UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADA EN INGENIERÍA DEL SOFTWARE

DISEÑO Y DESARROLLO DE UNA APLICACIÓN PARA EL SEGUIMIENTO DE UN PROGRAMA DE REHABILITACIÓN CARDIACA EN PACIENTES CON CARDIOPATÍA ISQUÉMICA

TRACKING ISCHEMIC CARDIOPATHOLOGY PATIENTS IN CARDIAC REHABILITATION PROGRAMS APPLICATION

Realizado por
Lucía Montiel Asensio

Tutorizado por
José Luis Pastrana Brincones

Departamento
Lenguaje y ciencias de la computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, MAYO 2023

Fecha defensa: junio de 2023

Resumen

Una persona que sufre un infarto de miocardio o una angina de pecho, tras abandonar el hospital, requiere una serie de atenciones dirigidas a mejorar su capacidad funcional, controlar los factores de riesgo que ocasionaron el evento y restablecer la actividad laboral y social previa. La rehabilitación cardiaca trata de dar respuesta a este problema mediante el abordaje multidisciplinar en la que intervienen: cardiólogo, enfermeros, fisioterapeuta, psicólogo, etc… Consiste en aplicar al paciente programas de ejercicio individualizado, con diferente grado de supervisión, pautas dietéticas, abandono de hábitos insanos como el alcohol o el tabaco, gestión del estrés emocional y readaptación a las actividades cotidianas.

Este trabajo de fin de grado, por una parte, pretende facilitar el trabajo del personal sanitario, ofreciéndole una plataforma que permita asignarle al paciente las pautas a seguir en su tratamiento, y por otra, ayudar al paciente en su proceso de rehabilitación, que podrá introducir y consultar en cualquier momento de forma sencilla y accesible los valores requeridos por su médico de cualquier aspecto relacionado con su proceso.

Para el desarrollo de esta aplicación web se ha usado el framework .NET para la creación de una API REST conectada a una base de datos SQL mediante Entity Framework. El frontend está hecho con Angular usando Typescript y HTML.

Palabras clave: Rehabilitación cardiaca, cardiología, aplicación web, Angular, .NET

Abstract

Someone that had suffered from myocardial infarction or angina pectoris, after leaving the hospital, requires some attention in order to improve their functional capacity, control the risk factors that caused the event and restore their previous working and personal life. Cardiac rehabilitation aims to solve this problem through a multidisciplinary approach in which cardiologists, nurses, physiotherapists, psychologists among others, take part of. This consists of applying individualized exercise programs with different supervision levels, in addition to nutritional guidelines, abandonment of unhealthy habits like alcohol or smoking, emotional stress management and getting back to daily activities.

The purpose of this Final Year Project is, from one side, to make health professionals' work easier by providing a platform that allows them to assign the patients the guidelines that they must follow to complete their treatment, and from the other side, to help the patient in their rehabilitation process, who will be able to populate and check all the values related to their process that their doctor required in a simple and user-friendly way in any given time.

For the development of this web application, we have used .NET Framework to create an API REST connected to a SQL database through Entity Framework. The frontend is made with Angular using Typescript and HTML.

Keywords: Cardiac rehabilitation, cardiology, web application, Angular, .NET

Índice

1

Introducción

1.1 Motivación

A día de hoy, una de las principales causas de mortalidad en España son las enfermedades cardiovasculares, cuya incidencia ha sido agravada por la pandemia de COVID-19 que hemos sufrido estos últimos años. Este fenómeno es bastante preocupante debido a la alta incidencia en la población, y a que, a menudo, afecta a personas jóvenes en edad productiva. Además, según la OMS, el 80% de este tipo de patologías son prevenibles adquiriendo y manteniendo hábitos saludables, que se han perdido a causa del estilo de vida que predomina en la actualidad ya que el sedentarismo, la alimentación poco saludable basada en ultraprocesados, el estrés laboral y personal, el consumo en exceso de alcohol, tabaco y otras sustancias toxicas son bastante frecuentes en nuestra sociedad.

En este trabajo se presenta una aplicación que facilita tanto el seguimiento por parte del personal sanitario de estos pacientes como la correcta ejecución y cumplimiento de las pautas marcadas por estos del lado del usuario afectado.

1.2 Objetivos

El objetivo de este trabajo de fin de grado es el diseño y desarrollo de una aplicación web de página única que permitirá crear y seguir tratamientos de rehabilitación cardiaca en pacientes que lo necesiten.

Este software tendrá dos versiones, la que usará el personal sanitario, con la que se podrán crear programas de ejercicios personalizados detallando la intensidad, duración y tipo de ejercicio. El profesional encargado podrá revisar sus pacientes asignados haciendo un seguimiento tanto de los programas completados como pendientes de ejecución, además de poder comprobar gráficamente la evolución de valores requeridos como el peso corporal o la tensión arterial, asegurando así la correcta progresión de este. Aparte de lo anterior, la nutrición del paciente también toma un papel fundamental, por lo que será posible proporcionar consejos nutricionales adaptados a cada caso.

Por otra parte, la versión que usará el paciente permitirá visualizar la lista y los detalles de los programas asignados tanto pendientes de ejecutar como ya completados. Podrá realizar y añadir comentarios, sensaciones, frecuencia cardiaca alcanzada, etc... al completar la rutina de ejercicio asignada para que su médico haga una revisión. Además de esto, será posible insertar de forma periódica medidas antes nombradas como el peso corporal o la tensión arterial y llevar un seguimiento gráfico de estas. Por último, se podrá consultar los consejos dietéticos pautados por el personal sanitario.

1.3 Estructura de la memoria

Tras una breve introducción al proyecto en el que se expone la motivación y los objetivos de nuestra aplicación, procedemos a explicar el proceso de desarrollo que hemos seguido para realizar este TFG, que será la estructura de nuestra memoria:

1. **Introducción:** Es el capítulo inicial donde se resumen los principales aspectos de la aplicación, como los objetivos, las metodologías y tecnologías usadas.
2. **Análisis del sistema:** Es el capítulo donde se especifican los requisitos funcionales y se desarrollan los casos de uso relacionados con estos. Su objetivo es aclarar el comportamiento del sistema en cada caso dado.
3. **Diseño de la aplicación:** En este capítulo mostraremos la estructura de la aplicación incluyendo modelos para facilitar la comprensión.
4. **Implementación:** En este capítulo describiremos cómo se ha implementado la aplicación, tanto la API como el frontend, de forma que contenga todas las funcionalidades descritas en el capítulo 2.
5. **Conclusiones y líneas futuras:** Este capítulo contiene las posibles ampliaciones que podría tener este proyecto y las reflexiones finales que hemos recopilado tras terminar el proyecto.
6. **Bibliografía:** En este capítulo se enumeran todas las fuentes de información utilizadas para realizar este trabajo.

1.4 Estado del arte

En esta sección vamos a enumerar y explicar brevemente en qué consisten algunas aplicaciones con funcionalidades similares a la desarrollada en este trabajo. En la actualidad, los programas de ejercicio personalizados están en auge, por lo que

haciendo una rápida búsqueda encontraremos múltiples aplicaciones con objetivos similares:

- **AvanzaT:** Esta plataforma surge del Servicio de Cardiología del Clínic. Ofrece un seguimiento del paciente en el que se incluye el entrenamiento físico, el apoyo emocional y el asesoramiento nutricional. Los objetivos de esta aplicación son los mismos que los de *ReHeart*, es decir, facilitar el proceso de rehabilitación de los pacientes para que vuelvan lo antes posible a llevar una vida normal.
- **Train2go:** Se trata de un software en la une que permite la creación de planes de entrenamiento personalizados, está enfocado a entrenadores personales, pero ofrece funcionalidades muy similares. Entre ellas están la monitorización por parte del entrenador de sus clientes, alertas y notificaciones de la actividad de estos, valoración del esfuerzo y sensaciones por parte del atleta tras ejecutar un programa de ejercicio…
- **RC Rehabilitación Cardiaca:** Esta aplicación es de uso exclusivo para trabajadores y pacientes del servicio de Cardiología del Hospital Arnau de Vilanova de Valencia. El objetivo de este sistema es el mismo que el nuestro, es decir, ayudar al usuario en su rehabilitación cardiaca. Sus principales funcionalidades son el seguimiento por parte del paciente y del sanitario de las tareas programadas, la gestión de tratamiento, el control de las variables de seguimiento y la posibilidad de comunicación entre el paciente y su médico.

1.5 Tecnologías utilizadas

En este punto pasaremos a enumerar y explicar brevemente las características principales de las tecnologías que hemos utilizado durante el desarrollo de nuestra aplicación.

- **.NET Framework:** .NET es un entorno de ejecución administrado para Windows y creado por Microsoft, que proporciona varios servicios a las aplicaciones en ejecución. Tiene dos principales componentes: Common Language Runtime (CLR), que es el motor de ejecución que controla las aplicaciones en funcionamiento, y la biblioteca de clases de .NET Framework, que proporciona una biblioteca de código probado y reutilizable al que pueden llamar los desarrolladores desde sus propias aplicaciones, facilitando así el trabajo de estos.

- **C#:** C# es un lenguaje de programación moderno y multiparadigma, pero principalmente orientado a objetos y con seguridad de tipos desarrollado por Microsoft. Su sintaxis es derivada de C/C++ y similar a Java, aunque incluye mejoras con respecto a estos. La última versión estable es la 10 aunque la 11 está ya en pruebas. Con este lenguaje se pueden desarrollar todo tipo de aplicaciones, desde servidores web hasta programas con interfaz gráfica. C# es uno de los más usados hoy día y forma parte de .NET Framework del que hemos hablado anteriormente.

- **Visual Studio 2022:** Microsoft Visual Studio es un IDE o entorno de desarrollo integrado compatible con Windows y macOS. Es compatible con multitud de lenguajes y frameworks, entre ellos C# y .NET, que son los utilizados en este caso. Permite crear desde aplicaciones web a aplicaciones para dispositivos móviles entre otras. Su versatilidad y estabilidad lo hace uno de los IDE más utilizados.

- **Angular:** Angular es un framework de código abierto mantenido por Google para crear aplicaciones web desarrollado en Typescript. Permite

desarrollar aplicaciones web de una sola página (SPA). El código se divide en componentes formados fundamentalmente por un archivo Typescript, uno HTML y otro CSS. Se usa la arquitectura de Modelo-Vista-Controlador, donde el fichero Typescript hace el papel de controlador y la vista es en HTML.

- **Typescript:** Typescript es un lenguaje de programación de código abierto y multiparadigma desarrollado por Microsoft. Es un superconjunto de Javascript, es decir, extiende la sintaxis de este, aunque se diferencian en que Typescript está fuertemente tipado.

- **Node.js:** Node.js es un entorno que trabaja en tiempo de ejecución, de código abierto y multiplataforma. Usa Javascript y se basa en el motor V8 de Google. Permite a los desarrolladores crear todo tipo de aplicaciones de lado servidor.

- **Bootstrap:** Bootstrap es una biblioteca multiplataforma de código abierto que contiene plantillas de diseño basadas en HTML y CSS, así como extensiones de Javascript por lo que facilita el desarrollo y diseño de sitios web responsivos.

- **Visual Studio Code:** Visual Studio Code es un editor de código gratuito y de código abierto desarrollado por Microsoft. Es multiplataforma y soporta multitud de lenguajes de programación, esto junto a que es fácilmente personalizable y extensible lo hace uno de los IDE más populares.

- **Entity Framework:** Entity Framework es una solución de código abierto multiplataforma desarrollada por Microsoft para un ORM, esto es un modelo de programación que permite mapear las tablas de una base de datos relacional sobre la estructura de entidades. Este framework permite convertir las estructuras de la base de datos en clases con las que poder trabajar, facilitando la comunicación con la base de datos. Entity Framework permite usar tanto consultas LINQ como SQL, lo que lo hace bastante versátil.

- **SQL Server Management Studio:** SQL Server Management Studio (SSMS) es un entorno de desarrollo integrado que permite administrar estructuras SQL. Con SSMS se pueden crear bases de datos y scripts SQL además de generar consultas, secuencias de comandos y procedimientos almacenados entre otras funcionalidades.

1.6 Metodología de trabajo

Para el desarrollo de este trabajo se ha hecho uso de Scrum, que es un marco de gestión de proyectos de metodología ágil. Es un proceso que consiste en aplicar de manera regular un conjunto de buenas prácticas para obtener el mejor resultado posible en el proyecto. Scrum se caracteriza principalmente por:

- Centrarse en la mejora continua.
- Adoptar una estrategia de desarrollo incremental e iterativa.
- Gestionar regularmente las expectativas del proyecto, ser flexibles y capaces de adaptarse a nuevas circunstancias.

- Basar más la calidad del resultado en los conocimientos y capacidades del equipo que en la calidad de los procesos empleados.
- Superponer las diferentes fases del desarrollo en vez de realizarlas de forma secuencial.

El desarrollo de proyectos siguiendo este proceso se divide en sprints, que son periodos de entre una y cuatro semanas cuyo objetivo es crear un incremento de software posiblemente entregable, es decir, ya se podría utilizar.
Las características que forman parte de cada sprint están recopiladas en el *Product Backlog,* que es un conjunto de requisitos funcionales ordenados por orden de prioridad.

Scrum está diseñado para proyectos en los que se trabaja en equipo, en este caso se ha trabajado en solitario, por lo que las reuniones no se han celebrado, pero sí se han definido 5 sprints periódicos con una duración de 15 días cada uno aproximadamente. Al inicio de cada uno de estos se han quedado establecidas las funcionalidades que deben estar presentes tras la finalización de este. Para fijar y especificar estas funcionalidades se han subdividido cada una de ellas en tareas más pequeñas para poder describirlas con más precisión y que nos sea más fácil asegurar la calidad del resultado. Estas tareas ya definidas y descritas pasan a formar parte del *Product Backlog* de ese sprint en concreto.
Al iniciar el desarrollo de cada sprint, ordenamos las tareas de forma que se completen antes aquellas que no tienen ninguna dependencia, es decir, lo que está desarrollado en el momento en el que se empiece con esa tarea es suficiente para el correcto funcionamiento de esta. Así no tendremos bloqueos y el flujo de trabajo será fluido. Una vez hecho esto comenzaremos con la ejecución del sprint en el orden establecido.

Siguiendo este proceso, hemos desarrollado una aplicación completa de forma incremental añadiendo en cada iteración nuevas funcionalidades.

2

Análisis del sistema

En este capítulo definiremos al detalle la funcionalidad y características de la aplicación desarrollada. Enumeraremos los tipos de usuario, es decir, los actores principales. Seguidamente especificaremos los requisitos funcionales capturados.

2.1 Actores principales de la aplicación

En nuestra aplicación se han identificado dos tipos de usuario, es decir, dos actores principales:

- **Usuario paciente:** Usuario autenticado que podrá visualizar todos sus datos como pueden ser sus planes de ejercicio y pautas nutricionales, pudiendo interactuar con el sistema completando estos últimos que estén pendientes e insertar las mediciones requeridas.

- **Usuario sanitario:** Usuario autenticado con permisos para crear planes de ejercicio, consejos nutricionales y revisar la actividad de los usuarios que tengan el rol de paciente y estén asignados a él.

2.2 Requisitos funcionales

En esta sección detallaremos una lista de los requisitos funcionales que hemos identificado en nuestro sistema.

Empezaremos enumerando las funcionalidades relativas a los usuarios con el rol de Sanitario:

RF1: Un usuario con rol de Sanitario podrá iniciar sesión en el sistema utilizando su nombre de usuario y contraseña.

RF2: Un usuario con rol de Sanitario podrá cerrar sesión en el sistema.

RF3: Un Sanitario podrá crear un programa de ejercicios para un paciente especificando el número de sesiones y detalles de las actividades requeridas como la intensidad, duración o número de repeticiones.

RF4: Un Sanitario podrá añadir consejos nutricionales a un paciente.

RF5: Un Sanitario podrá modificar o eliminar los consejos nutricionales de un paciente.

RF6: Un Sanitario podrá visualizar una lista de sus pacientes asignados.

RF7: Un Sanitario podrá filtrar su lista de pacientes por apellidos o número de historia.

RF8: Un Sanitario podrá visualizar los datos personales de un paciente.

RF9: Un Sanitario podrá visualizar los programas de ejercicio de un paciente tanto completados como no completados.

RF10: Un Sanitario podrá visualizar las pautas nutricionales de un paciente.

RF11: Un Sanitario podrá consultar la progresión de los valores requeridos por su médico y añadidos por el paciente.

RF12: Un sanitario podrá consultar las notificaciones creadas cada vez que un paciente actualice algún aspecto.

Ahora pasaremos a listar los requisitos correspondientes a los usuarios con rol de Paciente:

RF13: Un usuario con rol de Paciente podrá iniciar sesión en el sistema utilizando su nombre de usuario y contraseña.

RF14: Un usuario con rol de Paciente podrá cerrar sesión en el sistema.

RF15: Un Paciente podrá visualizar sus programas de ejercicio tanto completados como no completados.

RF16: Un Paciente podrá visualizar sus consejos nutricionales asignados por su médico.

RF17: Un Paciente podrá consultar la progresión de los valores requeridos por su médico y añadidos por este.

RF18: Un Paciente podrá completar una rutina de ejercicios pudiendo añadir observaciones finales como sensaciones, frecuencia cardiaca alcanzada, esfuerzo percibido, etc…

RF19: Un Paciente podrá añadir, cuando sea necesario, los valores requeridos por su médico.

2.3 Casos de uso

En esta sección se describirán los casos de uso, es decir, se expondrán las respuestas del sistema ante las posibles acciones de los usuarios. Como ya se ha explicado en un punto anterior, en nuestra aplicación hay dos tipos de usuarios, y por lo tanto dos actores, el sanitario y el paciente. A continuación, se muestran los casos de uso divididos por tipo de actor.

2.3.1 Casos de uso del usuario con rol de Sanitario

Empezaremos describiendo los casos de uso relacionados con los usuarios con rol de sanitario, para hacernos una idea general mostraremos un diagrama UML de ellos.

Figura 1 - Diagrama de casos de uso del rol Sanitario

Título	Iniciar sesión como sanitario
Descripción	El sistema permite a un usuario iniciar sesión como sanitario.
Precondición	No debe haber otro usuario con la sesión iniciada en el navegador.
Postcondición	El usuario ha iniciado sesión en el sistema.
Escenario principal	

1. El usuario accede a la página de inicio de sesión del sistema. 2. El usuario ingresa su nombre de usuario y contraseña. 3. El sistema comprueba que los datos son correctos. 4. El usuario inicia sesión correctamente. 5. El sistema dirige al usuario al panel inicial.
Escenario alternativo
3.a. El sistema comprueba que los datos no son correctos. 4.a. El usuario no inicia sesión. 5.a. El sistema vuelve a mostrar la página de inicio de sesión.

Figura 2 - Caso de uso Iniciar sesión como Sanitario

Título	Cerrar sesión como sanitario
Descripción	El sistema permite a un usuario cerrar sesión.
Precondición	Debe haber un usuario con rol de sanitario con la sesión iniciada.
Postcondición	El usuario ha cerrado sesión en el sistema.
Escenario principal	
1. El usuario hace click en el botón de cierre de sesión del header. 2. El sistema cierra la sesión del usuario y redirige la pantalla a la de inicio de sesión.	
Escenario alternativo	
Sin escenarios alternativos.	

Figura 3 - Caso de uso cerrar sesión como sanitario

Título	Crear un programa de ejercicios.
Descripción	Un sanitario puede crear un nuevo programa de ejercicios para un paciente.
Precondición	El usuario creador tiene que tener el rol de sanitario y tener asignado a dicho paciente.
Postcondición	El programa de ejercicios estará creado.
Escenario principal	
1. El usuario está en la página de los detalles de un paciente y hace click en "Crear nueva" en tareas asignadas. 2. El sistema le redirige al formulario de creación de un nuevo programa. 3. El usuario ingresa los datos necesarios como el número de sesiones necesarias y los ejercicios y hace click en "Crear rutina". 4. El sistema crea el programa y redirige al usuario a la pantalla de los detalles del programa.	
Escenario alternativo	
4.a. El sistema no crea el programa y sigue mostrando el formulario de creación.	

Figura 4 - Caso de uso crear un programa de ejercicios

Título	Añadir consejos nutricionales
Descripción	Un sanitario puede añadir consejos nutricionales a un paciente.
Precondición	El usuario creador tiene que tener el rol de sanitario y tener asignado a dicho paciente.
Postcondición	El consejo estará añadido a la lista de las pautas del paciente.
Escenario principal	

1. El usuario está en la página de los detalles de un paciente y hace click en "Modificar" en consejos nutricionales. 2. El sistema le redirige al formulario de modificación de las pautas nutricionales. 3. El usuario hace click en "Añadir consejo" y rellena el campo de texto. 4. El usuario hace click en "Guardar cambios". 5. El sistema guarda el nuevo consejo en la lista del paciente y redirige al usuario a la pantalla de detalles del paciente.
Escenario alternativo
4.a. El usuario indica al sistema que vuelva a la pantalla de detalles del paciente. 5.a. El sistema redirige al usuario a la pantalla de detalles del paciente.

Figura 5 - Caso de uso añadir consejos nutricionales

Título	Modificar consejos nutricionales
Descripción	Un sanitario puede modificar consejos nutricionales a un paciente.
Precondición	El usuario creador tiene que tener el rol de sanitario y tener asignado a dicho paciente.
Postcondición	El consejo estará añadido a la lista de las pautas del paciente.
Escenario principal	

1. El usuario está en la página de los detalles de un paciente y hace click en "Modificar" en consejos nutricionales. 2. El sistema le redirige al formulario de modificación de las pautas nutricionales. 3. El usuario rellena el campo de texto del consejo elegido con la nueva información. 4. El usuario hace click en "Guardar cambios". 5. El sistema guarda el nuevo consejo en la lista del paciente y redirige al usuario a la pantalla de detalles del paciente.
Escenario alternativo
4.a. El usuario indica al sistema que vuelva a la pantalla de detalles del paciente. 5.a. El sistema redirige al usuario a la pantalla de detalles del paciente.

Figura 6 - Caso de uso modificar consejos nutricionales

Título	Eliminar consejos nutricionales
Descripción	Un sanitario puede eliminar consejos nutricionales a un paciente.
Precondición	El usuario creador tiene que tener el rol de sanitario y tener asignado a dicho paciente.
Postcondición	El consejo será eliminado de la lista de las pautas del paciente.
Escenario principal	

1. El usuario está en la página de los detalles de un paciente y hace click en "Modificar" en consejos nutricionales. 2. El sistema le redirige al formulario de modificación de las pautas nutricionales. 3. El usuario hace click en "Eliminar" en el consejo en cuestión. 4. El usuario hace click en "Guardar cambios". 5. El sistema guarda el nuevo consejo en la lista del paciente y redirige al usuario a la pantalla de detalles del paciente.
Escenario alternativo
4.a. El usuario indica al sistema que vuelva a la pantalla de detalles del paciente. 5.a. El sistema redirige al usuario a la pantalla de detalles del paciente.

Figura 7 - Caso de uso eliminar consejos nutricionales

Título	Visualizar la lista de pacientes.
Descripción	Un sanitario puede ver la lista de sus pacientes asignados.
Precondición	El usuario creador tiene que tener el rol de sanitario.
Postcondición	El usuario ve sus pacientes asignados.
Escenario principal	
1. El usuario está la pantalla del panel inicial. 2. El sistema muestra la lista de pacientes asignados.	
Escenario alternativo	

No hay escenarios alternativos.

Figura 8 – Caso de uso visualizar la lista de pacientes

Título	Filtrar pacientes
Descripción	Un sanitario puede filtrar su listado de pacientes.
Precondición	El usuario creador tiene que tener el rol de sanitario y estar en la pantalla de panel principal.
Postcondición	La lista de pacientes aparecerá actualizada según el filtro aplicado.
Escenario principal	
1. El usuario selecciona un parámetro sobre el que filtrar y escribe en el buscador. 2. El sistema muestra la nueva lista filtrada de pacientes.	
Escenario alternativo	
1.a. El usuario selecciona un parámetro sobre el que filtrar y no escribe nada en el buscador. 2.a. El sistema muestra la lista de pacientes sin filtrar.	

Figura 9 – Caso de uso filtrar pacientes

Título	Visualizar datos personales de un paciente
Descripción	Un sanitario puede ver la información personal de un paciente.

Precondición	El usuario creador tiene que tener el rol de sanitario, tener asignado a dicho paciente y estar en el panel principal.
Postcondición	El usuario visualiza los datos de un paciente.
Escenario principal	
1. El usuario hace click en uno de los pacientes del listado. 2. El sistema le redirige a la página de los detalles del paciente seleccionado. 3. El usuario puede ver una tabla con los datos personales del paciente.	
Escenario alternativo	
No hay escenarios alternativos.	

Figura 10 – Caso de uso visualizar datos personales de un paciente

Título	Visualizar programas de ejercicio de un paciente
Descripción	Un sanitario puede ver todos los programas de ejercicio de un paciente.
Precondición	El usuario creador tiene que tener el rol de sanitario, tener asignado a dicho paciente y estar en el panel principal.
Postcondición	El usuario visualiza los programas de ejercicio de un paciente.
Escenario principal	
1. El usuario hace click en uno de los pacientes del listado. 2. El sistema le redirige a la página de los detalles del paciente seleccionado.	

3. El usuario puede ver una tabla con los programas de ejercicio completados y por completar del paciente.
Escenario alternativo
No hay escenarios alternativos.

Figura 11 - Caso de uso visualizar programas de ejercicio de un paciente

Título	Visualizar pautas nutricionales de un paciente
Descripción	Un sanitario puede ver las pautas nutricionales de un paciente.
Precondición	El usuario creador tiene que tener el rol de sanitario, tener asignado a dicho paciente y estar en el panel principal.
Postcondición	El usuario visualiza las pautas nutricionales de un paciente.
Escenario principal	
1. El usuario hace click en uno de los pacientes del listado. 2. El sistema le redirige a la página de los detalles del paciente seleccionado. 3. El usuario puede ver una tabla con los programas de ejercicio completados y por completar del paciente.	
Escenario alternativo	
No hay escenarios alternativos.	

Figura 12 - Caso de uso visualizar pautas nutricionales de un paciente

Título	Visualizar notificaciones de los pacientes
Descripción	Un sanitario puede ver las notificaciones de sus pacientes.
Precondición	El usuario creador tiene que tener el rol de sanitario y estar en el panel principal.
Postcondición	El usuario visualiza todas las notificaciones.
Escenario principal	
1. El usuario hace click en la pestaña "Notificaciones" de la tabla. 2. El sistema muestra una tabla con las notificaciones de los pacientes. 3. El usuario puede ver todas las notificaciones de sus pacientes asignados.	
Escenario alternativo	
No hay escenarios alternativos.	

Figura 13 - Caso de uso visualizar notificaciones de los pacientes

2.3.2 Casos de uso del usuario con rol de Paciente

Como en el punto anterior, antes de describir lo casos de uso vamos a mostrar un diagrama UML.

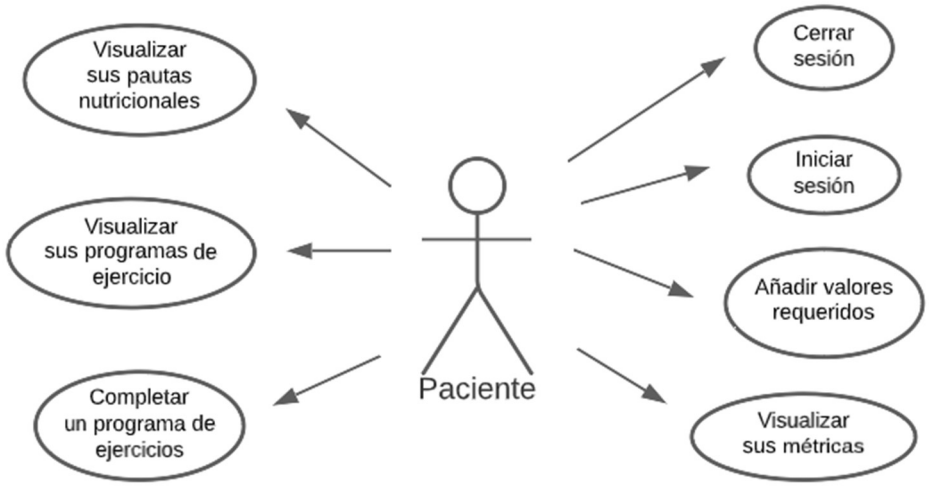

Figura 14 - Diagrama de casos de uso del rol Paciente

Título	Iniciar sesión como paciente
Descripción	El sistema permite a un usuario iniciar sesión como paciente.
Precondición	No debe haber otro usuario con la sesión iniciada en el navegador.
Postcondición	El usuario ha iniciado sesión en el sistema.
Escenario principal	
1. El usuario accede a la página de inicio de sesión del sistema. 2. El usuario ingresa su nombre de usuario y contraseña. 3. El sistema comprueba que los datos son correctos. 4. El usuario inicia sesión correctamente. 5. El sistema dirige al usuario a la pantalla inicial con sus datos.	
Escenario alternativo	

3.a. El sistema comprueba que los datos no son correctos. 4.a. El usuario no inicia sesión. 5.a. El sistema vuelve a mostrar la página de inicio de sesión.

Figura 15 - Caso de uso iniciar sesión como paciente

Título	Cerrar sesión como paciente
Descripción	El sistema permite a un usuario cerrar sesión.
Precondición	Debe haber un usuario con rol de paciente con la sesión iniciada.
Postcondición	El usuario ha cerrado sesión en el sistema.
Escenario principal	
1. El usuario hace click en el botón de cierre de sesión del header. 2. El sistema cierra la sesión del usuario y redirige la pantalla a la de inicio de sesión.	
Escenario alternativo	
Sin escenarios alternativos.	

Figura 16 - Caso de uso cerrar sesión como paciente

Título	Visualizar sus programas de ejercicio

Descripción	Un paciente puede ver sus programas de ejercicio tanto realizados como no realizados.
Precondición	El usuario creador tiene que tener el rol de paciente y estar en la página principal.
Postcondición	El usuario visualiza sus programas de ejercicio.
Escenario principal	
1. El usuario está en la página principal. 2. El sistema muestra la lista de programas de ejercicio del usuario.	
Escenario alternativo	
No hay escenarios alternativos.	

Figura 17 – Caso de uso visualizar sus programas de ejercicio

Título	Visualizar sus pautas nutricionales
Descripción	Un paciente puede ver sus pautas nutricionales.
Precondición	El usuario tiene que tener el rol de paciente y estar en la página principal.
Postcondición	El usuario visualiza sus pautas nutricionales.
Escenario principal	
1. El usuario está en la página principal. 2. El sistema muestra la lista de pautas nutricionales del usuario.	
Escenario alternativo	
No hay escenarios alternativos.	

Figura 18 – Caso de uso visualizar sus pautas nutricionales

Título	Visualizar sus métricas
Descripción	Un paciente puede ver sus métricas de los valores requeridos.
Precondición	El usuario tiene que tener el rol de paciente y estar en la página principal.
Postcondición	El usuario visualiza sus métricas.
Escenario principal	
1. El usuario está en la página principal. 2. El sistema muestra todas las métricas de los valores añadidos.	
Escenario alternativo	
No hay escenarios alternativos.	

Figura 19 – Caso de uso visualizar sus métricas

Título	Completar un programa de ejercicios.
Descripción	Un paciente puede completar una rutina de ejercicios.
Precondición	El usuario creador tiene que tener el rol de paciente y estar en la página principal.
Postcondición	El programa de ejercicios estará completado.
Escenario principal	

1. El usuario hace click en uno de los programas no completados. 2. El sistema le redirige a los detalles de la rutina. 3. El usuario hace click en "Realizar rutina". 4. El sistema muestra el formulario de "Completar rutina". 5. El usuario marca como completados los ejercicios y rellena los campos de las observaciones finales y hace click en "Completar Rutina". 6. El sistema redirige al usuario a la página de los detalles de la rutina ya completada.
Escenario alternativo
5.a. El usuario hace click en la flecha de retroceder. 6.a. El sistema muestra la pantalla de los detalles de la rutina.

Figura 20 – Caso de uso completar un programa de ejercicios

Título	Añadir valores requeridos.
Descripción	Un paciente puede añadir los valores requeridos cuando sea necesario.
Precondición	El usuario creador tiene que tener el rol de paciente y estar en la página principal.
Postcondición	El valor estará añadido a las métricas del paciente.
Escenario principal	
1. El sistema muestra un mensaje de que es necesario insertar una nueva medida. 2. El usuario hace click en "Registrar medida". 3. El sistema le muestra un campo donde escribir el valor. 4. El usuario introduce y envía el valor.	

5. El sistema muestra las métricas teniendo en cuenta el último valor.
Escenario alternativo
1.a. El sistema no muestra el mensaje porque no ha pasado el tiempo indicado para volver a añadir otra.

Figura 21 – Caso de uso añadir valores requeridos

3

Diseño de la aplicación

En este capítulo definiremos cómo hemos diseñado nuestra aplicación para que cumpla de la mejor forma posible los objetivos marcados y los requisitos listados en el punto anterior.

3.1. Modelo relacional

En esta sección mostramos un diagrama en el que se puede ver de forma sencilla la estructura que hemos diseñado para nuestra base de datos.

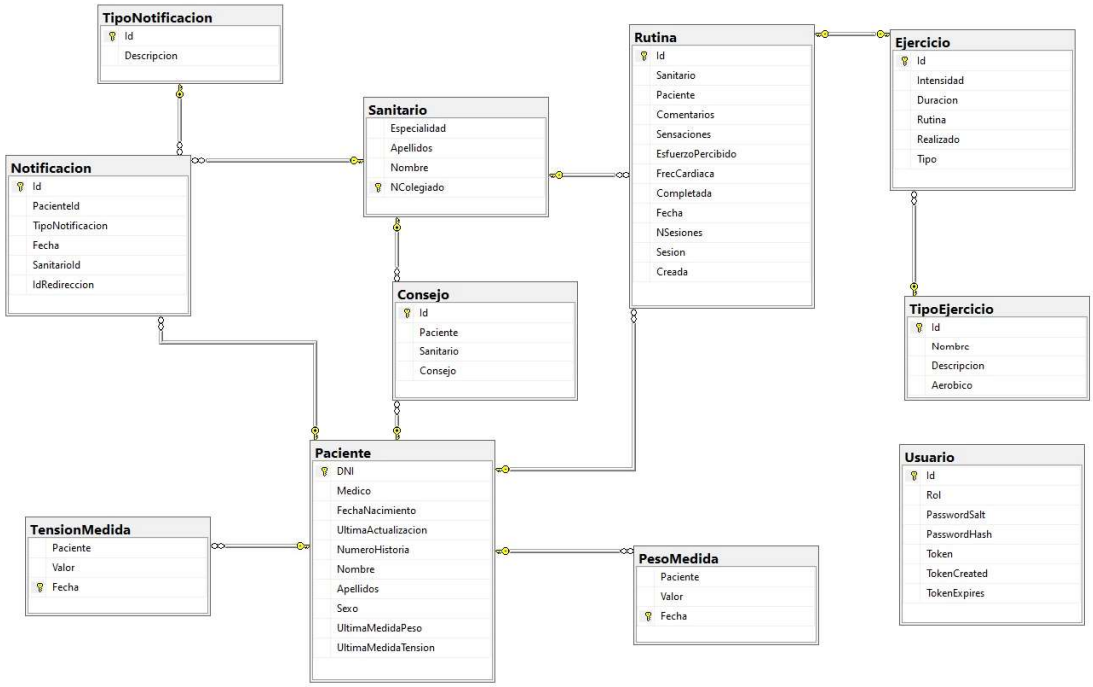

Figura 22 - Modelo relacional

3.2. Modelo de clases

En este sistema hemos utilizado un enfoque *Database First*, es decir, lo primero que hacemos al empezar el desarrollo de nuestra aplicación es diseñar la estructura de la base de datos que utilizaremos, y sobre esa base se construye el modelo de clases de nuestro proyecto. En este caso hemos utilizado Entity Framework, que facilita este trabajo ya que este software crea automáticamente todas las clases que nos hacen falta basándose en nuestro modelo de base de datos. Sabiendo esto, nuestro diagrama de clases será el mismo que el relacional por lo que pasamos a enumerar las clases creadas que nos permitirán implementar nuestro proyecto:

- **Usuario:** Representa a todos los usuarios de la aplicación, **ya** sean sanitarios o pacientes. Esta clase se utiliza para iniciar y cerrar sesión en la aplicación.
- **Sanitario:** Representa a los usuarios con rol de sanitario.
- **Paciente:** Representa a los usuarios con rol de paciente.

- **Rutina:** Representa una sesión de un programa de ejercicio creada por un sanitario para un paciente.
- **Ejercicio:** Representa a un ejercicio perteneciente a una sesión de un programa de ejercicios.
- **TipoEjercicio:** Representa a un tipo de ejercicio el cual se puede incluir en una rutina.
- **Consejo:** Representa a una pauta nutricional creada por un médico para un paciente.
- **Notificación:** Representa a una notificación creada tras una actualización de algún aspecto del paciente.
- **TipoNotificacion:** Representa un tipo de notificación dependiendo de si la actualización del paciente es de ejercicio, medidas, etc…
- **TensionMedida:** Representa una medida de la tensión añadida por un paciente en un momento concreto.
- **PesoMedida:** Representa una medida del peso corporal añadida por un paciente en un momento concreto.

La lista anterior contiene todas las clases creadas automáticamente por Entity Framework, aparte de todas estas, se han creado algunas más para facilitar el desarrollo de nuestra aplicación.

- **FrecCardiacaGraph:** Representa una medida de frecuencia cardiaca de un paciente en una fecha concreta, sacada de una rutina de ejercicios ya completada.
- **Login:** Representa al conjunto de datos ingresados por el usuario al iniciar sesión.
- **RefreshToken:** Representa al token creado cada vez que un usuario inicia la sesión.

- **Register:** Representa al conjunto de datos ingresados por el usuario al registrarse en el sistema.

4

Implementación

En este capítulo se expondrá cómo se ha implementado la aplicación para cumplir todos los requisitos funcionales que hemos listado en capítulos anteriores. Se hará un repaso del código producido para la creación de nuestra API, así como del proyecto de Angular generado para la creación de nuestra web.

4.1 Aspectos generales de la implementación

Para la implementación de nuestra aplicación hemos creado dos proyectos diferentes, una API REST realizada con .NET Framework en C#, que será nuestro backend y estará conectado a la base de datos. Se ha elegido trabajar con estas herramientas debido a las facilidades que ofrece a la hora de programar con

ellas, como pueden ser las bibliotecas de clases listas para usar o el componente LINQ, que facilita la obtención de los datos.

Por otra parte, el frontend que actuará como cliente de la API ya mencionada, es un proyecto de Angular en el que usaremos Typescript para tratar con los datos recibidos y HTML y CSS para crear la interfaz de usuario. Hemos optado por utilizar este framework ya que, entre otras cosas, enlaza el código Typescript y HTML lo que hace sumamente cómodo y rápido el desarrollo

Para el desarrollo del backend hemos utilizado como IDE Visual Studio 2022 y para el frontend Visual Studio Code.

Estos dos proyectos están conectados mediante peticiones HTTP (GET, POST, DELETE, PUT).

4.2 Implementación del Backend

4.2.1 Estructura de la solución

Hemos dividido nuestro proyecto en varias carpetas enumeradas a continuación:

- **Models:** En esta carpeta se incluyen todas las clases de los modelos creados a partir del diagrama relacional. Además, esta incluye la clase que modela el contexto de la base de datos, que fue creada automáticamente por Entity Framework.
- **Services:** En esta carpeta están recogidos todos los servicios, esto es, las clases que contienen los métodos que se comunican con la base de datos para, entre otras cosas, leer, guardar o modificar los registros. Cada modelo tiene su propio servicio. Las funciones que forman parte de cada uno de ellos obtienen o guardan datos en función de los *endpoints* de los controladores, es decir, por lo general cada *endpoint* llama a su función del servicio correspondiente.

- **Controllers:** De esta carpeta forman parte todos los controladores, que son las clases que, mediante los endpoints, manejan las peticiones HTTP y envían respuestas. Para comunicarse con la base de datos utilizan los servicios. Cada modelo tiene su propio controlador, de la misma manera que los servicios.

Aparte de estas carpetas, también es importante mencionar que la solución posee un archivo llamado *Program.cs* que es una clase de C# donde se configuran todos los servicios requeridos, como puede ser la inyección de dependencias.

4.2.2 Endpoints de la API

Para documentar nuestra API hemos hecho uso de *Swagger*, que es un framework que contiene una serie de especificaciones, reglas y herramientas creadas para facilitar este proceso, en concreto, *SwaggerUI*, que facilita en gran manera la organización y pruebas de nuestros métodos al tener una interfaz gráfica intuitiva y amigable para el usuario.

Vamos a pasar a explicar los endpoints de nuestra API que nos permitirán comunicarnos con nuestro cliente que será el proyecto de Angular que hemos mencionado antes. Todos estos métodos reciben la petición y devuelven su respuesta en formato JSON.

- Endpoints para la autenticación de usuarios:
    - POST "/api/Auth/Login": La petición se compone de un nombre de usuario y una contraseña e inicia la sesión del usuario en cuestión.
    - GET "/api/Auth/getById": Se pasa como parámetro un id de un usuario y devuelve el usuario al que pertenece ese id.
    - POST "/api/Auth/Register": Toma los datos de un nuevo usuario del sistema y lo registra en la base de datos.
- Endpoints relacionados con los consejos nutricionales:

    - GET "/api/Consejo": Obtiene todos los consejos nutricionales guardados en la base de datos.
    - POST “/api/Consejo": Recibe los datos de un consejo como el sanitario, el paciente… y se crea un nuevo consejo en la base de datos.
    - GET "/api/Consejo/{id}": Recibe un id y devuelve el consejo con ese identificador.
    - PUT "/api/Consejo/{id}": Recibe un id y un JSON con nuevos datos y actualiza el consejo con ese identificador.
    - DELETE "/api/Consejo/{id}": Recibe un id y elimina el consejo con ese identificador.
    - GET "api/Consejo/getByPaciente/{pacienteId}": Recibe un identificador de un paciente y devuelve todos sus consejos.
- Endpoints relacionados con Ejercicios:
    - GET "/api/Ejercicio": Obtiene todos los ejercicios guardados en la base de datos.
    - POST “/api/Ejercicio": Recibe los datos de un ejercicio como la rutina, la duración… y se crea un nuevo ejercicio en la base de datos.
    - GET "/api/Ejercicio/{id}": Recibe un id y devuelve el ejercicio con ese identificador.
    - PUT "/api/Ejercicio/{id}": Recibe un id y un JSON con nuevos datos y actualiza el ejercicio con ese identificador.
    - DELETE "/api/Ejercicio/{id}": Recibe un id y elimina el ejercicio con ese identificador.
    - GET "api/Ejercicio/getByRutina/{rutinaId}": Recibe un identificador de una rutina y devuelve todos sus ejercicios.
- Endpoints relacionados con los consejos nutricionales:

- GET "/api/Medidas/peso": Obtiene todas las medidas de peso corporal guardadas en la base de datos.
- GET "/api/Medidas/tension": Obtiene todas las medidas de tensión arterial guardadas en la base de datos.
- GET "/api/Medidas/getPeso/{paciente}": Recibe un identificador de un paciente y devuelve todas las medidas de peso corporal de ese paciente.
- GET "/api/Medidas/getTension/{paciente}": Recibe un identificador de un paciente y devuelve todas las medidas de tensión arterial de ese paciente.
- POST "/api/Medidas/createMedidaPeso": Recibe un JSON con una nueva medida de peso corporal y la guarda en la base de datos.
- POST "/api/Medidas/createMedidaTension": Recibe un JSON con una nueva medida de tensión arterial y la guarda en la base de datos.

– Endpoints relacionados con las notificaciones:
  - GET "/api/Notificacion": Obtiene todas las notificaciones guardadas en la base de datos.
  - POST "/api/Notificacion": Recibe los datos de una notificación y se crea una nueva notificación en la base de datos.
  - GET "/api/Notificacion/{id}": Recibe un id y devuelve la notificación con ese identificador.
  - DELETE "/api/Notificacion/{id}": Recibe un id y elimina la notificación con ese identificador.
  - GET "api/Notificacion/getByPaciente/{paciente}": Recibe un identificador de un paciente y devuelve todas sus notificaciones.

    - GET "api/Notificacion/getBySanitario/{sanitario}": Recibe un identificador de un sanitario y devuelve todas las notificaciones de los pacientes asignados a un sanitario.

- Endpoints relacionados con Pacientes:
    - GET "/api/Paciente": Obtiene todos los pacientes guardados en la base de datos.
    - POST “/api/Paciente": Recibe los datos de un paciente como el DNI, nombre, apellidos… y se crea un nuevo paciente en la base de datos.
    - GET "/api/Paciente/{id}": Recibe un id y devuelve el paciente con ese identificador.
    - PUT "/api/Paciente/{id}": Recibe un id y un JSON con nuevos datos y actualiza el paciente con ese identificador.
    - DELETE "/api/Paciente/{id}": Recibe un id y elimina el paciente con ese identificador.
    - GET "api/Paciente/getByMedico/{idMedico}": Recibe un identificador de un sanitario y devuelve todos sus pacientes asignados.
    - GET "api/Paciente/getByNumero/{unmero}": Recibe una cadena de caracteres y devuelve todos los pacientes cuyo número de historia coincida en parte con el parámetro.
    - GET "api/Paciente/getByApellidos/{apellidos}": Recibe una cadena de caracteres y devuelve todos los pacientes cuyos apellidos coincidan en parte con el parámetro.
- Endpoints relacionados con Rutinas:
    - GET "/api/Rutina": Obtiene todas las rutinas guardadas en la base de datos.

    - POST "/api/Rutina": Recibe los datos de una rutina como el sanitario que lo crea, el paciente… y se crea un nuevo paciente en la base de datos.
    - GET "/api/Rutina/{id}": Recibe un id y devuelve la rutina con ese identificador.
    - PUT "/api/Rutina/{id}": Recibe un id y un JSON con nuevos datos y actualiza la rutina con ese identificador.
    - DELETE "/api/Rutina/{id}": Recibe un id y elimina la rutina con ese identificador.
    - GET "api/Rutina/getByPaciente/{paciente}": Recibe un identificador de un paciente y devuelve todos los programas de ejercicio de este.
    - GET "api/Rutina/getByCompletada/{paciente}": Recibe un identificador de un paciente y devuelve todas las rutinas completadas de este.
    - GET "api/Rutina/getByNoCompletada/{paciente}": Recibe un identificador de un paciente y devuelve todas las rutinas no completadas de este.
    - GET "api/Rutina/getFrecCardiaca/{paciente}": Recibe un identificador de un paciente y devuelve una lista de la frecuencia cardiaca alcanzada en las rutinas completadas de este.
- Endpoints relacionados con Sanitarios:
    - GET "/api/Sanitario": Obtiene todos los sanitarios guardados en la base de datos.
    - POST "/api/Sanitario": Recibe los datos de un sanitario como el Nº de colegiado, nombre, apellidos… y se crea un nuevo sanitario en la base de datos.

    - GET "/api/Sanitario/{id}": Recibe un id y devuelve el sanitario con ese identificador.
    - PUT "/api/Sanitario/{id}": Recibe un JSON con nuevos datos y actualiza el sanitario con ese identificador.
    - DELETE "/api/Sanitario/{id}": Recibe un id y elimina el sanitario con ese identificador.
- Endpoints relacionados con TipoEjercicio:
    - GET "api/TipoEjercicio": Obtiene todos los tipos de ejercicio guardados en la base de datos.
    - GET "api/TipoEjercicio/{id}": Recibe un identificador y devuelve el tipo de ejercicio que tenga ese identificador.
    - GET "api/TipoEjercicio/getByNombre/{nombre}": Recibe un nombre y devuelve el tipo de ejercicio cuyo nombre contenga ese parámetro.
- Endpoints de TipoNotificacion:
    - GET "api/TipoNotificacion": Obtiene todos los tipos de notificaciones guardadas en la base de datos.
    - GET "api/TipoNotificacion/{id}": Recibe un identificador y devuelve el tipo de notificación con ese identificador.

4.2.3 Conexión con la base de datos

Para conectar nuestro servidor con la base de datos, en nuestro caso hemos utilizado una base de datos SQL en local utilizando SQL Server, hemos usado Entity Framework, como ya hemos mencionado en el capítulo anterior. Esta conexión se ha hecho mediante *connection string* usando el servidor local y autenticación de Windows.

4.3 Implementación del Frontend

4.3.1 Estructura del proyecto

Todos los proyectos de Angular tienen la misma estructura básica, los archivos importantes están recogidos en la carpeta "src". Esta carpeta está dividida de la siguiente manera:

- **App:** Dentro de esta carpeta están todos los componentes creados para el desarrollo de nuestra aplicación, además de los componentes base comunes para toda la aplicación.
- **Assets:** Aquí están las imágenes utilizadas dentro del proyecto.
- **Environments:** En esta carpeta se recogen los entornos de trabajo utilizados, ya sea la ruta local o la ruta del servidor utilizado.
- **Ficheros:** Dentro de la carpeta *src* hay varios ficheros comunes para toda la aplicación, cabe destacar *styles.css*, que es un archivo de estilos que se aplican en todo el proyecto.

4.3.2 Componentes del frontend

En este punto enumeraremos y explicaremos brevemente los componentes creados para el desarrollo de nuestra aplicación. Estos componentes, principalmente, contienen un archivo Typescript, que recoge todos los métodos que tratan los datos recogidos desde la API, un archivo HTML, que contiene la vista del componente, es decir, la pantalla.

- **create-consejo:** Este componente es accesible para los usuarios con rol de Sanitario, muestra la pantalla de edición de los consejos nutricionales de un paciente y permite añadir, eliminar y modificar estas pautas.
- **create-rutina:** Este componente es accesible para los sanitarios, muestra una pantalla que permite diseñar y crear una rutina de ejercicios para un paciente.

- **footer:** Este componente muestra el footer, es común para toda la aplicación.
- **grafica:** Este componente genera y dibuja las gráficas de las métricas de los pacientes, utilizando datos pasados desde otros componentes cuando es necesario.
- **header:** Este componente muestra el header, es común para toda la aplicación.
- **inicio-paciente:** Este componente es accesible a los usuarios con rol de Paciente, muestra la pantalla inicial que contiene avisos, sus programas de ejercicio, sus pautas nutricionales y sus métricas.
- **inicio-sanitario:** Este componente es accesible a los usuarios con rol de Sanitario, muestra una lista de sus pacientes asignados y notificaciones de las actualizaciones de estos.
- **login:** Este componente es la vista inicial para todos los usuarios, muestra un formulario donde ingresar los datos del usuario y permite
- **logout:** Este componente no tiene vista, pero permite al usuario cerrar la sesión.
- **paciente-details:** Este componente es accesible para los usuarios con rol de Sanitario, muestra los detalles personales de uno de los pacientes asignados al usuario logueado además de sus programas de ejercicio, pautas nutricionales y métricas.
- **rutina-complete:** Este componente es accesible para los usuarios con rol de Paciente, muestra los detalles de la rutina y un formulario con datos a completar. Permite completar una rutina.
- **rutina-details:** Este componente es accesible para todos los usuarios, muestra los detalles de un plan de ejercicio y si el usuario tiene rol de paciente, se muestra la opción de poder completarla.

4.3.3 Otras carpetas creadas

En este punto vamos a enumerar las demás carpetas creadas que son necesarias para nuestra aplicación:

- **models:** Esta carpeta contiene los archivos de todos los modelos de datos de la aplicación. Corresponden con los modelos del backend, es decir, tienen las mismas propiedades.
- **services:** Esta carpeta contiene todos los servicios, cada modelo tiene el suyo propio. Estos archivos contienen los métodos que se conectan con la API y permiten obtener los datos del backend.

5

Conclusiones

5.1 Conclusiones

Con la realización de este trabajo, he superado uno de los obstáculos que más me intimidaba a la hora de empezar el proyecto, que era llevar a cabo yo sola todo el diseño y desarrollo de esta aplicación. Para ello, he tenido que organizar y plantear todo el trabajo pendiente de manera meticulosa para que el resultado fuera de la mejor calidad posible.

En mi caso, las tecnologías utilizadas ya me eran familiares ya que las llevo utilizando más de un año. Esto no significa que no haya aprendido nada nuevo, al contrario, me ha hecho ser consciente de lo complicado que a veces es poner punto y final al alcance de un proyecto.

Una vez finalizado el proyecto, he sido mucho más consciente de lo difícil que es el acceso a las tecnologías por parte de personas que en función de su edad,

conocimientos o preparación no son capaces de utilizar estas aplicaciones. Hay que destacar que no solamente los conocimientos técnicos son importantes, también lo es informarse correctamente con ayuda de profesionales expertos en el ámbito con el que se relaciona el proyecto.

5.2 Líneas futuras

Al determinar el alcance de este proyecto teníamos que tener en cuenta que contábamos con un tiempo limitado y con una sola persona encargada de todo el desarrollo, por lo que habría que delimitar las funcionalidades que este tuviese. Para una ampliación futura de nuestra aplicación se han planteado las siguientes características:

- **Seguimiento diario de la nutrición del paciente:** Actualmente los pacientes disponen de pautas nutricionales personalizadas, pero no es posible que su médico compruebe su dieta diaria, en una futura versión, estará disponible ese seguimiento ya sea con una funcionalidad propia del sistema o accediendo a una API externa como *MyFitnessPal* o *Google Fit*.
- **Posibilidad de conexión con dispositivos móviles y relojes inteligentes:** Sería interesante que, sobre todo los pacientes, puedan sincronizar sus dispositivos a la aplicación para poder acceder a sus programas de ejercicio y poder completarlos en cualquier momento.

Bibliografía

1. Documentación de rehabilitación cardiaca: Visseren, Frank L.J., Mach, François, Smulders, Yvo M., Carballo, David, Koskinas, Konstantinos C., Back, Maria. 2021 ESC Guidelines on cardiovascular disease prevention in clinical practice. European Heart Journal
2. AvanzaT App - Google Play: https://play.google.com/store/apps/details?id=org.avanzaT.app&hl=es_419&gl=US
3. Train2Go – Google Play: https://train2go.com/
4. RC Rehabilitación Cardiaca – Google Play: https://play.google.com/store/apps/details?id=nabelia.cardioplan&hl=es&gl=US
5. .NET Framework – Microsoft Learn Center: https://dotnet.microsoft.com/es-es/learn/dotnet/what-is-dotnet-framework
6. C# - Microsoft Learn Center: https://learn.microsoft.com/es-es/dotnet/csharp/
7. Visual Studio 2022 – Microsoft Visual Studio Web: https://visualstudio.microsoft.com/es/vs/

8. Angular – Angular Web: https://angular.io/
9. Typescript – Typescript Web: https://www.typescriptlang.org/
10. Node.js – Node.js Web: https://nodejs.org/es/docs
11. Bootstrap – Bootstrap Framework: https://getbootstrap.com/
12. Visual Studio Code – Visual Studio Code Web: https://code.visualstudio.com/docs
13. Entity Framework – Microsoft Learn Center: https://learn.microsoft.com/en-us/ef/
14. Scrum - ¿Qué es Scrum?: https://www.atlassian.com/es/agile/scrum

Apéndice A

Manual de Usuario

A.1 Página de inicio de sesión

Al acceder a la aplicación, la primera pantalla que nos encontramos es la de inicio de sesión, el usuario debe introducir su nombre de usuario y su contraseña en los respectivos campos que aparecen en la imagen y pulsar en el botón "Entrar".

Figura 23 – Pantalla de inicio de sesión

A.2 Pantallas del usuario con rol de Sanitario

Una vez que el usuario ha iniciado sesión, si es sanitario, se mostrarán las siguientes páginas y funcionalidades.

A.2.1 Página principal de Sanitario

Si nuestro usuario tiene el rol de Sanitario, en la página principal se mostrará una lista que, si la pestaña seleccionada es la de "Pacientes", mostrará los pacientes asignados junto a varios datos relevantes de estos, como aparece en la imagen.

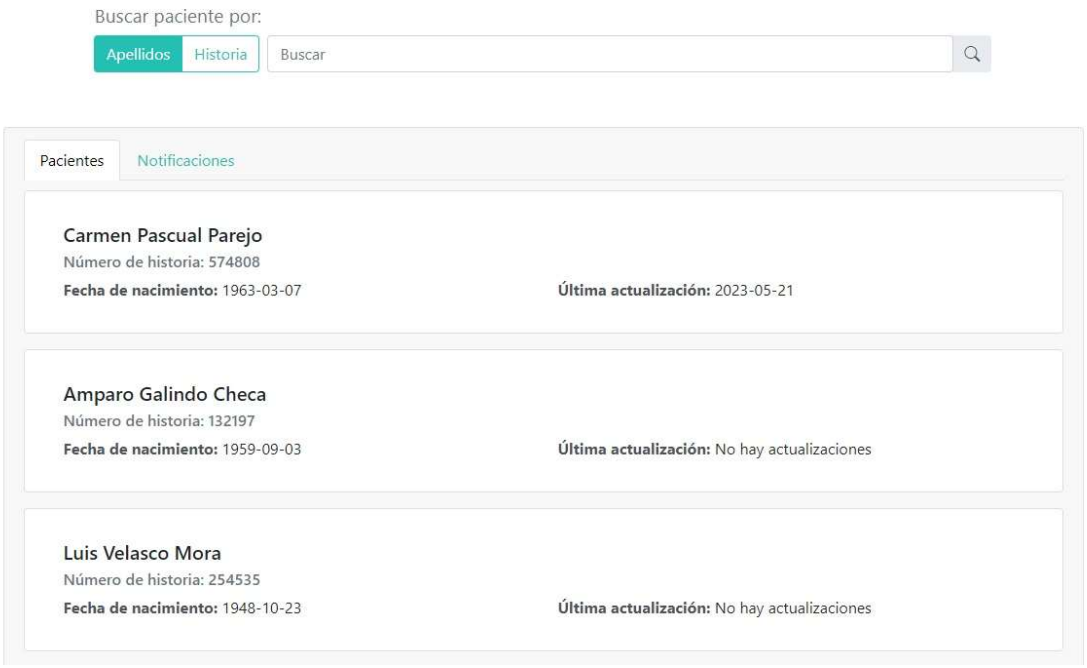

Figura 24 – Pantalla de la lista de pacientes

Si, por el contrario, se ha seleccionado "Notificaciones" se verán los avisos de las últimas actualizaciones, si se hace click sobre alguna fila de esta lista, el sistema redirigirá al usuario a la pantalla donde se muestran todos los detalles de ese paciente.

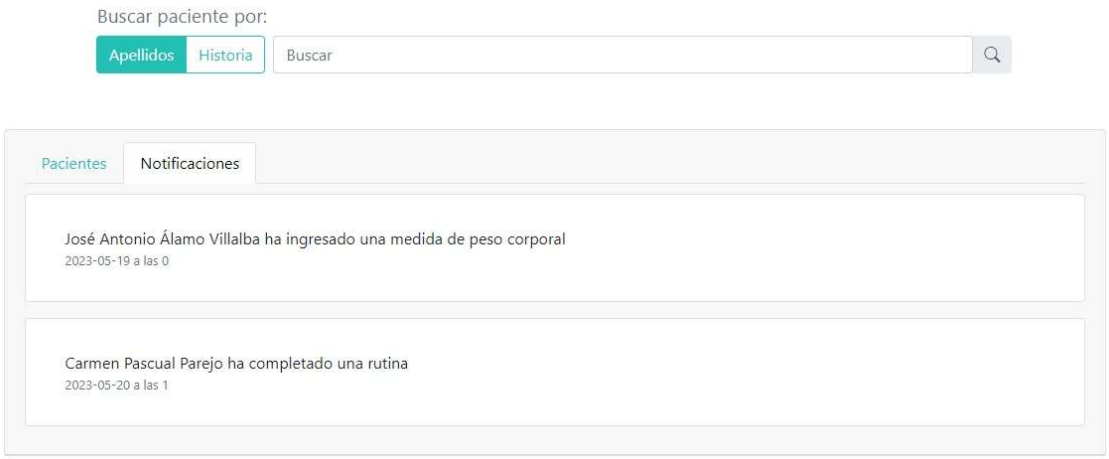

Figura 25 – Pantalla de las notificaciones de pacientes

Si queremos filtrar los pacientes tenemos dos opciones, buscar por apellido o número de historia. Para ello seleccionamos el parámetro de búsqueda, escribimos el texto para filtrar y pulsamos en el icono de la lupa.

A.2.2 Página de detalles del paciente

Una vez que el usuario ha seleccionado un paciente de su lista, el sistema mostrará la página con todos los detalles de este. Se podrá ver un apartado llamado "Ficha del paciente" donde se podrán ver los datos personales del paciente.

Figura 26 – Apartado de la ficha del paciente

En el apartado de "Tareas asignadas" donde dependiendo de si hemos seleccionado la pestaña "No completadas" o "Completadas" nos mostrará una lista de las rutinas del paciente que estén pendientes de completar o ya realizadas

respectivamente. Si se hace click en alguna de ellas se dirigirá a los detalles de la rutina.

Figura 27 – Apartado de tareas del paciente

En el caso de que no haya rutinas no completadas aparecerá un enlace "Crear nueva" que dirigirá al usuario a la página de creación de una nueva rutina.

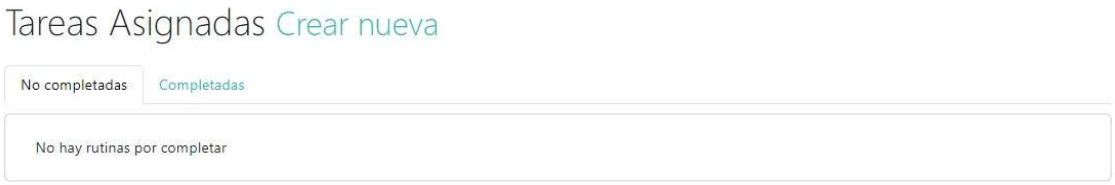

Figura 28 – Apartado de tareas del paciente cuando no hay sin completar

En el apartado "Consejos nutricionales" se pueden ver todas las pautas nutricionales del paciente y si se hace click en "Modificar", al lado del título, el sistema redirigirá al usuario a la pantalla de modificación de los consejos.

Figura 29 – Apartado de consejos nutricionales del paciente

Por último, el usuario podrá ver la evolución de las métricas del paciente, tal y como aparece en la siguiente imagen.

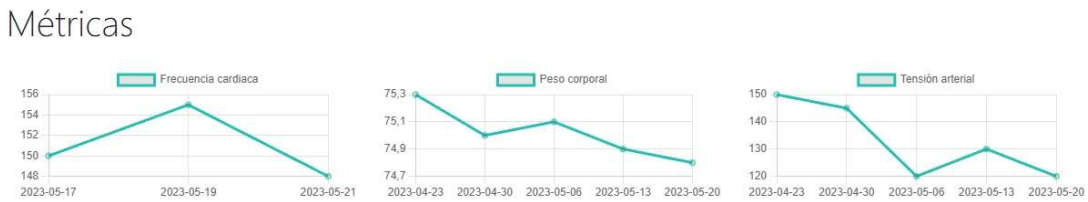

Figura 30 – Apartado de métricas del paciente

A.2.3 Detalles de la rutina

Al seleccionar una rutina, el sistema muestra esta pantalla donde aparece la información de la sesión, indicando el número de esta, el número total de sesiones, el estado y los ejercicios que la componen.

Figura 31 – Pantalla de detalles de la rutina

Si la rutina seleccionada está ya completada, aparecerán las observaciones finales que ha añadido el paciente al realizar la rutina.

Figura 32 – Pantalla de detalles de la rutina completada

A.2.4 Creación de una rutina de ejercicios

Como se ha explicado en el apartado A.2.2, si el paciente no tiene rutinas por completar, el usuario puede acceder al formulario de creación de un programa de ejercicios. En primer lugar, debemos introducir el número de sesiones que queremos que tenga ese programa. La rutina debe tener al menos un ejercicio, por lo que siempre por defecto aparecerá el formulario del primero, debemos completar de forma correcta la información de todos los campos. Si queremos añadir otro ejercicio, se pueden agregar tantos como se quiera, el usuario debe pulsar el botón "Añadir ejercicio" y aparecerá uno nuevo a completar. Una vez el usuario tenga los datos listos, hace click en "Crear rutina", si los datos son correctos el sistema lo redirigirá a la página con los detalles de la rutina ya creada explicado en el

apartado A.2.3. Por un contrario, si alguno de los datos no está completo o es incorrecto aparecerá un mensaje de error debajo del campo en cuestión.

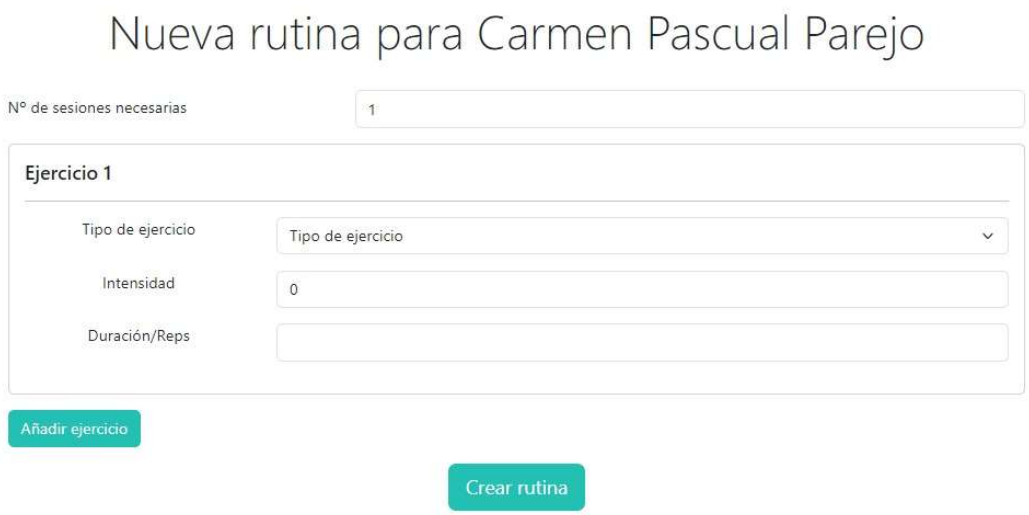

Figura 33 – Pantalla de creación de rutina

A.2.5 Modificación de pautas nutricionales

Como se menciona en el apartado A.2.2, el usuario puede acceder a la página de modificación de pautas nutricionales. Si el paciente aún no tiene ninguna pauta marcada, aparecerá un mensaje que lo indique como aparece en la imagen.

Figura 34 – Pantalla de modificación de consejos nutricionales cuando está vacío

En el caso en el que ya haya alguno, aparecerá una lista de ellos. El usuario podrá editar el contenido de estos con la condición de que el texto no esté vacío. Si es necesario crear otro consejo el usuario debe hacer click en "Añadir consejo" y

aparecerá un nuevo formulario en el que la descripción no puede estar vacía. Una vez hechos los cambios definitivos, para guardarlos, el usuario debe pulsar en "Guardar cambios", y si los datos son correctos, el sistema redirigirá al usuario a la página inicial. Si por el contrario la descripción de algún consejo tiene errores, el sistema mostrará un mensaje de error debajo del campo erróneo.

Figura 35 – Pantalla de edición de consejos nutricionales

A.3 Pantallas del usuario con rol de Paciente

A.3.1 Página principal de Paciente

Si nuestro usuario tiene el rol de Paciente, después de iniciar sesión, en la página principal se mostrará un apartado de avisos llamado "Mis tareas" donde, si fuera necesario, se verá un mensaje donde poder añadir nuevas medidas. Para ello

hacemos click en "Registrar", introducimos el valor en el campo y pulsamos en "Añadir".

Figura 36 – Apartado para añadir nuevas métricas

También veremos el apartado "Mis rutinas", donde dependiendo de si hemos seleccionado la pestaña "Por completar" o "Completadas" nos mostrará una lista de rutinas pendientes de completar o ya realizadas respectivamente. Haciendo click en cualquiera de las filas de esta lista el sistema redirigirá al usuario a los detalles de la rutina seleccionada.

Figura 37 – Apartado de la lista de tareas del paciente

Podremos ver nuestros consejos nutricionales y las métricas de la evolución de los valores que hemos ido añadiendo.

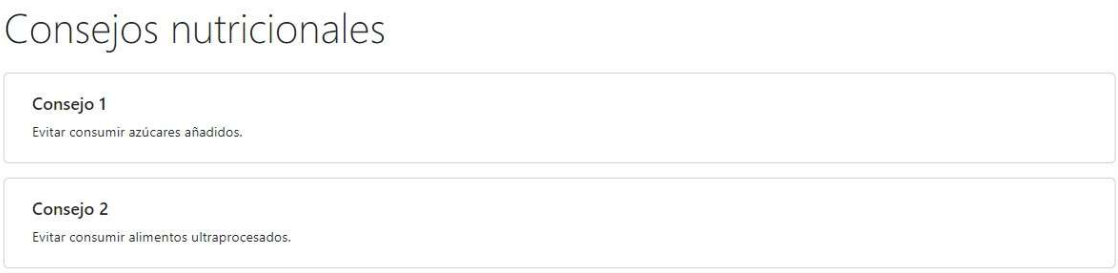

Figura 38 – Apartado de consejos nutricionales

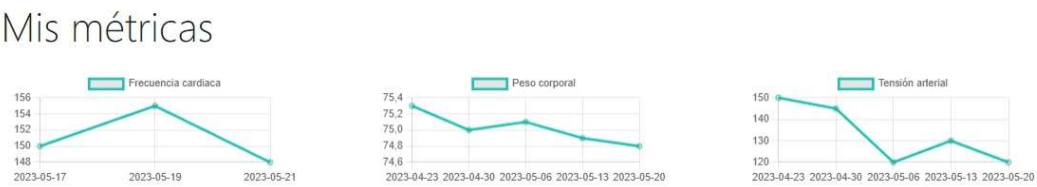

Figura 39 – Apartado de métricas del paciente

A.3.2 Detalles de la rutina

Esta pantalla funciona de la misma manera que la explicada en el apartado A.2.3, por lo que se puede referir a esa sección. La única diferencia es que, en este caso, si la rutina no está completada, aparecerá el botón "Realizar rutina", que al ser pulsado redirigirá al usuario a la página que permite completar la sesión, explicada en el apartado siguiente.

A.3.3 Completar sesión

Una vez el usuario ha accedido a una sesión sin completar aún, el sistema le muestra un formulario con la lista de ejercicios a completar y los detalles de estos como la intensidad y las repeticiones. Una vez realizado un ejercicio, el usuario puede marcarlo como completado pulsando en el botón "No completado", que pasará a aparecer con el mensaje de "Completado" y viceversa.

Figura 40 – Lista de ejercicios a completar

Una vez realizada la sesión de ejercicios, el usuario debe completar las observaciones finales, siendo todos los campos obligatorios excepto el de "Comentarios". Una vez añadido toda la información, el usuario debe pulsar en "Completar rutina" para guardar sus cambios. Si todos los datos son correctos el sistema mostrará la pantalla de detalles de la rutina con los datos actualizados. Por el contrario, si existe algún error en alguno de los campos se mostrará un mensaje de error debajo del campo al que se refiere.

Figura 41 – Formulario de observaciones finales

A.4 Header

El header es común a los dos tipos de usuario, en el existen dos funcionalidades.

A.4.1 Volver a inicio

Si pulsamos en la etiqueta con el logotipo y el nombre de la aplicación que está a la izquierda, el usuario volverá a la página de inicio.

A.4.2 Cerrar sesión

A la derecha de nuestro nombre y nuestro rol en la aplicación, tenemos el botón "Cerrar sesión", si pulsamos en él volveríamos a la página de inicio de sesión.

Figura 42 – Header de la aplicación