Trabajo de Fin de Grado

SEGUIMIENTO DE PESO

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Carlos Alberto Montoya Hidalgo

TUTORIZADO POR:
José Fortes Gálvez

Fecha [06/2025]

ULPGC Universidad de Las Palmas de Gran Canaria | EII ESCUELA DE INGENIERÍA INFORMÁTICA

TFT04

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D./Dª Carlos Alberto Montoya Hidalgo, autor del trabajo Seguimiento de Peso, correspondiente a la titulación Grado en Ingeniería Informática.

SOLICITA

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida, haciendo constar que

[X] se autoriza / [ ] no se autoriza la grabación en audio de la exposición y turno de preguntas.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

[x] Se ha iniciado o hay intención de iniciarlo (defensa no pública).

[ ] No está previsto.

Y para que así conste firma la presente. (fecha en firma electrónica)

El/La estudiante

Fdo._____

A rellenar y firmar **obligatoriamente** por el/la/los/las tutores

En relación con la presente solicitud, se informa:(firmar donde corresponda)

Positivamente (en caso de detección de copia, esta firma quedará invalidada)	Negativamente (justificación en TFT05)
Fdo._____	Fdo._____

DIRECTOR DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

Agradecimientos

Mis más sinceros agradecimientos a todas las personas que me han acompañado durante esta etapa de mi vida.

a mi familia por su apoyo incondicional,
a mis amigos por su compañía y ánimo,
y a mis profesores por su guía y enseñanzas.

Gracias a todos por acompañarme en este camino.

Resumen

WeightTrackerTFT es una aplicación móvil enfocada en el seguimiento de entrenamientos orientados a la pérdida de peso de manera saludable. El objetivo principal es proporcionar una experiencia sencilla, accesible e intuitiva a usuarios con distintos niveles de experiencia, motivándolos a iniciar o mantener la actividad física diaria y mejorar su salud.

La aplicación está especialmente enfocada en ejercicios de gimnasio y levantamiento de pesas, permitiendo a los usuarios personalizar sus rutinas, registrar sus entrenamientos diarios, analizar su progreso y evolución física mediante gráficos y estadísticas claras. De esta manera, se busca fomentar hábitos saludables que beneficien la movilidad y la calidad de vida.

Abstract

WeightTrackerTFT is a mobile application focused on tracking workouts aimed at healthy weight loss. The main goal is to provide a simple, accessible, and intuitive experience for users with different levels of training experience, motivating them to start or maintain their daily physical activity and improve their health. The app is specifically designed for gym exercises and weightlifting, allowing users to customize routines, log their daily workouts, and analyse their progress and physical development through clear graphs and estatistas. In doing so, it aims to promote healthy habits that enhance mobility and overall quality of life.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

"El progreso consiste en el cambio"

Miguel de Unamuno

Durante esta sección se presenta la aplicación Android **WeightTrackerTFT**, desarrollada como parte del Trabajo de Fin de Título (TFT). Su propósito principal es facilitar el seguimiento de entrenamientos mediante rutinas de gimnasio, promoviendo la disciplina, la constancia y un estilo de vida saludable. Al estar orientada a dispositivos Android, su alcance es amplio gracias a la alta demanda de este sistema operativo en el mercado.

Para facilitar la comprensión de ciertos términos técnicos, al final del documento se incluye un **glosario** que reúne todas las siglas y conceptos relevantes utilizados a lo largo del trabajo.

1.1. Contexto

El desarrollo de **WeightTrackerTFT** surge como respuesta a la creciente tendencia global hacia un estilo de vida más activo y saludable. En este escenario, la pérdida de peso continúa siendo uno de los objetivos más frecuentes entre quienes practican ejercicio físico. A través de la combinación de entrenamiento regular y una alimentación equilibrada, es posible reducir el porcentaje de grasa corporal de manera efectiva. Adicionalmente, la práctica constante de ejercicio incrementa la masa muscular, lo que eleva la **tasa metabólica basal** TMB y favorece la quema de calorías incluso en estado de reposo.

El mercado de aplicaciones deportivas cuenta con soluciones consolidadas como **Nike Training Club** o **FitKeeper**. Aunque **WeightTrackerTFT** se inspira en algunas de sus funcionalidades, se diferencia por adoptar un enfoque minimalista, centrado en la simplicidad y la eficiencia. Este enfoque busca optimizar la experiencia de usuario mediante una navegación intuitiva y ágil. Además, se facilita el seguimiento del progreso personal mediante

visualizaciones claras y datos detallados sobre el rendimiento a lo largo del tiempo. El proyecto se ha desarrollado siguiendo una metodología ágil con enfoque iterativo e incremental. Esta estrategia ha permitido organizar el trabajo en ciclos cortos, facilitando la incorporación continua de mejoras y la evaluación progresiva de nuevas funcionalidades.

1.2. Motivación

WeightTrackerTFT busca satisfacer la necesidad de herramientas tecnológicas que apoyen la adopción de hábitos saludables. Muchas personas enfrentan dificultades para iniciar o mantener rutinas de entrenamiento efectivas. Esta aplicación simplifica el proceso con una interfaz accesible, apta para usuarios de cualquier nivel de experiencia. La integración de tecnologías como bases de datos en la nube y asistentes virtuales garantiza una solución interactiva, escalable y alineada con las tendencias tecnológicas actuales [16].

Capítulo 2

Estado actual y objetivos iniciales

En la actualidad, el sector de las aplicaciones móviles para la salud y el bienestar está en constante expansión, con una gran cantidad de soluciones disponibles para ayudar a los usuarios a registrar su actividad física. Sin embargo, a pesar de la gran oferta, aún existen oportunidades para mejorar la experiencia de usuario mediante la simplificación de las interfaces y ofrecer soluciones más personalizadas.

2.1. Estado del arte

En esta sección se analizará el estado actual de las diferentes opciones disponibles en el mercado. En el campo de las aplicaciones móviles orientadas al seguimiento del entrenamiento físico y la mejora del rendimiento, existen múltiples alternativas ampliamente utilizadas por los usuarios, tales como:

1. **Nike Training Club**[13]. Es una de las aplicaciones más consolidadas en el ámbito del entrenamiento personal. Ofrece una amplia variedad de rutinas clasificadas por nivel, duración y objetivo, incluyendo entrenamientos sin equipamiento, de fuerza, movilidad y resistencia. No obstante, su enfoque está más centrado en rutinas generales que en el seguimiento específico de ejercicios y el control progresivo del rendimiento físico. Además, presenta una interfaz densa que puede ser abrumadora para usuarios que solo buscan registrar entrenamientos de fuerza de forma simple y rápida.

2. **FitKeeper** [14]. Esta aplicación está orientada a usuarios que entrenan en el gimnasio, con un fuerte énfasis en el registro de ejercicios de musculación. La aplicación permite llevar un control detallado de repeticiones, cargas, tiempos de descanso y evolución semanal o mensual mediante gráficas.

3. **Hevy** [15]. Es una herramienta moderna y visualmente atractiva para quienes practican entrenamiento de fuerza. Permite crear rutinas, registrar datos precisos y consultar gráficos de progreso con un diseño amigable. Aunque ofrece una experiencia muy completa, requiere tiempo para configurarse y aprovechar todo su potencial.

2.2. Comparativa y aportaciones de WeightTrackerTFT

WeightTrackerTFT ha sido diseñada considerando las virtudes y limitaciones de las aplicaciones previamente analizadas. Su enfoque es minimalista, evitando la sobrecarga de información y facilitando la navegación para cualquier tipo de usuario. A diferencia de otras herramientas con diseños complejos o de acceso restringido, WeightTrackerTFT permite registrar ejercicios, gestionar rutinas y consultar el progreso de forma clara y eficiente desde el primer uso.

Además, incorpora funcionalidades diferenciadoras como la integración con un chatbot basado en inteligencia artificial, que permite realizar consultas de manera rápida y personalizada, algo no presente en la mayoría de las alternativas mencionadas. El uso de tecnologías modernas como Firebase y Jetpack Compose contribuye también a una mejor mantenibilidad, escalabilidad y rendimiento de la aplicación.

En resumen, WeightTrackerTFT busca cubrir un espacio intermedio entre aplicaciones altamente completas pero complejas, y otras que, aunque sencillas, resultan limitadas en funcionalidad. Su propuesta se centra en ofrecer una experiencia optimizada, accesible y enfocada en la mejora del rendimiento físico a través del seguimiento ordenado y visual de los entrenamientos.

2.3. Objetivos, motivaciones y antecedentes

El proyecto **WeightTrackerTFT** nace del interés por profundizar en el desarrollo de aplicaciones Android, una tecnología líder en el mercado móvil debido a su amplia adopción global [17]. Este entorno ofrece una oportunidad clave para consolidar competencias técnicas y fomentar el crecimiento profesional. Complementando esta motivación técnica, el interés personal por el ámbito deportivo y de la salud inspiró la creación de una aplicación diseñada para promover un estilo de vida activo. Los objetivos principales son dos: fortalecer habilidades en el desarrollo Android, aprovechando su relevancia laboral, y facilitar a los usuarios el seguimiento de rutinas deportivas mediante una interfaz intuitiva y funcionalidades orientadas a la salud y el entretenimiento.

Capítulo 3

Aportaciones del trabajo

3.1. Principales aportaciones

El desarrollo de esta aplicación representa una contribución significativa en el ámbito del bienestar físico y el uso de tecnologías móviles aplicadas a la salud personal. Se ha creado una herramienta capaz de registrar, visualizar y analizar la evolución de rutinas de entrenamiento orientadas a la pérdida de peso, brindando al usuario recomendaciones personalizadas y funcionalidades interactivas que promueven la adherencia al ejercicio.

Desde un punto de vista técnico, se ha implementado una arquitectura moderna basada en MVVM, integrando Jetpack Compose, Firebase y otros servicios en la nube, lo cual permite una solución robusta, escalable y centrada en la experiencia del usuario.

El impacto esperado se encuentra en la mejora de la autogestión del entrenamiento físico por parte de los usuarios, así como en la digitalización de rutinas de seguimiento que antes se realizaban manualmente o mediante hojas de cálculo. Esta herramienta puede ser especialmente útil para personas con objetivos de pérdida de peso, entrenadores personales y centros deportivos.

3.2. Competencias específicas

Durante el desarrollo del trabajo se han abordado y aplicado múltiples competencias específicas del ámbito de la ingeniería informática, entre las que destacan:

- ✔ **CI1**: Diseño, desarrollo y evaluación de una aplicación asegurando fiabilidad, seguridad y calidad, conforme a principios éticos y normativas vigentes.
- ✔ **CI2**: Planificación y gestión integral del proyecto desde su concepción hasta su puesta en producción, valorando su impacto técnico y social.
- ✔ **CI5**: Administración y mantenimiento de bases de datos y servicios alojados en la nube.

- ✔ **CI8**: Construcción de una aplicación robusta, segura y eficiente utilizando los lenguajes y paradigmas adecuados.
- ✔ **CI12**: Diseño y utilización de una base de datos no relacional (Firestore) adaptada a los requerimientos del sistema.
- ✔ **CI16**: Aplicación de principios y metodologías de ingeniería del software a lo largo de todo el ciclo de vida del desarrollo.
- ✔ **CI17**: Creación de una interfaz que garantiza accesibilidad y usabilidad, considerando necesidades de distintos perfiles de usuario.
- ✔ **TI1**: Análisis de necesidades de usuarios reales en el ámbito de las tecnologías de la información.
- ✔ **TI3**: Aplicación de metodologías centradas en el usuario que aseguran accesibilidad, ergonomía y usabilidad.
- ✔ **TI6**: Implementación de un sistema interactivo basado en tecnologías móviles, servicios web y computación en la nube.
- ✔ **TI7**: Gestión de la privacidad y seguridad de los datos personales almacenados en la plataforma.

3.3. Alineamiento con los Objetivos de Desarrollo Sostenible

Este proyecto guarda relación directa con varios de los Objetivos de Desarrollo Sostenible (ODS) establecidos por la Agenda 2030. A continuación, se detallan aquellos con mayor grado de alineamiento:

- ✔ **ODS 3 – Salud y Bienestar (Alto)**: El objetivo principal de la aplicación es fomentar hábitos saludables mediante el seguimiento de entrenamientos, lo cual promueve la mejora de la salud física y el bienestar personal.
- ✔ **ODS 4 – Educación de calidad (Medio)**: La aplicación ofrece recomendaciones y explicaciones sobre ejercicios y técnicas de entrenamiento, contribuyendo a la educación del usuario en materia de actividad física.
- ✔ **ODS 8 – Trabajo decente y crecimiento económico (Bajo)**: La implementación de este tipo de soluciones puede apoyar a profesionales del deporte y del entrenamiento personal en su actividad laboral.
- ✔ **ODS 9 – Industria, innovación e infraestructuras (Medio)**: La solución desarrollada hace uso de tecnologías innovadoras, arquitecturas modernas y servicios distribuidos en la nube.

- ✔ **ODS 10 – Reducción de las desigualdades (Bajo)**: Al tratarse de una aplicación accesible y sin coste, contribuye a reducir barreras para el acceso al conocimiento sobre entrenamiento físico.
- ✔ **ODS 17 – Alianzas para lograr los objetivos (Bajo)**: Existe posibilidad de integración futura con otras plataformas o entidades interesadas en el fomento de la salud mediante la tecnología.

Cuadro 3.1: Grado de relación del TFT con los objetivos de desarrollo sostenible.

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

Capítulo 4

Desarrollo

4.1. Metodología

Para el desarrollo del Trabajo de Fin de Título se ha seguido una metodología iterativa e incremental, con el objetivo de construir la aplicación de forma progresiva y controlada, permitiendo así comprobar y validar resultados en cada fase y realizar mejoras continuas. En cada iteración se repite el mismo proceso de trabajo donde se revisan los requisitos, tanto funcionales como no funcionales, así como la implementación y evaluación de nuevas funcionalidades.

Esta metodología permite estar preparados ante posibles cambios durante el desarrollo con el fin de obtener versiones funcionales del sistema desde fases iniciales. Mediante ciclos iterativos, cada incremento ha permitido añadir nuevas funcionalidades, optimizar las existentes y asegurar una mayor calidad del producto final.

Para apoyar la aplicación de la metodología y el desarrollo del proyecto, se emplearon diversas herramientas, entre ellas Trello, Git y GitHub [11, 18, 12]. Asimismo, para el aprendizaje y la correcta implementación de tecnologías como Jetpack Compose, se recurrió tanto al curso formativo de Appcademy [1] como a la documentación oficial proporcionada por Android Developers [9].

4.2. Herramientas y tecnologías utilizadas

En esta parte, describiremos las herramientas utilizadas para el progreso del proyecto.

4.2.1. Entornos de desarrollo integrado (IDE)

En este proyecto, se utilizaron dos entornos de desarrollo integrado IDE complementarios: **Android Studio** para el desarrollo de aplicaciones Android y **Visual Studio Code** para

el desarrollo del backend con JavaScript.

Android Studio

Android Studio [22], el IDE oficial para la plataforma Android [2] desarrollado por Google, ofrece herramientas avanzadas para la creación, depuración y optimización de aplicaciones Android.

- ✔ **Emulador avanzado**: Incluye un emulador rápido y configurable que simula diversos dispositivos y versiones de Android sin necesidad de hardware físico.
- ✔ **Herramientas de Jetpack Compose**: Proporciona previsualizaciones en tiempo real y herramientas específicas para desarrollar interfaces declarativas y reactivas.
- ✔ **Integración con Firebase**: Simplifica la conexión con servicios como Authentication, Firestore y Crashlytics directamente desde el IDE.
- ✔ **Control de versiones**: Incluye interfaces gráficas para gestionar ramas, **commits** y **pull request** con Git y GitHub.

Visual Studio Code

Visual Studio Code [21] (VSC) es un editor ligero y multiplataforma optimizado para el desarrollo de aplicaciones web y backend, utilizado en este proyecto para programar en JavaScript con Node.js y Express.

- ✔ **Terminal integrado**: Permite ejecutar comandos de npm, iniciar servidores Node.js y visualizar **logs** sin salir del editor.
- ✔ **Extensiones de linting**: Plugins como ESLint y Prettier garantizan consistencia en el estilo del código y detección temprana de errores.

4.2.2. Trello

Trello [11] es una herramienta de gestión de proyectos basada en el enfoque Kanban, utilizada en este proyecto para organizar de forma visual y estructurada las tareas correspondientes a cada fase de desarrollo 4.1.

- ✔ **Listas por estado**: Las tareas se agrupan en listas que reflejan su estado actual, lo que facilita el seguimiento del avance de manera visual.
- ✔ **Tarjetas informativas**: Las tareas se representan mediante tarjetas donde se puede incluir una descripción detallada, listas de verificación, archivos adjuntos, etiquetas y fechas límite.

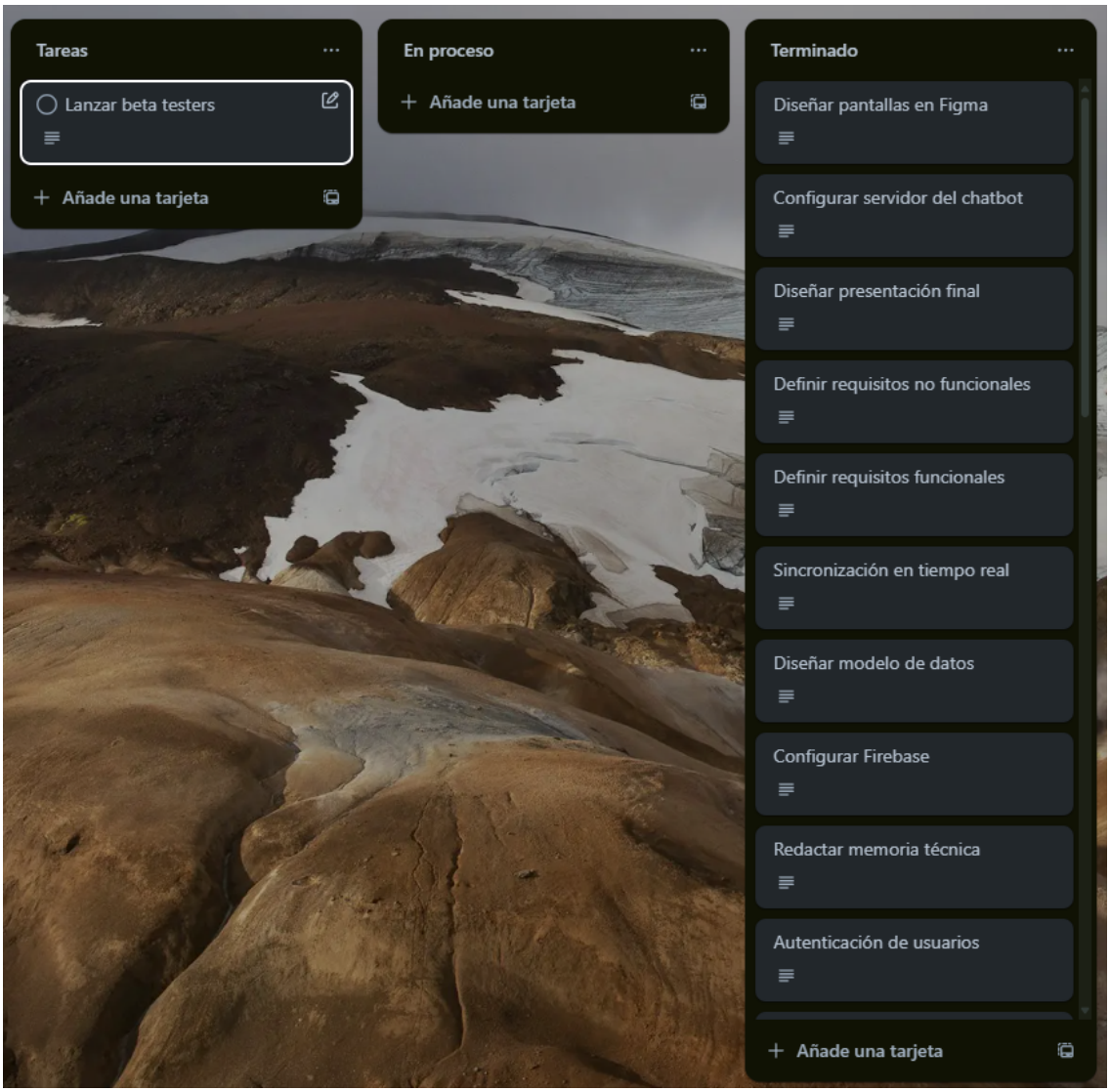

Ilustración 4.1: Tablero Trello utilizado para organizar las tareas del proyecto.

4.2.3. Figma

Figma [20] es una herramienta de diseño colaborativo utilizada en este proyecto para crear prototipos de interfaces y garantizar coherencia visual en la aplicación.

- ✔ **Componentes reutilizables**: Soporta la creación de componentes (botones, tarjetas, iconos) actualizables globalmente para mantener consistencia visual.
- ✔ **Estilos centralizados**: Gestiona paletas de colores, tipografías y efectos de forma unificada, asegurando uniformidad en el diseño.

Figma facilitó la representación de la idea principal de la aplicación, de manera que siempre había un boceto para seguir una lógica. Pero, a medida que se avanzaba en la aplicación, el diseño cambió a uno más uniforme y más receptivo para un usuario.

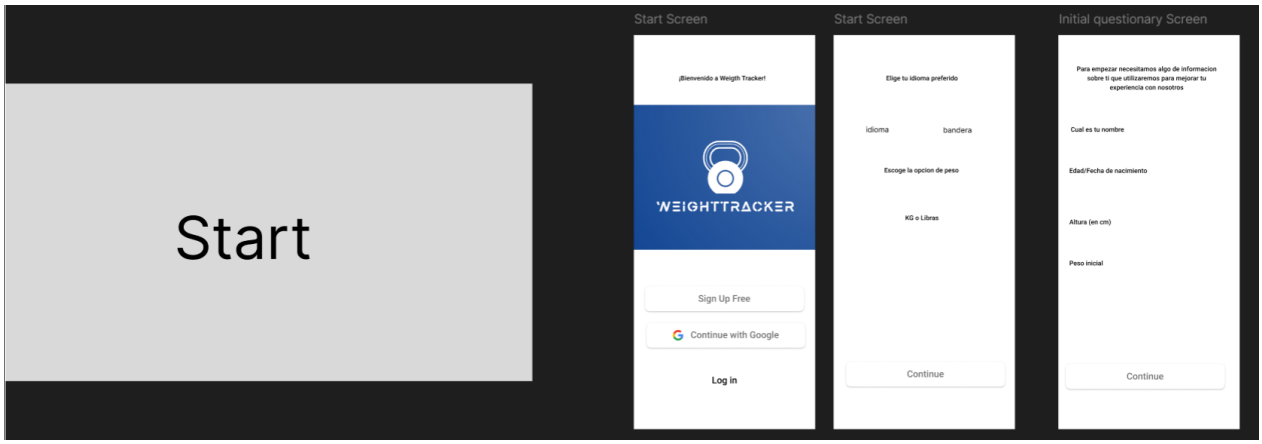

Ilustración 4.2: Mock-up de la pantalla del cuestionario inicial.

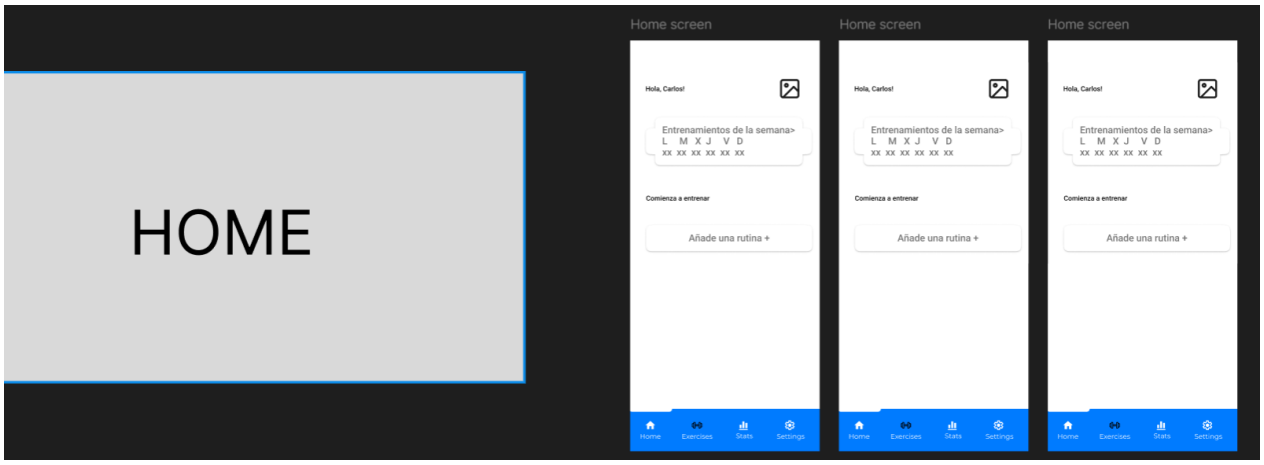

Ilustración 4.3: Mock-up de la pantalla home.

4.2.4. Control de versiones

GitHub [12] es una plataforma creada para que los desarrolladores suban el código de sus aplicaciones y un sistema de control de versiones distribuido para gestionar y rastrear cambios en el código, lo cual es muy importante en proyectos con más personas para revisar y asegurar buenas prácticas en el código. De acuerdo con la página web de git [18], ”Git te permite y te anima a tener múltiples ramas locales que pueden ser completamente independientes entre sí. La creación, fusión y eliminación de esas líneas de desarrollo tarda unos segundos.”Y de forma más concreta:

Git

- ✔ **Sistema distribuido**: Cada desarrollador cuenta con un repositorio completo que incluye el historial de cambios, habilitando el trabajo sin conexión y la sincronización

posterior.

- ✔ **Ramas**: Permite crear y fusionar ramas de manera rápida y eficiente para desarrollar funcionalidades de forma aislada.
- ✔ **Staging area**: Proporciona una zona intermedia para revisar y organizar cambios antes de confirmarlos mediante `commits`.
- ✔ **Reversión de cambios**: Comandos como `Git revert` y `Git reset` facilitan subir el código, revertir cambios o deshacer commits anteriores.

GitHub

- ✔ **Repositorios remotos**: Ofrece hospedaje centralizado de proyectos Git, facilitando la colaboración y el acceso desde cualquier ubicación.
- ✔ **Pull Requests**: Permite proponer, revisar y aprobar cambios antes de integrarlos en la rama principal, optimizando la calidad del código.

4.2.5. Firebase

Firebase [3] es una plataforma de Google para el desarrollo de aplicaciones web y móviles, ofreciendo backend en la nube. Sus principales funciones son base de datos en tiempo real, autenticación de usuarios y almacenamiento en la nube.

- ✔ **Firebase Authentication**: servicio para la autenticación de usuarios, permitiendo la creación de cuentas, inicio de sesión y manejo de sesiones mediante correo electrónico, Google, entre otros métodos.
- ✔ **Firestore**: base de datos en la nube de Firebase utilizada para almacenar y sincronizar datos de la aplicación en tiempo real. Firestore fue crucial para gestionar los datos relacionados con usuarios, entrenamientos y ejercicios.
- ✔ **Firebase Crashlytics**: herramienta para el monitoreo y reporte de errores en la aplicación, que permite identificar y solucionar problemas rápidamente, mejorando la estabilidad del sistema.

4.2.6. Dialogflow

Dialogflow [19], una plataforma de Google [5], permite desarrollar interfaces conversacionales basadas en inteligencia artificial, como chatbots y asistentes virtuales. En este proyecto, se empleó **Dialogflow ES** [19] (edición Essential), una versión optimizada para flujos conversacionales simples y aplicaciones de escala pequeña o mediana, que utiliza procesamiento de lenguaje natural (NLP) para crear interacciones fluidas y naturales con los usuarios.

Características de Dialogflow ES

- ✔ **Interfaz conversacional**: Facilita el diseño y entrenamiento de modelos de conversación para interacciones naturales y dinámicas con los usuarios.
- ✔ **Soporte multilingüe**: Permite crear chatbots en múltiples idiomas, adaptándose a diversos contextos lingüísticos y culturales.
- ✔ **Integración multiplataforma**: Compatible con aplicaciones móviles, sitios web, Google Assistant, Facebook Messenger y otras plataformas de mensajería.
- ✔ **Procesamiento de lenguaje natural (NLP)**: Emplea inteligencia artificial para interpretar y responder con precisión a las consultas de los usuarios.
- ✔ **Gestión de intents y entidades**: Utiliza intents y entidades para estructurar flujos conversacionales, optimizando la experiencia del usuario.
- ✔ **Integración con Firebase**: Simplifica la conexión con Firebase para almacenar datos y personalizar interacciones según el contexto del usuario.

4.2.7. Persistencia de datos: Room

Room [23], una biblioteca de persistencia de Android incluida en la suite de Jetpack, se utilizó en este proyecto para gestionar el almacenamiento local de datos en la aplicación a la hora de elegir el idioma en caso de querer cambiarlo, ya que de primeras se ve el idioma de tu dispositivo.

Características de Room

- ✔ **Abstracción de SQLite**: Ofrece una interfaz sencilla para realizar operaciones de bases de datos relacionales, reduciendo código repetitivo.
- ✔ **Anotaciones intuitivas**: Utiliza anotaciones como `@Entity`, `@Dao` y `@Database` para definir esquemas y consultas de forma clara.
- ✔ **Validación en tiempo de compilación**: Verifica la corrección de las consultas SQL durante la compilación, minimizando errores en tiempo de ejecución.
- ✔ **Integración con LiveData**: Soporta **LiveData** y **Flow** para observar cambios en los datos de forma reactiva, actualizando la interfaz automáticamente.
- ✔ **Migraciones simplificadas**: Facilita la gestión de cambios en el esquema de la base de datos mediante migraciones controladas.
- ✔ **Pruebas optimizadas**: Proporciona herramientas para pruebas unitarias de la base de datos, asegurando fiabilidad en las operaciones de persistencia.

Uso de JSON Crack 4.12 para visualizar la estructura de datos

- ✔ **Visualización clara**: Permite representar visualmente la estructura jerárquica de los archivos **JSON**, facilitando la comprensión de relaciones entre entidades como ejercicios y músculos.
- ✔ **Apoyo al modelado de datos**: Ayudó a validar el diseño de las colecciones y subcolecciones en Firestore antes de implementar el script de carga.

4.2.8. Librerías y dependencias externas

El proyecto utiliza una combinación de librerías nativas de Android [2] y dependencias externas para implementar funcionalidades clave, como la interfaz de usuario, la persistencia de datos, la integración con servicios en la nube y las pruebas automatizadas. A continuación, se describen las principales librerías empleadas, agrupadas por su propósito en el desarrollo de la aplicación.

Interfaz de usuario

- ✔ **Jetpack Compose (1.7.8, BOM 2025.02.00)**: Biblioteca moderna [9] para construir interfaces declarativas y reactivas en Android, utilizada para diseñar layouts dinámicos con soporte para animaciones y navegación (**Navigation Compose 2.8.8**).
- ✔ **ConstraintLayout Compose (1.1.1)**: Permite crear diseños complejos en Compose con restricciones flexibles, optimizando la disposición de elementos.
- ✔ **Coil Compose (2.5.0)**: Biblioteca para la carga y visualización eficiente de imágenes en Jetpack Compose, utilizada a lo largo de todo el proyecto para optimizar el rendimiento de la aplicación. Permitió mostrar imágenes de ejercicios de forma rápida y fluida, facilitando que el usuario reconozca visualmente el ejercicio a realizar desde la lista de ejercicios.

Persistencia de datos

- ✔ **Room (2.6.1)**: Biblioteca de persistencia de Android Jetpack [23], utilizada para gestionar una base de datos local con SQLite, proporcionando abstracción, validación en tiempo de compilación y soporte para **LiveData** (**LiveData 2.8.7**). En este proyecto, Room se emplea para almacenar configuraciones de idioma (español e inglés), permitiendo que la aplicación detecte y utilice automáticamente el idioma predeterminado del dispositivo móvil, mejorando la accesibilidad y la experiencia del usuario.
- ✔ **Paging Compose (3.3.6)**: Permite cargar datos de forma paginada en la interfaz, optimizando el manejo de grandes conjuntos de datos. En este proyecto, se utiliza para gestionar la lista de ejercicios, cargando dinámicamente subconjuntos de datos desde la

nube (Firebase), lo que reduce el consumo de memoria y mejora la fluidez de la interfaz al mostrar listas extensas de rutinas de entrenamiento.

Integración con servicios en la nube

- ✔ **Firebase (BOM 33.10.0) [29]**: Conjunto de servicios de Google Cloud utilizado para autenticación (**Firebase Authentication**), almacenamiento de datos en tiempo real (**Firestore**), almacenamiento de archivos (**Firebase Storage**), y ejecución de lógica en la nube (**Cloud Functions**). Integrado con Dialogflow para personalizar interacciones conversacionales.
- ✔ **Google Services Plugin (4.4.2)**: Habilita la configuración de servicios de Google Cloud en la aplicación Android.
- ✔ **Crashlytics Plugin (3.0.3)**: Proporciona análisis de errores en tiempo real, facilitando la identificación y resolución de fallos que pueda tener el usuario. En este proyecto ha sido muy util para detectar fallos que no aparecían en la terminal, como errores de navegación en Jetpack Compose causado por una ruta no definida o errores de estado del usuario sobre usuario no autenticado.

Inyección de dependencias

- ✔ **Hilt (2.51)[25]**: Biblioteca de inyección de dependencias basada en Dagger, utilizada a lo largo de todo el proyecto para gestionar dependencias de forma modular y simplificar la arquitectura de la aplicación. Se integra de forma nativa con Android, facilitando la creación de componentes reutilizables y mejorando la escalabilidad y mantenibilidad del código. En este proyecto, Hilt ha sido fundamental para inyectar ViewModels, repositorios y otras dependencias clave en las distintas capas de la arquitectura MVVM, reduciendo significativamente la complejidad del código, eliminando la creación manual de objetos y agilizando el desarrollo y las pruebas.

Otras utilidades

- ✔ **KMP DateTime Picker (1.0.7)[26]**: Componente para seleccionar fechas y horas en Compose , mejorando la interacción del usuario. Para poder visualizar distintos entrenamientos dentro de un rango con un calendario y obtener datos como la fecha de nacimiento.
- ✔ **YCharts Compose (2.1.0)[27]**: Biblioteca para visualización de gráficos en Compose, utilizada para representar datos en la aplicación. Mediante distintos gráficos se muestran los datos más relevantes al usuario como las calorias quemadas a lo largo de la semana.
- ✔ **OkHttp (4.12.0)[28]**: Cliente HTTP para realizar solicitudes de red de forma eficiente, complementando las integraciones con servicios en la nube.

- ✔ **OpenCSV (5.6)**[**30**]: Librería utilizada para generar archivos CSV que permiten exportar los datos de entrenamientos y mediciones de forma sencilla y estructurada.
- ✔ **iText7 (7.1.14)**[**31**]: Librería empleada para crear documentos PDF con tablas y formato personalizado, facilitando la exportación y compartición de información desde la aplicación.

4.3. Fases de desarrollo

El trabajo se ha dividido en cuatro fases principales, cada una con tareas específicas que han guiado el avance del desarrollo. La planificación temporal ha sido estimada en función de la complejidad de cada fase, como se detalla a continuación:

Fase 1: Modelado / Requisitos Duración estimada: 45 horas

- ✔ **Tarea 1.1:** Definición de los requisitos funcionales y no funcionales de la aplicación, incluyendo aspectos esenciales como la personalización de rutinas, el seguimiento del progreso y la sincronización en tiempo real.
- ✔ **Tarea 1.2:** Diseño detallado del modelo de datos para Firestore, contemplando entidades como usuarios, entrenamientos, ejercicios y estadísticas, junto con sus relaciones.

Fase 2: Diseño / Implementación Duración estimada: 140 horas

- ✔ **Tarea 2.1:** Diseño de las principales pantallas mediante Figma, enfocándose en la usabilidad, simplicidad visual y fluidez de navegación.
- ✔ **Tarea 2.2:** Implementación de funcionalidades clave utilizando tecnologías de Firebase:
  - **2.2.1:** Autenticación de usuarios mediante Firebase Authentication.
  - **2.2.2:** Almacenamiento de datos en Firestore.
  - **2.2.3:** Sincronización de la información en tiempo real entre dispositivos.
- ✔ **Tarea 2.3:** Implementación y desarrollo del chatbot inteligente mediante Dialogflow ES:
  - **2.3.1:** Entrenamiento del chatbot mediante un chat, y solicitar rutinas y consejos sobre los ejercicios del entrenamiento.
  - **2.3.2:** Implementación del servidor para que el chatbot sea desplegado y que tenga acceso a los ejercicios de Firebase.
  - **2.3.3:** Incorporación del chatbot en la interfaz de usuario, accesible desde la pantalla principal.

Fase 3: Test / Despliegue Duración estimada: 70 horas

- ✔ **Tarea 3.1:** Pruebas funcionales y de interfaz en dispositivos móviles con diferentes resoluciones, tanto en dispositivos reales como en emuladores y versiones del sistema operativo Android.
- ✔ **Tarea 3.2:** Despliegue de una versión beta destinada a recopilar opiniones reales por parte de usuarios y detectar posibles errores o áreas de mejora.

Fase 4: Documentación / Presentación Duración estimada: 45 horas

- ✔ **Tarea 4.1:** Redacción de la memoria técnica, incluyendo los aspectos metodológicos, técnicos y de evaluación del trabajo realizado.
- ✔ **Tarea 4.2:** Elaboración de una presentación clara, visual y estructurada, orientada a la defensa del proyecto.

Cuadro 4.1: Metodología iterativa incremental

Fases (estructura orientativa)	**Duración Estimada** (horas)	**Tareas** (nombre y descripción, obligatorio al menos una por fase)
Modelado / Requisitos	45	Tarea 1.1: Definir los requisitos funcionales y no funcionales de la aplicación, identificando las características esenciales y las opcionales. Tarea 1.2: Diseñar el modelo de datos detallado para Firestore, incluyendo las relaciones necesarias para gestionar usuarios, entrenamientos y estadísticas.
Diseño / Implementación	140	Tarea 2.1: Diseñar las pantallas principales con Figma y crear un flujo de navegación intuitivo. Tarea 2.2: Implementar las funcionalidades clave utilizando Firestore. Tarea 2.2.1: Autenticación de usuarios con Firebase Authentication. Tarea 2.2.2: Base de datos en la nube con Firestore. Tarea 2.2.3: Sincronización en tiempo real.

Fases	Duración Estimada	Tareas
Test / Despliegue	70	Tarea 3.1: Probar la aplicación en dispositivos de distintas resoluciones y versiones de Android. Tarea 3.2: Lanzar una versión de prueba y analizar opiniones.
Documentación / Presentación	45	Tarea 4.1: Redactar la memoria del proyecto. Tarea 4.2: Crear una presentación clara y visual.

4.4. Requisitos del sistema

La aplicación, diseñada para el seguimiento y optimización de entrenamientos físicos, debe cumplir una serie de requisitos funcionales y no funcionales que garantizan su funcionalidad, usabilidad y escalabilidad. Los requisitos funcionales describen las capacidades principales de la aplicación, mientras que los no funcionales establecen las condiciones de rendimiento, seguridad y compatibilidad. A continuación, se detallan ambos tipos de requisitos, junto con ejemplos de priorización y consideraciones adicionales.

4.4.1. Requisitos funcionales

Los requisitos funcionales definen las funcionalidades clave que la aplicación debe ofrecer para satisfacer las necesidades de los usuarios, implementadas mediante tecnologías como Jetpack Compose, Room, Firebase y Dialogflow.

Gestión de usuarios

- ✔ Registro e inicio de sesión mediante correo electrónico, Google u otros proveedores, utilizando **Firebase Authentication**. Se implementó el sistema de autenticación aprovechando Firebase Authentication para permitir a los usuarios registrarse e iniciar sesión con email y contraseña.
- ✔ Gestión de perfiles con datos personales (edad, peso, altura, objetivos), almacenados en **Firestore**. Los perfiles de usuario se gestionan en Firestore para facilitar el almacenamiento y sincronización en tiempo real. Se diseñó un esquema de datos que permite actualizar y recuperar atributos personales fácilmente.

Registro y seguimiento de entrenamientos

- ✔ Creación de rutinas con atributos (nombre, músculos trabajados, objetivos, ejercicios), almacenados en **Firebase**.
- ✔ Registro de series, repeticiones y cargas, con cálculo automático del volumen total (series × repeticiones × peso). Se implementó una lógica en el ViewModel que calcula automáticamente el volumen de entrenamiento basado en los datos introducidos por el usuario.
- ✔ Visualización de los ejercicios realizados y los músculos trabajados mediante gráficos implementados con **YCharts Compose**.

Seguimiento de composición corporal

- ✔ Registro de peso y medidas corporales, almacenados mediante **Firebase**. Estos datos iniciales se recaban a través de un cuestionario inicial que incluye preguntas sobre la altura y el peso del usuario.
- ✔ Cálculo del Índice de Masa Corporal (IMC) basado en los datos ingresados.
- ✔ Gráficas interactivas de evolución del peso, calorias y el volumen, generadas con **YCharts Compose**.

Cálculo de calorías

- ✔ Estimación de calorías quemadas por entrenamiento, basada en volumen e intensidad, utilizando algoritmos personalizados.

Progresión y recomendaciones

- ✔ Gráficas de progresión de cargas y composición corporal, implementadas con **Jetpack Compose** y **YCharts Compose**.

Historial y exportación

- ✔ Acceso al historial de entrenamientos y mediciones, gestionado con **Firestore** para sincronización.
- ✔ Exportación de datos en formatos CSV o PDF:
  - Para la generación de archivos CSV se utiliza la librería `OpenCSV` [30], que permite exportar los datos de entrenamientos y mediciones de forma sencilla y estructurada.

- Para la creación y manipulación de documentos PDF se emplea la librería **iText7 (versión 7.1.14)** [31], reconocida por su robustez y versatilidad.

4.4.2. Requisitos no funcionales

Los requisitos no funcionales establecen las condiciones de rendimiento, usabilidad, seguridad y escalabilidad que la aplicación debe cumplir, soportadas por la arquitectura basada en Firebase y Android Jetpack.

Rendimiento

- ✔ Tiempo de carga máximo de 2 segundos para pantallas principales, optimizado con **Jetpack Compose** y **Paging Compose**.
- ✔ Soporte para hasta 10.000 usuarios concurrentes, escalado automáticamente por **Firestore**.

Usabilidad

- ✔ Interfaz intuitiva, con un máximo de 3 clics para registrar un ejercicio, diseñada con **Jetpack Compose**.
- ✔ Diseño responsive compatible con dispositivos Android.

Escalabilidad

- ✔ Arquitectura modular basada en **Hilt** para facilitar la incorporación de nuevas funcionalidades.
- ✔ Uso de **Firestore** y **Firebase Cloud Functions** para escalar según la demanda de usuarios.

Disponibilidad

- ✔ Sincronización en tiempo real entre dispositivos mediante **Firestore**.

Compatibilidad

- ✔ Funcionamiento en Android 8.0 (Oreo) o superior, soportado por **Android Studio** y **Jetpack**.

Accesibilidad

- ✔ Soporte para modo oscuro y ajuste de tamaño de fuente, implementados en **Jetpack Compose**.

4.4.3. Ejemplos de priorización

- ✔ **Funcional**: La aplicación debe permitir registrar 3 series de 10 repeticiones en press banca, almacenando los datos en **Firestore**.
- ✔ **No funcional**: Las gráficas de progreso deben cargarse en menos de 1.5 segundos, optimizadas con **Paging Compose** y **YCharts Compose**.

4.5. Software desarrollado

Para comprender adecuadamente el funcionamiento del software implementado, es necesario introducir previamente algunos conceptos relacionados con los datos que la aplicación calcula y representa.

4.6. Conceptos básicos

La aplicación ha sido desarrollada en **Kotlin**, el lenguaje oficial para Android, lo que garantiza compatibilidad y modernidad en el entorno de desarrollo. La interfaz gráfica ha sido implementada con **Jetpack Compose**, un framework declarativo que permite construir interfaces reactivas y flexibles, mejorando así la mantenibilidad del código frente a soluciones tradicionales basadas en XML.

En el apartado de backend, **Firebase** desempeña un papel fundamental. Se ha utilizado **Firebase Firestore** como base de datos y **Firebase Authentication** para la gestión segura de usuarios. Estas tecnologías permiten mantener sincronizados los datos entre el cliente y el servidor con baja latencia y gran escalabilidad.

Asimismo, la aplicación incorpora un chatbot desarrollado con **Dialogflow ES**, el cual se comunica con la aplicación a través de un servidor construido con **Node.js** y **Express.js**. Este asistente virtual permite a los usuarios realizar consultas relacionadas con las rutinas o ejercicios, enriqueciendo la interacción y ofreciendo soporte personalizado dentro de la app.

4.6.1. Definiciones clave

Los conceptos más habituales en una sesión de entrenamiento son:

- ✔ **Ejercicio**: Actividad física específica que implica el uso de determinados grupos musculares. Por ejemplo, press de banca, dominadas o sentadillas.
- ✔ **Serie**: Conjunto de repeticiones de un mismo ejercicio realizadas de forma continua y sin pausas prolongadas.
- ✔ **Repetición**: Ejecución completa de un movimiento dentro de una serie. Por ejemplo, una bajada y subida en una sentadilla.
- ✔ **Sesión de entrenamiento**: Conjunto estructurado de ejercicios realizados en un mismo día, desde el inicio hasta el final del entrenamiento.
- ✔ **Peso utilizado**: Carga en kilogramos empleada por el usuario en un ejercicio. En ejercicios con el propio peso corporal, este se considera como referencia.
- ✔ **Volumen total**: Cantidad total de trabajo realizado durante una sesión, calculado como el producto del número de series, repeticiones y peso en cada ejercicio.
- ✔ **Esfuerzo**: Nivel de exigencia percibido por el usuario durante la sesión, influido por la cantidad de repeticiones, peso y número de ejercicios realizados.
- ✔ **Intensidad del entrenamiento**: Estimación del grado de exigencia física de una sesión, teniendo en cuenta el volumen total, el peso corporal del usuario y la duración de la actividad.
- ✔ **Calorías estimadas**: Energía consumida durante la sesión de entrenamiento, calculada a partir del tipo de ejercicio, volumen y carga aplicada.
- ✔ **Mancuerna:** Instrumento de entrenamiento formado por una barra corta con peso en ambos extremos, diseñado para ser sostenido con una o ambas manos. Se utiliza ampliamente en ejercicios de fuerza, permitiendo un trabajo unilateral o bilateral de distintos grupos musculares. Las mancuernas pueden ser fijas o ajustables, y se adaptan a una gran variedad de movimientos.

Ejemplo práctico

Supongamos que una persona quiere realizar un ejercicio de curl de bíceps con mancuernas. Como se muestra en la Figura 4.4, el curl de bíceps es un ejercicio común para trabajar este músculo.

Desglose del ejercicio según los conceptos:

- ✔ **Ejercicio**: El curl de bíceps es un ejercicio de aislamiento para el bíceps braquial.
- ✔ **Serie**: Realizar 10 repeticiones consecutivas con una mancuerna de 8 kg constituye una serie.
- ✔ **Repetición**: Cada flexión del codo que eleva la mancuerna hasta los hombros, seguida de un descenso controlado, es una repetición.

- ✔ **Sesión de entrenamiento**: El curl de bíceps puede integrarse en una sesión de tren superior junto con ejercicios como press de banca.
- ✔ **Peso utilizado**: Una persona podría usar mancuernas de 8 kg por brazo.
- ✔ **Volumen total**: Para 3 series de 10 repeticiones con 8 kg por brazo, el volumen total es $3 \times 10 \times 8 = 240\,\text{kg}$ por brazo, o 480 kg considerando ambos brazos.
- ✔ **Esfuerzo**: Un usuario principiante podría percibir un esfuerzo moderado (escala subjetiva 5/10) con este volumen y peso.
- ✔ **Intensidad del entrenamiento**: La intensidad depende del volumen (480 kg), el peso corporal del usuario (p. ej., 70 kg) y la duración de la sesión (p. ej., 30 minutos).
- ✔ **Calorías estimadas**: Para una sesión de 30 minutos que incluye curl de bíceps, se estiman unas 150-200 kcal, según la intensidad y el peso corporal.

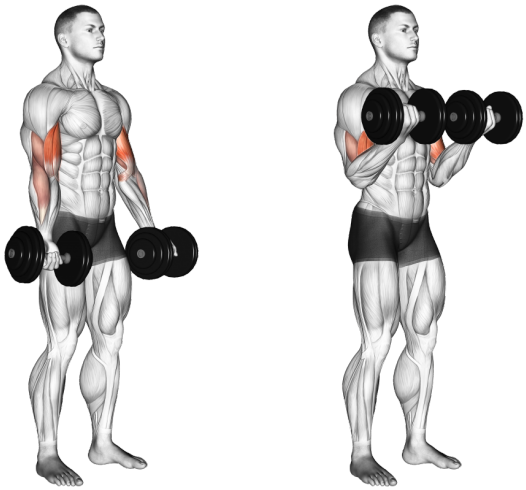

Ilustración 4.4: Ejemplo de ejercicio de curl de bíceps con mancuernas.

En esta aplicación, el modelo de resumen de entrenamiento procesa los datos registrados por el usuario (series, repeticiones y peso utilizado) para generar un modelo de datos con métricas importantes: volumen total, calorías estimadas e intensidad del entrenamiento.

Volumen total

El volumen total representa la cantidad acumulada de trabajo mecánico durante una sesión de entrenamiento, expresado en kilogramos. Se calcula como:

$$\text{Volumen} \triangleq \sum_{i=1}^{n} (\text{series}_i \times \text{repeticiones}_i \times \text{peso}_i) \tag{4.1}$$

Para ejercicios sin peso, se usa el 60 % del peso corporal del usuario por repetición:

$$\text{Volumen (sin peso)} \triangleq \text{repeticiones} \times (\text{peso del usuario} \times 0{,}6) \tag{4.2}$$

Por ejemplo, 2 series de 10 repeticiones con 20 kg generan un volumen de $2 \times 10 \times 20 = 400\,\text{kg}$. Esta métrica refleja el esfuerzo físico y el progreso en fuerza o resistencia.

Intensidad del entrenamiento

La intensidad mide el esfuerzo relativo de una sesión de entrenamiento, proporcionando una métrica que refleja la dificultad percibida por el usuario. Habitualmente, la intensidad se calcula como un porcentaje del peso máximo que una persona puede levantar en una sola repetición, conocido como RM. El RM es el peso más elevado que un usuario puede levantar correctamente en un ejercicio específico, realizando una sola repetición con técnica adecuada, antes de alcanzar el fallo muscular.

Por ejemplo, si un usuario puede realizar un press de banca con 80 kg una vez, pero no con 85 kg, su RM para ese ejercicio es de 80 kg.

En el sistema desarrollado, se adopta un enfoque alternativo para calcular la intensidad, que no depende exclusivamente del RM, sino que combina múltiples factores: el volumen total, el número de series y repeticiones completadas, la duración de la sesión y el peso corporal del usuario. Este método genera una métrica normalizada en un rango abierto y orientativo.

La intensidad se calcula como una combinación ponderada de los tres componentes mencionados, definida por la siguiente ecuación:

$$\text{Intensidad} \triangleq 0{,}7 \times \text{volumenIntensidad} + 0{,}2 \times \text{esfuerzoIntensidad} + 0{,}1 \times \text{tiempoIntensidad} \tag{4.3}$$

Donde:

- ✔ `volumenIntensidad` $\triangleq \dfrac{\text{volumen total}}{\text{peso corporal}}$
- ✔ `esfuerzoIntensidad` $\triangleq (\text{series completadas} \times 2) + (\text{repeticiones totales} \times 0{,}5)$
- ✔ `tiempoIntensidad` $\triangleq \dfrac{\text{volumen total}}{\text{duración (min)}}$

Esta estimación debe entenderse como una métrica orientativa. Los datos utilizados representan una interpretación aproximada basada en algunos estudios relacionados [32, 33] con el entrenamiento de fuerza y en la experiencia práctica acumulada.

- ✔ En primer lugar, la **intensidad por volumen** se calcula dividiendo el volumen total (peso movido en la sesión) entre el peso corporal del usuario. Este enfoque permite contextualizar el esfuerzo en relación con la capacidad física del individuo. Además, el volumen total integra tres factores clave del entrenamiento: número de series, repeticiones y peso utilizado.

- ✔ En segundo lugar, la **intensidad por esfuerzo** busca capturar la implicación del usuario durante la sesión, ponderando más las series realizadas (por su carga estructurada) y complementándolas con el número total de repeticiones, que también representa esfuerzo acumulado.
- ✔ Por último, la **intensidad por duración** mide la eficiencia del entrenamiento: una mayor cantidad de trabajo realizada en menos tiempo sugiere una mayor intensidad, al reflejar un ritmo más exigente durante la sesión.

La fórmula de intensidad combina los tres componentes (volumen, esfuerzo y duración) mediante una ponderación que refleja su importancia relativa en la percepción del esfuerzo. Se asignaron los siguientes pesos iniciales, basados en un análisis práctico y teórico:

- ✔ **70 % para la intensidad por volumen**: El volumen, calculado como series × repeticiones × peso, es el indicador más directo del esfuerzo físico, ya que refleja el trabajo mecánico total realizado. En la literatura sobre entrenamiento de fuerza, el volumen se considera un factor clave para el estrés mecánico y metabólico, que influyen directamente en la percepción de intensidad [34]. Por ello, se le asigna un peso mayoritario del 70 %.
- ✔ **20 % para la intensidad por esfuerzo**: Este componente, basado en series y repeticiones, captura la fatiga neuromuscular y el esfuerzo acumulado. Aunque es relevante, su contribución es secundaria frente al volumen, ya que no considera directamente el peso levantado. Se le asigna un 20 % para reflejar su importancia como complemento al volumen.
- ✔ **10 % para la intensidad por duración**: La duración mide la eficiencia del entrenamiento (volumen por unidad de tiempo). Aunque un mayor ritmo puede aumentar la intensidad percibida, su impacto es menor que el del volumen o el esfuerzo, ya que no refleja directamente la carga. Se le asigna un 10 % para incluir este factor contextual sin sobreponderarlo.

Ejemplo práctico de cálculo

Supóngase que un usuario con un peso corporal de 80 kg ha realizado una sesión de entrenamiento con las siguientes características:

- ✔ Volumen total: 20 000 kg
- ✔ Series completadas: 20
- ✔ Repeticiones totales: 200
- ✔ Duración de la sesión: 100 min

Se procede a calcular cada componente de la fórmula:

$$\begin{aligned}
\texttt{volumenIntensidad} &= \frac{20000}{80} = 250 \\
\texttt{esfuerzoIntensidad} &= 20 \times 2 + 200 \times 0{,}5 = 40 + 100 = 140 \\
\texttt{tiempoIntensidad} &= \frac{20000}{100} = 200
\end{aligned}$$

Sustituyendo en la fórmula general:

$$\begin{aligned}
\text{Intensidad} &\triangleq 0{,}7 \times 250 + 0{,}2 \times 140 + 0{,}1 \times 200 \\
&= 175 + 28 + 20 \\
&= 223
\end{aligned}$$

La intensidad estimada de la sesión es **223**. Se trata de un valor adimensional que permite comparar distintas sesiones del mismo usuario u otros con características similares. Este valor de 223 se considera elevado para una métrica práctica que represente una sesión promedio. Para reflejar un valor promedio más representativo, se determinó que la intensidad debería ser aproximadamente 65, un valor que, basado en experiencias prácticas, corresponde a una sesión de entrenamiento típica. Partiendo de una ponderación deseada de 70 % para el volumen, 20 % para el esfuerzo y 10 % para la duración, se calcularon las contribuciones objetivo:

- ✔ Contribución de volumen: $70\,\% \times 65 = 45,5$
- ✔ Contribución de esfuerzo: $20\,\% \times 65 = 13,0$
- ✔ Contribución de duración: $10\,\% \times 65 = 6,5$

Los pesos se obtienen dividiendo estas contribuciones entre los valores de los componentes:

- ✔ Peso para volumen: $\frac{45,5}{250} \approx 0,182$
- ✔ Peso para esfuerzo: $\frac{13,0}{140} \approx 0,09286$
- ✔ Peso para duración: $\frac{6,5}{200} = 0,0325$

Redondeando para simplificar, los pesos ajustados son aproximadamente 0,18, 0,09 y 0,03. Sin embargo, para mayor precisión, se utilizan los valores exactos.

La nueva fórmula de intensidad es:

$$\text{Intensidad} \triangleq 0,182 \times \text{volumenIntensidad} + 0,09286 \times \text{esfuerzoIntensidad} + 0,0325 \times \text{tiempoIntensidad} \tag{4.4}$$

Aplicando esta fórmula con los componentes (volumenIntensidad = 250, esfuerzoIntensidad = 140, tiempoIntensidad = 200):

$$\begin{aligned}
\text{Intensidad ajustada} &= (0,182 \times 250) + (0,09286 \times 140) + (0,0325 \times 200) \\
&= 45,5 + 13,0004 + 6,5 \\
&= 65
\end{aligned}$$

Con los pesos redondeados (0,18; 0,09; 0,03):

$$\begin{aligned}\text{Intensidad ajustada} &= (0,18 \times 250) + (0,09 \times 140) + (0,03 \times 200) \\ &= 45 + 12,6 + 6 \\ &= 63,6 \approx 65\end{aligned}$$

En conclusión, el ajuste de los pesos, basado en una ponderación deseada de 70 %-20 %-10 %, permite normalizar la intensidad a 65, un valor representativo de una sesión de entrenamiento promedio. El redondeo a 0,18, 0,09 y 0,03 proporciona una aproximación práctica de 63,6, lo suficientemente cercana para fines de comparación, facilitando el seguimiento del progreso por parte de entrenadores y usuarios.

Calorías estimadas

Las calorías quemadas representan el gasto energético estimado durante una sesión de entrenamiento, proporcionando al usuario una métrica sobre la energía consumida. Las calorías son como el "combustible" que el cuerpo utiliza al hacer ejercicio. Por ejemplo, levantar pesas o hacer flexiones quema energía, y se calcula cuántas basándose en las repeticiones realizadas y si se usó peso. Si se levanta mucho peso o se hacen muchas repeticiones, se queman más calorías. Para calcular estas calorías según sea un tipo de ejercicio con peso o no, lo calculamos de la siguiente manera:

- ✔ Para **ejercicios con peso** (por ejemplo, press de banca):

$$\text{Calorías por serie} \triangleq (\text{peso} \times \text{repeticiones}) \times 0,025 \tag{4.5}$$

- ✔ Para **ejercicios sin peso** (por ejemplo, flexiones):

$$\text{Calorías por serie} \triangleq \text{repeticiones} \times 0,4 \tag{4.6}$$

El total de calorías es la suma de las calorías de todas las series de todos los ejercicios en la sesión.

4.6.2. Cálculo automático de datos del entrenamiento

El método `calculateAndSetWorkoutData` es una función asíncrona en Kotlin, implementada en el `ViewModel` asociado a la pantalla de resumen de entrenamiento. Su propósito es procesar los datos de una sesión de entrenamiento para crear una instancia de `WorkoutModel`, que incluye métricas como el gasto calórico, el volumen total, la intensidad, el tipo de entrenamiento, los músculos involucrados y la duración. Este modelo se almacena en Firestore para que el usuario pueda consultar un resumen detallado de su sesión.

Los datos para calcular estas métricas provienen de la interacción del usuario con la aplicación durante un entrenamiento. El usuario puede elegir entre rutinas sugeridas por la

aplicación, generadas por el chatbot o creadas por él mismo. Estos datos se registran en la lista `_exercisesWithSeries`, que incluye los ejercicios, sus series, repeticiones y peso utilizado, ingresados manualmente a través de la interfaz. El peso corporal del usuario se obtiene del repositorio `userRepository`, y la duración de la sesión se registra automáticamente en `_workoutDuration` (en milisegundos).

A continuación, se presenta el fragmento de código correspondiente:

```
private suspend fun calculateAndSetWorkoutData(): WorkoutModel {
    var totalCalories = 0.0
    var totalVolume = 0.0
    val userWeight = userRepository.getCurrentUserWeight() ?: 70.0

    _exercisesWithSeries.value.forEach { exercise ->
        exercise.series.forEach { serie ->
            if (exercise.requiresWeight) {
                val weight = serie.weight?.toDoubleOrNull() ?: 0.0
                val reps = serie.reps?.toIntOrNull() ?: 0
                totalCalories += (weight * reps) * 0.025
                totalVolume += weight * reps
            } else {
                val reps = serie.reps?.toIntOrNull() ?: 0
                totalCalories += reps * 0.4
                totalVolume += (userWeight * 0.6) * reps
            }
        }
    }

    val primaryMuscleIds = _exercisesWithSeries.value
        .map { it.primaryMuscleRef?.id ?: "No muscle ref"}
        .toSet()
        .toList()

    val secondaryMuscleIds = _exercisesWithSeries.value
        .flatMap { it.secondaryMusclesRef.map { ref -> ref?.id ?: "No muscle ref" } }
        .toSet()
        .toList()

    val workoutType = determineWorkoutType(_exercisesWithSeries.value,
        totalVolume, _workoutDuration.value / 1000)
    val intensity = calculateIntensity(_exercisesWithSeries.value,
        totalVolume, _workoutDuration.value / 1000, userWeight)

    return WorkoutModel(
        id = "",
        userId = userRepository.getCurrentUser().uid,
        date = Timestamp.now(),
        exercises = _exercisesWithSeries.value,
        calories = totalCalories.toInt(),
        volume = totalVolume.toInt(),
        intensity = intensity,
        workoutType = workoutType,
        primaryMuscleIds = primaryMuscleIds,
        secondaryMuscleIds = secondaryMuscleIds,
        duration = _workoutDuration.value / 1000
    )
}
```

Descripción de la lógica del código

El método `calculateAndSetWorkoutData` procesa los datos de entrenamiento en varias etapas:

1. **Inicialización de variables**: Se inicializan `totalCalories` y `totalVolume` para acumular el gasto calórico y el volumen total, respectivamente. El peso del usuario (`userWeight`) se obtiene desde el repositorio.
2. **Cálculo de calorías y volumen**: El método itera sobre la lista `_exercisesWithSeries`, que contiene los ejercicios realizados y sus series asociadas. Para cada serie, se distingue entre ejercicios con peso (por ejemplo, levantamiento con mancuernas) y sin peso.
3. **Identificación de músculos involucrados**: Los identificadores de músculos primarios (`primaryMuscleIds`) se extraen de cada ejercicio, eliminando duplicados mediante un conjunto (`Set`). Los músculos secundarios (`secondaryMuscleIds`) se obtienen de forma similar, procesando las listas de referencias secundarias sobre cada ejercicio.
4. **Determinación del tipo de entrenamiento e intensidad**: La función `determineWorkoutType` clasifica el entrenamiento (por ejemplo, fuerza, hipertrofia, resistencia) basándose en los ejercicios, el volumen total y la duración (en segundos). La función `calculateIntensity` evalúa la intensidad considerando los mismos parámetros más el peso del usuario.
5. **Creación del modelo de datos**: Se genera una instancia de `WorkoutModel` con los datos calculados, incluyendo:
   - ✔ `id`: Identificador único (inicialmente vacío, asignado por Firestore).
   - ✔ `userId`: Identificador del usuario.
   - ✔ `date`: Fecha actual del entrenamiento.
   - ✔ `exercises`: Lista de ejercicios con sus series.
   - ✔ `calories`: Gasto calórico total (en kcal, convertido a entero).
   - ✔ `volume`: Volumen total (en kg, convertido a entero).
   - ✔ `intensity`: Intensidad calculada.
   - ✔ `workoutType`: Tipo de entrenamiento determinado.
   - ✔ `primaryMuscleIds` y `secondaryMuscleIds`: Listas de músculos involucrados.
   - ✔ `duration`: Duración del entrenamiento en segundos.

Justificación de las constantes

Las constantes utilizadas en los cálculos de calorías y volumen son aproximaciones empíricas basadas en principios de fisiología del ejercicio:

- ✔ El factor 0,025 en la ecuación (4.5) representa una estimación del gasto energético por unidad de trabajo mecánico en ejercicios con peso.
- ✔ El factor 0,4 en la ecuación (4.6) refleja el mayor componente metabólico y aeróbico de ejercicios sin carga externa.
- ✔ El factor 0,6 en la ecuación (4.2) estima el esfuerzo relativo al mover el peso corporal en ejercicios sin peso.

Propósito y uso

El método `calculateAndSetWorkoutData` genera un modelo de datos completo y estructurado que representa una sesión de entrenamiento. Este modelo se envía a Firestore para su almacenamiento, permitiendo su posterior consulta y análisis en la aplicación. La lógica implementada asegura que las métricas calculadas sean consistentes con los datos ingresados por el usuario, proporcionando un resumen útil y preciso del rendimiento del entrenamiento.

El método “calculateAndSetWorkoutData“ es una función asíncrona que procesa los datos obtenidos durante una sesión de entrenamiento para generar una instancia de ‘WorkoutModel‘, la cual se almacena posteriormente en Firestore. Este método calcula métricas clave, como el gasto calórico, el volumen total y la intensidad del entrenamiento, considerando las características de los ejercicios realizados (con o sin peso).

4.6.3. Implementación del Chatbot

Para complementar la funcionalidad del chatbot desarrollado con Dialogflow, se implementó un servidor intermedio utilizando Node.js y el framework **Express.js**. Este servidor actúa como middleware entre Dialogflow y la base de datos en Firebase, permitiendo generar respuestas dinámicas personalizadas para los usuarios.

Una de las funcionalidades principales del servidor consiste en acceder a la colección `exercises` de Firestore, donde se almacenan los ejercicios con sus respectivos datos, como el grupo muscular, el valor MET, instrucciones, errores comunes, entre otros.

A continuación, se muestra un fragmento de la lógica implementada para generar una rutina:

```
async function generateRoutine(type, muscles, prefix = "", language = "es") {
  const exercises = await getExercises(type, muscles);
  if (exercises.length === 0) {
    return language === "es"
      ? `Ups! No hay ejercicios para ${type || "esta␣categoria"}.`
      : `Oops! No exercises found for ${type || "this␣category"}.`;
  }
  const selected = shuffle(exercises).slice(0, 3);
  const routine = selected
    .map((ex, index) =>
      `${index + 1}. ${exerciseTranslations[ex.name]?.[language] || ex.name}`)
    .join("\n");
```

```
  return language === "es"
    ? `${prefix} ${routine} Quieres saber como hacer alguno de estos
    ejercicios o necesitas mas?`
    : `${prefix} ${routine} Want to know how to do any of these exercises
    or need more?`;
}
```

También se incluyó una ruta de prueba para validar la correcta conexión con Firestore y consultar los datos:

```
app.get("/test-firestore", async (req, res) => {
  try {
    const snapshot = await db.collection("exercises").get();
    const exercises = snapshot.docs.map((doc) => ({
      name: doc.data().name,
      type: doc.data().type,
      primaryMuscle: doc.data().primaryMuscle?.path || "Sin musculo primario",
    }));
    res.json({
      count: exercises.length,
      exercises: exercises,
    });
  } catch (error) {
    res.status(500).send("Error Firestore: " + error.message);
  }
});
```

4.6.4. Dialogflow

Diseño de `Intents` y `Entities`

El funcionamiento del bot se estructura a través de **intents**, que representan las posibles intenciones del usuario, y **entities**, que permiten extraer parámetros clave de sus mensajes.

- ✔ **Intents creados** 4.5: *intrucciones de ejercicios* 4.6, *consejos*, y *saludo*, entre otros.
- ✔ **Entities personalizadas** 4.7: Se han definido entidades como `@muscleGroup`, `@exerciseType`, y `@goal` para identificar los elementos relevantes en la conversación.

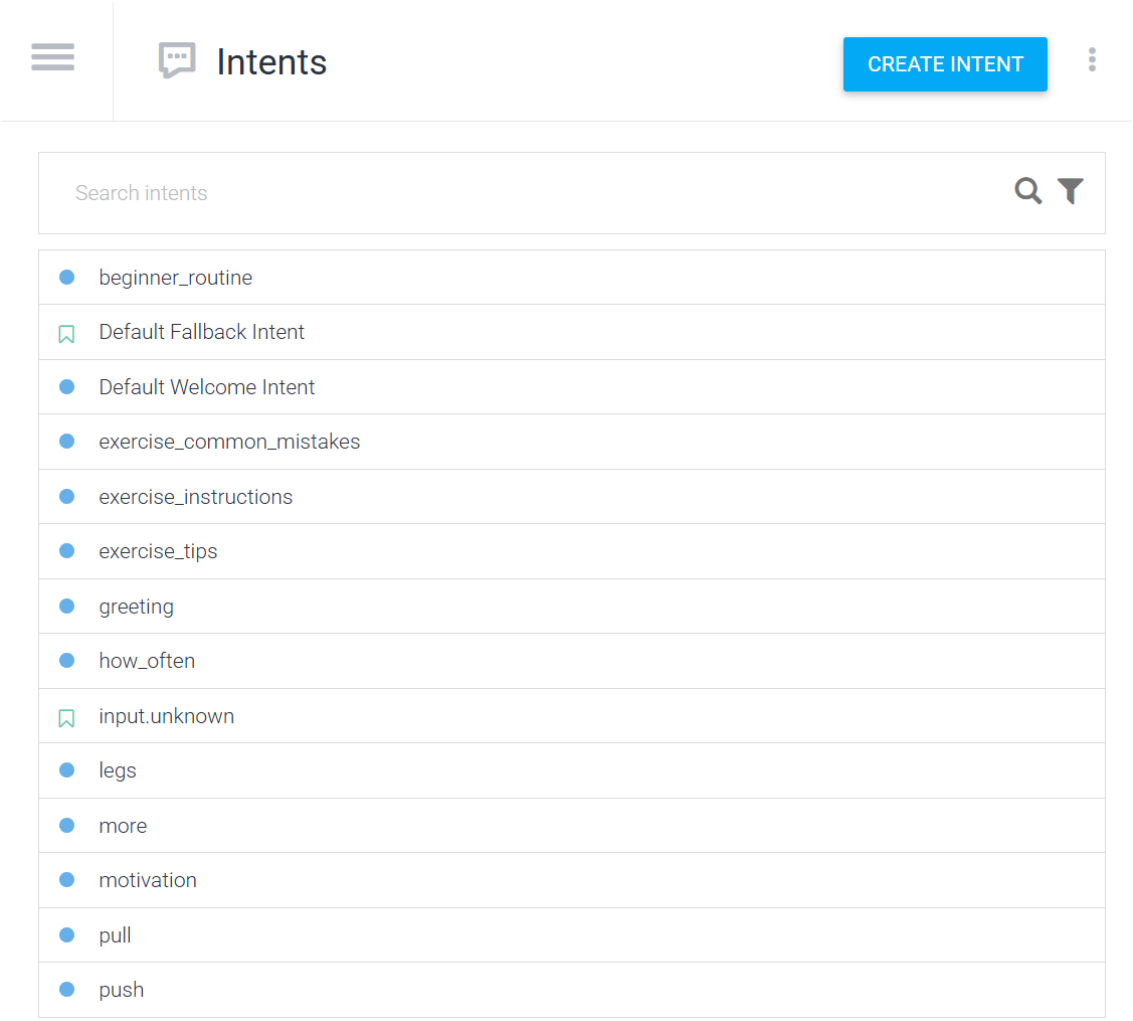

Ilustración 4.5: Intents creados, es decir, posibles intenciones de usuario a la hora de comunicarse con el chatbot.

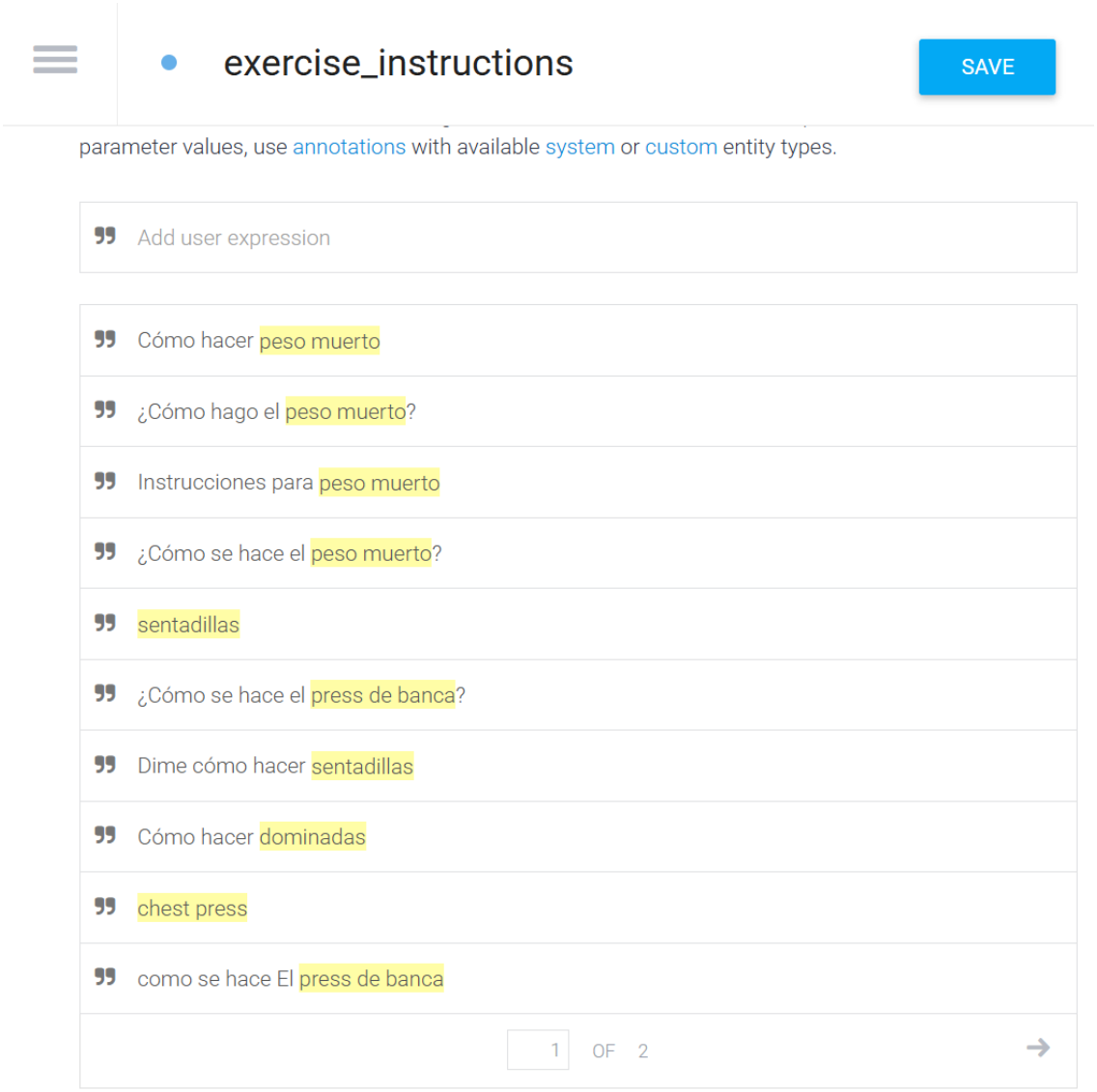

Ilustración 4.6: Ejemplo del intent de ejercicios, con sus frases de entrenamiento.

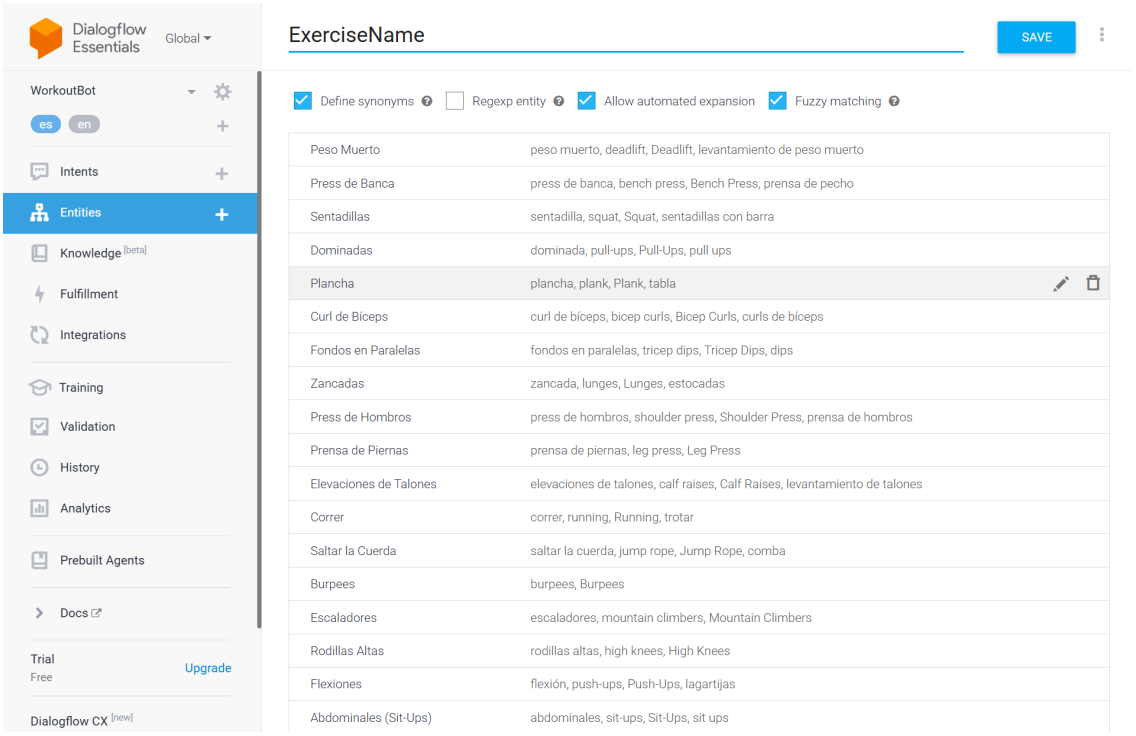

Ilustración 4.7: Ejemplo de entidad, frases o palabras claves a detectar sobre el nombre de un ejercicio.

Entrenamiento del modelo conversacional

Para cada intent 4.7, se ha proporcionado un conjunto variado de frases de entrenamiento. Esto permite que el agente pueda generalizar y comprender diferentes formas de expresar una misma intención. A continuación, se muestra un ejemplo de entrenamiento para el intent `Crear rutina`:

- ✔ *Quiero una rutina para pecho.*
- ✔ *Dame ejercicios de espalda.*
- ✔ *¿Qué puedo hacer para entrenar pierna?*

Conexión con el servidor y respuesta dinámica

Dialogflow se conecta con un servidor Node.js a través de un webhook, que procesa la intención del usuario, accede a Firestore para obtener los ejercicios y devuelve una respuesta personalizada.

Visualización del flujo conversacional

El flujo conversacional trata sobre la interacción entre el usuario y el chatbot. A continuación, se muestra un ejemplo de flujo y también, más adelante, hay una figura de ejemplo 5.6. Estos son los pasos del flujo:

1. El usuario escribe un mensaje con una petición (por ejemplo, "Recomiéndame ejercicios para pierna").
2. Dialogflow identifica el **intent** correspondiente mediante procesamiento de lenguaje natural (NLP).
3. Se extraen **entities** como el grupo muscular o tipo de ejercicio.
4. Se invoca el *fulfillment*, que es la lógica que se ejecuta cuando dialogflow detecta un intent, es decir, cuando interpreta lo que el usuario dice (intent + entidades).
5. El backend devuelve una respuesta con una rutina generada automáticamente.
6. El usuario puede solicitar más información o repetir el proceso.

4.7. Modelo de datos

El modelo de datos está diseñado para poder tener una escalabilidad sencilla y de fácil acceso a los datos. Durante esta sección, mostraremos algunas de las colecciones utilizadas en este proyecto.

4.7.1. Estructura general de la base de datos

Colección Users

La colección **users** 4.8 representa a los usuarios registrados en la aplicación. Cada documento dentro de la colección `users` almacena información personal, datos de la cuenta y de configuración que se obtienen al completar el cuestionario inicial.

- ✔ **activityLevel**: Nivel de actividad física.
- ✔ **birthdate**: Fecha de nacimiento del usuario.
- ✔ **createdAt**: Fecha y hora de creación del usuario.
- ✔ **email**: Correo electrónico del usuario.
- ✔ **goal**: Objetivo principal del usuario.
- ✔ **goalCalories**: Calorías objetivo diarias.

- ✔ **hasCompletedQuestionnaire**: Indica si el usuario ha completado el cuestionario inicial.
- ✔ **height**: Altura del usuario en centímetros.
- ✔ **name**: Nombre del usuario.
- ✔ **profileImageUrl**: URL de la imagen de perfil.
- ✔ **termsAccepted**: Indica si el usuario aceptó los términos y condiciones.
- ✔ **weight**: Peso del usuario en kilogramos.

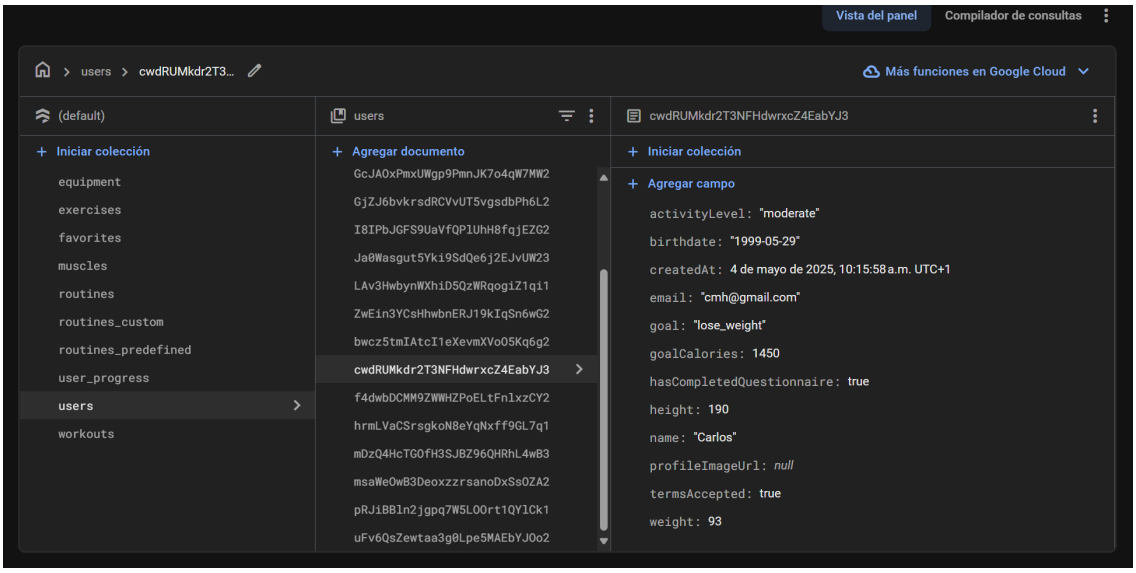

Ilustración 4.8: Ejemplo de documento en la colección **users**.

Colección Workouts

La colección **workouts** 4.9 representa una sesión de entrenamiento realizada por el usuario. Cada documento en la colección `workouts` contiene información detallada sobre el tipo de entrenamiento, los ejercicios realizados, la duración, el volumen y los músculos.

- ✔ **calories**: Estimación de calorías quemadas durante la sesión.
- ✔ **date**: Fecha y hora en la que se realizó el entrenamiento.
- ✔ **duration**: Duración total del entrenamiento en minutos.
- ✔ **exercises**: Lista de ejercicios realizados.
- ✔ **intensity**: Valor numérico que representa la intensidad del entrenamiento.
- ✔ **primaryMuscleIds**: Músculos principales trabajados en la sesión.
- ✔ **secondaryMuscleIds**: Músculos secundarios implicados.

- ✔ **userId**: Identificador del usuario que realizó el entrenamiento.
- ✔ **volume**: Volumen total del entrenamiento (por ejemplo, suma de repeticiones por peso).
- ✔ **workoutType**: Tipo de entrenamiento realizado.

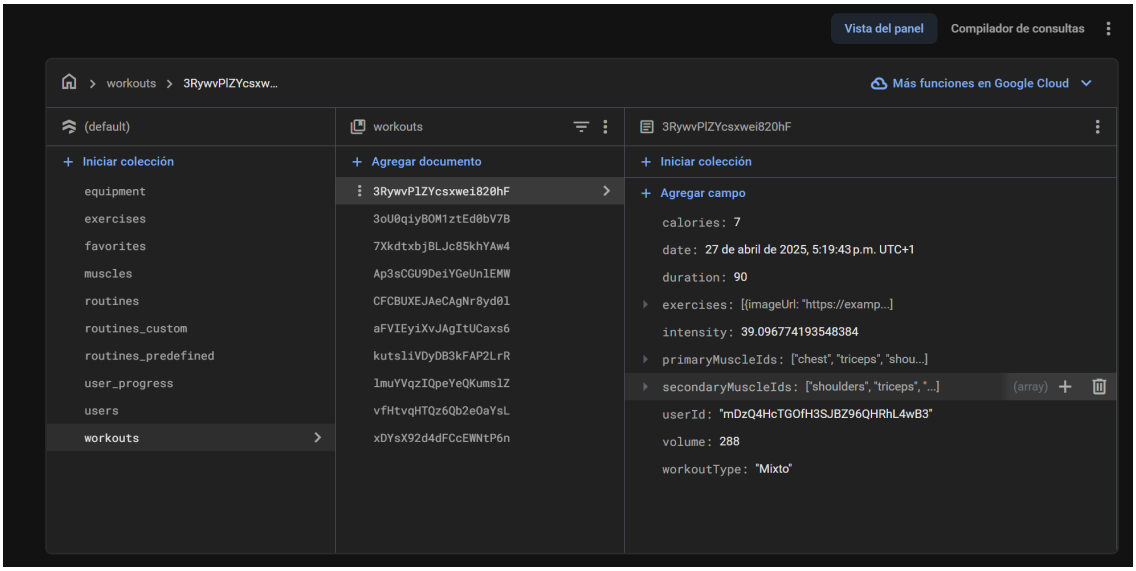

Ilustración 4.9: Ejemplo de documento en la colección **workouts**.

Colección Exercises

La colección **exercises** 4.10 contiene información detallada sobre cada ejercicio disponible en la aplicación. Cada documento representa un ejercicio específico y define atributos relevantes como los músculos implicados, instrucciones de ejecución y errores comunes.

- ✔ **commonMistakes**: Lista de errores comunes al realizar el ejercicio.
- ✔ **imageUrl**: URL de la imagen representativa del ejercicio.
- ✔ **instructions**: Instrucciones para realizar correctamente el ejercicio.
- ✔ **met**: Valor MET (Metabolic Equivalent of Task) asociado al ejercicio.
- ✔ **name**: Nombre del ejercicio.
- ✔ **primaryMuscle**: Músculo principal trabajado.
- ✔ **requiresWeight**: Indica si el ejercicio requiere peso adicional.
- ✔ **secondaryMuscle**: Músculos secundarios implicados.
- ✔ **tips**: Consejos para mejorar la ejecución del ejercicio.
- ✔ **type**: Tipo de ejercicio.

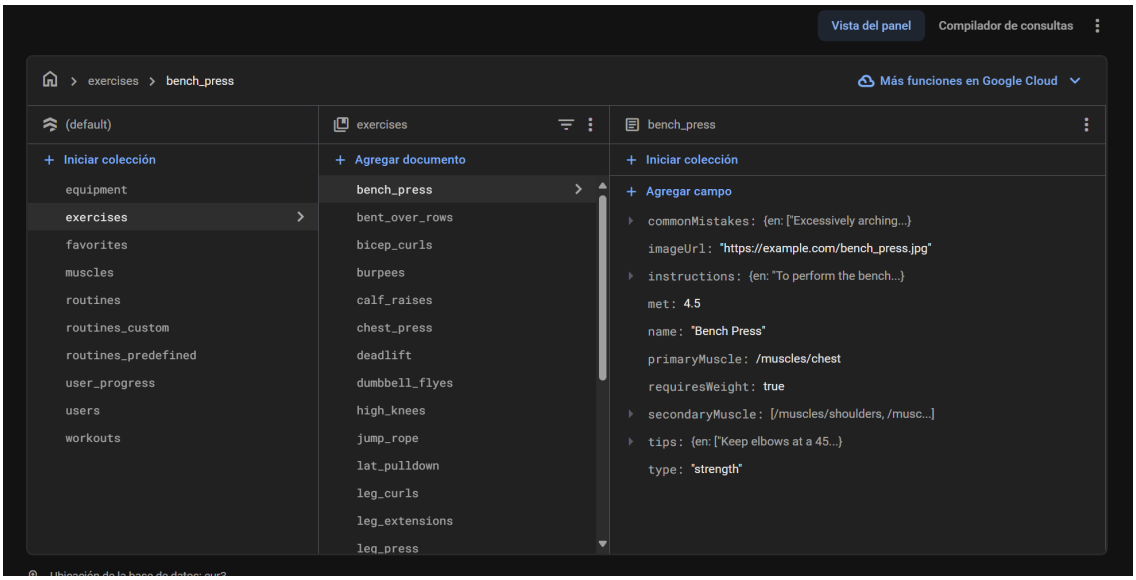

Ilustración 4.10: Ejemplo de documento en la colección **exercises**.

Colección Muscles

La colección **muscles** 4.11 contiene información detallada sobre cada músculo. Cada documento representa un músculo específico y define atributos relevantes como el nombre y los músculos secundarios.

- ✔ **imageUrl**: URL de la imagen representativa del músculo.
- ✔ **name**: Nombre del ejercicio.
- ✔ **primaryMuscle**: Músculo principal.
- ✔ **secondaryMuscle**: Músculos secundarios implicados.

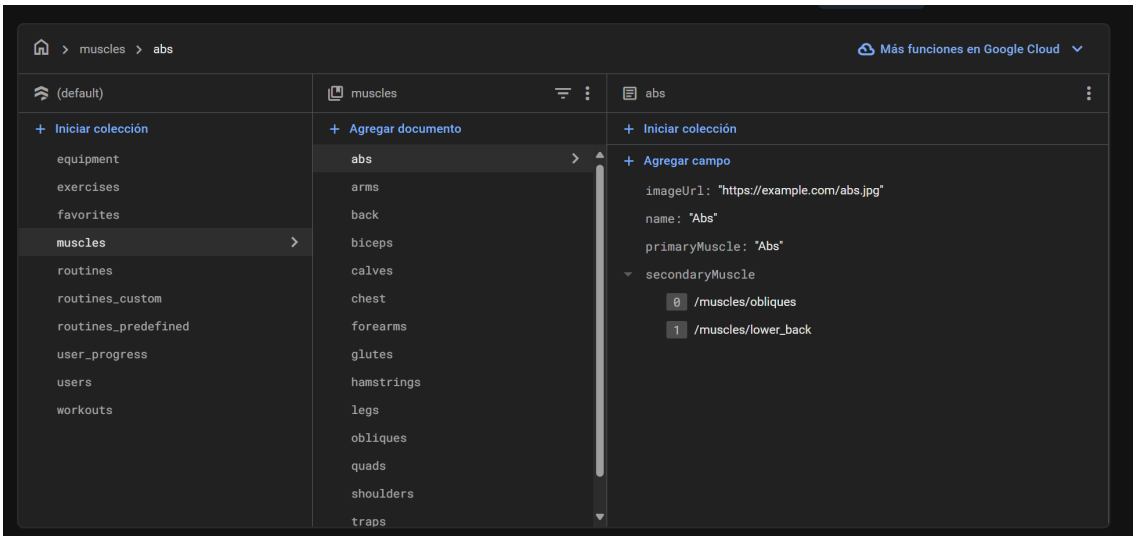

Ilustración 4.11: Ejemplo de documento en la colección **muscles**.

4.7.2. Carga de datos a Firestore

Durante el desarrollo de la app se implementó un script en **Node.js** con el objetivo de facilitar la carga inicial de datos en la base de datos con Firestore. Este script permite poner datos automáticamente en la colección elegida y a partir de un archivo en formato **JSON** 4.12 se sube esta información.

1. Inicializa la aplicación de Firebase utilizando las credenciales del servicio (service-account-key.json).
2. Carga el archivo `exercises.json`, que contiene los datos a insertar.
3. Para cada ejercicio, crea un nuevo documento en la colección `exercises`, vinculando el campo **primaryMuscle** como referencia a la colección **muscles**.
4. Los músculos secundarios (**secondaryMuscle**) se transforman en un arreglo de referencias.
5. Se utiliza un **batch** para realizar todas las inserciones de forma conjunta y eficiente.

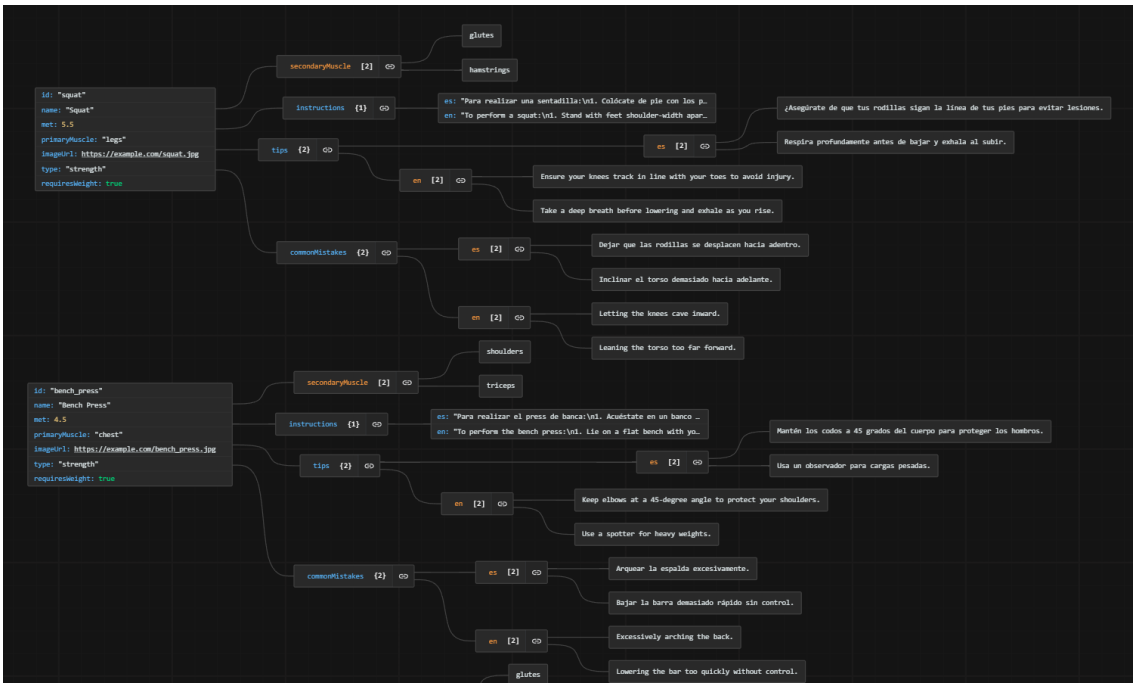

Ilustración 4.12: Visualización de json con JSONCrack.

4.7.3. Diagrama del modelo de datos

La figura 4.13 muestra el diagrama de clases del modelo de datos utilizado en la aplicación. En él se representan las entidades principales, sus atributos y las relaciones entre ellas.

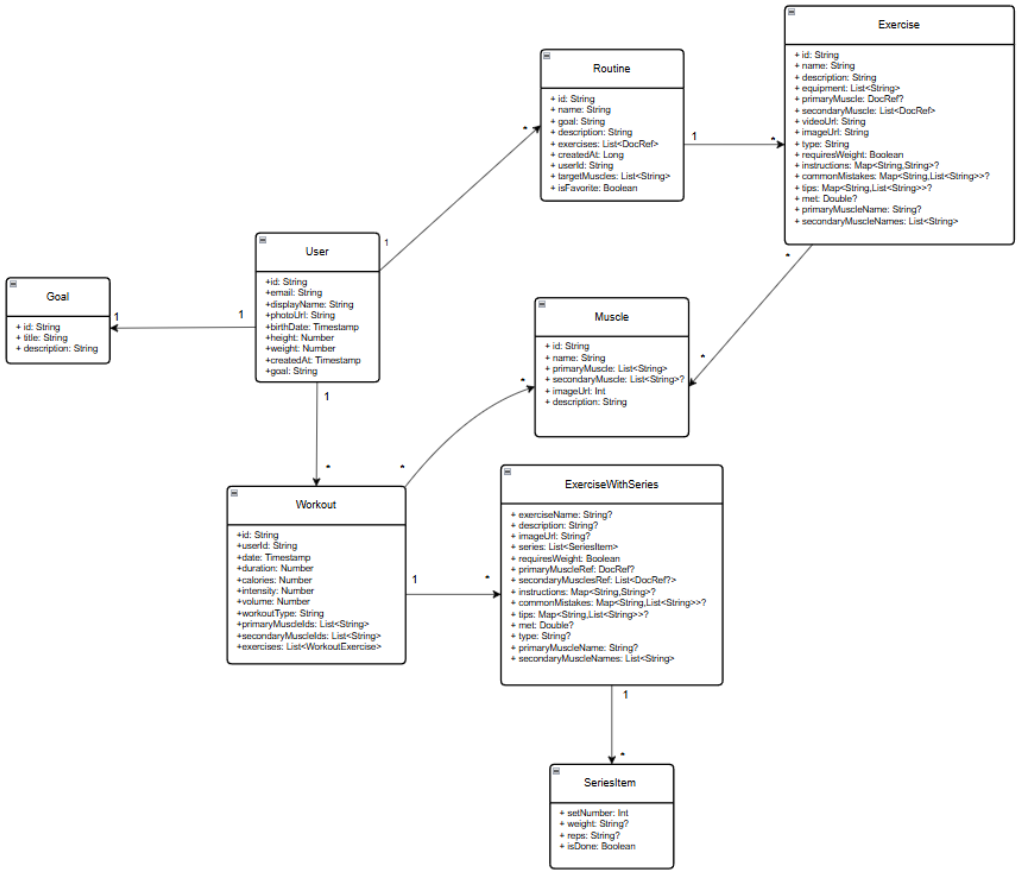

Ilustración 4.13: Diagrama de clases del modelo de datos.

4.8. Patrón de diseño

4.8.1. Arquitectura MVVM

El patrón MVVM 4.14 es una arquitectura de diseño que permite una separación clara de responsabilidades entre la lógica de negocio, la interfaz de usuario y el estado de la aplicación.

La librería ViewModel de Jetpack facilitó esta implementación, ya que permite almacenar y gestionar datos relacionados con la interfaz de usuario de forma reactiva, conservando el estado entre cambios de configuración, como las rotaciones de pantalla.

Esta arquitectura facilita el mantenimiento del código, promueve la reutilización de componentes y mejora la capacidad de testeo de la lógica de presentación, al desacoplarla completamente de la vista.

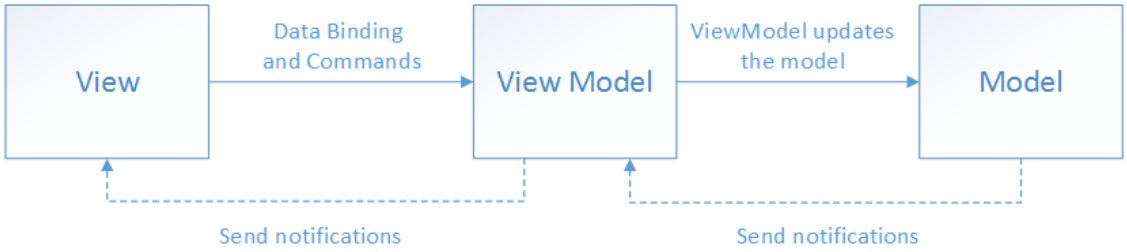

Ilustración 4.14: Diagrama del patrón MVVM [24].

A continuación, se presenta un ejemplo simplificado del uso de MVVM en la pantalla de metas:

```
@Composable
fun GoalsScreen(
    innerPadding: PaddingValues,
    viewModel: GoalsScreenViewModel,
    navHostController: NavHostController
) {
    val state by viewModel.state.collectAsState()
    val buttonConfigs = state.buttonConfigs

    FormScreen(
        modifier = Modifier,
        innerPadding = innerPadding,
        isContentScrolleable = true,
        formContent = {
            Text(
                text = stringResource(id = R.string.life_style),
                style = MaterialTheme.typography.headlineSmall,
                color = MaterialTheme.colorScheme.onBackground,
                modifier = Modifier.padding(bottom = 16.dp)
            )

            GoalsSection(
                options = viewModel.getGoalsOptions(),
                selectedOption = state.goal,
                onOptionSelected = viewModel::onGoalSelected
            )


        },
        formButton = {
            NewCustomButton(
                text = stringResource(id = R.string.next),
                onClick = {
                    viewModel.uploadData(navHostController = navHostController)
                },
                buttonType = ButtonType.FILLED,
                containerColor = Color.Black,
                textConfig = buttonConfigs.textConfig,
                layoutConfig = buttonConfigs.layoutConfig,
                stateConfig = buttonConfigs.stateConfig,
                borderConfig = buttonConfigs.borderConfig
            )
        }
```

```
    )
}
```

En este ejemplo, la pantalla (*View*) muestra los datos que le proporciona el `ViewModel`, y reacciona automáticamente cuando cambian. Por ejemplo, si el usuario selecciona una meta distinta, esa información se actualiza en la interfaz sin necesidad de realizarlo manualmente. Además, al seleccionar una de las metas, los datos se envían al *backend* y se gestionan desde el `ViewModel`, lo que permite mantener el código más organizado y separar claramente la interfaz de usuario de la lógica de negocio.

4.9. Estructura del código

4.9.1. Android

En esta sección se muestra la organización del código según el patrón MVVM. Este enfoque permite separar la lógica de presentación (UI) de la lógica de negocio y de acceso a datos, lo que facilita el mantenimiento, la escalabilidad y la legibilidad del código.

A continuación, se presentan capturas de pantalla que ilustran la organización del proyecto, mostrando las diferentes carpetas y archivos agrupados por su función dentro del patrón:

- ✔ **Model**: Contiene las clases que representan los datos de la aplicación (por ejemplo, usuarios, ejercicios, rutinas, etc.). Estas clases definen la estructura de la información y se utilizan para comunicar los datos entre Firebase y la interfaz.
- ✔ **View**: Incluye las pantallas (*composables*) que conforman la interfaz gráfica de usuario. Cada vista observa los datos expuestos por el ViewModel y reacciona a los cambios.
- ✔ **ViewModel**: Aloja la lógica de presentación. Los ViewModels se encargan de interactuar con los modelos y exponer los datos en un formato adecuado para las vistas. Además, gestionan el estado de la interfaz y responden a eventos del usuario.

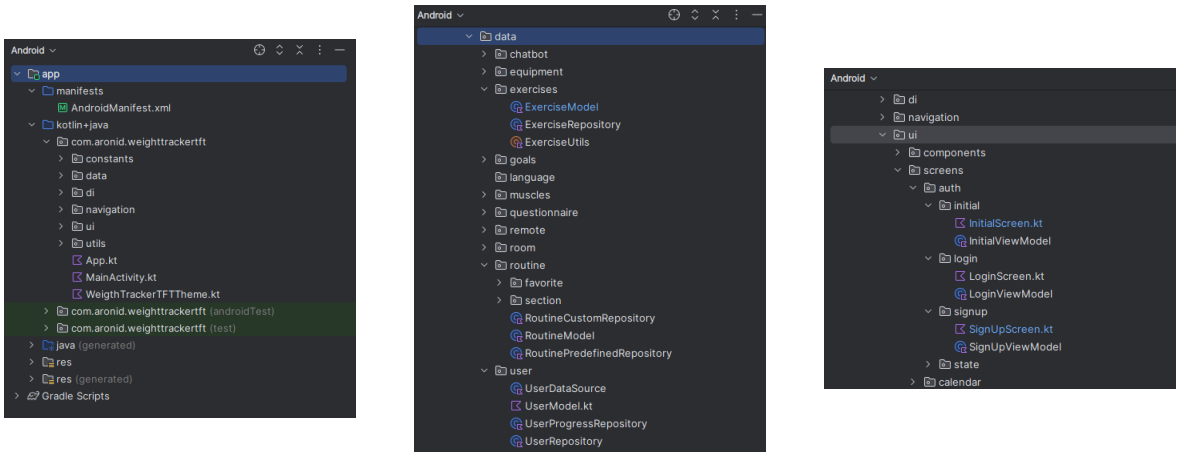

Ilustración 4.15: Estructura de carpetas del proyecto.

4.9.2. Servidor

La organización del servidor sigue una estructura modular para separar la lógica por responsabilidades, facilitando el mantenimiento y la escalabilidad del proyecto. A continuación, se describen las carpetas principales:

- ✔ **routes/**: Contiene las rutas del servidor. Aquí se define el endpoint `/webhook`, que recibe las peticiones provenientes de Dialogflow.
- ✔ **services/**: Incluye la lógica principal del sistema. Está dividido en varios archivos:
  - `exerciseService.js`: Funciones para obtener ejercicios desde Firestore.
  - `routineService.js`: Generación de rutinas de entrenamiento de forma dinámica.
  - `intentService.js`: Procesamiento de los mensajes del usuario e interpretación de intenciones.
- ✔ **utils/**: Contiene funciones auxiliares que apoyan al resto del código, como formateadores de texto o herramientas de traducción.
- ✔ **index.js**: Archivo principal del servidor. Se encarga de iniciar la aplicación, cargar las rutas y configurar los servicios.

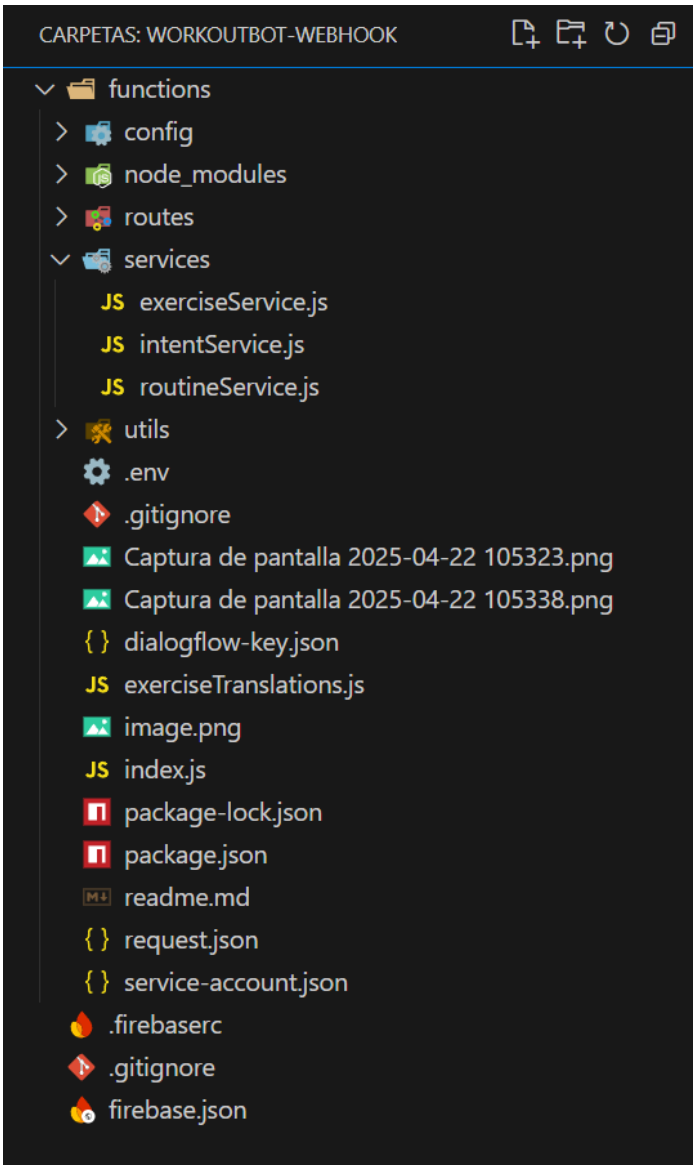

Ilustración 4.16: Estructura del proyecto del servidor Node.js para el chatbot.

4.10. Modificación en los objetivos planteados

Durante el transcurso del proyecto, los objetivos inicialmente definidos se mantuvieron en su mayoría, aunque fue necesario realizar varios cambios para poder obtener una mayor calidad del producto y adaptarse a los recursos disponibles de manera gratuita.

Uno de los cambios más relevantes fue la incorporación del desarrollo de un chatbot mediante Dialogflow, no contemplado en la propuesta inicial. Este nuevo objetivo surgió como respuesta al interés por mejorar la interacción entre la aplicación y el usuario, permitiendo resolver dudas frecuentes y ofrecer recomendaciones de manera más dinámica y accesible.

Por otro lado, se descartaron objetivos secundarios como la integración con dispositivos externos (por ejemplo, pulseras de actividad), debido a que su implementación hubiera requerido una inversión de tiempo considerable.

En conclusión, las modificaciones realizadas permitieron mantener el enfoque principal del proyecto, que es desarrollar una aplicación funcional y útil para el seguimiento de entrenamientos orientados a la pérdida de peso.

4.11. Casos de uso

En este apartado se presentan los conceptos fundamentales de los diagramas de casos de uso, así como su aplicación al sistema desarrollado.

Actor

En un diagrama de casos de uso, un **actor** representa una entidad externa que interactúa con el sistema. No se refiere a una persona específica, sino a un rol que puede ser desempeñado por un usuario, otro sistema o un dispositivo.

Los actores pueden clasificarse de la siguiente manera:

- ✔ **Actores primarios**: Aquellos que inician la interacción con el sistema.
- ✔ **Actores secundarios**: Aquellos que colaboran o complementan la ejecución de las funcionalidades del sistema.

En el contexto de esta aplicación, el actor principal es el **usuario**, quien puede registrar, consultar y exportar entrenamientos, así como visualizar estadísticas relacionadas con su progreso físico. La aplicación no diferencia entre tipos de usuario, permitiendo una experiencia homogénea.

Casos de uso

Un **caso de uso** representa una funcionalidad específica desde el punto de vista del usuario. Se define como una secuencia de interacciones entre el actor y el sistema que permite alcanzar un objetivo determinado, proporcionando un resultado valioso.

Los principales casos de uso identificados en esta aplicación son:

- ✔ Registrar un nuevo entrenamiento.
- ✔ Visualizar el historial de entrenamientos.
- ✔ Consultar estadísticas de progreso.
- ✔ Exportar entrenamientos en formato CSV o PDF.

Asociación entre actores y casos de uso

La **asociación** se representa mediante una línea que conecta un actor con un caso de uso. Esta conexión indica que el actor participa directamente en esa funcionalidad.

Por ejemplo, el actor **Usuario** está asociado con el caso de uso **Registrar entrenamiento**, lo cual implica que el usuario inicia dicha acción y el sistema responde almacenando la información correspondiente.

Diagrama de casos de uso

La Figura 4.17 ilustra gráficamente las interacciones entre el actor y los casos de uso descritos:

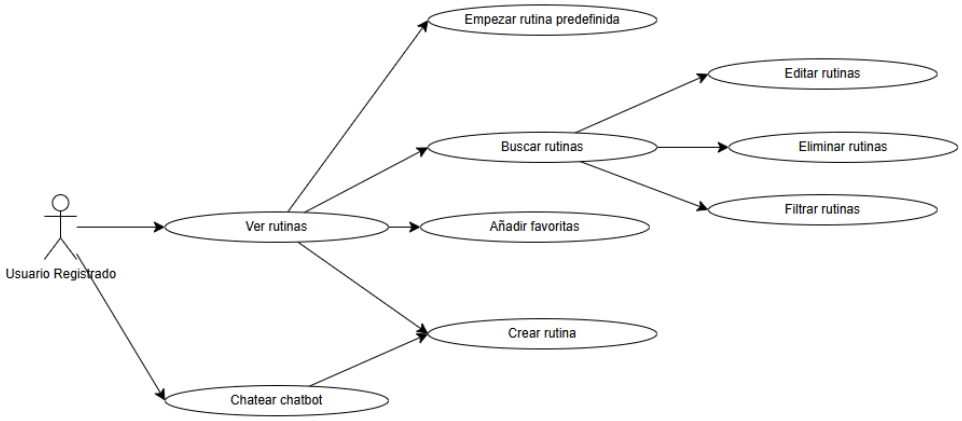

Ilustración 4.17: Diagrama de casos de uso de la aplicación.

Ejemplos de caso de uso

Crear cuenta

- ✔ **Descripción:** El actor podrá crear una cuenta utilizando su correo electrónico y contraseña.
- ✔ **Actor:** Usuario
- ✔ **Flujo de eventos:**
  1. El usuario decide crear una cuenta.

2. El sistema solicita al actor los datos necesarios para crear la cuenta: correo y contraseña.
3. El actor ingresa los datos y confirma la acción.
4. El sistema valida la información proporcionada y crea la cuenta.
5. El actor recibe una confirmación de que la cuenta ha sido creada exitosamente.

✔ **Precondiciones:** El actor no tendrá ninguna cuenta asociada a los datos introducidos.

✔ **Postcondiciones:** El actor ha creado una cuenta con éxito y está habilitado para iniciar sesión con los datos proporcionados.

Cuestionario Inicial

✔ **Descripción:** Tras iniciar sesión por primera vez, el usuario deberá completar un cuestionario para recopilar la información inicial necesaria para que la aplicación pueda iniciar correctamente.

✔ **Actor:** Usuario

✔ **Flujo de eventos:**

1. El usuario decide si aceptar el uso de datos.
2. El sistema solicita al usuario datos como el nombre, fechas de nacimiento, estilo de vida, el objetivo que quiere alcanzar, la altura y el peso.
3. El usuario ingresa los datos.
4. El sistema valida la información proporcionada.
5. El usuario tiene acceso completo a la aplicación.

✔ **Precondiciones:**

- El usuario debe haber iniciado sesión por primera vez en la aplicación.
- El usuario debe estar en la pantalla inicial del cuestionario.
- El sistema debe estar preparado para recibir y procesar los datos proporcionados por el usuario.

✔ **Postcondiciones:**

- El sistema ha guardado correctamente los datos ingresados por el usuario.
- El usuario tiene acceso completo a todas las funcionalidades de la aplicación, según la información proporcionada en el cuestionario.
- El usuario queda registrado en el sistema con los datos iniciales, y puede acceder a la aplicación sin necesidad de volver a completar el cuestionario.

Crear rutina

- ✔ **Descripción:** El usuario crea una rutina de entrenamiento personalizada seleccionando nombre, objetivo, descripción, músculos objetivos, ejercicios y guardando la rutina en la aplicación.
- ✔ **Actor:** Usuario
- ✔ **Flujo de eventos:**
  1. El usuario selecciona la opción para crear una nueva rutina.
  2. El sistema muestra una lista de ejercicios disponibles.
  3. El usuario selecciona los ejercicios que desea incluir en la rutina.
  4. El usuario establece la cantidad de repeticiones, series y otras configuraciones para cada ejercicio.
  5. El usuario guarda la rutina.
  6. El sistema valida la información proporcionada (revisando repeticiones, series y la selección de ejercicios).
  7. El sistema guarda la rutina en la base de datos.
  8. El sistema confirma que la rutina ha sido creada exitosamente.
- ✔ **Precondiciones:**
  - El usuario debe haber iniciado sesión en la aplicación.
  - El usuario debe estar en la pantalla de creación de rutina.
  - El sistema debe tener la lista de ejercicios disponibles para crear la rutina.
- ✔ **Postcondiciones:**
  - La rutina creada por el usuario queda registrada en el sistema.
  - La rutina puede ser visualizada en el historial de rutinas del usuario.
  - Los ejercicios y configuraciones de la rutina quedan guardados para ser accedidos en el futuro.

Capítulo 5

Resultados

5.1. Pantallas

En esta sección se muestran algunas de las pantallas de la aplicación.Todas siguen un diseño similar con el objetivo de ofrecer al usuario una experiencia más sencilla. La interfaz permite registrar entrenamientos fácilmente y consultar el progreso, teniendo en cuenta las calorías quemadas en cada sesión, lo que facilita la comparación con las calorías consumidas.

5.1.1. Pantalla Inicial

En primer lugar, cuando la aplicación se abre por primera vez, encontraremos esta pantalla inicial 5.1 que nos permite el acceso a registrarnos o iniciar sesión.

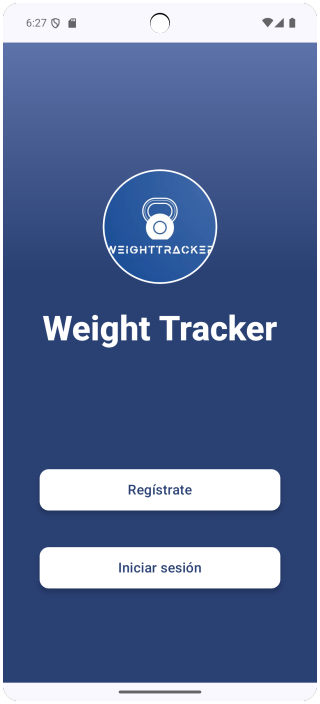

Ilustración 5.1: Pantalla que se ve al iniciar la aplicación.

5.1.2. Registrarse e iniciar sesión

Una vez que se selecciona la opción de registrarse o iniciar sesión 5.2, se presentan las pantallas correspondientes. El usuario tiene la posibilidad de iniciar sesión desde varios dispositivos móviles. En caso de cerrar sesión o cambiar de dispositivo, podrá volver a iniciar sesión sin perder sus datos, ya que toda la información se encuentra sincronizada con la nube mediante Firebase.

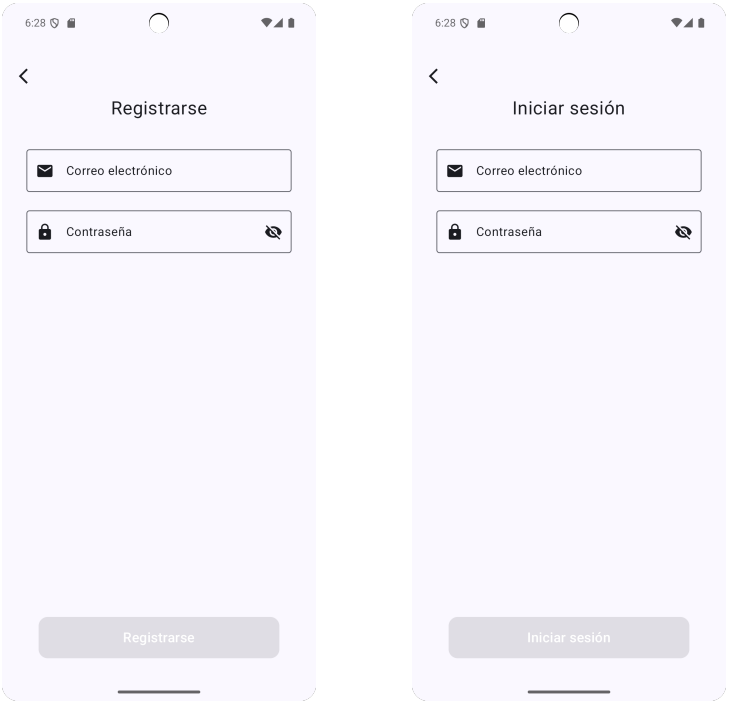

Ilustración 5.2: Pantalla de registro y de iniciar sesión de la aplicación.

5.1.3. Cuestionario Inicial

Al iniciar sesión por primera vez, se muestra un cuestionario inicial 5.3 cuyo objetivo es recopilar la información necesaria para personalizar la experiencia del usuario.

El cuestionario está compuesto por varias pantallas:

- ✔ **Términos y condiciones:** el usuario debe aceptar los términos y condiciones de uso antes de continuar con la configuración inicial.
- ✔ **Información personal:** se solicitan datos como el nombre y la fecha de nacimiento.
- ✔ **Nivel de actividad:** el usuario puede seleccionar entre diferentes tarjetas que representan su nivel de actividad física habitual (sedentario, moderado y activo.).
- ✔ **Objetivo principal:** mediante tarjetas visuales, se elige el objetivo del usuario (por ejemplo: perder grasa, ganar músculo o mantener peso).
- ✔ **Datos físicos:** se introducen valores como la altura y el peso actual del usuario, que serán utilizados en los cálculos y recomendaciones posteriores.

Este cuestionario inicial permite que la aplicación adapte los entrenamientos y sugerencias a las características y objetivos específicos de cada usuario.

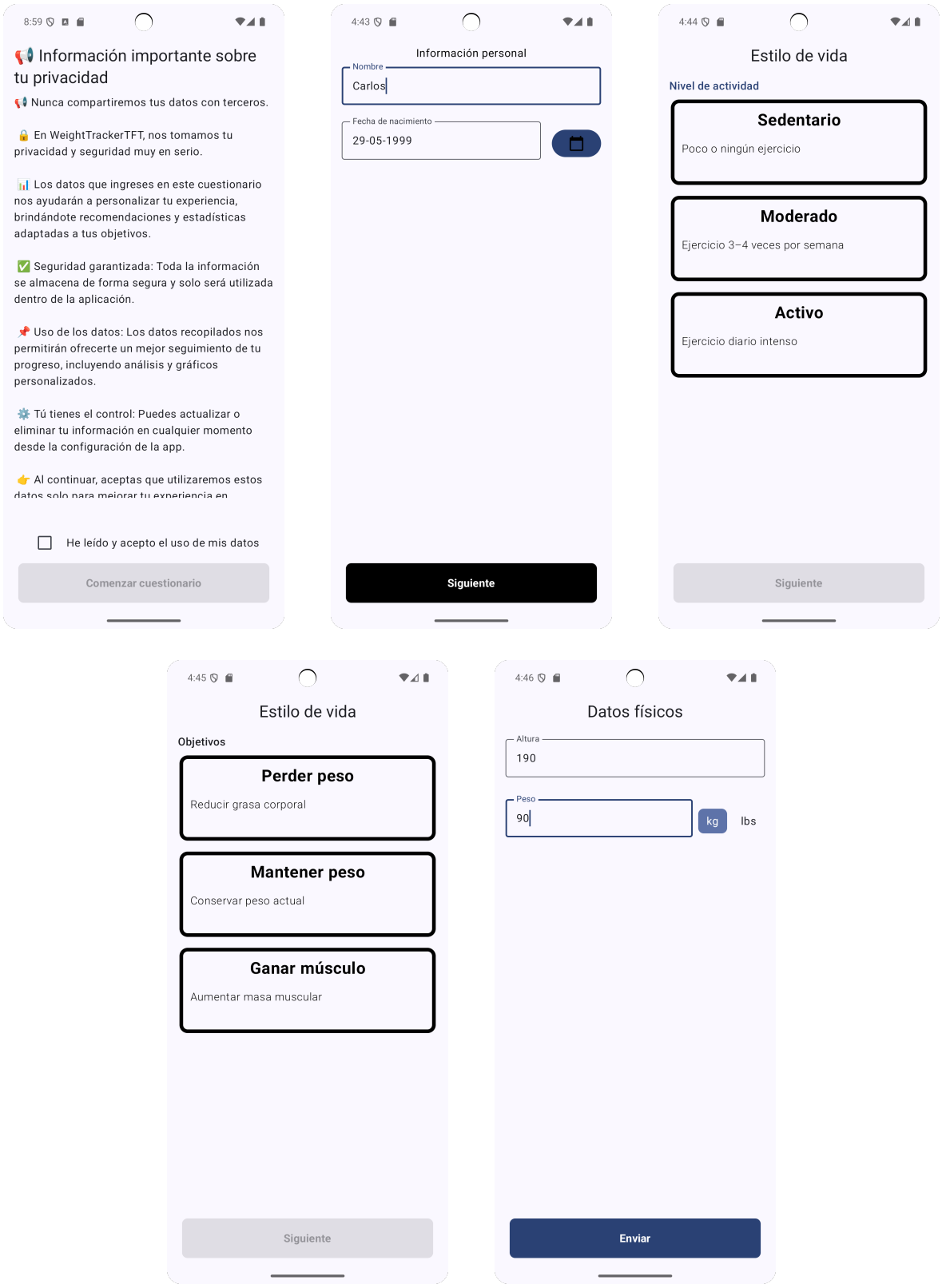

Ilustración 5.3: Cuestionario inicial para obtener la información del usuario.

5.1.4. Pantalla de inicio

Una vez iniciada la sesión, se muestra la pantalla principal 5.4, donde se presenta un calendario semanal correspondiente a la **semana actual**, en el cual se destacan visualmente los días en los que se ha entrenado. Tanto en este calendario como en el mensual, el usuario puede seleccionar cualquiera de los días marcados para acceder a un resumen detallado del entrenamiento realizado en esa fecha. Desde esta pantalla, el usuario dispone de varios accesos directos que facilitan la navegación:

- ✔ Un botón que permite cambiar al **calendario mensual** 5.5 del mes en curso, en el cual también se resaltan los días en los que se ha entrenado.
- ✔ Acceso a las **rutinas disponibles**, tanto las creadas por el usuario como las predefinidas por el sistema.
- ✔ Un botón específico para consultar la lista de **entrenamientos registrados**.
- ✔ Acceso rápido a los **entrenamientos marcados como favoritos**, lo que permite reducir la cantidad de interacciones necesarias al momento de planificar o repetir rutinas.
- ✔ Un botón ubicado en la esquina inferior derecha que permite acceder rápidamente a la pantalla del chatbot, desde donde el usuario puede resolver dudas o recibir asistencia personalizada.

Esta pantalla centraliza las funciones más utilizadas por el usuario, mejorando la usabilidad y facilitando la planificación semanal del entrenamiento. Además, la aplicación cuenta con un **menú de navegación inferior** permanente, que proporciona acceso directo a las principales secciones de la plataforma: **Inicio**, **Progreso**, **Ejercicios** y **Configuración**. Esto permite al usuario moverse entre funcionalidades clave de forma intuitiva y eficiente.

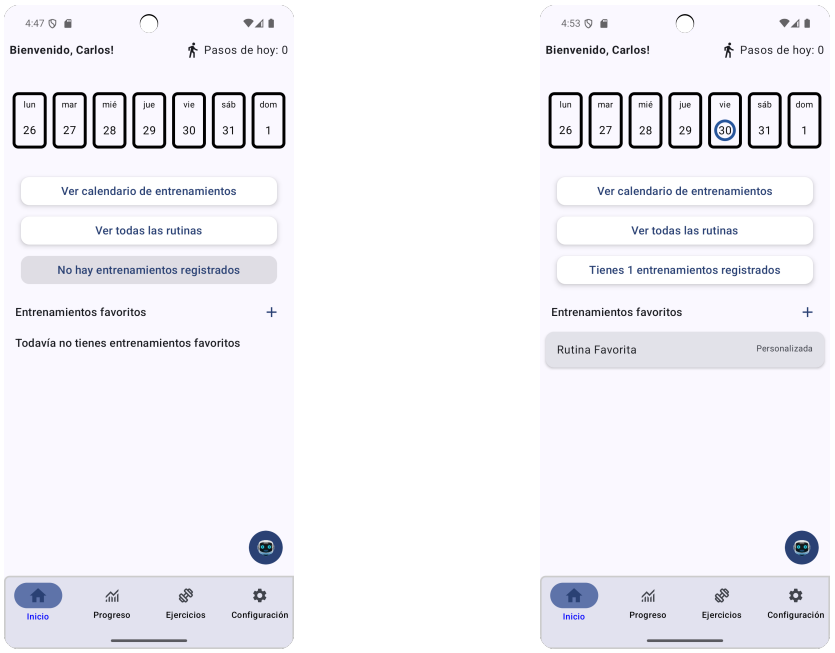

Ilustración 5.4: Pantalla de inicio vacía y con datos registrados.

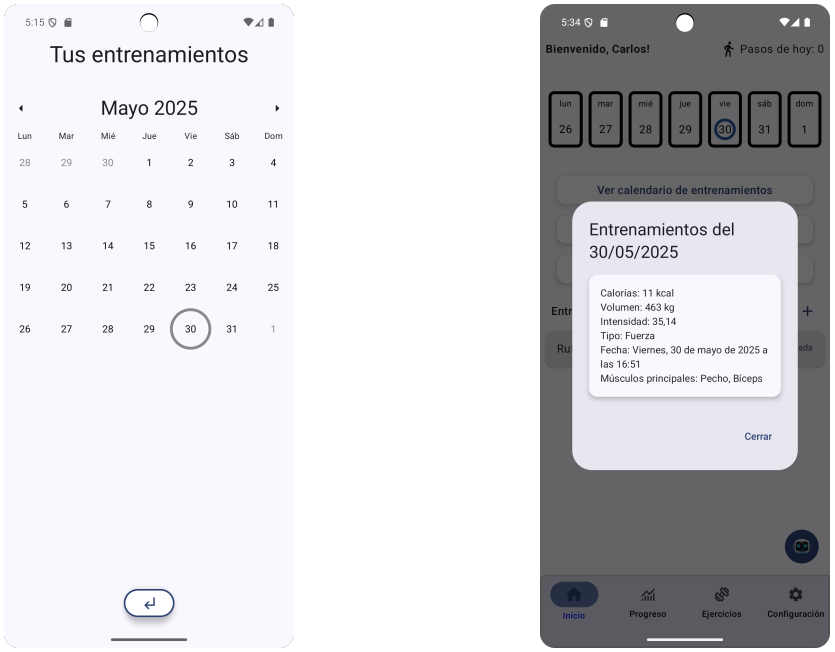

Ilustración 5.5: Calendario mensual y selección de días en los calendarios mensual y semanal, donde el usuario visualiza los entrenamientos del mes seleccionado.

5.1.5. Pantalla del Chatbot

Desde la pantalla del chatbot, mostrada en la figura 5.6, la cual es accesible desde la pantalla de inicio. Se da la bienvenida al usuario con un mensaje inicial y un ejemplo de las respuestas que puede recibir. A partir de este punto, el usuario puede iniciar una conversación sobre diferentes aspectos del entrenamiento, como la generación de rutinas personalizadas, que pueden guardarse para realizar posteriormente, o la consulta sobre los pasos correctos de un ejercicio en específico.

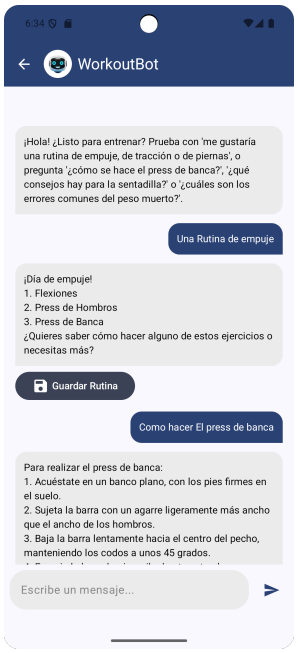

Ilustración 5.6: Pantalla del chatbot.

5.1.6. Pantalla de progreso

La pantalla de **progreso** 5.7 ofrece un resumen del rendimiento del usuario, incluyendo **calorías quemadas**, **volumen de entrenamiento**, **peso actual** e **IMC**. Permite seleccionar distintos **rangos de fechas** (semana actual por defecto, mensual, anual o personalizado mediante un calendario o botones), facilitando el análisis de la evolución física.

Incluye opciones para **exportar datos**, **introducir nuevo peso** y **establecer un objetivo diario de calorías**, todas accesibles desde la **barra superior**. Las **gráficas interactivas** permiten visualizar datos diarios, semanales, mensuales o anuales, con posibilidad de desplazamiento horizontal y selección de puntos para obtener detalles específicos.

En dichas gráficas, el **eje X** representa el tiempo y se adapta al rango seleccionado: si es **semanal**, se muestran los días de la semana; si es **mensual**, se representan por días y si es **anual**, los datos se agrupan por semanas para una visualización más clara. El **eje Y**

muestra el valor correspondiente a cada métrica, como las **calorías quemadas** (en kcal) o el **volumen de entrenamiento** (en kg totales movidos).

Las Figuras 5.7 y 5.8 muestran ejemplos de estas visualizaciones adaptadas a distintos períodos.

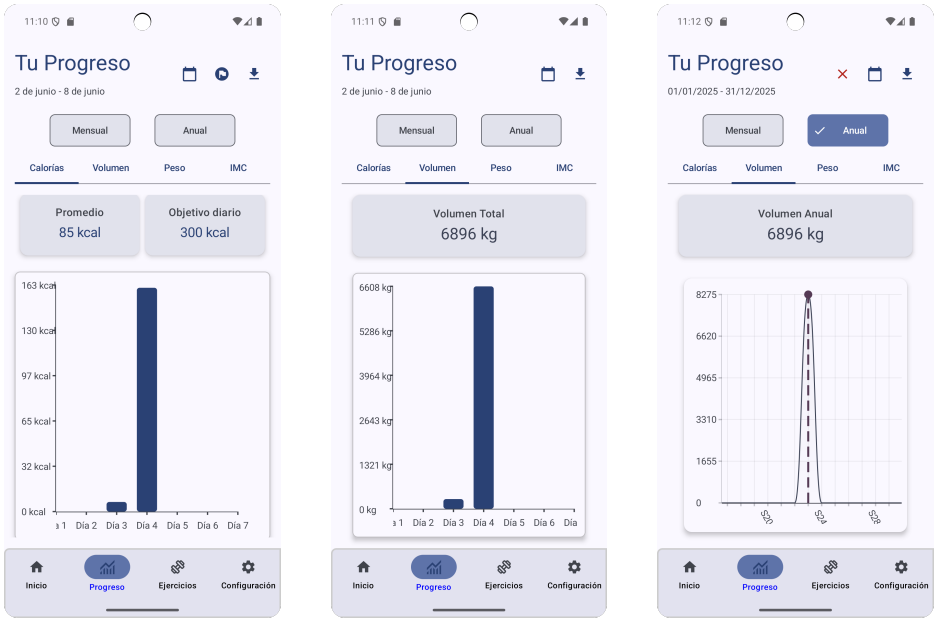

Ilustración 5.7: Pantalla de progreso con opción para seleccionar el rango de fechas.

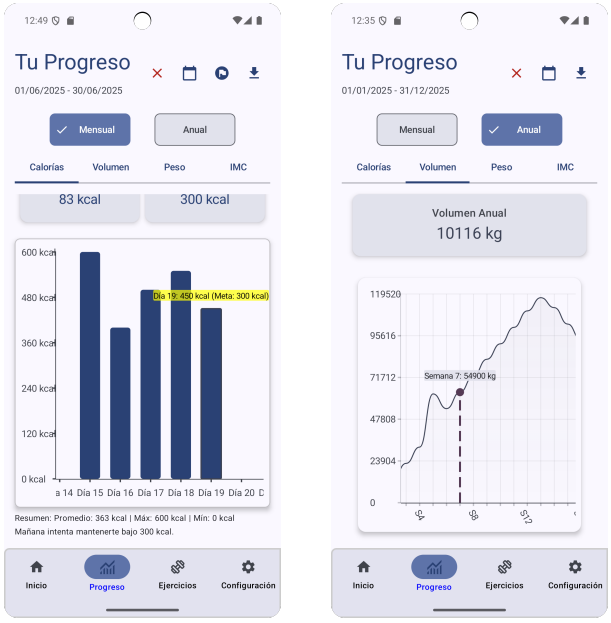

Ilustración 5.8: Resumen gráfico de calorías mensuales y volumen anual de entrenamiento.

5.1.7. Pantalla de entrenamiento

En la pantalla de entrenamiento mostrada en la figura 5.9, el usuario puede registrar los ejercicios seleccionados en su rutina de entrenamiento. En esta pantalla, se presenta una interfaz intuitiva que permite ingresar el peso utilizado (en kilogramos) y las repeticiones realizadas para cada serie del ejercicio. En este caso, el "Press de Bancaçuenta con varias series realizadas, las cuales aparecen con un peso y el número de repeticiones realizadas. Cada serie completada puede marcarse con una casilla de verificación, como se observa en las series 1, 2 y 3.

En la parte superior de la pantalla, se encuentra el nombre del ejercicio ("Press de Banca") junto a un botón de información (representado por un ícono de ï"). Al presionarlo, se despliega un diálogo con datos adicionales sobre el ejercicio, incluyendo los músculos trabajados (como el pecho, tríceps y hombros) y una guía sobre cómo realizarlo correctamente para maximizar los beneficios y evitar lesiones.

En la parte inferior de la pantalla, se incluye un temporizador (mostrando "00:00:20.$^{\text{en}}$ este caso), que ayuda al usuario a llevar un control del tiempo de descanso entre series o del tiempo total del entrenamiento. Además, una barra de navegación, representada con flechas y barras para indicar en qué ejercicio de la rutina se encuentra el usuario, y dichas flechas permiten al usuario desplazarse entre los diferentes ejercicios de la rutina de manera fluida.

Y por último, a la derecha del nombre del ejercicio, se aprecia el botón de "Guardar", que permite al usuario almacenar los datos registrados de las series completadas y terminar su rutina.

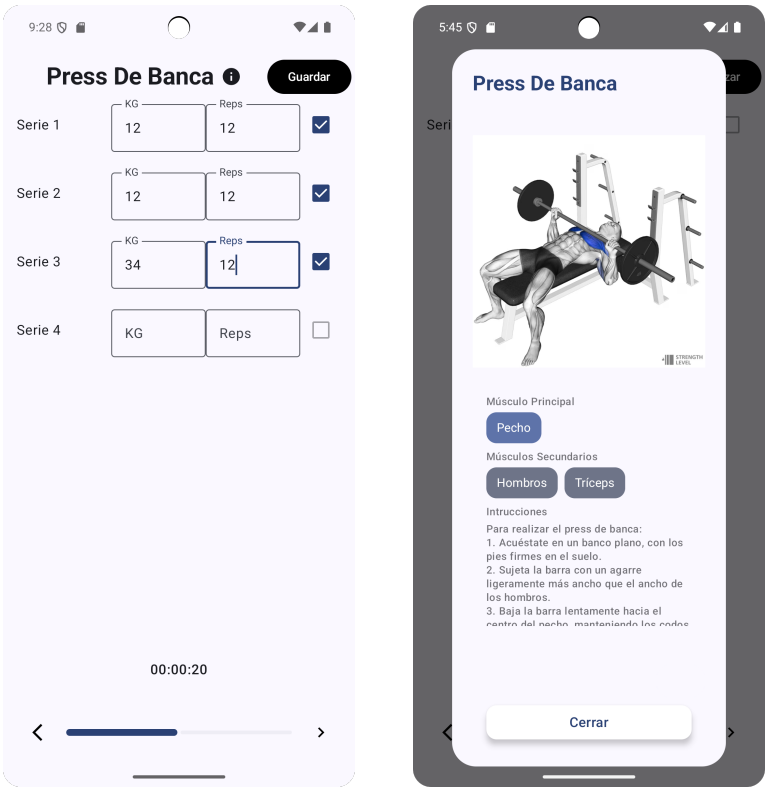

Ilustración 5.9: Pantallas de entrenamiento y diálogo con información adicional sobre el ejercicio.

5.1.8. Pantalla de resumen

En la pantalla de resumen 5.10, el usuario puede acceder a distintas pestañas que muestran información relevante sobre su entrenamiento. Estas incluyen:

- ✔ El tiempo de duración del entrenamiento, las calorías quemadas y el volumen total de entrenamiento, que ofrecen una visión rápida del esfuerzo realizado.
- ✔ Gráficos detallados que visualizan los músculos trabajados durante la sesión.
- ✔ Un listado con los ejercicios realizados, acompañados de las repeticiones y el peso utilizado en cada uno.

Esta organización permite al usuario evaluar fácilmente su progreso y planificar futuros entrenamientos.

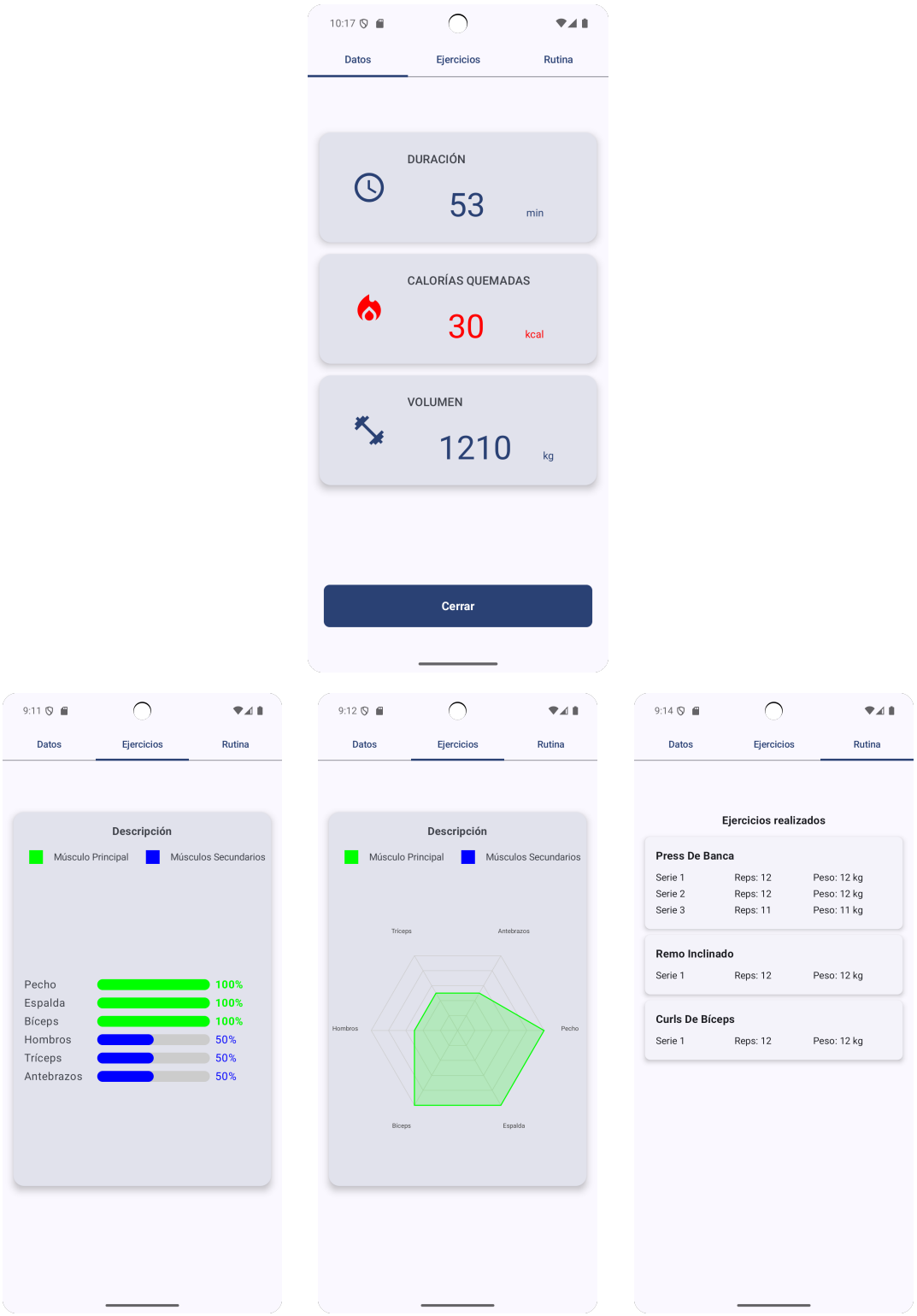

Ilustración 5.10: Pantalla de resumen del historial de datos y ejercicios del usuario.

5.1.9. Pantalla de exportar datos

La pantalla de exportar datos 5.11 permite al usuario generar un archivo PDF o CSV con el resumen de sus entrenamientos registrados. Desde la pantalla principal de la aplicación, el usuario puede acceder al historial completo de entrenamientos mediante el tercer botón del menú, facilitando la consulta rápida de sus sesiones anteriores. Una vez en esta sección, el usuario tiene la opción de seleccionar los entrenamientos que desee y, a continuación, se habilita la opción para descargar los datos en formato PDF o CSV.

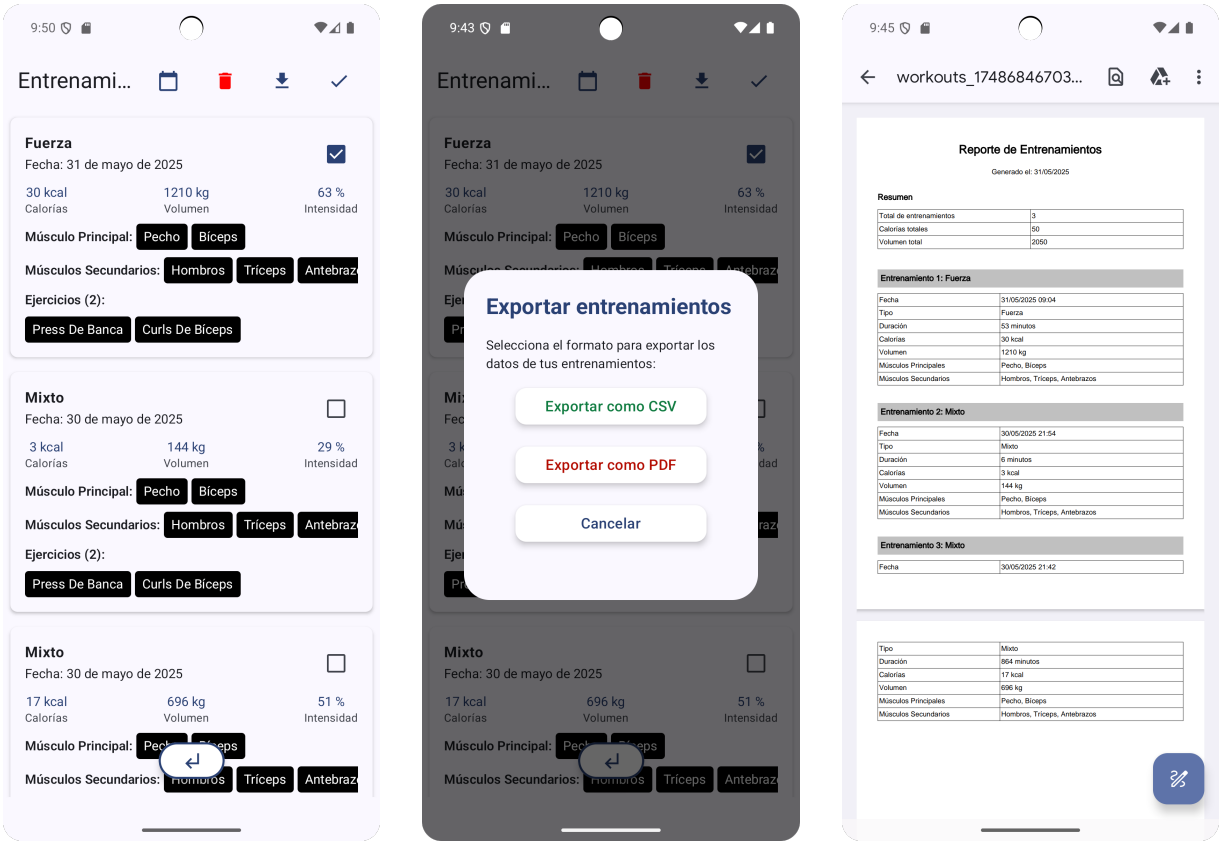

Ilustración 5.11: Pantalla de historial de entrenamientos, opción para exportar datos y vista previa del PDF generado.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

El desarrollo de este proyecto ha supuesto un reto significativo a nivel técnico, mental y organizativo, pero también ha sido una experiencia valiosa y una buena oportunidad de aprendizaje. Durante el proceso, se ha logrado diseñar e implementar una aplicación funcional para el seguimiento del progreso físico, integrando tecnologías como MVVM para la arquitectura y Dialogflow para la interacción con un chatbot, lo que aporta un valor añadido al usuario.

Uno de los aspectos más enriquecedores ha sido enfrentarse a problemas reales y aprender a resolverlos de forma práctica. Al principio, muchas partes del proyecto parecían imposibles, pero con constancia y perseverancia se ha conseguido avanzar poco a poco hasta lograr un resultado funcional y completo.

También ha sido una oportunidad para mejorar la organización del código, seguir buenas prácticas y estructurar el proyecto de forma profesional. Aunque ha habido momentos de frustración, errores difíciles de encontrar y decisiones que hubo que rehacer, todo eso ha servido para ganar experiencia y confianza en el desarrollo de aplicaciones reales.

El proyecto ha cumplido con los objetivos planteados inicialmente, tanto en cantidad como en calidad. Se ha desarrollado una interfaz intuitiva y eficiente, con funcionalidades clave como el registro de usuarios para tener una experiencia más personalizada y segura, la creación de tus propias rutinas para personalizar los entrenamientos, la visualización de gráficas interactivas para el seguimiento del peso, consumo calórico y el volumen levantado, la selección personalizada de rangos de fechas mediante un calendario, y la exportación de datos para análisis externos. La aplicación es capaz de gestionar y mostrar datos reales de forma precisa y clara, ofreciendo una experiencia de usuario completa y satisfactoria.

Es cierto que hay decisiones que ahora se habrían abordado de otra forma, pero la experiencia adquirida permite ver posibles mejoras tanto en la arquitectura como en la organización del desarrollo. Aun así, esas elecciones forman parte del aprendizaje y han sido

fundamentales para comprender mejor el proceso completo.

Además, la experiencia adquirida durante las prácticas de empresa ha sido fundamental para complementar la formación académica y aplicar conocimientos en un entorno profesional real. Esto ha facilitado la comprensión de procesos de desarrollo y gestión de proyectos, y ha permitido incorporar buenas prácticas y metodologías ágiles, enriqueciendo así la calidad del trabajo realizado.

En resumen, este trabajo ha representado un proceso de crecimiento técnico y personal, que ha reforzado la capacidad para afrontar proyectos complejos.

6.2. Trabajo futuro

Una de las principales áreas a desarrollar es el chatbot. Actualmente, ofrece una interacción básica, pero podría mejorarse añadiendo respuestas más personalizadas, integración con rutinas y sugerencias en tiempo real según el historial del usuario.

También sería interesante añadir nuevas funcionalidades relacionadas con la gestión de rutinas, como la posibilidad de compartirlas con otros usuarios. Además, se podría incorporar una estructura por secciones o categorías para agrupar múltiples rutinas de forma más clara, por ejemplo: ”Fuerza”, ”Piernas”, etc.

Por otro lado, se propone implementar un sistema de almacenamiento local de datos que permita a los usuarios utilizar la aplicación sin conexión a internet. Una vez se restablezca la conexión, los datos almacenados localmente se sincronizarán automáticamente en la nube.

Además, se podrían incorporar funcionalidades relacionadas con la nutrición. Una idea interesante sería integrar el uso de la cámara del dispositivo para escanear comidas y estimar sus calorías, aprovechando tecnologías de visión por computador que ya están comenzando a popularizarse en el ámbito de la salud y el fitness.

Otra posible funcionalidad sería la inclusión de un nuevo tipo de usuario: el entrenador. Este rol permitiría ofrecer asesoramiento más profesional dentro de la propia aplicación, junto a un sistema de chat donde el usuario podría consultar directamente con un especialista en entrenamiento o nutrición.

Bibliografía

[1] Appcademy, AristiDevs. (2023). *Jetpack Compose: Curso definitivo desde 0* [Curso en línea]. Appcademy. Recuperado en https://www.appcademy.dev/view/courses/jetpack-compose-curso-definitivo-desde-0-2023

[2] Google Developers. (2025). *Desarrollo de aplicaciones para Android*. Recuperado en https://developer.android.com/develop?hl=es-419

[3] Google. (2025). *Firebase: Plataforma para el desarrollo de aplicaciones*. Recuperado en https://firebase.google.com/?hl=es-419

[4] Google Cloud. (2025). *Dialogflow: Crear experiencias conversacionales*. Recuperado en https://cloud.google.com/dialogflow

[5] Google Cloud. (2025). *Google Cloud: Infraestructura y servicios en la nube*. Recuperado en https://cloud.google.com

[6] OpenAI. (2025). *ChatGPT*. Recuperado en https://chat.openai.com

[7] diagrams.net. (s.f.). *Diagramas en línea*. Recuperado en https://app.diagrams.net/

[8] Kotlin Foundation. (s.f.). *Kotlin Programming Language*. Recuperado en https://kotlinlang.org

[9] Android Developers. (s.f.). *Jetpack Compose*. Recuperado en https://developer.android.com/jetpack/compose

[10] Overleaf. (s.f.). *Editor de LaTeX colaborativo*. Recuperado en https://www.overleaf.com

[11] Trello. (s.f.). *Gestión de proyectos con Trello*. Recuperado en https://trello.com

[12] GitHub. (s.f.). *GitHub: Donde el mundo crea software*. Recuperado en 2025, de https://github.com

[13] Nike. (s.f.). *Aplicación Nike Training Club (NTC)*. Recuperado el 6 de mayo de 2025, de https://www.nike.com/es/NTC-app?msockid=2782d00936af66303ddbc48d37076782

[14] FitKeeper. (s.f.). *FitKeeper – Diario de entrenamiento para el gimnasio*. Recuperado el 6 de mayo de 2025, de https://fitkeeperapp.com/es/

[15] FitKeeper. (s.f.). *FitKeeper – Construye rutinas y sigue el progreso con tus amigos.*. Recuperado el 6 de mayo de 2025, de https://www.hevyapp.com/

[16] Ramos, David. (30/12/2024). *¿Qué tendencias marcarán el sector tecnológico en 2025?*. Silicon.es. Recuperado el 13 de mayo de 2025, de https://www.silicon.es/tendencias-sector-tecnologico-2025-2562104

[17] David Ramos. (13/07/2024). *Android se lleva la corona del número uno en el mundo de los celulares en 2024: qué pasó con iPhone*. El Tiempo. Recuperado el 13 de mayo de 2025, dehttps://www.infobae.com/tecno/2024/07/13/android-se-lleva-la-corona-del-numero-uno-en-el-mundo-de-los-celulares-en-2024-que-paso-con-iphone

[18] Git (s.f.). *Branching and Merging*. Recuperado el 13 de mayo de 2025, de https://git-scm.com/about/branching-and-merging

[19] Google Cloud. (s.f.). *Ediciones de Dialogflow*. Recuperado el 13 de mayo de 2025, de https://cloud.google.com/dialogflow/docs/editions?hl=es-419

[20] Figma. (s.f.). *Figma*. Recuperado el 13 de mayo de 2025, de https://www.figma.com/

[21] Microsoft. (s.f.). *Visual Studio Code*. Recuperado el 13 de mayo de 2025, de https://code.visualstudio.com/

[22] Google. (s.f.). *Android Studio*. Recuperado el 13 de mayo de 2025, de https://developer.android.com/studio?hl=es-419

[23] Google. (s.f.). *Room: Repositorio de base de datos en Android*. Recuperado el 13 de mayo de 2025, de https://developer.android.com/jetpack/androidx/releases/room?hl=es-419

[24] Microsoft. (09/10/2024). *Model-View-ViewModel (MVVM) en .NET MAUI*. Recuperado el 13 de mayo de 2025, de https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

[25] Google. (s.f.). *Hilt: Inyección de dependencias en Android*. Recuperado el 13 de mayo de 2025, de https://developer.android.com/training/dependency-injection/hilt-android?hl=es-419

[26] Chaintech Network. (s.f.). *Compose Multiplatform Date Time Picker*. Recuperado el 13 de mayo de 2025, de https://github.com/Chaintech-Network/compose_multiplatform_date_time_picker

[27] Code and Theory. (s.f.). *YCharts: A Jetpack Compose Chart Library*. Recuperado el 13 de mayo de 2025, de https://github.com/codeandtheory/YCharts

[28] Square. (s.f.). *OkHttp: An HTTP & HTTP/2 Client for Android and Java Applications*. Recuperado el 13 de mayo de 2025, de https://github.com/square/okhttp

[29] Google. (s.f.). *Configuración de Firebase para Android*. Recuperado el 13 de mayo de 2025, de https://firebase.google.com/docs/android/setup?hl=es-419

[30] OpenCSV. (s.f.). *OpenCSV: A CSV parser library for Java*. Recuperado el 30 de mayo de 2025, de http://opencsv.sourceforge.net/

[31] iText. (s.f.). *iText 7 PDF Library – iText PDF*. Recuperado el 30 de mayo de 2025, de https://itextpdf.com/

[32] Scott, B. R., Marston, K. J., Teo, S. Y. M., Forrest, M. R. L., Jonson, A., Walden, T. P., Galna, B., Peiffer, J. J. (2023). *The intensity of a resistance exercise session can be quantified by the work rate of exercise*. Recuperado el 5 de junio de 2025, de https://pmc.ncbi.nlm.nih.gov/articles/PMC10553797/

[33] Essentials of Strength Training and Conditioning, 4th Edition. (2016). *Medicine & Science in Sports & Exercise*. Recuperado el 5 de junio de 2025, de https://journals.lww.com/acsm-msse/Fulltext/2016/10000/Essentials_of_Strength_Training_and_Conditioning,.27.aspx

[34] Darrall-Jones, J. D., Jones, B., Till, K., "Anthropometric, Sprint, and High-Intensity Running Profiles of English Academy Rugby Union Players by Position," *Journal of Strength and Conditioning Research*, May 2016. Recuperado el 6 de junio de 2025, de https://journals.lww.com/nsca-jscr/fulltext/2016/05000/anthropometric,_sprint,_and_high_intensity_running.21.aspx

Índice Terminológico y de Siglas

JSON JavaScript Object Notation, formato de intercambio de datos ligero y legible por humanos. 14

MET Equivalente Metabólico de la Tarea (Metabolic Equivalent of Task). 30, 37

MVVM Modelo-Vista-VistaModelo. 5, 15, 41, 43

NLP Procesamiento de lenguaje natural. 12, 13, 35

Node.js Entorno de ejecución para JavaScript. 9

ODS Objetivos de Desarrollo Sostenible. 6

PDF Portable Document Format. 16

Prettier formateador de código que aplica reglas consistentes para mantener un estilo uniforme en archivos de código fuente. 9

pull request Propuesta de cambios en un repositorio que puede ser revisada y fusionada. 9

reset Comando de Git para cambiar el historial de confirmaciones y modificar el índice. 12

revert Comando de Git para deshacer cambios mediante un nuevo commit. 12

RM Repetición Máxima: Peso máximo que un usuario puede levantar en una sola repetición de un ejercicio con técnica adecuada. 24

script Conjunto de instrucciones o código que se ejecuta en un entorno de programación. 14

SQL Lenguaje de Consulta Estructurado. 13

Staging area Zona intermedia donde se agrupan los cambios antes de confirmarlos con un commit. 12

TFT Trabajo de Fin de Título. 1

TMB Tasa Metabólica Basal. 1

URL Localizador Uniforme de Recursos. 36, 37

VSC Visual studio code. 9

webhook Mecanismo para recibir notificaciones automáticas de eventos externos. 34

XML Extensible Markup Language. 21