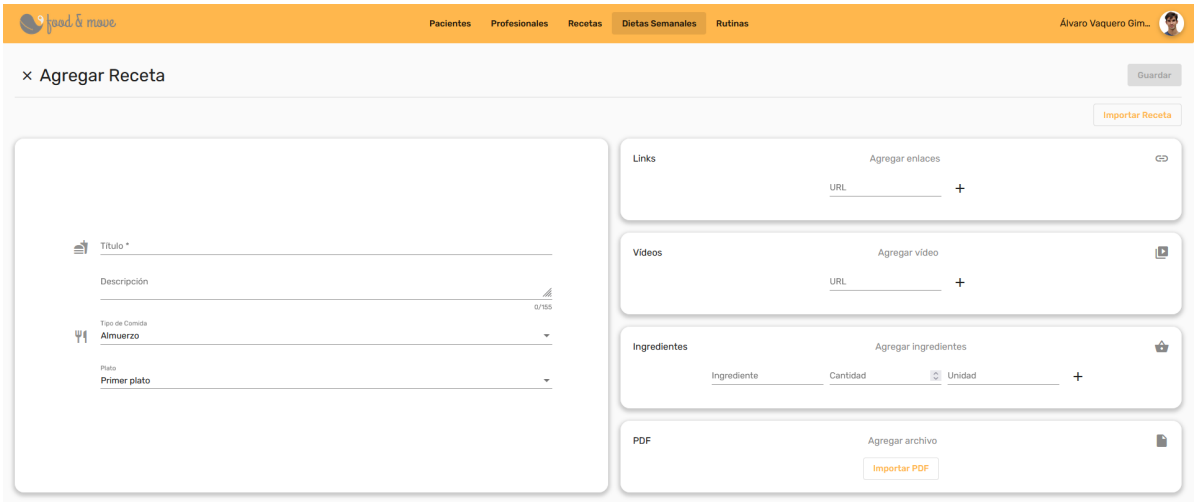

Figura 5.15: Pantalla de agregar comida

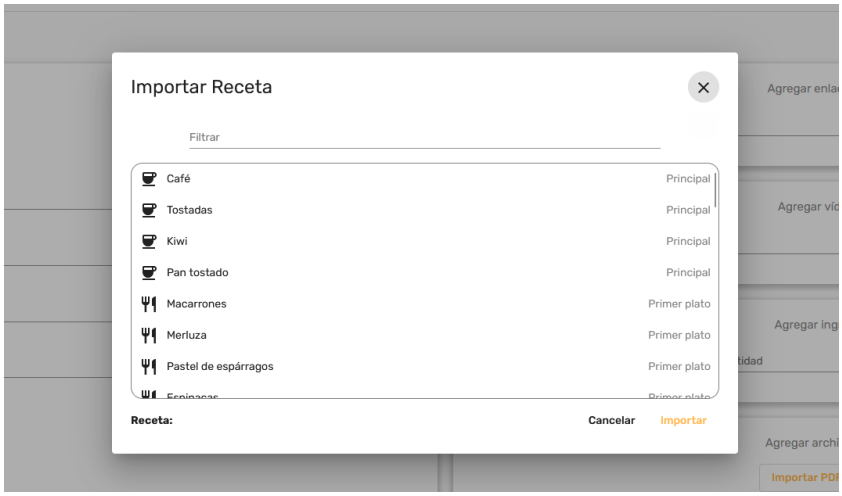

Figura 5.16: Ventana de importar receta

5.7 Pantalla del paciente

Dentro de la pantalla de los pacientes, figura 5.2, podemos seleccionar a uno de nuestros pacientes. Al hacerlo, nos redirigirá a la página del paciente, donde dentro de este podemos encontrar las siguientes páginas:

- Gráficas [5.7.1]
- Consultas [5.7.2]
- Comidas [5.7.3]
- Ejercicios [5.7.4]

5.7.1 Página de gráficas

En esta página, figura 5.17, podemos visualizar gráficamente la evolución temporal de cada uno de los parámetros fisiológicos del paciente. Los profesinales pueden seleccionar el rango de fechas que desean visualizar. En caso contario, se visualizará desde la primera consulta hasta la última.

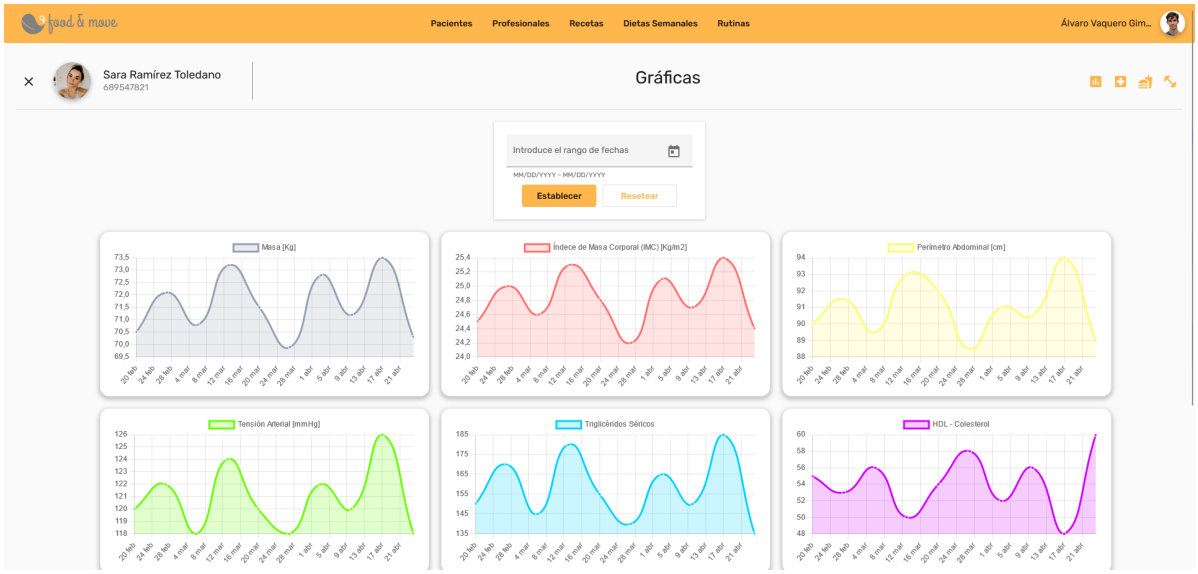

Figura 5.17: Página de las gráficas de un paciente

5.7.2 Página de consultas

En esta página, figura 5.18, los profesionales pueden gestionar todas las consultas del paciente seleccionado. Pulsando sobre les tres puntos de una consulta, nos permite visualizar su información, editarla o eliminarla.

Para poder agregar una nueva consulta, se debe pulsar sobre el botón 'Agregar Consulta'. Abrirá una nueva página como la de la figura 5.19. La interfaz de la edición de las consultas es la misma.

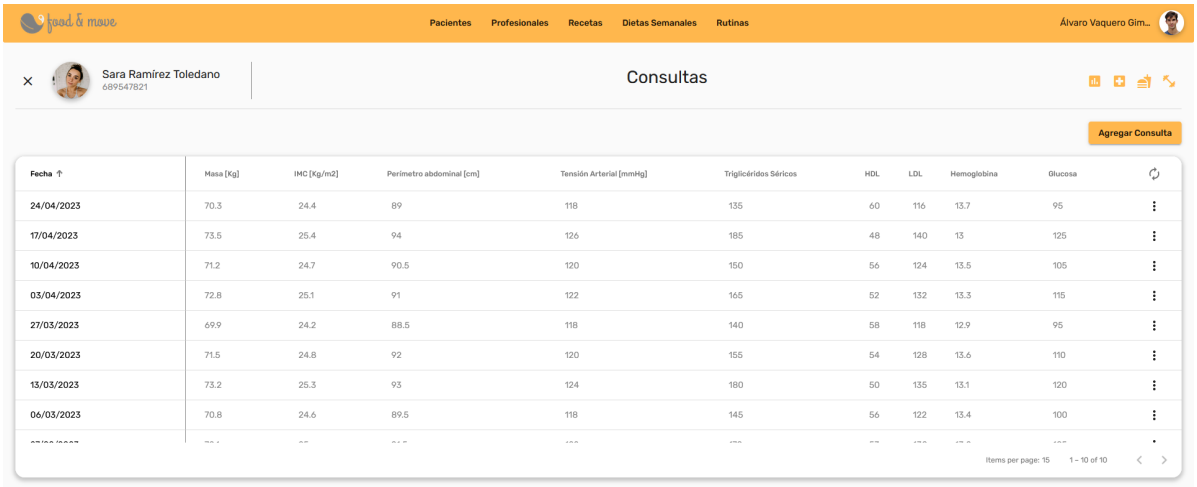

Figura 5.18: Página de las consultas de un paciente

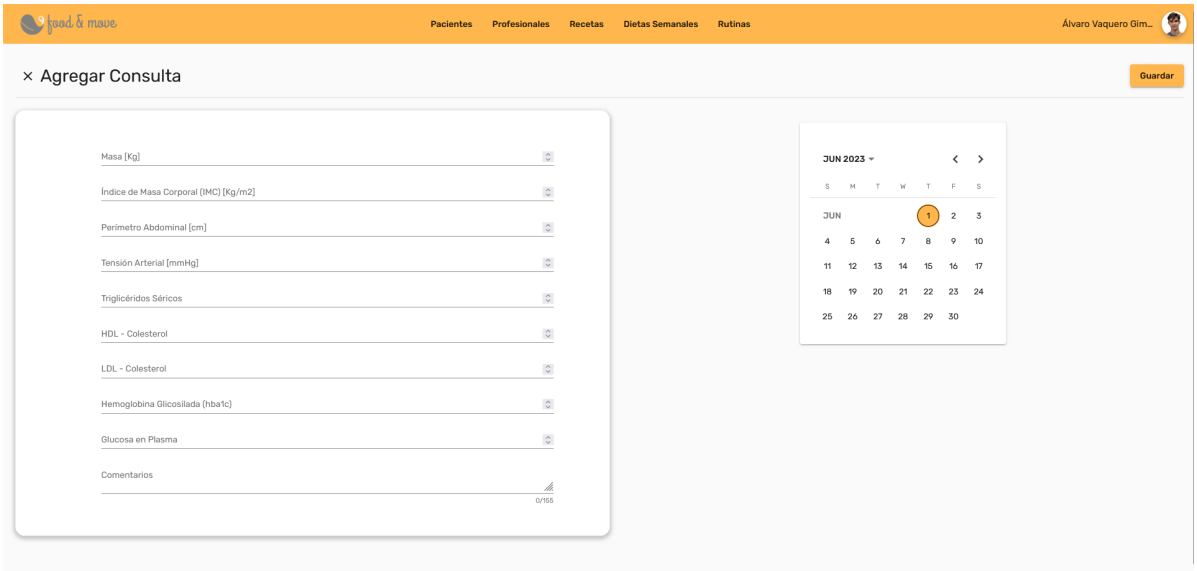

Figura 5.19: Página de agregar/editar una consulta

5.7.3 Página de comidas

Dentro de esta página, figura 5.20, podemos visualizar y gestionar las comidas que son asignadas al paciente seleccionado. Podemos visualizar un calendario, en la cual podemos ir cambiando de semana, y viendo qué comidas se le ha asignado en cada día.

Pulsando sobre el botón 'Importar Dieta' se abrirá una ventana, semejante a la de la figura 5.21, donde podemos buscar y seleccionar una dieta que queramos importar en la semana seleccionada.

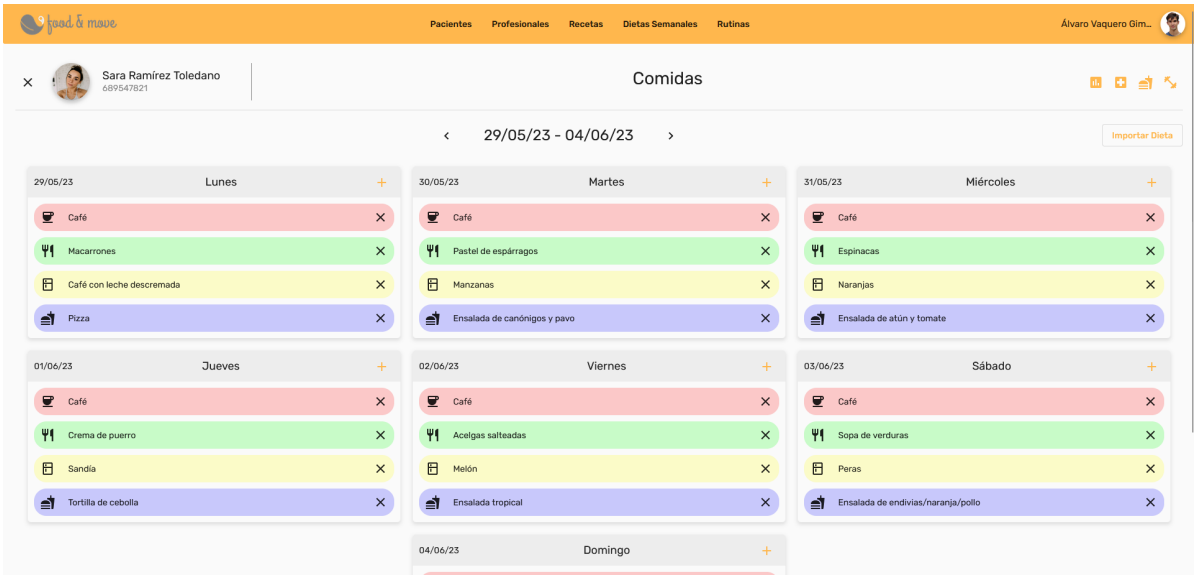

Figura 5.20: Página de comidas de un paciente

Pulsando sobre un plato de comida podemos ver su información y poder modificarlo. En el caso de que queramos agregar una nueva comida, podemos pulsar sobre el icono que se encuentra en la parte superior derecha del día deseado. Este abrirá una página como la de la figura 5.22. Esta interfaz es la misma tanto para la edición como para la creación de comidas.

Pulsando sobre el botón 'Importar Receta' podemos importar una receta ya prescrita.

Figura 5.21: Ventana de importación de dieta

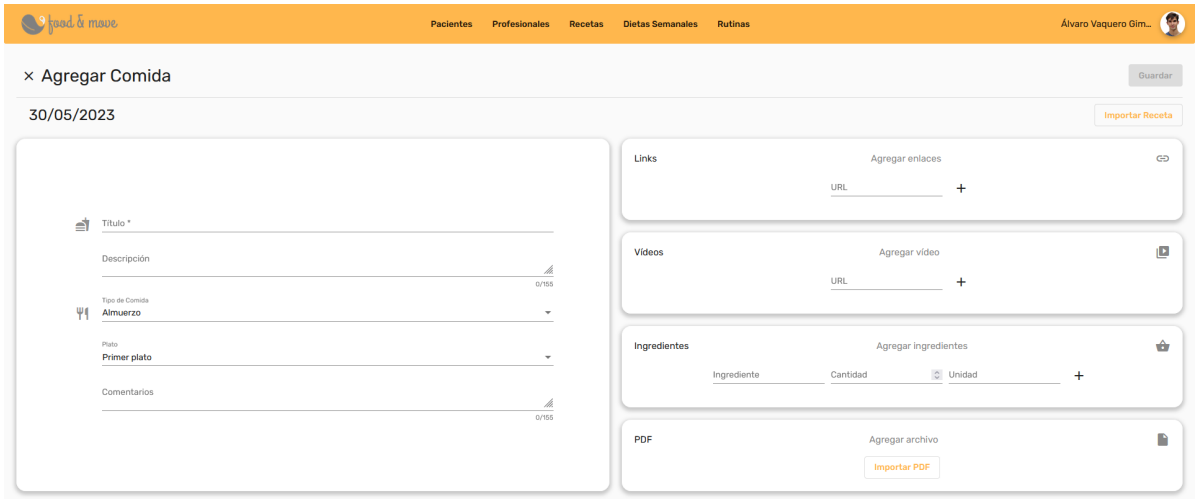

Figura 5.22: Página de agregar/editar comida de un paciente

5.7.4 Página de ejercicios

En esta página, figura 5.23, los profesionales pueden gestionar todos los ejercicios que son asignados al paciente seleccionado. Está página está constituido por un calendario, en la cual podemos ir cambiando de semana, donde podemos visualizar todos los ejercicios que son asignados en cada día.

Pulsando sobre el icono situado en la parte superior derecha del día deseado, podemos asignar un nuevo ejercicio. Aparecerá una nueva pantalla, como la de la figura 5.24. Ahí, los profesionales pueden rellenar el formulario o importar una rutina ya prescrita pulsando el botón de 'Importar Rutina'. En ese caso, aparecerá una ventana como la de la figura 5.25.

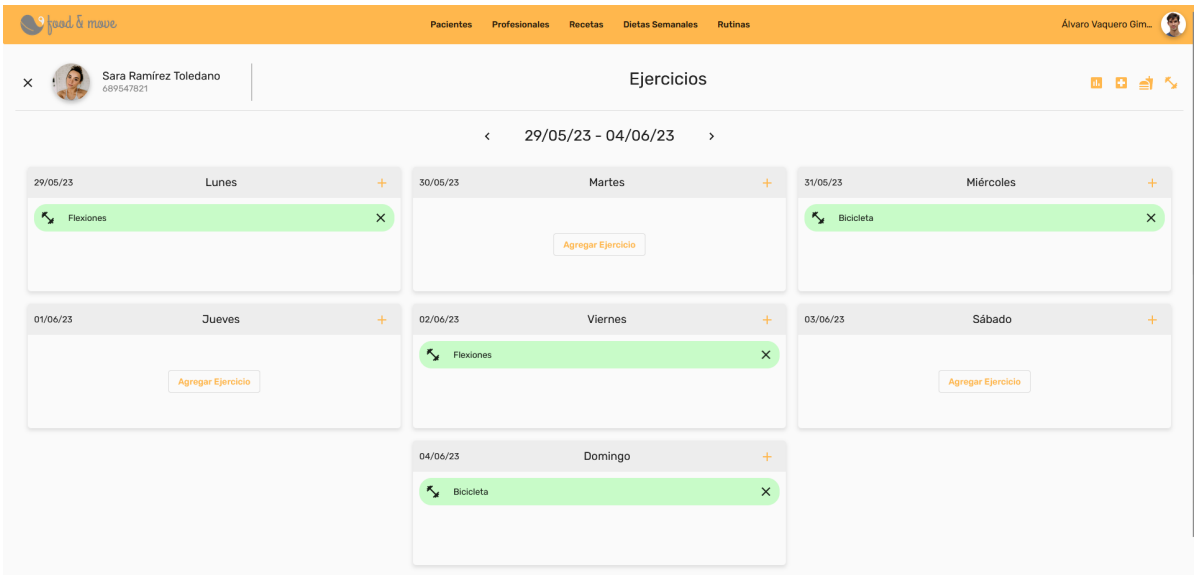

Figura 5.23: Página de ejercicios de un paciente

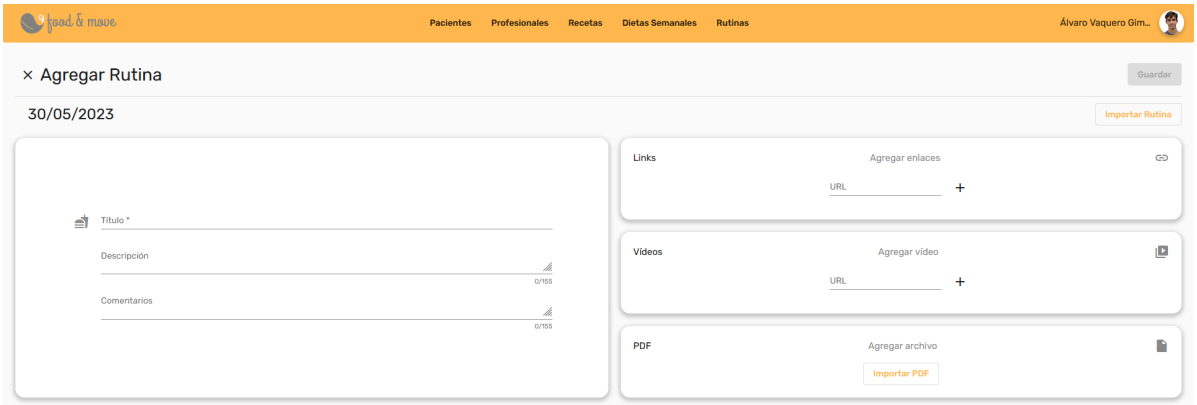

Figura 5.24: Página de agregar/editar un ejercicio de un paciente

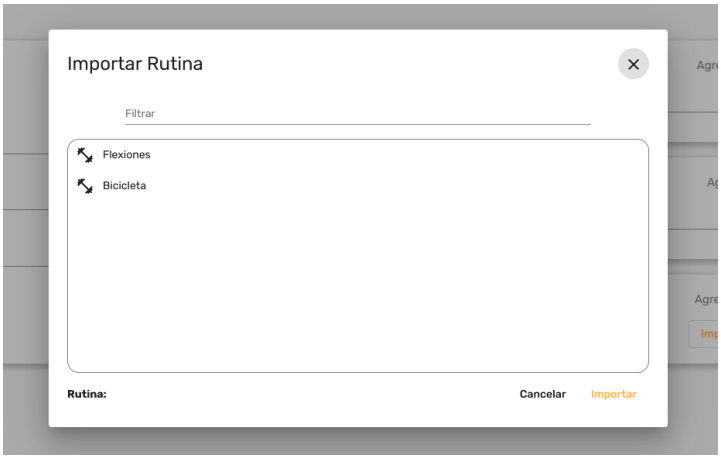

Figura 5.25: Ventana de importación de rutina

5.8 Pantalla de configuración

En el menú que podemos encontrar en la parte superior de las pantalla, si pulsamos sobre el profesional que está registrado, nos aparecerá una pequeña ventana como la de la figura 5.26.

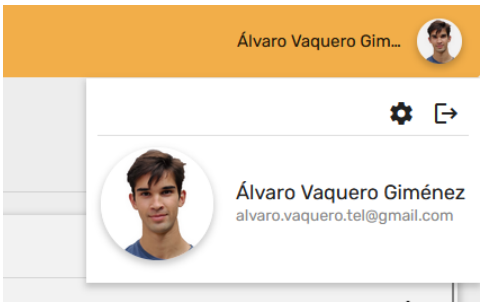

Figura 5.26: Ventana del profesional registrado

Esta ventana nos permite cerrar sesión y dirigirnos a la pantalla de configuración.

En la pantalla de configuración, el profesional registrado puede editar los datos de su perfil, figura 5.27, y poder cambiar la contraseña, figura 5.28.

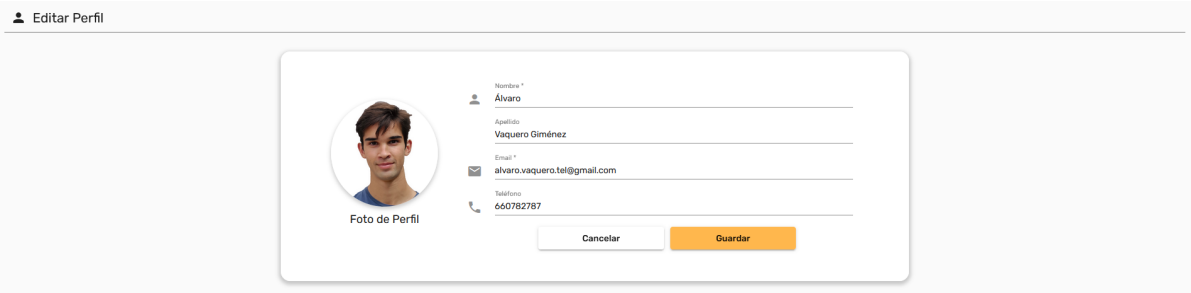

Figura 5.27: Edición del perfil del usuario

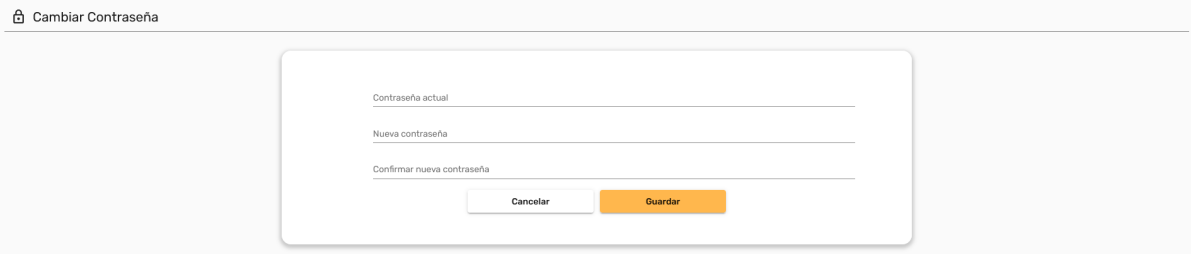

Figura 5.28: Cambio de contraseña del usuario

CAPÍTULO 6

MANUAL DEL DESARROLLADOR

6.1 Entorno de producción 81
6.1.1 Aplicación web 82
6.1.2 API 83
6.2 Variables de entorno 84

Una vez desarrollada la aplicación, el siguiente paso ha sido llevar el proyecto a un entorno de producción, es decir, que el proyecto pase de ejecutarse en un ordenador local a un servidor donde se pueda acceder desde otras redes.

En este apartado se prentende explicar el proceso que se ha llevado a cabo. Se hablará de cómo subir el proyecto al servidor y en qué zonas del código se encunentran las variables que definen las conexiones con el resto de componentes.

6.1 Entorno de producción

En esta sección se presentará un manual de cómo poder llevar a un entorno de producción este proyecto.

La base de datos, tanto en el entorno de desarrollo como en el entorno de producción, se encuentra ubicada en 'MongoDB Atlas Database' [2], que es una plataforma de alojamiento en la nube de bases de datos y que es administrada por MongoDB.

La aplicación web y la API se encuentra ubicadas en un servidor, estos son administrados por las siguientes herramientas:

- **NGINX:** Es un servidor web open source que ofrece el contenido estático de un sitio web. Este gestiona multitud de conexiones simultáneas [10].
- **PM2:** Es un gestor de procesos en producción para las aplicaciones Node.js. Este gestiona el registro de aplicaciones [13].

En la figura 6.1 podemos ver la configuración de NGINX. En este indicamos el nombre del servidor: *foodandmove.app.bluece.eu*, el puerto donde escucha: *80*, y la localización tanto de la aplicación web (ruta raíz) como la API (*/api*).

```
server {
    server_name foodandmove.app.bluece.eu;
    root /var/www/public/food-and-move/frontend-angular/;
    location / {
        try_files $uri $uri/ /index.html;
    }

    location /api/ {
        proxy_pass http://localhost:3002/api/;
    }


    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/foodandmove.app.bluece.eu/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/foodandmove.app.bluece.eu/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    if ($host = foodandmove.app.bluece.eu) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    return 404; # managed by Certbot
}
```

Figura 6.1: Configuración de NGINX

6.1.1 Aplicación web

Para poder subir la aplicación web he desarrollado un archivo llamado *subir_app.sh*, ubicado en el directorio */home/foodmove*. Este ejecuta un conjunto de comandos de BASH de Unix. El contenido de este fichero lo podemos ver en la figura 6.2.

Para llevar a producción la aplicación web se debe seguir los siguientes pasos:

1. Ubicarse en la carpeta */home/foodmove*.

```
#!/bin/bash
sudo -v
echo --- SUBIR APP ---
cd food-move-frontend
npm install
npm run build
sudo rm -rf ../../../var/www/public/food-and-move/frontend-angular
cd dist
sudo mv frontend-angular ../../../../var/www/public/food-and-move
sudo nginx -t && sudo systemctl restart nginx
echo Terminado!!
```

Figura 6.2: Subir a producción la aplicación web

2. Clonar el repositorio del frontend: *git clone https://github.com/alvarovaq/food-move-frontend.git*
3. Ejecutar el fichero anteriormente mencionado: *.\subir_app.sh*

6.1.2 API

Para poder subir la API he desarrollado un archivo llamado *subir_api.sh*, ubicado en el directorio */home/foodmove*. Este ejecuta un conjunto de comandos de BASH de Unix. El contenido de este fichero lo podemos ver en la figura 6.3.

```
#!/bin/bash
echo --- SUBIR API ---
cd food-move-backend
npm install
echo ---
npm run build
echo ---
pm2 restart foodmoveback
echo ---
echo Terminado!!
```

Figura 6.3: Subir a producción la API

Para llevar a producción la API debemos de seguir los siguientes pasos:

1. Ubicarse en la carpeta */home/foodmove*.
2. Clonar el repositorio del frontend: *git clone https://github.com/alvarovaq/food-move-backend.git*
3. Ejecutar el fichero anteriormente mencionado: *.\subir_api.sh*

Este solo sirve si el proceso pm2 ya está arrancado. Para ello debemos de ubicarnos en la carpeta del repositorio del backend y ejecutar el siguiente comando: *production=true pm2 start dist/main.js –watch –ignore-watch="node_modules-name foodmoveback*.

Para comprobar si el proceso ya está iniciado o no podemos escribir el siguiente comando: *pm2 status*. En el listado que aparece por pantalla debe aparecer uno con el nombre *foodmoveback*.

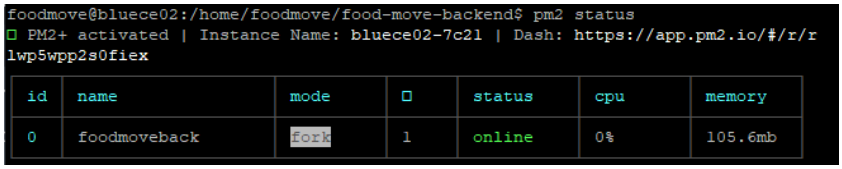

Figura 6.4: Estado de pm2

6.2 Variables de entorno

En esta sección se mostrará las ubicaciones de dónde se declaran las distintas conexiones entre los componentes del proyecto dentro del código. De esta forma nos resultará más fácil poder modificar el código en caso de cambiar la ubicación del servidor o de la base de datos.

Dentro de la aplicación web, debemos indicar la dirección de la API. Este está declarada en el fichero *src/environments/environment.prod.ts*. Podemos visualizarlo en la figura 6.5.

```
export const environment = {
  production: true,
  api: 'https://foodandmove.app.bluece.eu/api'
};
```

Figura 6.5: Dirección de la API

En el código del backend o API debemos indicar la dirección de la base de datos. Para ello lo indicamos en el módulo *src/app.module.ts*. Podemos visualizarlo en la figura 6.6.

```
imports: [MongooseModule.forRoot('mongodb+srv:/                    .mongodb.net/?retryWrites=true&w=majority', {
  useNewUrlParser: true
}),
```

Figura 6.6: Dirección de la base de datos

CAPÍTULO 7

CONCLUSIONES

7.1 Resultados finales 85
7.2 Líneas futuras 86
7.3 Valoración personal 86

7.1 Resultados finales

Una vez finalizado el desarrollo y habiendo subido el proyecto a un entorno de producción, pudimos comprobar el funcionamiento del proyecto en conjunto, tanto la aplicación web como la aplicación móvil, simulando un entorno real.

En él pudimos comprobar el correcto funcionamiento de cada una de las partes, así como el funcionamiento en conjunto. Los resultados fueron exitosos, habiendo cumplido con los requisitos del producto mínimo viable que nos marcamos al inicio del proyecto.

Lamentablemente, por falta de tiempo, no se ha podido probar en un entorno con pacientes y profesionales de la salud mental, y por lo tanto, no se ha podido recoger feedbacks de usuarios. Esto se hará posteriormente a este trabajo, utilizándolo en el centro especializado de Zamora, como ya se ha mencionado anteriormente.

Llegados a este punto, se ha desarrollado tanto una aplicación web como una API que cumplen satisfactoriamente con las funcionalidades principales del sistema, alcanzando los objetivos esperados.

7.2 Líneas futuras

Llegados a este punto, se me ocurren algunas nuevas funcionalidades que se podrían implementar en las posteriores versiones:

- Poder ceder un paciente a otro profesional. En numerosas ocasiones, como puede ser el caso de que un profesional sanitario se coja unos días de vacaciones o se dé de baja, estaría bien implementar una funcionalidad que permita ceder un paciente a otro profesional para que este pueda tener la gestión del mismo.
- Asignar una dieta semanal constante a cada paciente. Aquí me refiero a que los profesionales puedan asignar una dieta a un determinado paciente y que esta dieta sea igual para todas las semanas hasta que este lo cambie. Esto haría ahorrar bastante tiempo al profesional y que no tenga que preocuparse cada semana de asignar nuevas dietas a cada paciente.
- Análisis estadístico de la evolución temporal de los parámetros. Este es quizás el punto más interesante de mejorar, implementar un módulo de análisis de datos que combine múltiples variables (no solo las que recoge esa aplicación sino también otras como son las de hábitos de higiene o patrones conductuales en el hogar) con el objetivo de tener un mejor y más completo perfil de la persona para poder ayudarle mejor, adelantándonos a posibles crisis o recaídas.

7.3 Valoración personal

Durante el desarrollo de este proyecto, he podido aprender nuevas tecnologías que no conocía previamente, y he podido aumentar mis habilidades en el desarrollo de aplicaciones web y APIs.

Desde que empecé la carrera, la rama de las telecomunicaciones que más me ha apasionado es la programación, y después de realizar este TFG, sé con seguridad que es donde quiero enfocar mi carrera profesional.

A lo largo este proyecto, me he enfrentado a nuevos problemas y desafíos de forma autónoma, lo que me ha ayudado poder mejorar mis habilidades de resolución de problemas.

Estoy muy satisfecho con la experiencia de desarrollar este TFG y con ilusión de que este pueda ser utilizado en un entorno real y pueda ayudar a bastantes personas.

Índice de código fuente

- **Código del frontend**
  https://github.com/alvarovaq/food-move-frontend
- **Código del backend**
  https://github.com/alvarovaq/food-move-backend

BIBLIOGRAFÍA

[1] Angular: *Documentación oficial de angular*. https://angular.io/docs. Accedido el 10 de junio de 2023.

[2] MongoDB Atlas: *Documentación oficial de mongodb atlas*. https://www.mongodb.com/docs/atlas/. Accedido el 16 de junio de 2023.

[3] MongoDB Atlas: *Página oficial de mongodb atlas*. https://www.mongodb.com/atlas/database. Accedido el 17 de junio de 2023.

[4] Anjali Chauhan: *A review on various aspects of mongodb databases*. International Journal of Engineering Research Technology (INJERT), 8(5), may 2019.

[5] Visual Studio Code: *Página oficial de visual studio code*. https://code.visualstudio.com/. Accedido el 17 de junio de 2023.

[6] Git: *Página oficial de git*. https://git-scm.com/. Accedido el 17 de junio de 2023.

[7] GitHub: *Página oficial de github*. https://github.com/. Accedido el 17 de junio de 2023.

[8] MongoDB: *Documentación oficial de mongodb*. https://www.mongodb.com/docs/. Accedido el 13 de junio de 2023.

[9] NestJS: *Documentación oficial de nestjs*. https://docs.nestjs.com/. Accedido el 13 de junio de 2023.

[10] NGINX: *Documentación oficial de nginx*. https://docs.nginx.com/. Accedido el 16 de junio de 2023.

[11] OpenVPN: *Página oficial de openvpn*. https://openvpn.net/. Accedido el 17 de junio de 2023.

[12] Overleaf: *Página oficial de overleaf*. https://es.overleaf.com/. Accedido el 17

de junio de 2023.

[13] PM2: *Documentación oficial de pm2*. https://pm2.keymetrics.io/docs/usage/pm2-doc-single-page/. Accedido el 16 de junio de 2023.

[14] Postman: *Página oficial de postman*. https://www.postman.com/. Accedido el 17 de junio de 2023.

[15] React: *Documentación oficial de react*. https://legacy.reactjs.org/docs/getting-started.html. Accedido el 13 de junio de 2023.

[16] StarUML: *Página oficial de staruml*. https://staruml.io/. Accedido el 17 de junio de 2023.

[17] Narcisa Portalanza Valverde, Vanessa y Paulina Mora: *Análisis descriptivo de base de datos relacional y no relacional*. Revista Atlante: Cuadernos de Educación y Desarrollo, 3, 2019.

[18] Vue.js: *Documentación oficial de vue.js*. https://vuejs.org/guide/introduction.html. Accedido el 13 de junio de 2023.

[19] Yenisleidy Fernández Romero y Yanette Díaz González: *Patrón modelo-vista-controlador*. Telem@tica (La Habana), 11(1):47–57, abril 2012. https://biblat.unam.mx/es/revista/telemtica-la-habana/articulo/patron-modelo-vista-controlador.