Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Desarrollo de una Aplicación Multiplataforma para un *Marketplace* de Asesoramiento Deportivo

Autor: Manuel Arellano Guerrero

Tutor(a): Sergio José Ríos Aguilar

Madrid, junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Desarrollo de una Aplicación Multiplataforma para un Marketplace de Asesoramiento Deportivo
Junio de 2024
Autor: Manuel Arellano Guerrero

Tutor:

Sergio José Ríos Aguilar
Departamento de Ingeniería De Organización, Administración De Empresas Y Estadística
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

El propósito de este Trabajo Fin de Grado es desarrollar una aplicación multiplataforma llamada MyTrainer, concebida como un *marketplace* de asesoramiento deportivo. En el contexto actual, donde la búsqueda de un estilo de vida saludable y activo ha tomado relevancia, surge la necesidad de soluciones innovadoras que faciliten la conexión entre usuarios y asesores deportivos.

MyTrainer proporcionará un servicio integral que permitirá a sus usuarios encontrar asesores deportivos adecuados según sus objetivos, preferencias y presupuesto. Asimismo, los usuarios podrán llevar un seguimiento del progreso, de sus sesiones de entrenamiento y de sus planes nutricionales. Además, de poder recibir las revisiones periódicas de los asesores, entre otras muchas funcionalidades que optimizarán tiempo que dedican a su trabajo los asesores deportivos.

El desarrollo de la aplicación se basa en el uso del *framework* Flutter, asegurando una experiencia de usuario coherente y de alta calidad en múltiples plataformas con una única base de código. Por otra parte, se ha diseñado una arquitectura basada en una solución *Backend as a Service* con Firebase para gestionar la autenticación de usuarios, almacenamiento en tiempo real y sincronización de datos. Finalmente, también se examina el impacto del proyecto sobre los Objetivos de Desarrollo Sostenible de la Agenda 2030.

Palabras clave: Flutter, Firebase, *marketplace*, *fitness,* aplicación multiplataforma.

Abstract

The purpose of this Final Degree Project is to develop a multiplatform application called MyTrainer, designed as a marketplace for sports consulting. In the current context, where the pursuit of a healthy and active lifestyle has gained importance, the need arises for innovative solutions that facilitate the connection between users and sports advisors.

MyTrainer will provide a comprehensive service that will allow its users to find suitable sports advisors according to their goals, preferences, and budget. Additionally, users will be able to track their progress, training sessions, and nutritional plans. Furthermore, they will be able to receive periodic reviews from advisors, among many other functionalities that will optimize the time sports advisors dedicate to their work.

The development of the application is based on the use of the Flutter framework, ensuring a consistent and high-quality user experience on multiple platforms with a single codebase. Additionally, an architecture based on a Backend as a Service solution with Firebase has been designed to manage user authentication, real-time storage, and data synchronization. Finally, the project's impact on the Sustainable Development Goals of the 2030 Agenda is also examined.

Keywords: Flutter, Firebase, marketplace, fitness, multiplatform application.

Tabla de contenidos

1 Introducción

Este capítulo está compuesto por la motivación y la necesidad que se pretende cubrir, sobre la que se basa este proyecto, los objetivos que se plantean alcanzar, la planificación a seguir y la estructura de la memoria.

1.1 Motivación y necesidad del proyecto

En la actualidad, el acceso a información de calidad sobre la salud ha despertado una creciente conciencia en las personas sobre la importancia de cuidar su cuerpo. Dentro de este nuevo escenario, el deporte ha adquirido un protagonismo fundamental siendo uno de los pilares esenciales para aquellas personas que aspiran a mantener, de forma sostenida en el tiempo, un estilo de vida saludable. La búsqueda de una vida más saludable y activa no solo se ha convertido en una tendencia, sino prácticamente en una necesidad imperante para un segmento significativo de la sociedad actual. En este contexto, el papel de las asesorías deportivas cobra una relevancia aún mayor. Estas asesorías, prestadas principalmente de forma telemática por profesionales cualificados del deporte y la nutrición, consisten en un seguimiento y orientación online personalizada del entrenamiento y la nutrición con la meta de alcanzar los objetivos físicos, de salud y rendimiento de cada uno de los clientes. Estos profesionales ofrecen al cliente su apoyo y conocimiento para mantener, de manera simultánea, una rutina de ejercicio y alimentación equilibrada sostenible a largo plazo.

Sin embargo, a medida que la demanda de servicios de asesoramiento deportivo continúa creciendo, surge la necesidad de soluciones innovadoras que puedan satisfacer las cambiantes necesidades de los usuarios. De esta necesidad no cubierta en un mercado en pleno auge surge este proyecto, que consiste en el desarrollo de un *marketplace* de asesoramiento deportivo, donde los usuarios de la aplicación puedan encontrar fácilmente a los profesionales adecuados para el seguimiento de su entrenamiento y nutrición. Esta solución ofrece a los asesores una herramienta con la que podrán optimizar su trabajo y llevar el seguimiento de cada uno de sus clientes.

El plan de negocio de la plataforma será elaborado en mi Trabajo de Fin de Grado paralelo para mi grado en Administración y Dirección de Empresas. De esta forma abordaré el proyecto de forma integral, tanto el plan de negocio de la plataforma desde un enfoqué más relacionado con las competencias de Administración y Dirección de Empresas, como el desarrollo del software necesario.

La plataforma que se propone desarrollar tiene el objetivo de mejorar la forma en que las personas acceden e interactúan actualmente con las asesorías deportivas, ofreciendo una plataforma que permitirá conectar a usuarios con una amplia gama de asesores cualificados especializados en diversas disciplinas deportivas. Independientemente de los objetivos de los usuarios, ya sea que busquen perder peso, ganar masa muscular, mejorar su condición física, prepararse para una competición deportiva o simplemente adoptar un estilo de vida saludable para mejorar su calidad de vida.

La idea sobre la que se basa este proyecto va más allá de simplemente facilitar a los usuarios conectar con los asesores mediante la plataforma. La aplicación ofrecerá una experiencia integral que permitirá llevar un seguimiento del progreso, registrar sesiones de entrenamiento, realizar un seguimiento del plan nutricional y recibir las revisiones periódicas de los asesores de forma sencilla e intuitiva, entre otras muchas funcionalidades. Además, también permitirá encontrar al asesor adecuado que se adapte a los objetivos, preferencias personales y presupuesto de cada individuo. La plataforma aspira a convertirse en una herramienta indispensable para todos aquellos que buscan alcanzar sus objetivos para llevar una vida más saludable y activa.

Por otra parte, la necesidad de garantizar la accesibilidad desde diversas plataformas a la aplicación ha sido el factor clave para la elección de Flutter como *framework* para el desarrollo de la aplicación. Este *framework* permite ofrecer una experiencia de usuario coherente y de alta calidad en distintas plataformas con una única base de código. Facilitando una mayor eficiencia en el desarrollo y mantenimiento de la aplicación, a la vez que garantiza una experiencia uniforme para todos los usuarios, sin importar la plataforma desde la que prefieran utilizar la aplicación.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado será desarrollar una aplicación multiplataforma utilizando el *framework* Flutter para un *marketplace* de asesoramiento deportivo.

Para poder alcanzar el objetivo principal propuesto se deberán cumplir los siguientes objetivos específicos:

- Diseñar una aplicación que se ajuste a las necesidades de los usuarios y que optimice el trabajo de los asesores.
- Diseñar la arquitectura de la aplicación, base de datos e interfaz de usuario.
- Desarrollar funcionalidades clave de la aplicación.
- Utilizar Flutter para crear una aplicación multiplataforma con una única base de código.
- Realizar pruebas de calidad.

1.3 Planificación

Para la planificación de este proyecto se ha elaborado una lista de tareas y un Diagrama de Gantt.

1.3.1 Lista de tareas

Para el desarrollo de este TFG se ha desarrollado la siguiente lista de tareas:

- Toma de contacto con el *framework* Flutter
- Aprendizaje del lenguaje Dart
- Configuración del entorno de desarrollo
- Investigación de necesidades

- Toma de requerimientos
- Diseño de la arquitectura de la aplicación
- Diseño de la interfaz de usuario
- Diseño de la base de datos
- Desarrollo de un prototipo
- Pruebas sobre el prototipo
- Realización diagrama de navegabilidad
- Desarrollo de las funcionalidades clave
- Desarrollo del *backend* de la aplicación
- Pruebas y depuración de errores
- Análisis de *feedback* y realización de ajustes
- Redacción de la memoria
- Elaboración de la presentación

1.3.2 Diagrama de Gantt

Para la gestión de la planificación y establecer un seguimiento del desarrollo de cada una de las tareas presentadas en la lista del apartado anterior, se creó inicialmente el Diagrama de Gantt de la Figura 1. No obstante, el diagrama de Gantt fue ajustado durante el transcurso del proyecto, resultando en la versión definitiva que se presenta en el Anexo A.

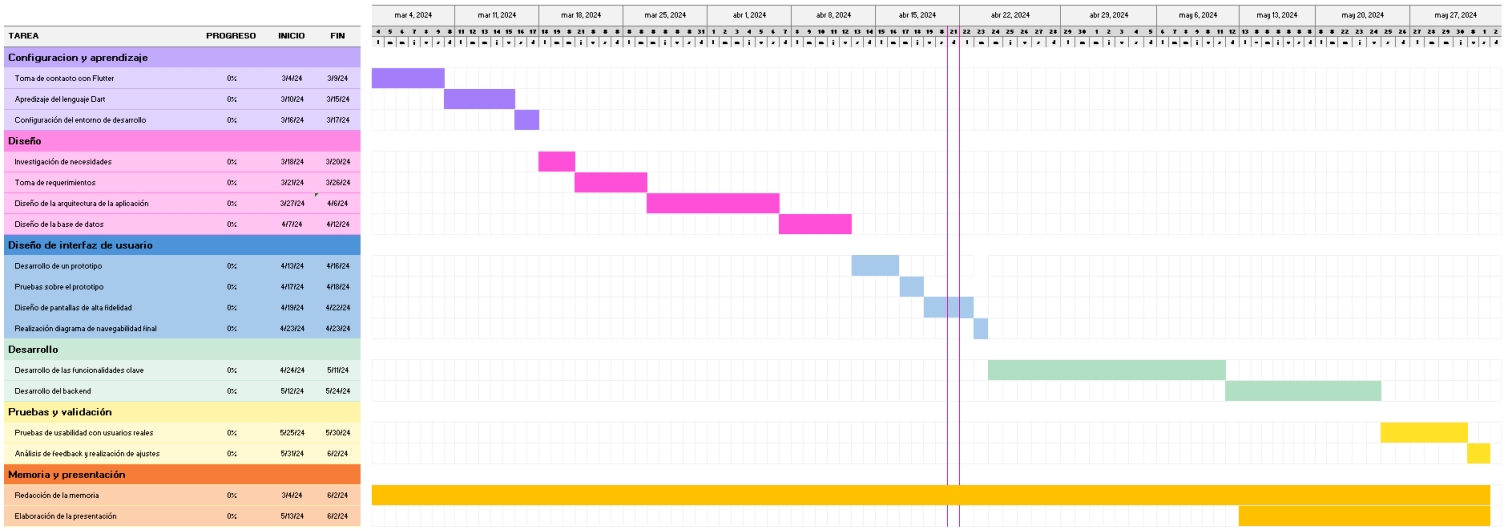


Figura 1: Diagrama de Gantt. Fuente: Elaboración propia.

1.4 Estructura de la memoria

En este apartado de la introducción como está organizada se resume la estructura de la memoria de este proyecto. Esta cuenta con los siguientes capítulos:

2. Estado del arte

   En este capítulo se realiza un análisis previo de las herramientas utilizadas en la actualidad para el desarrollo de aplicaciones multiplataforma.

3. Metodología de trabajo

   En este capítulo se presenta la metodología de trabajo seguida para el desarrollo del proyecto.

4. Análisis de requisitos

   En este capítulo se detallan los requisitos funcionales y no funcionales de la aplicación, describiendo las características que debe tener esta.

5. Diseño

   En este capítulo se expone la arquitectura de la solución, el modelo de datos, los casos de uso y la matriz de requisitos, proporcionando en conjunto una estructura detallada para el desarrollo de la aplicación.

6. Diseño de interfaz de usuario

   En este capítulo se presentan el prototipo de baja fidelidad, las pruebas sobre el prototipo y los diagramas de navegabilidad.

7. Desarrollo

   En este capítulo se expone la preparación y configuración del entorno de desarrollo, la estructura de clases y carpetas, y las principales librerías y extensiones utilizadas, además de los fragmentos clave del código.

8. Resultados y pruebas

   En este capítulo se detallan los resultados finales del desarrollo, las pruebas realizadas sobre la aplicación y el despliegue de esta.

9. Conclusiones y futuros trabajos

   En este capítulo se presentan las conclusiones obtenidas tras la realización del proyecto sobre el trabajo realizado y los trabajos y modificaciones que se podrían llevar en el futuro.

10. Análisis de impacto

    En este capítulo se examinará el impacto potencial de los resultados del proyecto. Además, se examinará la relación del proyecto con los Objetivos de Desarrollo Sostenible de la Agenda 2030.

2 Estado del arte

En este capítulo se realiza un análisis previo de las herramientas utilizadas en la actualidad para el desarrollo de aplicaciones multiplataforma, destacando las opciones seleccionadas para el desarrollo del proyecto.

2.1 Lenguajes de programación

En este contexto de desarrollo de software la selección del lenguaje de programación es un factor clave, ya que cada uno ofrece una capacidades y cualidades distintas. En este apartado se analizan varios lenguajes utilizados actualmente para el desarrollo multiplataforma, valorando sus diferentes características.

2.1.1 JavaScript

JavaScript [1] es un lenguaje comúnmente utilizado en el desarrollo web, aunque también se ha adaptado para utilizar para aplicaciones móviles mediante *frameworks* como React Native [2]. Es uno de los lenguajes con mayor adopción y cuenta con una gran comunidad de desarrolladores, lo que facilita encontrar soporte ante problemas que se den durante el desarrollo. Por otra parte, como desventaja podríamos señalar que cada navegador web interpreta el código en JavaScript de forma diferente, lo que puede provocar resultados inesperados y dificultades de compatibilidad entre diferentes navegadores.

Figura 2: Logo JavaScript. Fuente: Wikipedia.

2.1.2 Kotlin

Kotlin [3] es un lenguaje moderno que ha ganado mucha popularidad en los últimos años. Este lenguaje es usado principalmente para Android y cuenta con el respaldo oficial de Google. Aunque se utiliza primordialmente para aplicaciones Android también permite el desarrollo multiplataforma. Sin embargo, una desventaja de Kotlin es la comunidad de desarrolladores, aunque está creciendo, al tratarse de un lenguaje relativamente nuevo todavía es de tamaño reducido si lo comparamos con otras comunidades más asentadas.

Figura 3: Logo Kotlin. Fuente: Wikipedia.

2.1.3 Dart

Dart [4] es el lenguaje de programación elegido para el desarrollo de este proyecto. Este lenguaje, desarrollado por Google, ha ganado popularidad principalmente gracias a su integración con el *framework* Flutter [5], permitiendo utilizar la misma base de código para distintas plataformas entre las que se encuentran iOS, Android y navegadores web, aspecto crucial para este trabajo. Otra ventaja habría que destacar la compilación JIT (Just-In-Time), que agiliza el desarrollo gracias a la actualización e iteración con la aplicación durante su desarrollo. Esta característica es especialmente útil en las etapas tempranas del desarrollo de aplicaciones, ya que permite a los desarrolladores ver los cambios de inmediato y ajustar su código sobre la marcha.

Por otra parte, Dart cuenta con una sintaxis muy parecida a la de Java, que es el lenguaje con el que tengo más confianza y experiencia. Esto es una ventaja frente a otros lenguajes, ya que me va a facilitar el proceso aprendizaje.

Figura 4: Logo Dart. Fuente: Wikipedia.

2.2 Frameworks

Los *frameworks* en este contexto tienen una importancia crucial para el desarrollo de la aplicación, proporcionando un conjunto de herramientas y librerías que facilitan la programación. En esta sección, se presentan dos *frameworks* muy populares que facilitan el desarrollo de aplicaciones nativas e híbridas.

2.2.1 React Native

React Native [2] es un *framework* destinado al desarrollo de aplicaciones creado por Facebook, que permite el desarrollo de aplicaciones móviles nativas usando JavaScript, destacándose por la capacidad de integrar componentes nativos de cada plataforma.

Figura 5: Logo React Native. Fuente: Wikipedia.

2.2.2 Flutter

Flutter [5] es un *framework*, creado por Google, destinado al desarrollo de aplicaciones con Dart. Este *framework* permite con una misma base de código desarrollar aplicaciones para iOS, Android y navegadores web. Una de sus principales características es que permite compilar directamente con código nativo ya que cuenta con motor de renderizado propio, lo que le permite ofrecer un rendimiento superior frente a otros *frameworks*. Flutter está basado en el uso de *widgets*, que son los componentes que se utilizan para crear la interfaz de usuario, los cuales facilitan el desarrollo de la aplicación ya que su uso permite que el desarrollo sea más intuitivo y eficiente. Por último, otra de las grandes ventajas de Flutter es el *Hot Reload*, funcionalidad que permite a los desarrolladores ver los cambios de manera instantánea en la aplicación sin tener que reiniciar el estado de la aplicación.

Figura 6: Logo Flutter. Fuente: Wikipedia.

2.3 IDEs y editores de código

El entorno de desarrollo es una de las componentes esenciales para la creación de cualquier aplicación no solo para escribir código, sino también para depurar, probar y mantener los proyectos de manera eficiente. En esta sección se presentarán las herramientas que componen el entorno de desarrollo para la consecución del proyecto.

2.3.1 Android Studio

Android Studio [6] es el IDE oficial para desarrollo en Android, esencial para configurar las herramientas de plataforma necesarias y realizar pruebas de aplicaciones Flutter. Será utilizado para la ejecución del emulador que me permitirá probar la aplicación durante su desarrollo.

Figura 7: Logo Visual Studio Code. Fuente: Wikipedia.

2.3.2 Visual Studio Code

Visual Studio Code [7] es un editor de código desarrollado por Microsoft, cuyas principales características son su adaptabilidad y eficiencia. Por otra parte, posee un gran número de extensiones que facilitan la programación. La combinación de Android Studio y Visual Studio Code para el desarrollo del proyecto ofrecerá un equilibrio entre funcionalidad y flexibilidad de desarrollo.

Figura 8: Logo Visual Studio Code. Fuente: Wikipedia.

2.4 Backend

En este subapartado se analizarán varias opciones de *Backend as a Service*, que simplificarán la implementación de funcionalidades críticas como la autenticación de usuarios, el almacenamiento en la nube y procesamiento de datos en tiempo real.

2.4.1 Supabase

Supabase [8] es una plataforma de código abierto que ofrece servicios de *backend*, Alguno de estos servicios son la autenticación, bases de datos en tiempo real y almacenamiento. Otro factor relevante es el uso de PostgreSQL [9], base de datos relacional. Al ser de código abierto, permite a los desarrolladores tener un mayor control sobre la personalización de sus aplicaciones.

Figura 9: Logo Supabase. Fuente: Wikipedia.

2.4.2 Firebase

Firebase [10] es una plataforma de Google que ofrece servicios de *backend*. Especialmente será utilizada en el proyecto para la autenticación de usuarios y almacenamiento en tiempo real. Firebase a diferencia de Supabase utiliza Firestore [11], una base de datos NoSQL que proporciona almacenamiento flexible y escalable. Otra de las razones por la que se ha elegido es por su integración fluida con Flutter a través de FlutterFire [12], que son un conjunto

de *plugins* de Firebase para Flutter que permiten implementar funciones complejas con menos código y mayor eficiencia.

Figura 10: Logo Firebase. Fuente: Wikipedia.

2.5 Diseño de prototipos

Las aplicaciones de esta sección son utilizadas durante la fase de diseño, son cruciales para realizar prototipos que permitan detectar problemas de usabilidad y realizar pruebas con usuarios.

2.5.1 Adobe XD

Adobe XD [13] es una herramienta de pago desarrollada por Adobe, que permite el diseño y prototipado de interfaces de usuario. Donde realmente destaca esta aplicación frente a otras herramientas de prototipado es en el diseño de interfaces de alta fidelidad.

Figura 11: Logo Adobe XD. Fuente: Wikipedia.

2.5.2 Figma

Figma [14] es una herramienta de diseño de interfaces de usuario basada en la nube que facilita la colaboración entre usuarios y la compatibilidad con múltiples plataformas. Que cuente con versión gratuita y mi experiencia usándola previamente, han sido factores determinantes para seleccionarla como la aplicación con la que realizaré el prototipo de la aplicación.

Figura 12: Logo Figma. Fuente: Wikipedia.

3 Metodología de trabajo

En este capítulo se hará una pequeña pero necesaria descripción de la metodología de trabajo seguida para el desarrollo del proyecto. Tras un análisis exhaustivo de las diferentes opciones, finalmente se optó por la metodología en cascada [15] para el desarrollo del software de la aplicación de este proyecto. Esta elección se fundamenta en las facilidades que otorga su estructura lineal, clara y organizada

Esta metodología, desarrollada por el Dr. Winston W. Royce, propone un modelo secuencial de gestión de proyectos de desarrollo de software, dividido en distintas fases. En la actualidad, existen diferentes versiones del modelo que estructuran de distintas formas las fases del modelo en cascada [16]. La Figura 13 muestra las distintas etapas que componen el ciclo de vida de este proyecto.

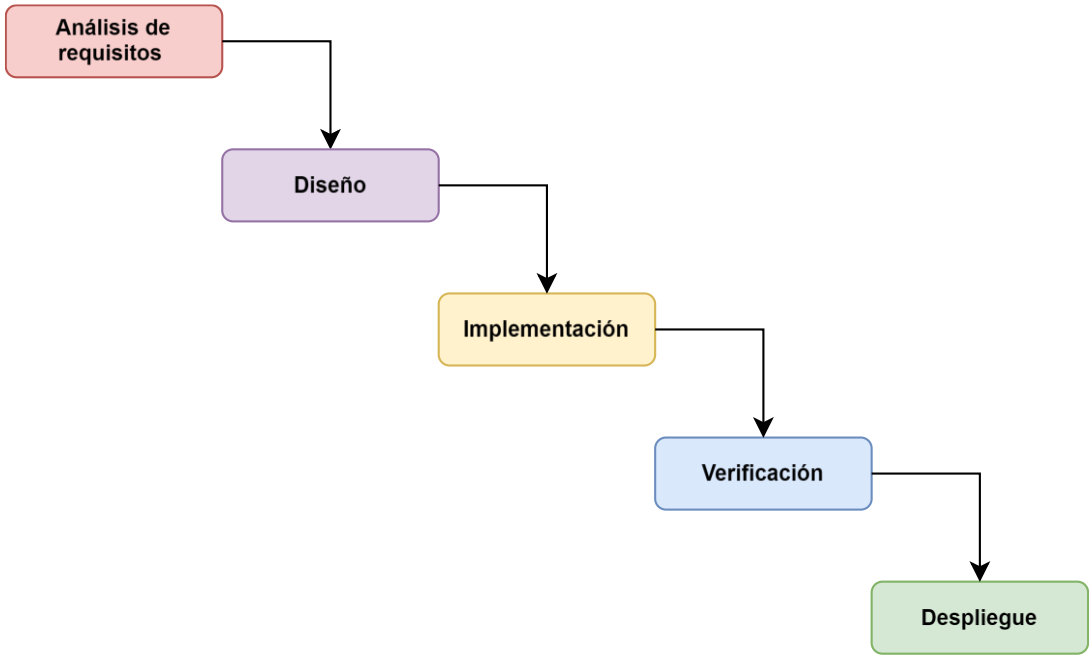


Figura 13: Metodología utilizada, modelo en cascada de 5 fases. Fuente: Elaboración propia.

En cada etapa del ciclo de vida del proyecto se realizan las tareas necesarias avanzando, de forma secuencial, a la siguiente fase únicamente cuando se ha completado la anterior. El modelo en cascada nos proporciona una estructura lineal para el proyecto, lo que facilita su comprensión y aporta claridad en el desarrollo de este.

No obstante, a pesar de las ventajas que tiene esta metodología también tiene puntos débiles, siendo el principal su rigidez, ya que una vez iniciada una nueva fase no se permiten cambios en fases anteriores. Para reducir el impacto de estas desventajas se ha llevado a cabo dos iteraciones del modelo: la primera de ellas para el desarrollo de un prototipo y la segunda para el software final del proyecto.

4 Análisis de requisitos

En este capítulo se especificarán los requisitos que deberá cumplir el software de MyTrainer, que es la aplicación objeto de desarrollo de este proyecto. Se especificarán tanto los requisitos funcionales que describen funcionalidades específicas que la aplicación debe proporcionar, como los requisitos no funcionales que definen otros aspectos a satisfacer por el sistema.

4.1 Requisitos funcionales

RQ1. Iniciar sesión.

La aplicación debe permitir a los usuarios iniciar sesión ingresando su correo electrónico y contraseña para acceder a sus cuentas.

RQ2. Cerrar sesión.

La aplicación debe ofrecer a los usuarios la opción de cerrar sesión cuando no estén utilizando la aplicación.

RQ3. Modificar información del perfil del usuario.

La aplicación debe permitir a los usuarios ver y modificar su información personal en el perfil, incluyendo nombre, apellidos y otros datos relevantes.

RQ4. Eliminar cuenta.

La aplicación debe proporcionar una opción para que los usuarios puedan eliminar su cuenta de manera permanente, incluyendo todos sus datos personales.

4.1.1 Específicos asesorados

RQ5. Crear usuario asesorado.

La aplicación debe permitir el registro de una cuenta para nuevos usuarios que quieran empezar a contar con los servicios de los asesores disponibles en MyTrainer. En el registro se deberá solicitar la siguiente información: nombre, apellidos, correo electrónico, contraseña y comprobación de contraseña.

RQ6. Búsqueda de asesores.

La aplicación permitirá a los usuarios con cuenta de asesorados la búsqueda de asesores deportivos filtrando por especialidad, titulación, valoración, precio o frecuencia de revisión.

RQ7. Ver perfil del asesor.

Los usuarios deben poder ver los perfiles de los asesores, incluyendo su experiencia, certificaciones, tarifas y valoraciones de otros asesorados.

RQ8. Selección de asesor.

La aplicación debe permitir a los usuarios seleccionar los servicios del asesor que deseen.

RQ9. Enviar cuestionario inicial.

Los asesorados, una vez hayan seleccionado los servicios de un asesor determinado, deben ser capaces de completar y enviar el cuestionario inicial que establezca el asesor. Este cuestionario contiene la información necesaria para que el asesor elabore sus planes de entrenamiento y nutrición personalizados.

RQ10. Acceder al plan de entrenamiento.

Los asesorados deben poder acceder a sus planes de entrenamiento personalizados proporcionados por sus asesores.

RQ11. Acceder al plan nutricional.

Los asesorados deben poder acceder a sus planes nutricionales personalizados proporcionados por sus asesores.

RQ12. Acceder al panel de inicio.

La aplicación debe proporcionar un panel de inicio donde los asesorados puedan ver sus tareas pendientes, acceder a sus planes y registrar las métricas de salud necesarias.

RQ13. Registrar entrenamientos.

Los asesorados deben poder registrar sus sesiones de entrenamiento completadas en la aplicación.

RQ14. Registrar peso.

Los asesorados deben poder registrar su peso diariamente para llevar un seguimiento de su progreso.

RQ15. Registrar pasos.

Los asesorados deben poder registrar la cantidad de pasos que dan diariamente.

RQ16. Registrar consumo de agua.

Los asesorados deben poder registrar su consumo de agua diario.

RQ17. Registrar horas de sueño.

Los asesorados deben poder registrar sus horas de sueño diario para monitorear sus patrones de descanso.

RQ18. Enviar y recibir mensajes al entrenador.

La aplicación debe permitir a los asesorados enviar mensajes y recibir mensajes de sus asesores para realizar consultas y recibir *feedback*.

RQ19. Recibir revisión periódica del entrenador.

Los asesorados deben poder recibir revisiones periódicas de sus asesores sobre su progreso, que podrán ser semanales, bisemanales o mensuales. Estas revisiones consisten en recomendaciones y modificaciones, si los asesores lo consideran necesario, de sus planes nutricionales y de entrenamiento.

RQ20. Valorar asesores.

Los asesorados deben poder valorar a los asesores que les hayan prestado servicios durante más de un mes. Las valoraciones deben ser anónimas y tienen como objetivo ayudar a otros clientes durante el proceso de selección del asesor adecuado.

RQ21. Acceder a un *dashboard* con el progreso.

Los clientes deben tener acceso a un *dashboard* donde puedan ver gráficos y estadísticas de su progreso en el entrenamiento y su nutrición.

RQ22. Acceder a un calendario.

Los clientes deben tener acceso a un calendario que les permita visualizar los eventos pasados y programados.

RQ23. Abandonar asesoría.

La aplicación debe permitir a los clientes abandonar o cancelar sus asesorías en cualquier momento.

4.1.2 Específicos asesores

RQ24. Crear usuario asesor.

La aplicación debe permitir el registro de una cuenta para nuevos asesores que quieran empezar a prestar servicios de asesoría deportiva en MyTrainer. En el registro se deberá solicitar la siguiente información: nombre, apellidos, correo electrónico, contraseña y comprobación de contraseña.

RQ25. Actualizar perfil profesional.

Los asesores deben poder actualizar su perfil profesional, que será visible para los clientes, incluyendo aspectos como la descripción del servicio, la experiencia previa, las certificaciones con las que cuentan, sus especialidades, sus tarifas o fotos de sus casos de éxito.

RQ26. Crear cuestionario inicial.

Los asesores deben poder crear un cuestionario inicial para que sea completado por los nuevos asesorados cuando les seleccionen para comenzar una nueva asesoría. Este cuestionario debe incluir información relevante como objetivos, historial médico, preferencias personales para elaborar la dieta y disponibilidad horaria para entrenar, entre otra información que el asesor podrá solicitar.

RQ27. Recibir cuestionario inicial.

El sistema debe avisar automáticamente al asesor cuando un nuevo cuestionario sea completado por un nuevo cliente, permitiéndole revisar la información para que pueda personalizar los planes de entrenamiento y nutricional.

RQ28. Enviar y recibir mensajes del asesorado.

La aplicación debe permitir a los asesores enviar y recibir mensajes de sus clientes para mantener una comunicación fluida mediante la cual poder dar *feedback* y resolver dudas.

RQ29. Recibir y enviar revisión periódica del asesorado.

Los asesores deben recibir datos periódicos de sus clientes para realizar una revisión periódica. Además, los asesores deben poder enviar estas revisiones en el período establecido.

RQ30. Crear plan de entrenamiento.

Los asesores deben poder crear planes de entrenamiento personalizados para sus clientes, indicando el número de sesiones, el número de ejercicios, el número de series, las repeticiones y los descansos.

RQ31. Modificar planes de entrenamiento.

Los asesores deben poder modificar los planes de entrenamiento existentes de sus clientes según sea necesario.

RQ32. Crear plan nutricional.

Los asesores deben poder crear planes nutricionales personalizados para sus clientes.

RQ33. Modificar planes nutricionales.

Los asesores deben poder modificar los planes nutricionales existentes según sea necesario.

RQ34. Acceder a datos históricos del cliente.

Los asesores deben tener acceso a las métricas y datos históricos de sus clientes para evaluar el progreso a largo plazo.

RQ35. Acceder a un calendario.

Los asesores deben tener acceso a un calendario que les permita visualizar los eventos pasados y programados, como las fechas límite para la entrega de las revisiones periódicas de cada uno de sus clientes.

RQ36. Acceder al panel de inicio de gestión de clientes.

La aplicación debe proporcionar a los asesores una pantalla con información para la gestión de sus clientes, donde poder ver: el número de clientes activos, las revisiones pendientes, el número de nuevos clientes y el número de visualizaciones que tiene su perfil.

RQ37. Acceso al *dashboard* del progreso individual de cada cliente.

Los asesores deben tener acceso a un *dashboard* donde puedan ver el progreso individual de cada uno de sus clientes, que les permita tomar decisiones sobre los planes de sus clientes.

4.2 Requisitos no funcionales

Seguridad

La aplicación debe cumplir con los estándares de seguridad para proteger los datos personales de los usuarios, incluyendo medidas de autenticación.

Usabilidad

La aplicación debe ser fácil de usar e intuitiva, proporcionando una experiencia de usuario fluida y accesible para todos los niveles de usuario, independientemente de su habilidad con las nuevas tecnologías.

Consistencia

La aplicación debe mantener una consistencia en el diseño y la funcionalidad en todas sus plataformas. Además, será fundamental que la información del

usuario, como planes de entrenamiento, registros de progreso y comunicaciones con el asesor, sea consistente y esté sincronizada en todas las plataformas.

Rendimiento óptimo

La aplicación debe ser rápida y eficiente, asegurando tiempos de carga mínimos y un rendimiento óptimo incluso con un alto número de usuarios concurrentes.

5 Diseño

En este capítulo se llevará a cabo el proceso de diseño de la aplicación MyTrainer. El objetivo principal del capítulo es detallar cómo se ha estructurado y organizado la aplicación para cumplir con los requisitos funcionales y no funcionales definidos anteriormente. Se abordarán aspectos clave como la arquitectura de la aplicación, el modelo de datos utilizado, los casos de uso identificados y la matriz de requisitos que guía el desarrollo del proyecto.

5.1 Arquitectura de la aplicación

La Figura 14 proporciona una representación simple de la arquitectura de la aplicación de MyTrainer. Como se puede observar la arquitectura está claramente dividida en dos partes principales: el *frontend* y el *backend*.

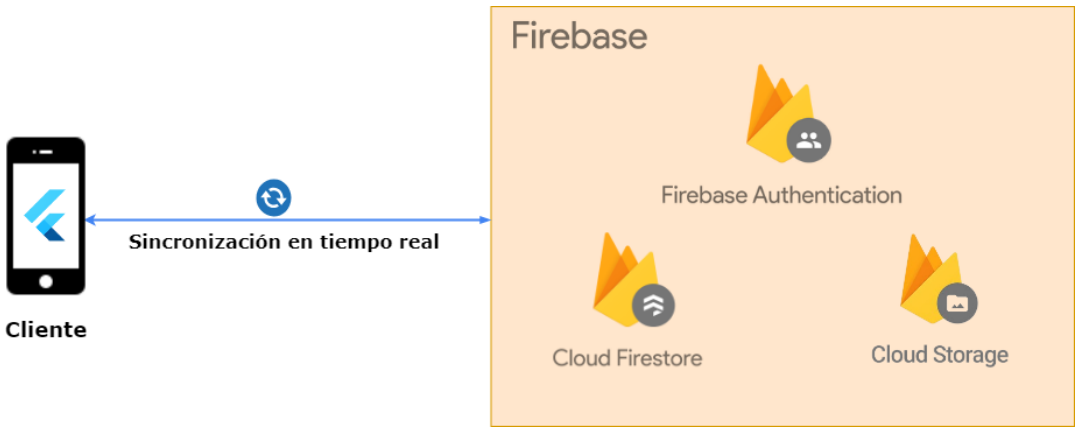


Figura 14: Arquitectura de la aplicación. Fuente: Elaboración propia.

El *frontend* es la parte de una aplicación con la que los usuarios interactúan directamente, es decir, toda la capa visual que los usuarios usan para navegar por la aplicación. El *frontend* de MyTrainer está desarrollado en Flutter, un *framework*, creado por Google, destinado a la creación de aplicaciones con Dart. Este *framework* nos permitirá con una misma base de código crear aplicaciones para múltiples plataformas, lo que optimizará el tiempo invertido en el desarrollo de la interfaz de usuario [17].

Por otro lado, el *backend* es la parte de la aplicación, ejecutada en el servidor, responsable de la lógica de negocio, la gestión de bases de datos, la autenticación de usuarios y otras funciones que no son visibles para el usuario final [17]. En la Figura 14 no se muestra ningún servidor, esto es debido a que se ha optado por una arquitectura *serverless* [18]. Este tipo de arquitecturas representan una evolución significativa en el desarrollo de aplicaciones, ya que permiten construir y ejecutar aplicaciones sin necesidad de gestionar la infraestructura del servidor gracias a la utilización de servicios en la nube, que ejecutan el código en respuesta a eventos específicos y administran automáticamente los recursos necesarios para la ejecución de dicho código.

Backend as a Service, BaaS, es el nombre comúnmente usado para referirse a las plataformas que prestan este tipo de servicios. Para el desarrollo de la aplicación de MyTrainer, se ha optado por utilizar Firebase [10] como la solución BaaS. Firebase es una plataforma de Google que permite simplificar el desarrollo del *backend*, eliminando la necesidad de gestionar servidores y proporcionando un sistema altamente escalable y seguro. Firebase se utilizará para gestionar la seguridad y el acceso de los usuarios, así como para administrar la base de datos y sincronizar los datos en tiempo real de la aplicación de MyTrainer.

Por un lado, utilizar un BaaS facilitará mantener el foco en el desarrollo de las funcionalidades clave de la aplicación sin tener que desarrollar y mantener un *backend* completo desde cero, reduciendo significativamente el tiempo necesario para desarrollar la aplicación. Sin embargo, también existen ciertas desventajas derivadas del uso de un BaaS, como la dependencia del proveedor. El funcionamiento de la aplicación depende completamente de la infraestructura de Firebase, por lo que cualquier problema con los servicios de Google podría afectar a la disponibilidad y rendimiento de la aplicación de MyTrainer.

5.2 Firebase

Como se ha anticipado en el apartado anterior, Firebase una herramienta de *Backend as a Service.* La plataforma proporciona una amplia gama de servicios a diferentes niveles, siendo los utilizados en este proyecto los siguientes:

- **Firebase Authentication**, proporciona servicios de autenticación y autorización, permitiendo gestionar el acceso de usuarios de manera segura. Se utilizará para facilitar la gestión de la autenticación y garantizar la seguridad de los usuarios de la aplicación. Permitirá implementar un sistema de autenticación robusto y seguro sin necesidad de desarrollar una solución desde cero.
- **Cloud Firestore**, es una base de datos NoSQL en la nube orientada a documentos, diseñada para almacenar datos estructurados y sincronizarlos en tiempo real entre dispositivos. La elección de Firestore para el almacenamiento de los datos de los usuarios está basada en su capacidad para proporcionar almacenamiento flexible y escalable, simplificando el manejo de datos estructurados y su sincronización en tiempo real. Además, su integración fluida con Flutter a través de FlutterFire, un conjunto de *plugins* de Firebase para Flutter, permite implementar funciones complejas con menos código y mayor eficiencia.
- **Cloud Storage**, es una solución diseñada para almacenar archivos grandes como imágenes, videos y archivos de usuario. Esta herramienta ha sido seleccionada debido a la necesidad de almacenar y gestionar imágenes dentro de la aplicación MyTrainer, esta herramienta nos ofrecerá una gestión eficiente de los recursos multimedia, asegurando su disponibilidad y accesibilidad para los usuarios de la aplicación.

5.3 Modelo de datos

Tal como se mencionó en el subapartado anterior, en este proyecto se utilizará Firebase Firestore como la base de datos principal. Esta base de datos NoSQL,

consta de una estructura flexible que permitirá almacenar y organizar datos de manera dinámica, que a diferencia de las bases de datos SQL tradicionales, no requiere de un esquema rígido. Este factor puede considerarse una ventaja significativa, ya que permite que el modelo de datos pueda adaptarse y evolucionar a lo largo del desarrollo del proyecto.

Firestore se basa en un modelo de datos jerárquico de documentos y colecciones, permitiendo la creación de colecciones anidadas, facilitando así el acceso a los datos. Las colecciones son contenedores que almacenan documentos. Cada colección puede contener un número ilimitado de documentos, pero no directamente otras colecciones. En lugar de ello, los documentos pueden referenciar otras colecciones. Los documentos son las unidades de almacenamiento de datos fundamentales en Firestore y actúan como contenedores de pares clave-valor, similares a los objetos JSON. Cada documento tiene un identificador único dentro de su colección y puede contener datos primitivos, *arrays*, *maps* y referencias a otros documentos. Por otro lado, Firestore permite realizar *queries* de forma sencilla, facilitando el desarrollo y la gestión de la base de datos.

En el contexto de este proyecto, la base de datos se organiza en tres colecciones principales: "users", "chats" y "hires". Cada una de estas colecciones almacena información específica correspondiente a usuarios, chats y contrataciones. Esta estructura permite un acceso rápido y eficiente a los datos relevantes para cada función del proyecto. A continuación, se desarrolla el contenido de los documentos de cada una de estas tres colecciones.

Campos de los documentos de la colección "*users*":

- *uid*: identificador único del usuario.
- *email*: correo electrónico del usuario.
- *password*: contraseña del usuario.
- *fname*: nombre del usuario.
- *sname*: apellidos del usuario.
- *rol*: rol del usuario ("Asesor" o "Asesorado").

Campos exclusivos de usuarios con rol "Asesor":

  - *fotoperfil*: URL a Cloud Storage done se encuentra la foto de perfil del usuario.
  - *description*: descripción de los servicios del asesor.
  - *price*: precio del servicio del usuario.
  - *frequency*: frecuencia de revisión de la asesoría ofrecida ("Semanal", "Bisemanal" o "Mensual").
  - *certificates*: lista con los nombres de las titulaciones del asesor.
  - *specialities*: lista con las especialidades del asesor.
  - *isAdVisible*: campo *booleano* que indica si el anuncio del asesor se encuentra visible al resto de usuarios o no.
  - *rating*: media de las valoraciones a los servicios del asesor por parte de sus asesorados.
  - *profileViews*: número histórico de visitas al perfil profesional del asesor.
  - *questions*: lista con las preguntas del cuestionario inicial de asesor.
  - *gallery*: lista que contiene las URLs a Cloud Storage de las imágenes que pueden incluir los asesores en su perfil profesional.

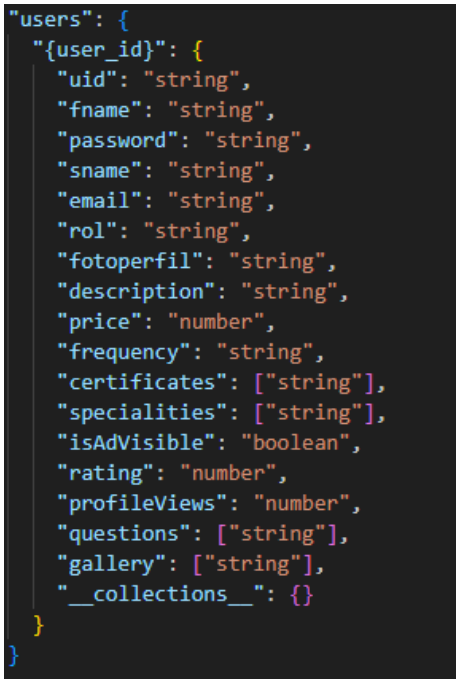


Figura 15: Campos de los documentos de la colección "users" y sus tipos de datos. Fuente: Elaboración propia.

Campos de los documentos de la colección "*hires*":

- *clientId:* identificador único del cliente que disfruta de la asesoría.
- *trainerId:* identificador único del asesor que presta servicios al cliente.
- *responses: map* que contiene las preguntas y respuestas dadas por el asesor al cuestionario inicial, siendo la preguntas la *key* y las respuestas el *value*.
- *timestamp:* marca temporal del instante de contratación de la asesoría.
- *isCancelled:* campo *booleano* que indica si la contratación ha sido cancelada.
- *isNew:* campo *booleano* que indica si la contratación es nueva.
- *pendingReview:* campo booleano que indica si la contratación está pendiente de revisión.
- *goals: map* que contiene los objetivos establecidos por el asesor que debe alcanzar del cliente en cuanto a sueño, peso, agua y pasos. Siendo la variable en cuestión la *key* y el valor a alcanzar el *value.*
- *meals:* campo que contiene la planificación activa de comidas del cliente por día de la semana. Es un *map* que contiene un par *key-value* por cada día de la semana, siendo la *key* el día de la semana y el *value* un *map* con las comidas que debe realizar el cliente ese día de la semana. Este *map* tiene como *key* el nombre que le desee dar el asesor a la comida ("Desayuno", "Merienda", "Cena", etc.) y como *value* un *map* llamado "ítems" que contiene los alimentos que forma dicha comida, siendo la *key* el nombre con la cantidad de gramos y el *value* las calorías del respectivo alimento. Además, este *map* contiene un par *key-value* adicional, donde la *key* es "orden" y el *value* es la posición de la comida a lo largo del día (por ejemplo, 1 para desayuno, 2 para almuerzo, etc.).
- *workoutPlan:* campo que contiene el plan de entrenamiento activo del cliente por día de la semana. Es un *map* que contiene un par *key-value* por cada día de la semana, siendo la *key* el día de la semana y el *value*

un *map* con los ejercicios planeados para ese día de la semana. Este *map* tiene como *key* el nombre del ejercicio y como *value* un *map* que incluye los siguientes pares *key-value*:
    - *orden:* número que indica la posición del ejercicio en la secuencia de entrenamiento del día.
    - *sets:* determina el número total de series que se deben completar para el ejercicio.
    - *reps:* especifica el número de repeticiones que se deben realizar en cada serie del ejercicio.
    - *rest:* define el tiempo de descanso en segundos entre series del ejercicio.
- *rating*: campo opcional que contiene la calificación establecida por el cliente del servicio prestado por el asesor.

Figura 16: Campos de los documentos de la colección "hires" y sus tipos de datos. Fuente: Elaboración propia.

Campos de los documentos de la colección anidada "dailyActivity":

- *date:* fecha de la actividad registrada en formato ISO 8601 que sigue la estructura "AAAA-MM-DD".
- *weekday:* día de la semana.
- *sleep:* cantidad de horas de sueño, con decimales, en el día.

- *weight:* peso del cliente en el día.
- *steps:* cantidad de pasos dados en el día.
- *water:* cantidad de agua consumida en mililitros en el día.

Campos de los documentos de la colección anidada "meals":

- *orden:* número que indica el orden de la comida en el día.
- *title:* nombre de la comida.
- *items: map* que contiene los alimentos que forma dicha comida, siendo la *key* el nombre con la cantidad de gramos y el *value* las calorías del respectivo alimento.

Campos de los documentos de la colección anidada "workoutPlan":

- *orden:* número entero que representa la posición del ejercicio en la secuencia de entrenamiento del día.
- *title:* nombre del ejercicio.
- *sets:* determina el número total de series que se deben completar para el ejercicio.
- *reps:* especifica el número de repeticiones que se deben realizar en cada serie del ejercicio.
- *rest:* define el tiempo de descanso en segundos entre series del ejercicio.

Figura 17: Campos de los documentos de la colección "dailyActivity" y sus tipos de datos. Fuente: Elaboración propia.

Campos de los documentos de la colección "*chats*":

- *{chat_id}:* Identificador del documento del chat, formado por el identificador de dos usuarios separados por un guion bajo.

Campos de los documentos de la colección anidada "messages":

- senderID: identificador único del usuario que envía el mensaje.
- receiverID: identificador único del usuario que recibe el mensaje.
- message: contenido del mensaje.
- timestamp: marca temporal del instante del mensaje.

Figura 18: Campos de los documentos de la colección "chats" y sus tipos de datos. Fuente: Elaboración propia.

5.4 Casos de uso

En este apartado se presentan los distintos casos de uso de la aplicación de MyTrainer, que ayudarán a entender mejor las funcionalidades requeridas y cómo se integran en el flujo general de la aplicación. En cada caso de uso se define un escenario con una serie de pasos detallados que especifican como deben interactuar los usuarios con el sistema para lograr el objetivo deseado, además se establecen las condiciones previas y posteriores a la ejecución de cada proceso.

CU1. Registro de usuario.

- **Requisitos:** RQ5, R24.
- **Descripción**: El usuario se registra en la aplicación proporcionando sus datos personales y creando una cuenta.
- **Actores**: Asesor, Asesorado.
- **Precondiciones**: El usuario debe tener acceso a internet.
- **Flujo principal**:
    1. El usuario selecciona la opción de "Crea una cuenta".
    2. La aplicación presenta un formulario de registro.
    3. El usuario completa el formulario con su nombre, correo electrónico y dos veces la contraseña. Además, indicará si se quiere registrar como asesor o asesorado.
    4. El usuario envía el formulario.
    5. La aplicación verifica la información y crea una cuenta nueva para el usuario.

6. La aplicación muestra una confirmación de éxito del registro.

- **Postcondiciones**: La cuenta del usuario se ha creado y el usuario puede iniciar sesión.
- **Excepciones**:
    1. Si el correo electrónico ya está registrado, se mostrará un mensaje de error y se solicitará un correo diferente.
    2. Si ambas contraseñas no coinciden, se mostrará un mensaje de error.

CU2. Iniciar sesión.

- **Requisitos:** R1.
- **Descripción**: El usuario se autentica en la aplicación.
- **Actores**: Asesor, Asesorado.
- **Precondiciones**: El usuario debe estar registrado, no estar autenticado y tener acceso a internet.
- **Flujo principal**:
    1. El usuario selecciona la opción de "Iniciar sesión".
    2. La aplicación muestra un formulario de inicio de sesión.
    3. El usuario introduce su correo electrónico y contraseña.
    4. El usuario envía el formulario.
    5. La aplicación verifica las credenciales. Si las credenciales son correctas, el usuario accede a su cuenta.
- **Postcondiciones**: El usuario ha iniciado sesión y puede acceder a su cuenta.
- **Excepciones**:
    1. Si las credenciales son incorrectas, se mostrará un mensaje de error.

CU3. Modificar información del perfil.

- **Requisitos:** RQ3.
- **Descripción**: El usuario ve y modifica la información personal de su cuenta desde la sección de perfil.
- **Actores**: Asesor, Asesorado.
- **Precondiciones**: El usuario debe tener acceso a internet y haber iniciado sesión.
- **Flujo principal**:
    1. El usuario accede a la sección de perfil.
    2. La aplicación muestra la información actual del perfil.
    3. El usuario edita la información deseada.
    4. El usuario guarda los cambios.
    5. La aplicación actualiza la información del perfil.
- **Postcondiciones**: La información del perfil del usuario se ha actualizado.
- **Excepciones**: N/A.

CU4. Cerrar sesión.

- **Requisitos:** RQ2.
- **Descripción**: El usuario cierra sesión.
- **Actores**: Asesor, Asesorado.
- **Precondiciones**: El usuario debe estar autenticado en la aplicación y tener acceso a internet.
- **Flujo principal**:
    1. El usuario accede a la sección de perfil.

2. La aplicación muestra la información actual del perfil y la opción de “Cerrar sesión”.
3. El usuario selecciona la opción de “Cerrar sesión”.
4. La aplicación cierra la sesión y redirige al usuario a la pantalla de inicio de sesión.

- **Postcondiciones**: La sesión del usuario se ha cerrado y ya no está autenticado.
- **Excepciones**: N/A.

CU5. Eliminar cuenta.

- **Requisitos:** RQ4.
- **Descripción**: El usuario elimina su cuenta de manera permanente.
- **Actores**: Asesor, Asesorado.
- **Precondiciones**: El usuario debe tener una cuenta, acceso a internet y estar autenticado.
- **Flujo principal**:
    1. El usuario accede a la sección de perfil.
    2. El usuario selecciona la opción de "Eliminar Cuenta".
    3. La aplicación solicita confirmación.
    4. El usuario confirma la eliminación de la cuenta.
    5. La aplicación elimina la cuenta.
    6. La aplicación redirige al usuario a la pantalla de inicio de sesión.
- **Postcondiciones**: La cuenta del usuario ha sido eliminada permanentemente. Si estaba en una asesoría, esta no se renovará el próximo mes.
- **Excepciones**: N/A.

CU6. Búsqueda y contratación de asesores deportivos.

- **Requisitos:** RQ6, RQ7, RQ8, RQ9.
- **Descripción**: El usuario busca, visualiza perfiles asesores, selecciona el que crea que mejor se adapta a sus necesidades y le envía el cuestionario inicial para que el asesor preparé sus planes iniciales.
- **Actores**: Asesorado.
- **Precondiciones**: El usuario debe estar autenticado en la aplicación y tener acceso a internet.
- **Flujo principal**:
    1. El usuario accede a la pantalla de búsqueda de asesores.
    2. El usuario introduce los filtros y criterios de búsqueda.
    3. La aplicación muestra los resultados que coinciden con los criterios.
    4. El usuario selecciona un asesor de la lista de resultados de búsqueda.
    5. La aplicación muestra el perfil detallado del asesor.
    6. El usuario decide contratar los servicios del asesor.
    7. El usuario selecciona el botón de contratar en el perfil del asesor.
    8. La aplicación muestra el cuestionario inicial al usuario.
    9. El usuario completa el cuestionario con la información solicitada.
    10. El usuario envía el cuestionario.
    11. La aplicación notifica al asesor sobre la adquisición de un nuevo cliente.
- **Postcondiciones**: La cuenta del usuario se ha creado y el usuario puede iniciar sesión.

- **Excepciones**:
    1. Si no hay resultados que coincidan con los criterios, se muestra un mensaje indicando que no se encontraron asesores.
    2. Si el usuario se encontraba ya en una asesoría, se muestra un mensaje indicando la situación en la que se encuentra y hasta que fecha.

CU7. Acceso a los planes personalizados.

- **Requisitos:** RQ10, RQ11, RQ12, RQ13.
- **Descripción**: El usuario accede a sus planes de entrenamiento, y nutricionales personalizados proporcionados por su asesor desde el panel de inicio.
- **Actores**: Asesorado.
- **Precondiciones**: El usuario debe estar autenticado en la aplicación, debe tener una asesoría contratada y tener acceso a internet.
- **Flujo principal**:
    1. El usuario accede a la pantalla de inicio.
    2. El usuario selecciona la opción de "Plan de Entrenamiento".
    3. La aplicación muestra el plan de entrenamiento personalizado del usuario para el día actual.
    4. El usuario revisa el entrenamiento.
    5. El usuario comienza el entrenamiento.
    6. El usuario registra el entrenamiento.
    7. El usuario vuelve a la pantalla de inicio.
    8. El usuario selecciona la opción de "Plan Nutricional".
    9. La aplicación muestra el plan nutricional personalizado del usuario.
    10. El usuario revisa el plan nutricional.
- **Postcondiciones**: El usuario ha accedido a su panel de inicio, registrado su entrenamiento y revisado sus planes de entrenamiento y nutricionales personalizados.
- **Excepciones**:
    1. Si el usuario no cuenta con los servicios de ningún asesor, se mostrará un mensaje informándole de la situación.

CU8. Registro diario de métricas de salud.

- **Requisitos:** RQ14, RQ15, RQ16, RQ17.
- **Descripción**: El usuario registra sus métricas de salud del día actual.
- **Actores**: Asesorado.
- **Precondiciones**: El usuario debe estar autenticado en la aplicación, debe tener una asesoría contratada y tener acceso a internet**.**
- **Flujo principal**:
    1. El usuario accede a la pantalla de inicio.
    2. El usuario introduce su peso actual.
    3. El usuario introduce el número de pasos dados en el día.
    4. El usuario introduce la cantidad de agua consumida en el día.
    5. El usuario introduce el número de horas dormidas.
    6. El usuario guarda el registro.
    7. La aplicación actualiza el historial del usuario.
- **Postcondiciones**: Los datos de peso, pasos, consumo de agua y horas de sueño se han registrado correctamente.
- **Excepciones**: N/A.

CU9. Enviar y recibir mensajes.

- **Requisitos:** RQ18, RQ28.
- **Descripción**: El usuario recibe y envía mensajes.
- **Actores**: Asesor, Asesorado.
- **Precondiciones**: Los usuarios deben estar autenticados en la aplicación, tener acceso a internet y ser asesor y asesorado.
- **Flujo principal**:
    1. El usuario accede a la sección de mensajes.
    2. El usuario selecciona el contacto al que desea enviar un mensaje.
    3. El usuario lee los últimos mensajes enviados por el contacto seleccionado.
    4. El usuario escribe y envía el mensaje.
- **Postcondiciones**: Los usuarios pueden mantener una comunicación efectiva a través de la aplicación.
- **Excepciones**:
    1. Si el usuario intenta enviar un mensaje a alguien con quien no tiene una relación establecida, se mostrará un mensaje de error indicando que no puede enviar mensajes a este contacto.

CU10. Recibir revisión periódica.

- **Requisitos:** RQ19.
- **Descripción**: La aplicación notifica al asesorado que ya tiene una nueva revisión de sus planes disponible.
- **Actores**: Asesorado.
- **Precondiciones**: El usuario debe estar autenticado en la aplicación, tener una asesoría contratada y tener acceso a internet. Además, el asesor debe haber realizado una revisión de los datos del asesorado y haber enviado la revisión al asesorado.
- **Flujo principal**:
    1. La aplicación notifica al asesorado sobre la nueva revisión.
    2. El asesor accede a la sección de notificaciones y lee que tiene una nueva actualización de sus planes.
    3. El asesorado accede a sus planes y puede comunicarse con el asesor para que le responda sus dudas.
- **Postcondiciones**: El usuario ha sido notificado de los cambios en sus planes y tiene acceso a ellos.
- **Excepciones**: N/A.

CU11. Valorar asesores.

- **Requisitos:** RQ20.
- **Descripción**: El asesorado valora a un asesor que le haya prestado servicios.
- **Actores**: Asesorado.
- **Precondiciones**: El usuario debe estar autenticado en la aplicación, haber recibido servicios de un asesor durante al menos un mes y tener acceso a internet.
- **Flujo principal**:
    1. El usuario accede a la pantalla de “Buscar”.
    2. El usuario selecciona al asesor que desea valorar.
    3. El usuario completa la valoración.
    4. El usuario envía la valoración.

5. La aplicación registra la valoración y la hace visible para otros usuarios.

- **Postcondiciones**: La valoración del asesor se ha registrado y está disponible para todos los usuarios.
- **Excepciones**:
    1. Si el usuario intenta valorar a un asesor que no le ha prestado servicios, el sistema no se lo permitirá y mostrará un mensaje informándole de los requisitos para valorar a un asesor.

CU12. Acceder a un *dashboard* con el progreso.

- **Requisitos:** RQ21.
- **Descripción**: El asesorado accede a un *dashboard* donde puede ver gráficos y estadísticas de su progreso en el entrenamiento y su nutrición.
- **Actores**: Asesorado.
- **Precondiciones**: El usuario debe estar autenticado en la aplicación, tener una asesoría contratada y tener acceso a internet.
- **Flujo principal**:
    1. El usuario navega a la sección de "Progreso".
    2. La aplicación muestra una pantalla con gráficos y estadísticas del progreso del usuario en diferentes áreas.
    3. El usuario revisa los gráficos y estadísticas para evaluar su progreso.
- **Postcondiciones**: El usuario tiene una visión clara y detallada de su progreso en el entrenamiento y la nutrición.
- **Excepciones**:
    1. Si el usuario no cuenta con los servicios de ningún asesor, se mostrará un mensaje informándole de la situación

CU13. Acceder al calendario.

- **Requisitos:** RQ22, RQ35.
- **Descripción**: El usuario deben tener acceso a un calendario que les permita visualizar los eventos pasados y programados.
- **Actores**: Asesor, Asesorado.
- **Precondiciones**: El usuario debe estar autenticado en la aplicación y tener acceso a internet.
- **Flujo principal**:
    1. El usuario accede a la sección de calendario.
    2. La aplicación muestra un calendario con los eventos pasados y programados.
    3. El usuario puede navegar por las fechas para revisar eventos.
- **Postcondiciones**: El asesor ha accedido a los eventos pasados o programados.
- **Excepciones**: N/A.

CU14. Abandonar asesoría.

- **Requisitos:** RQ23.
- **Descripción**: El usuario cancela los servicios de asesoría de los que disfruta actualmente.
- **Actores**: Asesorado.
- **Precondiciones**: El usuario debe estar en una asesoría, estar autenticado en la aplicación y tener acceso a internet.
- **Flujo principal**:
    1. El usuario navega a la pantalla de "Perfil".

2. El usuario selecciona la opción de "Abandonar asesoría".
3. La aplicación solicita confirmación del usuario para abandonar la asesoría.
4. El usuario confirma la acción.
5. El sistema marca la asesoría como abandonada.

- **Postcondiciones**: La asesoría no se renovará y el asesorado dejará de tener a su disposición los servicios del asesor.
- **Excepciones**: N/A.

CU15. Actualizar perfil profesional.

- **Requisitos:** RQ25.
- **Descripción**: El asesor actualiza su perfil profesional con la información relevante sobre sus servicios, para que esta sea visible para sus potenciales clientes.
- **Actores**: Asesor.
- **Precondiciones**: El asesor debe estar autenticado en la aplicación y tener acceso a internet.
- **Flujo principal**:
    1. El asesor accede a la pantalla de “Perfil”.
    2. El asesor selecciona a la opción de “Perfil profesional”.
    3. La aplicación muestra la información actual del perfil profesional.
    4. El asesor edita la información deseada.
    5. El asesor guarda los cambios.
- **Postcondiciones**: La información del perfil profesional del asesor se ha actualizado y está disponible para el resto de los usuarios.
- **Excepciones**: N/A.

CU16. Crear cuestionario inicial.

- **Requisitos:** RQ26.
- **Descripción**: El asesor crea un cuestionario inicial para que sea completado por sus nuevos clientes.
- **Actores**: Asesor.
- **Precondiciones**: El asesor debe estar autenticado en la aplicación y tener acceso a internet.
- **Flujo principal**:
    1. El asesor accede a la pantalla de “Perfil”.
    2. El asesor selecciona a la opción de “Perfil profesional”.
    3. El asesor selecciona la opción de “Cuestionario inicial”
    4. El asesor crea las preguntas que debe tener el cuestionario para que sea rellenado por los futuros nuevos clientes y así poder personalizar sus planes.
- **Postcondiciones**: El asesor ha creado el cuestionario inicial para que lo pueda ser completado por los potenciales nuevos asesorados.
- **Excepciones**: N/A.

CU17. Creación planes iniciales.

- **Requisitos:** RQ27, RQ30, RQ32, RQ36.
- **Descripción**: El asesor recibe un cuestionario inicial completado por un nuevo cliente, revisa la información proporcionada y crea los planes iniciales de entrenamiento y nutricionales personalizados basados en esa información.
- **Actores**: Asesor.

- **Precondiciones**: El asesor debe estar autenticado en la aplicación, haber recibido un nuevo cuestionario inicial y tener acceso a internet.
- **Flujo principal**:
    1. La aplicación informa en la pantalla de inicio al asesor sobre la recepción de un nuevo cuestionario. El asesor tiene disponible el cuestionario para revisar la información proporcionada por el nuevo asesorado en el cuestionario.
    2. El asesor accede a la pantalla de "Clientes".
    3. El asesor selecciona al cliente específico.
    4. El asesor selecciona la opción de "Plan de entrenamiento".
    5. La aplicación permite al asesor introducir los detalles del plan de entrenamiento (número de sesiones, ejercicios, número de series, repeticiones y descansos).
    6. El asesor completa y guarda el plan de entrenamiento.
    7. El asesor vuelve a la pantalla del cliente específico.
    8. El asesor selecciona la opción de "Crear plan de nutrición".
    9. La aplicación permite al asesor introducir los detalles del plan nutricional (número comidas, cantidades y alimentos).
    10. El asesor completa y guarda el plan nutricional.
- **Postcondiciones**: Los planes de entrenamiento y nutricionales se han creado y están disponibles para el cliente.
- **Excepciones**: N/A.

CU18. Gestionar revisiones.

- **Requisitos:** RQ29, RQ31, RQ33, RQ34, RQ36, RQ37.
- **Descripción**: El asesor recibe los datos periódicos de entrenamiento y de salud de sus clientes y realiza la revisión modificando los planes de entrenamiento y nutricionales existentes según sea necesario.
- **Actores**: Asesor.
- **Precondiciones**: El asesor debe estar autenticado en la aplicación, haber recibido los datos periódicos de salud de su cliente y tener acceso a internet.
- **Flujo principal**:
    1. El asesor accede a la pantalla de "Clientes".
    2. El asesor selecciona a un cliente específico.
    3. El asesor revisa los datos proporcionados por el cliente.
    4. El asesor vuelve a la pantalla de "Clientes".
    5. El asesor selecciona la opción de "Plan de entrenamiento".
    6. La aplicación muestra el plan de entrenamiento actual.
    7. El asesor realiza las modificaciones necesarias (número de sesiones, ejercicios, número de series, repeticiones y descansos).
    8. El asesor guarda los cambios.
    9. El asesor vuelve a la pantalla de "Clientes".
    10. El asesor selecciona la opción de "Plan nutricional".
    11. La aplicación muestra el plan de nutricional actual.
    12. El asesor realiza las modificaciones necesarias (número comidas, cantidades y alimentos).
    13. El asesor guarda los cambios.
    14. El asesor envía las modificaciones y la aplicación notifica al cliente sobre los cambios en sus planes actualizados.

- **Postcondiciones**: Se ha realizado una revisión periódica y se han enviado los nuevos planes de entrenamiento y nutricionales personalizados y se encuentran disponibles para el cliente.
- **Excepciones**: N/A.

5.5 Matriz de requisitos

En este apartado, se presenta la matriz de requisitos de la aplicación de MyTrainer. La Figura 19 nos permite observar de forma sencilla la relación entre los requisitos identificados y los casos de uso. Esta matriz asegura que todos los requisitos se abordan adecuadamente a través de los diferentes casos de uso, facilitando la verificación del sistema y garantizando que todos los aspectos necesarios para el correcto funcionamiento de la aplicación han sido considerados y desarrollados. Además, esta matriz sirve como una guía para el seguimiento del progreso del proyecto.

	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10	CU11	CU12	CU13	CU14	CU15	CU16	CU17	CU18
RQ1		X																
RQ2				X														
RQ3			X															
RQ4					X													
RQ5	X																	
RQ6						X												
RQ7						X												
RQ8						X												
RQ9						X												
RQ10							X											
RQ11							X											
RQ12							X											
RQ13							X											
RQ14								X										
RQ15								X										
RQ16								X										
RQ17								X										
RQ18									X									
RQ19										X								
RQ20											X							
RQ21												X						
RQ22													X					
RQ23														X				
RQ24	X																	
RQ25															X			
RQ26																X		
RQ27																	X	
RQ28									X									
RQ29																		X
RQ30																	X	
RQ31																		X
RQ32																	X	
RQ33																		X
RQ34																		X
RQ35													X					
RQ36																	X	X
RQ37																		X

Figura 19: Matriz de requisitos. Fuente: Elaboración propia.

6 Diseño interfaz de usuario

En este capítulo se describen los detalles del proceso de diseño de la interfaz de usuario para la aplicación MyTrainer. Se ha desarrollado un prototipo de baja fidelidad sobre el que se han llevado a cabo pruebas. Además, se ha elaborado un diagrama de navegabilidad con el objetivo de ofrecer una visión integral de la experiencia de usuario. Estos elementos en conjunto proporcionan una explicación de cómo los usuarios interactuarán con la aplicación, facilitando la implementación técnica.

6.1 Prototipo baja fidelidad

En este apartado se presentan las pantallas del prototipo de baja fidelidad de la aplicación móvil propuesta. Estas pantallas han sido elaboradas con Figma y serán posteriormente utilizadas en las pruebas, estas muestran una versión visual simplificada del flujo de trabajo del sistema, ofreciendo una idea general del aspecto y la funcionalidad de la aplicación. Cada figura se enfoca en pantallas correspondientes a una funcionalidad específica de la aplicación.

En la Figura 20, se muestran tres pantallas del prototipo relacionadas con el acceso a la aplicación. La primera pantalla es la de bienvenida, con el logotipo MyTrainer en el centro y un botón para avanzar a la siguiente pantalla. La segunda pantalla es para iniciar sesión ingresando el email y contraseña del usuario y permite navegar a la tercera pantalla, donde los nuevos usuarios pueden crear una cuenta en la aplicación, eligiendo el rol que tendrán en el sistema.

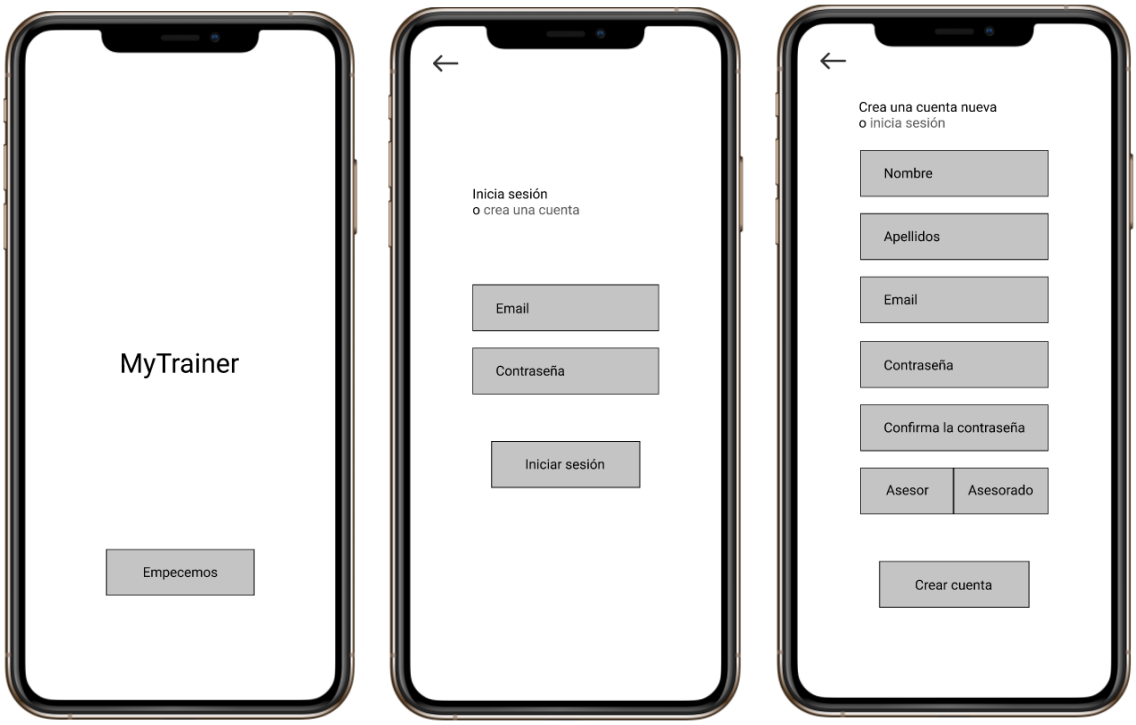


Figura 20: Pantallas prototipo de baja fidelidad de acceso a MyTrainer. Fuente: Elaboración propia.

6.1.1 Pantallas para usuarios con rol asesorado

En este subapartado se exhiben una serie de pantallas del prototipo de baja fidelidad para usuarios con rol de asesorado en la aplicación de MyTrainer. En la Figura 21 la primera pantalla es la de inicio, donde se registran las métricas diarias del asesorado y permite acceder a los planes del día. La segunda detalla el plan de entrenamiento con los ejercicios, series y repeticiones, además, el botón "Empezar sesión" permite acceder a la tercera pantalla donde se podrá seguir el entreno. Por último, la cuarta pantalla muestra el plan nutricional, desglosando las comidas y sus calorías.

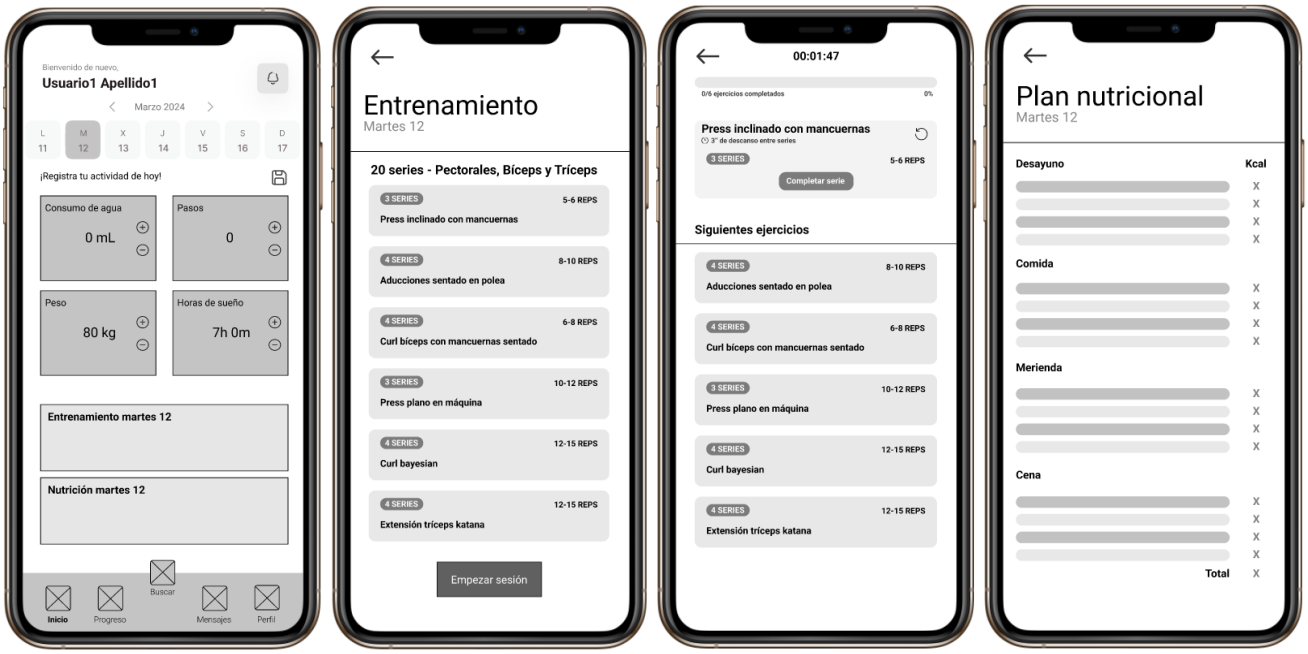


Figura 21: Pantallas prototipo de baja fidelidad de inicio. Fuente: Elaboración propia.

La Figura 22 representa la pantalla de progreso que incluye gráficos para que el asesorado pueda visualizar sus avances en el peso, pasos, consumo de agua y horas de sueño.

Figura 22: Pantallas prototipo de baja fidelidad de progreso. Fuente: Elaboración propia.

La Figura 23 muestra tres pantallas, la primera de ellas presenta los resultados de búsqueda de asesores con opciones de filtrado, que ayudan a los usuarios a encontrar y seleccionar al asesor adecuado. En la segunda pantalla podemos ver el perfil profesional del asesor con detalles de los servicios que ofrece. La tercera pantalla es el cuestionario inicial que los usuarios deben completar al contratar los servicios de un asesor, para que este pueda adaptar los planes de entrenamiento y nutrición a sus necesidades.

Figura 23: Pantallas prototipo de baja fidelidad de buscar. Fuente: Elaboración propia.

Las pantallas que podemos observar en la Figura 24 corresponden al servicio de mensajería del sistema. La primera presenta la lista de chats activos del asesorado con un campo de búsqueda, mientras que la segunda pantalla muestra una conversación específica con un entrenador.

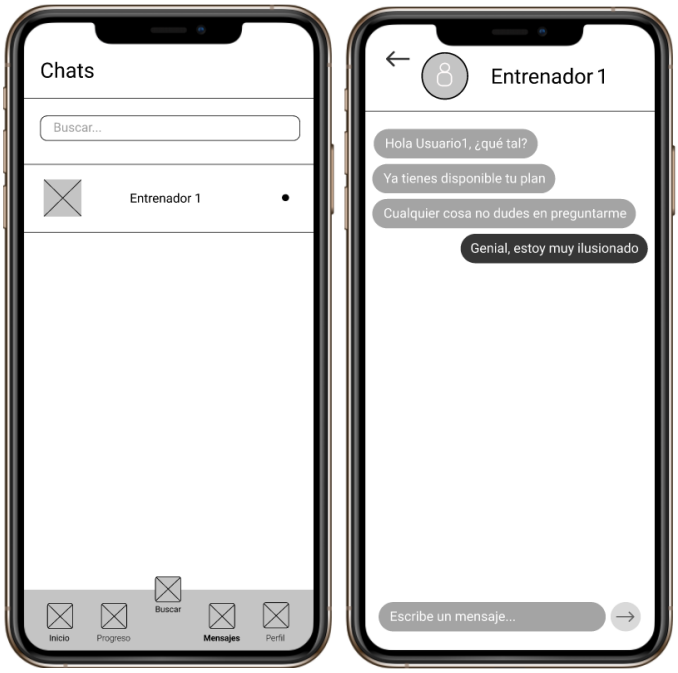


Figura 24: Pantallas prototipo de baja fidelidad de mensajes. Fuente: Elaboración propia.

La Figura 25 presenta la pantalla de perfil del asesorado, donde se visualiza la información personal del usuario junto con opciones para editar el perfil, cancelar la asesoría, cerrar sesión y eliminar la cuenta.

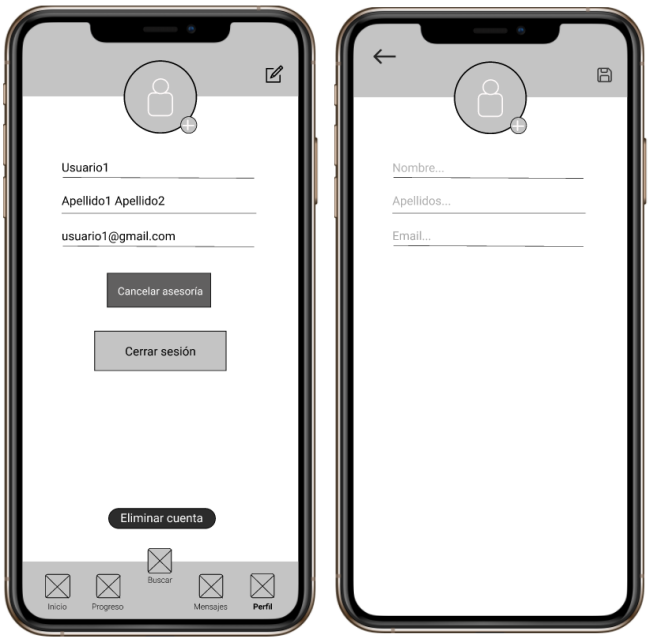


Figura 25: Pantallas prototipo de baja fidelidad de perfil. Fuente: Elaboración propia.

6.1.2 Pantallas para usuarios con rol asesor

En este subapartado se exhiben una serie de pantallas del prototipo de baja fidelidad para usuarios con rol de asesor en la aplicación de MyTrainer. En la Figura 26 podemos observar la pantalla de inicio para los entrenadores de MyTrainer, la cual muestra información relevante sobre el trabajo del asesor.

Figura 26: Pantallas prototipo de baja fidelidad de inicio. Fuente: Elaboración propia.

La primera pantalla de la Figura 27 presenta una lista con los clientes de "Entrenador 1". La segunda muestra el perfil de un cliente específico con accesos directos para ver el cuestionario inicial, el plan de entrenamiento, el plan nutricional, el progreso y los objetivos, además de una opción para enviar la nueva revisión de los planes. La tercera corresponde a las respuestas dadas por el cliente a las preguntas del cuestionario inicial y la última al progreso del cliente.

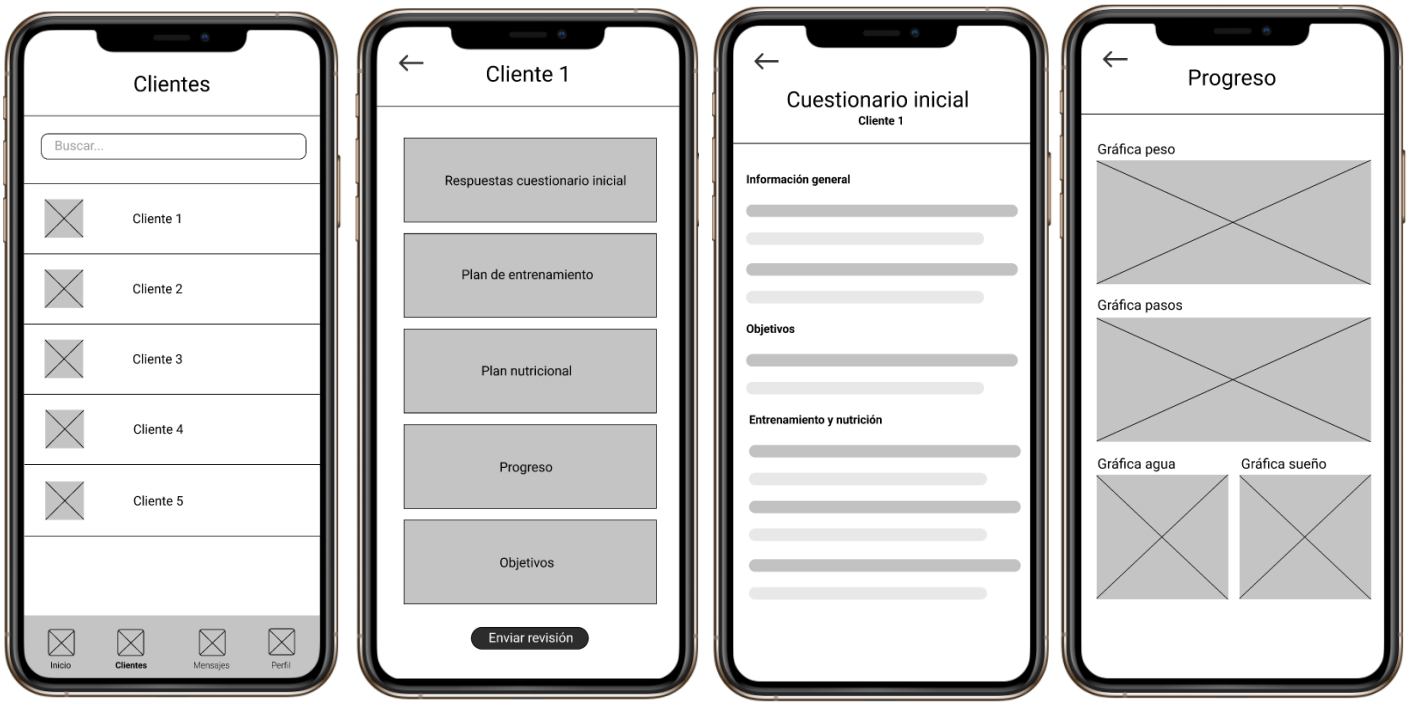


Figura 27: Pantallas prototipo de baja fidelidad de clientes. Fuente: Elaboración propia.

La primera pantalla de la Figura 28 permite seleccionar un día de la semana para poder editar el plan del cliente. La segunda y tercera pantalla presenta las pantallas para la edición de los planes de entrenamiento y de nutrición, respectivamente.

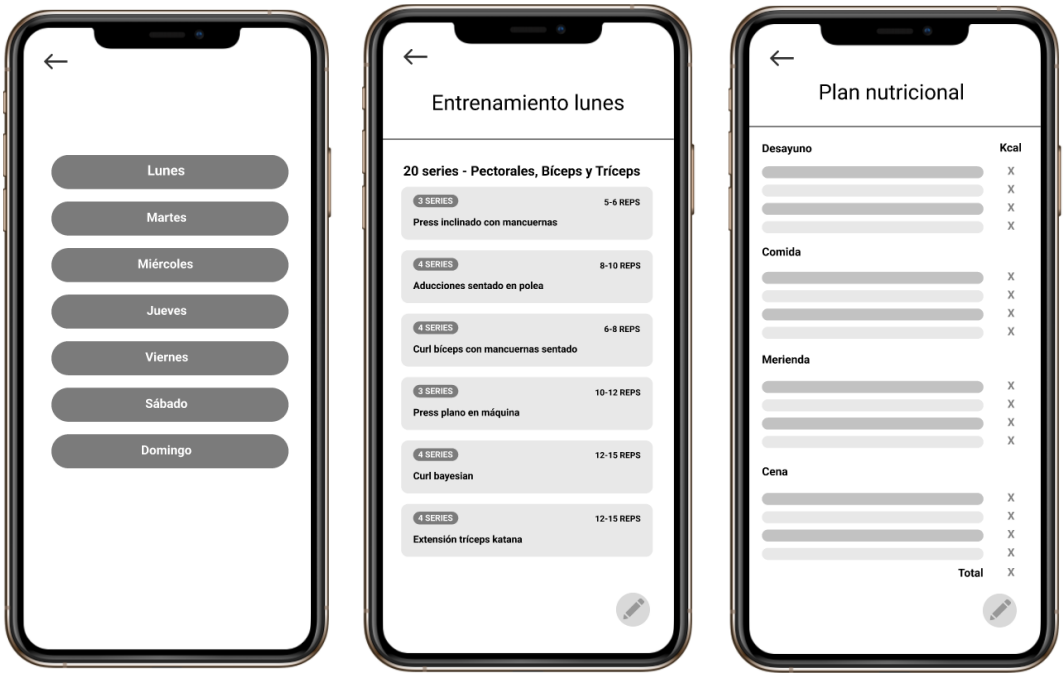


Figura 28: Pantallas prototipo de baja fidelidad de planes de clientes. Fuente: Elaboración propia.

Las pantallas que podemos observar en la Figura 29 corresponden al servicio de mensajería del sistema. La primera presenta la lista de chats activos del entrenador con un campo de búsqueda, mientras que la segunda pantalla muestra una conversación específica con un cliente.

Figura 29: Pantallas prototipo de baja fidelidad de mensajes. Fuente: Elaboración propia.

En la Figura 30, la primera imagen corresponde a la pantalla del perfil del asesor con su información personal y opciones para acceder al perfil profesional, cerrar sesión y eliminar la cuenta. La segunda pantalla presenta el perfil detallado del asesor que es el que verán los asesorados, además incluye un botón que permite navegar a la tercera pantalla, la cual permite al asesor crear el cuestionario inicial que le permitirá personalizar los planes de entrenamiento y nutrición de sus clientes.

Figura 30: Pantallas prototipo de baja fidelidad de perfil. Fuente: Elaboración propia.

6.2 Pruebas usabilidad prototipo baja fidelidad

Para evaluar el prototipo, se han realizado pruebas con diez usuarios, de los cuales seis actuaron como asesorados y los cuatro restantes como asesores. El objetivo de estas pruebas fue identificar áreas de mejora en la usabilidad del sistema y asegurarse de que las funciones clave fueran intuitivas y eficientes para los usuarios.

Para evaluar los resultados se midió el tiempo en segundos que necesitaron los usuarios para completar las tareas. Las pruebas sobre el prototipo se realizaron de forma individual y presencial. Para estas pruebas, se utilizó el emulador de Figma, que permitió replicar de manera precisa la interacción con el prototipo.

Para una mejor organización, las tareas se agruparon según el rol del usuario. Estas se realizaron de forma secuencial dentro de cada grupo, permitiendo a los usuarios realizar breves descansos para evitar la saturación y no interrumpir la sesión. Las pruebas se organizaron en los siguientes grupos de tareas.

Para asesorados:

- Gestión de la cuenta:
    - Registro de usuario.
    - Iniciar sesión.
    - Modificar la información del perfil.
    - Cerrar sesión.
    - Eliminar cuenta.
- Gestión de asesores:
    - Búsqueda y contratación de asesores deportivos.
    - Enviar y recibir mensajes.
    - Valorar asesores.
- Gestión de la asesoría:
    - Acceder al calendario.
    - Abandonar asesoría.
    - Recibir revisión periódica.
    - Acceso a los planes personalizados.
    - Registro diario de métricas de salud.
    - Acceder a un dashboard con el progreso.

Para asesores:

- Gestión de la cuenta:
    - Registro de usuario.
    - Iniciar sesión.
    - Actualizar perfil profesional.
    - Cerrar sesión.
    - Eliminar cuenta.
- Gestión de asesores:
    - Acceder a crear cuestionario inicial
    - Acceso a gestión de clientes
    - Acceder a crear planes iniciales
    - Enviar y recibir mensajes

Las tablas de las Figuras 31 y 32 muestran los resultados de las pruebas de usabilidad realizadas por los usuarios, indicando los tiempos que cada usuario tardó en completar cada tarea.

Grupo de tareas	Tareas	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5	Usuario 6
Gestión de la cuenta	Registro de usuario	15	22	25	20	18	37
	Iniciar sesión	6	7	8	7	5	4
	Modificar información del perfil	17	14	13	15	12	21
	Cerrar sesión	7	9	8	7	6	7
	Eliminar cuenta.	5	6	7	6	5	8
Gestión de asesores	Búsqueda y contratación de asesores deportivos	52	60	35	45	50	95
	Enviar y recibir mensajes	9	11	9	10	8	9
	Valorar asesores	11	25	30	14	18	16
Gestión de la asesoría	Acceder al calendario	21	17	16	19	14	12
	Abandonar asesoría	23	30	43	25	28	10
	Recibir revisión periódica	5	9	5	6	7	8
	Acceso a los planes personalizados	72	35	29	33	39	48
	Registro diario de métricas de salud	29	22	18	25	20	30
	Acceder a un *dashboard* con el progreso	4	8	7	5	6	6

Figura 31: Pruebas de usabilidad de asesorados. Fuente: Elaboración propia.

Grupo de tareas	Tareas	Usuario 1	Usuario 2	Usuario 3	Usuario 4
Gestión de la cuenta	Registro de usuario	20	22	25	30
	Iniciar sesión	3	4	4	5
	Actualizar perfil profesional	18	20	19	17
	Cerrar sesión	8	9	8	7
	Eliminar cuenta	6	9	7	10
Gestión de clientes	Acceder a crear cuestionario inicial	29	28	30	40
	Acceso a gestión de clientes	17	18	20	22
	Acceder a crear planes iniciales	8	12	10	35
	Enviar y recibir mensajes	14	13	15	12

Figura 32: Pruebas de usabilidad de asesor. Fuente: Elaboración propia.

En rojo se marcan las tareas en las que se encontró algún problema. El Usuario 1 tardó 72 segundos en la tarea de "Acceso a los planes personalizados". Aunque encontró rápidamente cómo acceder a los planes, tardó más de lo necesario en registrar el plan de entrenamiento. En cuanto al Usuario 3, completó esta misma tarea en el menor tiempo, 29 segundos, pero esto se debió a que solo registró una serie de ejercicios en lugar de las tres requeridas. Además, en la tarea de "Registro diario de métricas de salud", completó correctamente la entrada de datos, pero olvidó registrarlos. El Usuario 6 tardó 95 segundos en "Búsqueda y contratación de asesores deportivos" porque intentó acceder a los asesores a través de los mensajes, algo que ningún otro usuario hizo, lo que le llevó a moverse por varias pantallas antes de completar correctamente la tarea. En cuanto a los asesores, el Usuario 4 tardó 35 segundos en "Acceder a crear planes iniciales", le llevo un tiempo excesivo acceder a la creación de los planes iniciales, ya que el usuario pensaba que podía hacerlo desde una pantalla diferente a la de clientes.

En general, los resultados han sido positivos, siendo la navegación de los usuarios por la aplicación fluida. De los problemas encontrados, se concluyó que los registros de las métricas de salud diarias y las actualizaciones del perfil deben guardarse automáticamente sin la necesidad de presionar un botón de guardado. En cuanto al resto de las funcionalidades y la navegabilidad del sistema, se mantendrán tal como están para la aplicación final, ya que no se encontraron inconvenientes significativos que requieran modificaciones.

6.3 Diagrama navegabilidad

En esta sección se presentan los diagramas de flujo de pantallas de la aplicación MyTrainer, tanto para los usuarios con rol de asesorado, Figura 33, como para los asesores, Figura 34. Estos diagramas ilustran cómo los usuarios podrán navegan a través de las diferentes pantallas de la aplicación, facilitando la comprensión del diseño y estructura del sistema.

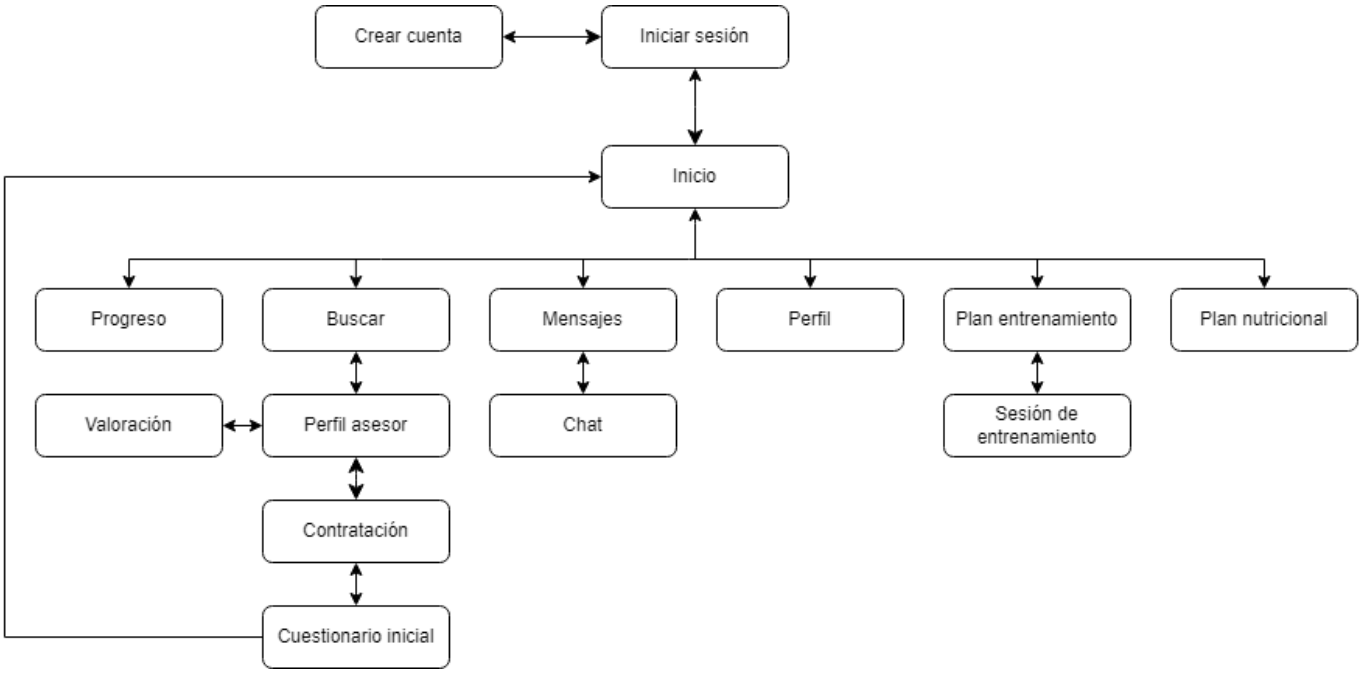


Figura 33: Diagrama de flujo de pantallas de la aplicación para los asesorados. Fuente: Elaboración propia.

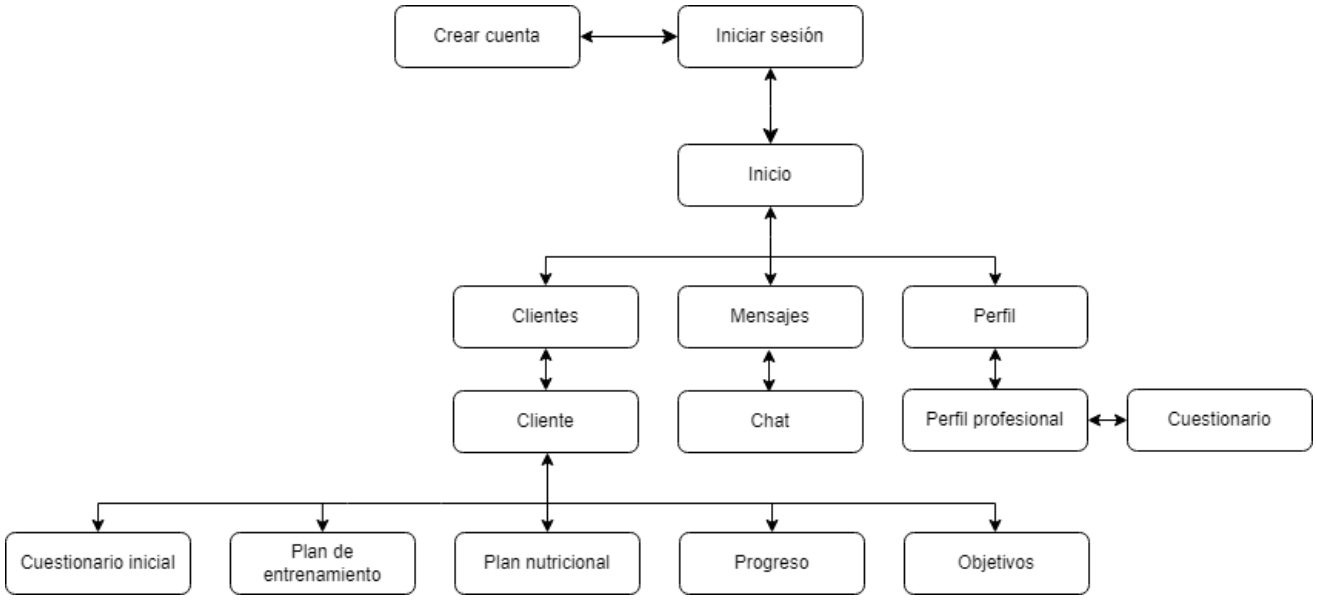


Figura 34: Diagrama de flujo de pantallas de la aplicación para los asesores. Fuente: Elaboración propia.

7 Desarrollo

En este capítulo se detallarán los elementos clave para el desarrollo de la aplicación MyTrainer. Se abordan aspectos desde la preparación del entorno de desarrollo hasta la estructura del código, pasando las librerías y extensiones utilizadas.

7.1 Preparación y configuración entorno

Los primeros pasos en el desarrollo incluyen todas las tareas relacionadas con la configuración y preparación del entorno donde se desarrollará la aplicación. La preparación y configuración del entorno de desarrollo es esencial para garantizar un proceso de desarrollo fluido y eficiente. A continuación, se detallan las herramientas y tecnologías utilizadas en el proyecto MyTrainer, así como los pasos necesarios para su configuración.

La primera tarea que se llevó a cabo fue la instalación del SDK de Flutter, que es un kit de desarrollo de software que proporciona las herramientas necesarias para la creación de aplicación en Flutter.

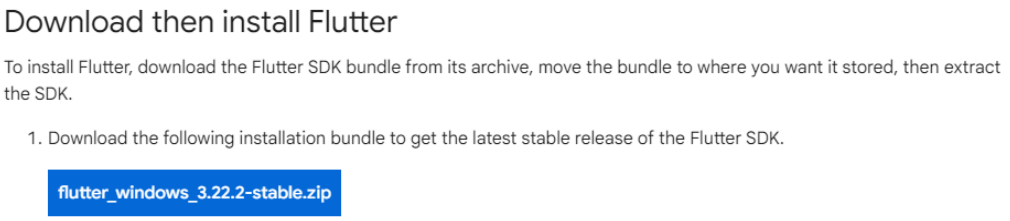


Figura 35: Descarga SDK Flutter. Fuente: Elaboración propia.

Una vez descargado el archivo de la Figura 35 desde la página oficial de Flutter, procedí a extraerlo en la carpeta de destino. Seguidamente, agregué la ruta del binario de Flutter a la variable de entorno PATH del sistema para poder ejecutar los comandos de Flutter desde cualquier ubicación en la terminal.

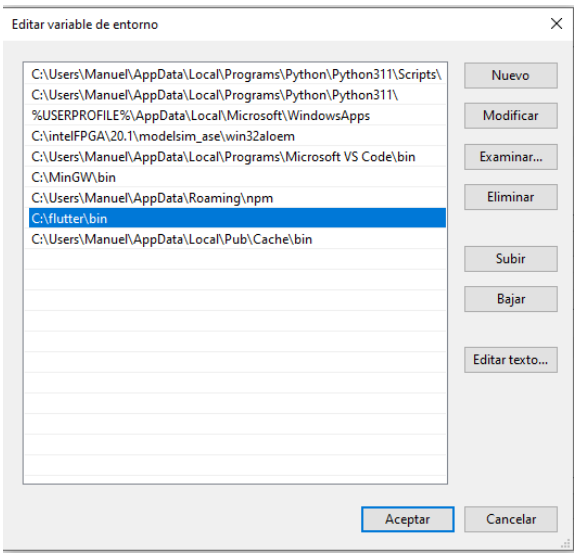


Figura 36: Actualización de la Variable PATH en Windows. Fuente: Elaboración propia.

Posteriormente se realizó la instalación de Android Studio y Visual Studio Code. Aunque Android Studio proporciona un entorno completo de desarrollo con herramientas avanzadas para la depuración, pruebas y construcción de aplicaciones Android, decidí emplear Visual Studio Code como editor de código principal debido a mi familiaridad con él y su capacidad de personalización, utilizando Android Studio para la ejecución del emulador y pruebas en dispositivos Android.

Durante el desarrollo de la aplicación, se utilizó únicamente el emulador de Android que se muestra en la captura de la Figura 37, debido a la falta de acceso a un equipo Mac, el cual es necesario para emular aplicaciones en iOS. Sin embargo, al finalizar el desarrollo del proyecto, se consiguió un equipo Mac para realizar las pruebas y ejecutar la aplicación en un dispositivo iOS. Para ello, se instaló Xcode [19] en el Mac, lo cual permitió la creación de la aplicación en iOS tras el previo ajuste de los archivos de configuración en la carpeta *ios* del proyecto Flutter.

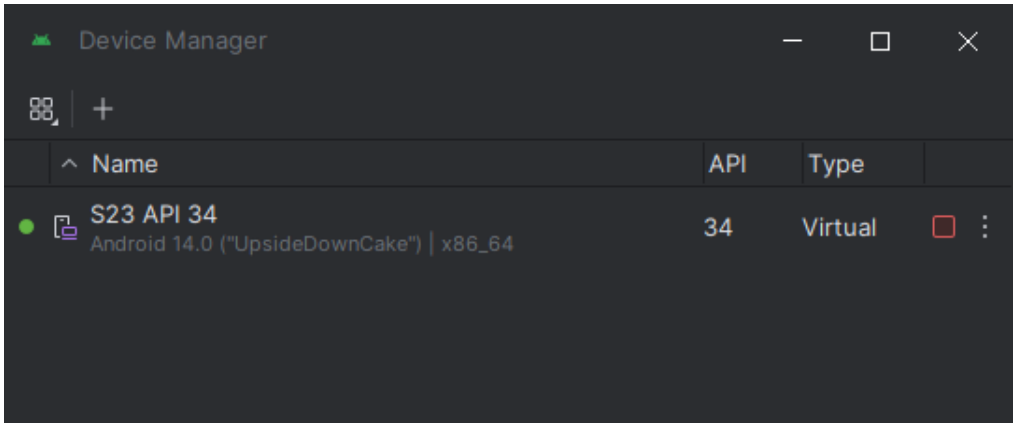


Figura 37: Captura emulador Android Studio. Fuente: Elaboración propia.

Por otra parte, para la configuración de Firebase fue necesario crear un proyecto el cual se puede observar en la Figura 38. Además, se tuvieron que descargar los archivos de configuración y agregarlos a las carpetas correspondientes en el proyecto Flutter. Por último, para la integración correcta de Firebase con Flutter, se utilizó FlutterFire CLI, que simplifica la configuración de Firebase en proyectos Flutter gracias a un conjunto de *plugins* que permite utilizar los servicios de Firebase.

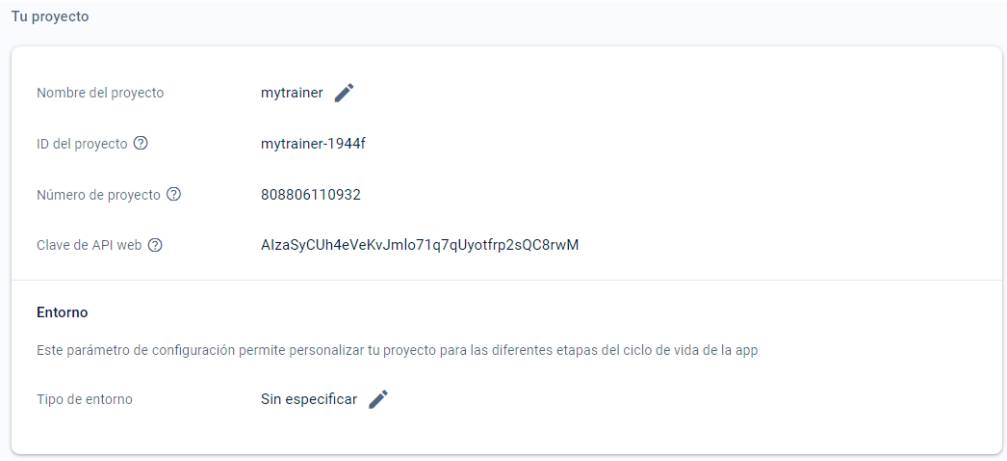


Figura 38: Proyecto de MyTrainer en Firebase Console. Fuente: Elaboración propia.

7.2 Librerías y extensiones utilizadas

Para el desarrollo del proyecto, fue necesario utilizar una serie de librerías y extensiones que facilitaron el proceso de desarrollo, ayudando a proporcionar funcionalidades específicas y a mejorar la eficiencia del código. A continuación, se describen las librerías y extensiones utilizadas en el proyecto.

Las librerías utilizadas fueron las siguientes:

- ***intl***. Proporciona herramientas para la internacionalización del formato de las fechas.
- ***cupertino_icons***. Ofrece un conjunto de iconos de estilo iOS.
- ***firebase_core.*** Librería principal de Firebase, imprescindible para usar cualquier otro servicio de Firebase.
- ***firebase_auth***. Librería de Firebase que proporciona los métodos necesarios para la gestión de la autenticación de usuarios.
- ***cloud_firestore.*** Librería de Firebase que proporciona acceso a Firestore, mediante métodos que permiten almacenar y sincronizar los datos.
- ***firebase_storage***. Librería de Firebase utilizada en el proyecto para subir archivos multimedia a Firebase Storage.
- ***image_picker***. Librería que facilita la selección de imágenes desde la galería del dispositivo o la cámara.
- ***multi_select_flutter.*** Librería que ofrece *widgets* para seleccionar múltiples elementos de una lista.
- ***flutter_rating_bar***. Librería que proporciona un *widget* para implementar barras de calificación.
- ***floating_action_bubble***. Librería que ofrece un botón de acción flotante con opciones desplegables.
- ***table_calendar***. Librería que proporciona un *widget* de calendario personalizable.
- ***percent_indicator***. Librería con *widgets* para mostrar indicadores de progreso en forma de barras o círculos.
- ***fl_char.*** Biblioteca utilizada para crear gráficos en la aplicación.
- ***font_awesome_flutter***. Librería que proporciona diversos iconos que serán utilizados en la aplicación.

En la Figura 39 podemos observar una captura del archivo *pubspec.yaml* del proyecto Flutter de MyTrainer, donde se han añadido las librerías, con su versión, utilizadas para el desarrollo del mismo.

```yaml
dependencies:
  flutter:
    sdk: flutter
  intl: ^0.17.0
  cupertino_icons: ^1.0.6
  firebase_core: ^3.1.0
  firebase_auth: ^5.1.0
  cloud_firestore: ^5.0.1
  image_picker: ^1.1.2
  firebase_storage: ^12.0.1
  multi_select_flutter: ^4.0.0
  flutter_rating_bar: ^4.0.0
  floating_action_bubble: ^1.1.4
  table_calendar: ^3.0.0
  percent_indicator: ^4.2.2
  fl_chart: ^0.68.0
  font_awesome_flutter: ^10.1.0
```

Figura 39: Librerías utilizadas en el proyecto. Fuente: Elaboración propia.

En cuanto a las extensiones utilizadas, encontramos las siguientes:

- **Dart**. Extensión necesaria para trabajar con Flutter en Visual Studio Code, ya que este está basado en el lenguaje Dart. Añade soporte para el lenguaje detectando errores y ofreciendo recomendaciones en tiempo real, proporcionando herramientas para la edición.

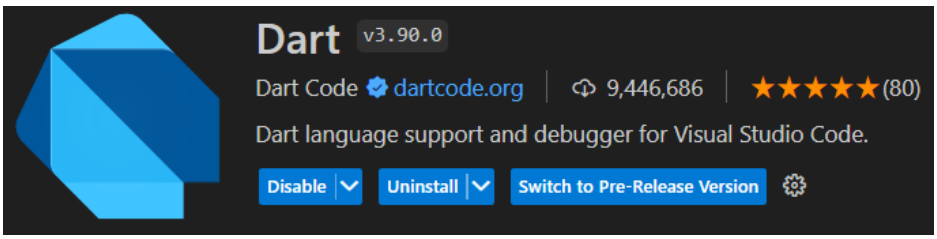


Figura 40: Captura de la extensión Dart en Visual Studio Code. Fuente: Elaboración propia.

- **Flutter.** Al igual que la anterior extensión es necesaria para trabajar con Flutter soporte adicional para Flutter en Visual Studio Code. Permite recargar los cambios durante el desarrollo, "Hot Reload", sin necesidad de reiniciar la aplicación. Además, facilita la depuración y proporciona asistencia para la edición de *widgets*.

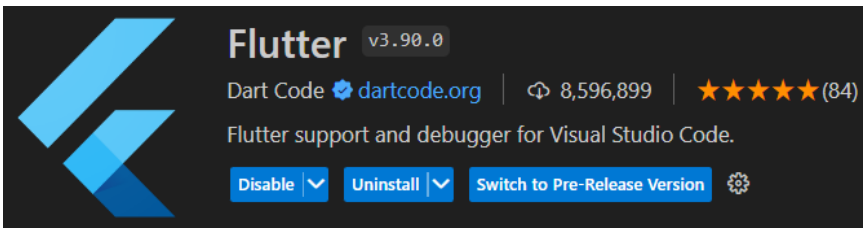


Figura 41: Captura de la extensión Flutter en Visual Studio Code. Fuente: Elaboración propia.

- **Error Lens.** Aunque esta extensión no es imprescindible, facilita el desarrollo notablemente ya que proporciona soporte para la visualización de errores y advertencias en el código. Resalta las líneas donde se producen errores y muestra mensajes de error en línea, facilitando la detección y corrección de problemas antes de la compilación

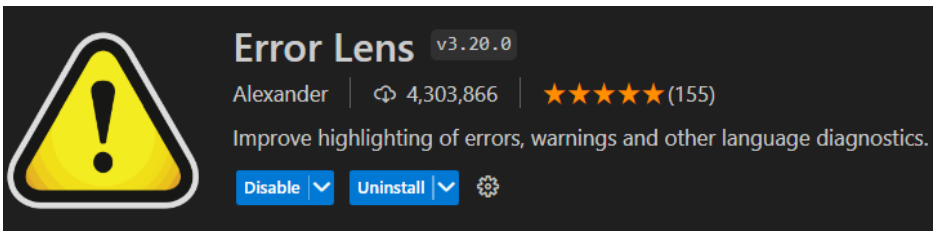


Figura 42: Captura de la extensión Error Lens en Visual Studio Code. Fuente: Elaboración propia.

7.3 Estructura del código

Los proyectos de Flutter se crean con una estructura predefinida que proporciona la configuración básica para que el proyecto se pueda ejecutar y no

se pierda tiempo en realizar configuraciones. Este esqueleto tiene una estructura que ayuda a mantener la organización y el orden del código, separándolo en carpetas. En la Figura 43 se observa la organización del proyecto Flutter de MyTrainer. Cada una de las carpetas y archivos que aparecen en la captura cumplen con una función determinada:

- **Carpeta *.dart_tool*.** Carpeta de configuración que contiene archivos utilizados por las extensiones de Dart y Flutter.
- **Carpeta *.idea*.** Carpeta que contiene las configuraciones necesarias para Android Studio.
- **Carpeta *android*.** Carpeta que contiene los archivos específicos para configurar todo lo necesario para compilar y ejecutar la aplicación en Android.
- **Carpeta *assets*.** Carpeta usada en el proyecto para almacenar imágenes estático, aunque también puede albergar otros tipos de archivos. Para poder hacer uso de estos archivos en la aplicación, se deben declarar en el archivo *pubspec.yaml*.
- **Carpeta *ios*.** Carpeta que contiene los archivos específicos para configurar todo lo necesario para compilar y ejecutar la aplicación en iOS.
- **Carpeta *lib*.** Carpeta, organizada en subcarpetas, que contiene el código fuente de la aplicación.
- **Carpeta *test*.** Carpeta, organizada en subcarpetas, que contiene archivos de pruebas unitarias y de integración.
- **Carpeta *web*.** Carpeta que contiene los archivos específicos para configurar todo lo necesario para compilar y ejecutar la aplicación en su versión web.
- ***.flutter-plugins*.** Archivo generado automáticamente al crear el proyecto Flutter, que contiene una lista de los *plugins* utilizados en el proyecto, indicando cuáles están en uso y dónde se encuentran.
- ***.flutter-plugins-dependencies*.** Archivo generado automáticamente al crear el proyecto Flutter, que mantiene un registro de las dependencias de los *plugins*. Ayuda a gestionar las dependencias y a asegurarse de que se utilizan las versiones correctas.
- ***.gitignore*.** Archivo que especifica qué archivos y directorios deben ser ignorados por Git [20].
- ***.metadata*.** Archivo generado automáticamente por Flutter que contiene metadatos del proyecto. Incluye información sobre la configuración y estado del proyecto.
- ***firebase.json*.** Archivo de configuración de Firebase, que contiene configuraciones específicas relacionadas con los servicios de Firebase que se utilizan en el proyecto.
- ***key.json*.** Este archivo contiene claves y credenciales para los servicios externos de Firebase. Es crucial para la configuración de los servicios de Firebase Authentication, Firestore y Cloud Storage.
- ***mytrainer.iml*.** Archivo de configuración específico del proyecto para Android Studio.
- ***pubspec.lock*.** Archivo generado automáticamente por el gestor de paquetes de Dart, que contiene las referencias a las versiones exactas de las dependencias utilizadas en el proyecto.
- ***pubspec.yaml*.** Archivo principal de configuración del proyecto Flutter, donde se definen las dependencias del proyecto, como se puede ver en la Figura 39. Además, también se declaran los *assets*, el nombre del proyecto y la versión del SDK, entre otros ajustes.

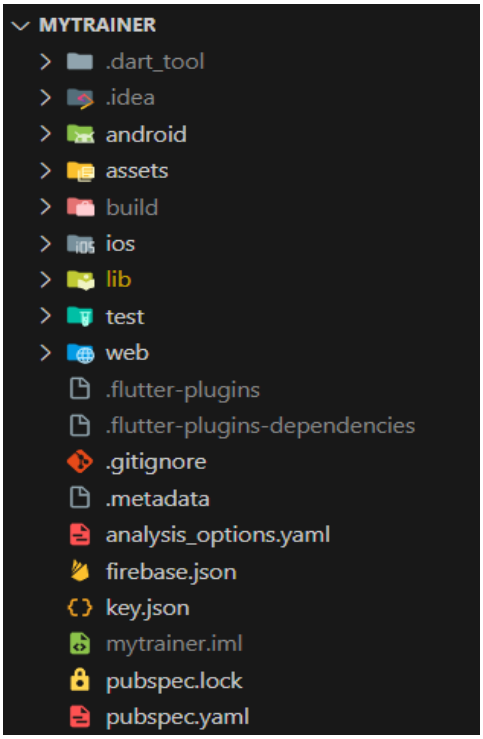


Figura 43: Carpeta del proyecto Flutter de MyTrainer. Fuente: Elaboración propia.

La Figura 44 muestra la estructura de la carpeta *lib*, como se ha mencionado anteriormente, esta carpeta es elemental para cualquier proyecto Flutter ya que contiene el código fuente de la aplicación.

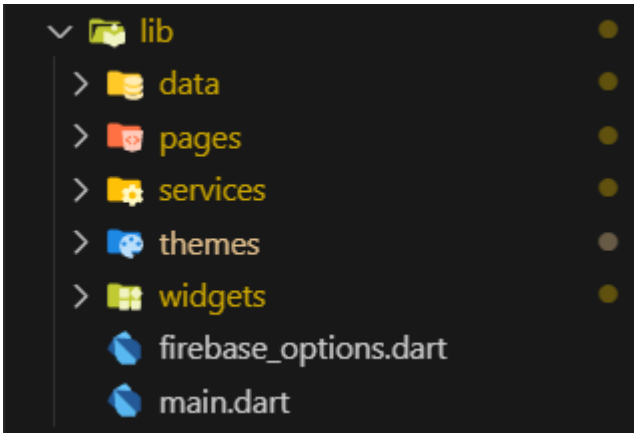


Figura 44: Carpeta lib. Fuente: Elaboración propia.

Esta carpeta es organizada de la siguiente forma:

- ***data/.*** Esta subcarpeta contiene las clases de modelos de datos, incluyendo su definición para que sean almacenados correctamente.
- ***pages/.*** En esta subcarpeta se encuentran el código correspondiente a la interfaz de usuario de las distintas pantallas de la aplicación. Cada archivo representa una pantalla completa de MyTrainer. Dentro de esta subcarpeta encontramos otras dos *asesor/* y *asesorado/* las cuales contienen el código de las pantallas de la aplicación para los usuarios con rol asesor y asesorado, respectivamente. Además de estas carpetas

también se encuentran los ficheros *.dart* correspondientes a las pantallas comunes entre ambos tipos de usuarios.

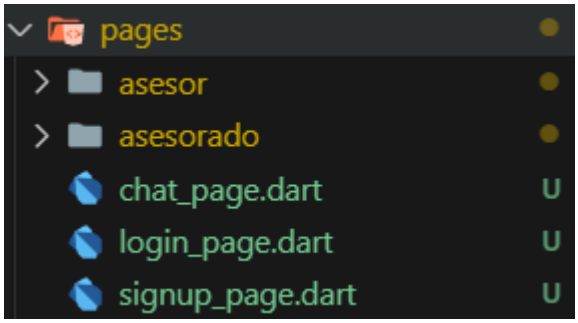


Figura 45: Carpeta pages. Fuente: Elaboración propia.

- ***services/.*** En esta subcarpeta se encuentran el código que maneja las interacciones con los distintos servicios de Firebase. Se encuentra a la vez dividida en otras carpetas según las funcionalidades que proporcionan los métodos de sus archivos.

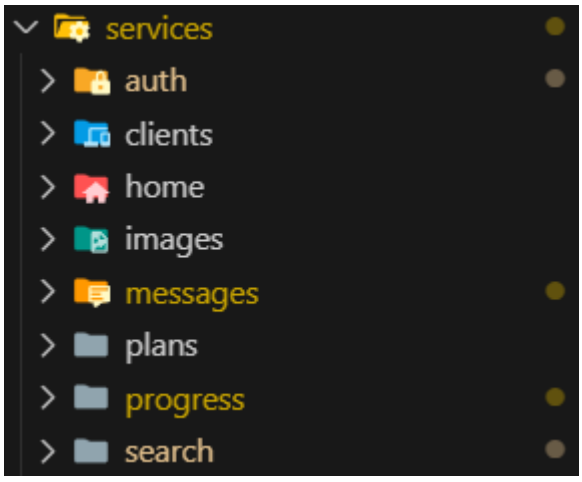


Figura 46: Carpeta services. Fuente: Elaboración propia.

- ***themes/.*** Esta subcarpeta contiene los archivos *.dart* que sirven para la configuración de los temas visuales de la aplicación.
- ***widgets/.*** En esta subcarpeta se encuentran los *widgets* creados de forma personalizada para ser reutilizados en distintas partes de la aplicación, ganando de esta forma organización y reduciendo en gran medida la cantidad de líneas de código.
- ***firebase_options.dart.*** Archivo que contiene la información necesaria para la configuración de la conexión con el proyecto de Firebase que contiene las opciones necesarias para inicializar Firebase en la aplicación.
- ***main.dart***. Este archivo contiene la función *main()* que inicializa la aplicación.

7.4 Fragmentos clave del código

En este apartado se presentan algunos de los fragmentos de código más importantes de la aplicación. Estos fragmentos son fundamentales para correcto funcionamiento de diversas funcionalidades dentro de la aplicación. A continuación, se explican algunos de los detalles más relevantes para su funcionamiento.

7.4.1 main.dart

Como se mencionó en el apartado anterior, el archivo *main.dart* es el punto de entrada de la aplicación Flutter. Este archivo requiere la importación de paquetes esenciales para inicializar Firebase, además del paquete para los *widgets* de Material Design, lenguaje de diseño de Google y el archivo con el tema de aplicación. En la función principal *main()*, se asegura que los *widgets* de Flutter estén inicializados antes de ejecutar el resto del código. Luego, se inicializa Firebase y se ejecuta la aplicación llamando a *runApp*. Por otra parte, la clase *MyApp* es un *widget* sin estado que contiene el método *build*, estableciendo *AuthListener()* como la pantalla principal para gestionar el estado de autenticación del usuario y aplicando el tema definido en *light_mode.dart*.

Figura 47: Código main.dart. Fuente: Elaboración propia.

7.4.2 Auth_listener.dart

El archivo *auth_listener.dart*, el cual se encuentra en *lib/services/auth/*, gestiona la autenticación del usuario con Firebase Authentication y según el rol del usuario autenticado (asesorado o asesor) muestra la barra de navegación adecuada. Para ello se utiliza *StreamBuilder*, que permite escuchar los cambios

en el estado de autenticación de Firebase y *FutureBuilder*, para determinar el rol del usuario. Por otra parte, si el usuario está autenticado, verifica cual es el rol que posee y se muestra la barra de navegación correspondiente, ya que para cada tipo de usuario esta es distinta. Si el usuario no está autenticado, se muestra la pantalla de inicio de sesión. Este fragmento, el cual se puede observar en la Figura 48, es fundamental, pues sin él no se podría asegurar una navegación adecuada y personalizada según el estado y el rol del usuario en la aplicación.

lib > services > auth > auth_listener.dart > ...

```dart
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:mytrainer/services/auth/auth.dart';
import 'package:mytrainer/services/auth/login_or_signup.dart';
import 'package:mytrainer/widgets/my_bar.dart';
import 'package:mytrainer/widgets/asesor_my_bar.dart';

// Utiliza un StreamBuilder para escuchar los cambios en el estado de autenticación de Firebase.
// Si el usuario está autenticado, se verifica si es un asesorado o un asesor.
// Dependiendo del resultado, se muestra la barra de navegación correspondiente.
class AuthListener extends StatelessWidget {
  const AuthListener({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: StreamBuilder<User?>(
        stream: FirebaseAuth.instance.authStateChanges(),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
          } else if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          } else if (snapshot.hasData) {
            return FutureBuilder<bool>(
              future: AuthService().isUserAsesorado(),
              builder: (context, isUserAsesoradoSnapshot) {
                if (isUserAsesoradoSnapshot.connectionState == ConnectionState.waiting) {
                  return const Center(child: CircularProgressIndicator());
                } else if (isUserAsesoradoSnapshot.hasError) {
                  return Center(child: Text('Error: ${isUserAsesoradoSnapshot.error}'));
                } else if (isUserAsesoradoSnapshot.hasData && isUserAsesoradoSnapshot.data == true) {
                  return const MyBarAsesorado();
                } else {
                  return const MyBarAsesor();
                }
              },
            ); // FutureBuilder
          } else {
            return const LoginOrSignup();
          }
        },
      ), // StreamBuilder
    ); // Scaffold
  }
}
```

Figura 48: Código auth_listener.dart. Fuente: Elaboración propia.

7.4.3 asesor_my_bar.dart

El archivo *asesor_my_bar.dart,* el cual se encuentra en *lib/widgets/*, define un *widget* con estado llamado *MyBarAsesor*, que proporciona una barra de navegación inferior para asesores en la aplicación MyTrainer. Esta barra permite a los asesores navegar entre diferentes páginas: Inicio, Clientes, Mensajes y Perfil. Además, contiene iconos para cada pestaña, y utiliza el esquema de colores del tema de la aplicación.

La clase utiliza la lista de *widgets _widgetOptions*, en la que cada elemento corresponde a cada una de las pestañas de la barra de navegación. Cuando uno

de los iconos de la barra de navegación es presionado, el índice *_selectedIndex* se actualiza y se muestra el contenido correspondiente.

lib > widgets > asesor_my_bar.dart > ...

```dart
import 'package:flutter/material.dart';
import 'package:mytrainer/pages/asesor/asesor_clients_page.dart';
import 'package:mytrainer/pages/asesor/asesor_home_page.dart';
import 'package:mytrainer/pages/asesor/asesor_messages_page.dart';
import 'package:mytrainer/pages/asesor/asesor_profile_page.dart';

// Widget con estado para la barra de navegación del asesor.
class MyBarAsesor extends StatefulWidget {
  const MyBarAsesor({super.key});

  @override
  // ignore: library_private_types_in_public_api
  _MyBarAsesorState createState() => _MyBarAsesorState();
}

class _MyBarAsesorState extends State<MyBarAsesor> {
  int _selectedIndex = 0;

  static final List<Widget> _widgetOptions = <Widget>[
    const AsesorHomePage(),
    const AsesorClientsPage(),
    AsesorMessagesPage(),
    const AsesorProfilePage(),
  ]; // <Widget>[]

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: IndexedStack(
        index: _selectedIndex,
        children: _widgetOptions,
      ), // IndexedStack
      bottomNavigationBar: BottomNavigationBar(
        type: BottomNavigationBarType.fixed,
        iconSize: 32,
        selectedLabelStyle: const TextStyle(fontSize: 16),
        unselectedLabelStyle: const TextStyle(fontSize: 14),
        fixedColor: Theme.of(context).colorScheme.primary,
        unselectedItemColor: Theme.of(context).colorScheme.tertiary,
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Inicio',
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.people),
            label: 'Clientes',
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.message),
            label: 'Mensajes',
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Perfil',
          ), // BottomNavigationBarItem
        ], // <BottomNavigationBarItem>[]
        currentIndex: _selectedIndex,
        onTap: _onItemTapped,
      ), // BottomNavigationBar
    ); // Scaffold
  }
}
```

Figura 49: Código asesor_my_bar.dart. Fuente: Elaboración propia.

7.4.4 my_bar.dart

El fragmento de código de la Figura 50, corresponde con el archivo *my_bar.dart,* el cual se encuentra en *lib/widgets/.,* define un *widget* con estado llamado MyBarAsesor define un *widget* con estado. Este *widget* actúa igual que el comentado en el apartado anterior, pero para los usuarios con rol de asesorado.

Este *widget* permite a los asesorados navegar entre las páginas de: Inicio, Progreso, Buscar, Mensajes y Perfil.

lib > widgets > my_bar.dart > ...

```dart
import 'package:flutter/material.dart';
import 'package:mytrainer/pages/asesorado/messages_page.dart';
import 'package:mytrainer/pages/asesorado/profile_page.dart';
import 'package:mytrainer/pages/asesorado/progress_page.dart';
import 'package:mytrainer/pages/asesorado/search_page.dart';
import 'package:mytrainer/pages/asesorado/home_page.dart';

// Widget con estado para la barra de navegación del asesorado.
class MyBarAsesorado extends StatefulWidget {
  const MyBarAsesorado({super.key});

  @override
  // ignore: library_private_types_in_public_api
  _MyBarAsesoradoState createState() => _MyBarAsesoradoState();
}

class _MyBarAsesoradoState extends State<MyBarAsesorado> {
  int _selectedIndex = 0;

  // ignore: prefer_final_fields
  static List<Widget> _widgetOptions = <Widget>[
    const HomePage(),
    ProgressPage(),
    SearchPage(),
    MessagesPage(),
    const ProfilePage(),
  ]; // <Widget>[]

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: IndexedStack(
        index: _selectedIndex,
        children: _widgetOptions,
      ), // IndexedStack
      bottomNavigationBar: BottomNavigationBar(
        type: BottomNavigationBarType.fixed,
        iconSize: 32,
        selectedLabelStyle: const TextStyle(fontSize: 16),
        unselectedLabelStyle: const TextStyle(fontSize: 14),
        fixedColor: Theme.of(context).colorScheme.primary,
        unselectedItemColor: Theme.of(context).colorScheme.tertiary,
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Inicio',
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.show_chart),
            label: 'Progreso',
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.search),
            label: 'Buscar',
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.message),
            label: 'Mensajes',
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Perfil',
          ), // BottomNavigationBarItem
        ], // <BottomNavigationBarItem>[]
        currentIndex: _selectedIndex,
        onTap: _onItemTapped,
      ), // BottomNavigationBar
    ); // Scaffold
  }
}
```

Figura 50: Código my_bar.dart. Fuente: Elaboración propia.

8 Resultados y pruebas

En este capítulo se presentarán los resultados del desarrollo llevado a cabo, junto con las pruebas y el despliegue del sistema.

8.1 Resultados del desarrollo

En este apartado se presentan figuras con diversas capturas de pantalla del producto final de la aplicación junto con una breve explicación de las funcionalidades clave, con la intención de proporcionar una visión clara de la interfaz de usuario y la forma en que los usuarios interactuarán con ella.

En la Figura 51 se muestran dos pantallas clave de la aplicación MyTrainer: la pantalla de inicio de sesión y la pantalla de registro. La pantalla de inicio de sesión permite a los usuarios registrados acceder a sus cuentas ingresando su correo electrónico y contraseña, con un enlace para nuevos usuarios que redirige a la pantalla de registro en el caso de que el usuario no tenga una cuenta en la aplicación. Desde la pantalla de registro se permite a los nuevos usuarios crear una cuenta ingresando su información personal y seleccionando su rol como "Asesor" o "Asesorado". También incluye un enlace para usuarios que ya tienen una cuenta y desean iniciar sesión.

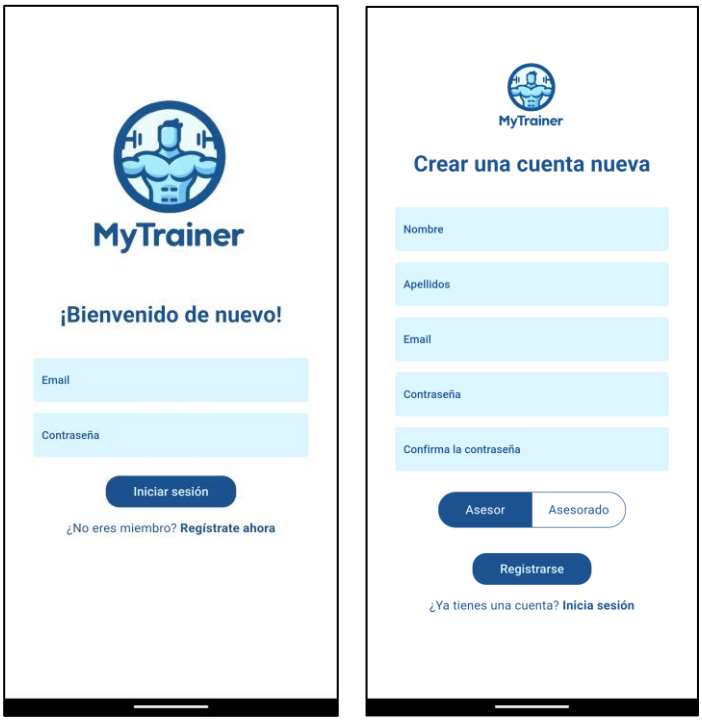


Figura 51: Captura de pantallas de acceso al sistema. Fuente: Elaboración propia.

Ambas pantallas de la Figura 51 son las únicas comunes para todos los usuarios. A continuación, se han divido las pantallas en dos grupos: las pantallas para asesorados y las pantallas para asesores.

8.1.1 Pantallas para usuarios con rol asesorado

La Figura 52 muestra dos pantallas clave de la aplicación MyTrainer. La primera pantalla de inicio da la bienvenida al usuario y notifica si no tiene ninguna asesoría contratada. La pantalla de progreso indica que no hay datos disponibles ya que no hay ninguna métrica de salud registrada.

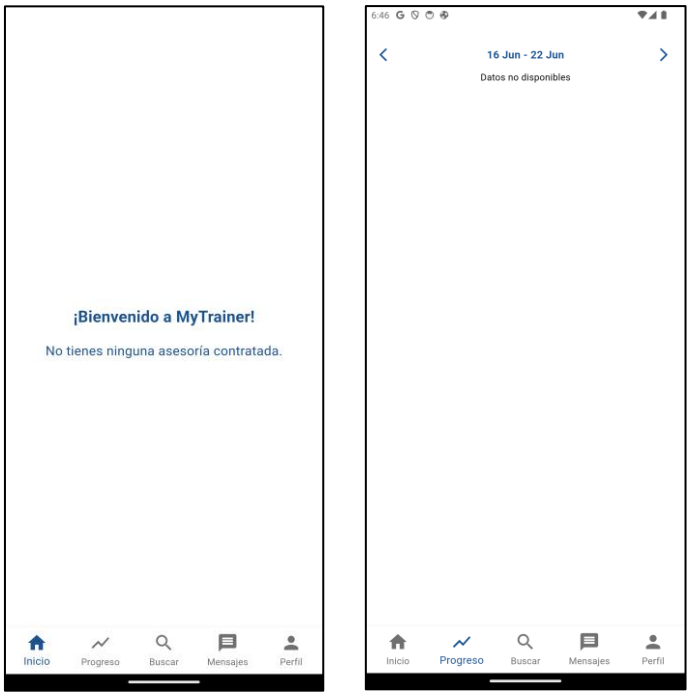


Figura 52: Captura de pantallas de inicio y progreso sin asesoría. Fuente: Elaboración propia.

En la Figura 53 se presenta la pantalla de perfil de los usuarios asesorados. Esta pantalla permite a los usuarios ver y editar su perfil de usuario. Además, presenta opciones de cuenta, como cambiar contraseña y cerrar sesión.

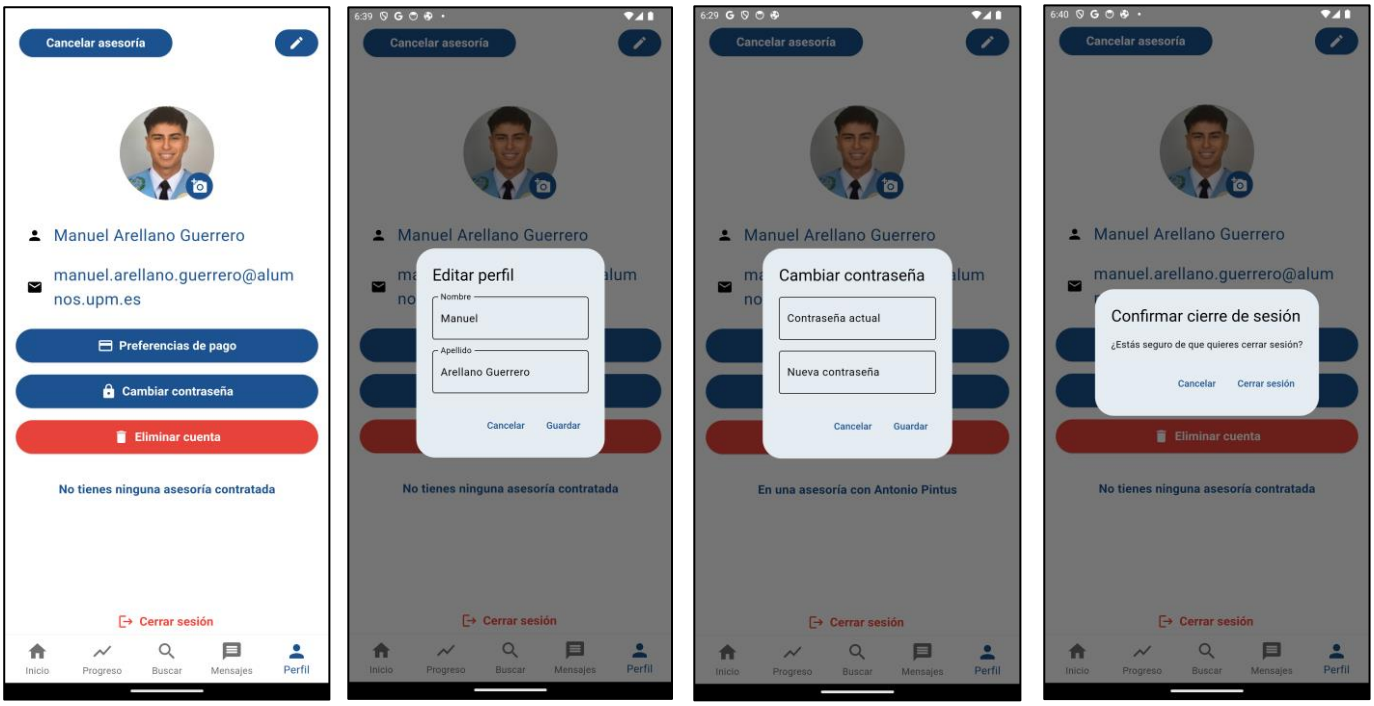


Figura 53: Captura de pantallas de perfil del asesorado. Fuente: Elaboración propia.

La Figura 54 detalla tres pantallas donde los usuarios confirman acciones críticas: la eliminación de cuenta ingresando su contraseña, la cancelación de asesoría contratada. Además, en la tercera se muestra el mensaje de éxito.

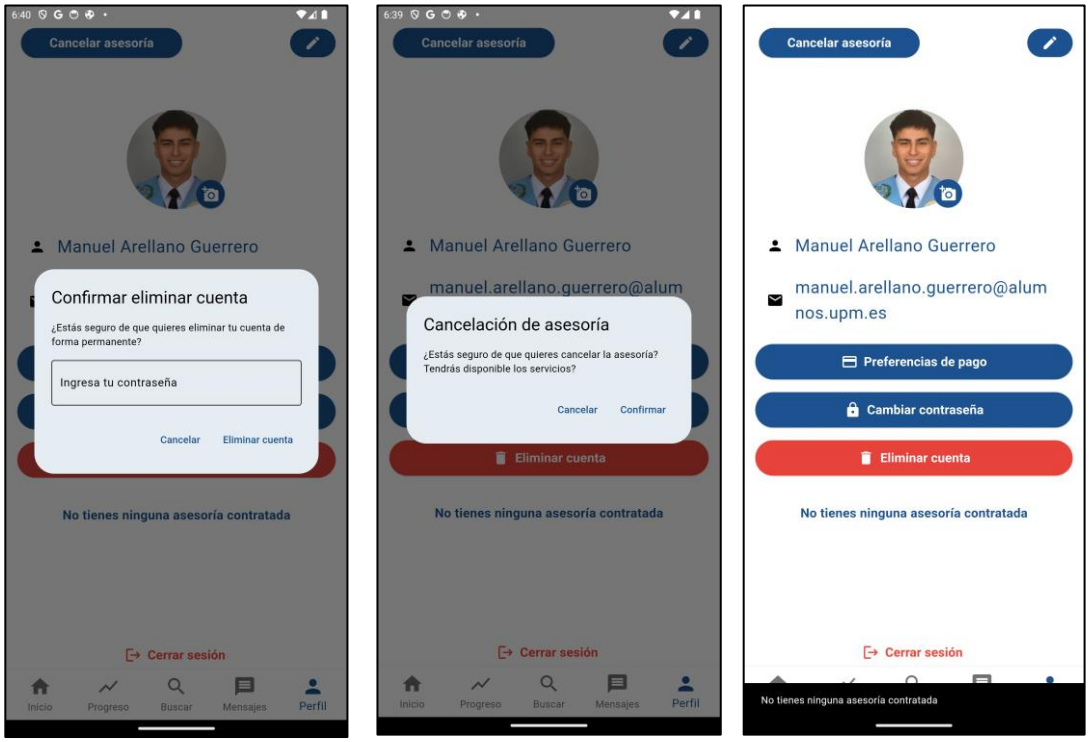


Figura 54: Captura de pantallas de cancelación y eliminación. Fuente: Elaboración propia.

En la Figura 55 se muestra el proceso de búsqueda de asesores con filtros y el perfil profesional de un asesor con sus certificaciones y especialidades, y una descripción detallada con precio y galería de imágenes.

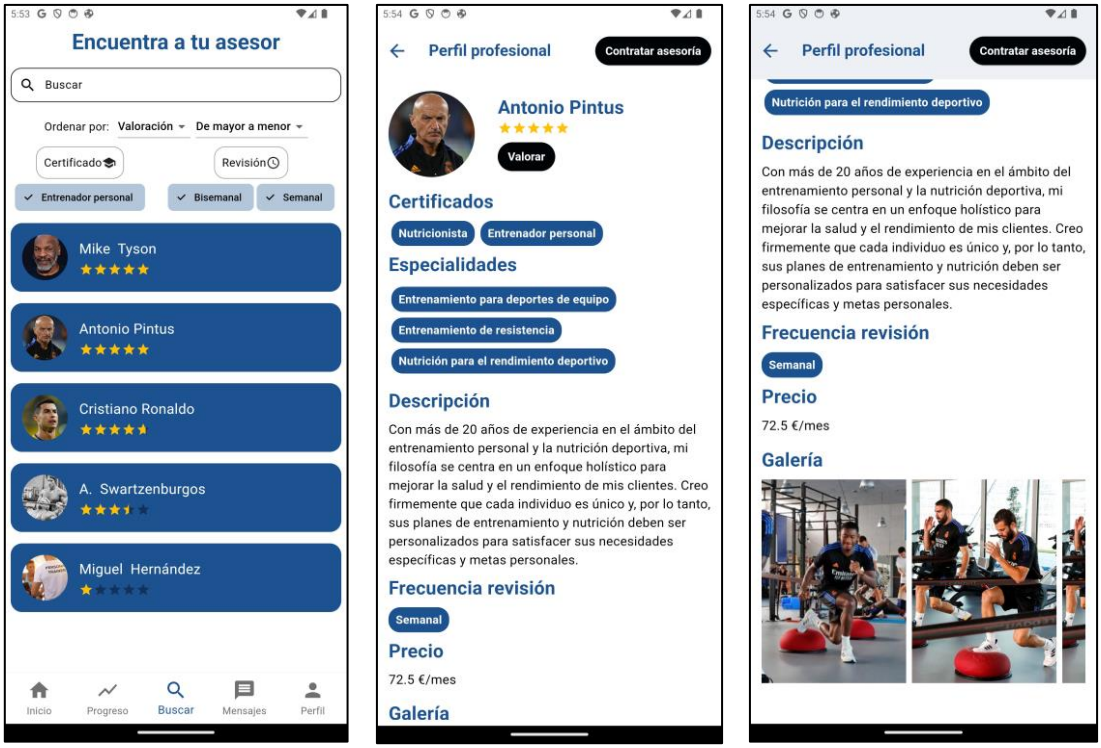


Figura 55: Captura de pantallas de buscar del sistema. Fuente: Elaboración propia.

La Figura 56 presenta cuatro pantallas relacionadas con la confirmación de la contratación de los servicios del asesor, el cuestionario inicial para la personalización de los servicios y la valoración de los asesores.

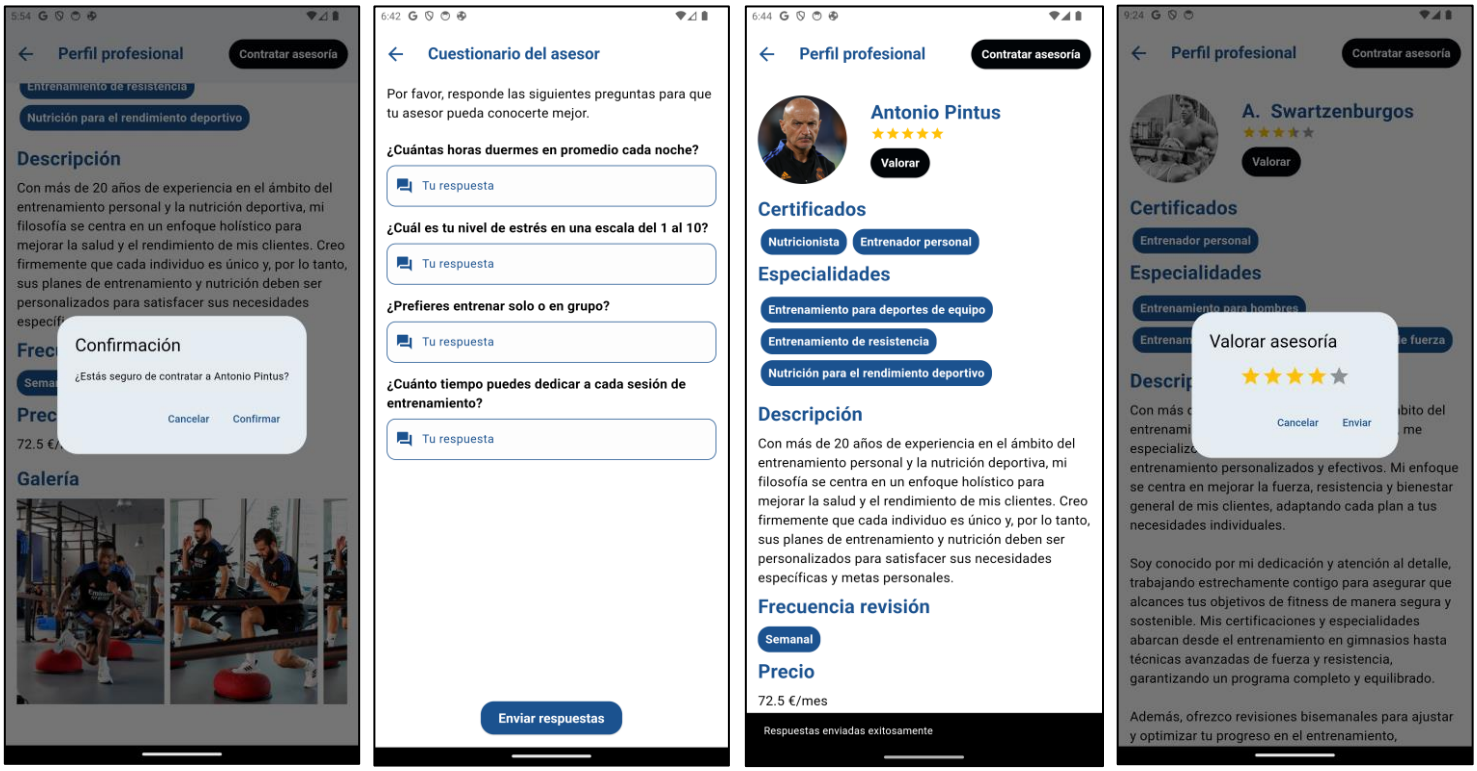


Figura 56: Captura de pantallas de contratación del sistema. Fuente: Elaboración propia.

En la Figura 57 se muestra la funcionalidad de mensajería, permitiendo a los usuarios comunicarse directamente con sus asesores para recibir consejos, resolver dudas y mantener una interacción constante

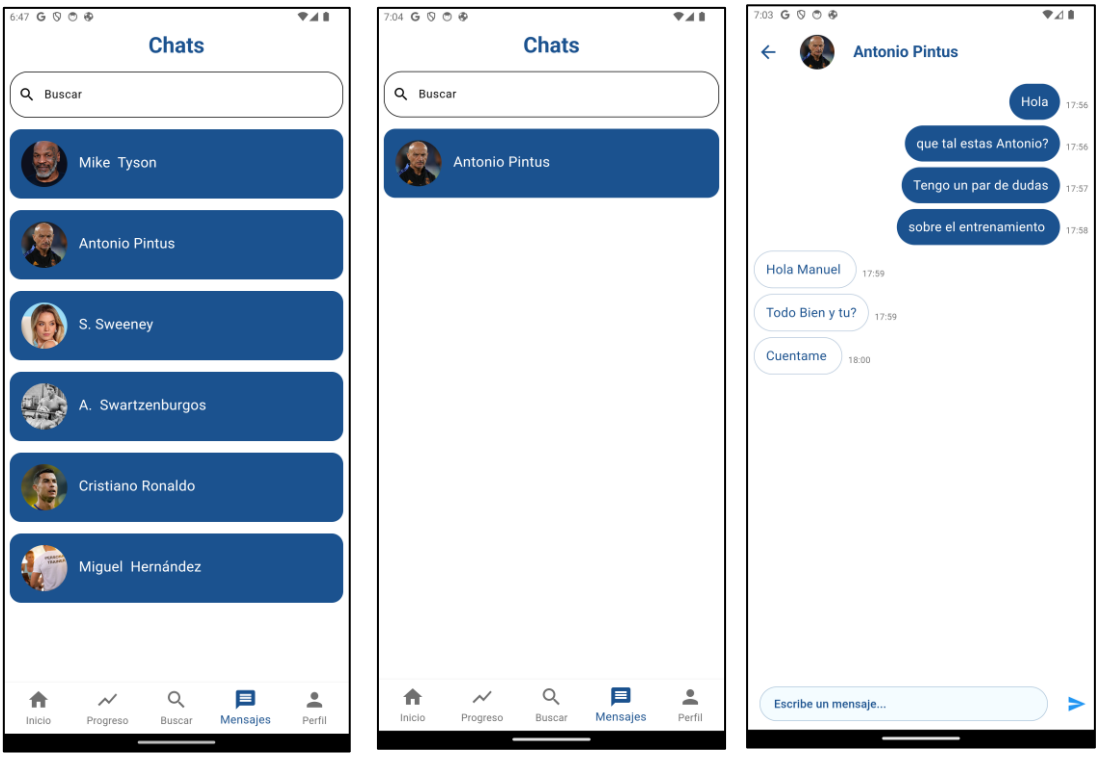


Figura 57: Captura de pantallas del chat del sistema. Fuente: Elaboración propia.

En la Figura 58 se presentan las pantallas que permiten a los usuarios seguir y registrar su progreso diario, así como acceder a sus planes personalizados de nutrición y entrenamiento.

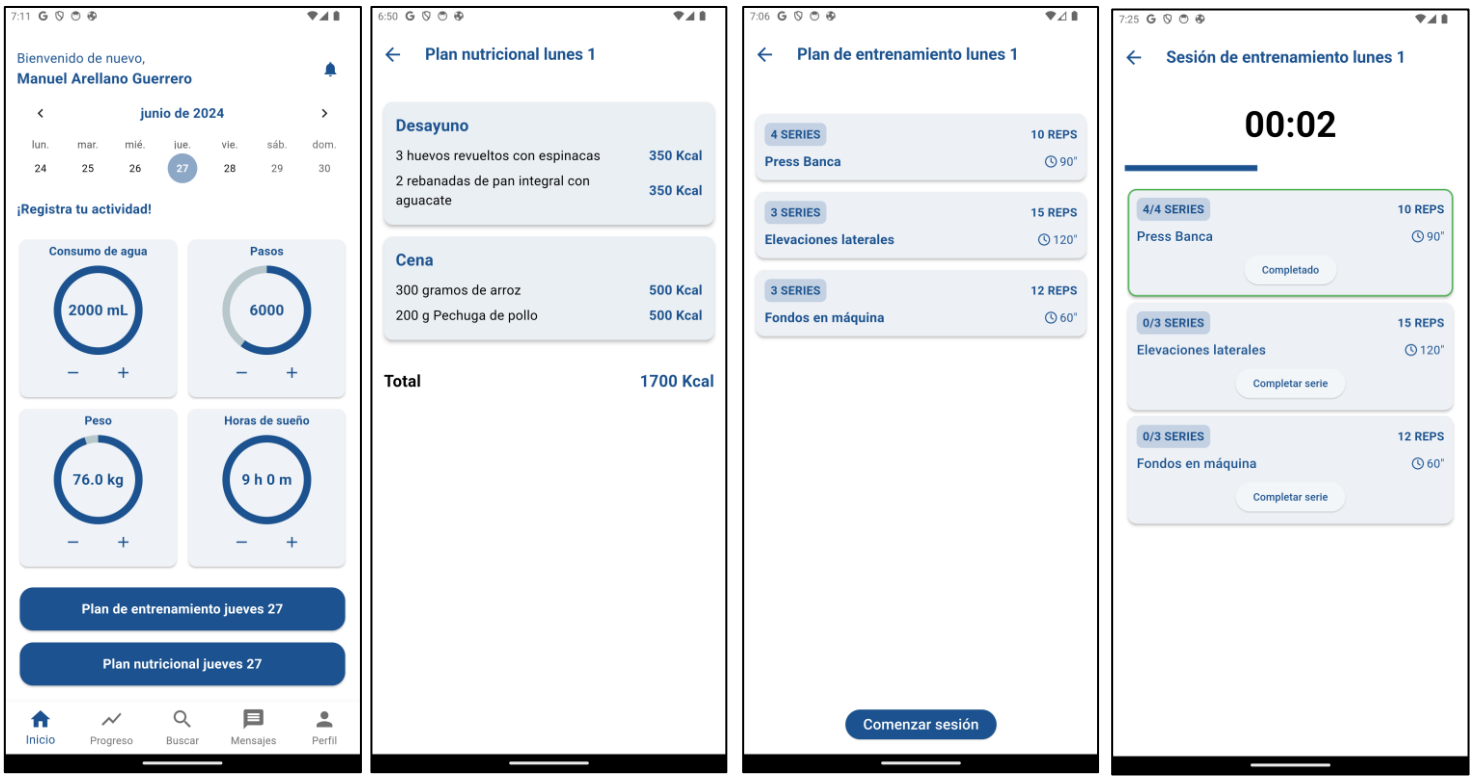


Figura 58: Captura de pantallas de los planes del asesorado. Fuente: Elaboración propia.

En la Figura 59 se presentan dos pantallas de la aplicación MyTrainer que muestran gráficos de seguimiento semanal de las métricas de salud del asesorado.

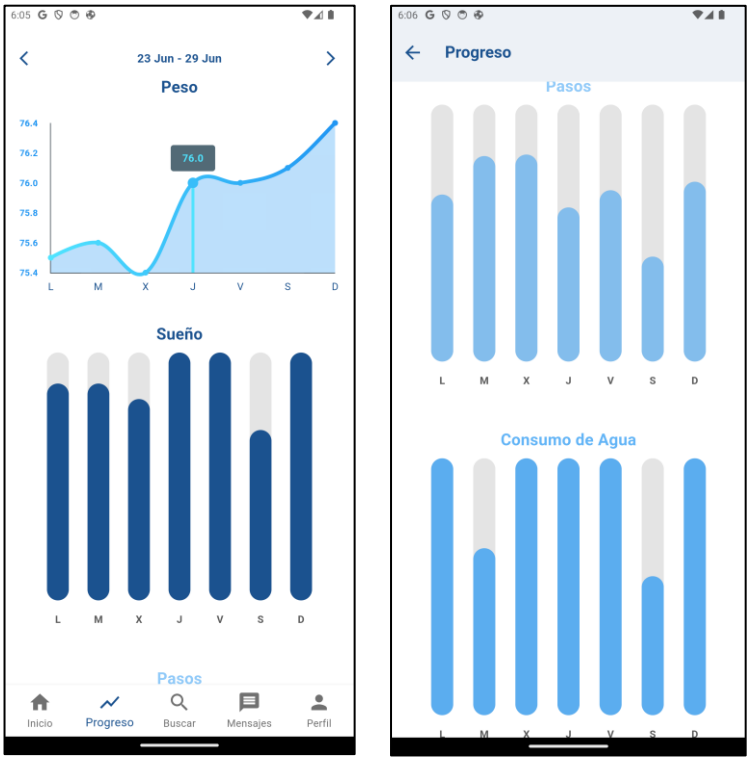


Figura 59: Captura de pantallas del progreso. Fuente: Elaboración propia.

8.1.2 Pantallas para usuarios con rol asesor

En la Figura 60 se muestra cuatro pantallas de la aplicación: una pantalla de inicio que resume la actividad del asesor (clientes, nuevos clientes, revisiones pendientes, visualizaciones del perfil), una lista de clientes actuales, una pantalla de chats con los clientes y el perfil profesional del asesor.

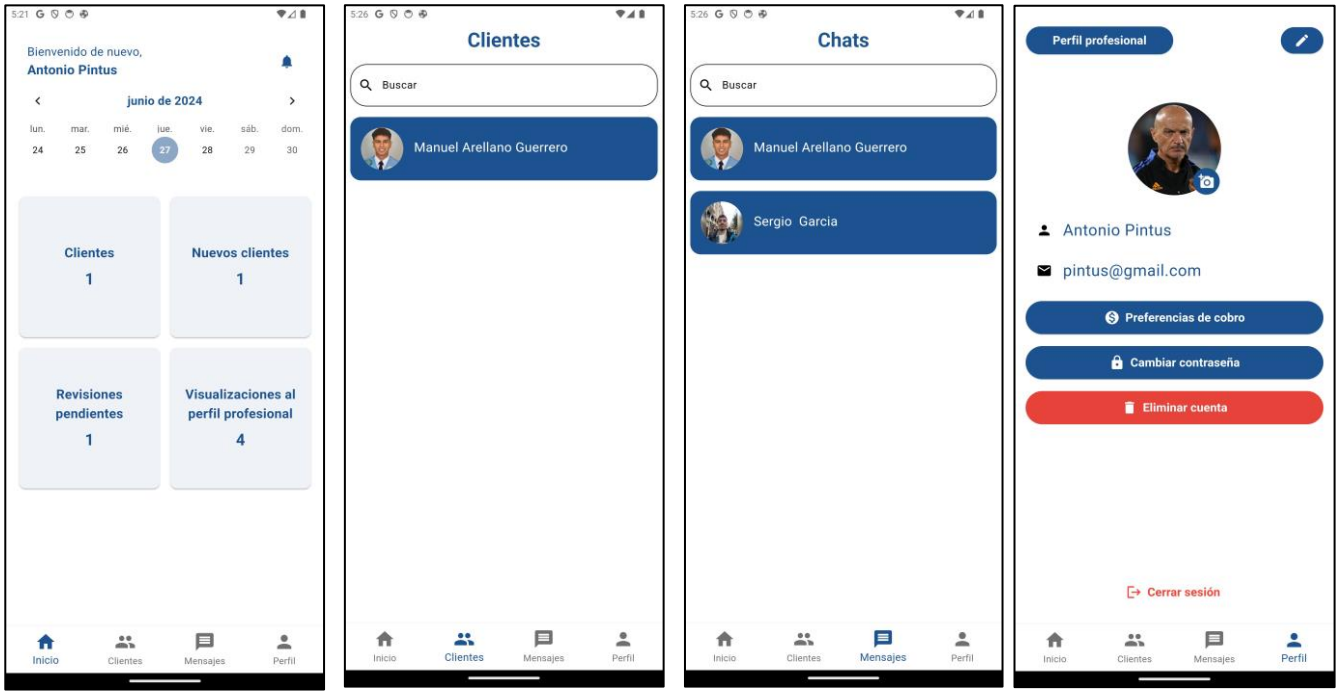


Figura 60: Captura de pantallas principales para asesores. Fuente: Elaboración propia.

En la Figura 61 se presenta tres pantallas del perfil profesional del asesor junto con la pantalla para la creación de un cuestionario inicial para adaptar los planes a las necesidades de los clientes.

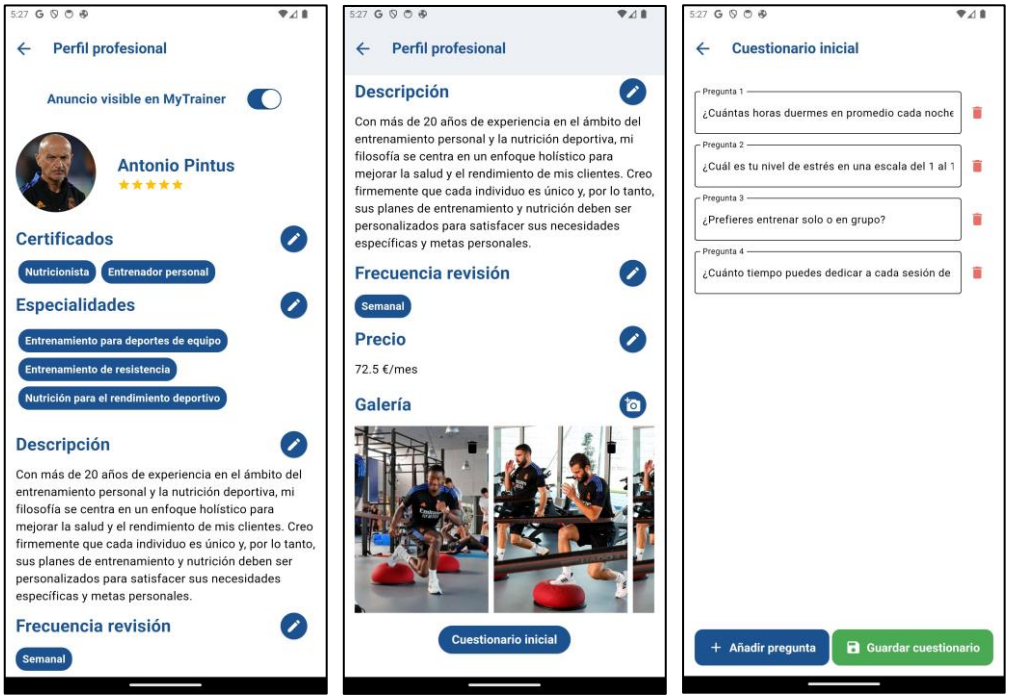


Figura 61: Captura de pantallas del perfil profesional del asesor. Fuente: Elaboración propia.

La Figura 62 detalla cuatro pantallas relacionadas con la gestión de clientes del aseso y la edición del plan nutricional.

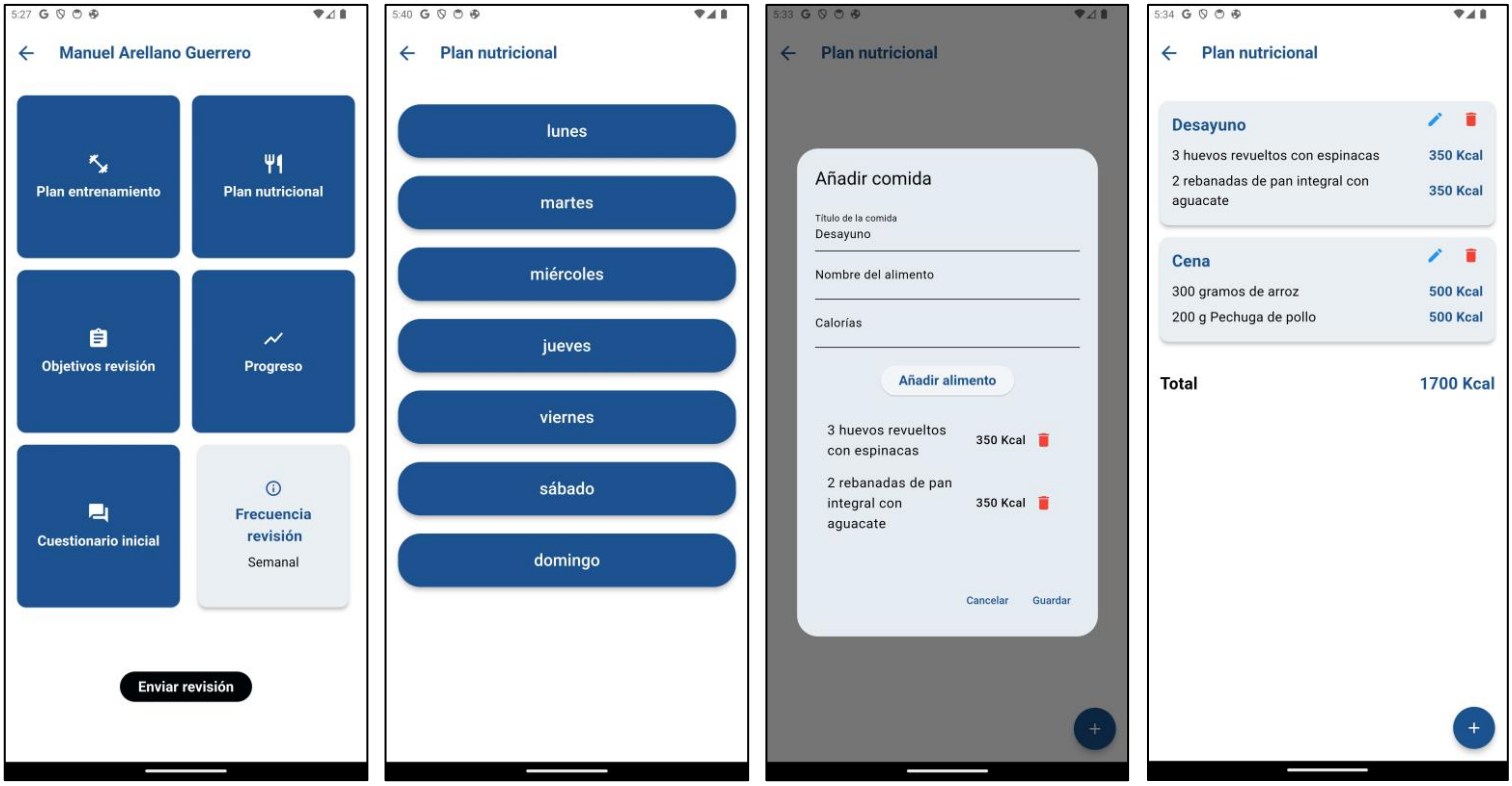


Figura 62: Captura de pantallas de gestión de clientes. Fuente: Elaboración propia.

En la Figura 63 se muestran cuatro pantallas: edición del plan de entrenamiento, cuestionario inicial del usuario, objetivos de revisión (sueño, peso, agua, pasos), y una pantalla que muestra como es la edición de los objetivos de la revisión.

Figura 63: Captura de pantallas de planes, cuestionario y objetivos. Fuente: Elaboración propia.

La Figura 64 presenta dos pantallas de la aplicación enfocadas en el seguimiento del progreso del cliente escogido.

Figura 64: Captura de pantallas del progreso del cliente. Fuente: Elaboración propia.

En la Figura 65 se muestra la funcionalidad de mensajería, permitiendo a los asesores comunicarse directamente con sus clientes para dar *feedback*, consejos, resolver dudas y mantener una interacción constante

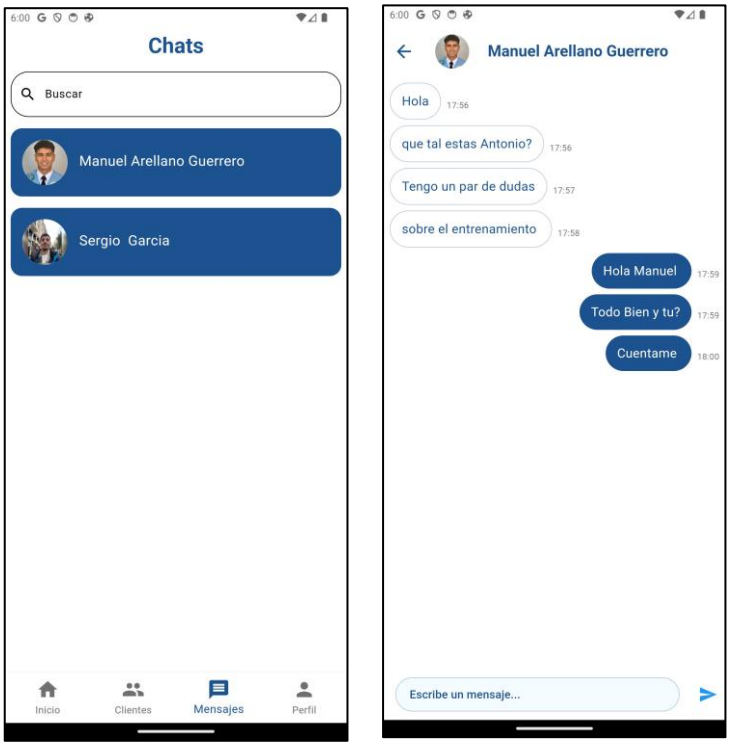


Figura 65: Captura de pantallas del servicio de mensajería. Fuente: Elaboración propia.

8.2 Pruebas de la aplicación

Para garantizar la funcionalidad y usabilidad de la aplicación MyTrainer, se llevarán a cabo una serie de pruebas sobre el producto final. Estas pruebas replicarán las realizadas durante el desarrollo del prototipo de la aplicación, enfocándose en validar que todas las funcionalidades clave operan de manera efectiva en el entorno real de la aplicación.

Grupo de tareas	Tareas	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5	Usuario 6
Gestión de la cuenta	Registro de usuario	33	32	40	33	38	31
	Iniciar sesión	14	15	14	16	15	17
	Modificar información del perfil	13	20	17	13	15	13
	Cerrar sesión	3	4	5	3	4	3
	Eliminar cuenta.	12	16	13	12	11	18
Gestión de asesores	Búsqueda y contratación de asesores deportivos	61	70	42	75	60	87
	Enviar y recibir mensajes	37	42	39	45	51	36
	Valorar asesores	30	29	34	27	29	31
Gestión de la asesoría	Acceder al calendario	8	9	10	11	9	8
	Abandonar asesoría	12	11	13	12	15	12
	Acceso a los planes personalizados	45	62	55	48	27	49
	Registro diario de métricas de salud	29	28	30	29	28	27
	Acceder a un *dashboard* con el progreso	6	7	8	9	7	6

Figura 66: Pruebas de usabilidad de asesorados. Fuente: Elaboración propia.

Grupo de tareas	Tareas	Usuario 1	Usuario 2	Usuario 3	Usuario 4
Gestión de la cuenta	Registro de usuario	33	32	37	33
	Iniciar sesión	14	15	13	16
	Actualizar perfil profesional	75	90	70	82
	Cerrar sesión	4	5	3	4
	Eliminar cuenta	14	13	15	14
Gestión de clientes	Acceder a crear cuestionario inicial	40	32	26	25
	Acceso a gestión de clientes	14	15	13	14
	Acceder a crear planes iniciales	106	90	107	140
	Enviar y recibir mensajes	33	32	34	33

Figura 67: Pruebas de usabilidad de asesor. Fuente: Elaboración propia.

En rojo se marcan las tareas en las que se encontró algún problema. En cuanto a los asesorados, el Usuario 6 tardó 87 segundos en la tarea de "Búsqueda y contratación de asesores deportivos". Aunque no es una cifra preocupante sí que es cierto que es superior a la del resto de usuarios y por ello ha sido resaltada. Este tiempo adicional que tardó el Usuario 6 fue debido a que se detuvo a leer los perfiles profesionales de distintos asesores antes de contratar los servicios de uno. En cuanto al Usuario 5, tardó únicamente 27 segundos en realizar la tarea de “Acceso a los planes personalizados”, esto solo accedió al plan de entrenamiento, pero no comenzó la sesión. En relación con los asesores, el Usuario 4 tardó 140 segundos en "Acceder a crear planes iniciales", le llevo un tiempo excesivo la creación de los planes iniciales, ya que se entretuvo creando más comidas y ejercicios que los otros usuarios.

En general, al igual que en las pruebas realizadas sobre el prototipo, los resultados han sido muy favorables. Los usuarios han experimentado una navegación fluida por la aplicación y se han recibido comentarios positivos de

la mayoría de los participantes en las pruebas. Dado los buenos resultados, se ha concluido que no es necesario realizar cambios que afecten a las funcionalidades y la usabilidad del sistema. Por lo tanto, estas permanecerán sin alteraciones en la versión final de la aplicación, ya que no se identificaron problemas significativos que requieran modificaciones.

8.3 Despliegue

El despliegue de una aplicación como MyTrainer implica varios pasos cruciales para asegurar que esté disponible para los usuarios finales de manera eficiente y sin problemas. En este apartado se detallan los procedimientos y consideraciones esenciales que se deberían seguir para el despliegue de la aplicación en Android.

1. **Preparativos**. Hay que tener el SDK de Flutter y Dart instalados. Además, también se necesita Android Studio para compilar y probar la aplicación.
2. **Crear cuenta de desarrollador en Google Play Console**. Para registrarnos como desarrolladores se necesita un número de teléfono, un correo electrónico valido y una forma de pago ya que hay que pagar una cuota de registro.
3. **Configurar la aplicación para Android**. Es necesario configurar los archivos *AndroidManifest.xml* y *build.gradle* del proyecto Flutter.
4. **Generar archivos de construcción.** Se debe ejecutar el comando *flutter build appbundle* en la terminal para generar el archivo AAB.
5. **Crear aplicación en Google Play Console.** Se debe proporcionar información esencial como el nombre de la aplicación, el idioma, la categoría, el modelo de monetización, la política de privacidad, la gestión de anuncios y la recopilación de datos, entre otros detalles.
6. **Configurar ficha de Play Store.** En este paso hay que completar la ficha de Play Store con una descripción, un icono, capturas de pantalla de alta calidad y una portada. Estos elementos visuales y descriptivos serán la primera impresión que los usuarios tengan de la aplicación.
7. **Crear prueba cerrada.** Antes de publicar la aplicación en producción, es recomendable realizar una prueba cerrada con un grupo de 20 *testers* durante 14 días. Esta fase de prueba permite identificar posibles errores o problemas de usabilidad antes de exponer la aplicación a un público más amplio.
8. **Publicar en Google Play Store.** Una vez finalizada la prueba cerrada y corregidos los aspectos detectados, se puede publicar la aplicación en producción. Desde Google Play Console, se crea una versión de producción, se sube el archivo AAB generado y se envía la aplicación para revisión. Google revisará la aplicación y si cumple con sus políticas y requisitos, la aprobará para su publicación en Play Store.

9 Conclusiones y futuros trabajos

El desarrollo de la aplicación MyTrainer ha sido un proceso enriquecedor que me ha permitido aplicar de manera práctica los conocimientos que he adquirido durante la carrera. Tras el desarrollo de este proyecto se ha obtenido una aplicación que proporciona una plataforma robusta y funcional tanto para asesores como para asesorados, que facilita la conexión entre ambos, la gestión de las asesorías y la comunicación efectiva entre ambos roles.

Con respecto a los objetivos planteados al inicio del proyecto MyTrainer, se puede concluir que se han alcanzado todos, incluido el objetivo principal que era desarrollar una aplicación multiplataforma robusta y funcional tanto para asesores como para asesorados. Esto puede observarse mediante las pruebas realizadas sobre la aplicación y los resultados mostrados en forma de capturas de la aplicación final, que demuestran que se cumplen los requisitos identificados al inicio del proyecto.

Por otra parte, se ha logrado implementar una serie de funcionalidades clave que incluyen la creación y edición de perfiles, la búsqueda de asesores, el registro y seguimiento de planes de entrenamiento y nutrición y un sistema de mensajería para mantener una comunicación fluida entre usuarios, entre otras funcionalidades.

A pesar de los logros obtenidos, existen varias áreas que pueden ser mejoradas o ampliadas en futuros desarrollos:

- **Gestión de cobros y pagos.** Integrar un sistema de pagos para que los usuarios puedan pagar por los servicios de asesoría directamente a través de la aplicación y los asesorados puedan transferirse los ingresos a sus cuentas personales.
- **Subir fotos de estado físico.** Permitir a los usuarios mandar fotos de su estado físico a su asesor a través del chat para que este pueda llevar un seguimiento visual del progreso.
- **Registrar medidas corporales.** Añadir la funcionalidad de registro y seguimiento de medidas corporales, teniendo así otra variable disponible para los asesores para la toma de decisiones sobre los planes de sus clientes.
- **Integración con dispositivos de salud.** Integrar la aplicación con dispositivos de seguimiento de salud, como las pulseras de actividad física y *smartwatches* para una recopilación automática de datos de actividad física y salud.
- **Personalización de planes.** Mejorar las herramientas de creación de planes, permitiendo al asesor guardarse una serie de plantillas creadas por el mismo a partir de las cuales comenzar la personalización de los planes de entrenamiento y nutrición de sus clientes, permitiendo así reducir el tiempo que emplean los asesores en estas tareas.
- **Mejorar la versión web.** Aunque se puede acceder a la aplicación desde Android, iOS y web, la versión web presenta varios aspectos que necesitan mejorarse. Debido a las dimensiones significativamente diferentes en comparación con las otras dos plataformas, es necesario realizar una adaptación gráfica para asegurar que los usuarios puedan utilizar la aplicación de manera óptima, al igual que lo hacen desde los dispositivos móviles. Esta adaptación incluirá la mejora de la interfaz de usuario y la usabilidad en pantallas más grandes, garantizando así una

experiencia de usuario consistente y satisfactoria en todas las plataformas.

- **Añadir anuncios.** Para diversificar las fuentes de ingresos, se implementará la integración de anuncios en la aplicación mediante el uso de Google AdMob [21], que es uno de los muchos servicios que ofrece Firebase. Este servicio permite monetizar aplicaciones a través de anuncios.
- **Implementar el servicio de notificaciones.** Con el objetivo de mantener a los usuarios informados sobre eventos importantes, como mensajes o actualizaciones de planes de entrenamiento y nutrición. Es crucial implementar un sistema de notificaciones, lo que permitirá mejorar la comunicación.

La implementación de estas mejoras no solo aumentará la funcionalidad de la aplicación, sino que también mejorará la experiencia del usuario, haciendo de MyTrainer una herramienta aún más valiosa para la promoción de un estilo de vida saludable y activo.

10 Análisis de impacto

En este capítulo se evaluará el impacto potencial de este proyecto en diferentes dimensiones y se analizará como se relaciona el trabajo realizado durante este proyecto con los Objetivos de Desarrollo Sostenible, ODS, de la Agenda 2030 [22].

En cuanto a la dimensión personal, este proyecto me ha brindado la oportunidad de aplicar y demostrar los conocimientos adquiridos a lo largo de mi formación universitaria, especialmente en áreas relacionadas con el desarrollo de software. He adquirido un profundo entendimiento de un lenguaje y un framework que desconocía por completo, como son Dart y Flutter. Además, este proyecto me ha permitido familiarizarme con herramientas de Backend as a Service, como Firebase. Considero que en un futuro cercano podré aprovechar enormemente los conocimientos adquiridos gracias a esta experiencia, los cuales son ideales para iniciar un proyecto escalable sin necesidad de invertir grandes recursos económicos.

Por otra parte, el proyecto MyTrainer está particularmente alineado con los Objetivos de Desarrollo Sostenible de la Agenda 2030, específicamente con el siguiente:

- ODS 3: Salud y bienestar.

Figura 68: Objetivos de Desarrollo Sostenible. Fuente: ONU.

MyTrainer desempeña un papel crucial en la promoción de la salud y el bienestar al facilitar el acceso a estilos de vida saludables de manera fácil y asequible. A través de la plataforma, los usuarios pueden encontrar a asesores deportivos que les ayudarán a mejorar sus hábitos, incentivando la actividad física y proporcionando un control nutricional personalizado. De esta forma, MyTrainer contribuye a la prevención de enfermedades derivadas del sedentarismo, alineándose directamente con el ODS 3.

En resumen, este proyecto no solo ha sido una valiosa experiencia educativa para mí, sino que también tiene el potencial de generar un impacto positivo significativo en la sociedad, promoviendo estilos de vida saludables y contribuyendo al bienestar general de la población.

11 Bibliografía

[1] JavaScript. Accedido el 1 de abril de 2024. [En línea]. Disponible: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[2] React Native. Accedido el 1 de abril de 2024. [En línea]. Disponible: https://reactnative.dev/

[3] Kotlin. “Kotlin Programming Language”. Accedido el 1 de abril de 2024. [En línea]. Disponible: https://kotlinlang.org/

[4] Dart. "Dart Programming Language”. Accedido el 1 de abril de 2024. [En línea]. Disponible: https://dart.dev/

[5] Google. “Flutter: Beautiful native apps in record time". Accedido el 1 de abril de 2024. [En línea]. Disponible: https://flutter.dev/

[6] Google. "Android Studio and SDK tolos". Accedido el 1 de abril de 2024. [En línea]. Disponible: https://developer.android.com/studio

[7] Microsoft. "Visual Studio Code - Code Editing. Redefined". Accedido el 1 de abril de 2024. [En línea]. Disponible: https://code.visualstudio.com/

[8] Supabase. "Supabase: The open source Firebase alternative”. Accedido el 1 de abril de 2024. [En línea]. Disponible: https://supabase.io/

[9] PostgreSQL. "PostgreSQL: The World's Most Advanced Open-Source Relational Database”. Accedido el 1 de abril de 2024. [En línea]. Disponible: https://www.postgresql.org/

[10] Google Firebase. Accedido el 2 de abril de 2024. [En línea]. Disponible: https://firebase.google.com/

[11] Google FireStore. Accedido el 2 de abril de 2024. [En línea]. Disponible: https://firebase.google.com/docs/firestore?hl=es

[12] FlutterFire. Accedido el 2 de abril de 2024. [En línea]. Disponible: https://firebase.flutter.dev/

[13] Adobe XD. Accedido el 2 de abril de 2024. [En línea]. Disponible: https://www.adobe.com/products/xd.html

[14] Figma. "Figma: the collaborative interface design tool". Accedido el 2 de abril de 2024. [En línea]. Disponible: https://www.figma.com/

[15] Dr. Winston W. Rovce. “Managing the development of large software systems”. Accedido el 4 de junio de 2024. [En línea]. Disponible: https://www.praxisframework.org/files/royce1970.pdf

[16] El modelo en cascada: desarrollo secuencial de software. IONOS Digital Guide. Accedido el 4 de junio de 2024. [En línea]. Disponible: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/

[17] Serverless Architectures. martinfowler.com. Accedido el 8 de junio de 2024. [En línea]. Disponible: https://martinfowler.com/articles/serverless.html

[18] Front End frente a back-end: diferencia entre el desarrollo de aplicaciones - AWS. Amazon Web Services, Inc. Accedido el 8 de junio de 2024. [En línea]. Disponible: https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/#:~:text=El%20front%20end%20es%20aquello,permiten%20que%20la%20aplicación%20funcione.

[19] Xcode - Apple Developer. Apple Developer. Accedido el 8 de junio de 2024. [En línea]. Disponible: https://developer.apple.com/xcode/

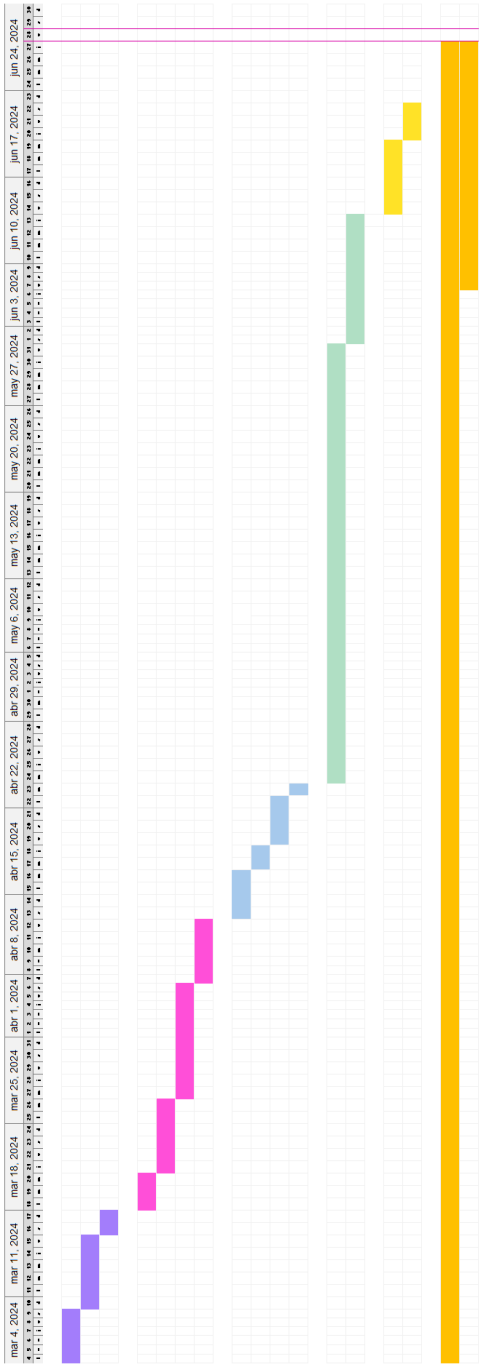
[20] Git. Git. Accedido el 8 de junio de 2024. [En línea]. Disponible: https://git-scm.com/

[21] Google AdMob. ¿Qué es AdMob?. Accedido el 10 de junio de 2024. [En línea]. Disponible: https://admob.google.com/intl/es-419/home/resources/what-is-admob/

[22] Agenda 2030. Iberdrola. Accedido el 12 de junio de 2024. [En línea]. Disponible: https://www.iberdrola.com/sostenibilidad/comprometidos-objetivos-desarrollo-sostenible/que-es-agenda-2030

Anexos

Anexo A: Diagrama de Gantt definitivo



TAREA	PROGRESO	INICIO	FIN
Configuración y aprendizaje			
Toma de contacto con Flutter	0%	3/4/24	3/9/24
Aprendizaje del lenguaje Dart	0%	3/10/24	3/15/24
Configuración del entorno de desarrollo	0%	3/16/24	3/17/24
Diseño			
Investigación de necesidades	0%	3/18/24	3/20/24
Toma de requerimientos	0%	3/21/24	3/26/24
Diseño de la arquitectura de la aplicación	0%	3/27/24	4/6/24
Diseño de la base de datos	0%	4/7/24	4/12/24
Diseño de interfaz de usuario			
Desarrollo de un prototipo	0%	4/13/24	4/16/24
Pruebas sobre el prototipo	0%	4/17/24	4/18/24
Diseño de pantallas de alta fidelidad	0%	4/19/24	4/22/24
Realización diagrama de navegabilidad final	0%	4/23/24	4/23/24
Desarrollo			
Desarrollo de las funcionalidades clave	0%	4/24/24	5/31/24
Desarrollo del backend	0%	6/1/24	6/13/24
Pruebas y validación			
Pruebas de usabilidad con usuarios reales	0%	6/14/24	6/19/24
Análisis de feedback y realización de ajustes	0%	6/20/24	6/22/24
Memoria y presentación			
Redacción de la memoria	0%	3/4/24	6/27/24
Elaboración de la presentación	0%	6/7/24	6/27/24

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Sun Jun 30 23:32:10 CEST 2024
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)