DESARROLLO DE UN SISTEMA DE GESTIÓN DE UN CLUB DE RUNNING

TRABAJO FIN DE GRADO
CURSO 2022-2023

AUTOR
MARÍA ESMERALDA PANIAGUA PUENTE

DIRECTOR
ANTONIO SARASA CABEZUELO

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

DESARROLLO DE UN SISTEMA DE GESTIÓN DE UN CLUB DE RUNNING

DEVELOPMENT OF A MANAGING SYSTEM FOR A RUNNING CLUB

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

AUTOR

MARÍA ESMERALDA PANIAGUA PUENTE

DIRECTOR

ANTONIO SARASA CABEZUELO

CONVOCATORIA: JUNIO 2023

GRADO EN INGENIERÍA INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

DÍA DE MES DE AÑO

RESUMEN

Desarrollo de un sistema de gestión de un club de running

El proyecto consiste en la realización de una aplicación para que los miembros de un club de running tengan la posibilidad de gestionar las actividades y necesidades relacionadas con el club.

Las actividades y necesidades que los usuarios tienen disponibles en la aplicación son varias. Pueden ver los entrenamientos que van a realizar con su entrenador, y apuntar sus marcas para ver su progreso. También tienen la opción de crear o apuntarse a entrenamientos grupales para ejercitarse con otros miembros del club. Además, también tienen la opción de ver los eventos y carreras que se van a realizar, y también apuntarse a estos y guardar sus resultados para llevar un seguimiento de su desempeño en las competiciones. Por último, los usuarios pueden proponer sus rutas favoritas para correr para que otros *runners* puedan realizarlas también.

El proyecto ha sido desarrollado como aplicación Android, que trabaja con la base de datos Firebase para la gestión de los datos.

Palabras clave

Aplicación Android, Correr, Carreras, Maratón, Club, Entrenamientos, Rutas

ABSTRACT

Title

The project consists in the development of an application so that the members of a running club have the possibility of managing the activities and necessities related with the club.

The activities and necessities that the users have available in the app are many. They can see the training sessions they have with their trainer and enter their marks to check their progress. Users also have the option to create or to sign up for group training sessions to exercise with other members of the club. They also have the option to see the events and races that are happening and signing up for them and save their results to monitor their performance in the competitions. Finally, users can suggest their favourite routes for running so other runners can also use them.

The project has been developed as an Android application, that works with Firebase database to manage the data.

Keywords

Android Application, Running, Races, Marathon, Club, Trainings, Route

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

En este capítulo se describirán las razones para de la elaboración de este trabajo, y el propósito de este junto con las funcionalidades que se quieren ofrecer a los usuarios. Después, se detallarán los propósitos específicos, la metodología empleada y la forma en la que ha sido planificado el desarrollo del proyecto.

1.1 Motivación

Correr (running) es uno de los deportes más populares a nivel mundial, ya que proporciona muchos beneficios y acerca a las personas a un estilo de vida más saludable. En 2021 un 12,6% de españoles practicaban running [1], la cifra más alta hasta la fecha. Con el aumento en la popularidad de este deporte, han surgido muchos clubs de running que unen a corredores fanáticos de este deporte para entrenar y socializar juntos. Sin embargo, la gestión de estos clubs puede resultar complicada debido a la necesidad de coordinar eventos, entrenamientos y mantener una comunicación fluida entre los miembros, por ello la creación de este tipo de clubs crea la necesidad de algún tipo de gestión online, debido a que en la era actual de las tecnologías es algo muy común, y facilita mucho la comunicación entre el club y los socios de este.

Normalmente, para crear entrenamientos o apuntarse a eventos los corredores tienen que realizar una llamada telefónica o apuntarse en persona; con una aplicación esto se puede realizar de forma rápida, sencilla y desde cualquier lugar. A los runners también les gusta llevar un seguimiento de su progreso, de sus marcas en competiciones y guardar sus rutas de entrenamiento, y todo eso son necesidades que han de ser cubiertas.

Para cubrir todos esos requisitos, y que los miembros puedan tener acceso a ellos desde un mismo lugar, se ha desarrollado una aplicación para Android, en la que los usuarios y los organizadores del club pueden realizar las actividades mencionadas, y de esta forma mejorar y facilitar la gestión del club y sus eventos.

1.2 Objetivos

El objetivo principal del proyecto es desarrollar un sistema que mejore la experiencia de los usuarios que forman parte del club de running, así como crear una mejor comunicación entre el club y sus miembros. Esto se consigue a través de los siguientes objetivos:

- Implementar funcionalidades para la gestión de la cuenta de usuario, como el registro, inicio de sesión, edición y eliminación del perfil.
- Desarrollar un sistema para la organización de eventos, que permita a los usuarios visualizarlos y apuntarse a ellos, así como apuntar sus marcas conseguidas en ellos.
- Implementar una gestión de entrenamientos grupales e individuales, para que los usuarios puedan ver sus próximos entrenamientos, o crear algunos con otros runners y apuntarse a ellos, además de registrar sus estadísticas.
- Ofrecer la posibilidad de compartir sus rutas de entrenamiento favoritas y ver las rutas publicadas por otros usuarios.
- Crear sistema de gestión por parte de los entrenadores del club, para poder organizar los entrenamientos y publicar los eventos de running.

1.3 Plan de Trabajo

La estrategia de planificación se llevó a cabo a través de hitos. Se realizó una primera reunión con el tutor del TFG para establecer los objetivos de la aplicación, y a partir de eso se realizó una especificación de requisitos.

Una vez que se establecieron los requisitos, se clasificaron en diferentes categorías, y cada una de estas categorías se convirtió en una iteración independiente. La organización temporal de las iteraciones se muestra en el diagrama de Gantt de la figura 1.1.

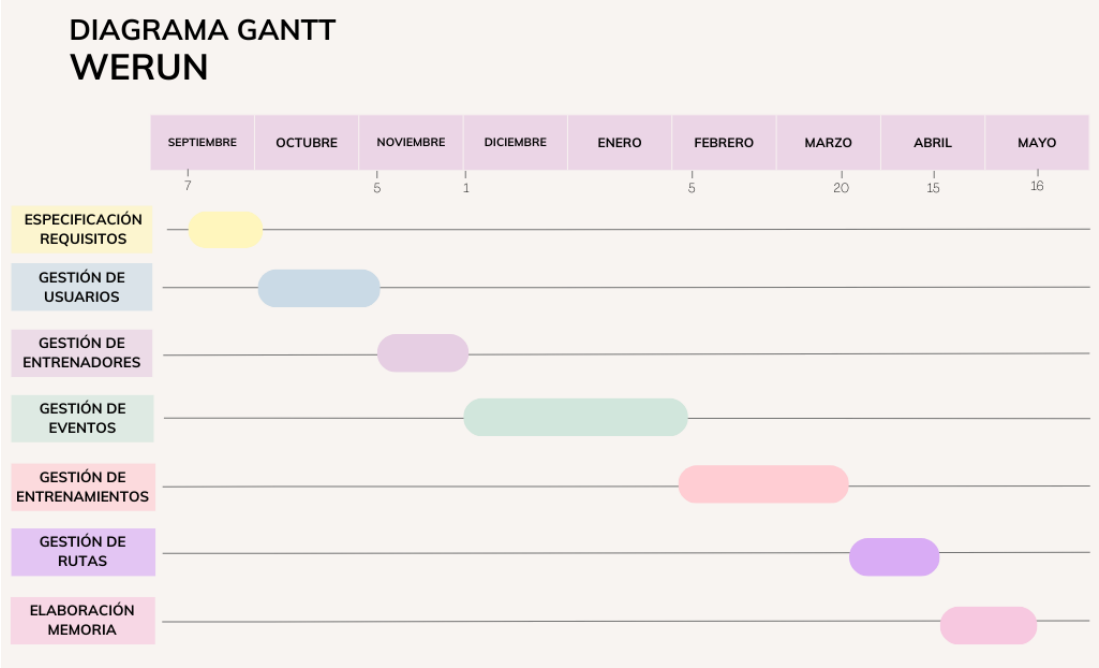

Figura 1 Diagrama de Gantt con las etapas del desarrollo del proyecto

Etapa	Duración
Especificación de Requisitos	7 Septiembre – 30 Septiembre
Gestión de Usuarios	1 Octubre – 5 Noviembre
Gestión de Entrenadores	6 Noviembre – 30 Noviembre
Gestión de Eventos	1 Diciembre – 5 Febrero
Gestión de Entrenamientos	6 Febrero – 20 Marzo
Gestión de Rutas	21 Marzo – 15 Abril
Elaboración Memoria	16 Abril – 16 Mayo

Tabla 1 Etapas en el desarrollo del proyecto

A continuación, se detallarán las fases del proyecto:

Fase 1: Especificación de requisitos

Esta primera fase consistió en decidir y describir las funcionalidades de la aplicación y los casos de uso, dividirlas en categorías y en realizar la organización de los hitos.

Fase 2: Gestión de Usuarios

En la segunda fase se configuró la base de datos y se empezó con la programación de la aplicación. Se implementaron funciones de registro, inició de sesión y gestión del perfil de usuario.

Fase 3: Gestión de Entrenadores

Esta fase estuvo dedicada a implementar la parte administrativa de la aplicación, en la que los entrenadores podían crear y gestionar los entrenamientos con sus alumnos y publicar los eventos disponibles.

Fase 4: Gestión de Eventos

En esta fase se desarrollaron las funcionalidades relacionadas con eventos por parte del usuario, para que este pueda visualizar la lista de eventos disponibles, apuntarse o desapuntarse a ellos y añadir y ver sus marcas.

Fase 5: Gestión de Entrenamientos

Durante esta etapa se llevó a cabo la programación de las herramientas necesarias para la gestión de los entrenamientos, esto incluye ver los entrenamientos individuales con el entrenador como la administración de los entrenamientos grupales por parte de los usuarios.

Fase 6: Gestión de Rutas

La fase 6 fue la última fase de programación, y esta estuvo dedicada a implementar todas las funcionalidades relacionadas con las rutas.

Fase 7: Elaboración Memoria

Esta última fase se centró en la creación y elaboración de la memoria del proyecto.

1.4 Estructura de la memoria

En este apartado se describirá la estructura de este documento:

- **Capítulo 1 – Introducción**: motivación para la realización del proyecto, objetivos marcados y plan de trabajo y su estructura.

- **Capítulo 2 - Estado del arte:** descripción de aplicaciones similares y sus características.
- **Capítulo 3 - Tecnología empleada:** enumeración y explicación de las diversas tecnologías utilizadas para la realización del proyecto.
- **Capítulo 4 - Especificación de Requisitos:** definición de los actores y diagramas de casos de uso del proyecto y sus tablas explicativas.
- **Capítulo 5 - Arquitectura de la aplicación:** descripción de la estructura de la aplicación y el modelo de datos, explicación de cada colección de la base de datos.
- **Capítulo 6 – Diseño:** explicación del diseño de la aplicación.
- **Capítulo 7 – Funcionalidades:** descripción de las funcionalidades del proyecto, con visuales de la aplicación y fragmentos de código.
- **Capítulo 8 - Conclusiones y trabajo futuro:** conclusiones del desarrollo de la aplicación y las posibles mejoras y adiciones que podrían ser implementados.

1-Introduction

In this chapter, the reasons for the elaboration of this work and the purpose of it along with the functionalities that are intended to be offered to users will be described. Then, the specific purposes, the methodology employed, and the way in which the development of the project has been planned will be detailed.

1.1 Motivation

Running is one of the most popular sports worldwide, as it provides many benefits and brings people closer to a healthier lifestyle. In 2021, 12.6% of Spanish people were practicing running [1], the highest number to date. With the increase in the popularity of this sport, many running clubs have emerged, bringing running enthusiasts together to train and socialize together. However, managing these clubs can be complicated due to the need to coordinate events, training sessions, and maintain fluid communication between members. Therefore, the creation of these types of clubs creates a need for some type of online management, as it is a common practice in the current era of technology, and greatly facilitates communication between the club and its members. Usually, to create training sessions or sign up for events, runners must make a phone call or sign up in person; with an app, this can be done quickly, easily, and from anywhere. Runners also like to keep track of their progress, their performance in competitions, and save their training routes, and all of these are needs that must be met.

To meet all these requirements and provide members with access to them from one place, an Android application has been developed, in which users and club organizers can perform the mentioned activities, and in this way, improve and facilitate the management of the club and its events.

1.2 Objectives

The main goal of the project is to develop a system that improves the experience of the users who are part of the running club, as well as creating better communication between the club and its members. This will be achieved through the following objectives:

- Implement functionalities for user account management, such as registration, login, profile editing, and deletion.

- Develop a system for events organizing, which will allow users to view and sign up for them, as well as record their achieved marks in them.
- Implement group and individual training management system, so that users can view their upcoming workouts, or create some with other runners and sign up for them, as well as record their statistics.
- Offer the possibility of sharing their favourite training routes and view routes published by other users.
- Create a management system for the coaches of the club, so they can organize the training sessions and publish running events.

1.3 Work Plan

The planning strategy was carried out through milestones. A first meeting was carried out with the TFG supervisor to establish the objectives of the application, and from there a requirements specification was made.

Once the requirements were established, they were classified into different categories, and each of these categories became an independent iteration. The temporal organization of the iterations is shown in the Gantt chart in Figure 1.1.

Next, the phases of the project will be detailed:

Phase 1: Requirements Specification

This first phase consisted of deciding and describing the functionalities of the application and use cases, dividing them into categories, and organizing the milestones.

Phase 2: Users Management

In the second phase, the database was configured, and the programming of the application began. User registration, login, and profile management functions were implemented.

Phase 3: Trainers Management

This phase was dedicated to implementing the administrative part of the application, in which coaches could create and manage trainings with their students and publish current events.

Phase 4: Events Management

In this phase, functionalities user related to events were developed, so that they can view the list of available events, sign up or cancel their participation, and add and view their marks.

Phase 5: Trainings Management

During this stage, the necessary tools for managing the trainings were programmed. This includes viewing individual trainings with the coach as well as managing group trainings by the users.

Phase 6: Route Management

Phase 6 was the final programming phase, which was dedicated to implementing all the functionalities related to routes.

Phase 7: Report Elaboration

This last phase focused on creating and writing the project report.

1.4 Memory Structure

In this section, the structure of this document will be described:

- **Chapter 1 – Introduction**: Motivation for the project, goals, work plan and its structure.
- **Chapter 2 – State of the art:** Description of similar applications and their features.
- **Chapter 3 - Technology used:** Enumeration of the various technologies used for the project's implementation.
- **Chapter 4 - Requirements Specification:** Definition of the actors and use case diagrams of the project, along with their explanatory tables.
- **Chapter 5 – App Arquitecture:** Description of the application structure and data model, explanation of each collection in the database.
- **Chapter 6 – Design:** explanation of the app's design.
- **Chapter 7 – Functionalities:** Description of the project's functionalities, including visuals of the application and fragments of the code.

- **Chapter 8 – Conclusions and future work:** Conclusions of the application development and the possible improvements and additions that could be implemented.

Capítulo 2 - Estado del arte

En este capítulo se mencionan otras aplicaciones que tienen características parecidas al sistema creado en este proyecto.

2.1 Adidas Running

Adidas Running [2] es una aplicación de la mano de Adidas está completamente centrada en el running y los entrenamientos. Se puede conectar con el Apple Watch para que tenga acceso a los datos de salud durante el entrenamiento, y para poder ver los datos de este mientras se realiza. Las funcionalidades principales de la aplicación son:

- Listado de retos de running a los que los usuarios pueden unirse y competir con otros participantes.
- Listado de eventos a los que los usuarios pueden apuntarse para participar en ellos. Incluye carreras "online", cada usuario realiza la carrera el mismo día, pero en el lugar que ellos quieran, para competir de forma global con todos los usuarios de la aplicación.
- Registro de actividades. Los usuarios pueden comenzar una ruta, y la aplicación sigue la ruta a través de un mapa y la guarda, así como los datos de distancia, calorías, ritmo medio…
- Seguimiento de progreso en el que se pueden añadir las actividades realizadas con datos como los kilómetros, el tiempo, el pulso medio… Con este apartado se puede monitorizar el progreso en los entrenamientos.

2.2 Strava: corre, camina, pedalea

Strava [3] es una aplicación está dedicada a actividades como la bicicleta, carreras, senderismo y hasta esquí, y está orientada a ser un tipo de red social. Se puede conectar con el Apple Watch. Las funcionalidades principales de la aplicación son:

- Registrar las actividades de carreras, caminatas, bicicletas… y llevar un seguimiento de progreso.
- Visualizar rutas de otros deportistas y crear rutas propias.

- Participar en la comunidad y compartir actividades con amigos y competir con ellos.
- Unirse a retos mensuales y crearlos.

2.3 CR7 Fitness by Crunch

CR7 Fitness by Crunch [4] es una aplicación que está dedicada a los miembros del gimnasio "CR7 Fitness by Crunch", y solo puede ser utilizada si tienes una membresía en uno de sus centros. Las funcionalidades principales de la aplicación son:

- Ver horarios del gimnasio.
- Ver horario de las clases y apuntarse a clases.
- Realizar entrenamientos en la aplicación y ver como se utilizan máquinas del gimnasio.
- Monitorizar actividad física y apuntar datos de entrenamientos.
- Monitorizar progreso de peso y otros parámetros corporales

Capítulo 3 - Tecnología empleada

En este capítulo se enumeran las tecnologías que se han utilizado para el desarrollo del proyecto.

3.1 Android Studio

Android Studio [5] es el entorno de desarrollo integrado oficial para la plataforma de Android. Ha sido desarrollado por Google y está disponible para su utilización en Microsoft Windows, macOs, Chrome OS y GNU/Linux. Admite la utilización de los lenguajes Kotlin, Java y C++, siendo el primero el favorito por parte de Google para desarrollar aplicaciones Android.

Entre sus características están el soporte para la construcción Gradle, plantillas de diseños comunes de Android, editor de diseño con arrastre de componentes, dispositivo virtual de teléfono Android para probar aplicaciones, renderizado en tiempo real…

3.2 Java

Java [6] es un lenguaje de programación de alto nivel derivado de C y C++, pero sin tantas funcionalidades de bajo nivel. Es un lenguaje orientado a objetos, lo que permite estructurar el programa y evita la repetición de código. Debido a que Android ha sido diseñado con Java, es popular el uso de Java para diseñar aplicaciones Android, además es rápido y ofrece variedad de funcionalidades.

3.3 XML

XML [7] son las siglas de Extensible Markup Language o Lenguaje de Marcado Extensible en español. Es un metalenguaje similar a HTML, Es un conjunto de reglas para codificar documentos en un formato que sea compatible con distinto hardware, y no está predefinido, hay que definir las etiquetas. En Android se utiliza para el diseño de la aplicación.

3.4 Firebase

Firebase [8] es una plataforma para facilitar el desarrollo de aplicaciones web y móviles que fue adquirida por Google en 2014. Se encuentra alojada en la nube y es escalable. Los módulos de Firebase utilizados son:

3.4.1 Firebase Authentication

Proporciona servicios de autenticación con bibliotecas preparadas para autenticar a los usuarios de la aplicación.

3.4.2 Cloud Firestore

Base de datos NoSQL para almacenar y sincronizar datos de la aplicación o web. Almacena los datos en documentos.

3.4.3 Cloud Storage

Servicio de almacenamiento que permite guardar contenido pesado de la aplicación como fotografías o vídeos.

Capítulo 4 - Especificación de Requisitos

En este capítulo se van a describir los actores y los casos de uso asociados a la aplicación.

4.1 Actores

En esta sección se definen los actores que intervienen en la aplicación:

- **Usuario no registrado**: este usuario no podrá navegar por la aplicación, solo tendrá la opción de registrarse.
- **Usuario registrado**: representa los miembros del club que están registrados en la aplicación, estos usuarios podrán navegar por la aplicación y ver sus entrenamientos, guardar datos de entrenamientos y carreras, apuntarse a eventos…
- **Administrador**: estos usuarios representarían los entrenadores del club, son los encargados de añadir los eventos a la aplicación y de gestionar los entrenamientos de los usuarios.

4.2 Casos de uso

4.2.1 Gestión de Usuarios

En esta sección se describen los casos de uso del módulo funcional de "Gestión de usuarios".

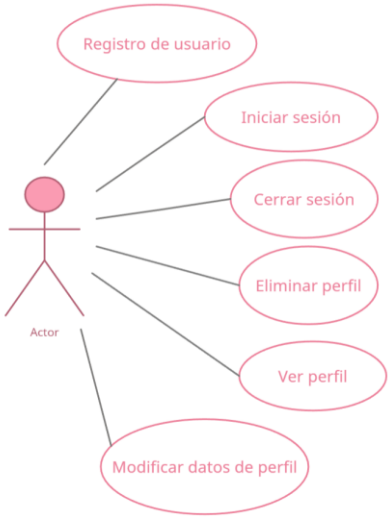

Figura 2 Diagrama de caso de uso Gestión de Usuarios

Requisito: Registro de usuario	
Identificador	1.1
Prioridad	Alta
Precondición	-
Descripción	Creación de cuenta de un usuario para poder utilizar la aplicación
Entrada	Nombre, Apellidos, Fecha de Nacimiento, Email, Nombre de Usuario, Contraseña, Número de Socio
Salida	-
Secuencia normal	1- Usuario accede a la aplicación y se encuentra con la pantalla de inicio 2- El usuario pulsa en el botón "Registrarse" 3- Aparece la pantalla de registro con todos los datos que hay que rellenar 4- El usuario rellena los datos, y pulsa en el botón "Registrarse" 5- La aplicación guarda los datos y se accede al sistema
Postcondición	El usuario se ha registrado
Excepciones	▪ Nombre de usuario ya existe ▪ Ya existe una cuenta con ese email ▪ El email no es correcto
Comentarios	-

Actores	Usuario no registrado

Tabla 2 Requisito: Registro de usuario

Requisito: Iniciar sesión	
Identificador	1.2
Prioridad	Alta
Precondición	El usuario está registrado
Descripción	Inicio de sesión de un usuario registrado para el acceso a la aplicación
Entrada	Nombre de Usuario, Contraseña
Salida	-
Secuencia normal	1- Usuario accede a la aplicación y se encuentra con la pantalla de inicio 2- El usuario pulsa en el botón "Iniciar Sesión" 3- Aparece la pantalla de inicio de sesión con los datos que hay que rellenar 4- El usuario rellena los datos, y pulsa en el botón "Iniciar Sesión" 5- La aplicación valida los datos y se accede al sistema
Postcondición	El usuario ha iniciado sesión
Excepciones	▪ Usuario no existe ▪ Contraseña incorrecta
Comentarios	-
Actores	Usuario registrado

Tabla 3 Requisito: Iniciar Sesión

Requisito: Cerrar sesión	
Identificador	1.3
Prioridad	Alta
Precondición	Hay una sesión iniciada
Descripción	Cierre de sesión en la aplicación para usuarios con la sesión iniciada
Entrada	-
Salida	-
Secuencia normal	1- Usuario accede a su perfil 2- Pulsa en el botón "Cerrar sesión" 3- El sistema cierra la sesión y muestra la pantalla de inicio
Postcondición	Se ha cerrado la sesión
Excepciones	-
Comentarios	-
Actores	Usuario registrado

Tabla 4 Requisito: Cerrar sesión

Requisito: Eliminar perfil	
Identificador	1.4
Prioridad	Baja
Precondición	La cuenta debe existir y la sesión debe estar iniciada
Descripción	Eliminar el perfil del usuario en la aplicación
Entrada	-
Salida	-

Secuencia normal	1- Usuario accede a su perfil 2- Pulsa en el botón “Eliminar cuenta” 3- El sistema pregunta si está seguro, y muestra las opciones “Sí” y “No” 4- El usuario pulsa el botón “Sí” 5- El sistema cierra sesión, elimina la cuenta y muestra la pantalla de inicio
Postcondición	La cuenta se ha borrado
Excepciones	▪ El usuario pulsa el botón “No”
Comentarios	-
Actores	Usuario registrado

Tabla 5 Requisito: Eliminar perfil

Requisito: Ver perfil	
Identificador	1.5
Prioridad	Alta
Precondición	Hay una sesión iniciada
Descripción	El usuario puede ver su perfil y comprobar y cambiar sus datos
Entrada	-
Salida	Perfil del usuario
Secuencia normal	1- El usuario pulsa en el botón/icono del perfil 2- La aplicación muestra el perfil del usuario
Postcondición	Se muestra el perfil
Excepciones	-
Comentarios	-
Actores	Usuario registrado

Tabla 6 Requisito: Ver perfil

Requisito: Modificar datos de perfil	
Identificador	1.6
Prioridad	Media
Precondición	Hay una sesión iniciada
Descripción	El usuario puede modificar sus datos (nombre, cumpleaños, nombre de usuario, email, contraseña)
Entrada	-
Salida	-
Secuencia normal	1- El usuario pulsa en el botón/icono del perfil 2- La aplicación muestra el perfil del usuario 3- El usuario accede al apartado “Datos personales” 4- El usuario pulsa el botón/icono de editar 5- El usuario cambia los datos que desee 6- El sistema valida los datos y los guarda en caso de éxito 7- Se vuelve a mostrar el perfil
Postcondición	Se han modificado los datos
Excepciones	▪ Nombre de usuario ya existe ▪ Ya existe una cuenta con ese email ▪ El email no es correcto

Comentarios	-
Actores	Usuario registrado

Tabla 7 Requisito: Modificar datos de perfil

4.2.2 Gestión de Eventos

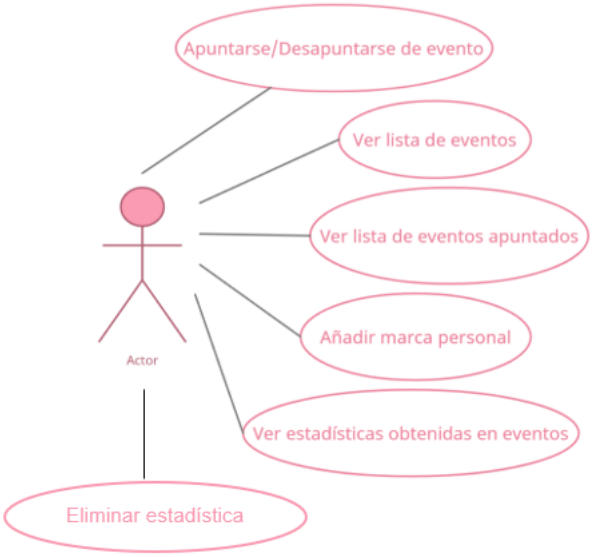

Figura 3 Diagrama de caso de uso Gestión de Eventos

Requisito: Apuntarse/Desapuntarse de evento	
Identificador	2.1
Prioridad	Alta
Precondición	Hay una sesión iniciada
Descripción	El usuario puede apuntarse a una carrera o competición que se vaya a producir, o desapuntarse de un evento al que se haya apuntado previamente
Entrada	-
Salida	-
Secuencia normal	1- El usuario accede al apartado de eventos 2- La aplicación muestra la lista de futuros eventos 3- El usuario pulsa en un evento 4- La aplicación muestra una pantalla con la descripción del evento 5- El usuario pulsa el botón “Apuntarse” o “Desapuntarse”
Postcondición	El usuario se ha apuntado o desapuntado de un evento
Excepciones	▪ El evento está completo
Comentarios	-
Actores	Usuario registrado

Tabla 8 Requisito: Apuntarse/Desapuntarse de evento

Requisito: Ver lista de eventos	
Identificador	2.2
Prioridad	Alta

Precondición	Hay una sesión iniciada
Descripción	Ver la lista de futuras carreras o competiciones
Entrada	-
Salida	Lista de eventos
Secuencia normal	1- El usuario accede al apartado de eventos 2- La aplicación muestra la lista de futuros eventos
Postcondición	Se ha mostrado la lista de eventos
Excepciones	▪ No hay ningún evento en la lista
Comentarios	-
Actores	Usuario registrado

Tabla 9 Requisito: Ver lista de eventos

Requisito: Ver lista de eventos apuntados	
Identificador	2.3
Prioridad	Alta
Precondición	Hay una sesión iniciada
Descripción	El usuario puede ver los eventos a los que se ha apuntado
Entrada	-
Salida	Lista de eventos apuntados
Secuencia normal	1- El usuario accede al apartado de eventos 2- El usuario pulsa el botón “Apuntados” 3- La aplicación muestra la lista de eventos en los que está apuntado el usuario
Postcondición	Se ha mostrado la lista de eventos apuntados
Excepciones	▪ El usuario no se ha apuntado a ningún evento
Comentarios	-
Actores	Usuario registrado

Tabla 10 Requisito: Ver lista de eventos apuntados

Requisito: Añadir marca personal	
Identificador	2.4
Prioridad	Alta
Precondición	Hay una sesión iniciada
Descripción	Guardar marcas personales de carreras o competiciones
Entrada	Nombre Carrera, Fecha, Km, Tiempo, Puesto
Salida	-
Secuencia normal	1. El usuario pulsa en el botón/icono del perfil 2. La aplicación muestra el perfil del usuario 3. El usuario accede al apartado “Marcas personales” 4. La aplicación muestra la lista de datos 5. El usuario pulsa el botón “Añadir marca” 6. La aplicación muestra una pantalla con los datos a rellenar 7. El usuario rellena los datos y pulsa el botón “Añadir” 8. El sistema muestra los datos y se muestra la pantalla de “Marcas personales”
Postcondición	Se ha guardado una marca personal nueva
Excepciones	▪ No se han rellenado todos los datos

Comentarios	-
Actores	Usuario registrado

Tabla 11 Requisito: Añadir marca personal

Requisito: Ver estadísticas obtenidas en eventos	
Identificador	2.5
Prioridad	Media
Precondición	Hay una sesión iniciada, hay marcas de eventos registradas
Descripción	El usuario puede ver las estadísticas de sus marcas obtenidas en eventos, así como sus mejores y peores marcas
Entrada	-
Salida	Estadísticas de eventos
Secuencia normal	1- El usuario pulsa en el botón/icono del perfil 2- La aplicación muestra el perfil del usuario 3- El usuario accede al apartado "Marcas personales 4- La aplicación muestra la lista de datos 5- El usuario pulsa el botón "Estadísticas" 6- La aplicación muestra las estadísticas de eventos
Postcondición	Se han mostrado las estadísticas de entrenamientos
Excepciones	▪ El usuario no ha registrado ningún dato de entrenamiento
Comentarios	-
Actores	Usuario registrado

Tabla 12 Requisito: Ver estadísticas obtenidas en eventos

Requisito: Eliminar estadística	
Identificador	2.4
Prioridad	Baja
Precondición	Hay una sesión iniciada
Descripción	Eliminar marca personal obtenida en un evento
Entrada	-
Salida	-
Secuencia normal	1. El usuario pulsa en el botón/icono del perfil 2. La aplicación muestra el perfil del usuario 3. El usuario accede al apartado "Marcas personales" 4. La aplicación muestra la lista de datos 5. El usuario pulsa el botón "Eliminar" 6. El sistema elimina la Marca Personal
Postcondición	Se ha eliminado una marca personal
Excepciones	-
Comentarios	-
Actores	Usuario registrado

Tabla 13 Requisito: Eliminar estadística

4.2.3 Gestión de Entrenamientos

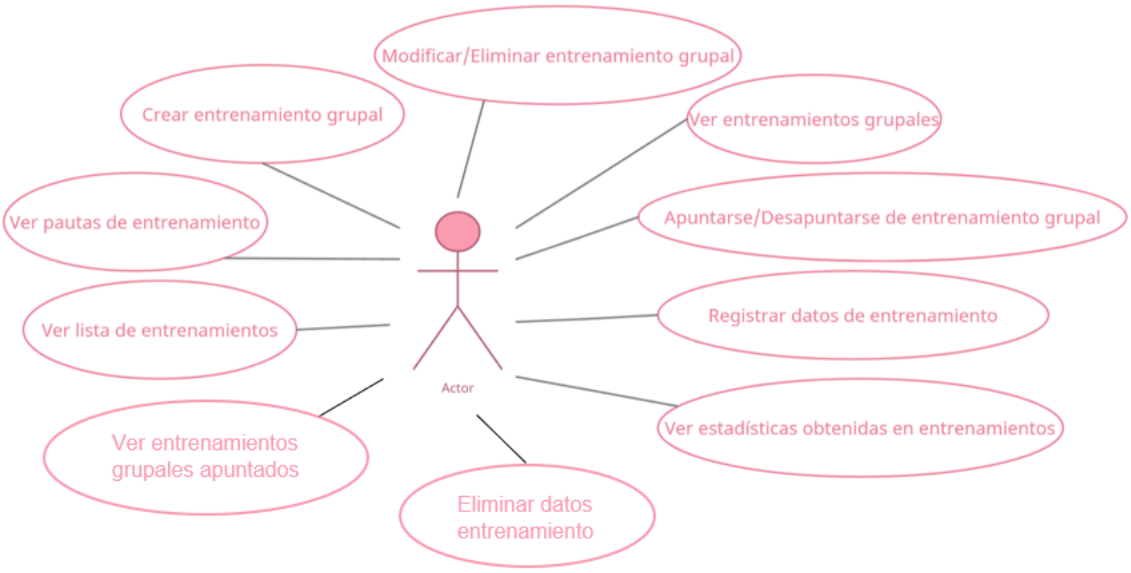

Figura 4 Diagrama de caso de uso Gestión de Entrenamientos

Requisito: Ver lista de entrenamientos	
Identificador	3.1
Prioridad	Alta
Precondición	Hay una sesión iniciada, el usuario tiene entrenamientos registrados
Descripción	El usuario puede ver los días y horas que entrena
Entrada	-
Salida	Lista de entrenamientos
Secuencia normal	1- El usuario accede al apartado de entrenamientos 2- La aplicación muestra la lista de futuros entrenamientos
Postcondición	Se ha mostrado la lista de entrenamientos
Excepciones	▪ El usuario no tiene entrenamientos futuros
Comentarios	-
Actores	Usuario registrado

Tabla 14 Requisito: Ver lista de entrenamientos

Requisito: Ver pautas entrenamiento	
Identificador	3.2
Prioridad	Media
Precondición	Hay una sesión iniciada, el usuario tiene entrenamientos registrados
Descripción	El usuario puede ver las pautas de cada entrenamiento asignadas por su entrenador
Entrada	-
Salida	Descripción del entrenamiento
Secuencia normal	1- El usuario accede al apartado de entrenamientos 2- La aplicación muestra la lista de futuros entrenamientos 3- El usuario pulsa en un entrenamiento

	4- La aplicación muestra la descripción del entrenamiento
Postcondición	Se han mostrado las pautas de un entrenamiento
Excepciones	▪ El usuario no tiene entrenamientos futuros ▪ El entrenamiento seleccionado no tiene pautas registradas
Comentarios	-
Actores	Usuario registrado

Tabla 15 Requisito: Ver pautas entrenamiento

Requisito: Crear entrenamiento grupal	
Identificador	3.3
Prioridad	Media
Precondición	Hay una sesión iniciada
Descripción	Los usuarios pueden crear entrenamientos grupales para poder correr con otros miembros del club, así como modificarlos o eliminarlos
Entrada	Fecha, Hora, Lugar, Aforo, Km
Salida	-
Secuencia normal	1- El usuario accede al apartado de entrenamientos 2- El usuario accede al apartado de entrenamientos grupales 3- El usuario pulsa el botón "Crear entrenamiento grupal" 4- La aplicación muestra una pantalla con los datos que hay que rellenar 5- El usuario rellena los datos y pulsa el botón "Crear entrenamiento"
Postcondición	Se ha creado un entrenamiento grupal
Excepciones	▪ El usuario no ha rellenado todos los datos ▪ Algún dato es incorrecto
Comentarios	-
Actores	Usuario registrado

Tabla 16 Requisito: Crear entrenamiento grupal

Requisito: Modificar/Eliminar entrenamiento grupal	
Identificador	3.4
Prioridad	Media
Precondición	Hay una sesión iniciada, el usuario ha creado algún entrenamiento grupal
Descripción	Los usuarios pueden modificar o eliminar los entrenamientos grupales creados para poder correr con otros miembros del club
Entrada	Fecha, Hora, Lugar, Aforo, Km
Salida	-
Secuencia normal	1- El usuario accede al apartado de entrenamientos 2- El usuario accede al apartado de entrenamientos grupales 3- El usuario pulsa en un entrenamiento grupal 4- La aplicación muestra la descripción del entrenamiento grupal 5- El usuario pulsa el botón "Editar" que aparece si ese usuario ha creado el entrenamiento 6- La aplicación permite modificar los datos o eliminar el entrenamiento
Postcondición	Se ha modificado o eliminado el entrenamiento grupal

Excepciones	▪ El usuario no ha rellenado todos los datos ▪ Algún dato es incorrecto ▪ El usuario no creó el entrenamiento
Comentarios	-
Actores	Usuario registrado

Tabla 17 Requisito: Modificar/Eliminar entrenamiento grupal

Requisito: Ver entrenamientos grupales	
Identificador	3.5
Prioridad	Media
Precondición	Hay una sesión iniciada, hay entrenamientos grupales registrados
Descripción	Los usuarios pueden ver la lista de entrenamientos grupales propuestos por otros usuarios
Entrada	-
Salida	Lista de entrenamientos grupales
Secuencia normal	1- El usuario accede al apartado de entrenamientos 2- El usuario accede al apartado de entrenamientos grupales 3- La aplicación muestra la lista de entrenamientos grupales
Postcondición	Se ha mostrado la lista de entrenamientos grupales
Excepciones	▪ No hay ningún entrenamiento grupal registrado
Comentarios	-
Actores	Usuario registrado

Tabla 18 Requisito: Ver entrenamientos grupales

Requisito: Ver entrenamientos grupales apuntados	
Identificador	3.5
Prioridad	Media
Precondición	Hay una sesión iniciada, el usuario está apuntado a algún entrenamiento grupal
Descripción	Los usuarios pueden ver la lista de entrenamientos grupales a los que se han apuntado
Entrada	-
Salida	Lista de entrenamientos grupales apuntados
Secuencia normal	1- El usuario accede al apartado de entrenamientos 2- El usuario accede al apartado de entrenamientos grupales apuntados 3- La aplicación muestra la lista de entrenamientos grupales
Postcondición	Se ha mostrado la lista de entrenamientos grupales apuntados
Excepciones	▪ El usuario no se ha apuntado a ningún entrenamiento grupal
Comentarios	-
Actores	Usuario registrado

Tabla 19 Requisito: Ver entrenamientos grupales apuntados

Requisito: Apuntarse/Desapuntarse de entrenamiento grupal	
Identificador	3.6
Prioridad	Media
Precondición	Hay una sesión iniciada, hay entrenamientos grupales registrados

Descripción	Los usuarios pueden apuntarse entrenamientos grupales propuestos por otros usuarios o desapuntarse de uno al que se haya apuntado previamente
Entrada	-
Salida	-
Secuencia normal	1- El usuario accede al apartado de entrenamientos 2- El usuario accede al apartado de entrenamientos grupales 3- La aplicación muestra la lista de entrenamientos grupales 4- El usuario pulsa en un entrenamiento grupal 5- La aplicación muestra la descripción del entrenamiento grupal 6- El usuario pulsa el botón "Apuntarse" o "Desapuntarse"
Postcondición	El usuario se ha apuntado o desapuntado de un entrenamiento grupal
Excepciones	▪ No hay ningún entrenamiento grupal registrado ▪ El aforo está lleno
Comentarios	-
Actores	Usuario registrado

Tabla 20 Requisito: Apuntarse/Desapuntarse de entrenamiento grupal

Requisito: Registrar datos entrenamiento	
Identificador	3.7
Prioridad	Alta
Precondición	Hay una sesión iniciada
Descripción	El usuario puede guardar los datos de sus entrenamientos
Entrada	Kilómetros, Tiempo, Latidos
Salida	-
Secuencia normal	1- El usuario pulsa en el botón/icono del perfil 2- La aplicación muestra el perfil del usuario 3- El usuario accede al apartado "Datos entrenamientos" 4- La aplicación muestra la lista de datos 5- El usuario pulsa el botón "Añadir datos" 6- La aplicación muestra una pantalla con los datos a rellenar 7- El usuario rellena los datos y pulsa el botón "Añadir" 8- La aplicación guarda los datos y muestra la lista de datos
Postcondición	Se han guardado unos datos de entrenamiento
Excepciones	▪ El usuario no ha rellenado todos los datos ▪ Algún dato es incorrecto
Comentarios	-
Actores	Usuario registrado

Tabla 21 Requisito: Registrar datos entrenamiento

Requisito: Ver estadísticas obtenidas en entrenamientos	
Identificador	3.8
Prioridad	Media
Precondición	Hay una sesión iniciada, hay datos de entrenamiento registrados
Descripción	El usuario puede ver las estadísticas de sus entrenamientos, así como sus mejores y peores marcas
Entrada	-

Salida	Estadísticas de entrenamientos
Secuencia normal	1- El usuario pulsa en el botón/icono del perfil 2- La aplicación muestra el perfil del usuario 3- El usuario accede al apartado “Datos entrenamientos” 4- La aplicación muestra la lista de datos 5- El usuario pulsa el botón “Estadísticas” 6- La aplicación muestra las estadísticas de entrenamientos
Postcondición	Se han mostrado las estadísticas de entrenamientos
Excepciones	▪ El usuario no ha registrado ningún dato de entrenamiento
Comentarios	-
Actores	Usuario registrado

Tabla 22 Requisito: Ver estadísticas obtenidas en entrenamientos

Requisito: Eliminar datos entrenamiento	
Identificador	3.8
Prioridad	Baja
Precondición	Hay una sesión iniciada, hay datos de entrenamiento registrados
Descripción	El usuario puede eliminar las estadísticas de sus entrenamientos
Entrada	-
Salida	Estadísticas de entrenamientos
Secuencia normal	1- El usuario pulsa en el botón/icono del perfil 2- La aplicación muestra el perfil del usuario 3- El usuario accede al apartado “Datos entrenamientos” 4- La aplicación muestra las estadísticas de entrenamientos 5- El usuario pulsa el botón “Eliminar” 6- El sistema elimina los datos del entrenamiento
Postcondición	Se han mostrado las estadísticas de entrenamientos
Excepciones	▪ El usuario no ha registrado ningún dato de entrenamiento
Comentarios	-
Actores	Usuario registrado

Tabla 23 Requisito: Eliminar datos entrenamiento

4.2.4 Gestión de Rutas

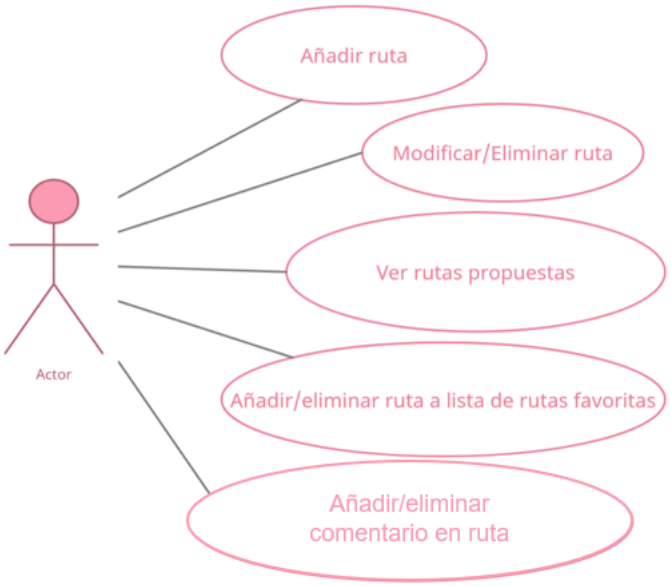

Figura 5 Diagrama de caso de uso Gestión de Rutas

Requisito: Añadir ruta	
Identificador	4.1
Prioridad	Baja
Precondición	Hay una sesión iniciada
Descripción	Los usuarios pueden proponer sus rutas favoritas de entrenamiento para que otros usuarios puedan verlas y realizarlas
Entrada	Inicio ruta, Fin ruta, Mapa Ruta, Kilómetros
Salida	-
Secuencia normal	1- El usuario accede al apartado de rutas 2- La aplicación muestra la lista de rutas 3- El usuario pulsa el botón "Añadir ruta" 4- La aplicación muestra una pantalla con los datos a rellenar 5- El usuario rellena los datos y pulsa el botón "Añadir" 6- La aplicación muestra la lista de rutas
Postcondición	Se ha añadido una ruta nueva
Excepciones	▪ El usuario no ha rellenado todos los datos ▪ Algún dato es incorrecto
Comentarios	-
Actores	Usuario registrado

Tabla 24 Requisito: Añadir ruta

Requisito: Modificar/Eliminar ruta	
Identificador	4.2
Prioridad	Baja
Precondición	Hay una sesión iniciada, el usuario ha añadido alguna ruta
Descripción	Los usuarios pueden modificar o eliminar sus rutas propuestas

Entrada	Inicio ruta, Fin ruta, Mapa Ruta, Kilómetros
Salida	-
Secuencia normal	1- El usuario accede al apartado de rutas 2- La aplicación muestra la lista de rutas 3- El usuario pulsa en una ruta 4- La aplicación muestra los datos de la ruta 5- El usuario pulsa el botón "Editar" que aparece si el usuario ha creado la ruta 6- La aplicación permite modificar los datos o eliminar la ruta
Postcondición	Se ha modificado o eliminado la ruta
Excepciones	▪ El usuario no ha rellenado todos los datos ▪ Algún dato es incorrecto ▪ El usuario no creó la ruta
Comentarios	-
Actores	Usuario registrado

Tabla 25 Requisito: Modificar/Eliminar ruta

Requisito: Ver rutas propuestas	
Identificador	4.3
Prioridad	Baja
Precondición	Hay una sesión iniciada, hay rutas propuestas
Descripción	Los usuarios pueden proponer sus rutas favoritas de entrenamiento para que otros usuarios puedan verlas y realizarlas
Entrada	-
Salida	Lista de rutas
Secuencia normal	1- El usuario accede al apartado de rutas 2- La aplicación muestra la lista de rutas
Postcondición	Se ha mostrado la lista de rutas
Excepciones	▪ No ha sido añadida ninguna ruta
Comentarios	-
Actores	Usuario registrado

Tabla 26 Requisito: Ver rutas propuestas

Requisito: Añadir/Eliminar ruta a lista de rutas favoritas	
Identificador	4.4
Prioridad	Baja
Precondición	Hay una sesión iniciada, hay rutas propuestas
Descripción	Los usuarios pueden agregar rutas a su lista de rutas favoritas
Entrada	-
Salida	-
Secuencia normal	1- El usuario accede al apartado de rutas 2- La aplicación muestra la lista de rutas 3- El usuario pulsa en una ruta 4- La aplicación muestra los datos de la ruta 5- El usuario pulsa el botón "Añadir" o "Eliminar" de favoritos
Postcondición	Se ha añadido o eliminado una ruta de la lista de favoritos
Excepciones	▪ No ha sido añadida ninguna ruta

Comentarios	-
Actores	Usuario registrado

Tabla 27 Requisito: Añadir/Eliminar ruta a lista de rutas favoritas

Requisito: Añadir /Eliminar comentario en ruta	
Identificador	4.5
Prioridad	Baja
Precondición	Hay una sesión iniciada, hay rutas propuestas
Descripción	Los usuarios pueden agregar comentarios a las rutas así como modificarlos o eliminarlos
Entrada	Comentario
Salida	-
Secuencia normal	1- El usuario accede al apartado de rutas 2- La aplicación muestra la lista de rutas 3- El usuario pulsa en una ruta 4- La aplicación muestra los datos de la ruta 5- El usuario pulsa el botón "Añadir" o "Eliminar" comentario 6- Se muestra el cuadro de comentario en el caso de "Añadir" para escribir el comentario a. El usuario pulsa el botón "Enviar comentario" 7- El comentario se añade o se elimina
Postcondición	Se ha añadido o eliminado un comentario a la ruta
Excepciones	▪ No ha sido añadida ninguna ruta ▪ El usuario no ha añadido ningún comentario
Comentarios	-
Actores	Usuario registrado

Tabla 28 Requisito: Añadir /Eliminar comentario en ruta

4.2.5 Gestión de Entrenadores

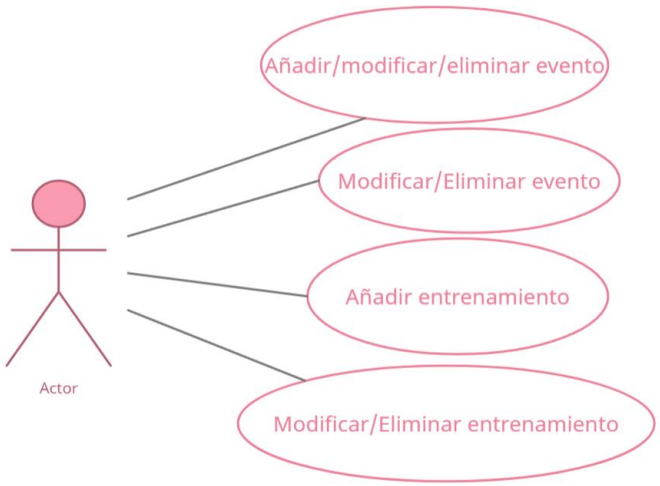

Figura 6 Diagrama de caso de uso Gestión de Entrenadores

Requisito: Añadir evento	
Identificador	5.1
Prioridad	Alta
Precondición	Hay una sesión iniciada, el usuario es Administrador
Descripción	Los administradores pueden añadir las carreras y competiciones a la lista de eventos así como editarlas o eliminarlas
Entrada	Nombre, Tipo, Fecha, Hora, Lugar, Kilómetros, Descripción
Salida	-
Secuencia normal	1- El usuario accede al "Panel Administración" 2- El usuario pulsa el botón "Añadir Evento" 3- La aplicación muestra una pantalla con los datos a rellenar 4- El usuario rellena los datos y pulsa el botón "Añadir"
Postcondición	Se ha añadido un nuevo evento
Excepciones	▪ El usuario no ha rellenado todos los datos ▪ Algún dato es incorrecto
Comentarios	-
Actores	Administrador

Tabla 29 Requisito: Añadir evento

Requisito: Modificar/Eliminar evento	
Identificador	5.2
Prioridad	Alta
Precondición	Hay una sesión iniciada, el usuario es Administrador, el usuario ha creado algún evento
Descripción	Los administradores pueden editar o eliminar las carreras y competiciones creadas previamente
Entrada	Nombre, Tipo, Fecha, Hora, Lugar, Kilómetros, Descripción
Salida	-
Secuencia normal	1- El usuario accede al "Panel Administración" 2- El usuario pulsa el botón "Eventos creados" 3- La aplicación muestra una pantalla con la lista de eventos creados 4- El usuario pulsa en un evento 5- El usuario pulsa el botón "Editar" o "Eliminar" 6- Si pulsa el botón "Editar": a. La aplicación muestra los datos con la posibilidad de modificarlos b. El usuario modifica los datos que desee c. El usuario pulsa el botón "Guardar"
Postcondición	Se ha modificado o eliminado un evento
Excepciones	▪ El usuario no ha rellenado todos los datos ▪ Algún dato es incorrecto ▪ El usuario no creó ningún evento
Comentarios	-
Actores	Administrador

Tabla 30 Requisito: Modificar/Eliminar evento

Requisito: Añadir entrenamiento	
Identificador	5.3
Prioridad	Alta
Precondición	Hay una sesión iniciada, el usuario es Administrador
Descripción	Los administradores pueden añadir los entrenamientos de los usuarios
Entrada	Fecha, Nombre Usuario
Salida	-
Secuencia normal	1- El usuario accede al "Panel Administración" 2- El usuario accede al apartado "Entrenamientos" 3- El usuario pulsa el botón "Añadir Entrenamiento" 4- La aplicación muestra una pantalla con los datos a rellenar 5- El usuario rellena los datos y pulsa el botón "Añadir"
Postcondición	Se ha añadido un entrenamiento
Excepciones	▪ El usuario no ha rellenado todos los datos ▪ Algún dato es incorrecto
Comentarios	-
Actores	Administrador

Tabla 31 Requisito: Añadir entrenamiento

Requisito: Modificar/Eliminar entrenamiento	
Identificador	5.4
Prioridad	Alta
Precondición	Hay una sesión iniciada, el usuario es Administrador, el usuario ha creado algún entrenamiento
Descripción	Los administradores pueden editar o eliminar los entrenamientos creados previamente
Entrada	Nombre, Tipo, Fecha, Hora, Lugar, Kilómetros, Descripción
Salida	-
Secuencia normal	1- El usuario accede al "Panel Administración" 2- El usuario accede al apartado "Entrenamientos" 3- La aplicación muestra una pantalla con la lista de entrenamientos creados 4- El usuario pulsa en un entrenamiento 5- El usuario pulsa el botón "Editar" o "Eliminar" 6- Si pulsa el botón "Editar": a. La aplicación muestra los datos con la posibilidad de modificarlos b. El usuario modifica los datos que desee c. El usuario pulsa el botón "Guardar"
Postcondición	Se ha modificado o eliminado un entrenamiento
Excepciones	▪ El usuario no ha rellenado todos los datos ▪ Algún dato es incorrecto ▪ El usuario no creó ningún entrenamiento
Comentarios	-
Actores	Administrador

Tabla 32 Requisito: Modificar/Eliminar entrenamiento

Capítulo 5 - Arquitectura de la aplicación

En este capítulo se detalla la arquitectura de la aplicación y el modelo de datos.

5.1 Estructura de la aplicación

El tipo de arquitectura utilizado para la aplicación es serverless (sin servidor) que, a pesar del nombre involucra servidores. Esta arquitectura permite crear y ejecutar aplicaciones y servicios sin tener que administrar infraestructura. Esto permite a los desarrolladores centrarse en el desarrollo del producto principal sin preocuparse por administrar y operar servidores.

En este proyecto se ha utilizado Firebase, que difiere de las bbdd tradicionales:

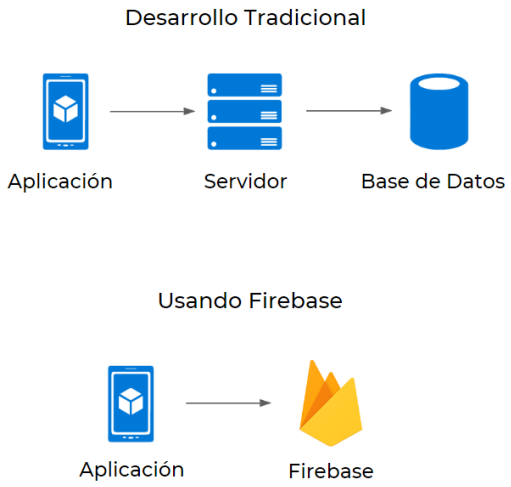

Figura 7 Comparación Desarrollo Tradicional y Firebase

Figura 8 Arquitectura Aplicación

El cliente (la aplicación) interactúa a través de la API con el servidor de Firebase, y este se encarga tanto de acceder a la base de datos mediante Firestore Firebase y al almacenamiento de imágenes mediante Firebase Storage. El cliente, a la hora de iniciar sesión y registrarse hace una petición desde la app y accede a través de Firebase Authentication.

5.2 Modelo de datos

Para la aplicación se necesitan almacenar datos sobre los distintos elementos que la componen, como pueden ser entrenamientos, eventos… También se necesitan almacenar imágenes para la sección de rutas.

Para almacenar los datos se ha utilizado la base de datos Firestore de Firebase, que es una base de datos NoSQL, lo que permite almacenar diferente información por cada tipo de objeto. Respecto a las imágenes, se ha utilizado el módulo Storage de Firebase, que permite almacenar elementos fotográficos.

La base de datos está organizada en colecciones, y estas contienen documentos. La información que contiene es la de los usuarios, los eventos las rutas y los entrenamientos grupales.

Figura 9 Modelo de datos Firebase Authentication

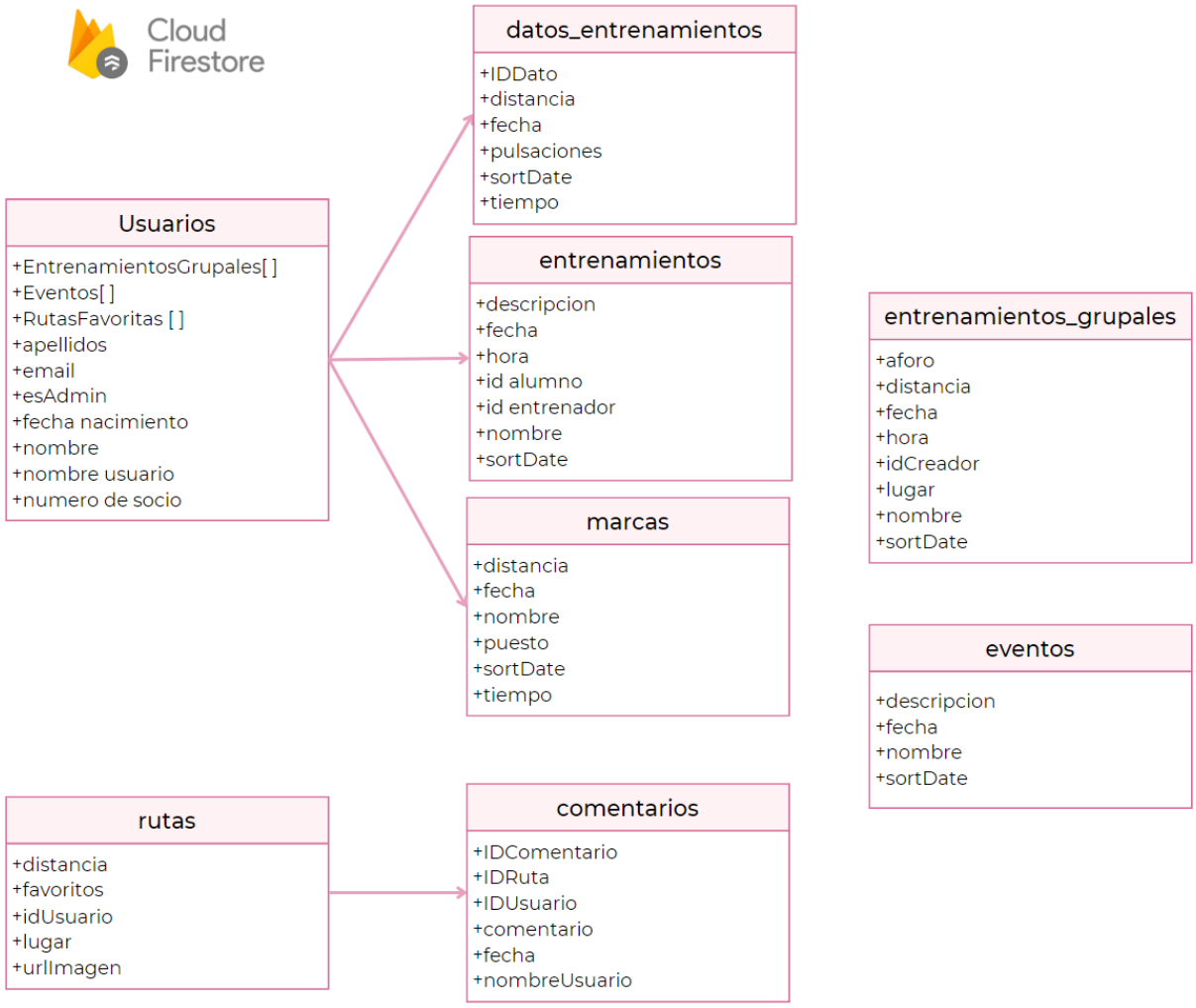

Figura 10 Modelo de datos Firebase Firestore

5.2.1 Colección Autenticación

Esta colección forma parte del módulo de Autenticación de Firebase, y se utiliza para identificar a los usuarios a través de un email y contraseña. Almacena el identificador de usuario, que es el email, los proveedores, la fecha de creación y de último acceso y el UID de usuario.

Identificador	Proveedores	Fecha de creación ↓	Fecha de acceso	UID de usuario
ana@gmail.com	✉	13 mar 2023	14 may 2023	1fNX671AMgPOWjDSSaea7OdZY...
lucia@gmail.com	✉	8 dic 2022	20 abr 2023	JN9DSQPQijOklws35mkWn93huc...
vi@gmail.com	✉	7 dic 2022	22 mar 2023	fmd2Qdpvb8fmMUWV84jMtTNI2o...

Figura 11 Colección Autenticación

5.2.2 Colección usuarios

La colección de usuarios está almacenada en Firestore Firebase, y almacena todos los datos de los usuarios de la aplicación: nombre de usuario, nombre, apellidos, email, fecha de nacimiento, número de socio, bool indicando si es administrador y mapas con los entrenamientos grupales y eventos a los que el usuario está apuntado, y las rutas favoritas. También contiene tres colecciones para almacenar los datos de entrenamientos, los entrenamientos y las marcas personales.

Figura 12 Colección Usuarios

5.2.2.1 Colección datos_entrenamientos

Esta colección está destinada a almacenar los datos de los entrenamientos para almacenar las estadísticas contenidos en estos. Los datos almacenados en la colección son el IDDato (para temas administrativos), la distancia recorrida en el entrenamiento, la fecha, las pulsaciones medias, sortDate (para temas administrativos) y el tiempo de duración del entrenamiento.

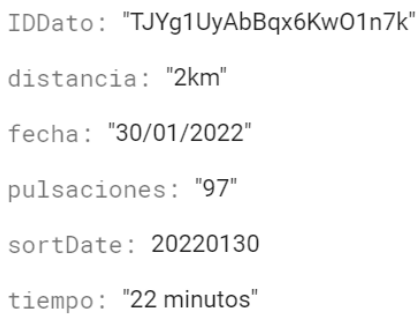

Figura 13 Colección datos_entrenamientos

5.2.2.2 Colección entrenamientos

La colección de entrenamientos está destinada a almacenar los entrenamientos de un usuario. Esta colección contiene como datos una descripción del entrenamiento, la fecha y la hora de este, los id del alumno y el entrenador que realizarán el entrenamiento, el nombre y sortDate (para temas administrativos).

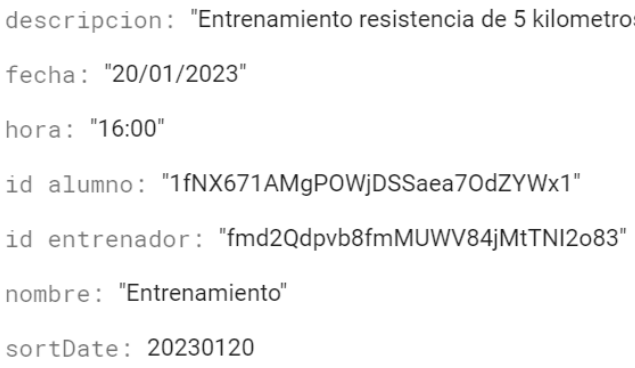

Figura 14 Colección entrenamientos

5.2.2.3 Colección marcas

Esta colección se utiliza para almacenar las marcas personales que el usuario obtiene en los eventos que ha participado. Los datos incluidos en la colección son la distancia del evento, la fecha en la que se realizó, el nombre del evento, el puesto en el que quedó el usuario, y el tiempo que tardó en realizar el recorrido.

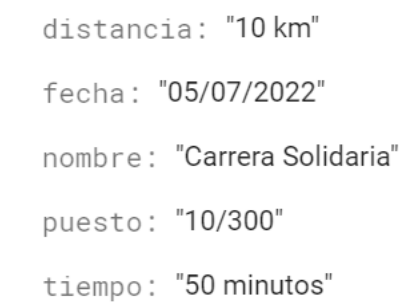

Figura 15 Colección marcas

5.2.3 Colección entrenamientos_grupales

En la colección de entrenamientos_grupales se guardan los entrenamientos grupales que publican los usuarios. Los datos que forman parte de los documentos son el aforo del entrenamiento, la distancia, la fecha y la hora en la que será realizado, el id del creador del entrenamiento grupal, el lugar en el que se entrenará, el nombre dado al entrenamiento y sortDate (para temas administrativos).

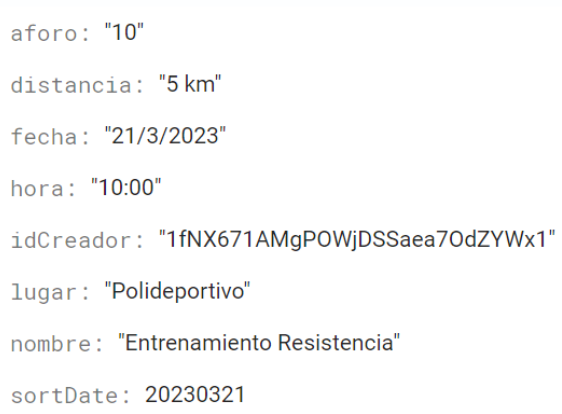

Figura 16 Colección entrenamientos_grupales

5.2.4 Colección eventos

En esta colección están almacenados los eventos que publican los entrenadores para que los miembros del club los vean y puedan apuntarse. Estos eventos pueden ser maratones, carreras, eventos solidarios, reuniones/quedadas de runners…

Los datos de los eventos son una descripción del evento, la distancia (si procede), la fecha, hora, y lugar en los que se realizará el evento, el nombre y tipo de evento y sortDate (para temas administrativos).

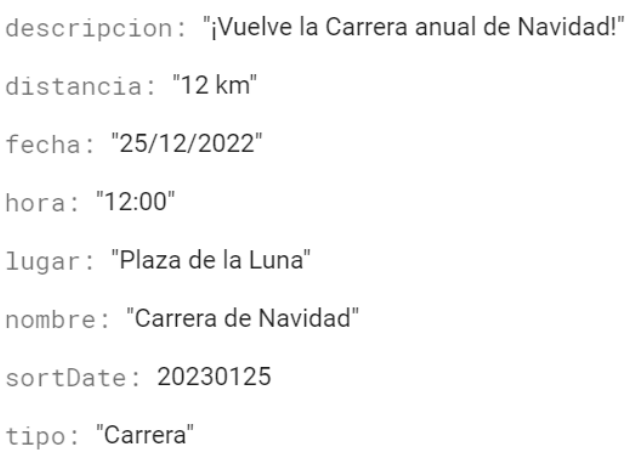

Figura 17 Colección eventos

5.2.5 Colección rutas

La función de la colección de rutas es guardar las rutas publicadas por los usuarios. En los documentos de esta colección están los siguientes datos: distancia de la ruta, número de personas que han añadido la ruta a favoritos, el id del usuario que ha publicado la ruta, el lugar en el que está la ruta, y por último la url de la imagen de la ruta, que indica la dirección en la que está almacenada en el módulo Firebase Storage de Firebase. La colección también tiene una subcolección para almacenar los comentarios de la ruta.

Figura 18 Colección Rutas

5.2.5.1 Colección Comentarios

Esta colección está dedicada a almacenar los comentarios que los usuarios realizan en las rutas. Los datos que contiene el documento son los identificadores del comentario, ruta y usuario que comenta (para temas administrativos), el comentario, la fecha en la que se realiza el comentario y el nombre de usuario comentador.

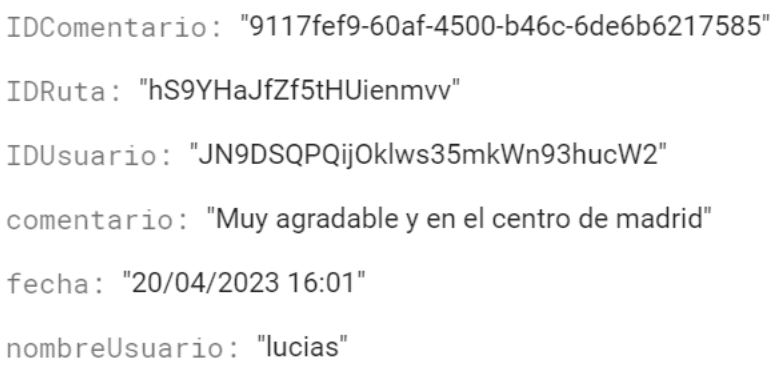

Figura 19 Colección comentarios

Capítulo 6 - Diseño

La interfaz de usuario es el entorno en el cual una persona puede manejar un software o hardware en particular. Una interfaz de usuario tiene que ser clara, atractiva, coherente e interactiva.

Los colores utilizados en el diseño de aplicación son principalmente blancos y rosados, con algún detalle morado. Los textos usan color blanco, negro o marrón oscuro dependiendo del fondo. El objetivo es utilizar pocos colores y específicos para crear cohesión en la aplicación y no saturar al usuario con muchos colores.

Las fuentes utilizadas son spartan, comfortaa y sans-serif (la fuente por defecto de Android Studio).

En cuanto a la estructura, se ha realizado con una barra de navegación inferior que contiene los módulos principales de la aplicación (Inicio, Eventos, Entrenamientos, Rutas y Perfil de usuario). Se han utilizado iconos representativos de cada módulo y el icono adquiere otro color cuando es seleccionado, para hacer la barra más visual. También aparece el nombre del módulo seleccionado para saber con más exactitud en que módulo se está y los elementos que contiene.

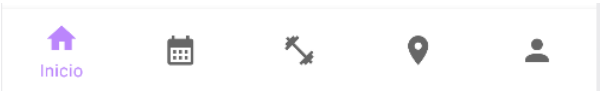

Figura 20 Barra de Navegación

En el fragmento de inicio se incluyeron los apartados relacionados con los usuarios. Se añadieron en forma de botones y consiste en los eventos en los que se ha apuntado, las marcas personales, los entrenamientos grupales publicados por los usuarios, y las estadísticas de los entrenamientos. En el caso de que el usuario tenga permiso de administración, que es el caso de los entrenadores, también aparece un botón para acceder al panel de administración. Se han creado los botones con un icono representativo de la sección además del nombre de esta para que sea más visual para el usuario.

Figura 21 Diseño Inicio

La navegación en los distintos apartados está estructurada en listas que contienen los datos básicos del objeto, y si se pulsa en un elemento de la lista se accede al elemento con todos los datos detallados. En la figura 22 se muestran los distintos botones utilizados en la aplicación

1- **Botón básico rosa**: para añadir o cancelar la agregación de elementos en formularios, y los botones de modificación o eliminación de elementos.

2- **Botón básico morado**: para seleccionar imágenes en formularios y para acceder a secciones de la aplicación que no forman parte de la barra de navegación ni el panel de inicio.

3- **Botón de añadir**: para acceder al formulario de elementos añadidos por el usuario como datos de entrenamiento, entrenamientos grupales, marcas personales y rutas.

4- **Papelera**: para borrar elementos añadidos por el usuario, como marcas personales, datos de entrenamiento, y comentarios.

5- **Corazón**: para añadir ruta a favorita. Cuando no es favorita aparece vacío y cuando está añadida relleno.

6- **Avión de papel**: el usuario pulsa este botón para añadir un comentario a una ruta.

7- **Flecha:** se utiliza para navegar hacia atrás en la aplicación.

Figura 22 Botones Aplicación

Por último, se utilizan dos tipos de fondos en la aplicación, mostrados en la Figura 23.

- Fondo 1: se utiliza para la sección de inicio de sesión y panel de administración, y para todos los formularios de añadir elementos.
- Fondo 2: se utiliza en el resto de la aplicación. También se utiliza en los formularios de modificar elemento en lugar del fondo 1, porque quedaba más cohesivo que el formulario de modificar elemento fuese similar a la pantalla de mostrar elemento.
- En las pantallas en las que hay muchas elementos y colores se utiliza un fondo blanco, ya que si no sería demasiado contenido y no cumplía la necesidad de que la interfaz de usuario sea clara y atractiva.

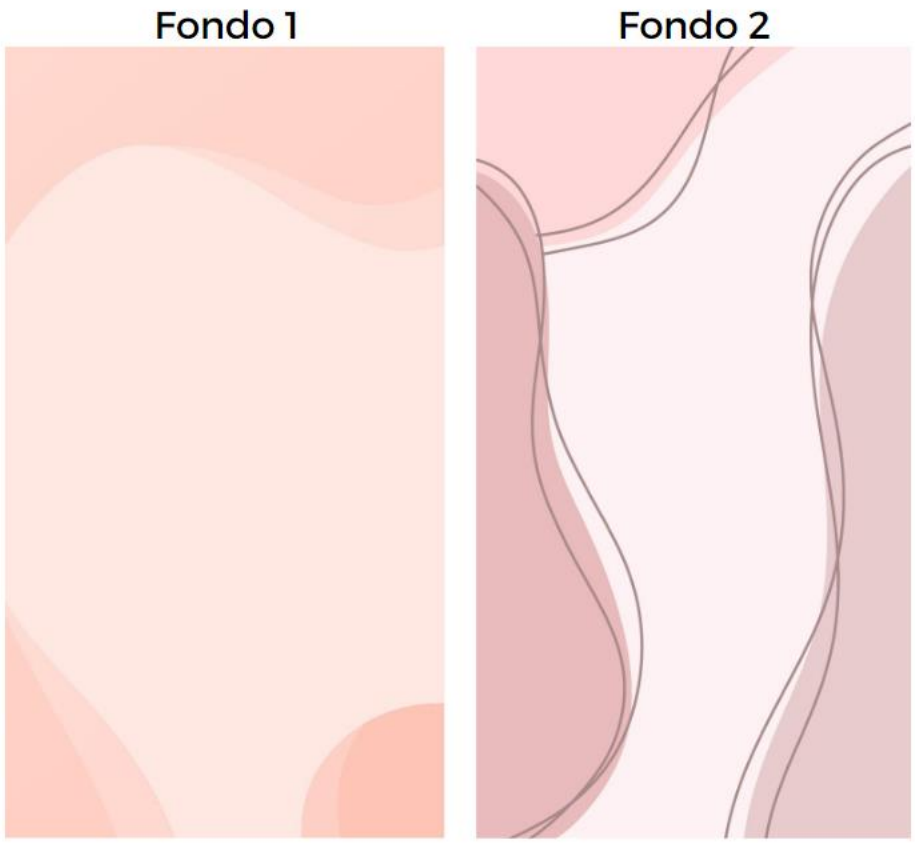

Figura 23 Fondos de la Aplicación

Capítulo 7 - Funcionalidades

En este capítulo se describen las funcionalidades de la aplicación divididas por módulos:

- Módulo usuarios
- Módulo entrenadores
- Módulo eventos
- Módulo entrenamientos
- Módulo rutas

7.1 Módulo usuarios

7.1.1 Iniciar sesión

La primera pantalla que aparece al iniciar la aplicación es la de inicio de sesión. Si el usuario tiene una cuenta debe introducir los datos y pulsar el botón de "Iniciar Sesión", y la aplicación revisa si los datos de email y contraseña están rellenos. Si no están rellenos manda un mensaje de aviso y si lo está realiza la autenticación con Firebase.

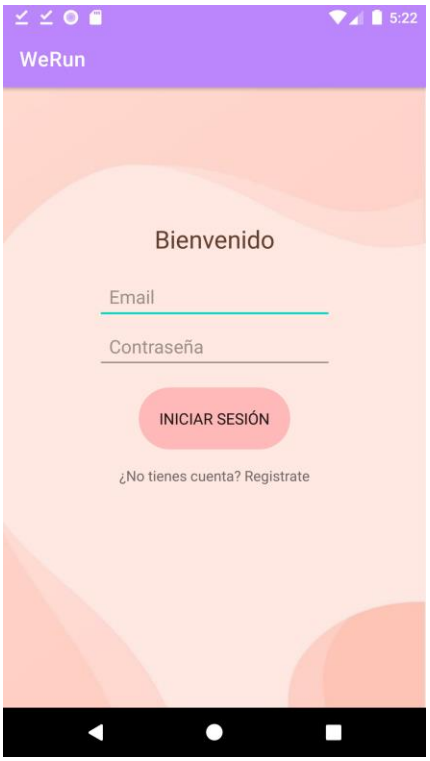

Figura 24 Funcionalidad Iniciar Sesión

```
iniciarSesionbtn_.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //revisa campos
        String email = email_.getText().toString().trim();
        String contrasena = contrasena_.getText().toString().trim();
        if (TextUtils.isEmpty(email) || TextUtils.isEmpty(contrasena)) {
            if (TextUtils.isEmpty(email))
                email_.setError("El email no puede estar vacío");
            if (TextUtils.isEmpty(contrasena))
                contrasena_.setError("La contraseña no puede estar vacía");
            return;
        }

        //inicio de la sesion
        autenticacion.signInWithEmailAndPassword(email, contrasena).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(task.isSuccessful()){
                    Toast.makeText( context: IniciarSesion.this,  text: "Sesión iniciada", Toast.LENGTH_SHORT).show();
                    startActivity(new Intent(getApplicationContext(), MainActivity.class));
                }
                else{
                    Toast.makeText( context: IniciarSesion.this,  text: "No se ha podido iniciar sesión"+task.getException().getMessage(), Toast.LENGTH_SHORT).sh
                }
            }
        });
    }
});
```

Figura 25 Código Funcionalidad Iniciar Sesión

7.1.2 Registro de usuario

Si el usuario no está registrado puede acceder a la pantalla de registrar, en la que aparecerá un formulario para crear su cuenta. El usuario deberá rellenar sus datos y pulsar el botón de "Registrarse". Una vez pulsado el botón, la aplicación revisara si todos los datos están rellenos, si no es así muestra una advertencia, y si los datos están rellenos procede a realizar el registro. Para ello, lo primero añade el usuario al módulo Autenticación de Firebase para que pueda iniciar sesión. Posteriormente, añade todos los datos del usuario a un mapa, y lo inserta en la base de datos de Firestore en la colección de usuarios.

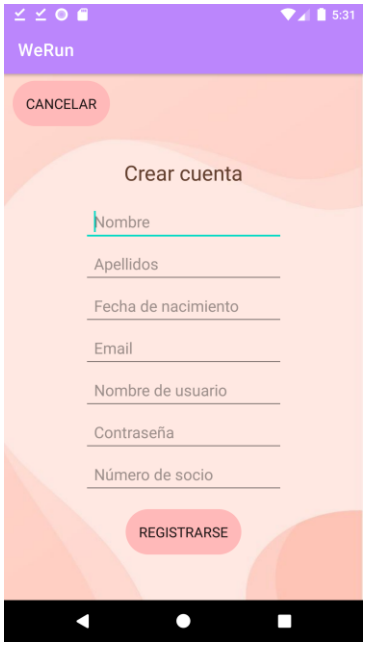

Figura 26 Funcionalidad Registro de usuario

```
autenticacion.createUserWithEmailAndPassword(email, contrasena).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            Toast.makeText( context: Registro.this,  text: "Usuario creado", Toast.LENGTH_SHORT).show();
            IDUsuario=autenticacion.getCurrentUser().getUid();
            DocumentReference doc=fstore.collection( collectionPath: "usuarios").document(IDUsuario);
            Map<String, Object> usuario=new HashMap<>();
            usuario.put( k: "email", email);
            usuario.put( k: "nombre", nombre);
            usuario.put( k: "apellidos", apellidos);
            usuario.put( k: "fecha nacimiento", fechaNacimiento);
            usuario.put( k: "nombre usuario", nombreUsuario);
            usuario.put( k: "numero de socio", numSocio);
            usuario.put( k: "esAdmin",  v: "0");
            Map<String, Object> map=new HashMap<>();
            usuario.put( k: "RutasFavoritas",map);
            usuario.put( k: "Eventos",map);
            usuario.put( k: "EntrenamientosGrupales",map);
            //insertar en la bbdd
            doc.set(usuario).addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void unused) {
                    Log.d(TAG,  msg: "onSuccess: se ha creado el perfil para"+IDUsuario);
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Log.d(TAG,  msg: "onFailure: No se ha podido crear el perfil"+e.toString());
                }
            });
            startActivity(new Intent(getApplicationContext(), MainActivity.class));
        }
        else{
            Toast.makeText( context: Registro.this,  text: "No se ha podido crear el usuario"+task.getException().getMessage(), Toast.LENGTH_SHORT).show()
        }
```

Figura 27 Código Funcionalidad Registro de usuario

7.1.3 Ver perfil

El usuario puede acceder a su perfil desde la barra de navegación y ver los datos de este, así como acceder a los botones para modificarlos y cerrar sesión. Para mostrar los datos se crean atributos para almacenar los elementos de la vista, y después de acceder al usuario en la base de datos establece los datos del usuario en los elementos del perfil.

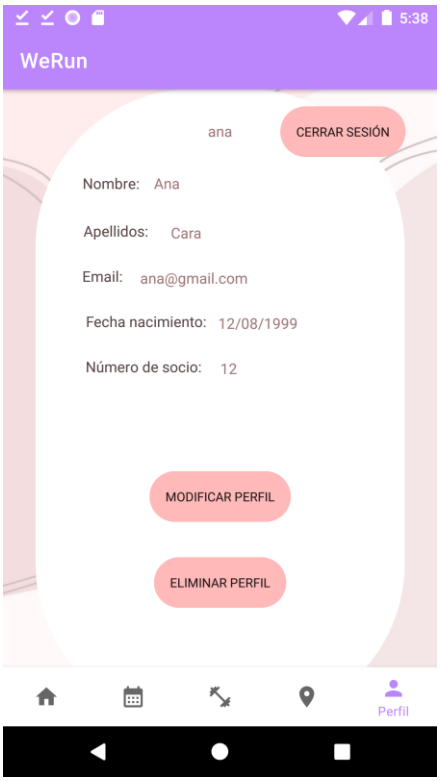

Figura 28 Funcionalidad Ver perfil

```
nombreUsuario=view.findViewById(R.id.nombreUsuarioPerfil);
nombre=view.findViewById(R.id.nombrePerfil);
apellidos=view.findViewById(R.id.apellidosPerfil);
email=view.findViewById(R.id.emailPerfil);
fechaNacimiento=view.findViewById(R.id.fechaNacPerfil);
numSocio=view.findViewById(R.id.numSocioPerfil);
modificarPerfilbtn=view.findViewById(R.id.modificarPerfilbtn);
eliminarPerfilbtn=view.findViewById(R.id.eliminarPerfilbtn);
cerrarSesionbtn=view.findViewById(R.id.cerrarSesionbtn);

autenticacion=FirebaseAuth.getInstance();
fStore=FirebaseFirestore.getInstance();
usuario=autenticacion.getCurrentUser();

IDUsuario=usuario.getUid();
DocumentReference doc=fStore.collection( collectionPath: "usuarios").document(IDUsuario);
doc.addSnapshotListener((documentSnapshot, e) -> {
    if(doc!=null &&documentSnapshot.exists()) { //revisamos que el documento sea valido
        nombreUsuario.setText(documentSnapshot.getString( field: "nombre usuario"));
        nombre.setText(documentSnapshot.getString( field: "nombre"));
        apellidos.setText(documentSnapshot.getString( field: "apellidos"));
        email.setText(documentSnapshot.getString( field: "email"));
        fechaNacimiento.setText(documentSnapshot.getString( field: "fecha nacimiento"));
        numSocio.setText(documentSnapshot.getString( field: "numero de socio"));

    }
});
```

Figura 29 Código Funcionalidad Ver perfil

7.1.4 Modificar datos perfil

Para acceder a esta funcionalidad se pulsa el botón "Modificar Perfil" en el fragmento perfil (Figura 28). Una vez en la actividad de modificar perfil, aparecen los datos del usuario con la posibilidad de cambiarlos. Si se decide no modificar el perfil se puede pulsar el botón "Cancelar" para volver al perfil, y si se quiere modificarlo, una vez cambiados los datos necesarios se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y si es así actualiza los datos en la base de datos.

Figura 30 Funcionalidad Modificar datos perfil

```
aceptarbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if(modNombre.getText().toString().isEmpty() || modApellidos.getText().toString().isEmpty() || modEmail.getText().toString().isEmpty() ||
            Toast.makeText( context: ModificarPerfil.this,  text: "No puedes dejar campos vacios", Toast.LENGTH_SHORT).show();
            return;
        }
        String email=modEmail.getText().toString();
        usuario.updateEmail(email).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                DocumentReference doc=fStore.collection( collectionPath: "usuarios").document(usuario.getUid());
                Map<String, Object> cambios=new HashMap<>();
                cambios.put( k: "nombre", modNombre.getText().toString());
                cambios.put( k: "apellidos", modApellidos.getText().toString());
                cambios.put( k: "email", email);
                cambios.put( k: "fecha nacimiento", modfechaNac.getText().toString());
                cambios.put( k: "numero de socio", modnumSocio.getText().toString());
                doc.update(cambios).addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void unused) {
                        Toast.makeText( context: ModificarPerfil.this,  text: "Los datos se han cambiado correctamente", Toast.LENGTH_SHORT).show();
                        //startActivity(new Intent(getApplicationContext(), PerfilFragment.class));
                        finish();
                    }
                });
                Toast.makeText( context: ModificarPerfil.this,  text: "El email se ha cambiado correctamente", Toast.LENGTH_SHORT).show();
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText( context: ModificarPerfil.this, e.getMessage(), Toast.LENGTH_SHORT).show();
            }
        });
    }
```

Figura 31 Código Funcionalidad Modificar datos perfil

7.1.5 Eliminar perfil

Para acceder a esta funcionalidad se pulsa el botón "Eliminar Perfil" en el fragmento perfil (Figura 28). Aparece un mensaje para confirmar la eliminación, si se pulsa "Cancelar" no se elimina, y si se pulsa "Eliminar" se accede a la base de datos y se elimina el perfil tanto del módulo de Autenticación de Firebase, como el documento con los datos del usuario en la colección "usuarios" de la base de datos Firestore.

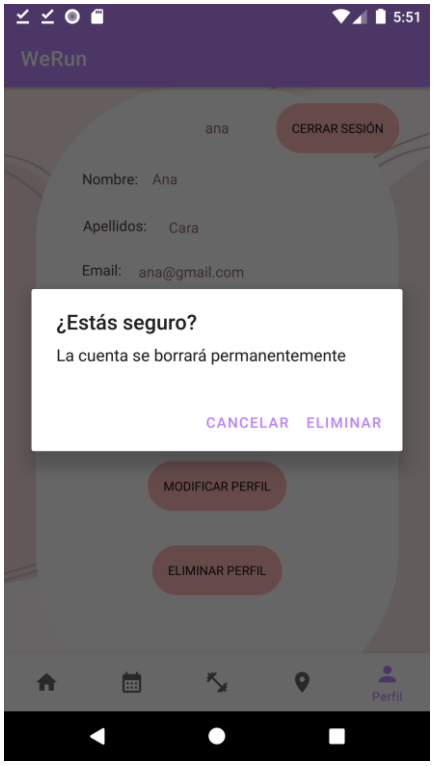

Figura 32 Funcionalidad Eliminar Perfil

```
eliminarPerfilbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder dialogo=new AlertDialog.Builder(getActivity());
        dialogo.setTitle("¿Estás seguro?");
        dialogo.setMessage("La cuenta se borrará permanentemente");
        dialogo.setPositiveButton( text: "Eliminar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                usuario.delete().addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if(task.isSuccessful()) {
                            doc.delete();
                            fStore.collection( collectionPath: "usuarios").document(IDUsuario).delete();
                            Toast.makeText(getActivity(),  text: "La cuenta ha sido eliminada", Toast.LENGTH_SHORT).show();
                            FirebaseAuth.getInstance().signOut();
                            Intent iniciar = new Intent(getActivity(), IniciarSesion.class);
                            startActivity(iniciar);
                            getActivity().finish();
                        }
                        else
                            Toast.makeText(getActivity(), task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                    }
                });
            }
        });
```

Figura 33 Código Funcionalidad Eliminar Perfil

7.1.6 Cerrar Sesión

Para acceder a esta funcionalidad se pulsa el botón "Cerrar Sesión" en el fragmento perfil (Figura 28). Se cierra la autenticación y se vuelve a la actividad de iniciar sesión.

```
cerrarSesionbtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        autenticacion.signOut();
        startActivity(new Intent(getActivity(),IniciarSesion.class));
        getActivity().finish();
    }
});
```

Figura 34 Código Funcionalidad Cerrar Sesión

7.2 Módulo entrenadores

Este módulo lo utilizan los entrenadores para añadir eventos y entrenamientos y la gestión de estos. Para añadir deben acceder al panel de administración (Figura 35), para ello hay que pulsar el botón "Panel Admin" que aparece en la Figura 21; este panel solo aparece si el usuario tiene cuenta de administrador.

Figura 35 Funcionalidad Panel Administración

7.2.1 Añadir evento

Para acceder a esta funcionalidad se pulsa el botón "Añadir Evento" en el panel de administración (Figura 35). Si se decide no añadir un evento se puede pulsar el botón "Cancelar" para volver al panel de administración, y si se quiere añadir, una vez añadidos los datos se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y si es así añade el documento a la base de datos.

Figura 36 Funcionalidad Añadir Evento

```
CollectionReference doc=fStore.collection( collectionPath: "eventos");
Map<String, Object> evento=new HashMap<>();
evento.put( k: "nombre", nombre);
evento.put( k: "tipo", tipo);
evento.put( k: "sortDate", fechaSort);
evento.put( k: "fecha", fecha);
evento.put( k: "hora", hora);
evento.put( k: "lugar", lugar);
evento.put( k: "distancia", distancia);
evento.put( k: "descripcion", descripcion);
//insertar en la bbdd
doc.add(evento);
finish();
```

Figura 37 Código Funcionalidad Añadir Evento

7.2.2 Modificar/Eliminar evento

Para acceder a estas funcionalidades hay que meterse en un evento. Si el usuario es administrador aparecerán los botones de "Modificar Evento" y "Eliminar Evento".

Figura 38 Funcionalidad Modificar/Eliminar Evento

7.2.2.1 Modificar evento

Para acceder a esta funcionalidad hay que pulsar el botón "Modificar Evento". Una vez en la actividad de modificar evento, aparecen los datos del evento con la posibilidad de cambiarlos. Si se decide no modificar el evento se puede pulsar el botón "Cancelar" para volver al evento, y si se quiere modificarlo, una vez cambiados los datos necesarios se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y si es así actualiza los datos en la base de datos.

Figura 39 Funcionalidad Modificar Evento

```
DocumentReference doc=fStore.collection( collectionPath: "eventos").document(id);
Map<String, Object> cambios=new HashMap<>();
cambios.put( k: "nombre", mNombre.getText().toString());
cambios.put( k: "tipo", mTipo.getText().toString());
cambios.put( k: "sortDate", fechaSort);
cambios.put( k: "fecha", fecha);
cambios.put( k: "hora", mHora.getText().toString());
cambios.put( k: "lugar", mLugar.getText().toString());
cambios.put( k: "distancia", mDistancia.getText().toString());
cambios.put( k: "descripcion", mDescripcion.getText().toString());
doc.update(cambios).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                Toast.makeText( context: ModificarEvento.this,  text: "Los datos se han cambiado correctamente", Toast.LENGTH_SHORT).show();
                finish();
            }
        });
```

Figura 40 Código Funcionalidad Modificar Evento

7.2.2.2 Eliminar Evento

Para acceder a esta funcionalidad hay que pulsar el botón "Eliminar Evento". Aparece un mensaje para confirmar la eliminación, si se pulsa "Cancelar" no se elimina, y si se pulsa "Eliminar" se accede a la base de datos y se elimina el documento con los datos del evento en la colección "eventos" de la base de datos Firestore.

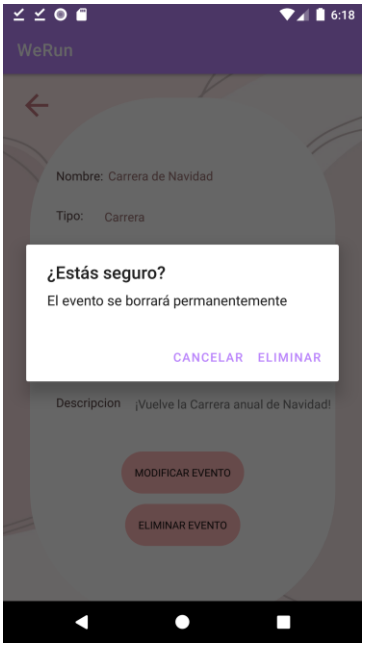

Figura 41 Funcionalidad Eliminar Evento

```
eliminarEventoBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder dialogo=new AlertDialog.Builder( context: Evento.this);
        dialogo.setTitle("¿Estás seguro?");
        dialogo.setMessage("El evento se borrará permanentemente");
        dialogo.setPositiveButton( text: "Eliminar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                doc.delete();
                fStore.collection( collectionPath: "eventos").document(id_).delete();
                Toast.makeText( context: Evento.this, text: "El evento ha sido eliminado", Toast.LENGTH_SHORT).show();
                finish();
            }
        });
        dialogo.setNegativeButton( text: "Cancelar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.dismiss();
            }
        });
        AlertDialog dialogoAlerta= dialogo.create();
        dialogoAlerta.show();
    }
});
```

Figura 42 Código Funcionalidad Eliminar Evento

7.2.3 Añadir entrenamiento

Para acceder a esta funcionalidad se pulsa el botón "Añadir Entrenamiento" en el panel de administración (Figura 35). Primero el entrenador tiene que elegir con que usuario realizará el entrenamiento. Una vez elegido se accede al formulario para añadir el entrenamiento. Si se decide no añadir un entrenamiento se puede pulsar el botón "Cancelar" para volver al panel de administración, y si se quiere añadir, una vez añadidos los datos se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y si es así añade el documento a la base de datos de ambos usuarios (el entrenador y el alumno).

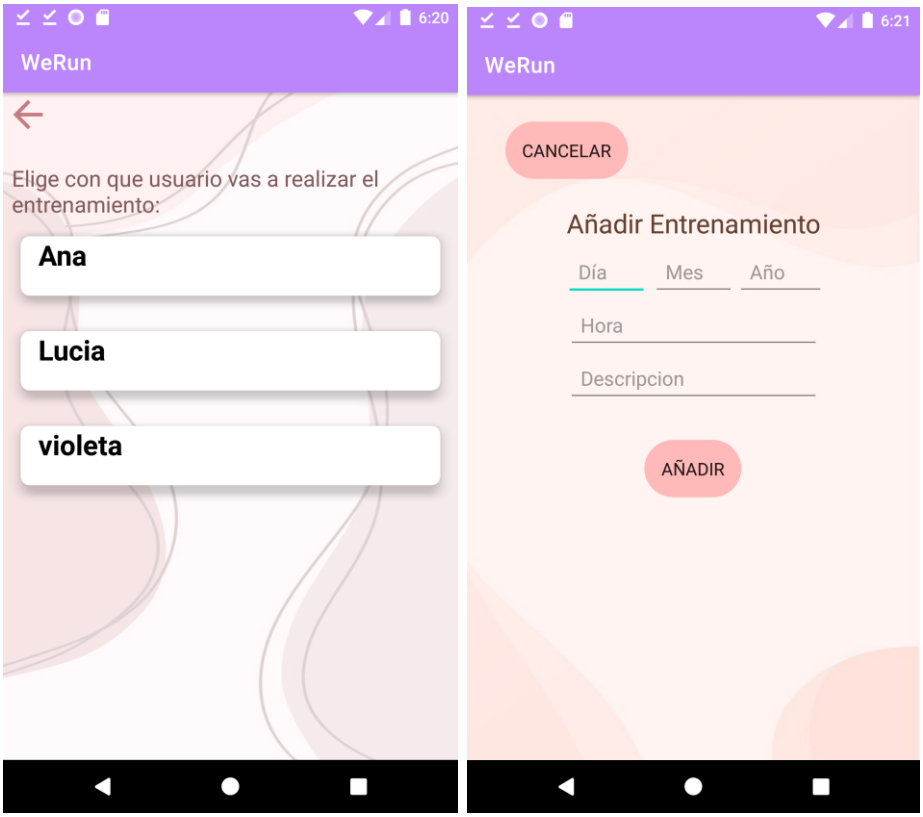

Figura 43 Funcionalidad Añadir Entrenamiento Elegir Usuario

```java
//para guardar la coleccion en los usuarios correspondientes
CollectionReference doc=fStore.collection( collectionPath: "usuarios").document(IDEntrenador).collection( collectionPath: "entrenamientos");
CollectionReference doc2=fStore.collection( collectionPath: "usuarios").document(IDAlumno).collection( collectionPath: "entrenamientos");

Map<String, Object> entrenamiento=new HashMap<>();
entrenamiento.put( k: "nombre",  v: "Entrenamiento");
entrenamiento.put( k: "sortDate", fechaSort);
entrenamiento.put( k: "fecha", fecha);
entrenamiento.put( k: "hora", hora);
entrenamiento.put( k: "descripcion", descripcion);
entrenamiento.put( k: "id alumno", IDAlumno);
entrenamiento.put( k: "id entrenador", IDEntrenador);
//insertar en la bbdd

String id = UUID.randomUUID().toString();
doc2.document(id).set(entrenamiento);
entrenamiento.put( k: "IDAlumnoDoc", id);
doc.add(entrenamiento);     //entrenamientos del entrenador
startActivity(new Intent(getApplicationContext(), PanelAdmin.class));
finish();
```

Figura 44 Código Funcionalidad Añadir Entrenamiento

7.2.4 Modificar/Eliminar entrenamiento

Para acceder a estas funcionalidades hay que meterse en un entrenamiento. Si el usuario es administrador aparecerán los botones de "Modificar Entrenamiento" y "Eliminar Entrenamiento".

Figura 45 Funcionalidad Modificar/Eliminar entrenamiento

7.2.4.1 Modificar Entrenamiento

Para acceder a esta funcionalidad hay que pulsar el botón "Modificar Entrenamiento". Una vez en la actividad de modificar entrenamiento, aparecen los datos del entrenamiento con la posibilidad de cambiarlos. Si se decide no modificar el entrenamiento se puede pulsar el botón "Cancelar" para volver al entrenamiento, y si se quiere modificarlo, una vez cambiados los datos necesarios se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y si es así actualiza los datos en la base de datos.

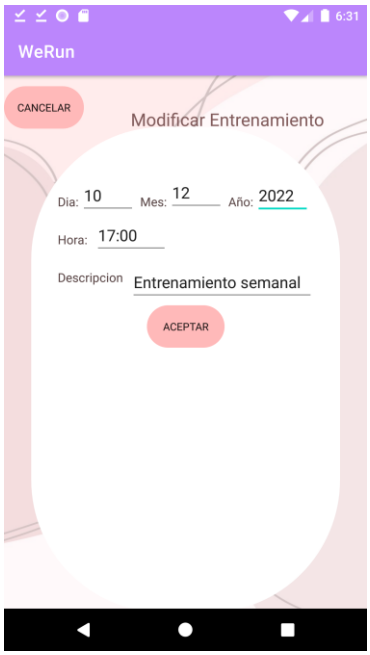

Figura 46 Funcionalidad Modificar Entrenamiento

```
DocumentReference doc= fStore.collection( collectionPath: "usuarios").document(IDEntrenador).collection( collectionPath: "entrenamientos").document(idDocEntrenador);
DocumentReference doc2= fStore.collection( collectionPath: "usuarios").document(IDAlumno).collection( collectionPath: "entrenamientos").document(idDocAlumno);

Map<String, Object> cambios=new HashMap<>();
cambios.put( k: "sortDate", fechaSort);
cambios.put( k: "fecha", fecha);
cambios.put( k: "hora", mHora.getText().toString());
cambios.put( k: "descripcion", mDescripcion.getText().toString());
doc.update(cambios).addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void unused) {
        Toast.makeText( context: ModificarEntrenamiento.this,  text: "Los datos se han cambiado correctamente en el documento entrenador", Toast.LENGTH_SHORT).sho
    }
});
doc2.update(cambios).addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void unused) {
        Toast.makeText( context: ModificarEntrenamiento.this,  text: "Los datos se han cambiado correctamente en el documento alumno", Toast.LENGTH_SHORT).show();
        finish();
    }
});
```

Figura 47 Código Funcionalidad Modificar Entrenamiento

7.2.4.2 Eliminar Entrenamiento

Para acceder a esta funcionalidad hay que pulsar el botón "Eliminar Entrenamiento", y aparece un mensaje para confirmar la eliminación, si se pulsa "Cancelar" no se elimina, y si se pulsa "Eliminar" se accede a la base de datos y se elimina el documento de la base de datos Firestore.

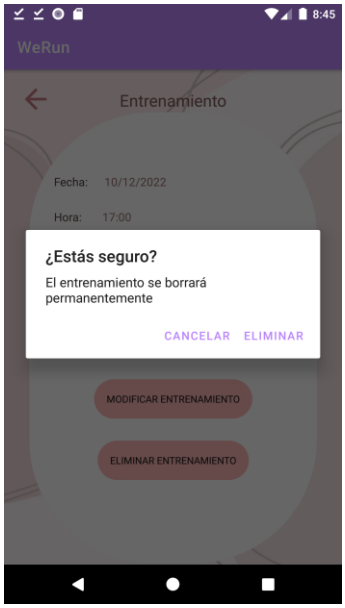

Figura 48 Funcionalidad Eliminar Entrenamiento

```
eliminarEntrenamientoBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder dialogo=new AlertDialog.Builder( context: Entrenamiento.this);
        dialogo.setTitle("¿Estás seguro?");
        dialogo.setMessage("El entrenamiento se borrará permanentemente");
        dialogo.setPositiveButton( text: "Eliminar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                doc.delete();
                String idA=IDAlumno.getText().toString();
                String idD=IDDoc.getText().toString();
                fStore.collection( collectionPath: "usuarios").document(IDUsuario).collection( collectionPath: "entrenamientos").document(id_).delete();
                fStore.collection( collectionPath: "usuarios").document(idA).collection( collectionPath: "entrenamientos").document(idD).delete();

                Toast.makeText( context: Entrenamiento.this, text: "El entrenamiento ha sido eliminado", Toast.LENGTH_SHORT).show();
                finish();

            }
        });
        dialogo.setNegativeButton( text: "Cancelar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.dismiss();
            }
        });
        AlertDialog dialogoAlerta= dialogo.create();
        dialogoAlerta.show();
    }
});
```

Figura 49 Código Funcionalidad Eliminar Entrenamiento

7.3 Módulo eventos

7.3.1 Ver lista de eventos

Para acceder a la lista de Eventos hay que pulsar el icono correspondiente en la barra de navegación de la aplicación. Con un adapter se recorre la lista y se muestra cada evento en una lista.

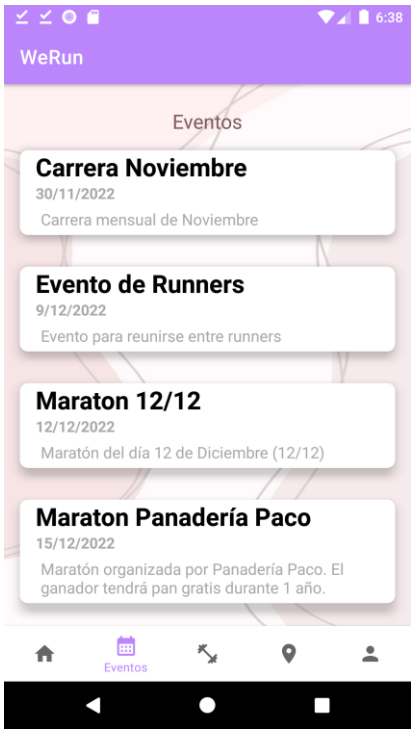

Figura 50 Funcionalidad Ver Lista de Eventos

```
fStore.collection( collectionPath: "eventos").orderBy( field: "sortDate", Query.Direction.ASCENDING).addSnapshotListener(new EventListener<QuerySnapshot>() {
    @Override
    public void onEvent(@Nullable QuerySnapshot value, @Nullable FirebaseFirestoreException error) {
        if(error!=null){
            Log.e( tag: "Error de Firestore en coleccion eventos", error.getMessage());
            return;
        }
        int i=0;
        for(DocumentChange docChange: value.getDocumentChanges()){  //se obtienen los datos de fireStore y se añaden a "lista"
            if(docChange.getType()==DocumentChange.Type.ADDED){
                lista.add(docChange.getDocument().toObject(ModelosEventoEntrenamiento.class));
                IDEventos[i]=docChange.getDocument().getId();
            }
            adapter.notifyDataSetChanged();
            i++;
        }
    }

});
```

Figura 51 Código Funcionalidad Ver Lista de Eventos

7.3.2 Apuntarse/Desapuntarse de evento

Para poder apuntarse o desapuntarse de un evento hay que acceder a un evento pulsando en uno en la lista de eventos.

El programa revisa si el usuario está o no apuntado y muestra el botón correspondiente. Sí no está apuntado y se quiere apuntar, se añade el id del evento al mapa de "Eventos" del usuario.

Si el usuario ya estaba apuntado y pulsa el botón de desapuntarse, el identificador del evento se elimina de la lista de eventos del usuario.

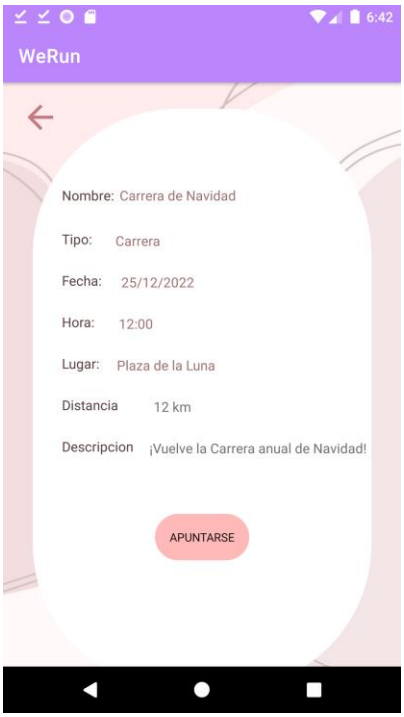

Figura 52 Funcionalidad Apuntarse/Desapuntarse de Evento

```
public void apuntarEvento(String idEvento, FirebaseAuth aut){
    IDUsuario=aut.getCurrentUser().getUid();
    fStore=FirebaseFirestore.getInstance();
    fStore.collection( collectionPath: "usuarios").document(IDUsuario).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            HashMap<String,String> eventosApuntados = (HashMap<String, String>) documentSnapshot.get("Eventos");
            Map<String, Object> eventos=new HashMap<>();
            String nombreEv= (String) nombre.getText();
            eventosApuntados.put(idEvento, nombreEv);
            eventos.put( k: "Eventos", eventosApuntados);
            fStore.collection( collectionPath: "usuarios").document(IDUsuario).update(eventos);
            estaApuntado=true;
            apuntarEventoBtn.setText("Desapuntarse");
        }
    });
}

public void desapuntarEvento(String idEvento, FirebaseAuth aut){
    IDUsuario=aut.getCurrentUser().getUid();
    fStore=FirebaseFirestore.getInstance();
    fStore.collection( collectionPath: "usuarios").document(IDUsuario).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            HashMap<String,String> eventosApuntados = (HashMap<String, String>) documentSnapshot.get("Eventos");
            Map<String, Object> eventos=new HashMap<>();
            if(eventosApuntados.containsKey(idEvento)){
                eventosApuntados.remove(idEvento);
            }
            eventos.put( k: "Eventos", eventosApuntados);
            fStore.collection( collectionPath: "usuarios").document(IDUsuario).update(eventos);
            estaApuntado=false;
            apuntarEventoBtn.setText("Apuntarse");
        }
    });
```

Figura 53 Código Funcionalidad Apuntarse/Desapuntarse de Evento

7.3.3 Ver lista de eventos apuntados

Para acceder a esta lista hay que pulsar el botón "Eventos Apuntados" que aparece en la Figura 21. El programa recorre el mapa de "Eventos" del documento del usuario y los añade a una lista, si no hay ningún evento en el mapa la actividad muestra un mensaje notificando que no está apuntado a ningún evento, y en el caso de que si haya eventos muestra la lista con ayuda de un adapter.

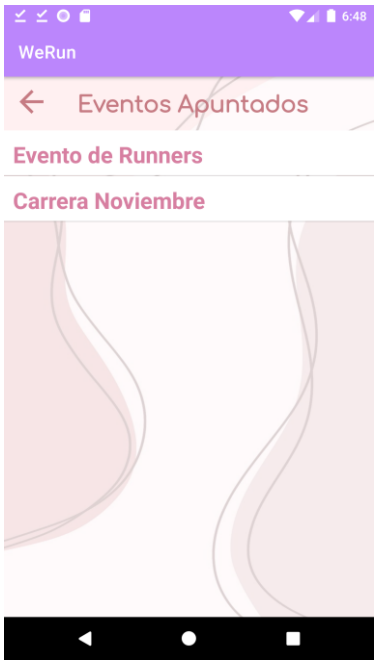

Figura 54 Funcionalidad Ver Lista de Eventos Apuntados

```
for(String ev:eventosApMap.values()){
    String nombreEvento = ev.replace( target: "_", replacement: " ");
    listaEventos.add(nombreEvento);
    //mapaEventos.put(nombreEvento, ev);
}
if(listaEventos.isEmpty()){
    ningunApuntado.setVisibility(View.VISIBLE);
}
else {
    int i=0;
    for(String ev:eventosApMap.keySet()){
        IDEventos[i]=ev.replace( target: "_", replacement: " ");
        i++;
    }
    lW_eventosApuntados = findViewById(R.id.listView_eventosApuntados);
    ArrayAdapter adapter_eventosApuntados = new ArrayAdapter(getApplicationContext(), R.layout.elemento_lista_sencillo, listaEventos);
    lW_eventosApuntados.setAdapter(adapter_eventosApuntados);

    lW_eventosApuntados.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            Intent i = new Intent(getApplicationContext(), Evento.class);
            i.putExtra( name: "IDEvento", IDEventos[position]);
            startActivity(i);
        }
    });
}
```

Figura 55 Código Funcionalidad Ver Lista de Eventos Apuntados

7.3.4 Ver estadísticas obtenidas en eventos

Para acceder a esta lista hay que pulsar el botón "Marcas Personales" que aparece en la Figura 21. El programa recorre el mapa de "Marcas" del documento del usuario y las añade a una lista, y muestra la lista con ayuda de un adapter.

Figura 56 Funcionalidad Ver estadísticas obtenidas en eventos

```
for(DocumentChange docChange: value.getDocumentChanges()){  //se obtienen los datos de fireStore y se añaden a "lista"
    if(docChange.getType()==DocumentChange.Type.ADDED){
        listaMarcas.add((String) docChange.getDocument().get("nombre"));
        IDMarcas[i]=docChange.getDocument().getId();
    }
    i++;
}

lW_marcasPersonales = findViewById(R.id.listView_marcasPersonales);
ArrayAdapter adapter_marcasPersonales = new ArrayAdapter(getApplicationContext(), R.layout.elemento_lista_sencillo, listaMarcas);
lW_marcasPersonales.setAdapter(adapter_marcasPersonales);

lW_marcasPersonales.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Intent i = new Intent(getApplicationContext(), MarcaPersonal.class);
        i.putExtra( name: "IDMarca", IDMarcas[position]);
        startActivity(i);
    }
});
```

Figura 57 Código Funcionalidad Ver estadísticas obtenidas en eventos

7.3.5 Añadir marca personal

Para acceder a esta lista hay que pulsar el botón de añadir (icono +). Aparecerá un formulario para añadir la marca personal. Si se decide no añadir la marca se puede pulsar el botón "Cancelar", y si se quiere añadir, una vez añadidos los datos se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y si es así añade el documento a la colección "marcas" de la base de datos del usuario.

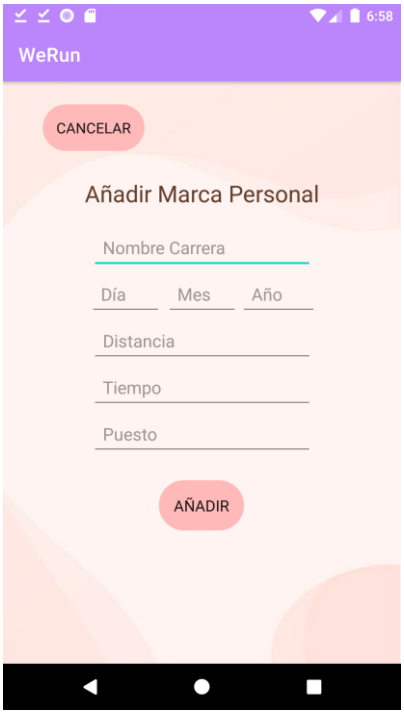

Figura 58 Funcionalidad Añadir Marca Personal

```
CollectionReference doc=fStore.collection( collectionPath: "usuarios").document(IDUsuario).collection( collectionPath: "marcas");
Map<String, Object> marca=new HashMap<>();
marca.put( k: "nombre", nombre);
marca.put( k: "sortDate", fechaSort);
marca.put( k: "fecha", fecha);
marca.put( k: "distancia", distancia);
marca.put( k: "tiempo", tiempo);
marca.put( k: "puesto", puesto);

//insertar en la bbdd
doc.add(marca);
```

Figura 59 Código Funcionalidad Añadir Marca Personal

7.3.6 Eliminar estadística

Para ello hay que acceder a una marca personal a través de la lista de marcas personales. Se pulsa el botón de la papelera y aparece un mensaje para confirmar la eliminación, si se pulsa "Cancelar" no se elimina, y si se pulsa "Eliminar" se accede a la base de datos y se elimina el documento con los datos del evento en la colección "marcas" de la colección del usuario en base de datos Firestore.

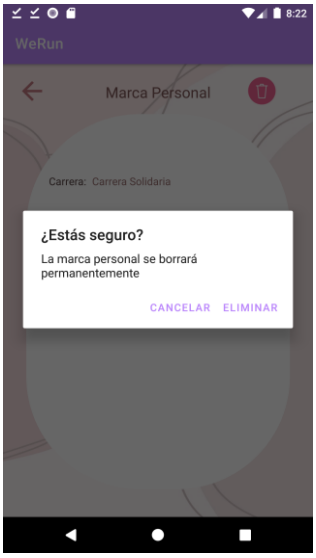

Figura 60 Funcionalidad Eliminar Estadística

```
eliminarMarcaBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder dialogo=new AlertDialog.Builder( context: MarcaPersonal.this);
        dialogo.setTitle("¿Estás seguro?");
        dialogo.setMessage("La marca personal se borrará permanentemente");
        dialogo.setPositiveButton( text: "Eliminar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                doc.delete();
                fStore.collection( collectionPath: "usuarios").document(IDUsuario).collection( collectionPath: "marcas").document(id_).delete();
                Toast.makeText( context: MarcaPersonal.this,  text: "La marca personal ha sido eliminado", Toast.LENGTH_SHORT).show();
                finish();
                Intent inte = new Intent(getApplicationContext(), MarcasPersonales.class);
                startActivity(inte);
            }
        });
        dialogo.setNegativeButton( text: "Cancelar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.dismiss();
            }
        });
        AlertDialog dialogoAlerta= dialogo.create();
        dialogoAlerta.show();
    }
});
```

Figura 61 Código Funcionalidad Eliminar Estadística

7.4 Módulo entrenamientos

7.4.1 Ver lista de entrenamientos

Para acceder a la lista de entrenamientos hay que pulsar el icono correspondiente en la barra de navegación de la aplicación. Con un adapter se recorre la lista y se muestra cada entrenamiento en una lista.

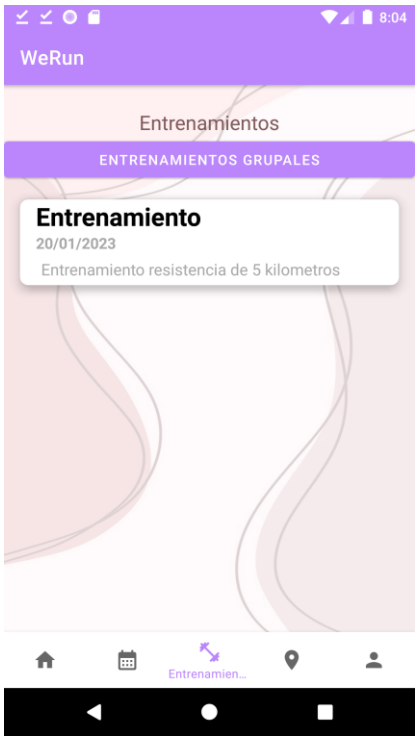

Figura 62 Funcionalidad Ver lista de entrenamientos

```
fStore.collection( collectionPath: "usuarios").document(IDUsuario).collection( collectionPath: "entrenamientos").orderBy( field: "sortDate", Query.Direction.ASCENDING)
.addSnapshotListener(new EventListener<QuerySnapshot>() {
    @Override
    public void onEvent(@Nullable QuerySnapshot value, @Nullable FirebaseFirestoreException error) {
        if(error!=null){
            Log.e( tag: "Error de Firestore en coleccion entrenamientos", error.getMessage());
            return;
        }
        int i=0;
        for(DocumentChange docChange: value.getDocumentChanges()){  //se obtienen los datos de fireStore y se añaden a "lista"
            if(docChange.getType()==DocumentChange.Type.ADDED){
                lista.add(docChange.getDocument().toObject(ModelosEventoEntrenamiento.class));
                IDEntrenamientos[i]=docChange.getDocument().getId();
            }
            adapter.notifyDataSetChanged();
            i++;
        }
    }
});
```

Figura 63 Código Funcionalidad Ver lista de entrenamientos

7.4.2 Ver pautas de entrenamiento

Para ver los datos del entrenamiento hay que pulsar en él en la lista de entrenamientos. El programa accede al entrenamiento en la base de datos y pone los datos en los campos de texto de la aplicación.

Figura 64 Funcionalidad Ver pautas de entrenamiento

```
DocumentReference doc=fStore.collection( collectionPath: "usuarios").document(IDUsuario).collection( collectionPath: "entrenamientos").document(id_);

doc.addSnapshotListener((documentSnapshot, e) -> {
    if(doc!=null &&documentSnapshot.exists()) { //revisamos que el documento sea valido
        IDDoc.setText(documentSnapshot.getString( field: "IDAlumnoDoc"));
        IDAlumno.setText(documentSnapshot.getString( field: "id alumno"));
        fecha.setText(documentSnapshot.getString( field: "fecha"));
        hora.setText(documentSnapshot.getString( field: "hora"));
        descripcion.setText(documentSnapshot.getString( field: "descripcion"));
    }
});
```

Figura 65 Código Funcionalidad Ver pautas de entrenamiento

7.4.3 Ver entrenamientos grupales

Para acceder a la lista de entrenamientos grupales publicados hay que pulsar en el botón "Ent Grupales" de la Figura 21. El programa recorre la colección de

entrenamientos grupales y los añade a una lista, y muestra la lista con ayuda de un adapter.

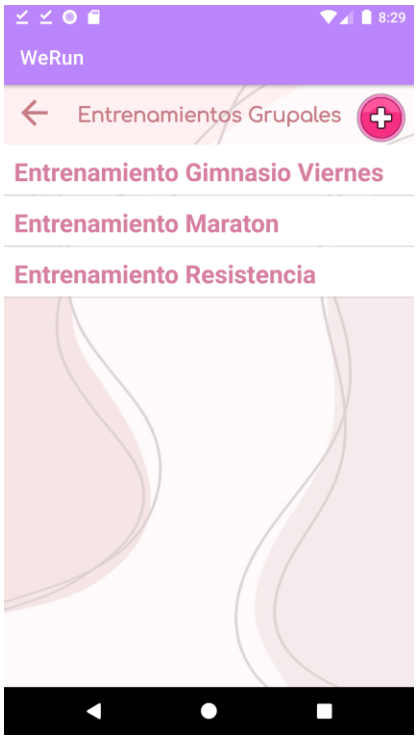

Figura 66 Funcionalidad Ver entrenamientos grupales

```
for(DocumentChange docChange: value.getDocumentChanges()){  //se obtienen los datos de fireStore y se añaden a "lista"
    if(docChange.getType()==DocumentChange.Type.ADDED){
        listaEntrenamientosGrupales.add((String) docChange.getDocument().get("nombre"));
        IDEntrenamientosGrupales[i]=docChange.getDocument().getId();
    }
    //adapter.notifyDataSetChanged();
    i++;
}
lW_entrenamientosGrupales = findViewById(R.id.listView_entrenamientosGrupales);
ArrayAdapter adapter_entrenamientosGrupales = new ArrayAdapter(getApplicationContext(), R.layout.elemento_lista_sencillo, listaEntrenamientosGrup
lW_entrenamientosGrupales.setAdapter(adapter_entrenamientosGrupales);
```

Figura 67 Código Funcionalidad Ver entrenamientos grupales

7.4.4 Ver entrenamientos grupales apuntado

Para acceder a la lista de entrenamientos grupales en los que el usuario está apuntado, hay que pulsar en el botón "Entrenamientos Grupales" de la Figura 60. El programa recorre el mapa de "EntrenamientosGrupales" del documento del usuario y los añade a una lista, si no hay ningún entrenamiento grupal en el mapa la actividad muestra un mensaje notificando que no está apuntado a ningún evento, y en el caso de que si haya eventos muestra la lista con ayuda de un adapter.

Figura 68 Funcionalidad Ver entrenamientos grupales apuntado

```
Map<String,String> entGrupApMap = new HashMap<>((HashMap<String,String>) document.getData().get("EntrenamientosGrupales"));
for(String eg:entGrupApMap.values()){
    String nombreEntGrup = eg.replace( target: "_", replacement: " ");
    listaEntGrup.add(nombreEntGrup);
}
if(listaEntGrup.isEmpty()){
    ningunApuntado.setVisibility(View.VISIBLE);
}
else {
    int i=0;
    for(String eg:entGrupApMap.keySet()){
        IDEntGrup[i]=eg.replace( target: "_", replacement: " ");
        i++;
    }
    lW_entGrupApuntados = findViewById(R.id.listView_entrenamientosGrupalesApuntados);
    ArrayAdapter adapter_entGrupApuntados = new ArrayAdapter(getApplicationContext(), R.layout.elemento_lista_sencillo, listaEntGrup);
    lW_entGrupApuntados.setAdapter(adapter_entGrupApuntados);
```

Figura 69 Código Funcionalidad Ver entrenamientos grupales apuntado

7.4.5 Crear entrenamiento grupal

Para acceder a esta funcionalidad hay que acceder a la lista de entrenamientos grupales (Figura 64) y pulsar el botón de añadir (icono de +). Aparecerá un formulario para añadir el entrenamiento grupal. Si se decide no añadir el entrenamiento se puede pulsar el botón "Cancelar", y si se quiere añadir, una vez añadidos los datos se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y si es así añade el documento a la colección "entrenamientos_grupales" de la base de datos de Firebase.

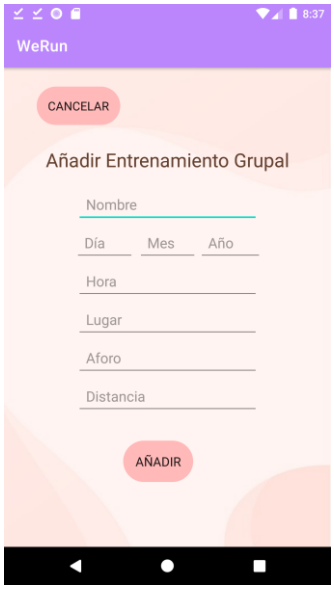

Figura 70 Funcionalidad Crear entrenamiento grupal

```
CollectionReference doc=fStore.collection( collectionPath: "entrenamientos_grupales");
Map<String, Object> entrenamientoGrupal=new HashMap<>();
entrenamientoGrupal.put( k: "nombre", nombre);
entrenamientoGrupal.put( k: "sortDate", fechaSort);
entrenamientoGrupal.put( k: "fecha", fecha);
entrenamientoGrupal.put( k: "hora", hora);
entrenamientoGrupal.put( k: "lugar", lugar);
entrenamientoGrupal.put( k: "aforo", aforo);
entrenamientoGrupal.put( k: "distancia", distancia);
entrenamientoGrupal.put( k: "idCreador", IDUsuario);
//insertar en la bbdd
doc.add(entrenamientoGrupal);
```

Figura 71 Código Funcionalidad Crear entrenamiento grupal

7.4.6 Modificar/Eliminar entrenamiento grupal

Para realizar estas funciones hay que acceder a un entrenamiento grupal que se haya creado, a través de la lista de entrenamientos grupales. Si el usuario ha creado el entrenamiento grupal aparecerán los botones de "Modificar Entrenamiento Grupal" y "Eliminar Entrenamiento Grupal".

Figura 72 Funcionalidad Modificar/Eliminar entrenamiento grupal

7.4.6.1 Modificar entrenamiento grupal

Para acceder a esta funcionalidad hay que pulsar el botón "Modificar Entrenamiento Grupal". Una vez en la actividad de modificar entrenamiento grupal, aparecen los datos del entrenamiento con la posibilidad de cambiarlos. Si se decide no modificar el entrenamiento se puede pulsar el botón "Cancelar" y si se quiere modificar, una vez cambiados los datos necesarios se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y si es así actualiza los datos en la base de datos.

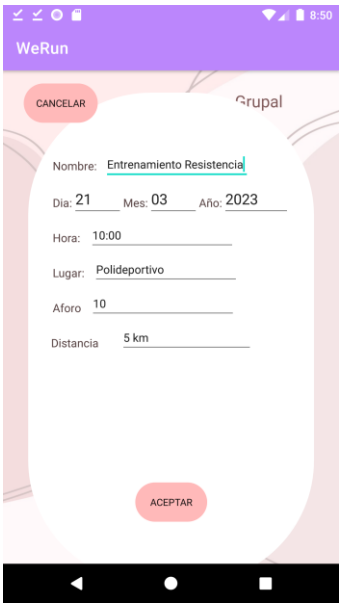

Figura 73 Funcionalidad Modificar entrenamiento grupal

```
DocumentReference doc=fStore.collection( collectionPath: "entrenamientos_grupales").document(id);
Map<String, Object> cambios=new HashMap<>();
cambios.put( k: "nombre", mNombre.getText().toString());
cambios.put( k: "sortDate", fechaSort);
cambios.put( k: "fecha", fecha);
cambios.put( k: "hora", mHora.getText().toString());
cambios.put( k: "lugar", mLugar.getText().toString());
cambios.put( k: "aforo", mAforo.getText().toString());
cambios.put( k: "distancia", mDistancia.getText().toString());
doc.update(cambios).addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void unused) {
        Toast.makeText( context: ModificarEntrenamientoGrupal.this,  text: "Los datos se han cambiado correctamente", Toast.LENGTH_SHORT).show();
        finish();
    }
});
```

Figura 74 Código Funcionalidad Modificar entrenamiento grupal

7.4.6.2 Eliminar entrenamiento grupal

Para acceder a esta funcionalidad hay que pulsar el botón "Eliminar Entrenamiento Grupal", y aparece un mensaje para confirmar la eliminación, si se pulsa "Cancelar" no se elimina, y si se pulsa "Eliminar" se accede a la base de datos y se elimina el documento de la base de datos Firestore.

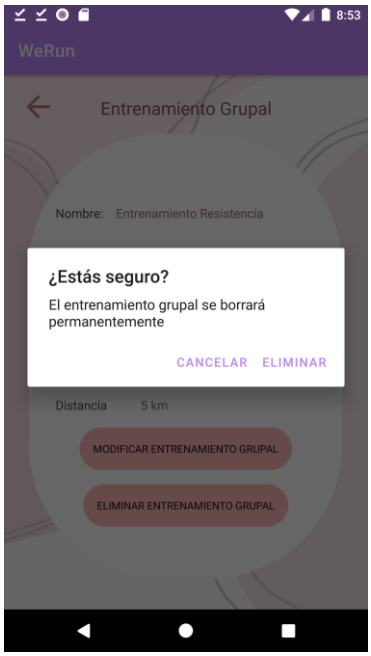

Figura 75 Funcionalidad Eliminar entrenamiento grupal

```
eliminarEntrenamientoGrupalBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder dialogo=new AlertDialog.Builder( context: EntrenamientoGrupal.this);
        dialogo.setTitle("¿Estás seguro?");
        dialogo.setMessage("El entrenamiento grupal se borrará permanentemente");
        dialogo.setPositiveButton( text: "Eliminar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                doc.delete();
                fStore.collection( collectionPath: "entrenamientos_grupales").document(id_).delete();
                Toast.makeText( context: EntrenamientoGrupal.this,  text: "El entrenamiento grupal ha sido eliminado", Toast.LENGTH_SHORT).show();
                finish();
                startActivity(new Intent(getApplicationContext(), EntrenamientosGrupales.class));

            }
        });
        dialogo.setNegativeButton( text: "Cancelar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.dismiss();
            }
        });
        AlertDialog dialogoAlerta= dialogo.create();
        dialogoAlerta.show();
    }
});
```

Figura 76 Código Funcionalidad Eliminar entrenamiento grupal

7.4.7 Apuntarse/Desapuntarse de entrenamiento grupal

Para poder apuntarse o desapuntarse de un entrenamiento grupal hay que acceder a un entrenamiento grupal pulsando en uno en la lista de entrenamientos grupales.

El programa revisa si el usuario está o no apuntado y muestra el botón correspondiente. Sí no está apuntado y se quiere apuntar, se añade el id del entrenamiento grupal al mapa de "EntrenamientosGrupales" del usuario.

Si el usuario ya estaba apuntado y pulsa el botón de desapuntarse, el identificador del entrenamiento grupal se elimina de la lista de entrenamientos grupales del usuario.

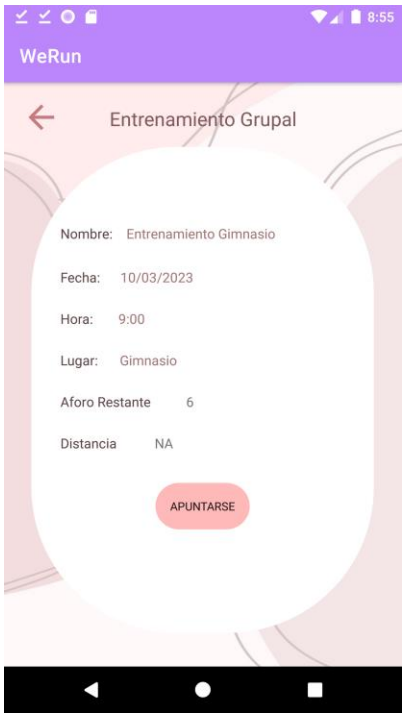

Figura 77 Funcionalidad Apuntarse/Desapuntarse de entrenamiento grupal

```
public void apuntarEntrenamientoGrupal(String idEntrenamientoGrupal, FirebaseAuth aut){
    IDUsuario=aut.getCurrentUser().getUid();
    fStore=FirebaseFirestore.getInstance();
    fStore.collection( collectionPath: "usuarios").document(IDUsuario).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            HashMap<String,String> entrenamientosApuntados = (HashMap<String, String>) documentSnapshot.get("EntrenamientosGrupales");
            Map<String, Object> entrenamientos=new HashMap<>();
            String nombreEv= (String) nombre.getText();
            entrenamientosApuntados.put(idEntrenamientoGrupal, nombreEv);
            entrenamientos.put( k: "EntrenamientosGrupales", entrenamientosApuntados);
            fStore.collection( collectionPath: "usuarios").document(IDUsuario).update(entrenamientos);
            estaApuntado=true;
            apuntarEntrenamientoGrupalBtn.setText("Desapuntarse");
        }
    });
}

public void desapuntarEntrenamientoGrupal(String idEntrenamientoGrupal, FirebaseAuth aut){
    IDUsuario=aut.getCurrentUser().getUid();
    fStore=FirebaseFirestore.getInstance();
    fStore.collection( collectionPath: "usuarios").document(IDUsuario).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            HashMap<String,String> entrenamientosGrupalesApuntados = (HashMap<String, String>) documentSnapshot.get("EntrenamientosGrupales");
            Map<String, Object> entrenamientos=new HashMap<>();
            if(entrenamientosGrupalesApuntados.containsKey(idEntrenamientoGrupal)){
                entrenamientosGrupalesApuntados.remove(idEntrenamientoGrupal);
            }
            entrenamientos.put( k: "EntrenamientosGrupales", entrenamientosGrupalesApuntados);
            fStore.collection( collectionPath: "usuarios").document(IDUsuario).update(entrenamientos);
            estaApuntado=false;
            apuntarEntrenamientoGrupalBtn.setText("Apuntarse");
        }
```

Figura 78 Código Funcionalidad Apuntarse/Desapuntarse de entrenamiento grupal

7.4.8 Ver estadísticas obtenidas en entrenamientos

Para acceder a esta lista hay que pulsar el botón "Est. Entrenamientos" que aparece en la Figura 21. El programa recorre el mapa de "datos_entrenamientos" del documento del usuario y las añade a una lista, y muestra la lista con ayuda de un adapter.

Figura 79 Funcionalidad Ver estadísticas obtenidas en entrenamientos

```
fStore.collection( collectionPath: "usuarios").document(IDUsuario).collection( collectionPath: "datos_entrenamientos").orderBy( field: "sortDate", Query.Direction.ASCENDING)
    @Override
    public void onEvent(@Nullable QuerySnapshot value, @Nullable FirebaseFirestoreException error) {
        if(error!=null){
            Log.e( tag: "Error de Firestore en coleccion datos_entrenamientos", error.getMessage());
            return;
        }
        for(DocumentChange docChange: value.getDocumentChanges()){  //se obtienen los datos de fireStore y se añaden a "lista"
            if(docChange.getType()==DocumentChange.Type.ADDED){
                lista.add(docChange.getDocument().toObject(ModeloDatosEntrenamientos.class));
            }
            deAdapter.notifyDataSetChanged();
        }
    }
```

Figura 80 Código Funcionalidad Ver estadísticas obtenidas en entrenamientos

7.4.9 Registrar datos de entrenamiento

Para acceder a esta funcionalidad hay que acceder a la lista de estadísticas de entrenamientos (Figura 79) y pulsar el botón de añadir (icono de +). Aparecerá un formulario para añadir los datos. Si se decide no añadir los datos se puede pulsar el botón "Cancelar", y si se quiere añadir, una vez añadidos los datos se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y si es así añade el documento a la colección "datos_entrenamientos" de la colección del usuario en la base de datos de Firebase.

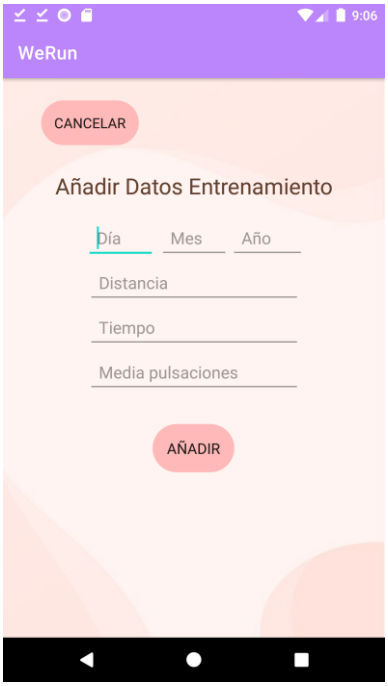

Figura 81 Funcionalidad Registrar datos de entrenamiento

```
CollectionReference doc=fStore.collection( collectionPath: "usuarios").document(IDUsuario).collection( collectionPath: "datos_entrenamientos");
Map<String, Object> datos=new HashMap<>();
datos.put( k: "sortDate", fechaSort);
datos.put( k: "fecha", fecha);
datos.put( k: "distancia", distancia);
datos.put( k: "tiempo", tiempo);
datos.put( k: "pulsaciones", pulsaciones);
datos.put( k: "IDDato", idDato);

//insertar en la bbdd
doc.add(datos);
```

Figura 82 Código Funcionalidad Registrar datos de entrenamiento

7.4.10 Eliminar datos de entrenamiento

Para ello hay que acceder a la lista de datos (Figura 79) y pulsar el botón de la papelera y aparece un mensaje para confirmar la eliminación, si se pulsa "Cancelar" no se elimina, y si se pulsa "Eliminar" se accede a la base de datos y se elimina el documento con los datos del entrenamiento en la colección "datos_entrenamiento" de la colección del usuario en base de datos Firestore.

```
holder.borrarDato.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        FirebaseAuth autenticacion;
        FirebaseUser usuario;
        String IDUsuario;
        autenticacion=FirebaseAuth.getInstance();
        FirebaseFirestore fStore=FirebaseFirestore.getInstance();
        usuario=autenticacion.getCurrentUser();
        IDUsuario=usuario.getUid();
        listaDatos.remove(position);
        notifyItemRemoved(position);
        notifyItemRangeChanged(position, listaDatos.size());
        notifyDataSetChanged();

        DocumentReference doc=fStore.collection( collectionPath: "usuarios").document(IDUsuario).collection( collectionPath: "datos_entrenamientos").document(idDato);
        doc.delete();
    }
});
```

Figura 83 Código Funcionalidad Eliminar datos de entrenamiento

7.5 Módulo rutas

7.5.1 Ver rutas propuestas

Para acceder a la lista de entrenamientos hay que pulsar el icono correspondiente en la barra de navegación de la aplicación. Con un adapter se recorre la lista y se muestra cada entrenamiento en una lista.

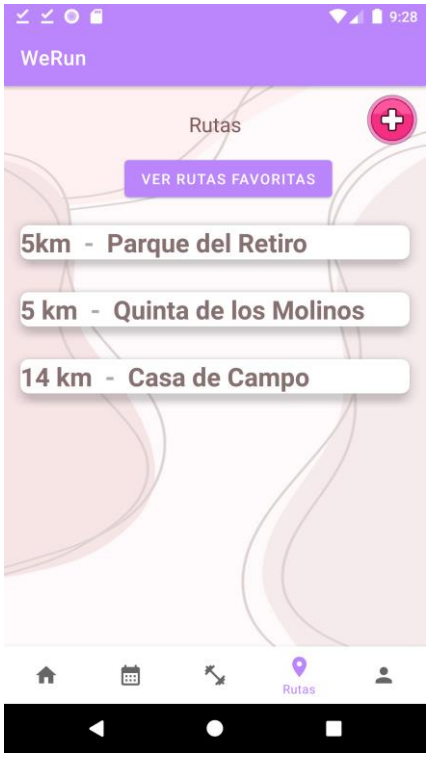

Figura 84 Funcionalidad Ver rutas propuestas

```
fStore.collection( collectionPath: "rutas").addSnapshotListener(new EventListener<QuerySnapshot>() {
    @Override
    public void onEvent(@Nullable QuerySnapshot value, @Nullable FirebaseFirestoreException error) {
        if(error!=null){
            Log.e( tag: "Error de Firestore en coleccion rutas", error.getMessage());
            return;
        }
        int i=0;
        for(DocumentChange docChange: value.getDocumentChanges()){  //se obtienen los datos de fireStore y se añaden a "lista"
            if(docChange.getType()==DocumentChange.Type.ADDED){
                lista.add(docChange.getDocument().toObject(ModeloRuta.class));
                IDRutas[i]=docChange.getDocument().getId();
            }
            adapter.notifyDataSetChanged();
            i++;
        }
    }
```

Figura 85 Código Funcionalidad Ver rutas propuestas

7.5.2 Añadir ruta

Para añadir una ruta hay que acceder a la lista de rutas y pulsar el botón de añadir (icono de +). Aparecerá un formulario para añadir la ruta. Si se decide no añadir se puede pulsar el botón "Cancelar", y si se quiere añadir, una vez añadidos los datos se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y

si es así añade el documento a la colección “rutas” de la base de datos de Firebase, y se guarda la imagen en el módulo de Firebase Storage.

Figura 86 Funcionalidad Añadir ruta

```
//subir imagen
SimpleDateFormat formatoFecha=new SimpleDateFormat( pattern: "yyyy_MM_dd__HH_mm_ss", Locale.GERMANY);
Date fecha= new Date();
String nombreArchivo=formatoFecha.format(fecha);
sref= FirebaseStorage.getInstance().getReference( location: "rutas/"+nombreArchivo);
sref.putFile(imgUri);

//preparar documento
CollectionReference doc=fStore.collection( collectionPath: "rutas");
Map<String, Object> ruta=new HashMap<>();
ruta.put( k: "distancia", distancia);
ruta.put( k: "lugar", lugar);
ruta.put( k: "idUsuario", IDUsuario);
ruta.put( k: "urlImagen", nombreArchivo);
ruta.put( k: "favoritos", String.valueOf(0));
Map<String, Object> listaComents=new HashMap<>();
ruta.put( k: "comentarios", listaComents);

//insertar en la bbdd
doc.add(ruta);
```

Figura 87 Código Funcionalidad Añadir ruta

7.5.3 Modificar/Eliminar ruta

Para realizar estas funciones hay que acceder a una ruta que se haya creado, a través de la lista de rutas. Si el usuario ha creado la ruta aparecerán los iconos para modificarla y eliminarla.

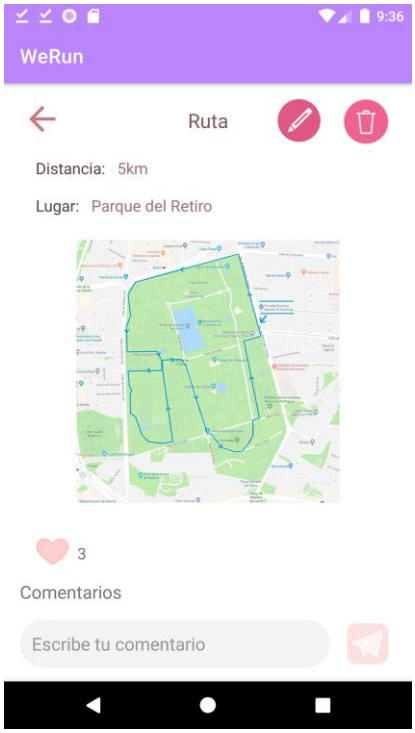

Figura 88 Funcionalidad Modificar/Eliminar ruta

7.5.3.1 Modificar ruta

Para modificar una ruta creada hay que pulsar el botón del lápiz (Figura 88). Una vez en la actividad de modificar ruta, aparecen los datos de la ruta con la posibilidad de cambiarlos. Si se decide no modificar la ruta se puede pulsar el botón "Cancelar" y si se quiere modificar, una vez cambiados los datos necesarios se pulsa el botón "Aceptar". El programa revisa que todos los campos están rellenos, y si es así actualiza los datos en la base de datos.

Figura 89 Funcionalidad Modificar ruta

```
DocumentReference doc=fStore.collection( collectionPath: "rutas").document(id);
Map<String, Object> cambios=new HashMap<>();
cambios.put( k: "distancia", mDistancia.getText().toString());
cambios.put( k: "lugar", mLugar.getText().toString());
cambios.put( k: "urlImagen", urlImg);

doc.update(cambios).addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void unused) {
        Toast.makeText( context: ModificarRuta.this,  text: "Los datos se han cambiado correctamente", Toast.LENGTH_SHORT).show();
        finish();
    }
});
```

Figura 90 Código Funcionalidad Modificar ruta

7.5.3.2 Eliminar ruta

Para acceder a esta funcionalidad hay que pulsar el botón de la papelera (Figura 88, y aparece un mensaje para confirmar la eliminación, si se pulsa "Cancelar" no se elimina, y si se pulsa "Eliminar" se accede a la base de datos y se elimina el documento de la base de datos Firestore.

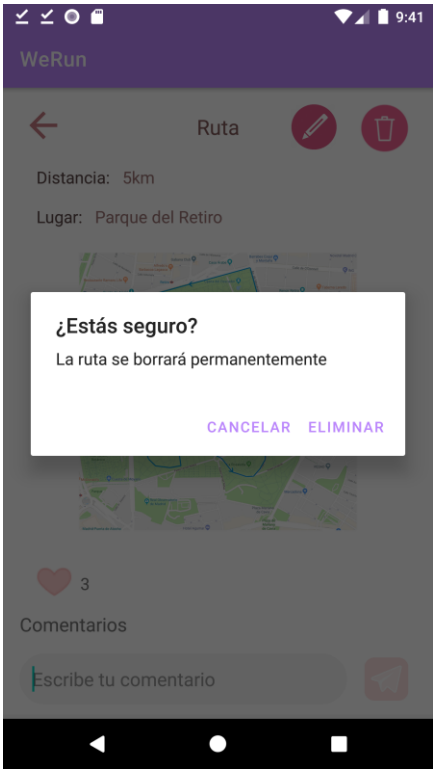

Figura 91 Funcionalidad Eliminar ruta

```
eliminarRutaBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder dialogo=new AlertDialog.Builder( context: Ruta.this);
        dialogo.setTitle("¿Estás seguro?");
        dialogo.setMessage("La ruta se borrará permanentemente");
        dialogo.setPositiveButton( text: "Eliminar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                doc.delete();
                fStore.collection( collectionPath: "rutas").document(id_).delete();
                Toast.makeText( context: Ruta.this,  text: "La ruta ha sido eliminada", Toast.LENGTH_SHORT).show();
                finish();

            }
        });
        dialogo.setNegativeButton( text: "Cancelar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.dismiss();
            }
        });
        AlertDialog dialogoAlerta= dialogo.create();
        dialogoAlerta.show();
    }
});
```

Figura 92 Código Funcionalidad Eliminar ruta

7.5.4 Añadir/eliminar ruta a lista de rutas favoritas

Para poder apuntarse o desapuntarse de un entrenamiento grupal hay que acceder a una ruta (Figura 88), y pulsar el icono del corazón. El programa revisa si el usuario tiene la ruta añadida a favoritos y muestra el botón correspondiente. Sí no está añadida y quiere añadirla, se añade el id de la ruta al mapa de "RutasFavoritas" del usuario. Si el usuario la quiere quitar de favoritas, el identificador de la ruta se elimina del mapa.

```
public void addFav(String idRuta, FirebaseAuth aut){
    IDUsuario=aut.getCurrentUser().getUid();
    fStore=FirebaseFirestore.getInstance();
    fStore.collection( collectionPath: "usuarios").document(IDUsuario).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            HashMap<String,String> rutasFav = (HashMap<String, String>) documentSnapshot.get("RutasFavoritas");
            Map<String, Object> rutas=new HashMap<>();
            String descr= distancia.getText()+" - "+lugar.getText();
            rutasFav.put(idRuta, descr);
            rutas.put( k: "RutasFavoritas", rutasFav);
            fStore.collection( collectionPath: "usuarios").document(IDUsuario).update(rutas);
            esFav=true;
            favBtn.setImageResource(R.drawable.heart_icon_filled);
        }
    });
}

public void removeFav(String idRuta, FirebaseAuth aut){
    IDUsuario=aut.getCurrentUser().getUid();
    fStore=FirebaseFirestore.getInstance();
    fStore.collection( collectionPath: "usuarios").document(IDUsuario).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            HashMap<String,String> rutasFav = (HashMap<String, String>) documentSnapshot.get("RutasFavoritas");
            Map<String, Object> rutas=new HashMap<>();
            if(rutasFav.containsKey(idRuta)){
                rutasFav.remove(idRuta);
            }
            rutas.put( k: "RutasFavoritas", rutasFav);
            fStore.collection( collectionPath: "usuarios").document(IDUsuario).update(rutas);
            esFav=false;
            favBtn.setImageResource(R.drawable.heart_icon_empty);
        }
```

Figura 93 Código Funcionalidad Añadir/eliminar ruta a lista de rutas favoritas

7.5.5 Añadir/eliminar comentario en ruta

7.5.5.1 Añadir comentario

Para escribir un comentario en una ruta (Figura 88), hay que escribir el comentario deseado en el campo que pone "Escribe tu comentario", y pulsar el icono del avión de papel. El programa guarda los datos del comentario en un mapa y lo añade a la base de datos, a la colección comentarios de la ruta.

```java
//FUNCIONALIDAD AÑADIR COMENTARIO
addComentariobtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        addComentariobtn.setVisibility(View.INVISIBLE);
        String idComentario = UUID.randomUUID().toString();
        String coment=comentario.getText().toString();
        CollectionReference col=fStore.collection( collectionPath: "rutas").document(id_).collection( collectionPath: "comentarios");
        Map<String, Object> comentarioMapa=new HashMap<>();
        comentarioMapa.put( k: "comentario", coment);
        comentarioMapa.put( k: "IDUsuario", IDUsuario);
        comentarioMapa.put( k: "nombreUsuario", nombreUsuario);
        comentarioMapa.put( k: "IDComentario", idComentario);
        comentarioMapa.put( k: "IDRuta", id_);
        SimpleDateFormat formatoFecha=new SimpleDateFormat( pattern: "dd/MM/yyyy HH:mm", Locale.GERMANY);
        Date fechaDate= new Date();
        String fecha=formatoFecha.format(fechaDate);
        comentarioMapa.put( k: "fecha", fecha);
        col.document(idComentario).set(comentarioMapa).addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void unused) {
                        Toast.makeText( context: Ruta.this, text: "Comentario añadido", Toast.LENGTH_LONG).show();
                        comentario.setText("");
                        addComentariobtn.setVisibility(View.VISIBLE);
                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText( context: Ruta.this, text: "No se ha podido añadir el comentario"+e.getMessage(), Toast.LENGTH_LONG).show();
            }
        });
    }
});
```

Figura 94 Código Funcionalidad Añadir Comentario

7.5.5.2 Eliminar comentario

Para eliminar un comentario que se ha realizado, el usuario tiene que acceder a la lista de comentarios de la ruta y pulsar el botón de la papelera que se encuentra al lado del comentario que se desea eliminar. Un usuario solo puede borrar comentarios que ha realizado el propio usuario. Cuando se pulsa el botón de la papelera el programa accede a la base de datos y elimina el comentario.

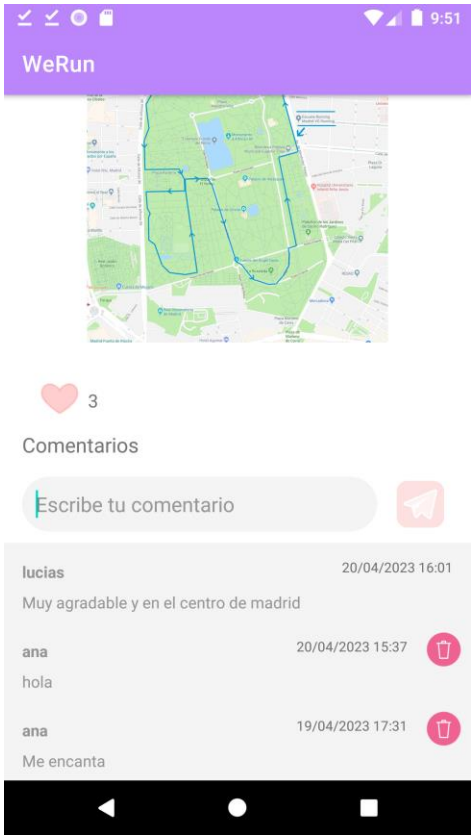

Figura 95 Funcionalidad Eliminar comentario en ruta

```
if(idComentador.equals(IDUsuarioActual)){
    holder.borrarComentario.setVisibility(View.VISIBLE);
    String idComentario=listaComent.get(position).getIDComentario();
    String idRuta=listaComent.get(position).getIDruta();
    holder.borrarComentario.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            listaComent.remove(position);
            notifyItemRemoved(position);
            notifyItemRangeChanged(position, listaComent.size());
            FirebaseFirestore fStore=FirebaseFirestore.getInstance();
            notifyDataSetChanged();

            DocumentReference doc=fStore.collection( collectionPath: "rutas").document(idRuta).collection( collectionPath: "comentarios").document(idComentario);
            doc.delete();
        }
    });
}
```

Figura 96 Código Funcionalidad Eliminar comentario en ruta

Capítulo 8 - Conclusiones y trabajo futuro

Conclusiones

En este proyecto se ha desarrollado un sistema para gestionar las actividades y necesidades de un club de running, con el objetivo de facilitar la interacción entre el club y sus miembros.

Con el fin de conseguir esta administración, se han implementado funcionalidades para añadir, modificar y eliminar entrenamientos y eventos por parte del club, y ofrece a los miembros la posibilidad de apuntarse o desapuntarse de estos eventos.

A su vez, los usuarios miembros pueden llevar una monitorización de sus marcas en competiciones y sus estadísticas en entrenamientos, así como publicar entrenamientos grupales con el fin de entrenar con otros miembros del club, que pueden apuntarse a estos. Los usuarios también pueden publicar rutas de entrenamiento y otros miembros pueden añadirlas a favoritos y comentar en ellas.

Se han realizado todas las funcionalidades planificadas para la aplicación cumpliendo con los objetivos establecidos para ella.

Enlace a los ficheros:

https://drive.google.com/drive/folders/1C4KS6PngZ1YOYLHmSMKv_jVcXV85l9dI?usp=sharing

Trabajo Futuro

Para que la aplicación sea más completa, hay funcionalidades que se podrían mejorar o añadir. Son las siguientes:

- Añadir la opción para los entrenadores de ver las personas que se han apuntado a los eventos.
- Desactivar la visibilidad de los eventos y entrenamientos que ya han pasado para que los usuarios no los vean.
- Añadir la opción para que los creadores de los entrenamientos grupales puedan ver las personas que se han apuntado a ellos.
- Crear gráficos con los datos de los entrenamientos para poder ver el progreso de forma visual.
- Añadir la posibilidad de dibujar la ruta a publicar aparte de poder subir la foto.
- Añadir sección de retos deportivos para motivar a los miembros del club y que compitan entre ellos para conseguir los retos.

Conclusions and Future Word

Conclusions

In this project, a system has been developed to manage the activities and needs of a running club, with the aim of facilitating the interaction between the club and its members.

To achieve this club management, functionalities have been implemented to add, modify, and delete trainings and events by the club administrators, and it offers members the possibility of signing up to the events.

Member users can also track their performance in competitions and their training statistics, as well as publish group workouts to train with other club members, who can sign up for these sessions. Users can also share training routes, and other members can bookmark them and leave comments.

All the planned functionalities for the application have been implemented, meeting the established goals for it.

Future Work

To make the application more complete, there are functionalities that could be improved or added. The functionalities are:

- Add the option for trainers to see the people who have signed up for the events.
- Disable the visibility of events and training sessions that have passed so that users cannot see them.
- Add the option so that group training sessions creators can see which people have signed up for them.
- Create graphics with the training data so users can see their progress in a more visual way.
- Add the possibility to draw the route to be publishes aside from uploading the picture.
- Add a section for sports challenges to motivate club members and encourage them to compete against each other to achieve the challenges.

BIBLIOGRAFÍA

[1] Statista: https://es.statista.com/estadisticas/569559/evolucion-del-porcentaje-de-individuos-que-practicaron running/#:~:text=La%20popularidad%20del%20running%20en,punto%20m%C3%A1s%20que%20en%202019.

[2]Adidas Running: https://apps.apple.com/es/app/adidas-running-correr-caminar/id336599882

[3] Strava: https://apps.apple.com/es/app/strava-corre-camina-pedalea/id426826309

[4] CR7 Fitness by Crunch: https://apps.apple.com/es/app/cr7-fitness-by-crunch/id1592974517

[5] Android Studio: https://developer.android.com/studio

[6] Java: https://www.java.com/es/

[7] XML: https://es.wikipedia.org/wiki/Extensible_Markup_Language

[8] Firebase: https://firebase.google.com/?hl=es

Apéndice A - Guía de uso

Acceso a la aplicación

Al iniciar la aplicación aparece la pantalla de inicio de sesión, y el usuario puede iniciar sesión introduciendo sus datos y pulsando el botón "Iniciar Sesión", o pulsar el texto de registro para acceder a un formulario de registro. Tanto al pulsar el botón de "Iniciar Sesión como el de registro, la aplicación lleva al usuario a la página de Inicio de la aplicación.

Figura 97 Uso: Acceso a la aplicación

Modificar y eliminar perfil

Para modificar o eliminar el perfil hay que acceder a la zona del perfil desde la barra de navegación, y pulsar el botón correspondiente según lo que se desee hacer. Si se elige modificar el perfil, se cambian los datos a los deseados y se pulsa el botón "Aceptar". En caso de eliminarlo, aparecerá un mensaje de confirmación y hay que pulsar en "Eliminar".

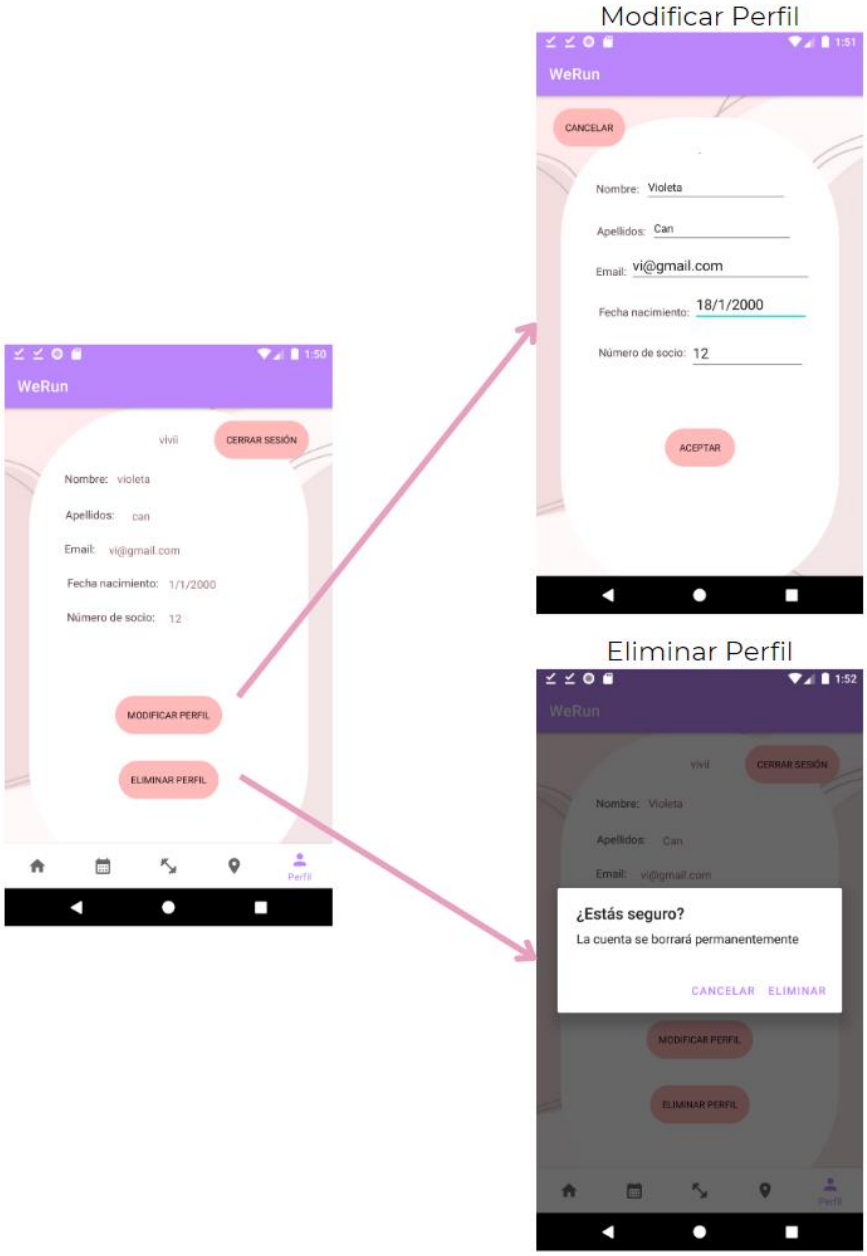

Figura 98 Uso: Modificar y Eliminar Perfil

Panel de administración

Para acceder al panel de administración se necesita estar utilizando una cuenta de entrenador, y acceder a través del botón "Panel Admin" de la pantalla de Inicio. Pulsando los botones se pueden añadir eventos o entrenamientos. En el caso de entrenamientos hay que elegir primero al usuario con el que se realizará el entrenamiento. Después de rellenar el formulario correspondiente se pulsa el botón "Añadir" para agregar el evento o entrenamiento a la base de datos.

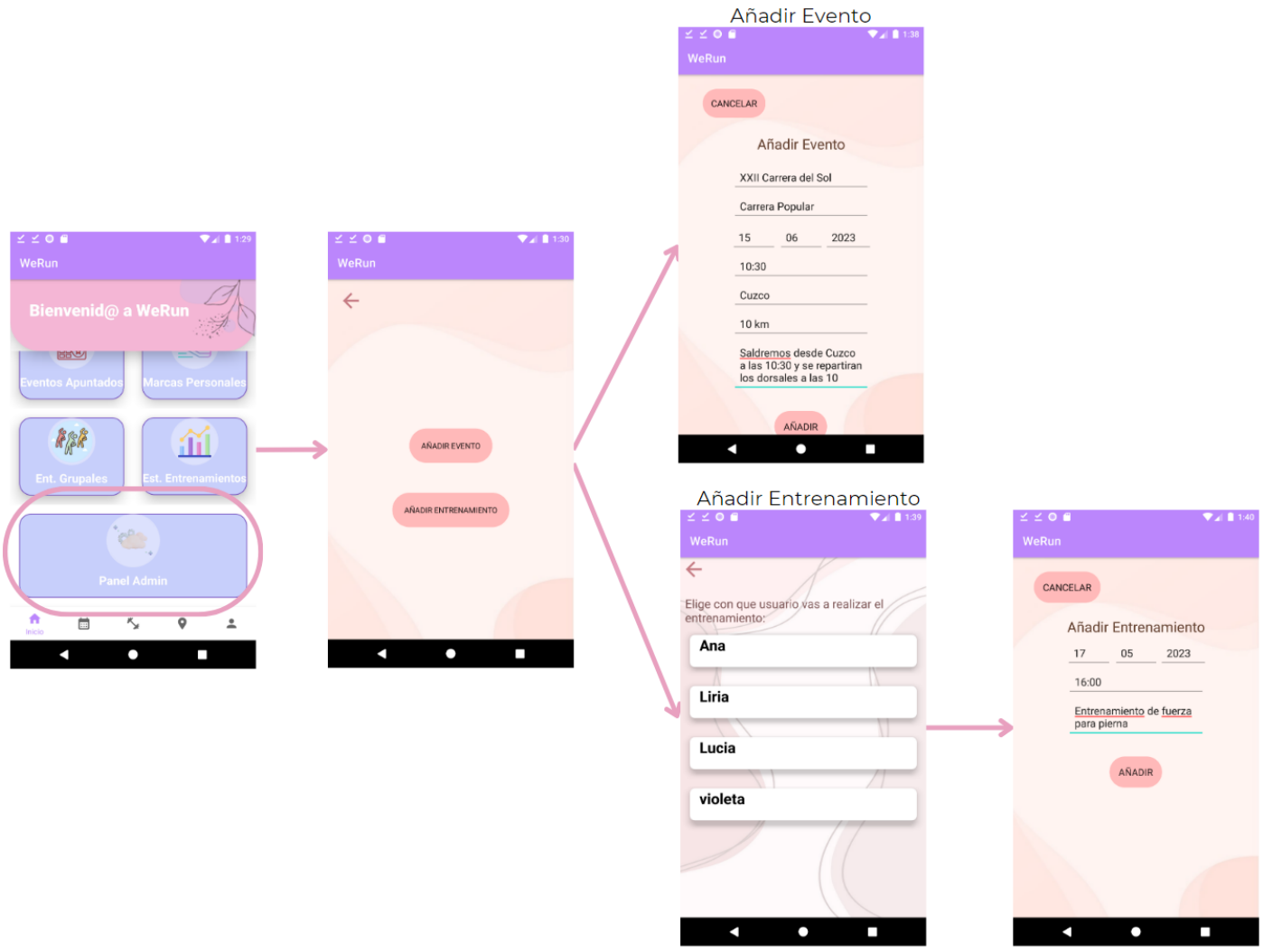

Figura 99 Uso: Panel de Administración

Modificar y Eliminar Evento

Para modificar o eliminar un evento hay que ser administrador. Hay que acceder a la zona de eventos desde la barra de navegación, y pulsar el en el evento que se quiere gestionar. Aparece la pantalla del evento y hay que pulsar el botón correspondiente según lo que se desee hacer. Si se elige modificar el evento, se cambian los datos a los deseados y se pulsa el botón "Aceptar". En caso de eliminarlo, aparecerá un mensaje de confirmación y hay que pulsar en "Eliminar".

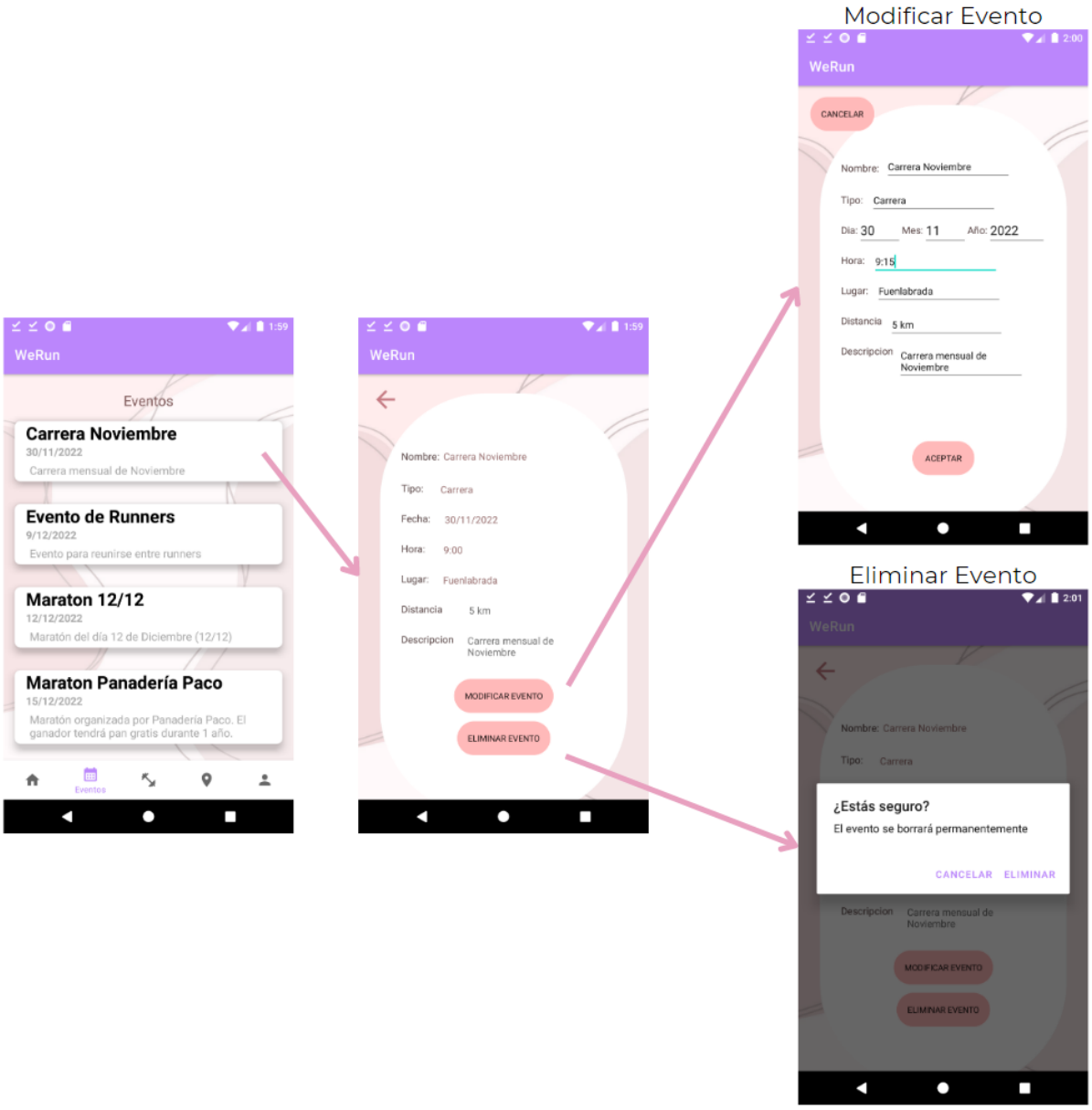

Figura 100 Uso: Modificar y Eliminar Evento

Modificar y Eliminar Entrenamiento

Para modificar o eliminar un evento hay que ser administrador. Hay que acceder a la zona de entrenamientos desde la barra de navegación, y pulsar el en el entrenamiento que se quiere gestionar. Aparece la pantalla del entrenamiento y hay que pulsar el botón correspondiente según lo que se desee hacer. Si se elige modificar el entrenamiento, se cambian los datos a los deseados y se pulsa el botón "Aceptar". En caso de eliminarlo, aparecerá un mensaje de confirmación y hay que pulsar en "Eliminar".

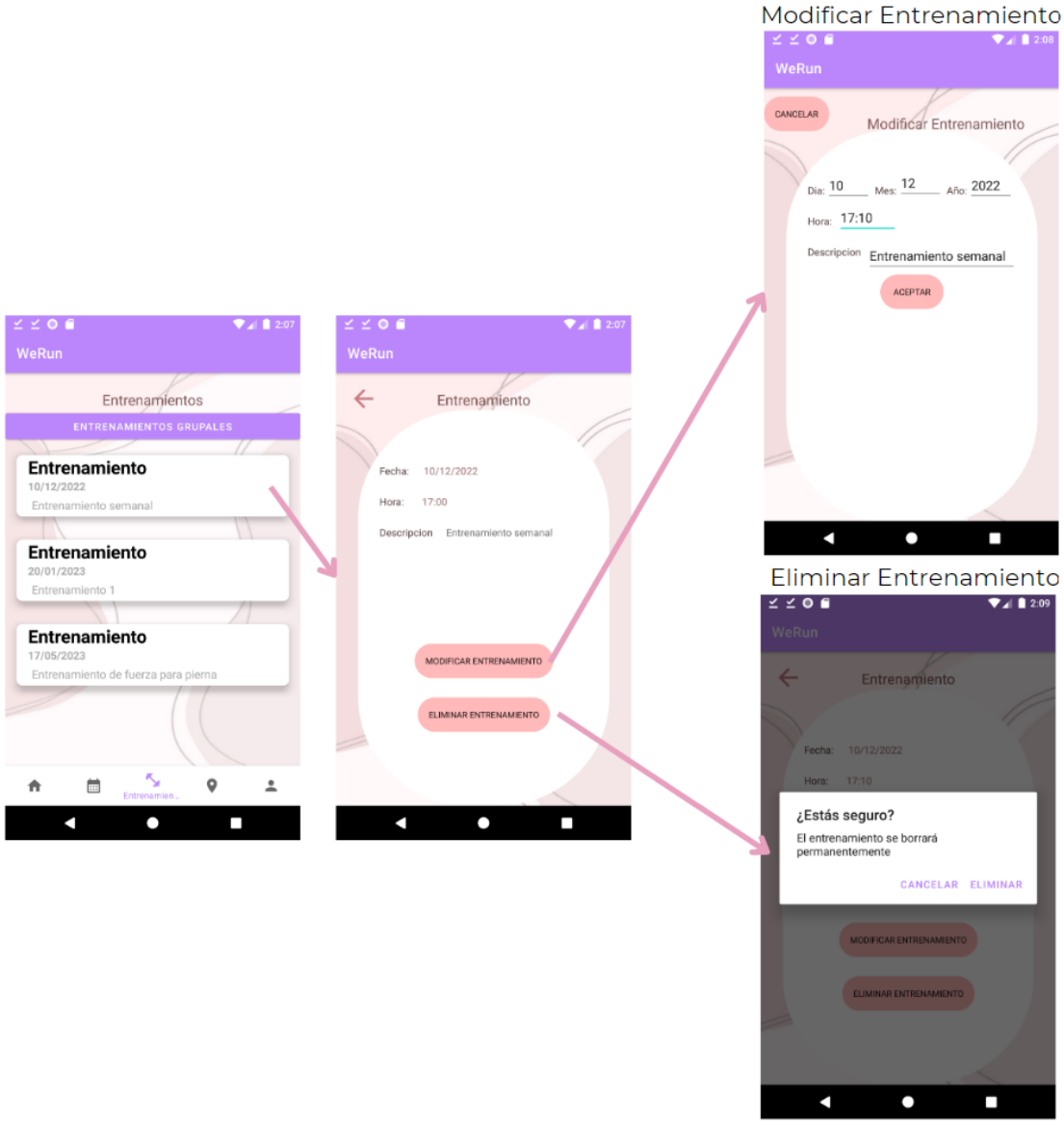

Figura 101 Uso: Modificar y Eliminar Entrenamiento

Apuntarse/Desapuntarse de evento

Para apuntarse o desapuntarse de un evento hay que acceder a la lista de eventos a través de la barra de navegación. Se pulsa en el entrenamiento deseado y en el botón "Apuntarse" si quiere apuntarse, o "Desapuntarse si quiere desapuntarse.

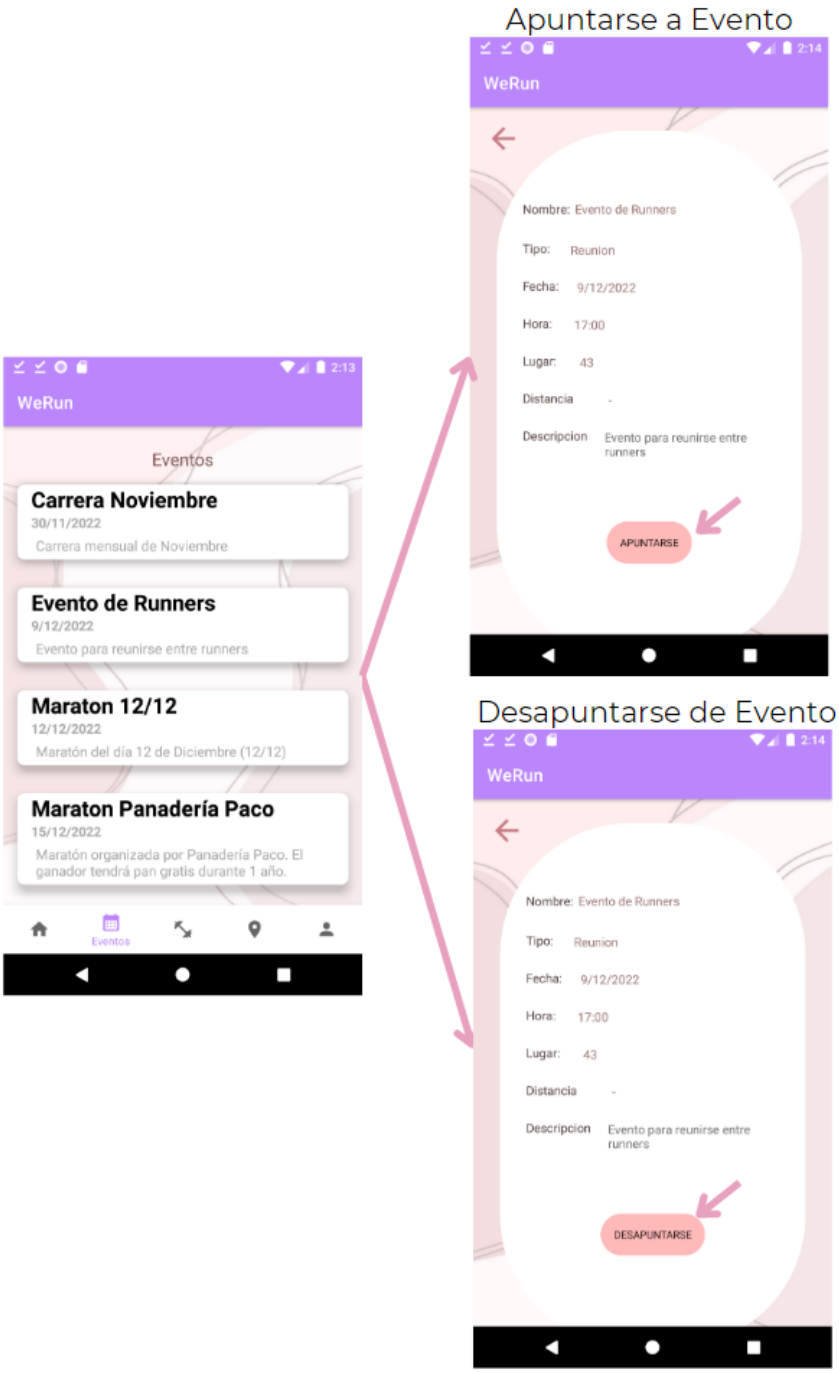

Figura 102 Uso: Apuntarse/Desapuntarse de Evento

Ver lista de Eventos Apuntados

Para ver la lista de eventos apuntados hay que acceder al menú de inicio a través de la barra de navegación. Se pulsa en el botón "Eventos Apuntados", y así se accede a la lista de los eventos en los que esta apuntado.

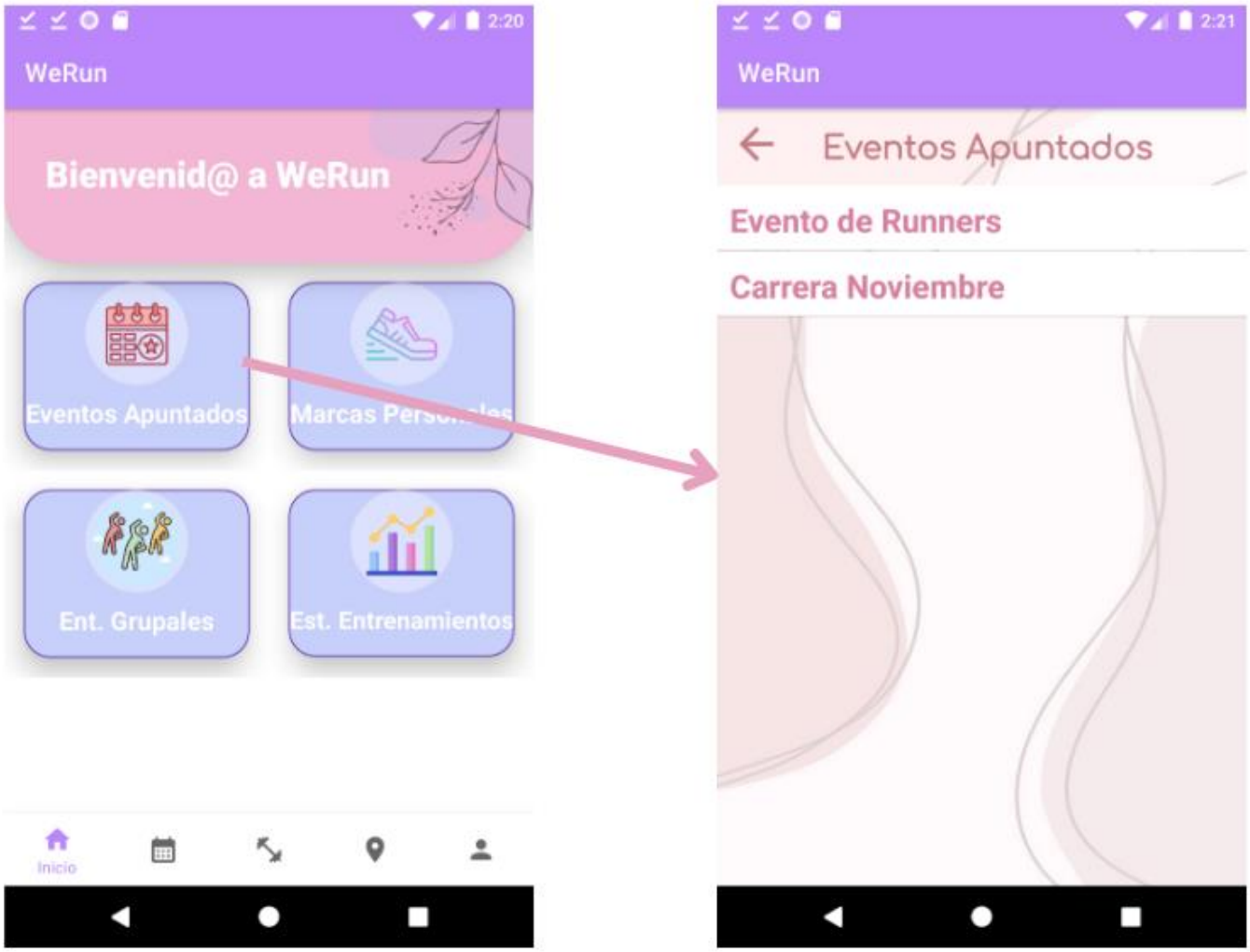

Figura 103 Uso: Ver lista de Eventos Apuntados

Ver estadísticas en eventos y añadir o eliminar marca personal

Para acceder a la lista de marcas personales hay que acceder al menú de inicio a través de la barra de navegación. Se pulsa en el botón "Marcas Personales", y aparece la lista de estadísticas. Si se pulsa en una se ven los datos de la estadística, y si se pulsa en el botón de la papelera se elimina, aparece mensaje de confirmación y se pulsa en eliminar. Para añadir una marca nueva, hay que pulsar en el icono del + en la lista de marcas, rellenar el formulario y pulsar el botón "Añadir".

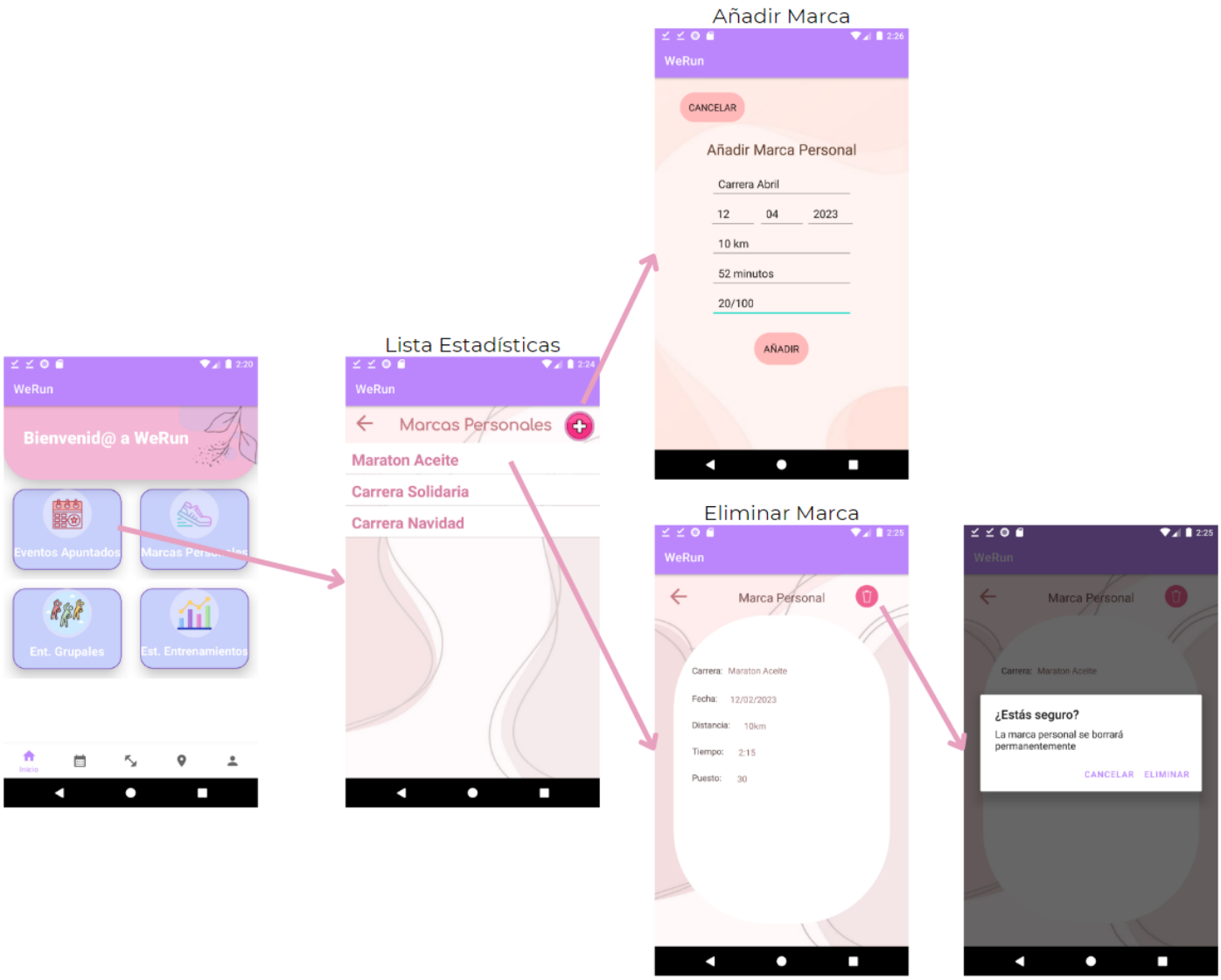

Figura 104 Uso: Ver estadísticas en eventos y añadir o eliminar marca personal

Entrenamientos grupales

Hay distintas funcionalidades que se pueden realizar.

- Para acceder a la lista hay que acceder al menú de inicio a través de la barra de navegación. Se pulsa en el botón "Ent. Grupales", y aparece la lista de entrenamientos.
- Si se quiere añadir un entrenamiento se pulsa el icono + de la lista, se rellenan los datos y se pulsa "Añadir".
- Si se quiere apuntar a un entrenamiento de otro usuario, se pulsa en el entrenamiento y el botón de "Apuntarse", para desapuntarse se realizan los mismos pasos, pero el botón cambiaría a "Desapuntarse".
- Para modificar un entrenamiento grupal el usuario tiene que ser el creador de este. Hay que acceder al entrenamiento pulsando en él en la lista, y dentro de él pulsar en el botón "Modificar Entrenamiento Grupal". Se cambian los datos necesarios y se pulsa en el botón "Aceptar".
- Para eliminar un entrenamiento grupal el usuario tiene que ser el creador de este. Hay que acceder al entrenamiento pulsando en él en la lista, y dentro de él pulsar en el botón "Eliminar Entrenamiento Grupal". Aparece un aviso de confirmación, y se pulsa en el botón "Eliminar".

Figura 105 Uso: Entrenamientos grupales

Registrar y Eliminar datos de entrenamiento

Para acceder a la lista hay que acceder al menú de inicio a través de la barra de navegación. Se pulsa en el botón "Est. Entrenamientos", y aparece la lista de datos. Para añadir un dato se pulsa el icono + de la lista, se rellenan los datos y se pulsa "Añadir". Si se desea eliminar un dato hay que pulsar el icono de la papelera al lado del dato.

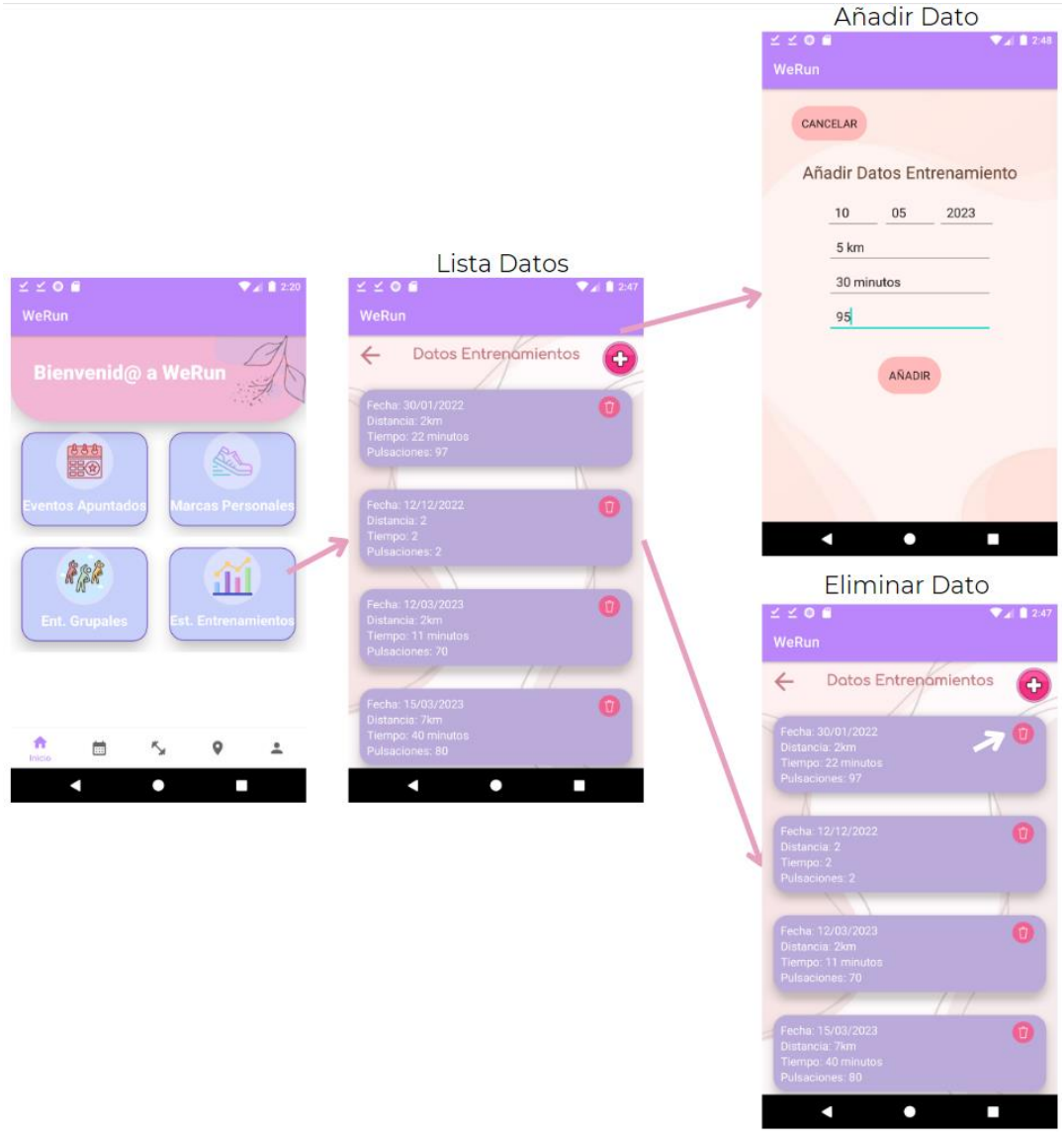

Figura 106 Uso: Registrar y Eliminar datos de entrenamiento

Añadir ruta

Para añadir una ruta hay que acceder a la lista de rutas. Para ello se acceder a la zona de rutas desde la barra de navegación. Después, se pulsa en el botón + y se accede al formulario para añadir una ruta.

Se rellena el formulario con la distancia y el lugar de la ruta, y se añade una fotografía del mapa. Se pulsa el botón "Añadir" y la ruta es agregada a la lista.

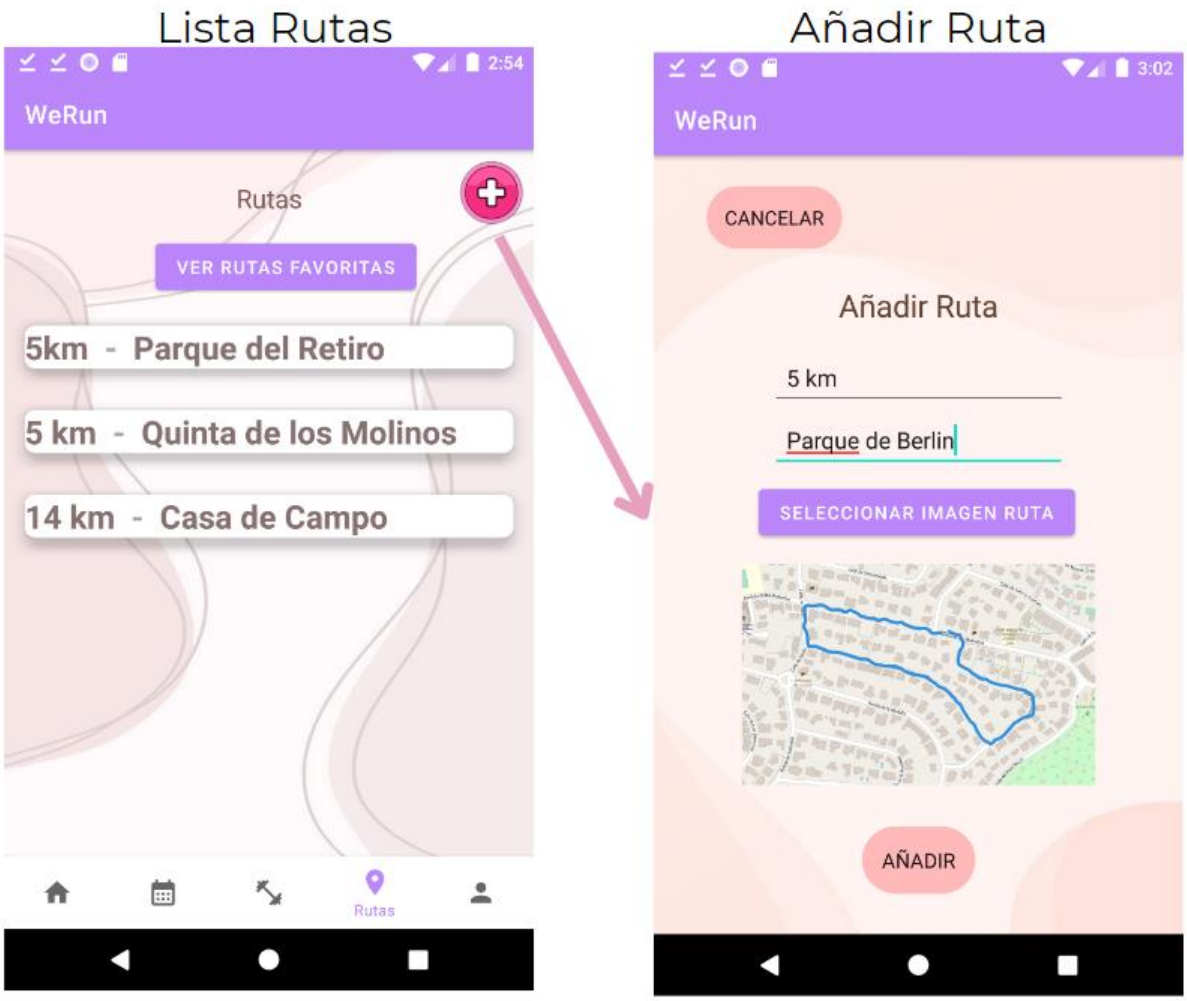

Figura 107 Uso: Añadir ruta

Modificar y Eliminar Ruta

Para modificar o eliminar una ruta hay que ser el creador. Hay que acceder a la zona de rutas desde la barra de navegación, y pulsar el en la ruta que se quiere gestionar. Aparece la pantalla de la ruta y hay que pulsar el botón correspondiente según lo que se desee hacer. Si se elige modificarla, se cambian los datos a los deseados y se pulsa el botón "Aceptar". En caso de eliminarla, aparecerá un mensaje de confirmación y hay que pulsar en "Eliminar".

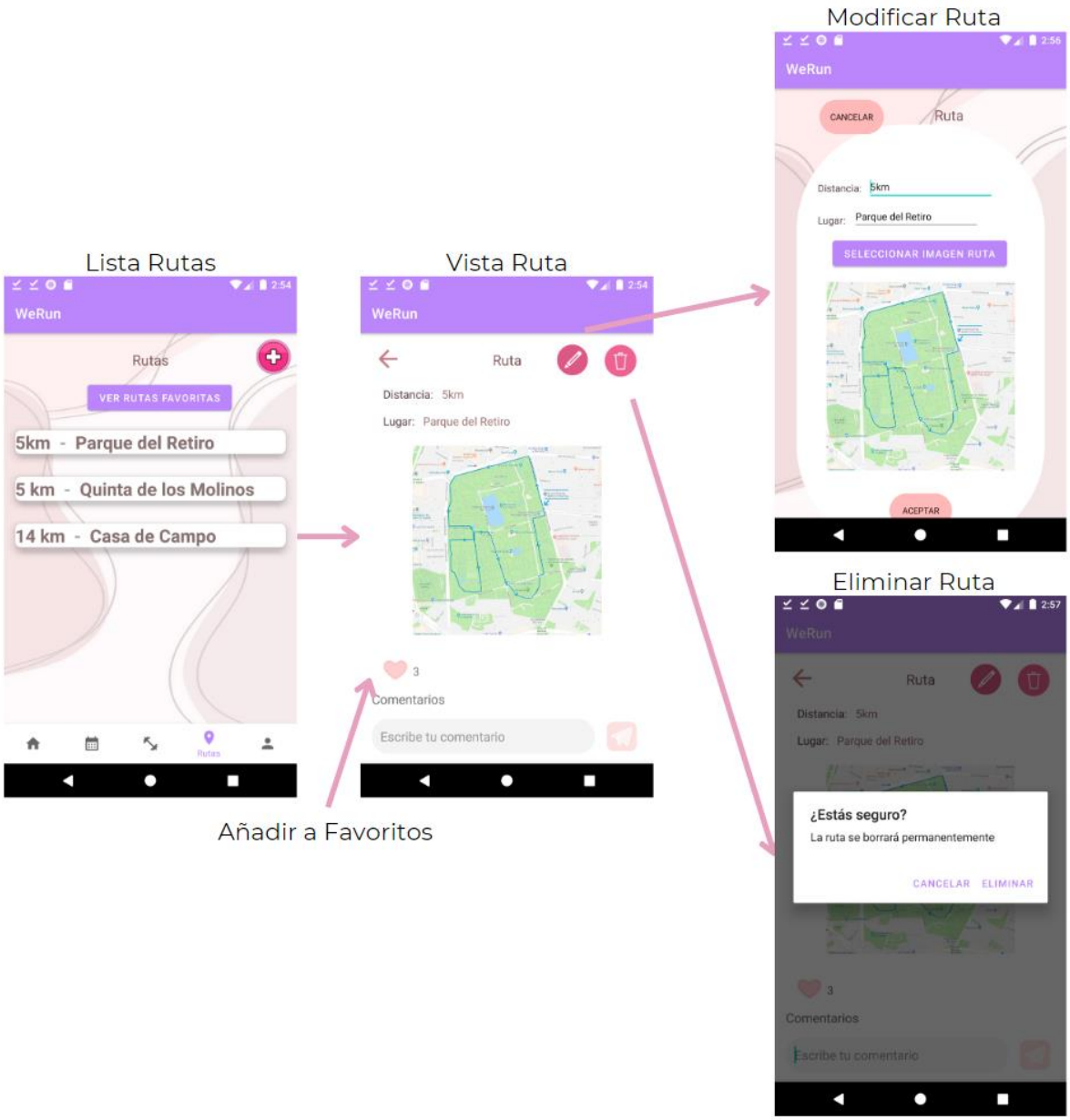

Figura 108 Uso: Modificar y Eliminar Ruta

Añadir y eliminar comentarios en ruta

Para añadir o eliminar un comentario en una ruta hay que acceder a la zona de rutas desde la barra de navegación, y pulsar el en la ruta que se quiere comentar. Para comentar se escribe en el recuadro y se pulsa el icono del avión de papel. Para eliminar un comentario hay que ser el creador, y hay que pulsar en el icono de la papelera.

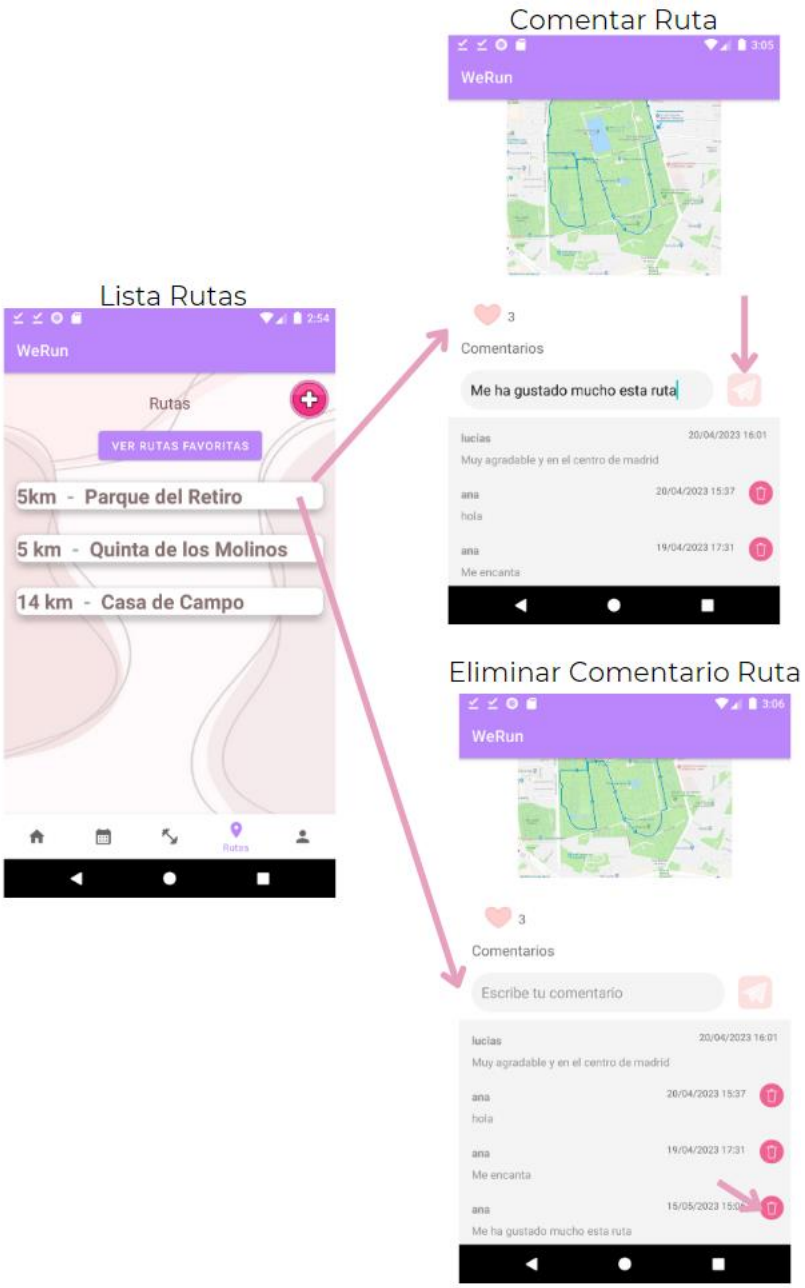

Figura 109 Uso: Añadir y eliminar comentarios en ruta