UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una aplicación móvil para el seguimiento y monitorización en personas con esclerosis múltiple.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Toledo Gonzalez, Javier

Tutor/a: Herrero Cucó, Carlos

CURSO ACADÉMICO: 2021/2022

RESUMEN

En los últimos 20 años, el número de enfermos diagnosticados con Esclerosis Múltiple se ha duplicado, llegando a cifras de 2,5 millones de personas en todo el mundo, 50.000 solo en España. La esclerosis es una enfermedad crónica degenerativa que se suele diagnosticar entre los 20 y 40 años.

Es por esto por lo que la manera de controlar el desarrollo de la enfermedad en los pacientes debe evolucionar, para ello se propone el desarrollo de una aplicación web para *smartphones* que ayude al seguimiento particular de la enfermedad, propuestas de actividad física, recordatorios como ayuda para el día a día, así como poniendo a disposición consejos para que el usuario pueda incrementar su conocimiento de esta enfermedad. Para ello, se hará uso de Angular y Node.js para la implementación de la aplicación móvil.

Palabras claves: Desarrollo móvil, aplicación móvil, Angular, esclerosis múltiple, actividad física.

RESUM

En els últims 20 anys, el numere de malalts diagnosticats amb Esclerosi Múltiple s'ha duplicat, arribant a xifres de 2,5 milions de persones a tot el món, 50.000 només a Espanya. L'esclerosi és una malaltia crònica degenerativa que se sol diagnosticar entre els 20 i 40 anys.

És per això que la manera de controlar el desenvolupament de la malaltia en els pacients ha d'evolucionar, per a això es proposa el desenvolupament d'una aplicació web per a telèfons intel·ligents que ajude al seguiment particular de la malaltia, propostes d'activitat física, recordatoris com a ajuda per al dia a dia, així com posant a disposició consells perquè l'usuari puga incrementar el seu coneixement d'aquesta malaltia. Per a això, es farà ús d'Angular i Node.js per a la implementació de l'aplicació mòbil.

ABSTRACT

In the last 20 years, the number of patients diagnosed with Multiple Sclerosis has doubled, reaching figures of 2.5 million people worldwide, 50,000 in Spain only. Sclerosis is a chronic degenerative disease that is usually diagnosed between the ages of 20 and 40.

This is why the way to control the development of the disease in patients must evolve, for this purpose the development of a web application for smartphones is proposed to help the particular monitoring of the disease, proposals for physical activity, reminders as an aid for day by day, as well as providing advice so that the user can increase their knowledge of this disease. To do this, Angular and Node.js will be used for the implementation of the mobile application.

CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

La esclerosis múltiple (EM) es una de las enfermedades más comunes del sistema nervioso central, que está compuesto por el cerebro y la médula espinal, los cuales desempeñan el trabajo principal de todo el sistema nervioso y controlan todas las funciones del cuerpo [1].

La EM es una afección inflamatoria desmielinizante idiopática. Esto significa que es causado por daño a la mielina, un material graso que aísla los nervios. La pérdida de mielina se acompaña de una interrupción en la capacidad de los nervios para conducir impulsos eléctricos hacia y desde el cerebro. Esto produce los diversos síntomas de la EM.
Estos síntomas varían ampliamente e incluyen visión borrosa, extremidades débiles, sensación de hormigueo, inestabilidad y fatiga. Para algunas personas, la esclerosis múltiple se caracteriza por períodos de recaída y remisión, mientras que para otras tiene un patrón progresivo.

Mejorar la calidad de vida y seguir un estilo de vida responsable ayuda a las personas con esclerosis múltiple a gestionar su enfermedad. Esta mejora incluye seguir una dieta saludable, la actividad física, rehabilitación, un sueño adecuado y la estimulación del cerebro, entre otros aspectos orientados al bienestar.

La EM es una enfermedad autoinmune, crónica, inflamatoria, desmielinizante del sistema nervioso central que se presenta en individuos genéticamente susceptibles y que involucra a factores inmunológicos como anticuerpos y mediadores de la respuesta inmune innata. La esclerosis se caracteriza por ataques recurrentes multifocales con grados variables de recuperación. Estos ataques son llamados brotes, episodios en los que empeorarán los síntomas ya existentes o aparecerán nuevos. Estos episodios necesitarán una recuperación que puede ser completa, o bien puede dejar alguna secuela [2].

1.1. MOTIVACIÓN

La esclerosis múltiple es una enfermedad que no tiene cura, así que el tratamiento se basa en acelerar el proceso de recuperación tras los brotes, tratar los síntomas y de prevenir otras enfermedades relacionadas. Es por ello por lo que un correcto seguimiento y monitorización de la enfermedad es necesario para el enfermo o su persona cuidadora.

La realización de este proyecto está motivada por el desarrollo de una aplicación móvil, cuyas principales funcionalidades son la facilitación de consejo, uso de un calendario para citas médicas, recomendación de actividades físicas y seguimiento de la medicación, con el fin de facilitar la ayuda necesaria a los enfermos con esclerosis múltiple, haciéndoles llegar el conocimiento necesario para poder convivir con esta enfermedad con mayor facilidad. Tampoco se pretende sustituir el trabajo de los profesionales sanitarios con este proyecto, si no a la cooperación y trabajo conjunto.

1.2. OBJETIVOS

Para este proyecto se han marcado una serie de objetivos, dado las necesidades de los usuarios diagnosticados. Como objetivo fundamental, se desea que el usuario afectado pueda participar activamente en la gestión de su enfermedad, trabajando juntamente con los profesionales. Además de ello, también se han marcado algunos objetivos secundarios:

- Permitir a los usuarios hacer un seguimiento de su medicación.
- Permitir a los usuarios gestionar sus documentos para tenerlos siempre a mano.
- Permitir a los usuarios una gestión de citas o recordatorios a través de un calendario personal.
- Mantener informado al usuario de avances, aprendizajes y consejos relacionados con la esclerosis múltiple.
- Facilitar al usuario distintos entrenamientos ideados para personas diagnosticadas con esclerosis múltiple.

1.3. ESTRUCTURA

La memoria de este Trabajo de Fin de Grado se divide en ocho capítulos junto a dos anexos. A continuación, se enumeran estos capítulos de forma ordenada junto a una breve explicación.

- Introducción: Capitulo formado por una breve explicación de la enfermedad, la motivación atrás de la aplicación desarrollada, los objetivos y la estructura de la memoria.
- Estado del arte: En este apartado se exponen las aplicaciones principales con funcionalidades similares al proyecto, terminando con la propuesta de la solución.
- Diseño de la solución: Esta sección está conformada por una explicación de la metodología usada, un estudio e identificación de las necesidades de los usuarios y la exposición de las tecnologías usadas.
- Desarrollo de la solución: Capitulo formado por los detalles del desarrollo de la aplicación.
- Conclusiones: Para finalizar, en este capítulo se incluye una conclusión del proyecto y los conocimientos adquiridos junto a una enumeración de futuras mejoras.
- Bibliografía: Numeración de las referencias con su fuente bibliográfica.
- Anexo A: ODS

2. ESTADO DEL ARTE

La esclerosis múltiple necesita un seguimiento personalizado para cada paciente, imprescindible para conocer la evolución de la enfermedad, identificar las necesidades de un tratamiento y valorar la respuesta a este mismo, por lo que es lógico que ya existan alternativas digitales para realizar este seguimiento.

Existen varias aplicaciones móviles en el mercado que ofrecen este asesoramiento para los pacientes de esclerosis múltiple. Varías de ellas son solo aplicaciones informativas y pocas de ellas reciben actualizaciones regulares, proporcionando información desfasada o inútil.

En este apartado se explican las principales alternativas donde destacan Cleo y Emilyn. Ambas aplicaciones disponen de más de 4 sobre 5 de valoración. Estas ofrecen unas funciones similares a la aplicación a desarrollar, con algunas características únicas y otras más deficientes que se pueden mejorar.
Las funciones principales en las que coinciden las competidoras son recabar información del usuario, como el estado de ánimo o medicación, un calendario para el usuario y un desglose de consejos y recomendaciones.

A continuación, se analizan las dos aplicaciones ya mencionadas.

2.1. CLEO

Desarrollada por Biogen, una empresa de Neurociencia, Cleo [3] basa su característica principal en un chat con una enfermera. Aquí puedes mandar tus dudas o preguntas en un chat donde podrás contactar con un enfermero cuyo horario de consulta es de lunes a sábado de 09:00 a 20:00h.
En la última actualización se ha anunciado que el servicio de chat dejara de estar disponible a partir del 31 de agosto de 2022, para centrarse, según la empresa, en otros aspectos más importantes.

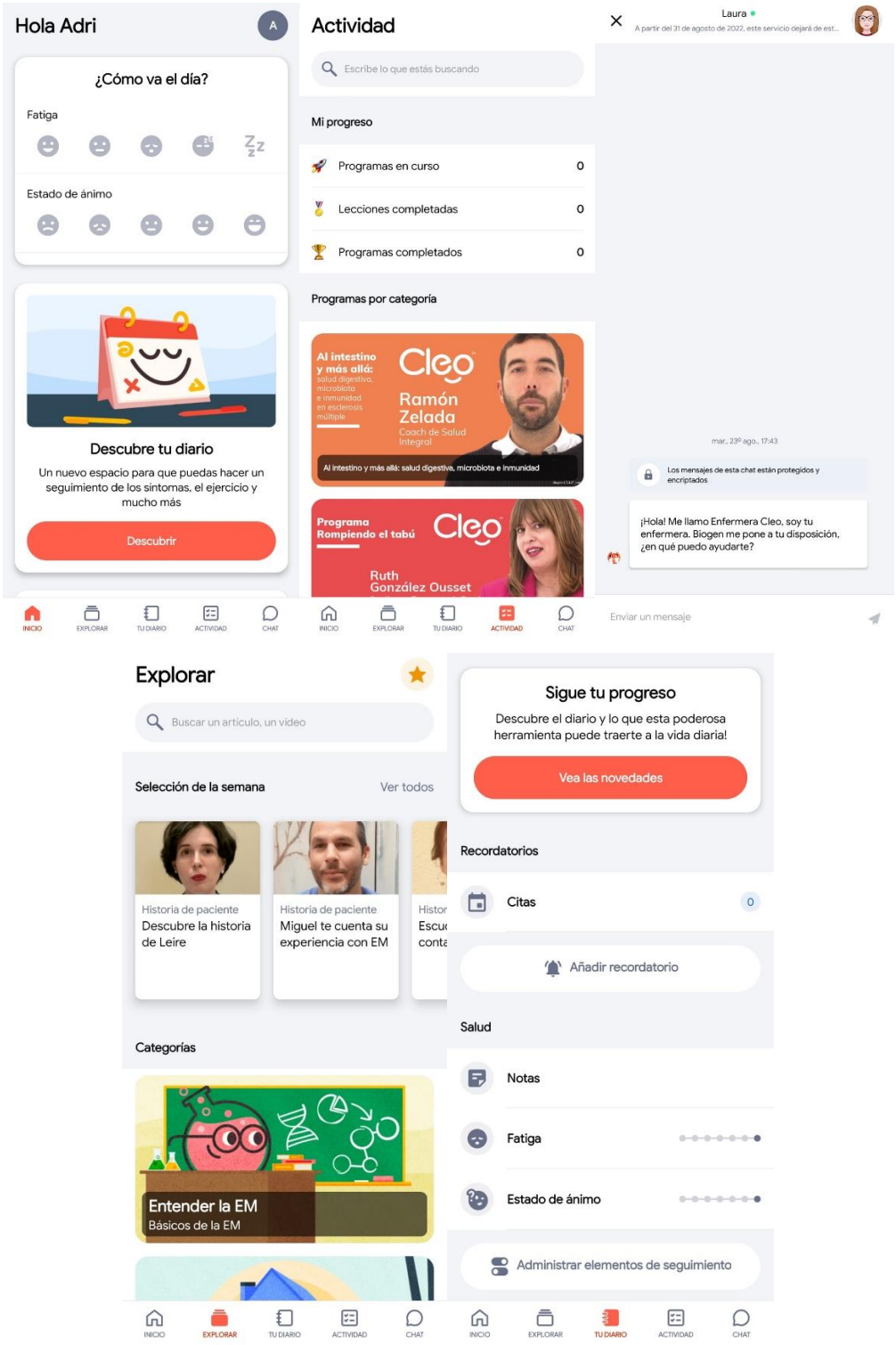

Figura 1 Interfaz de usuario de Cleo

Además de esta función, también ofrece una gestión de citas médicas, una página de programas y videos elaborados por profesionales y otra página de consejos y recomendaciones.

Como se puede observar, su interfaz es clara y directa, pero sus componentes no tanto, distribuyendo por varias vistas lo que podría ser una sola.

2.2. EMILYN

Emilyn [4] ofrece un sistema de monitorización centrado en el seguimiento de síntomas. Junto a esto, ofrece varios cuadros de información, con la medicación tomada o el estado de ánimo. El calendario que ofrece puede llegar a ser bastante confuso.

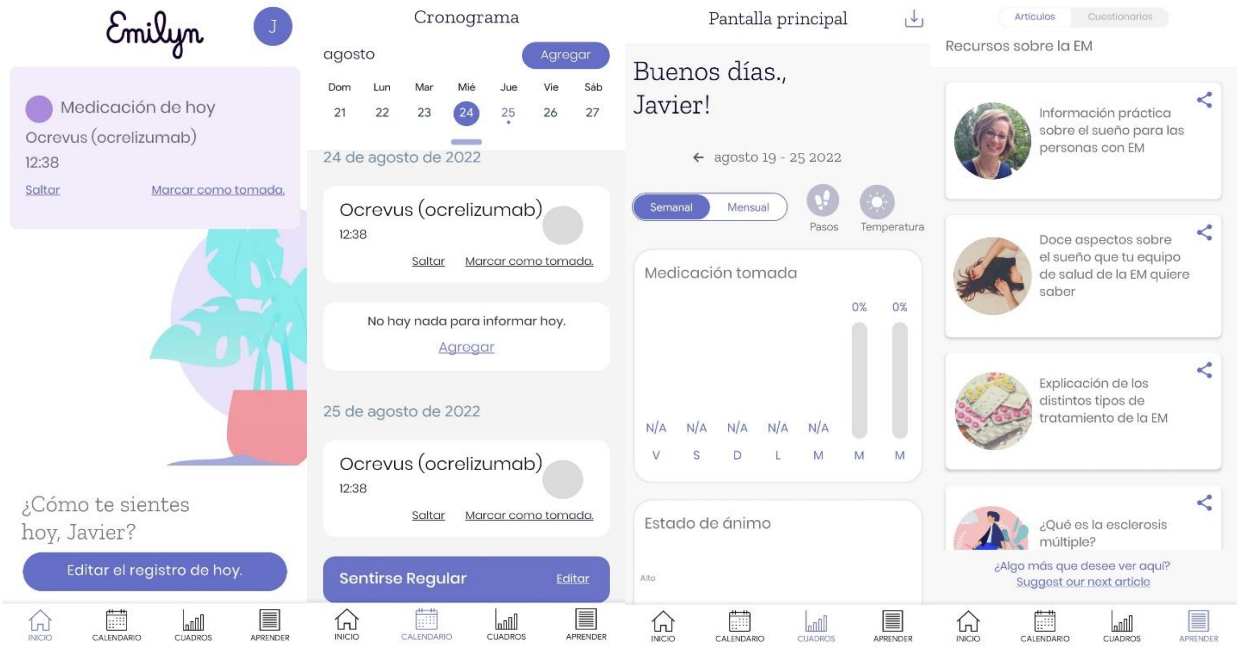

Figura 2 Interfaz de usuario de Emilyn

Cabe destacar la calidad de la información proporcionada en la sección Aprender, donde colabora con expertos para la redacción de artículos de divulgación sobre la enfermedad.

2.3. SOLUCIÓN PROPUESTA

Una vez analizadas las principales aplicaciones basadas en la monitorización de personas diagnosticadas con esclerosis múltiple, se va a exponer la solución propuesta, con algunas funcionalidades similares a las alternativas ya desarrolladas.

Como se va a ver más adelante, la aplicación dispone de inicio de sesión y registro, permitiendo el acceso también mediante los servicios de Google. Una vez inicia la sesión se ofrece al usuario información personalizada, permitiendo crear y gestionar nuevas citas, documentos en formato PDF y su medicación. Además de información sobre la enfermedad y entrenamientos para la mejora física del usuario.

El esquema de colores elegido para la parte gráfica de la aplicación es una paleta de distintos tonos de naranja, ya que este color es el elegido mundialmente para representar la esclerosis múltiple.

Figura 3 Logotipo de la solución

Para el desarrollo de esta aplicación, se ha contado con la ayuda de Carlos Toledo [5], profesional graduado en Ciencias de la Actividad Física y del Deporte y especializado en el campo del Deporte Adaptado. Siendo personal investigador de varios proyectos nacionales, especialmente en entrenamiento para pacientes con Esclerosis Múltiple, y la clasificación de jugadores de fútbol con Parálisis Cerebral a nivel internacional.

El Sr. Toledo ha proporcionado los datos necesarios sobre la enfermedad en lo referente a la actividad física, listando una serie de conocimientos aplicados en el desarrollo de la aplicación y en la redacción de esta memoria. Cabe destacar los distintos entrenamientos especializados, ideados a medida para los enfermos de esclerosis múltiple.

A continuación, se presentan en una tabla las distintas características apreciables durante el análisis de las principales aplicaciones competidoras y de las sugeridas para implementarse en la aplicación escleAPP.

.

	Cleo	Emilyn	EscleAPP
Consejos	X	X	X
Medicación		X	X
Calendario		X	X
Agregar citas	X	X	X
Cuadros de información		X	
Notas	X		
Chat personal	X		
Subir y gestionar documentos		X	X
Registro y creación de usuario	X	X	X
Entrenamientos específicos			X

Tabla 1 Tabla comparativa de características de las aplicaciones

Las aplicaciones analizadas cubren las necesidades básicas para un enfermo de EM, como es el seguimiento de la medicación y eventos, además del registro y creación del usuario. Cada aplicación dispone de una característica única, como es el chat en Cleo y los cuadros de información en Emilyn. Es por ello, que se propone una característica distintiva para la aplicación desarrollada, que pretende centrarse en el asesoramiento de la actividad física, proponiendo distintos entrenamientos adecuados a diferentes tipos de usuarios, dependiendo de la capacidad física.

3. DISEÑO DE LA SOLUCIÓN

3.1. METODOLOGÍA

La metodología durante el desarrollo de software es imprescindible para controlar y organizar un proyecto. Es por ello por lo que durante el aprendizaje en la carrera hemos aprendido distintas metodologías que se adaptan a distintos grupos de trabajo.

La metodología seleccionada para la realización de este proyecto es el modelo incremental [6], que permite el trabajo en fases, añadiendo funcionalidad o mejoras al software. Como el trabajo es individual y no necesita de comunicación efectiva ni división del trabajo, se ha decidido ir añadiendo funcionalidades una a una, con sus respectivas fases de análisis, desarrollo y pruebas de integración.

Se ha basado en una primera versión de la aplicación muy simple pero ejecutable, a la que se han ido añadiendo funciones y en las que se ha podido probar antes de terminar el desarrollo de la herramienta. Este modelo también da facilidades a los cambios imprevistos.

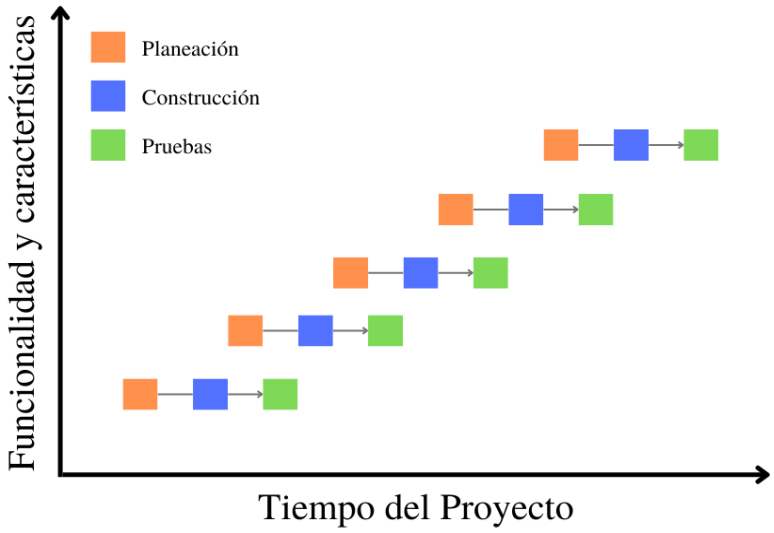

Figura 4 Esquema del modelo incremental usado

Las fases usadas durante el proyecto son una simplificación del modelo incremental para equipos:

- Planeación: Análisis de requisitos, viabilidad, características y funcionalidades.
- Construcción: Desarrollo de la capsula planeada, realizando nuevas funcionalidades o mejoras.
- Pruebas: Realización de pruebas de aceptación y de rendimiento sobre las funciones añadidas.

3.2. ESPECIFICACIÓN DE REQUISITOS

3.2.1. INTRODUCCIÓN

Esta sección conforma la Especificación de Requisitos Software (ERS) como una descripción completa del comportamiento de la aplicación desarrollada. Esta especificación se ha estructurado basándose en el estándar IEEE 830-1998 [7].

3.2.1.1. PROPÓSITO

La Especificación de Requisitos Software tiene como propósito definir las especificaciones funcionales y no funcionales para el desarrollo de una aplicación web que permita monitorizar a las personas diagnosticadas con esclerosis múltiple.

Todos los requerimientos establecidos en este informe deben ser suficientes para poder crear el software.

3.2.1.2. ÁMBITO DEL SISTEMA

La aplicación desarrollada se llama escleAPP, y sus funciones pasan por un inicio de sesión y registro de usuario, gestión de medicación, documentos y citas, y aprendizaje de consejos y actividad física.

3.2.1.3. DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

Nombre	Descripción
Usuario	Persona que va a usar el sistema
ERS	Especificación de Requisitos Software
RF	Requisito Funcional
RNF	Requisito No Funcional

Tabla 2

3.2.1.4. RESUMEN

Este documento consta de tres secciones. En la primera sección se realiza una introducción al mismo y se proporciona una visión general de la especificación de recursos del sistema software.

En la segunda sección del documento se realiza una descripción general del sistema, con el fin de conocer las principales funciones que éste debe realizar, los datos asociados y los factores que afectan al desarrollo.

Por último, la tercera sección del documento es aquella en la que se definen detalladamente los requisitos que debe satisfacer el sistema.

3.2.2. DESCRIPCIÓN GENERAL

3.2.2.1. PERSPECTIVA DEL PRODUCTO

El sistema de monitorización para enfermos de esclerosis múltiple es un producto diseñado para trabajar en entornos web desde smartphones, con la posibilidad de crear un acceso directo en cajón de aplicaciones para poder usarse como una más.

3.2.2.2. FUNCIONALIDAD DEL PRODUCTO

Para poder ver la funcionalidad principal del producto, se ha creado un diagrama de casos de uso, con el usuario como actor principal y donde cada funcionalidad a desarrollar se ha definido como un caso de uso.

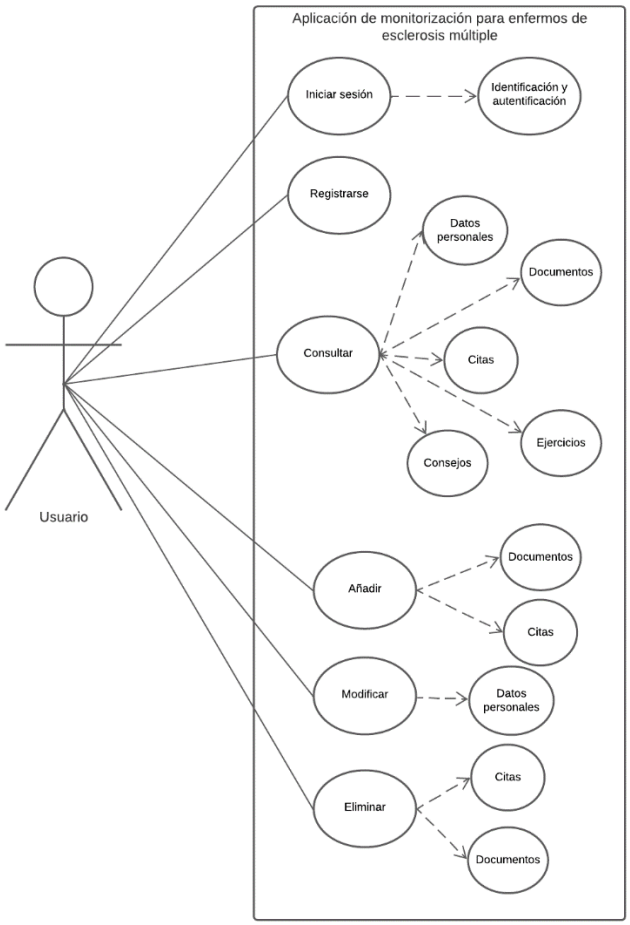

Figura 5 Diagrama de casos de uso

3.2.2.3. CARACTERÍSTICAS DEL USUARIO

Tipo de usuario	Usuario
Formación	Ninguna
Actividades	Interactúa con la aplicación, observa la información y participa activamente en añadir/modificar/eliminar sus datos individuales.

Tabla 3

3.2.2.4. RESTRICCIONES

- Sistema creado para ser usado con internet.
- Sistema creado para ser usado desde un móvil.
- El servidor de base de datos debe ser capaz de atender consultas concurrentes y atender consultas de varios usuarios a la vez.
- Sistema diseñado según un modelo cliente/servidor, por lo tanto, el programa no funciona si no se encuentra conexión con la base de datos.
- El sistema debe tener un diseño e implementación sencilla.

3.2.2.5. SUPOSICIONES Y DEPENDENCIAS

Ahora se abordarán los factores que pueden afectar al funcionamiento del sistema. Para el funcionamiento del sistema es necesario que el servidor en el cual se está trabajando deba contar con una conexión a internet, en caso contrario el programa no funcionará.

3.2.3. REQUISITOS ESPECÍFICOS

La especificación de requisitos específicos ayuda a visualizar cual es el comportamiento esperado de la aplicación al fin de su desarrollo, que se van a dividir en dos categorías, los requisitos funcionales y los requisitos no funcionales.

3.2.3.1. REQUISITOS FUNCIONALES

Los requisitos funcionales son aquellos que describen las características que debe proporcionar el sistema.

Identificación del requisito	RF01
Nombre del Requisito	Registro de usuario
Características	El usuario debe registrarse en el sistema para acceder cualquier parte del sistema.
Descripción del Requisito	El sistema permite al usuario registrarse. El usuario debe indicar los siguientes datos: Nombre, Apellido/s, edad, sexo, email, contraseña. De estos datos solo serán obligatorios el email y la contraseña.
Requisitos NO funcionales	RNF01 RNF02 RNF03 RNF04
Prioridad de requisito	Alta

Tabla 4

Identificación del requisito	RF02
Nombre del Requisito	Identificación de usuario
Características	El usuario debe identificarse si ya está registrado para acceder cualquier parte del sistema.
Descripción del Requisito	El sistema permite al usuario identificarse. El usuario debe indicar los siguientes datos: email y contraseña.
Requisitos NO funcionales	RNF01 RNF02 RNF03 RNF04
Prioridad de requisito	Alta

Tabla 5

Identificación del requisito	RF03
Nombre del Requisito	Consulta de información de usuario
Características	El sistema ofrece al usuario su información personal.
Descripción del Requisito	El sistema permite al usuario ver sus datos personales. Estos son Nombre, Apellido/s, edad, sexo y email.
Requisitos NO funcionales	RNF01 RNF02 RNF04
Prioridad de requisito	Alta

Tabla 6

Identificación del requisito	RF04
Nombre del Requisito	Consulta de consejos
Características	El sistema ofrece al usuario una serie de consejos.
Descripción del Requisito	El sistema permite al usuario ver consejos e información sobre la esclerosis múltiple. Cada uno de los consejos se conformará con un título, una descripción y una imagen.
Requisitos NO funcionales	RNF01 RNF02
Prioridad de requisito	Alta

Tabla 7

Identificación del requisito	RF05
Nombre del Requisito	Consulta de ejercicios físicos.
Características	El sistema ofrece una serie de ejercicios y deportes.
Descripción del Requisito	El sistema permite al usuario ver una lista de ejercicios. Cada uno de los ejercicios se conformará con un título, una descripción y una imagen.
Requisitos NO funcionales	RNF01
Prioridad de requisito	Alta

Tabla 8

Identificación del requisito	RF06
Nombre del Requisito	Añadir documento
Características	El usuario puede añadir un documento médico.
Descripción del Requisito	El sistema permite al usuario añadir un documento, que debe estar en formato PDF.
Requisitos NO funcionales	RNF01 RNF02 RNF03
Prioridad de requisito	Media

Tabla 9

Identificación del requisito	RF07
Nombre del Requisito	Añadir cita
Características	El usuario puede añadir una cita a su calendario.
Descripción del Requisito	El sistema permite al usuario añadir una cita. Una cita está conformada por un título, una descripción y una fecha. Esta fecha no podrá ser anterior a la fecha de creación de la cita.
Requisitos NO funcionales	RNF01 RNF02 RNF03
Prioridad de requisito	Alta

Tabla 10

Identificación del requisito	RF08
Nombre del Requisito	Consulta de calendario
Características	El sistema ofrece al usuario un calendario.
Descripción del Requisito	El sistema permite al usuario ver un calendario donde estarán añadidas sus citas creadas anteriormente.
Requisitos NO funcionales	RNF01 RNF02
Prioridad de requisito	Alta

Tabla 11

Identificación del requisito	RF09
Nombre del Requisito	Consulta documentos.
Características	El usuario puede consultar sus documentos.
Descripción del Requisito	El sistema permite al usuario ver una lista de documentos subidos anteriormente por el mismo usuario.
Requisitos NO funcionales	RNF01 RNF02
Prioridad de requisito	Media

Tabla 12

Identificación del requisito	RF10
Nombre del Requisito	Modificar de datos personales.
Características	El usuario puede editar sus datos personales.
Descripción del Requisito	El sistema permite al usuario editar los datos introducidos durante el registro.
Requisitos NO funcionales	RNF01 RNF03 RNF04
Prioridad de requisito	Media

Tabla 13

Identificación del requisito	RF11
Nombre del Requisito	Eliminar citas.
Características	El usuario puede eliminar las citas.
Descripción del Requisito	El sistema permite al usuario eliminar las citas creadas anteriormente.
Requisitos NO funcionales	RNF01
Prioridad de requisito	Alta

Tabla 14

Identificación del requisito	RF12
Nombre del Requisito	Eliminar documentos.
Características	El usuario puede eliminar las citas.
Descripción del Requisito	El sistema permite al usuario eliminar los documentos subidos anteriormente.
Requisitos NO funcionales	RNF01
Prioridad de requisito	Alta

Tabla 15

3.2.3.2. REQUISITOS NO FUNCIONALES

Estos requisitos se refieren a los requisitos que no tienen relación con la funcionalidad propia del sistema.

Identificación del requisito	RNF01
Nombre del Requisito	Interfaz del sistema
Características	El sistema presenta una interfaz de usuario sencilla para que sea de fácil manejo a los usuarios.
Descripción del Requisito	El sistema debe tener una interfaz de uso intuitiva y sencilla.
Prioridad de requisito	Alta

Tabla 16

Identificación del requisito	RNF02
Nombre del Requisito	Aprendizaje sencillo
Características	El tiempo de aprendizaje debe ser reducido.
Descripción del Requisito	El tiempo de aprendizaje del sistema por un usuario debe ser menor a 4 horas.
Prioridad de requisito	Alta

Tabla 17

Identificación del requisito	RNF03
Nombre del Requisito	Mensajes de error adecuados
Características	Los mensajes de error deben ser claros y concisos.
Descripción del Requisito	El sistema debe proporcionar mensajes de error que sean informativos.
Prioridad de requisito	Media

Tabla 18

Identificación del requisito	RNF04
Nombre del Requisito	Seguridad de la información
Características	El sistema garantiza a los usuarios una seguridad en cuanto a la información sensible.
Descripción del Requisito	Garantiza la seguridad respecto a información y datos tales como documentos, imágenes y contraseñas.
Prioridad de requisito	Alta

Tabla 19

3.3. PROTOTIPADO DE LA IU

Un prototipo es una representación de prueba de un producto, donde se hace tangible una idea para poder ser consultada y validada. El prototipado permite desarrollar diversas opciones de solución y definir el alcance de cara a la propuesta de diseño final del producto.

Para la creación de los prototipos se ha utilizado la herramienta Figma [8], una herramienta de prototipado web y editor de gráficos vectorial.

En la figura 6, se muestran las interfaces de inicio de sesión y registro, ventanas por las que debe navegar un usuario si desea acceder a las demás funcionalidades. El texto debajo de "Ingresar" permite acceder al formulario de registro, donde se introducen los datos principales del usuario.

El campo de "Sexo" es un desplegable que permite elegir el género del usuario entre varias opciones, y por su parte el campo "Tipo de EM" está formado por un selector que proporciona los cuatro tipos distintos de esclerosis múltiple como opciones.

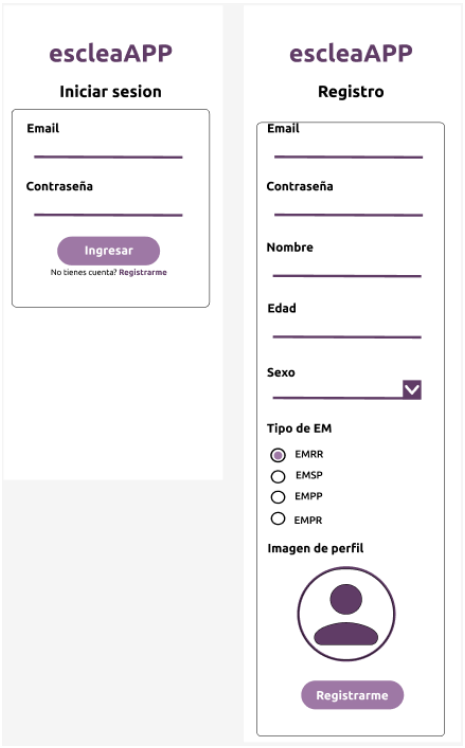

Figura 6 Prototipos de las interfaces de Inicio de sesión y Registro

Tras el inicio de sesión, la siguiente vista que se muestra al usuario es la de su perfil, donde se muestran sus datos personales ya introducidos anteriormente, como la imagen de perfil, el email la edad y el sexo.

Además de esto, también se muestra una lista de medicamentos, que representa la medicación que está tomando el paciente en la actualidad, mostrando su información y permitiendo el borrado de esta información pulsando sobre el botón a la derecha.

Junto a la lista de medicamentos está la lista de documentos, una sección donde se muestran los documentos médicos subidos desde otra parte desde la aplicación y desde donde se permite consultar estos archivos. Además, como con la medicación, el usuario puede eliminar el documento que desee pulsando sobre el botón.

Figura 7 Prototipo de los datos del usuario

Figura 8 Prototipos de las interfaces de Ejercicios y Consejos

Las interfaces de “Consejos” y “Ejercicios” se acceden a través de los botones de la barra de navegación inferior, el segundo y cuarto botón respectivamente.

En la ventana de “Ejercicios” se muestran distintos niveles de entrenamientos, y cada uno de ellos redirige a una lista de ejercicios de dicho nivel, donde cada ejercicio muestra una descripción para la correcta realización de este.

En cuanto a la interfaz de “Consejos”, conforma de una lista de consejos que al pulsar se mostrará un texto, donde se desarrolla el tema del título.

En la figura 9 se aprecian los prototipos de interfaz de “Calendario” y “Añadir elemento”, en el primero, se muestra un calendario donde se pueden apreciar las futuras citas de un usuario, y debajo de este una lista donde se amplía la información de estas, permitiendo también la eliminación de cualquiera de ellas si se desea.

Por otra parte, se tiene la interfaz de “Añadir elemento”, que permite añadir al sistema distintos elementos que se han podido ver en otros prototipos como son las citas, la medicación y los documentos.

Figura 9 Prototipos de las interfaces Calendario y Añadir elemento

3.4. TECNOLOGÍAS

Ahora se exponen las diferentes tecnologías usadas, que se pueden catalogar en tres fases distintas: frontend, backend y base de datos.

Para el frontend, es decir la aplicación principal, se ha usado el Framework de Angular [9], en su versión 12.2, junto a Bootstrap [10] y Firebase [11]. Angular es un framework desarrollado por Google basado en TypeScript, una variante de JavaScript, para crear aplicaciones de una sola página. Usa un patrón Moldeo-Vista-Controlador (MVC) junto al uso de componentes que cambian la vista dinámicamente. Es un framework modular y escalable que se adapta a nuevas necesidades.

Junto a Angular se ha usado Bootstrap, un Framework CSS diseñado para la creación de interfaces limpias y diseño *responsive*, y Firebase, una plataforma en la nube para el desarrollo de aplicaciones web y móvil adquirida por Google que facilita la autentificación, almacenamiento de archivos y análisis de rendimiento.

Figura 10 Logotipo de las tecnologías usadas en el frontend

Para el *backend* se ha desarrollado una API Rest con Node.js [12] y Express.js [13]. Node.js es un entorno de ejecución de JavaScript de código abierto, basado en la gestión de eventos asíncronos. Por su parte, Express.js es un Framework basado en Node, que permite la creación del servidor que conectará con la base de datos.

Figura 11 Logotipos de las tecnologías usadas en el backend

En el almacenamiento de datos se ha utilizado MySQL [14], un sistema de gestión de bases de datos relacional. Junto a este, se ha usado MySQL Workbench [15], un entorno grafico distribuido por Oracle que facilita la gestión de las bases de datos creadas con MySQL.

Figura 12 Logotipos de las tecnologías usadas para la gestión de la base de datos

Además, se ha utilizado Postman para crear peticiones a la API Rest que rescaten los datos. Esto ha permitido hacer pruebas sobre la implementación de la API Rest y de la base de datos, ejecutando llamadas GET, PUT, POST y DELETE.

4. DESARROLLO DE LA SOLUCIÓN

4.1. BASE DE DATOS

Se ha creado una base de datos relacionas bastante sencilla en MySQL, compuesta de tablas simples con un ID como clave primaria. Todas las relaciones de esta base de datos son muchos a muchos, por ejemplo, muchos usuarios pueden tener recetada en su medicación muchos medicamentos y este mismo medicamento puede estar en la receta de muchos usuarios.

Para este tipo de relación se debe crear una tabla asociativa, con un ID como clave primaria y el ID del usuario y el ID de la medicación como claves ajenas.

Con esta tabla asociativa podemos obtener luego la medicación de un usuario, filtrando la lista de relaciones usuario_medicación por el ID del usuario.

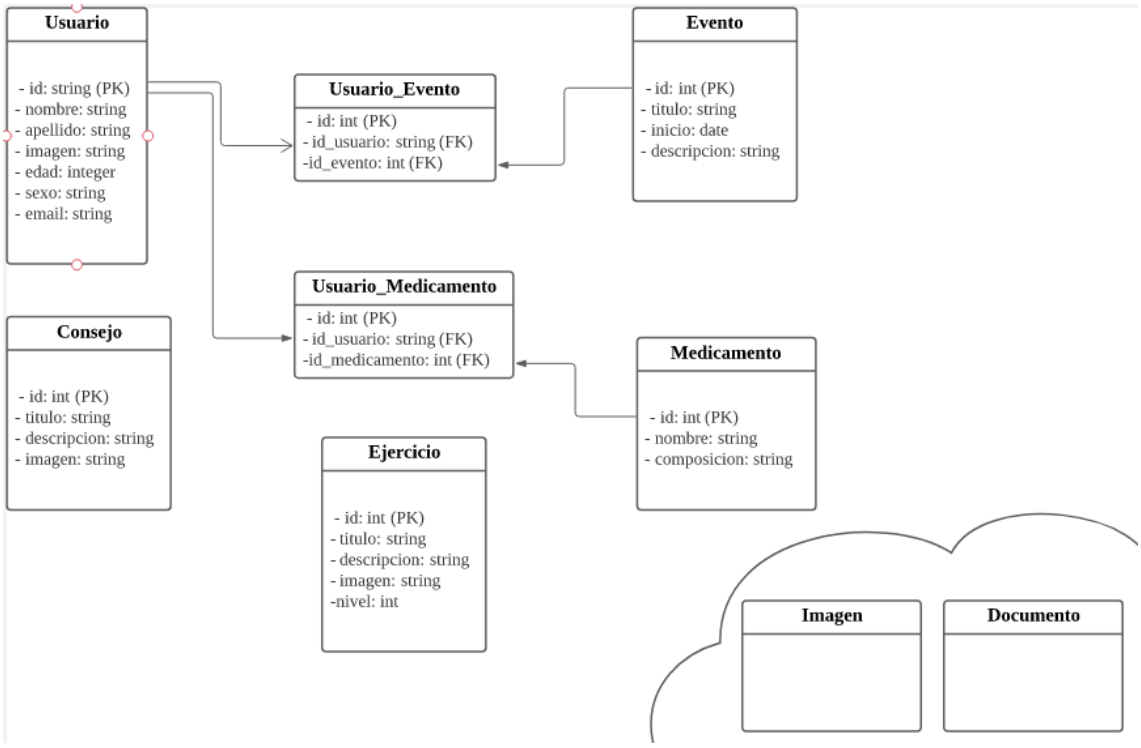

Figura 13 Esquema básico de la base de datos

Además, también se han creado procedimientos almacenados. Un procedimiento almacenado MySQL no es más que una porción de código que se puede guardar y reutilizar posteriormente, con intención de usarse en la API Rest que se va a implementar a continuación.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `usersAddOrEdit`(
 IN _id VARCHAR(30),
 IN _nombre VARCHAR(45),
 IN _apellido VARCHAR(45),
 IN _imagen VARCHAR(200),
 IN _edad INT(11),
 IN _sexo VARCHAR(45),
 IN _email VARCHAR(45)

)
BEGIN
 IF NOT EXISTS (SELECT * FROM users WHERE id = _id) THEN
  INSERT INTO users (id, nombre, apellido, imagen, edad, sexo, email)
  VALUES (_id, _nombre, _apellido, _imagen, _edad, _sexo, _email);

  SET _id = LAST_INSERT_ID();
 ELSE
  UPDATE users
  SET
  nombre = _nombre,
  apellido = _apellido,
  imagen = _imagen,
  edad = _edad,
  sexo = _sexo,
  email = _email
  WHERE id = _id;
 END IF;
 SELECT _id AS id;
END$$
```

Figura 14 Ejemplo de un procedimiento almacenado, usersAddOrEdit

Estos procedimientos, uno por cada tabla creada, se encargan de insertar un nuevo dato en la tabla indicada, o bien actualizarlo si ya existe una fila con el mismo ID.

4.2. API REST

La API Rest se ha creado con Node.js y Express.js. Una API Rest [16] es un conjunto de reglas que definen como pueden las aplicaciones conectarse y comunicarse entre sí cumpliendo los principios de diseño de la arquitectura REST.

La arquitectura REST se basa en un cliente enviando peticiones para recuperar o modificar recursos, y el servidor respondiendo con el resultado, que puede ser con los datos que hemos pedido o el estado de la petición.

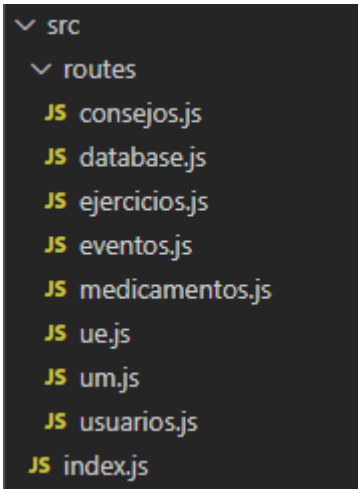

Figura 15 Estructura de api Rest

La estructura de nuestra API Rest es la siguiente, donde se puede apreciar la carpeta de *routes* , que contiene los archivos JavaScript que se encargan de realizar las peticiones HTTP. Debe haber un .js por cada tabla de nuestra base de datos

Además, también habrá un archivo index.js que contiene la configuración de rutas, middleware e inicio de servidor.

Con este fragmento de código definido en index.js se otorgan los permisos para acceder a los recursos del servidor. CORS es un mecanismo que utiliza cabeceras HTTP adicionales para permitir un intercambio de recursos cruzado. Es por esto por lo que se debe otorgar los permisos permitiendo el acceso a todas las peticiones.

```
// Middlewares
app.use(cors({
    origin: '*'
}));
app.use(express.json());
```

Figura 16 Permisos de CORS

4.3. FRONTEND

Para el desarrollo del frontend se ha usado el framework de angular junto a módulos de Bootstrap y Angular Material. La arquitectura de un proyecto Angular es sencilla, dividiendo nuestro frontend en Componentes, Modelos y Servicios.

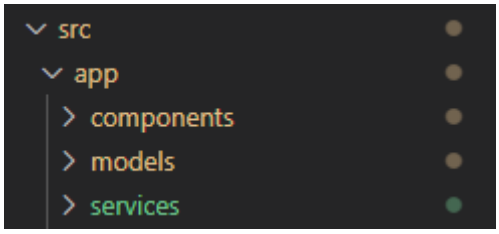

Figura 17 Estructura principal del proyecto

Los **componentes** son fragmentos de código que definen una clase que contiene los datos y la lógica de la aplicación. Cada componente define una parte de la interfaz de usuario y está compuesto por un archivo HTML que define la vista, un archivo TypeScript, que se encarga de la lógica del componente, y un archivo CSS que define el estilo.

Cada componente puede definir cada vista del sistema, o bien desde un mismo componente se puede conformar de llamadas a otros componentes. Así pues, esta aplicación funciona ejecutando dos componentes a la vez, en la parte superior la página en la que deseamos estar, y que es cambiante según las acciones del usuario; y una barra de navegación que será fija excepto en contadas ocasiones.

```
<div class="container">
  <router-outlet></router-outlet>
</div>
  <app-navegation></app-navegation>
```

Figura 18 HTML del componente principal

En la Figura 13 se puede ver el código que define la ejecución de estos dos componentes: la ruta y la navegación. Estas rutas se definen en el archivo app-routing.module.ts.

Para esta aplicación se ha definido un único **modelo**, Modelo.ts y que estará en la carpeta *app/models*. En este modelo se exportarán las interfaces que definen nuestros objetos con propiedades específicas que se recuperan de la base de datos, como por ejemplo la interfaz Help o Ejercicio.

```
export interface Help {
    id: number;
    title: string;
    description: string;
    image: string
}

You, hace 1 minuto | 1 author (You)
export interface Ejercicio {
  id: number;
  nombre: string;
  descripcion: string;
  imagen: string;
  nivel: number
}
```

Figura 19 Interfaces de Help y Ejercicio en Modelo.ts

Y, por último, dentro de *app/services* se almacenan los **servicios** que sirven para compartir información entre componentes o hacer peticiones a la API Rest o APIs externas.

En esta aplicación hay dos tipos de servicios, los que mandan o reciben información a través de la API Rest a la base de datos, o los que conectan con Firebase para ejecutar diferentes servicios ofrecidos por Google.

```
export class EjerciciosService {


API_URI = 'http://localhost:3000';

constructor(private http: HttpClient) {}

  getEjercicios() {
    return this.http.get(`${this.API_URI}/ejercicios`);
  }

  getEjercicio(id: string) {
    return this.http.get(`${this.API_URI}/ejercicios${id}`);
  }

  saveEjercicio(help: Help) {
    return this.http.post(`${this.API_URI}/ejercicios`, help);
  }

  deleteEjericicio(id: string) {
    return this.http.delete(`${this.API_URI}/ejercicios/${id}`);
  }

  updateEjercicio(id: string, ejercicio: Ejercicio) {
    return this.http.put(`${this.API_URI}/ejercicios/${id}`, ejercicio);
  }
}
```

Figura 20 Código del servicio EjerciciosService

En la figura 14 se aprecia la clase que se encarga de enviar las peticiones GET, POST, PUT o DELETE a la API Rest creada anteriormente con relación a los ejercicios de actividad física, y que se puede llamar desde los componentes que deseen tratar con los datos de la clase Ejercicio, definido en el modelo.

Por otra parte, están los servicios que conectan con Firebase. Firebase facilita la función de desarrollar y crear aplicaciones para móvil, conteniendo diversas funciones para que cualquier desarrollador pueda combinar y adaptar la plataforma a medida de sus necesidades. En este caso, se van a usar los servicios de autentificación y de almacenamiento en la nube.

La mayoría de las aplicaciones necesitan identificar a los usuarios, permitiendo guardar sus datos en la nube de forma segura.

```
export class AuthService {

  constructor(private auth:AngularFireAuth) { }

  async register(email:string, password:string) {
    try {
      return await this.auth.createUserWithEmailAndPassword(email, password);
    } catch(err) {
      console.log(err);
      return null;
    }
  }

  async login(email:string, password:string) {
    try {
      return await this.auth.signInWithEmailAndPassword(email, password);
    } catch(err) {
      console.log(err);
      return null;
    }
  }

  async loginGoogle(email:string, password:string) {
    try {
      return await this.auth.signInWithPopup(new firebase.auth.GoogleAuthProvider());
    } catch(err) {
      console.log(err);
      return null;
    }
  }
```

Figura 21 Código del servicio AuthService

En la figura superior se puede apreciar como definiendo la constante *auth* en el constructor se pueden crear los métodos de registro, inicio de sesión, tanto por nombre como por Google y también otras como cerrar sesión y obtener datos del usuario que ha iniciado sesión.

Para el servicio de almacenamiento en Firebase se ha construido una clase similar a la vista y se entrará en detalle más adelante en este documento.

Además de la carpeta app y el archivo app-routing.module.ts nombrado anteriormente hay también una hoja de estilos CSS global para la aplicación entera y el módulo app.module.js que proporciona el mecanismo de arranque para iniciar la aplicación.

En cuanto al estilo de la interfaz gráfica se han instalado las bibliotecas de Bootstrap y Angular Material. Bootstrap es un framework desarrollado por Twitter que combina CSS y JavaScript para estilizar los elementos de una página HTML, proporcionando interactividad con la página facilitando la comunicación con el usuario. Bootstrap funciona con plantillas instaladas a través de una biblioteca. Para instalar Bootstrap en el proyecto Angular hay que escribir por consola sobre la carpeta frontend la siguiente línea de comando:

```
npm install bootstrap
```

Y tras una instalación personalizada según el usuario, podremos usar las plantillas personalizadas que se pueden consultar en la documentación en línea en la página web de Bootstrap.

Angular material es un framework desarrollado en TypeScript por y para Angular, que permite implementar componentes Angular con un diseño basado en Material Design. Para instalar Angular Material en el proyecto Angular hay que escribir por consola sobre la carpeta frontend la siguiente línea de comando:

```
ng add @angular/material
```

4.3.1. INICIO DE SESIÓN

Para la vista de inicio de sesión se ha seguido un modelo típico, con un campo de email y otro de contraseña.

Estos datos privados del usuario son completamente inaccesibles y ya que los datos son tratados mediante Firebase, Google se compromete a proteger la confidencialidad y la seguridad de los datos mediante las leyes que protegen la privacidad del usuario, como el Reglamento General de Protección de Datos del Espacio Económico Europeo [17].

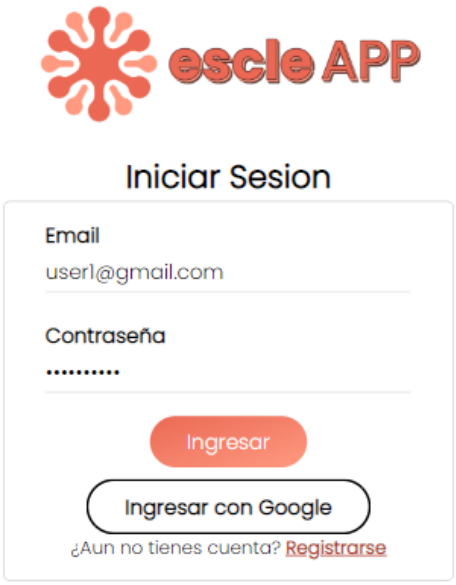

Figura 22 Interfaz de inicio de sesión

El sistema ofrecerá dos alternativas para el inicio de sesión, bien por mail y contraseña, o usando los servicios de Google. Además, desde aquí se puede acceder al registro, clicando sobre el link "Registrarse".

El HTML creado para este componente *login* es bastante sencillo y no se va a entrar en detalle ya que más adelante se explican las etiquetas y conocimientos necesarios en otros archivos HTML más complejos para así evitar la reiteración. Solo he de destacar que al pulsar sobre los botones de "Ingresar" e "Ingresar con Google" se ejecutan los métodos que se han definido en el archivo login.component.ts Login() y LoginGoogle() respectivamente.

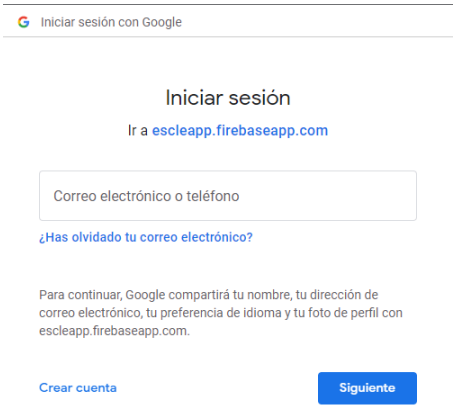

Figura 23 Dialogo que aparece al pulsar "Ingresar con Google"

En un archivo TypeScript de un componente se definen las clases y métodos que se usan desde el HTML. Así pues, en el archivo login.component.ts se define la clase principal LoginComponent, que está compuesto de un método constructor, el método inicializador y los métodos Login() y LoginGoogle() que se usan en los botones.

```
constructor(private authService:AuthService, private router : Router) {}
```

Figura 24 Método constructor de LoginComponent

Por norma general, el método constructor de los componentes Angular está vacío, y solo se usa para llamar a los servicios que son necesarios en los demás métodos. Por su parte, el método ngOnInit() debe estar en todos los componentes, bien vacío, o bien con código que se ejecuta cada vez que se carga en componente.

En Angular se ejecuta antes el constructor que el inicializador, y se usa el primero para inicializar variables y el segundo para inicializar o ejecutar tareas que tienen que ver con Angular.

Para el caso del componente *login*, se usa este inicializador para hacer las llamadas necesarias al servicio de autentificación de Firebase, AuthService. Este servicio está definido en la carpeta services, como se puede apreciar en la Figura 15.

```
ngOnInit(): void {
  try {
    this.authService.getUser().subscribe(res => {
      this.id = res?.uid!;
      console.log(this.id)
      this.usuario.email = res?.email!;
    });
  } catch(err) {
    console.log(err);
  }
}
```

Figura 25 Código del método ngOnInit de LoginComponent

En Firebase, los datos almacenados por usuario son el email registrado, un identificador y la fecha de creación y de acceso. En ngOnInit() se ejecuta este servicio para obtener si se han persistido datos en la aplicación. Grosso modo, esto sirve para si un usuario cierra sesión, tener el campo de *email* autocompletado en esta vista.

```
Login() {
  const {email, password} = this.usuario;
  this.authService.login(email, password).then(res => {
    console.log("se logeo " + res);

    setTimeout(() => {
      window.location.href = "http://localhost:4200/userdata";
      }, 1000)
  })

}
```

```
LoginGoogle() {
  try {
    const {email, password} = this.usuario;
    this.authService.loginGoogle(email, password).then(res => {
      console.log("se logeo con Google " + res);

      setTimeout(() => {
        window.location.href = "http://localhost:4200/userdata";
        }, 1000)
    })
    .catch(error => {
      console.log('Error - ' + error.message)
    })
  } catch (error) {
    console.log('Error - ' + error)
  }

}
```

Figura 26 Código de los métodos Login() y LoginGoogle()

Los métodos de inicio de sesión son muy similares, pero solo se va a entrar en detalle en uno de ellos, el más básico, el método Login(). En primer lugar, se crean dos constantes, *email* y *password* y se les asignan los valores de los campos de *this.usuario*. En this.usuario es donde anteriormente se señalan los valores en el inicializador y de donde se lee y se escribe en el formulario.

Tras esto, se llama a la función de *login* de AuthService que comprueba en Firebase si existe un usuario con los datos previstos. Después, y tras margar un tiempo de espera de un usuario, la aplicación navega a la página de *datos personales*.

La única variación con respecto al método LoginGoogle() es al método que se llama desde AuthService, siendo en este caso el método *loginGoogle()*.

También cabe destacar que Firebase mismo se encarga de comprobar si el usuario y contraseña coinciden, capturando la excepción y mostrándola por consola. En un futuro se pretende recoger estas excepciones y mostrarlas al usuario acordemente.

4.3.2. REGISTRO

En caso de querer registrar un nuevo usuario en el sistema se debe navegar a la ventana de registro clicando sobre el link "Registrarse" desde el inicio de sesión. Este registro se llevará a cabo a través de dos componentes y vistas. En una siguiente versión del producto se prevé disponer de una solicitud de consentimiento en esta misma interfaz que provea de información de para que serán utilizados los datos privados del usuario, como mail, contraseñas y nombre, con el fin de cumplir con la Ley Orgánica de Protección de Datos [18].

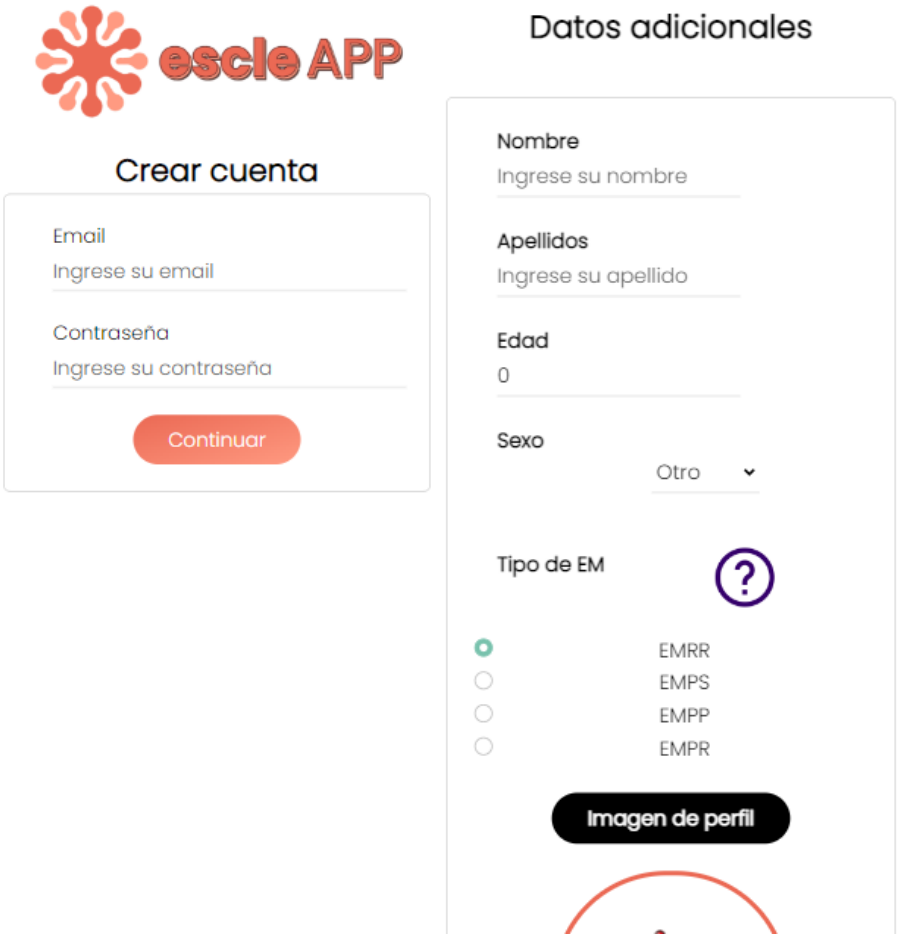

Figura 27 Interfaz de usuario de registro

Estos dos componentes son nombrados *registro* y *datos adicionales*. La primera vista se encarga de comprobar que no existe un usuario en Firebase y crearlo según el email y contraseña

ingresados. Una vez creado Firebase generará un ID para ese usuario que se inserta en la base de datos.

El segundo componente añade y amplía los datos necesarios para mejorar la información que se necesita del usuario. Estos campos son los de nombre, apellidos, edad, sexo, tipo de EM y una imagen de perfil.

El código HTML y TypeScript del primer componente de registro es muy sencillo y similar al ya explicado en el punto 3.3.1 con la diferencia de que se llama al método *register* del servicio de autentificación de Firebase. Por su parte, si se va a ahondar en el contenido de código del segundo componente de datos adicionales.

En primer lugar, se debe dejar claro el concepto de **directivas**, una serie de elementos que se aplican sobre el código HTML con el fin de añadir nuevas funcionalidades a las etiquetas. Principalmente se hace uso de *ngModel*, una directiva de atributo que implementa *binding* o enlace, entre los datos del archivo TypeScript y el HTML del componente.

```
<input class="input-field" type="nombre" placeholder="Ingrese su nombre" [(ngModel)] = "usuario.nombre">
```

Figura 28 Ejemplo de ngModel

De esta forma, al editar el campo de texto, se está editando directamente el nombre del usuario registrado. Este usuario se recupera como se ha visto anteriormente mediante Firebase, recuperando el ID del usuario y simplificando la actualización de datos personales.

```
updateData() {

  this.usuarioService.updateUsuario(this.usuario.id, this.usuario).subscribe(
    res => {
      console.log("Usuario guardado " + res);
    },
    err => console.log(err)
    );
}
```

Figura 29 Método updateData() de AditionaldataComponent

Esta actualización se hace mediante el método updateData() definido en el componente de datos adicionales. Para ello se hace uso del servicio que conecta con la tabla de usuarios de la base de datos y se manda una petición PUT con los datos ingresados en el formulario.

También se debe destacar cómo se realiza la subida de archivos, en el caso del registro, de una imagen local. Para ello se ha hecho uso del anteriormente mencionado sistema de almacenamiento en la nube de Firebase.

El método cargarImagen() usa el servicio *storage.service* ya creado llamando a la función subirImagen(). Esta función recibe como parámetros una cadena y un archivo, devolviendo una URL con la imagen almacenada en el sistema de almacenamiento de Firebase.

```
async subirImagen(nombre:string, imgBase64:any) {
  try {
    let respuesta = await this.storage.child("users/" + nombre).putString(imgBase64, 'data_url');
    console.log(respuesta)
    return  await respuesta.ref.getDownloadURL();
  } catch(err) {
      console.log(err);
      return null;
  }

}
```

Figura 30 Método subirImagen() de la clase StorageService

Esta URL se guarda en campo de imagen del usuario en la base de datos como una cadena de caracteres que funciona como un link a internet que se puede mostrar en el HTML.

4.3.3. DATOS PERSONALES

Una vez que se ha iniciado sesión o registrado un nuevo usuario se puede acceder a la ventana de información personal. Aquí se aprecian los datos introducidos durante el registro, así como la lista de documentos que se han subido o nuestra medicación. Para añadir estos elementos se debe de hacer desde la ventana de Añadir elementos como se ve en el apartado 4.3.7.

Figura 31 Interfaz de usuario de los datos personales

En este componente entra en juego el uso de gran cantidad de servicios, definidos en el constructor, y la mayoría de ellos usados en el inicializador ngOnInit(). Durante la inicialización del componente se hacen varias peticiones GET, con sobreescritura de variables. De esta forma se hace uso durante la inicialización de los servicios de autentificación de Firebase, y de los servicios que conectan con base de datos, en este caso UsuarioService, MedicamentoService, User_medicamentoService y DocumentoService.

```
ngOnInit(): void {
  try {
    this.auth.getUser().subscribe(res => {
      this.usuario.id = res?.uid!;
      this.idBusqueda = res?.uid!;
      console.log(this.idBusqueda);
      this.usuarioService.getUsuario(this.idBusqueda).subscribe(
        res => {
          this.usuario = res;
```

Figura 32 Código de la obtención de usuario en UserdataComponent

Mediante AuthService y UsuarioService se obtienen los datos del usuario que ha iniciado sesión, mostrándolos en la interfaz por el HTML.

En cuanto a la obtención de la lista de medicación y documentos, los pasos a realizar son bastante parecidos, con la diferencia de que para obtener la lista de medicamentos de un usuario se hace uso de dos servicios distintos, MedicamentoService y User_medicamentoService, mientras que para obtener la lista de documentos solo es necesario un servicio, DocumentoService.

```
43          this.medicamentoService.getMedicamentos().subscribe(
44            medicals => {
45              this.medicamentos = medicals;
46              this.user_medicamentoService.getUser_medicamentos().subscribe(
47                userm => {
48                  this.um = userm;
49                  this.um.forEach((m: { id_user: string; }) => {
50                    if(m.id_user == this.idBusqueda) {
51                      this.uma.push(m);
52                    }
53                  },
54
55                  console.log(this.uma),
56                  setTimeout(() => {
57                    this.uma.forEach((ma: { id_medicamento: any; }) => {
58                    this.medicamentos.forEach((m: { id: any; }) => {
59                      if(ma.id_medicamento == m.id) {
60                        this.umedicin.push(m)
61                      }
62                    });
63                  })
64                  }, 500)
65                  );
```

Figura 33 Código usado para obtener la lista de medicamentos de un usuario

Para poder obtener la lista de medicamentos de un usuario, serán necesario definir cuatro variables distintas. Dos de estas variables, *um* y *uma*, almacenarán listas de objetos de tipo *User_medicamento*. Hay que recordar que User_medicamento es una de las tablas intermedias definidas en la base de datos, que permite relacionar muchos a muchos los usuarios con los medicamentos.

Um almacena la primera lista de User_medicamento, que se conforma de todos los User_medicamento almacenados en base de datos, esta petición se puede consultar en la línea 46 de la figura 29. Sobre esta lista se filtran los user_medicamento cuyo *user_id* sea el del usuario que tiene la sesión iniciada y se guardan en la variable *uma* , como se aprecia en la línea 49 de la figura 29.

Una vez que se han obtenido todos los User_medicamento pertenecientes al usuario y almacenados en uma, se obtiene de la lista que almacena todos los medicamentos los medicamentos cuyo ID y el *id_medicamento* de *uma*. Esta búsqueda se puede consultar en la línea 57 de la figura 29, y está conformada por un bucle doble que recorren las variables *uma* y *medicamentos*, que en el caso de coincidir las IDs, se guarda ese medicamento en la lista final *umedicin*, que debe tener al final del método los medicamentos pertenecientes a un usuario.

```
<div class="card-body p-4">
  <label class="float:left" type="documentos" >Medicación:</label>
  <li *ngFor="let m of umedicin" >
    <div class="card">
      <div class="card-body" style="display: flex;">
        <div style=" flex: 5;">
          <h5 class="card-title">{{m.nombre}}</h5>
        <p class="card-text">{{m.composicion}}</p>
        </div>
        <div style="flex: 0.5; display: flex; align-items: center;">
          <a class="button-62"  style="padding: 0 1.4rem" (click)="eliminarMedicina(m)">x</a>
        </div>

      </div>
    </div>
  </li>
</div>
```

Figura 34 Código HTML que muestra la lista de medicamentos de un usuario

Para mostrar esta lista, se hace uso de la directiva *ngFor, esta directiva permite generar varios elementos HTML repetidos a partir del recorrido de una lista de datos. En este caso, se recorre la lista de medicinas del usuario, umedicin, y muestra el título, la composición y un botón que permite borrar este medicamento de la lista del usuario mediante el método eliminarMedicina().

```
eliminarMedicina(event: any) {
  console.log(event);
  this.user_medicamentoService.getUser_medicamentos().subscribe(res => {
    this.dum = res;
    this.dum.forEach((element: { id_user: string; id_medicamento: any; id: number; }) => {
      if(element.id_user == this.idBusqueda && element.id_medicamento == event.id) {
        this.user_medicamentoService.deleteUser_medicamento(element.id).subscribe(result => {
          console.log(result)
        })
      }
    });
  })
```

Figura 35 Código del método eliminarMedicina() en UserdataComponent

El método eliminarMedicina(), activado al pulsar sobe la cruz en la lista de medicinas, es el encargado de borrar de base de datos la relación entre el usuario y el medicamento. Para ello se guarda en la variable *dum* una lista de todas las relaciones User_medicamento y se realiza un recorrido donde si el *id_user* coincide con el ID del usuario que tiene la sesión iniciada y el *id_medicamento* coincide con el ID del medicamento seleccionado, se lanza una petición DELETE a la base de datos. El ID del medicamento se obtiene mediante el *ngFor en el HTML, como se aprecia en la figura 30.

Para el caso de los documentos, los pasos seguidos son muy similares, con la diferencia de que no se precisa ningún servicio de la tabla relacional, ya que un *documento* en base de datos está compuesto por todos los datos necesarios, id, titulo, URL e id_user.

4.3.4. CONSEJOS

La interfaz de consejos es bastante sencilla, esta facilita información al usuario sobre la enfermedad, divulgando conocimientos para la mejora física y psicológica del paciente.

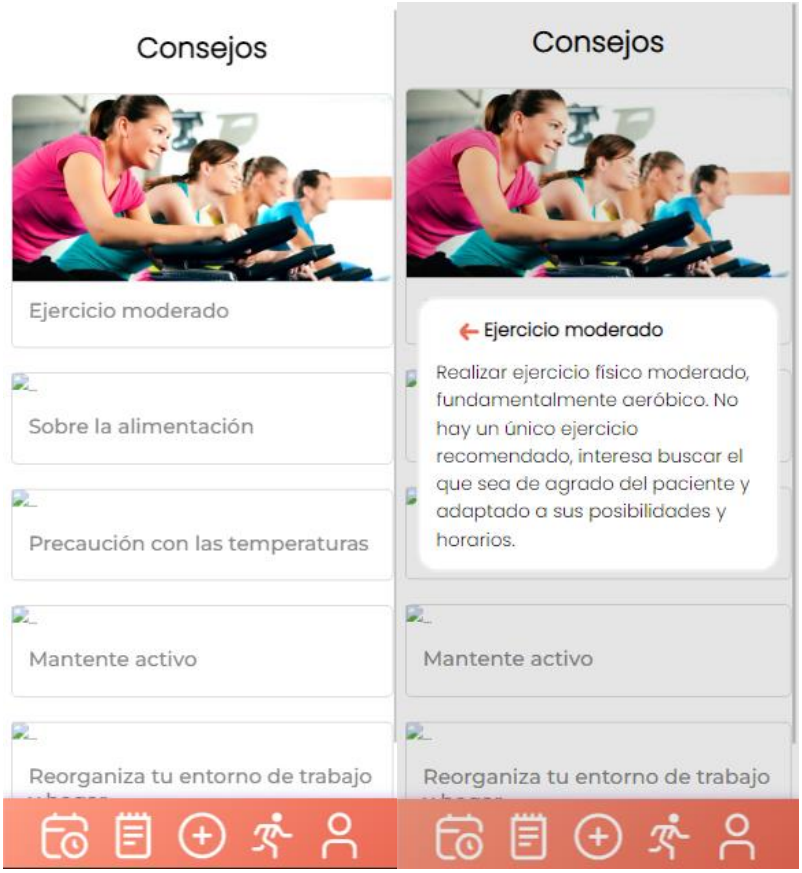

Figura 36 Interfaz de usuario Consejos

Como se aprecia en la figura 37, la información se ofrece en forma de lista de elementos, en la que cada elemento estará formado por el titulo y una imagen que lo acompañan. Si pulsamos sobre cualquier elemento se mostrará un dialogo o ventana modal con más información sobre el mismo.

```
onclick="window.modal.showModal();"
```

Figura 37 Código HTML que muestra el diálogo

Un dialogo es una etiqueta básica de HTML, que se debe definir a parte de los elementos que conforman la vista del componente. Este dialogo se muestra cuando se pulsa, en este caso, sobre una tarjeta de consejo. El dialogo muestra tanto el titulo como la descripción del consejo seleccionado, como esta información es dinámica y varía según que tarjeta se selecciona, se deben definir en el componente dos variables para ello.

Una vez definidas estas variables, se editarán mediante el método cambiarDescripcion(), que pasa como elementos el titulo y la descripción del consejo y sobrescriben las variables que se muestran en el dialogo.

```
<li *ngFor="let h of help">
  <div class="card" (click)="cambiarDescripcion(h.title, h.description)"
  onclick="window.modal.showModal();"  >      You, hace 1 segundo • Unco
    <img src="{{h.image}}" class="card-img-top" alt="...">
    <div class="card-body">
      <h5 class="card-title">{{h.title}}</h5>
    </div>
  </div>
```

Figura 38 Código completo de cada consejo en el HTML

4.3.5. EJERCICIOS

La interfaz de ejercicios es muy similar a la de consejos, ya que solo se busca recuperar datos, y no se debe hacer ninguna modificación sobre los mismos. En este apartado se muestra, en un primer componente, cinco tipos distintos de entrenamiento. Cada uno de ellos es un entrenamiento personalizado, creado por un profesional de la salud física y que se adaptan a las necesidades de un enfermo diagnosticado en esclerosis múltiple. Estos cinco entrenamientos están numerados según el nivel de intensidad de este.

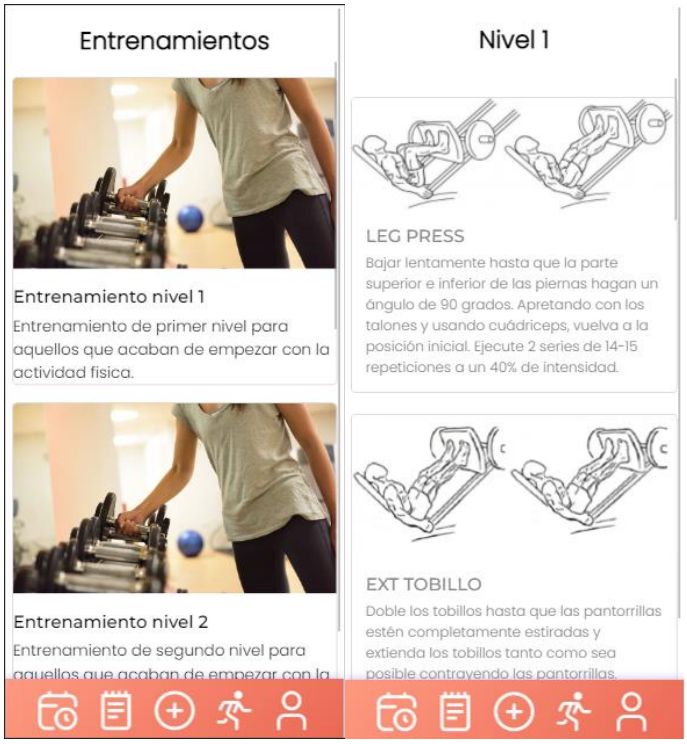

Figura 39 Interfaz de usuario de la ventana de ejercicios

Una vez elegido un entrenamiento, la aplicación navega a otro componente para mostrar la lista de ejercicios, con una descripción y una imagen demostrativa para facilitar la comprensión.

La lógica detrás de este componente es sencilla, haciendo solo uso del inicializador de la clase y del servicio EjerciciosServices, mediante el cual se recupera la lista de ejercicios de base de datos. Cada ejercicio tiene un atributo que lo asocia a un nivel de intensidad, y es a través de este atributo con el que se compara con cada uno de los cinco niveles que se desee mostrar.

```
ngOnInit(): void {
  this.ejericicosService.getEjercicios().subscribe(
    res => {
      this.help = res;
      this.help.forEach((h: { nivel: number; }) => {
        if(h.nivel == 1) {
          this.ejercicios.push(h);
        }

      });
    },
    err => console.log(err)
  )
}
```

Figura 40 Código del método ngOnInit() de NivelunoComponent

Por ejemplo, para mostrar los ejercicios de nivel uno, se recorre la lista de ejercicios y si el elemento pertenece al primer nivel, se almacena en la variable *ejercicios* que se muestra en el HTML.

4.3.6. CALENDARIO

La ventana de Calendario se compone de la lista de eventos del usuario que tiene iniciada la sesión y de un calendario per se, que muestra de manera gráfica la fecha de esta misma lista. Esta lista sigue el mismo patrón que el visto en la lista de medicación o de documentos, permitiendo eliminar un evento de un usuario pulsando sobre el botón a la derecha del elemento.

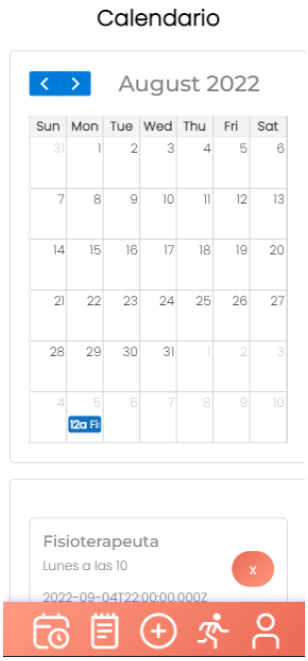

Figura 41 Interfaz de usuario de Calendario

Para mostrar el calendario se ha hecho uso de FullCalendar [19], un plugin jQuery que permite el uso de un calendario completo con una simple importación. Toda la información se puede conseguir desde la documentación referenciada.

```
this.options = {
  plugins: [dayGridPlugin, timeGridPlugin, interactionPlugin],
  defaultDate: new Date(),
  header: {
    left: 'prev,next',
    center: 'title',
    right: ''
  },
  body: {
    height: 'auto'
  },
  editable: false,
  height: 'auto'
};
```

Figura 42 Código de las opciones de FullCalendar

FullCalendar hace uso de una lista de objetos, definida en el componente como *options*, y que permite la customización del plugin. Además, necesita de una lista de eventos que se debe mostrar en el calendario.

Esta lista de eventos se calcula en el método inicializador, que con el ID del usuario registrado y una lista de todos los eventos registrados en el sistema, se hace una búsqueda en la tabla relacional *user_events*.

A parte, se ofrece al usuario el mismo array mostrado en el calendario en forma de lista, que permite la eliminación del evento pulsando sobre la cruz. Este método es similar al mostrado en la figura 35, eliminarMedicamento ().

4.3.7. AÑADIR ELEMENTOS

La última interfaz mostrada es la perteneciente a la funcionalidad de añadir distintos elementos que se han podido apreciar durante la navegación de la aplicación. Estos son, los eventos, la medicación y los documentos.

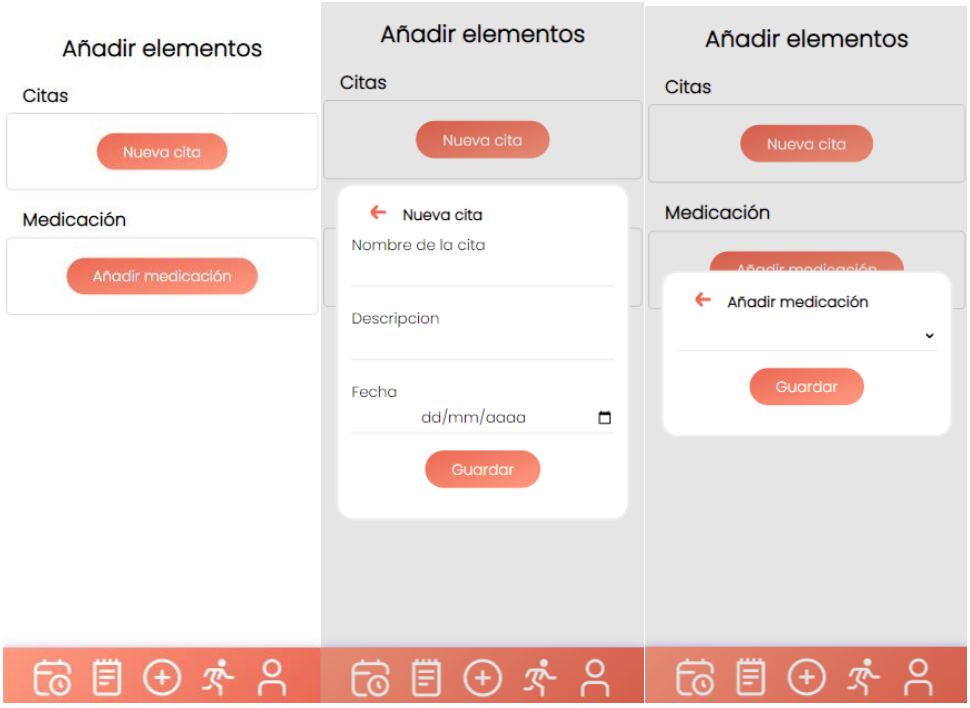

Figura 43 Interfaz de usuario de Añadir elementos

Por el momento solo se ha podido añadir funcionalidad a dos de los tres botones, Nueva cita y Añadir medicación, dejando la opción de Subir documento para el futuro debido a unos problemas durante el desarrollo los cuales se detallan al final de esta memoria.

Las funciones añadidas son similares, mostrando una ventana de dialogo donde se deben añadir los datos solicitados, teniendo la opción de "Guardar" o cancelar la acción, cerrando este modal sin guardar los cambios.

```
nuevaCita() {
  try {
    this.evento.title = this.nombre;
    this.evento.description = this.descripcion;
    this.evento.start = this.date;
    this.eventService.saveEvent(this.evento).subscribe(res => {
      this.ani = res;
      const e = this.ani[0];
      this.user_event.id_event = e[0].id;
      this.usereventService.saveUser_event(this.user_event).subscribe(res => {
        console.log(res)
```

Figura 44 Código de nuevaCita() en NewitemsComponent

Como ejemplo se explica en detalle el código del método de añadir citas, ya que su funcionamiento es bastante parecido a añadir documento y añadir medicación. Para poder guardar un evento, primero se debe crear una variable *evento*, que tiene un título, una descripción y una fecha. La información se obtiene de los campos formulario del dialogo “Nueva cita”.

Una vez actualizados los campos se hace uso del servicio de eventos para crear un nuevo evento con los datos del evento, y una vez creado, con su ID y el ID del usuario que tiene la sesión iniciada se pasa a crear un nuevo user_events mediante su respectivo servicio.

4.4. DESPLIEGUE

Finalmente, se ha decidido desplegar la aplicación para que cualquier usuario pueda hacer uso de ella. Para este despliegue se ha hecho uso de distintas herramientas, como son Vercel y Heroku.

Figura 45 Logotipos de Vercel y Heroku

Vercel es una plataforma de código abierto que permite el despliegue de distintos Frameworks, entre ellos Angular, por lo que se ha decidido usar esta herramienta. [20]. A la web se puede acceder mediante el siguiente enlace:

`escleapp.vercel.app`

Se recomienda abrir la aplicación en un teléfono móvil, y si se quiere acceder desde un ordenador, hacerlo mediante el DevTool para habilitar la vista smartphone, ya que la aplicación no está pensada para verse en formato escritorio.

Para el despliegue de la Api REST y de la base de datos se ha hecho uso de Heroku [21], una plataforma de servicios en la nube que permite el manejo de servidores, y que se ha podido hacer uso gracias a su plan gratuito que permite tener hasta veinte proyectos.

Para poder subir la base de datos, se ha tenido que hacer una migración de bases de datos, ya que no soporta MySQL, teniendo que convertirla a PostgreSQL, otro sistema de bases de datos relacional que si esta soportado por Heroku.

5. CONCLUSIONES

Como se aprecia durante la lectura de esta memoria, este proyecto se sustenta del trabajo aprendido y realizado durante el grado cursado, en especial en la rama de Ingeniería del Software, aplicando los conocimientos desarrollados en cuanto a bases de datos, desarrollo y metodologías. Se ha llegado al final del proyecto y se destaca la satisfacción personal del resultado logrado. La realización de una aplicación software en solitario, con sus problemas y dificultades, me ha ayudado a gestionar el tiempo y los recursos y a la mejorar mi mentalidad para sacarlo adelante.

Con respecto al trabajo final, se puede afirmar que se han alcanzado en su mayoría los objetivos planteados al inicio de esta memoria, a destacar el objetivo principal, y es que ahora un enfermo de esclerosis puede participar activamente en la gestión de su enfermedad.

En cuanto a los objetivos secundarios, el usuario es capaz de realizar cualquier tipo de acción propuesta durante la especificación de requisitos, a falta de mejorar la gestión de documentos personales.

Durante la realización de la carrera, se ha dedicado muy poco tiempo de aprendizaje personal sobre bases de datos y la gestión de estas, marcando como un objetivo personal ser capaz de mejorar en este aspecto, finalizando en un conocimiento más avanzado sobre la parte backend de una solución software, esperando que en el futuro estos conocimientos se amplíen más si cabe.

En definitiva y como conclusión, el trabajo desarrollado ha significado un gran número de horas de aprendizaje, conocimiento de herramientas, diseño, desarrollo y dificultades que se espera resulte en una ampliación de conocimientos y experiencia de cara a un futuro laboral más exitoso y provechoso.

TRABAJO FUTURO

La aplicación desarrollada corresponde a una primera versión que se espera poder ampliar próximamente. Esta primera versión está a la espera de una validación del producto y del feedback de posibles usuarios. Es por ello por lo que a continuación se listan una serie de tareas pendientes y trabajos futuros que se espera puedan llevarse a cabo para cumplir con todos los objetivos marcados en el proyecto.

- Finalización de la gestión de documentos, ya que por falta de conocimientos y de tiempo, no se ha podido cumplir con el requisito funcional de "Añadir documento", debido a problemas con Storage de Firebase, haciendo saltar una excepción en la inicialización del servicio.
  Se cree que este problema es debido a que el constructor de nuestro componente hace uso del servicio de Storage demasiado rápido, antes de que este se haya podido inicializar.

- Mejorar en la información que se proporciona al usuario, y es que como se ha podido apreciar, el usuario no recibe ningún tipo de feedback con la aplicación.
  Para el futuro, se quiere implementar todo tipo de mensajes de información, error y advertencias para que, si el usuario comete algún error, o finaliza una acción, esta información se muestre en pantalla.

- Añadir distinta información y opciones al usuario que estipulen los aspectos esenciales para la protección de la privacidad, tales como el consentimiento informado y previo del usuario para tratar los datos según la Ley Orgánica de Protección de Datos.

- Implementación de animaciones durante la navegación en la aplicación, mejorando la transición entre diferentes pantallas o la aparición suave de las ventanas modales.
  Para la implementación de estas animaciones, ya está instalado en el proyecto Angular Material, modulo que contiene una amplia biblioteca de animaciones, dejando como trabajo futuro el aprendizaje en el uso de estos.

6. BIBLIOGRAFÍA

[1] Qué es la Esclerosis Múltiple - Esclerosis Múltiple España [Internet]. Esclerosis múltiple España. 2015. Disponible en: https://esclerosismultiple.com/esclerosis-multiple/que-es-la-esclerosis-multiple/ [Consultado: 16/06/2022]

[2] Porras-Betancourt M, Lilia N-O, Ni P-Á, Sergio S-S. Esclerosis múltiple [Internet]. Revmexneurociencia.com. Disponible en: http://previous.revmexneurociencia.com/wp-content/uploads/2014/06/Nm071-10.pdf [Consultado: 25/07/2022]

[3] Palazzo S. Un sencillo seguimiento de la EM que te ayudará a sentirte bajo control [Internet]. Emilyn. 2020. Disponible en: https://es.emilyn.app/home/ [Consultado: 26/07/2022]

[4] Biogen. Te presentamos a Cleo, la app diseñada a ayudarte en tu día a día con Esclerosis Múltiple [Internet]. Biogen. Disponible en: https://cleo-app.es/index.html [Consultado: 26/07/2022]

[5] Pressman RS. Ingenieria del software. McGraw-Hill Interamericana; 1992. [Consultado: 19/06/2022]

[6] Toledo González, C. Efectos de un programa de entrenamiento multimodal de equilibrio, fuerza y estabilidad del tronco en personas con esclerosis múltiple. [Consultado: 23/07/2022]

[7] Especificación de Requisitos según el estándar de IEEE 830 [Internet]. Ucm.es. Disponible en: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf [Consultado:20/08/2022]

[8] Bracey K. ¿Qué es Figma? [Internet]. Web Design Envato Tuts+. Envato Tuts; 2018. Disponible en: https://webdesign.tutsplus.com/es/articles/what-is-figma--cms-32272 [Consultado: 18/07/2022]

[9] Angular [Internet]. Angular.lat. Disponible en: https://docs.angular.lat/docs [Consultado: 16/06/2022]

[10] Otto M, Thornton J. Introduction [Internet]. Getbootstrap.com. Disponible en: https://getbootstrap.com/docs/4.1/getting-started/introduction/ [Consultado: 19/06/2022]

[11] Firebase documentation [Internet]. Firebase. Disponible en: https://firebase.google.com/docs [Consultado: 01/07/2022]

[12] Documentación [Internet]. Node.js. Disponible en: https://nodejs.org/es/docs/ [Consultado: 03/07/2022]

[13] Express - Infraestructura de aplicaciones web Node.js [Internet]. Expressjs.com. Disponible en: https://expressjs.com/es/ [Consultado: 03/07/2022]

[14] MySQL 8.0 reference manual [Internet]. Mysql.com. Disponible en: https://dev.mysql.com/doc/refman/8.0/en/ [Consultado: 05/07/2022]

[15] Krogh JW. MySQL Workbench. En: MySQL 8 Query Performance Tuning. Berkeley, CA: Apress; 2020. p. 199–226. [Consultado: 05/07/2022]

[16] Fazt. Nodejs y Mysql Rest API [Internet]. Youtube; 2018. Disponible en: https://www.youtube.com/watch?v=p8CoR-wymQg [Consultado: 11/07/2022]

[17] Reglamento General de Protección de Datos del Espacio Económico Europeo [Internet]. Boe.es. Disponible en: https://www.boe.es/doue/2016/119/L00001-00088.pdf [Consultado: 01/09/2022]

[18] BOE.es - BOE-A-2018-16673 Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales [Internet]. Boe.es. Disponible en: https://www.boe.es/eli/es/lo/2018/12/05/3 [Consultado: 02/09/2022]

[19] Documentation [Internet]. Fullcalendar.io. Disponible en: https://fullcalendar.io/docs [Consultado: 19/08/2022]

[20] Guides [Internet]. Vercel Documentation. Vercel; Disponible en: https://vercel.com/guides [Consultado: 28/08/2022]

[21] Documentation [Internet]. Heroku.com. Disponible en: https://devcenter.heroku.com/categories/reference [Consultado: 28/08/2022]

7. ANEXO A: RELACIÓN CON LOS ODS

El 25 de septiembre de 2015, los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda de desarrollo sostenible. Cada objetivo tiene metas específicas que deben alcanzarse en los próximos 15 años.

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Proceede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**	**X**			
ODS 4. **Educación de calidad.**				**X**
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**				**X**
ODS 9. **Industria, innovación e infraestructuras.**				**X**
ODS 10. **Reducción de las desigualdades**.		**X**		
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**				**X**
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Los Objetivos de Desarrollo Sostenible (ODS) muestran una mirada integral, indivisible y una colaboración internacional renovada. En conjunto, construyen una visión del futuro que queremos. Con el desarrollo del Trabajo de Fin de Grado se busca desarrollar, por petición de la escuela, uno o varios objetivos de desarrollo sostenible. Para este trabajo se han identificado concretamente dos objetivos que se adecuan al propósito de la aplicación desarrollada. Estos son el ODS de Salud y bienestar y Reducción de las desigualdades.

La aplicación, entre sus objetivos principales, busca promover el bienestar y la actividad física, con el fin de mejorar la salud de las personas enfermas de esclerosis múltiple. Las personas afectadas de Esclerosis Múltiple tienen la posibilidad de disminuir las consecuencias de la enfermedad y mejorar la vivencia de esta mediante los autocuidados que cada persona decide poner en marcha.

Esta mejora viene dada gracias a los consejos, hábitos y entrenamientos que pone la app a disposición del usuario. Los diferentes entrenamientos, que varían según el nivel de intensidad con el que se desea entrenar, están ideados especialmente para la mejora física de una persona con esclerosis múltiple, que, combinado con los consejos de hábitos diarios que giran en torno a esta enfermedad permite a los usuarios de la aplicación obtener una mejora de la salud.

Reducir las desigualdades y garantizar que nadie se queda atrás forma parte integral de la consecución de los Objetivos de Desarrollo Sostenible. Esta desigualdad viene dada de las dificultades que tienen los pacientes de esclerosis múltiple en su día a día. Es por ello por lo que, gracias al trabajo desarrollado, en especial la gestión de documentos médicos, de un calendario de citas y el control de la medicación, es posible mejorar e incrementar la inclusión de los enfermos de esclerosis múltiple en la sociedad.

Para finalizar, me gustaría remarcar la importancia de que la Universitat Politècnica de Valencia se comprometa a contribuir en el desarrollo de las ODS, promoviendo al alumnado a que reflexione sobre la relación de estas con el Trabajo de Fin de Grado, permitiendo al estudiante a indagar más sobre estos objetivos, y facilitando la implementación de estos objetivos en proyectos futuros.