ugr | Universidad de Granada

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

Desarrollo de una aplicación para el seguimiento de deportistas

---

Autor
David Peinado Perea

Director
Juan Manuel Fernández Luna

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Junio de 2019

Desarrollo de una aplicación para el seguimiento de deportistas

Autor
David Peinado Perea

Director
Juan Manuel Fernández Luna

Desarrollo de una aplicación para el seguimiento de deportistas

David Peinado Perea

Palabras clave: aplicación móvil, Android, Flutter, Firebase, Firestore, actividad física, deporte, entrenador, comparaciones, seguimiento, GPS.

Resumen

El desarrollo de este Trabajo Fin de Grado (TFG) tiene como principal objetivo la creación de una aplicación móvil en la cual los usuarios puedan realizar un seguimiento de sus actividades físicas. Durante las actividades se recopilarán datos desde el GPS del dispositivo para luego ser procesados y mostrados al usuario mediante interfaces claras e intuitivas. Estas actividades y sus respectivos datos asociados podrán ser comparadas por los usuarios implementando para ello un sistema de grupos y otro de comparaciones.

Development of an application to track athletes' activities

David Peinado Perea

Abstract

Keywords: mobile app, mobile application, Android, Flutter, Firebase, Firestore, physical activity, sport, athlete, trainer, comparison, tracking, GPS.

This final project has as its main aim the creation of a mobile app that can be used to monitor its users' physical activities. During their sessions data will be gathered from the device's GPS to be processed and shown to the user through clear and intuitive interfaces. This activities and their respective associated data could be compared by the users implementing a group system and a comparison one.

Yo, **David Peinado Perea**, alumno de la titulación **GRADO EN INGENIERÍA INFORMÁTICA** de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75573069Z, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo. David Peinado Perea

Granada, 18 de Junio de 2019.

D. **Juan Manuel Fernández Luna**, Profesor del Área de Ciencias de la Computación e Inteligencia Artificial del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Desarrollo de una aplicación para el seguimiento de deportistas***, ha sido realizado bajo su supervisión por **David Peinado Perea**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 18 de Junio de 2019.

El tutor:

D. Juan Manuel Fernández Luna

Agradecimientos

A Yanira, Paco, Pablo, Félix, y Rafa.

Índice general

Índice de figuras

1. Introducción

1.1 Motivación

La realización de actividades físicas es una práctica habitual para gran parte de la población [1] . Las hormonas que se liberan durante el proceso, así como la mejora física, la mejora psicológica, y en algunos casos la interacción social, hace a las personas "adictas" a realizar actividad física [2].

Esto sumado a la gran ayuda que ofrecen las aplicaciones móviles a la población que quiere tener una vida activa [3] ha creado un nicho de mercado en el sector tecnológico, donde han aparecido distintas aplicaciones destinadas a satisfacer las necesidades de los usuarios con este tipo de vida, orientadas al seguimiento de la actividad física, o incluso aplicaciones sociales para realizar deporte en grupo o planificarlo.

Entrando en este sector, nos centramos en un grupo en particular: los entrenadores deportivos. Esta aplicación se ha concebido con el fin de ayudarles a realizar un seguimiento de sus jugadores, así como comparaciones entre varios de ellos. Sin embargo, esta aplicación, aunque esté pensada para entrenadores, puede ser usada por cualquier persona, sea o no entrenador, pues no se necesita ningún tipo de conocimiento técnico para usarla. Puede usarse como herramienta para el análisis de la actividad física de una o varias personas a lo largo del tiempo, puede tener un uso completamente individual o bien tener un carácter grupal. El usuario elegirá cómo usarla en función de sus necesidades.

Esta aplicación puede llegar a ser de gran utilidad tanto para los particulares como para organizaciones deportivas. Centrándose en proporcionar información visual de gran importancia a entrenadores o profesionales del deporte, podrá ser utilizada por instituciones relacionadas con el deporte y clubes privados, pudiendo colaborar con estos para hacer un producto final que se ajuste perfectamente a las necesidades de los mismos.

Si se trabaja en conjunto con instituciones deportivas o educativas se podría ayudar a los preparadores físicos y entrenadores de multitud de disciplinas deportivas. Así pues este es un proyecto con mucho potencial de futuro.

1.2 Objetivos

El objetivo de este proyecto es la creación de una aplicación móvil donde los usuarios puedan realizar un seguimiento de sus actividades físicas y visualizar los datos extraídos a partir del GPS del dispositivo durante las actividades.

A continuación se muestra una lista de los objetivos específicos, que desglosan el objetivo principal en objetivos más concretos y acotados, y que deberán ser cumplidos en el proyecto. También se incluyen objetivos opcionales, destinados a mejorar la calidad de la aplicación.

A continuación se muestra exponen los objetivos específicos del proyecto:

Objetivo 1	Diseñar las interfaces de usuario aplicación
Tipo	Obligatorio
Descripción	Se creará un conjunto de interfaces intuitivas y visualmente atractivas, que engloben todas las funcionalidades necesarias de la aplicación

Objetivo 2	Gestionar usuarios
Tipo	Obligatorio
Descripción	La aplicación debe permitir dar de alta a un nuevo usuario, así como permitirle iniciar sesión y desconectarlo

Objetivo 3	Gestionar actividades físicas
Tipo	Obligatorio
Descripción	Hacer que la aplicación pueda crear, borrar y realizar seguimiento de actividades físicas

Objetivo 4	Gestionar grupos
Tipo	Obligatorio
Descripción	Implementar un sistema de grupos que permita a los usuarios crear, borrar y gestionar grupos para que más tarde puedan realizar comparaciones sobre los miembros de los mismos

Objetivo 5	Gestionar comparaciones
Tipo	Obligatorio
Descripción	Permitir al usuario realizar comparaciones de sus actividades o de las actividades de otros usuarios con los que tenga un grupo en común

Objetivo 6	Implementar métricas avanzadas
Tipo	Opcional
Descripción	Se implementarán métricas de carácter avanzado para un mejor análisis por parte de los entrenadores

1.3 Estudio del estado del arte

Para tener una idea consistente de cómo se quería hacer la aplicación se realizó un estudio del estado del arte. Para ello se analizaron aplicaciones de este sector,como Runtastic [4] o Strava [5]. Estas dos fueron las que se instalaron y probaron en un smartphone.

Tanto Runtastic como Strava son dos aplicaciones consolidadas dentro de este mercado, contando con más de 1 millón de descargas entre la versión gratuita y de pago de Runtastic y más de 400.000 por parte de Strava.

Se utilizaron para el uso personal durante unos cuantos días, donde se pudo analizar cómo se estructuraban, qué tipo de interfaces tenían y cómo mostraban los datos relativos a la actividad física al usuario.

Ambas aplicaciones muestran una buena cantidad de métricas para cada actividad física realizada, y también muestran el recorrido que realizó el usuario durante la misma. También incluyen gráficas para añadir valor añadido a los datos recopilados de la actividad. También incorporan un sistema de grupos que hace posible la interacción social en estas plataformas, siendo esta característica prácticamente obligatoria en toda aplicación del sector que quiera triunfar.

Así pues, una vez finalizado el estudio, se pasó a la creación de los requisitos de la aplicación, pues se tenía una idea clara de cuáles eran las características que debía poseer la aplicación y qué funcionalidades necesitaba implementar.

2. Planificación

2.1 Fases

En este capítulo se expone la planificación del proyecto para que se clarifiquen las tareas que se han llevado a cabo hasta conseguir el objetivo propuesto.
Se utilizó el enfoque de desarrollo ágil [6], haciendo así un desarrollo iterativo e incremental donde los requisitos y soluciones evolucionan con el tiempo según las nuevas necesidades y problemas que plantee el proyecto.

Se optó por este tipo de desarrollo frente al desarrollo en cascada [7] ya que se previó que a medida que se fuera desarrollando, surgirían nuevos retos que obligarían a modificar los requisitos previamente establecidos, así pues se iteraría sobre estos procesos hasta dejarlos estables.

Esta metodología está también muy extendida en el mundo empresarial, así pues, seguirla aportaría un valor añadido en el proceso de aprendizaje que implica este trabajo.

En la primera iteración se hizo un buen trabajo conceptual que hizo que las siguientes iteraciones no presentaran grandes cambios respecto a la primera. Sólo cambiaban sutilezas que, o bien no se tuvieron en cuenta en su momento, o cuando se profundizó en ellas hubo que cambiarlas debido a la imposibilidad de hacerlas como se habían planeado anteriormente.

2.1.1 Fase inicial

En esta fase inicial se realizó un breve estudio del arte, donde se analizaron aplicaciones similares como Runtastic o Strava[1-19]. Se gestó una idea de cómo podía llegar a ser esta aplicación. Aportando ideas tanto mi tutor como mis familiares y amigos como yo mismo.
Una vez se tuvo la idea global de la aplicación, se definieron las características claves de ésta y se procedió a la fase de análisis.

2.1.2 Fase de análisis

Teniendo en mente la idea inicial de la aplicación, se realizó un estudio de aquellos requisitos software necesarios para el correcto funcionamiento de la aplicación. Así, se lleva a cabo una especificación de requisitos que la aplicación debe cumplir para cubrir todos los objetivos y necesidades. Esta especificación contiene requisitos funcionales, requisitos no funcionales y casos de uso del sistema.

2.1.3 Fase de diseño

Se lleva a cabo el diseño integral de la aplicación. Se hace el diagrama de actividades de la aplicación,el diseño de la base de datos y un diseño aproximado de la interfaces de usuario. También se realizó un estudio de las tecnologías que se iban a podrían usar.

2.1.4 Fase de implementación

Aquí se presentaron las dos grandes decisiones del proyecto:

-Desarrollar con Flutter o desarrollar con Android Studio.
-Crear y administrar una base de datos propia o usar Firebase.

Posteriormente se explicarán las ventajas e inconvenientes de cada una de estas opciones, qué implican y por cuáles se optó.
Gracias a las herramientas que se utilizaron, la fase de desarrollo y la de prueba y corrección, se fusionan. Haciendo que la velocidad de desarrollo del proyecto aumente significativamente.

2.1.5 Redacción de la memoria

Esta es la última fase, que se realizó después de la primera iteración del desarrollo de la aplicación. Se plasmó la información que se obtuvo hasta el momento y se actualizó conforme los cambios y correcciones iban ocurriendo.

2.2 Planificación temporal

Al principio del proyecto se creó un diagrama de Gantt con una estimación aproximada del tiempo que tomaría la realización de cada fase de éste. El ajuste real a esta planificación fue casi exacto.

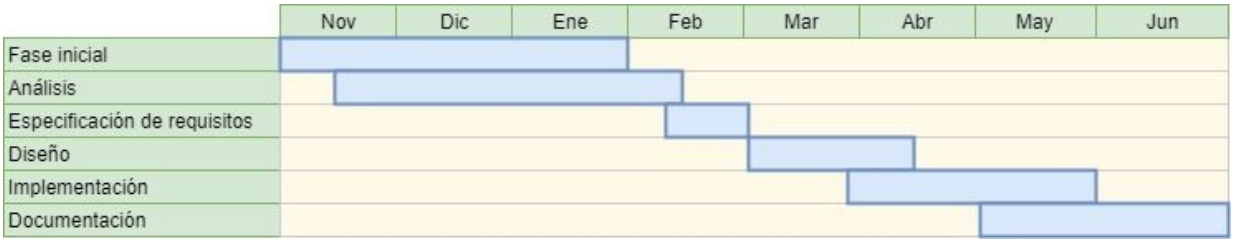

Figura 1 Diagrama de Gantt con la planificación inicial del proyecto

2.3 Recursos

En este apartado se listan todos los recursos hardware y software utilizados.

2.3.1 Hardware

Ordenador personal: CPU i5-5500, 16 GB RAM, 1 TB HDD + 256 GB SSD.
Smartphone personal: Xiaomi Redmi Note 5 Pro

2.3.2 Software

SO: Windows 11

Editor de código: Visual Studio Code con multitud de extensiones para soportar Git, Dart y Flutter entre otros.

SDK: Flutter 1.2.1

Lenguaje de programación: Dart 2.2

Servicios cloud utilizados: Firebase Auth para la autenticación y Firebase Firestore para la base de datos.
Herramientas para realizar los diagramas: draw.io, MockFlow y VisualParadigm Online.

2.4 Presupuesto

2.4.1 Licencias

Todo el software usado para el desarrollo es gratuito. No es necesario un plan de pago para los servicios de Firebase, así que el coste es de 0€.

2.4.2 Recursos hardware

Como recurso hardware tenemos el ordenador personal montado por piezas que costó unos 1.100€ hace 2 años. También tenemos un smartphone Xiaomi Redmi Note 5 Pro que costó 130€.

2.4.3 Recursos humanos

Para la realización de este proyecto se necesitaría un desarrollador de aplicaciones móviles con Flutter. Este trabajador tendría un sueldo de unos 25.000€ al año, y sería necesario que dedicara solamente 2 meses al proyecto.

2.4.4 Coste total

Descripción	Explicación	Coste total
Licencias	Ninguna licencia de pago requerida	0€
Hardware	Ordenador y smartphone Android	1.230€
Desarrollador Flutter	25.000€ / Año = 2.083 € / mes	4.166€
Coste total		5.396€

3. Análisis

3.1 Especificación de requisitos

En esta sección pondremos el foco de atención en todos los requisitos y condiciones que proyecto debe cumplir para conseguir cubrir todos los objetivos propuestos.

3.1.1 Requisitos funcionales

Gestión de usuarios

RF1	Descripción
Registro de usuario	Un usuario podrá registrarse en la aplicación rellenando un formulario con su email, contraseña y nick, automáticamente se creará una entidad correspondiente a ese usuario y se almacenarán esos datos

RF1	Descripción
Acceso de usuario	Siempre que un usuario esté previamente registrado en la aplicación, accederá a la aplicación mediante su email y contraseña

RF1	Descripción
Desconexión de usuario	Una vez el usuario ha accedido a la aplicación, dispondrá de un botón para desconectarse, se cerrará su sesión en la aplicación, mientras no se pulse, el usuario seguirá conectado aunque se cierre la aplicación

RF1	Descripción
Modificar nick	Un usuario podrá modificar su nick después de su registro en la aplicación

Gestión de actividades

RF1	Descripción
Visualización de actividades de usuario	El usuario podrá ver una lista con todas sus actividades realizadas, en orden cronológico, que aparecerá en la pantalla principal

RF1	Descripción
Crear actividad	Se creará un formulario con nombre, descripción y hora de inicio, cuando se envíe se procederá al seguimiento vía GPS de la actividad

RF1	Descripción
Borrar actividad	Al mantener pulsado sobre una actividad aparecerá una alerta dándose las opciones de borrar la actividad o cancelar borrado

RF1	Descripción
Visualizar actividad	Cuando se pulse sobre una actividad, se verán el nombre, descripción(si la hubiere), hora del inicio, así como distintas métricas y un mapa donde se pueda ver qué recorrido ha realizado el usuario

RF1	Descripción
Seguimiento de una actividad	Cuando se inicie el seguimiento se empezará a obtener la localización del dispositivo, que será refrescada cada segundo, con la precisión máxima disponible en el dispositivo. Cuando lo desee, el usuario podrá acabar esta actividad

RF1	Descripción
Almacenaje de datos geográficos y métricas	Los datos recopilados durante el seguimiento de una actividad serán: latitud, longitud y marca de tiempo. En base a estos datos se calcularán unas métricas determinadas que luego se mostrarán. Los datos se almacenarán en la base de datos una vez haya terminado la actividad

Gestión de grupos

RF1	Descripción
Visualización de grupos de usuario	El usuario podrá ver una lista con todos los grupos a los que pertenece, que aparecerá en la pantalla principal

RF1	Descripción
Crear grupo	Cualquier usuario podrá crear un grupo y añadir los miembros que desee en el momento de la creación. Este usuario será por defecto Administrador, y los añadidos, si hubiere, Miembros

RF1	Descripción
Borrar grupo	Una vez se pulse el botón de borrar grupo, se creará una alerta que lo notifique, una vez se acepte, se borrará el grupo

RF1	Descripción
Añadir miembro a un grupo	Un usuario Administrador de un grupo determinado podrá añadir todos los miembros que desee a ese grupo introduciendo los emails de los mismos

RF1	Descripción
Eliminar miembro de un grupo	Un usuario Administrador de un grupo determinado podrá eliminar a todos los miembros que desee de ese grupo introduciendo los emails de los mismos

RF1	Descripción
Administrar permisos de los usuarios del grupo	Un usuario Administrador de un grupo determinado podrá modificar el status de cualquier miembro de ese grupo que desee

Gestión de las comparaciones

RF1	Descripción
Visualización de comparaciones realizadas por el usuario	Un usuario podrá ver todas las comparaciones que haya realizado

RF1	Descripción
Añadir comparación	Un usuario podrá crear comparaciones, usando para ello las actividades de miembros de un grupo en el que él esté

RF1	Descripción
Eliminar comparación	Un usuario podrá eliminar cualquiera de sus comparaciones

RF1	Descripción
Visualizar comparación	El usuario podrá ver con gráficos la evolución temporal del rendimiento de uno o varios usuarios que haya decidido comparar

3.1.2 Requisitos no funcionales

RNF1	Descripción
Seguridad de los datos	Ante un posible fallo en el almacenaje del sistema, no deben perderse los datos previamente almacenados

RNF1	Descripción
Disponibilidad	Todos los servicios deben de estar operativos un 99, 9% del tiempo como mínimo, independientemente de la localización geográfica, el único requisito para usar la aplicación es que se tenga acceso a internet

RNF1	Descripción
Encriptación de los datos	Toda la información sensible de los usuarios debe estar debidamente almacenada y encriptada

RNF1	Descripción
Restricciones de implementación	Se debe usar un SDK compatible con el desarrollo de aplicaciones para Android

RNF1	Descripción
Permisos de usuario	Los permisos que existirán en los grupos para los distintos usuarios serán Administrador y Miembro, el administrador podrá añadir y eliminar usuarios, así como cambiar los permisos de los demás usuarios

3.2 Modelos de casos de uso

Para explicar de forma más clara y visual el comportamiento que tendrá el sistema de cara al usuario, usaremos un diagrama de casos de uso.

3.2.1 Diagrama de casos de uso

3.2.1.1 Actores

En este caso hablaremos de tan solo un actor, el usuario final. El usuario final será el único encargado de hacer llamadas a todas las funcionalidades de la aplicación, así como de aportar información a la misma.

3.2.1.2 Diagrama

A continuación se puede ver el diagrama en el que se visualiza la forma en la que el usuario puede interactuar con la aplicación.

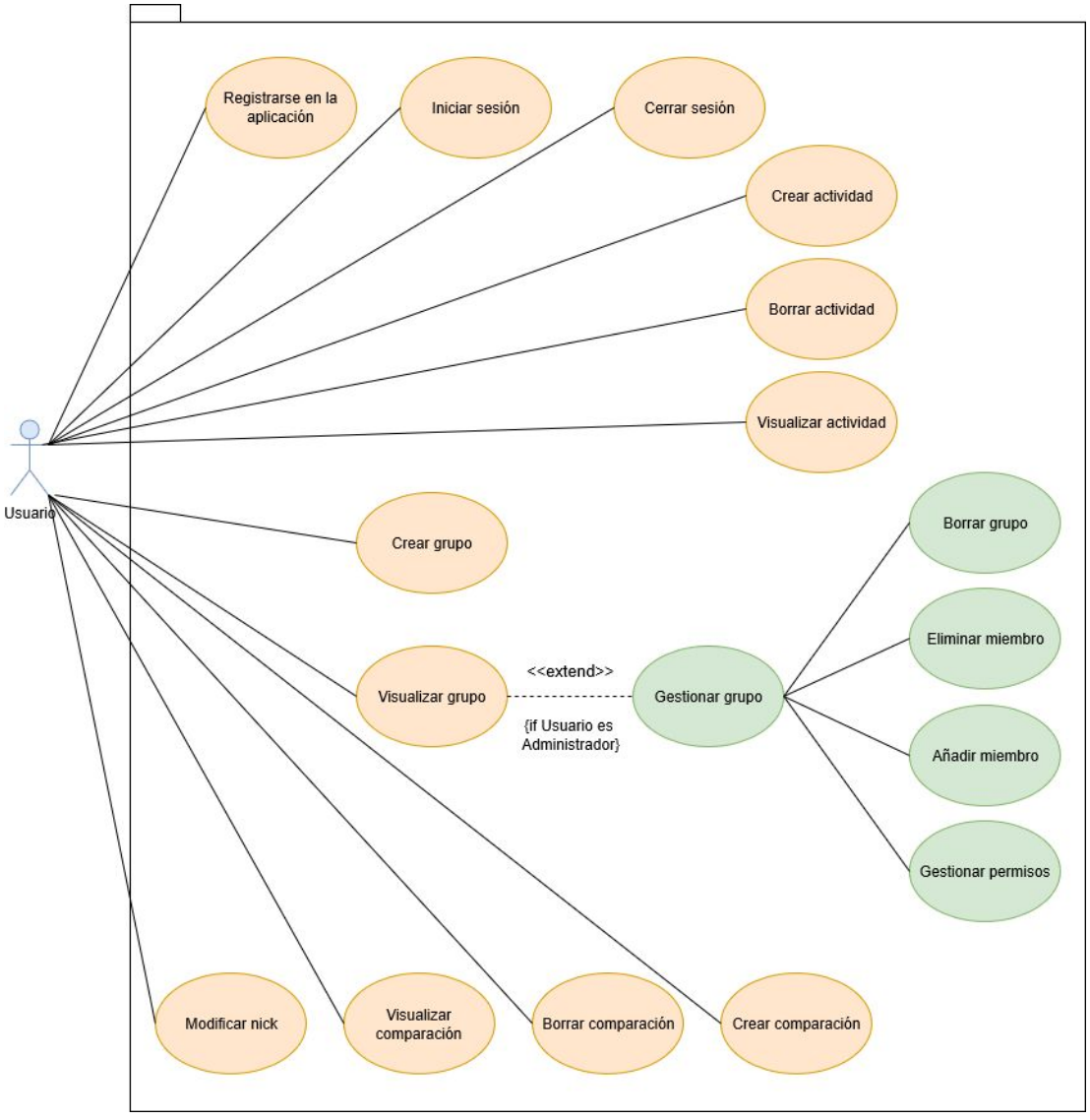

Figura 2 Diagrama de casos de uso

4. Diseño

4.1 Diseño de la base de datos

Sin saber si se utilizaría una base de datos relacional o no, se diseñó la estructura de la información en la base de datos para ser posible su implementación en ambos casos, y que viene dada en el siguiente diagrama:

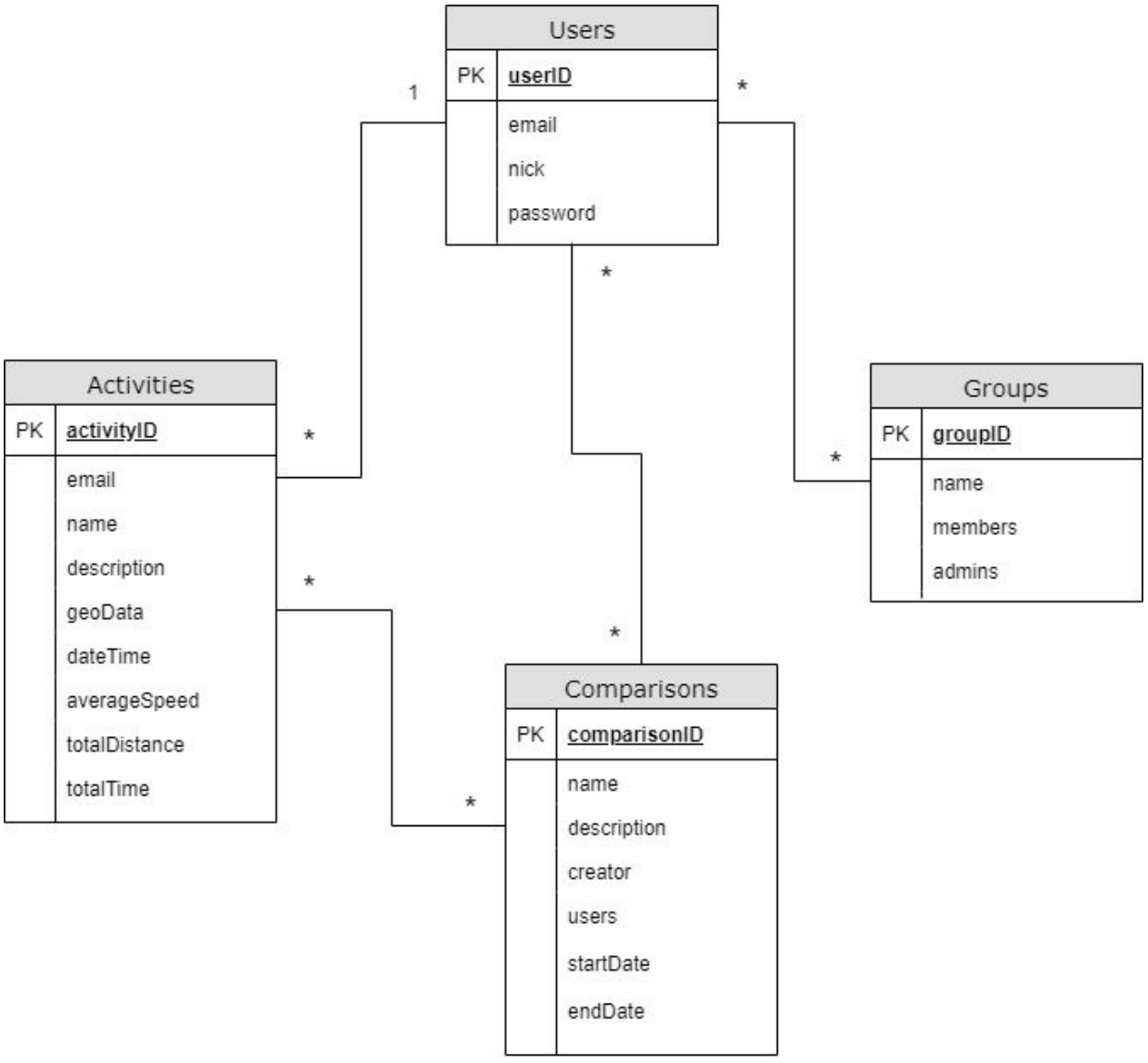

Figura 3 Estructura de la base de datos

Donde cada una de estas cuatro entidades es una colección de elementos del mismo tipo. Cada objeto de una colección concreta tendrá los campos que aparecen en la entidad correspondiente.

La tabla usuarios tendrá la información relativa a estos. Un usuario podrá estar en varios grupos al mismo tiempo y tener varias actividades y comparaciones realizadas. Un grupo se compone de varios usuarios. Una actividad puede tener un único usuario asociado, su creador. Una comparación podrá tener varias actividades a comparar en ella, y a su vez, una actividad podrá ser analizada en varias comparaciones.

4.2 Arquitectura del sistema

Una vez completados los requisitos y elegidas las herramientas con las que se trabajaría, se empezó a diseñar la estructura que tendría la aplicación. Para ello se creó el siguiente diagrama con la herramienta SiteMap de MockFlow [8]:

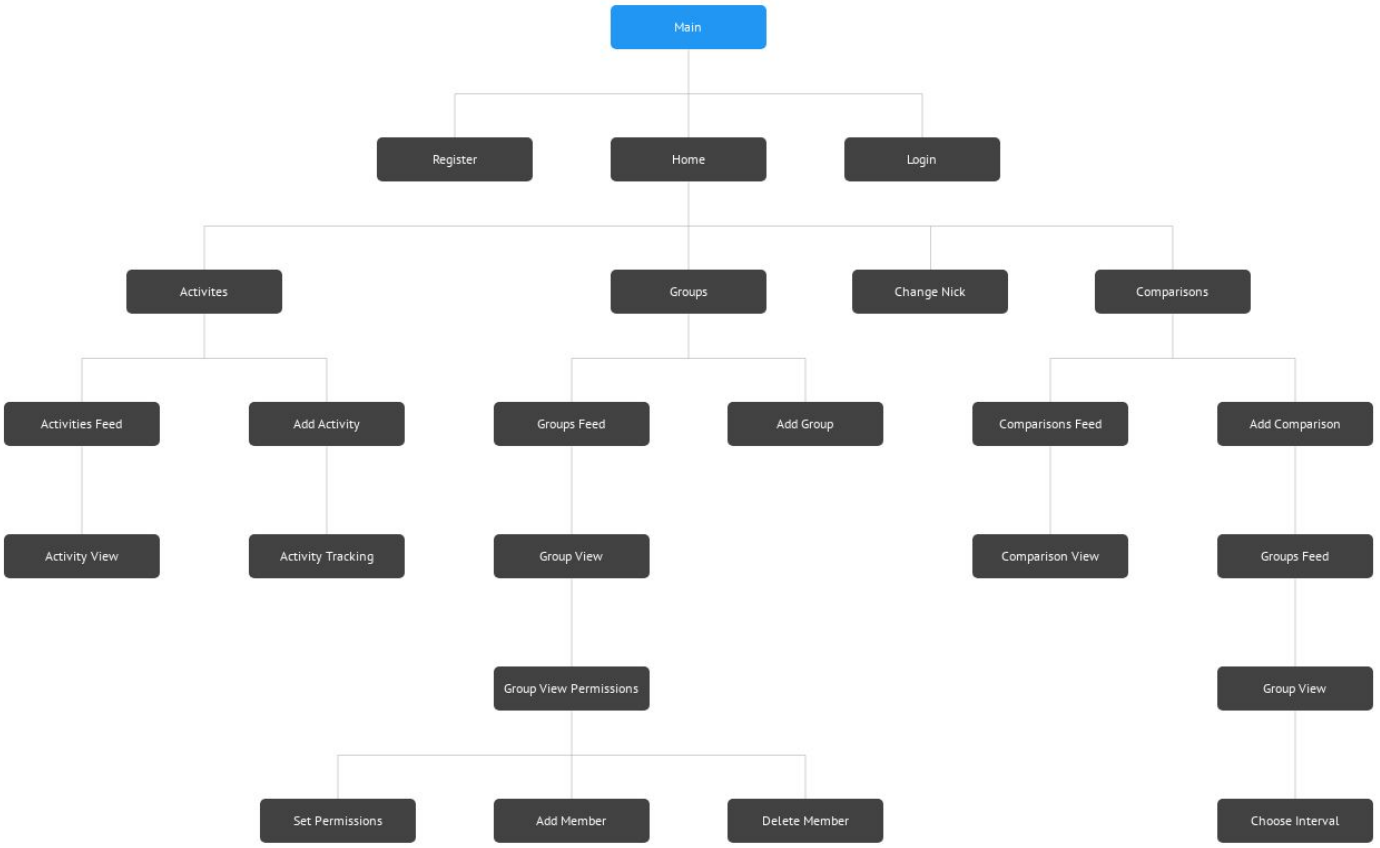

Figura 4 Vista lógica de la aplicación

Este diagrama muestra las funcionalidades y componentes de la aplicación, facilitando así la descomposición de la misma en módulos para así tener clara la estructura del desarrollo.

Una vez iniciada la aplicación se mandará al usuario a la página de autenticación si es la primera vez que entra o si se desconectó anteriormente. Si el usuario permanecía conectado entonces se le mostrará directamente la vista principal de la aplicación, que tiene como predeterminada la vista de Actividades.

Una vez en la página principal, el usuario podrá moverse entre las distintas secciones de Actividades, Grupos o Comparaciones, o bien entrar en los Ajustes de la aplicación. En este caso sólo se ha pensado en el cambio de nick del usuario.

El usuario tendrá acceso a todas las funcionalidades de la sección en la que se encuentre, ya sea Actividades, Grupos o Comparaciones. Las dos acciones primarias que podrá realizar el usuario que esté en alguna de estas 3 páginas serán las de ver en detalle un elemento o crear un nuevo elemento de esta sección.

4.3 Diagrama de clases

Lo primero que hay que hacer en esta parte es elaborar el diagrama de clases donde se pondrán de forma clara los atributos y operaciones que tendrá cada clase del sistema.

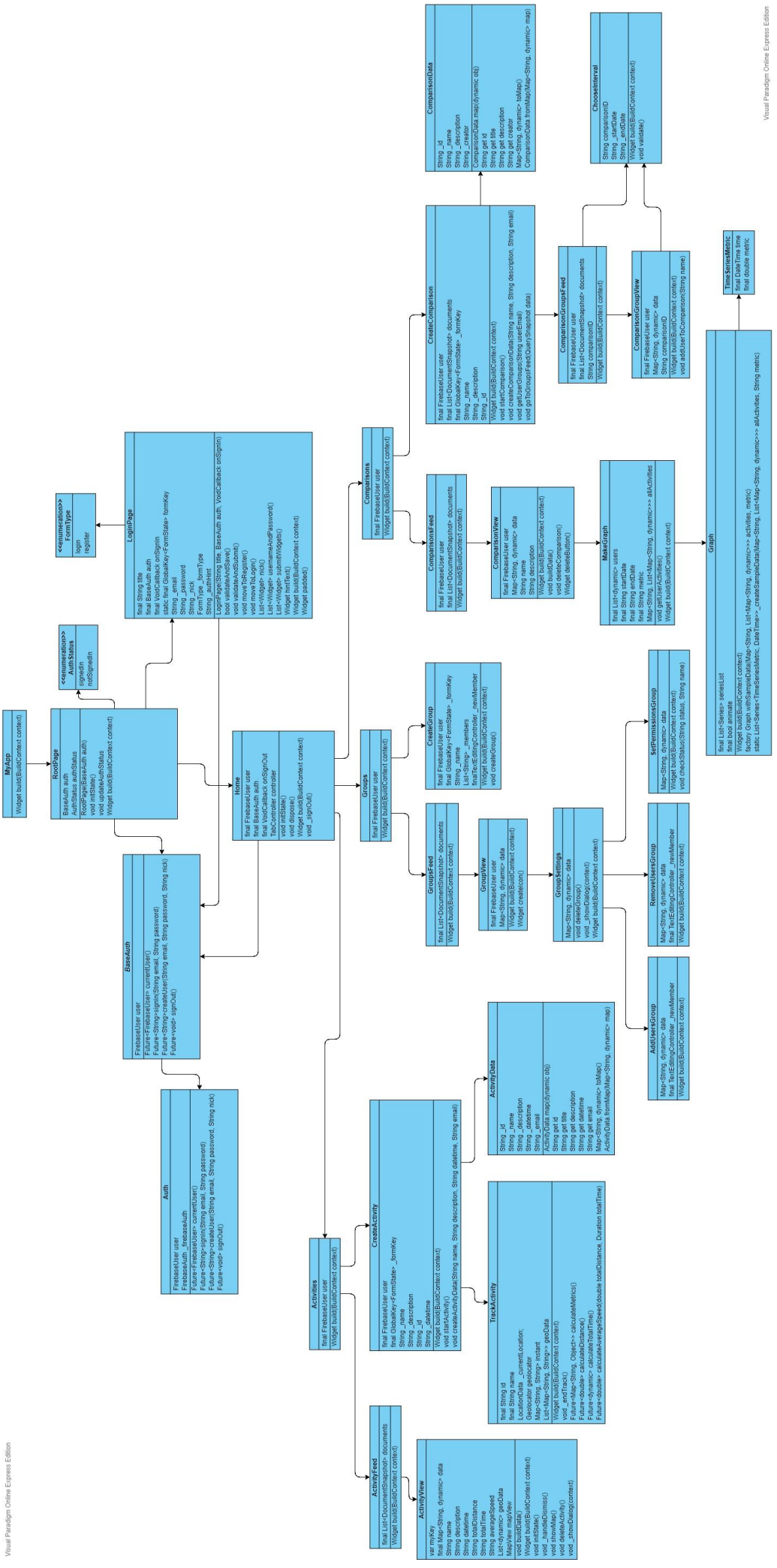

Figura 5 Diagrama de clases

En este caso todas las clases usan en alguno de sus métodos una instancia de la clase a la que señalan(salvo que el objetivo sea un enumerado, entonces usan éste).

A continuación se explican las funciones de cada clase:

-RootPage: se encarga de ver si el usuario estaba previamente autenticado y en base a esto le envía a la página principal o bien a la página de autenticación.

-BaseAuth: clase abstracta que define el comportamiento de Auth, la clase encargada de la autenticación.

-LoginPage: se encarga de gestionar las páginas de autenticación y registro del usuario.

-Home: gestiona la página principal, en la que habrá 3 pestañas para cada una de las 3 secciones.

-Activities, Groups y Comparisons: se encargan de recopilar los datos del usuario desde la base de datos para que las clases del nivel inferior los presenten.

-ActivityFeed, GroupsFeed, ComparisonsFeed: se encargan de mostrar una lista con todos los elementos que tenga el usuario en esta sección.

-ActivityView, GroupView, ComparisonView: muestran en detalle el elemento seleccionado en la feed.

-CreateActivity, CreateGroup, CreateComparison: crean un elemento básico de su clase, este proceso de creación puede continuar en sucesivas clases, como en el caso de las Actividades o Comparaciones o quedar finalizado, como en el caso de los Grupos.

-TrackActivity: cuando se crea una actividad entra en juego esta clase, encargada de realizar el seguimiento por GPS del dispositivo hasta que se dé por finalizada la actividad.

-MakeGraph, Graph y TimeSeriesMetric: clases encargadas de la construcción y visualización de las gráficas para que muestren la evolución en el tiempo de las métricas de la comparación que se está mostrando.

-GroupSettings: se encarga de mostrar la página de ajustes de grupo.

-AddUsersGroup, RemoveUsersGroup, SetPermissionsGroup: muestran las páginas que añaden, eliminan y gestionan los permisos de los usuarios de un grupo y actualizan la información en la base de datos.

-ActivityData, ComparisonData: clases auxiliares para realizar operaciones con la base de datos.

-ComparisonGroupsFeed, ComparisonGroupView: clases similares a GroupsFeed y GroupView, modificadas para realizar las funciones necesarias que precisan las comparaciones.

-TimeSeriesMetric: clase auxiliar para la construcción de las series temporales de los gráficos de las comparaciones.

4.4 Diseño de interfaces de usuario

Para facilitar la maquetación de la aplicación se diseñaron inicialmente mockups para las interfaces de usuario, en los que se intentó conseguir cumplir los objetivos explicados a continuación. Para ello se utilizó la herramienta gratuita WireframePro de MockFlow.

Todas las interfaces debían tener un diseño limpio y claro, con los mínimos componentes posibles, para así ser lo más intuitivas posibles. La clave pretendía ser la simpleza. Por consiguiente toda la aplicación tiene la misma gama cromática en la barra de aplicación y secciones, así como los botones flotantes. El objetivo es que un usuario que entre por primera vez a la aplicación no tenga problemas para manejarla y descubra las funcionalidades que brinda en el instante en el que se le presentan.

A continuación se muestran los mockups de las distintas interfaces:

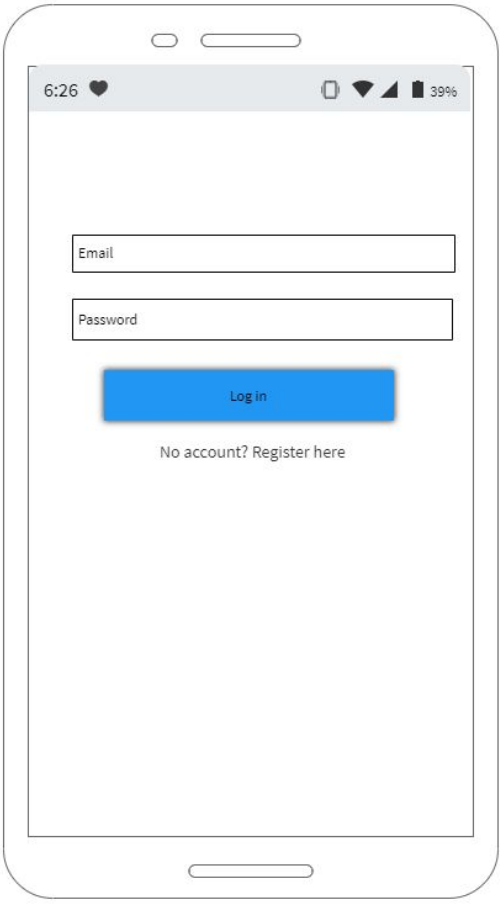

Figura 6 Interfaz de inicio de sesión

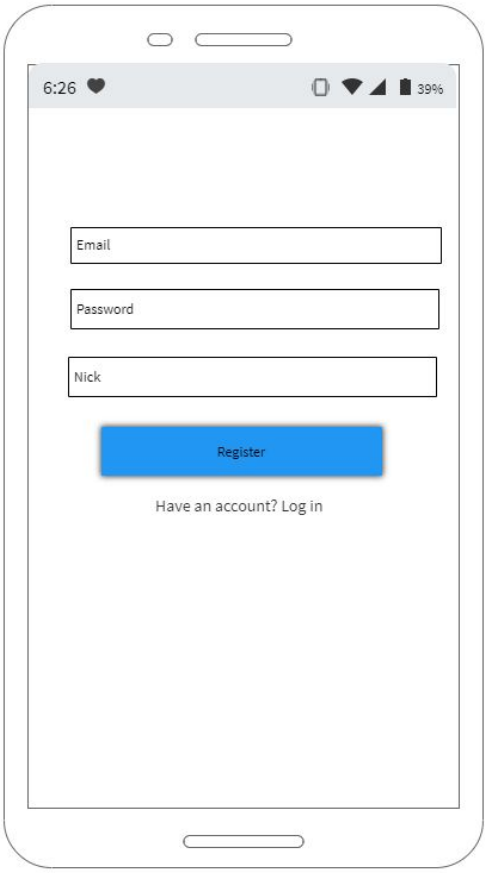

Figura 7 Interfaz de registro de usuario

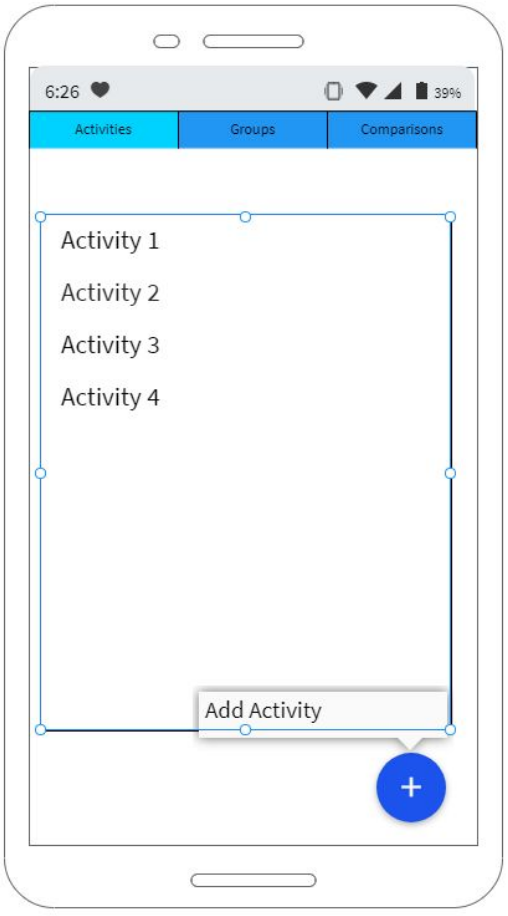

Figura 8 Interfaz de la sección de actividades

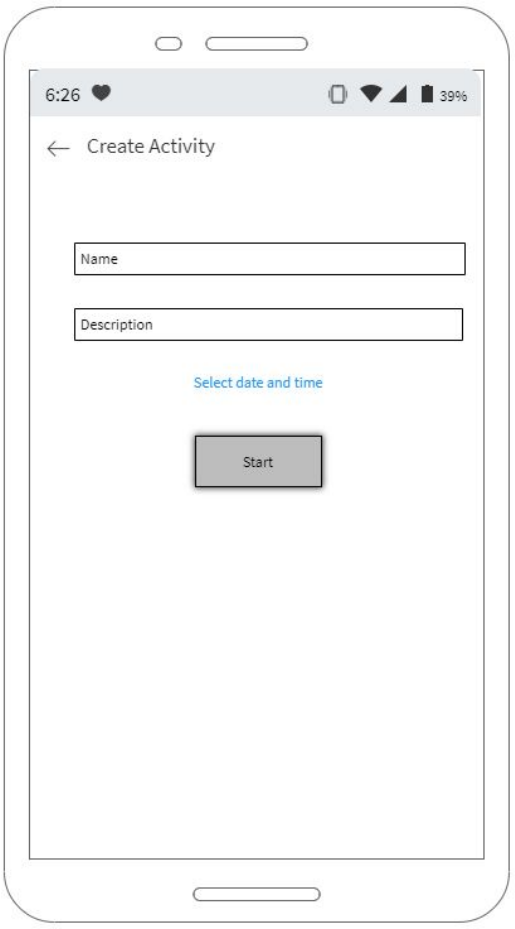

Figura 9 Interfaz de creación de actividad

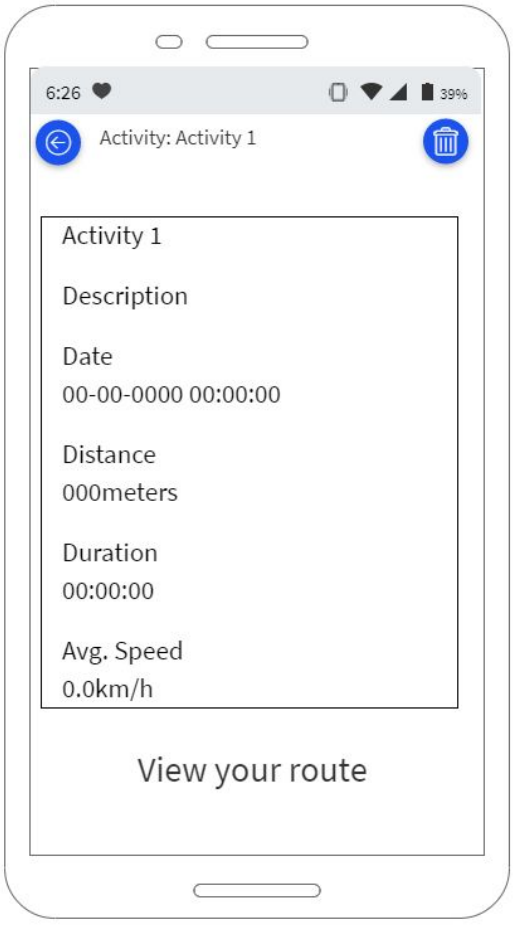

Figura 10 Interfaz de vista de actividad

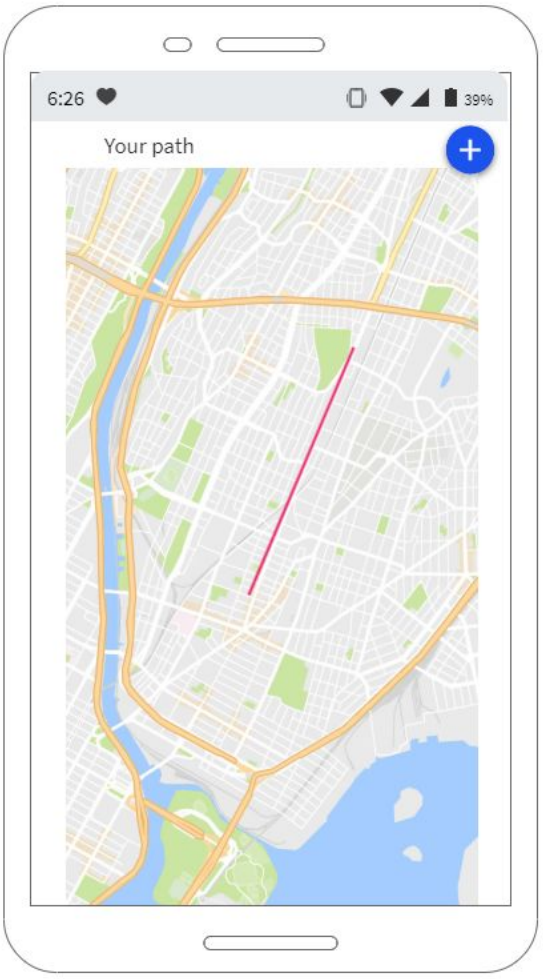

Figura 11 Interfaz de visualización de la ruta de la actividad

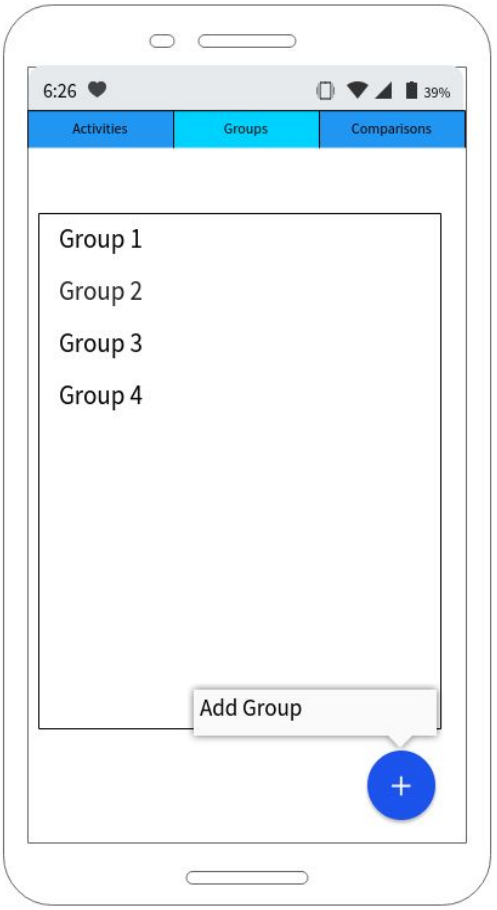

Figura 12 Interfaz de la sección de grupos

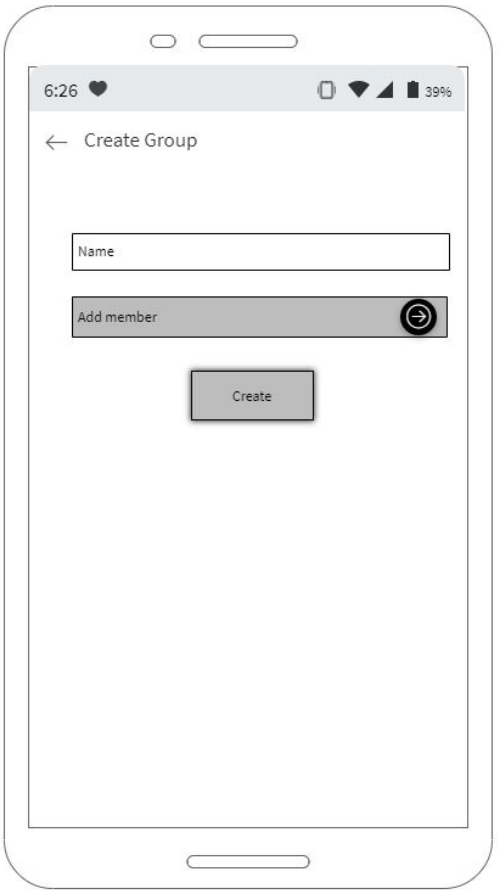

Figura 13 Interfaz de creación de grupo

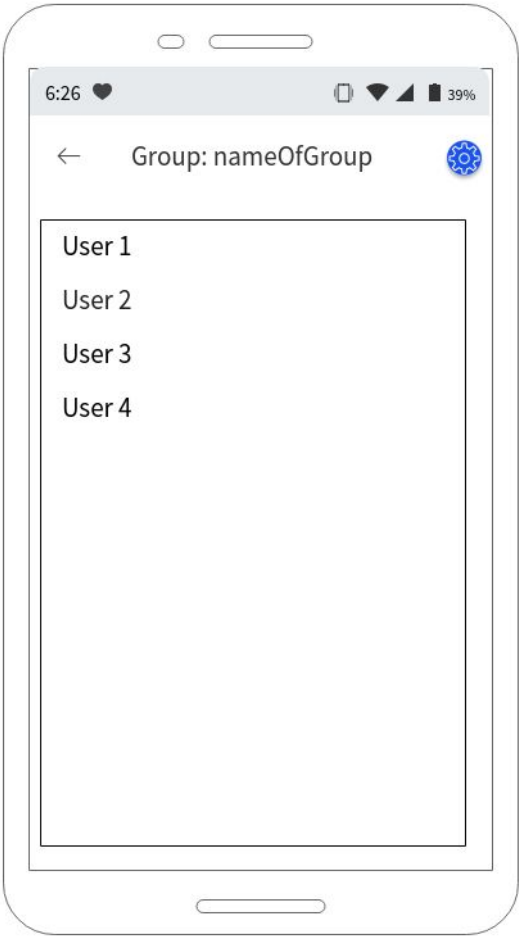

Figura 14 Interfaz de vista de grupo

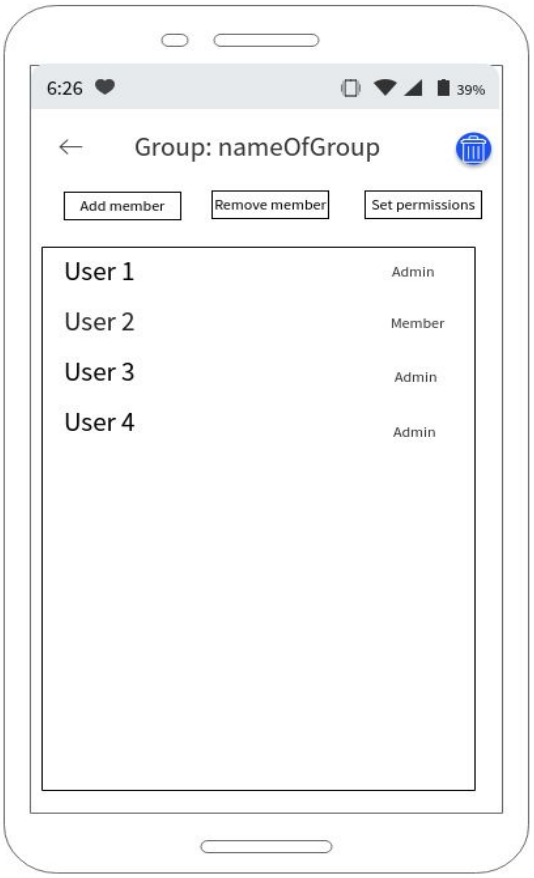

Figura 15 Interfaz de ajustes de grupo

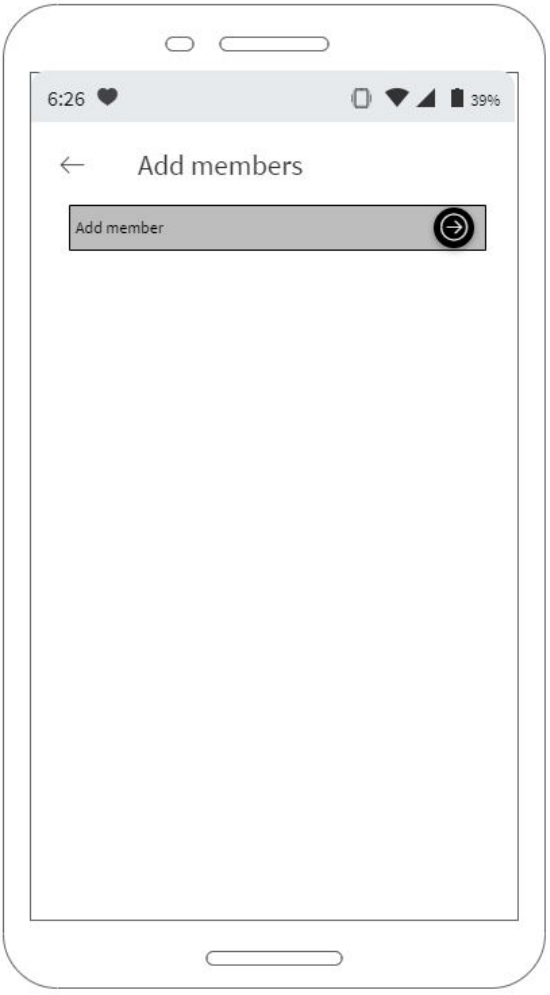

Figura 16 Interfaz para añadir miembros a un grupo

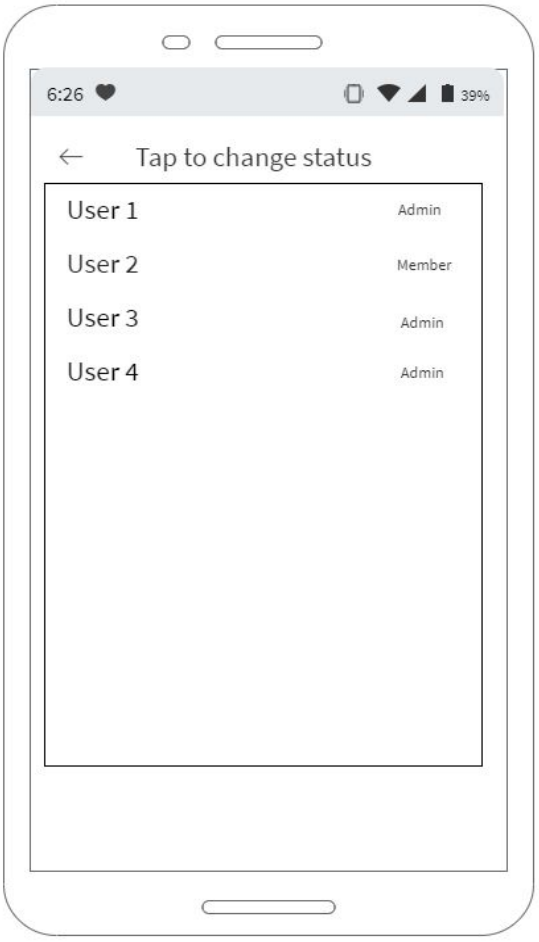

Figura 17 Interfaz de gestión de permisos de un grupo

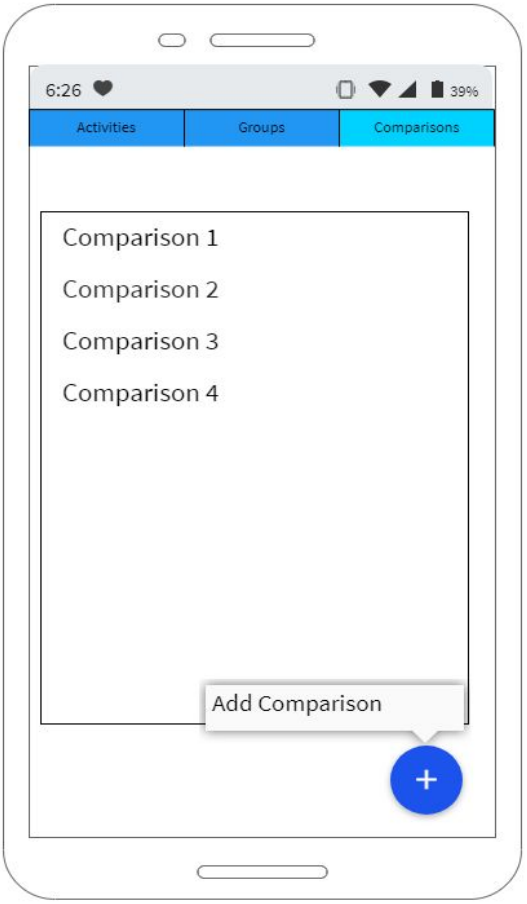

Figura 18 Interfaz de la sección de comparaciones

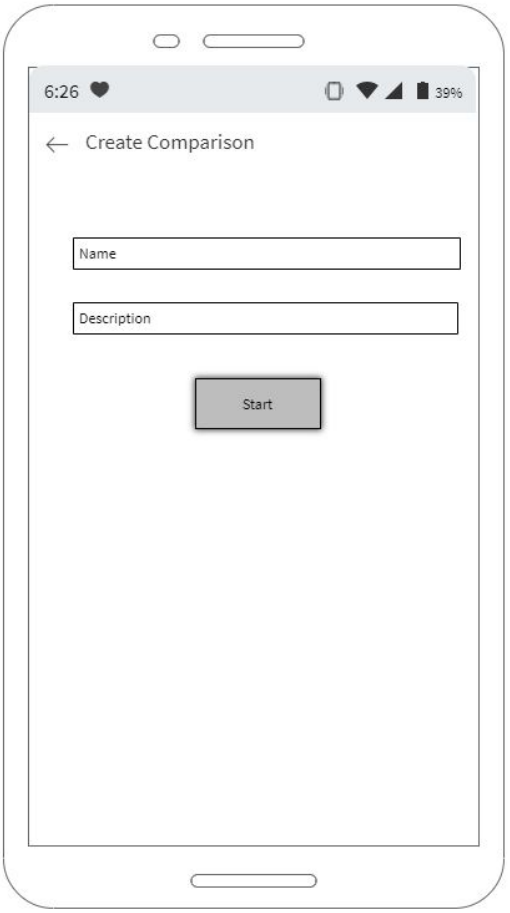

Figura 19 Interfaz de creación de una comparación

5. Implementación

En esta fase se presentan las dos grandes decisiones del proyecto, pues hay que elegir las herramientas que se usarán para llevar a cabo la implementación de la aplicación. Aquí habrá que tomar dos decisiones: una referente al framework y lenguaje de programación que se usará para el desarrollo, y otra sobre la implementación de la base de datos.

En lo que respecta al framework, se considerará con qué programa se va a desarrollar la aplicación, decidiendo entre Flutter o Android Studio, tras haber analizado ambos entornos.

Por otro lado, en lo respectivo a la implementación de la base de datos, se tendrá en consideración si crear y administrar una propia o si, por el contrario, utilizar el servicio administrado por Google, Firebase Firestore.

5.1 Elección de lenguajes de programación y herramientas de desarrollo

Para escoger con qué framework se va a desarrollar la aplicación, se han estudiado tanto Flutter como Android Studio, y se ha llegado a las siguientes conclusiones.

Flutter es un novedoso framework desarrollado por Google que ha sido diseñado para crear aplicaciones con interfaces de usuario atractivas y adaptables, compiladas de forma nativa para dispositivos móviles, web y de escritorio desde un solo código base. Usa el lenguaje de programación Dart para dotar a las aplicaciones de una interfaz de usuario vistosa y flexible, obtener un rendimiento nativo y ofrecer un desarrollo rápido y ágil, usando para ello el *hot reload*.

Así pues, este entorno es un firme competidor de Android Studio, considerado por muchos como primera opción cuando se piensa en desarrollo de aplicaciones móviles para Android. Teniendo en cuenta este factor, se procedió a analizar ambas herramientas para poder decidir por cual se optaría para llevar a cabo la implementación.

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Desarrollar con esta herramienta hubiera significado programar en Java (un recurso comúnmente utilizado) e ir al terreno seguro y en parte conocido pero tedioso. Esta herramienta, robusta y consolidada en este ámbito, tiene casi 7 años de antigüedad, mientras que Flutter, con apenas 2 años, proporcionaría a este trabajo un reto que quería asumir ya que consideré que sería muy bueno abordarlo, de cara tanto a mi desarrollo personal como al profesional.

Quería ver cómo me desenvolvía yo solo, sin un guión de prácticas de por medio, sin unos apuntes del profesor que me enseñaran cómo aprender un lenguaje paso a paso. Navegando por YouTube encontraba vídeos de aplicaciones hechas en Flutter que me motivaban mucho, así que decidí coger el camino tortuoso pero con una alta recompensa antes que el standard en el que sería uno más de la lista.

Los puntos negativos que tenía esta opción eran principalmente el tiempo que se debería emplear en aprender todo lo necesario para desarrollar la aplicación y la poca madurez y robustez de Flutter. Hubo algunos problemas debidos a que no es una herramienta sólida con muchos años en el mercado: incompatibilidades, errores varios debidos a las

versiones de paquetes instalados o bien de herramientas necesarias para hacer funcionar Flutter, etc..

Aún así los puntos positivos superaban enormemente a los negativos y también a la otra opción de desarrollar con Android Studio, la forma "tradicional" de desarrollar aplicaciones móviles para Android

Finalmente se decidió desarrollar con Flutter, ya que esta opción era mucho más atractiva que la otra. Esto se debe a que Flutter [9] es un framework novedoso con mucho potencial, que permite el desarrollo de interfaces de usuario muy atractivas fácilmente. También tiene una fácil integración con Firebase Firestore y Firebase Auth, lo cual permite que la autenticación y el manejo de la base de datos sean muy simples. También suponía un reto académico interesante, que era el de aprender un nuevo framework, un lenguaje de programación y un paradigma de programación distintos a los aprendidos en la carrera.

Una vez escogido Flutter, la otra decisión sería la de qué base de datos escoger.

Aquí la primera opción fue Firebase Firestore [10], ya que la integración con Flutter era muy sencilla, el manejo sería simple y no supondría demasiados problemas, lo que me permitiría centrarme en otras partes del desarrollo antes que tener que estar peleándome con la base de datos. Además es un tipo de base de datos NoSQL orientada a documentos. Esto era un plus, ya que en la carrera y en mi trabajo en las Prácticas de Empresa había trabajado con bases de datos relacionales y quería probar cómo era trabajar con bases de datos no relacionales.

Además no tendría ningún coste utilizar estos servicios ya que el volumen de operaciones realizadas por la base de datos sería muy inferior al límite de operaciones gratuito [11].

Otro punto positivo que me ayudaría mucho a desarrollar la aplicación con estas herramientas era la ingente cantidad de documentación [12], ejemplos y vídeos en YouTube sobre Flutter.
Recursos como el Cookbook [13], el Language Tour de Dart [14], Awesome Flutter [15] y el manejador de paquetes de Flutter [16] junto a los cientos de videos de YouTube que vi me hicieron adquirir los conocimientos necesarios para desarrollar la aplicación que necesitaba.

5.2 Frameworks, bibliotecas y herramientas utilizadas

El framework que se utilizó para el desarrollo de la aplicación, como se dijo antes, fue Flutter, en su versión 1.2.1.
El editor de código que se utilizó fue Visual Studio Code, ya que disponía de una muy buena integración con Flutter, haciendo el desarrollo más fácil y eficiente, haciendo que labores como el Hot-Reload, el manejo de Git y el formato del código fueran muy simples y rápidas.

Para usar Firebase se creó un proyecto con el plan gratuito que ofrece Google.Se realizó la integración de Firestore a nuestro proyecto de Firebase en Android, siguiendo la

documentación oficial [17] y diversos tutoriales de YouTube. La integración de Authentication [18] se realizó de la misma manera, optando por el método de autenticación basada en correo electrónico y contraseña.

Diversos paquetes de Flutter se usaron para la implementación, como *flutter_datetime_picker* [19] para la selección de fechas dentro de la aplicación, *flutter_spinkit* [20] para las animaciones, *geolocator* [21] y *location* [22] para la localización GPS, *screen* [23] para mantener la pantalla del dispositivo activa, *charts_flutter* [24] para hacer los gráficos de las comparaciones y *map_view* [25] para visualizar el recorrido realizado en el mapa.

5.3 Lenguajes de programación utilizados

El lenguaje de programación que se utilizó fue Dart 2.2, que es el lenguaje que se utiliza para desarrollar con Flutter. También se necesitó utilizar Kotlin para arreglar errores de incompatibilidades.

Dart es, según Google [26], un lenguaje optimizado para el cliente para el desarrollo de aplicaciones rápidas en múltiples plataformas. Aunque cuando se empezó a estudiar en este trabajo estaba destinado principalmente a aplicaciones móviles, ahora Google quiere expandir su uso hasta la Web, aplicaciones de escritorio e incluso dispositivos embebidos [27].

Es un lenguaje multiparadigma, con una sintaxis estilo C, hecho para ser rápido en todas las plataformas, optimizar la interfaz de usuario y desarrollar productivamente y eficientemente con él.

5.4 Implementación de la base de datos

Como se explicó anteriormente, en este proyecto se usó Firebase Firestore, una base de datos en la nube de tipo NoSQL orientada a documentos. A continuación se enseña la implementación final, que consiste en 4 colecciones de documentos correspondientes a cada una de las entidades que constan en el diseño de la base de datos. Estas colecciones tendrán documentos de una clase determinada, por ejemplo, en la clase actividades, tendremos todos los documentos correspondientes a las actividades. Cada documento tendrá un identificador autogenerado por la herramienta y tendrá toda la información que se plasmó en el diseño de esta base de datos.

Todos los valores almacenados en esta base de datos son cadenas de caracteres, pero a su vez, estas cadenas pueden estar organizadas en arrays o diccionarios (*maps*). Estos diccionarios (un ejemplo de uso en la aplicación es el campo geoData, de actividades) son implementados como diccionarios con un par clave/valor donde la clave será un entero y el valor será otro diccionario que contendrá un número determinado de claves con sus correspondientes valores asociados.

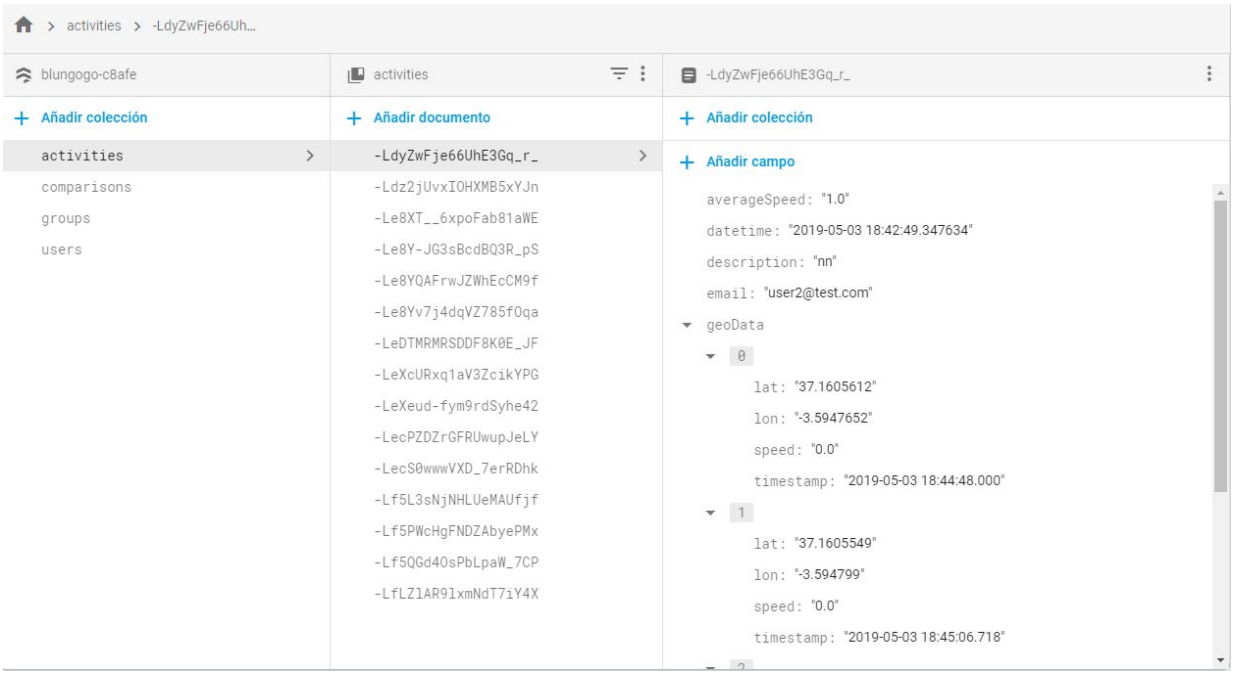

Figura 20 Colección de documentos de actividades

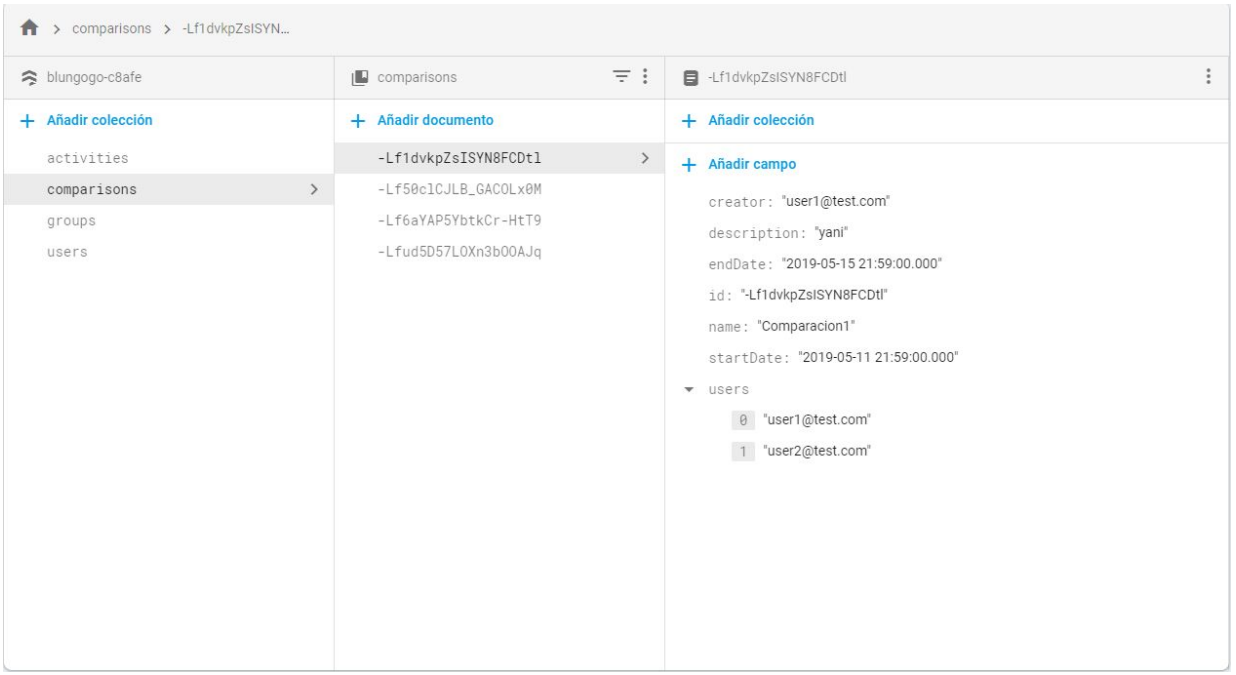

Figura 21 Colección de documentos de comparaciones

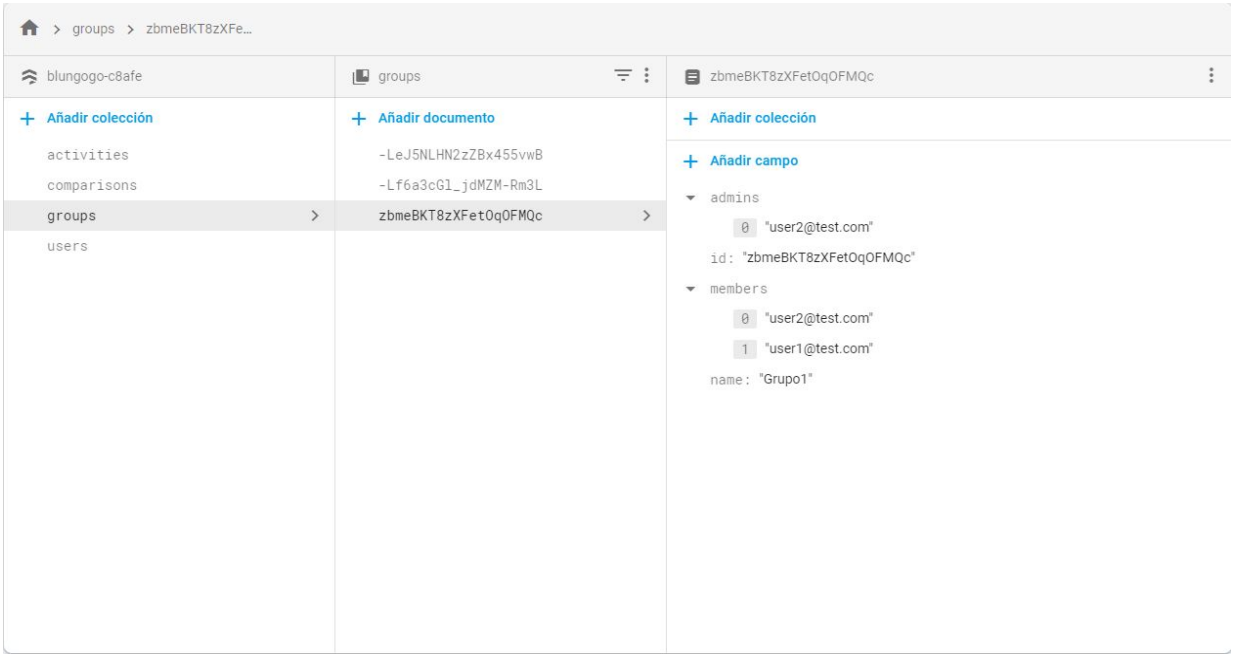

Figura 22 Colección de documentos de grupos

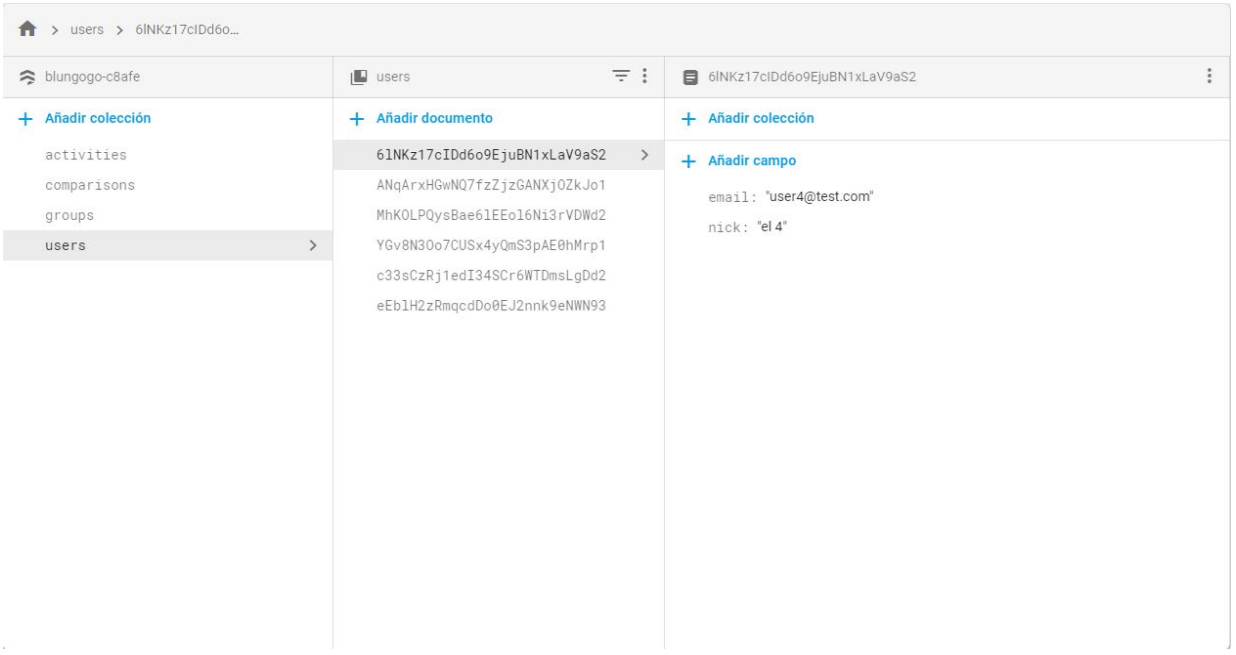

Figura 23 Colección de documentos de usuarios

5.5 Operaciones básicas con la base de datos

El manejo de Firestore es super sencillo, no hay que configurar nada de código para que funcione, el único código requerido es el necesario para ejecutar las operaciones, y este es muy simple. Seguidamente se explicará cómo realizar cada una de las operaciones básicas o CRUD (Create Read Update Delete).

Crear documento:

Seleccionamos la colección a la que queremos que pertenezca, le asignamos un identificador al documento(en caso de no asignarle identificador se genera uno automáticamente) e introducimos los datos en forma de Map<String, dynamic>(diccionario).

```
Firestore.instance
    .collection('users')
    .document(user.uid)
    .setData({'email': email, 'nick': nick});
```

Figura 24 Crear documento

Leer documento:

Seleccionamos la colección donde queremos buscar los documentos, filtramos con el método where y seleccionamos los snapshots correspondientes a los documentos que cumplen la condición.

```
Firestore.instance
.collection('activities')
.where('email', isEqualTo: widget.user.email)
.snapshots(),
```

Figura 25 Leer documento

Actualizar documento:

Seleccionamos la colección a la que pertenece el documento, introducimos su identificador y actualizamos los datos que deseemos. En caso de que estos datos no existieran previamente se añaden.

```
Firestore.instance
    .collection("comparisons")
    .document(widget.comparisonID)
    .updateData({"startDate": _startDate, "endDate": _endDate});
```

Figura 26 Actualizar documento

Borrar documento:

Para borrar un documento basta con seleccionar su colección y buscarlo por su identificador y borrarlo, usando el método .delete().

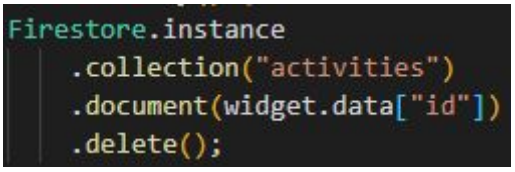

Figura 27 Borrar documento

5.6 Autenticación

A continuación se muestra la clase encargada de la autenticación. Esta clase usa FirebaseAuth. El uso de este sistema de autenticación es muy simple y requiere poco código.

Para crear un usuario se usa el método createUserWithEmailAndPassword(). Sólamente con pasar los parámetros de email y contraseña estaría creado en el sistema. La clave sería encriptada y almacenada adecuadamente. De todo esto se encarga Google.

Para iniciar la sesión de un usuario tendremos que usar el método signInWithEmailAndPassword() y pasarle los parámetros email y contraseña que el usuario haya introducido en nuestro formulario.

Para ver cuál es el usuario activo, en caso de que haya, se usa el método currentUser() y para desconectar a este usuario se usa el método signOut().

```dart
class Auth implements BaseAuth {
  final FirebaseAuth _firebaseAuth = FirebaseAuth.instance;

  Future<String> signIn(String email, String password) async {
    FirebaseUser user = await _firebaseAuth.signInWithEmailAndPassword(
        email: email, password: password);
    return user.uid;
  }

  Future<String> createUser(String email, String password, String nick) async {
    FirebaseUser user = await _firebaseAuth.createUserWithEmailAndPassword(
        email: email, password: password);
    Firestore.instance
        .collection('users')
        .document(user.uid)
        .setData({'email': email, 'nick': nick});
    return user.uid;
  }

  Future<FirebaseUser> currentUser() async {
    FirebaseUser user = await _firebaseAuth.currentUser();
    this.user = user;
    return user != null ? user : null;
  }

  Future<void> signOut() async {
    return _firebaseAuth.signOut();
  }

  @override
  FirebaseUser user;
}
```

Figura 28 Autenticación

5.7 Mostrar el recorrido de una actividad en el mapa

Para mostrar el recorrido que ha seguido un usuario en una actividad disponemos de una serie de instantes que han sido recopilados durante la realización de dicha actividad. Cada uno de estos instantes tendrá una latitud, longitud y marca de tiempo.

Lo que se muestra en el siguiente código es la creación de la lista llamada _lines, que contiene todas las líneas que representan el movimiento del usuario entre un instante y el siguiente.

```
List<Polyline> _lines = List<Polyline>();
for (var i = 1; i < geoData.length - 1; i++) {
  Location start = Location(double.parse(geoData.elementAt(i - 1)["lat"]),
      double.parse(geoData.elementAt(i - 1)["lon"]));
  Location end = Location(double.parse(geoData.elementAt(i)["lat"]),
      double.parse(geoData.elementAt(i)["lon"]));
  List<Location> line = List<Location>();
  line.add(start);
  line.add(end);
  Polyline polyline = Polyline(i.toString(), line);
  _lines.add(polyline);
}
mapView.setPolylines(_lines);
```

Figura 29 Generación de la línea que muestra el recorrido

5.8 Construcción de los gráficos de las comparativas

Para construir este gráfico disponemos de un Map<String, List<Map<String, dynamic>>>, esto es, un diccionario donde las claves son los nombres de los usuarios que existen en la comparación actual y los valores asociados son listas con todas las actividades de esos usuarios en el periodo indicado por la comparativa.

Estos datos se procesan para cada usuario creando así una serie temporal de los valores de una determinada métrica para un determinado usuario en el tiempo dictado por la comparativa.

Las series creadas tendrán como identificador el nombre del usuario, un color asignado en función de su posición en la lista de usuarios, un dominio y recorrido común para ese gráfico y los datos correspondientes a la métrica evaluada para ese usuario.

```
static List<charts.Series<TimeSeriesMetric, DateTime>> _createSampleData(
    Map<String, List<Map<String, dynamic>>> allActivities, String metric) {
  // hacer N listas que sean el nombre de cada user y de valor la lista de actividades
  // para cada lista recorrerla haciendo new TimeSeriesMetric(new DateTime(2017, 9, 19), 5),

  List<String> userNames = allActivities.keys.toList();
  List<List<Map<String, dynamic>>> activitiesByUser =
      allActivities.values.toList();
  List colors = [···

  List<charts.Series<TimeSeriesMetric, DateTime>> series = [];
  for (var i = 0; i < allActivities.length; i++) {
    List<TimeSeriesMetric> data = [];

    for (var j = 0; j < activitiesByUser[i].length; j++) {
      if (metric != "totalTime") {
        data.add(TimeSeriesMetric(activitiesByUser[i][j]["datetime"],
            double.parse(activitiesByUser[i][j][metric])));
      }
      if (metric == "totalTime") {
        String totalTimeString = activitiesByUser[i][j][metric];
        //digit of hour
        double hours = double.parse(totalTimeString.substring(0, 1));
        // 2 digits of minutes
        double minutes = double.parse(totalTimeString.substring(2, 4));
        // 2 digits of seconds
        double seconds = double.parse(totalTimeString.substring(5, 7));
        //gets total time in minutes
        double totalTimeDouble = (hours * 60) + minutes + (seconds / 60);
        data.add(TimeSeriesMetric(
            activitiesByUser[i][j]["datetime"], totalTimeDouble));
      }
    }
    if (data.isNotEmpty) {
      charts.Series<dynamic, dynamic> serie =
          new charts.Series<TimeSeriesMetric, DateTime>(
        id: userNames[i],
        colorFn: (_, __) => colors[i],
        domainFn: (TimeSeriesMetric metric, _) => metric.time,
        measureFn: (TimeSeriesMetric metric, _) => metric.metric,
        data: data,
        displayName: userNames[i],
      );

      series.add(serie);
    }
  }
  return series;
}
```

Figura 30 Construcción de las series temporales de los gráficos

6. Conclusiones y vías futuras

Una vez finalizado este proyecto, se procede a presentar las conclusiones obtenidas como resultado de la finalización del mismo. En este capítulo se va a realizar un resumen de los objetivos conseguidos y se propondrán posibles mejoras y ampliaciones que podrían realizarse en el futuro para esta aplicación.

6.1 Conclusiones

Se han conseguido realizar todos los objetivos obligatorios del proyecto y no se ha conseguido realizar el único opcional. Esto ha sido debido a la falta de tiempo en el tramo final de este trabajo, lo que ha imposibilitado añadir estas características a la aplicación.

No obstante, el haber conseguido cumplir todos los objetivos marcados es motivo de alegría, ya que llegué a pensar que no se podría finalizar el trabajo a tiempo, debido a que me vi obligado a matricularme el segundo cuatrimestre de asignaturas con las que no contaba, haciendo que el tiempo del que disponía se viera mermado en gran medida.

Estoy muy satisfecho con este trabajo, ya que no sólo lo he podido acabar a tiempo, que ya para mi hubiera sido un logro, además he aprendido desarrollo de aplicaciones móviles en este caso con Flutter, lo cual me parece de mucho interés ya que el trabajo de Google con esta herramienta es muy prometedor. Antes no tenía ni idea de este ámbito, pero haber realizado este trabajo satisfactoriamente me ha brindado una visión más abierta de la ingeniería informática, ha puesto a prueba lo que he aprendido en esta carrera y me ha hecho, en cierta medida, salir del cascarón, apañármelas yo sólo y aprender a manejar un framework y un lenguaje de programación de manera totalmente autodidacta. Tener que hacer algo que no se parece a nada de lo que haya hecho anteriormente y buscarme la vida para cumplir los objetivos que se han fijado. Considero que esto ha sido el broche de oro al final de una carrera tan larga y tan dura como esta.

6.2 Vías futuras

A pesar de haber cumplido con todos los objetivos obligatorios que se marcaron desde el principio, esta aplicación tiene a un margen de mejora enorme, pudiendo añadir multitud de funcionalidades tales como:

- Implementar el último objetivo: añadir métricas avanzadas para de alto interés para los entrenadores, que permitan analizar en función de mesociclos y microciclos.

- Añadir más funcionalidades sociales, tales como la inclusión de amigos o añadirle fotos de perfil a los usuarios.

- Crear una versión para wearables así como una versión web de la aplicación.

- Implementar un sistema de visualización y planificación de actividades en el calendario.

- Creación de una nueva métrica personalizada: el rendimiento, basada en todas las métricas, que los entrenadores podrán ajustar como ellos mejor estimen.

7. Bibliografía

[1] www.scielosp.org/scielo.php?pid=S0102-311X2003000700011&script=sci_arttext
[2] https://core.ac.uk/reader/39140933
[3] https://recyt.fecyt.es/index.php/retos/article/view/66628/42191
[4] https://play.google.com/store/apps/details?id=com.runtastic.android
[5] https://play.google.com/store/apps/details?id=com.strava
[6] https://en.wikipedia.org/wiki/Agile_software_development
[7] https://en.wikipedia.org/wiki/Waterfall_model
[8] https://mockflow.com
[9] https://flutter.dev/
[10] https://firebase.google.com/docs/firestore/?hl=es-419
[11] https://firebase.google.com/pricing/?hl=es-419
[12] https://flutter.dev/docs
[13] https://flutter.dev/docs/cookbook
[14] https://dart.dev/guides/language/language-tour
[15] https://github.com/Solido/awesome-flutter
[16] https://pub.dev/
[17] https://firebase.google.com/docs/firestore/quickstart?hl=es-419
[18] https://firebase.google.com/docs/auth/?hl=es-419
[19] https://pub.dev/packages/flutter_datetime_picker
[20] https://pub.dev/packages/flutter_spinkit
[21] https://pub.dev/packages/geolocator
[22] https://pub.dev/packages/location
[23] https://pub.dev/packages/screen
[24] https://pub.dev/packages/charts_flutter
[25] https://pub.dev/packages/map_view
[26] https://dart.dev/
[27] https://developers.googleblog.com/2019/05/Flutter-io19.html

8. Anexo

8.1 Manual de usuario

En este apartado se explica el uso de la aplicación pantalla por pantalla. Para usar esta aplicación es necesario disponer de conexión a internet y un smartphone Android.

Lo primero que aparecerá cuando instalemos la aplicación en nuestro dispositivo será la pantalla de inicio de sesión (Figura 31). Si ya tenemos una cuenta introduciremos las credenciales e iremos a la página de inicio. Si por lo contrario no disponemos de una cuenta, pulsaremos debajo del botón de Log in para ir a la pantalla de registro (Figura 32).

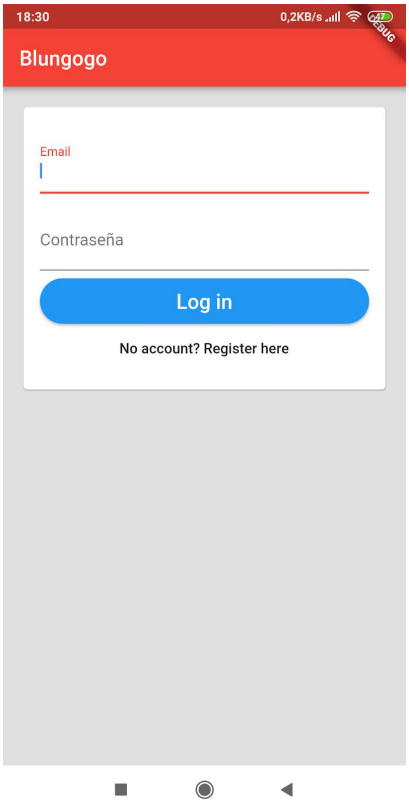

Figura 31 Inicio de sesión

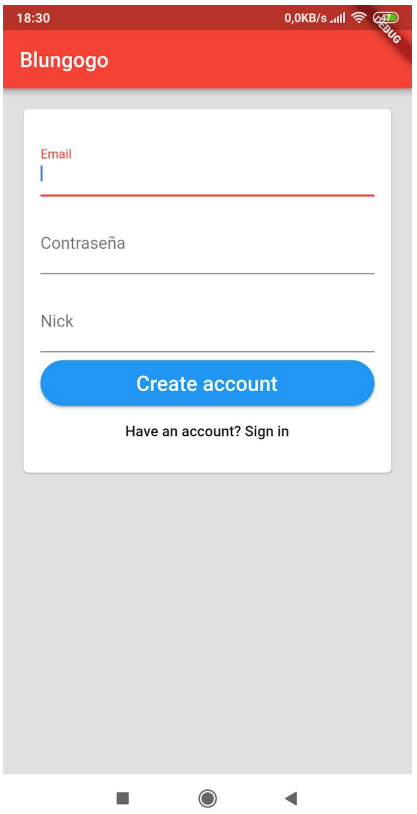

Figura 32 Registro

Una vez hayamos iniciado la sesión nos encontraremos con la página principal (Figura 33), donde se encuentran las 3 secciones principales. Cuando cerremos la aplicación y volvamos a entrar en ella ya no nos pedirá que iniciemos sesión, sólo tendremos que volver a hacerlo si nos desconectamos expresamente, pulsando el botón de la esquina superior derecha, o bien reiniciando nuestro dispositivo.

En esta pantalla nos saldrá por defecto la feed con todas nuestras actividades realizadas ordenadas por orden cronológico descendente. Aquí podremos realizar múltiples acciones como: desconectarnos de la aplicación, cambiar nuestro nick, navegar entre las distintas secciones, añadir una nueva actividad o ver una de nuestras actividades en detalle.

Para crear una actividad nueva, tenemos que pulsar el botón rojo de la parte inferior, esto nos llevará a la pantalla de creación de actividad (Figura 34), donde deberemos introducir, por lo menos, el nombre de ésta.

Una vez hayamos rellenado el formulario y pulsemos el botón para empezar, nos llevará a la siguiente pantalla (Figura 35) en la que se nos indicará que se está registrando nuestra actividad. Para finalizar la actividad pulsamos el botón y ya estará la nueva actividad disponible en nuestra feed.

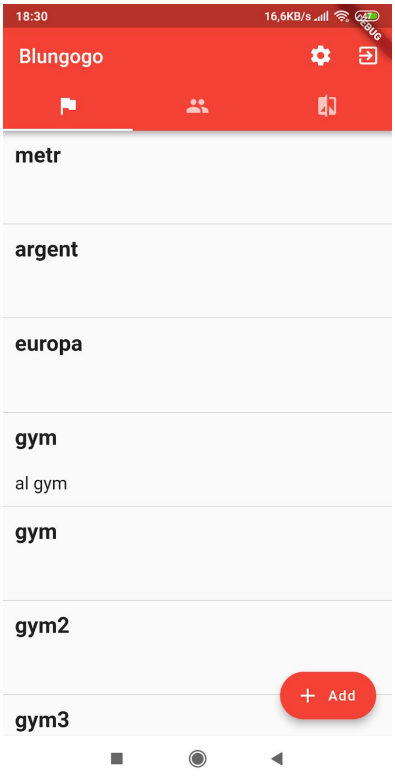

Figura 33 Feed de actividades

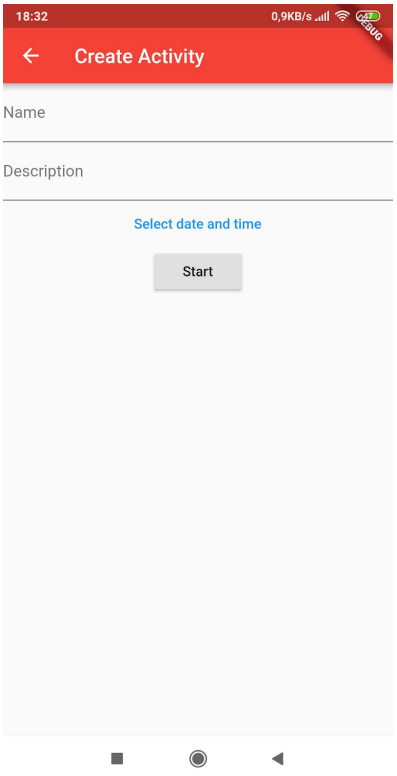

Figura 34 Crear actividad

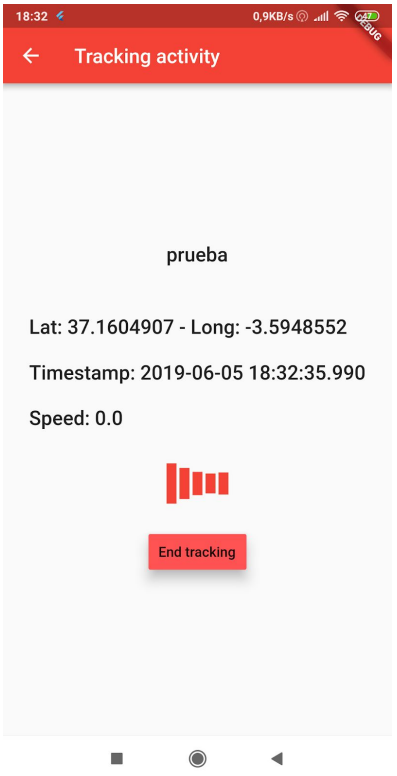

Figura 35 Seguimiento de actividad

Para ver una actividad basta con pulsar en la feed de actividades en la actividad que queramos observar en detalle. Tras hacer esto, nos aparecerá la siguiente pantalla, que nos mostrará los datos asociados a la actividad, así como las métricas derivadas de esta (Figura 36). También nos permitirá ver el recorrido que hicimos durante la actividad (Figura 37) y borrarla.

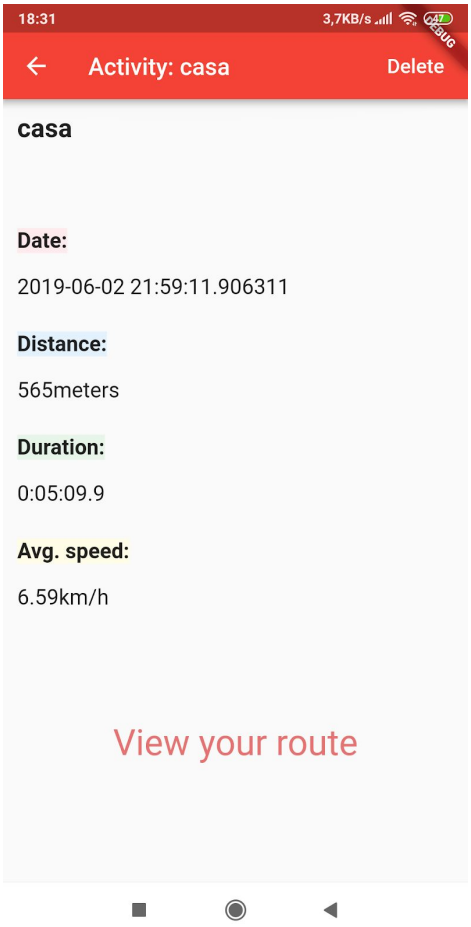

Figura 36 Vista de actividad

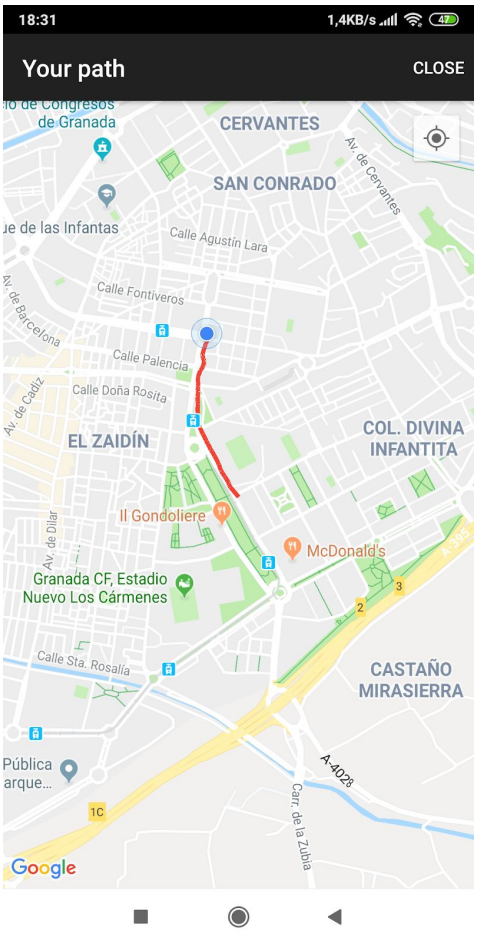

Figura 37 Vista de la ruta realizada en la actividad

Una vez vistas las actividades pasamos a la siguiente pestaña, los grupos (Figura 38). Aquí disponemos de prácticamente las mismas funcionalidades que en la pestaña de actividades, cambiando actividades por grupos.

Para crear un grupo pulsamos el botón de la parte inferior, entonces nos aparecerá la pantalla de creación de grupos (Figura 39). Para crearlo necesitamos introducir el nombre y antes de proceder a crearlo podemos introducir a los usuarios que deseemos, introduciendo sus emails en el campo de texto Add Member, una vez se ponga el nombre y se pulse en el botón del campo, se añadirá ese usuario y se limpiará el campo. Cuando consideremos que está todo hecho pulsamos el botón Create.

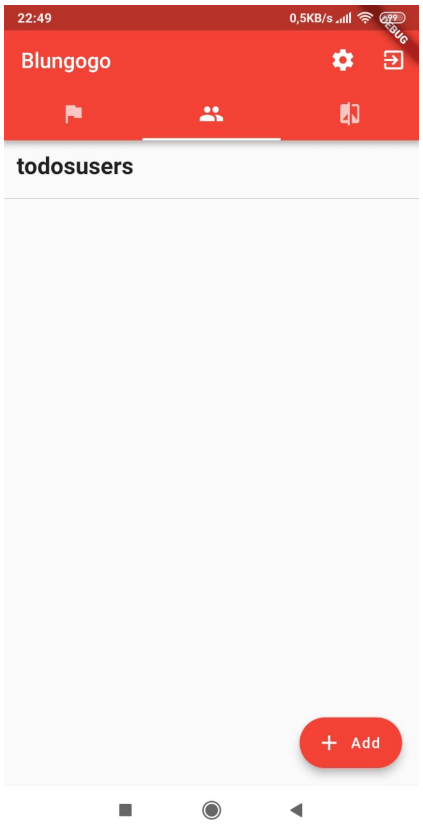

Figura 38 Feed de grupos

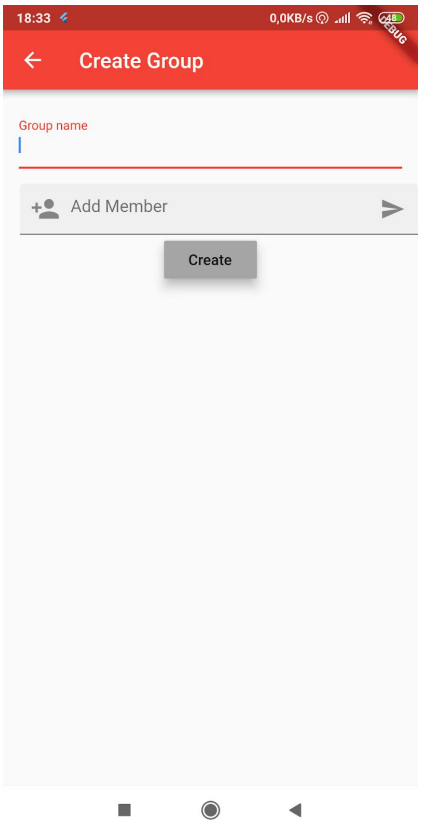

Figura 39 Creación de grupo

Para ver un grupo pulsamos sobre este en la feed, nos aparecerá la siguiente pantalla (Figura 40). Si somos administradores nos aparecerá el icono que se muestra en la pantalla. Si por el contrario no somos administradores, nos saldrá un icono de prohibido.

Para acceder a los ajustes pulsamos en este icono y nos aparecerá la pestaña de ajustes de grupo (Figura 41). En esta pestaña podremos añadir (Figura 42) o eliminar (Figura 43) miembros del grupo, gestionar sus permisos (Figura 44) o borrar el grupo. Para añadir y eliminar miembros se hace lo mismo que en la creación del grupo. Para cambiar los permisos se pulsa sobre el usuario y se le cambia el status automáticamente.

Figura 40 Vista de grupo

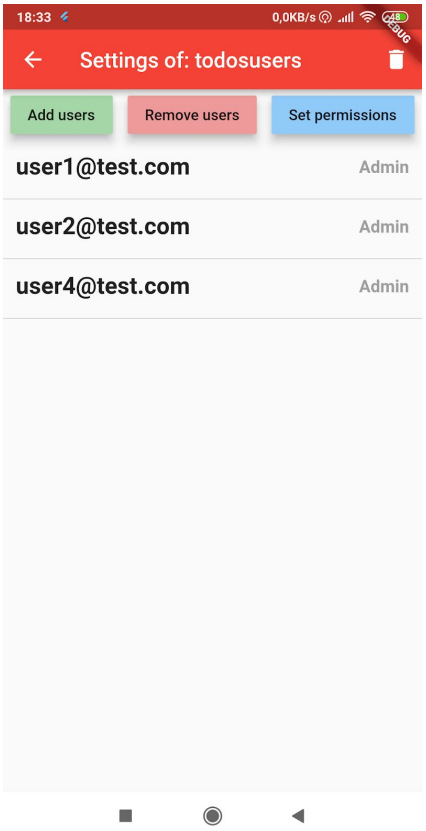

Figura 41 Vista ajustes de grupo

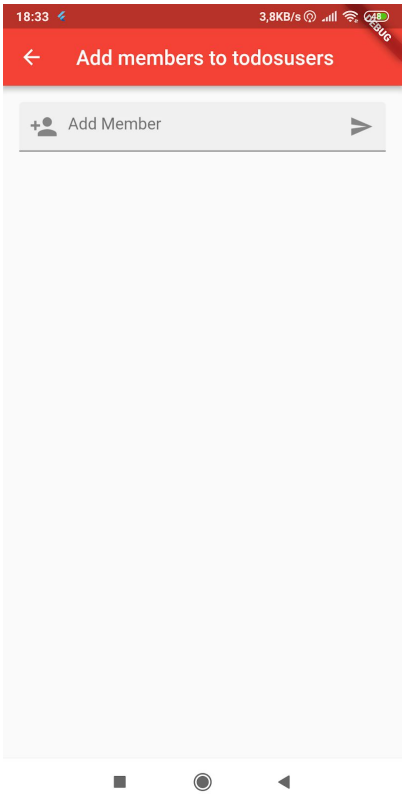

Figura 42 Añadir miembro

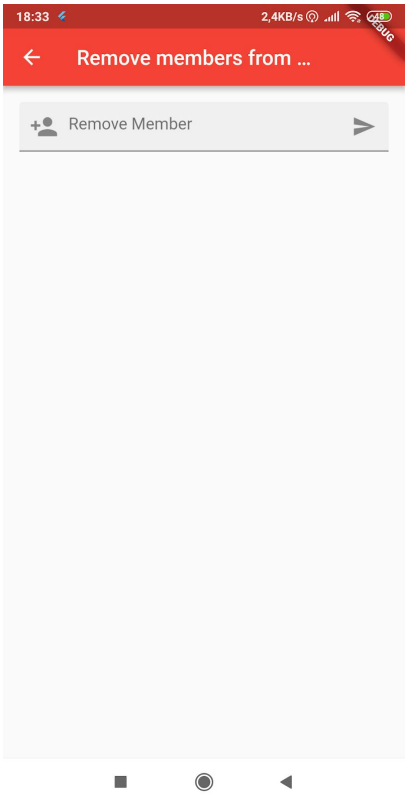

Figura 43 Eliminar miembro

Figura 44 Cambiar permisos

Por último tenemos las comparaciones. Igual que en las otras dos secciones, disponemos de las mismas funcionalidades en la pantalla principal de la sección (Figura 45).

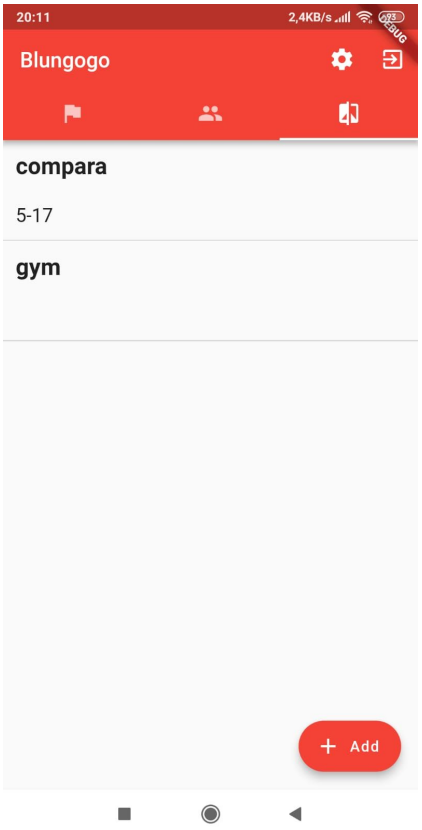

Figura 45 Vista comparaciones

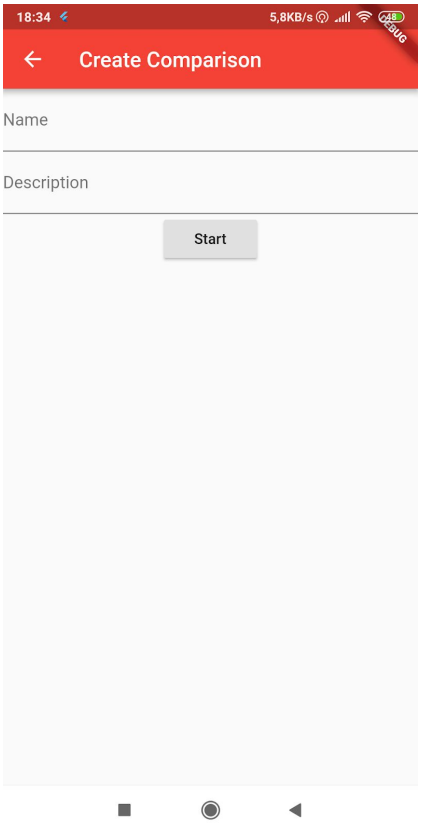

Figura 46 Crear comparación

A continuación vamos a explicar cómo crear una comparación.

Lo primero que tenemos que hacer es pulsar el botón de la parte inferior de la feed. Nos llevará a la pantalla de creación de comparación. Una vez rellenemos el formulario pulsamos el botón.

Una vez pulsado, nos aparecerán los grupos en los que estamos. Al pulsar en un grupo nos mostrará sus miembros, y al pulsar sobre ellos se les añadirá a la comparación. Una vez hayamos acabado de añadir miembros, pulsamos el icono de la parte superior derecha.

Ahora nos aparecerá la pantalla para escoger el intervalo de tiempo que queremos analizar en nuestra comparación. Seleccionamos el inicio y el fin y pulsamos el botón. Ya estará creada la comparación.

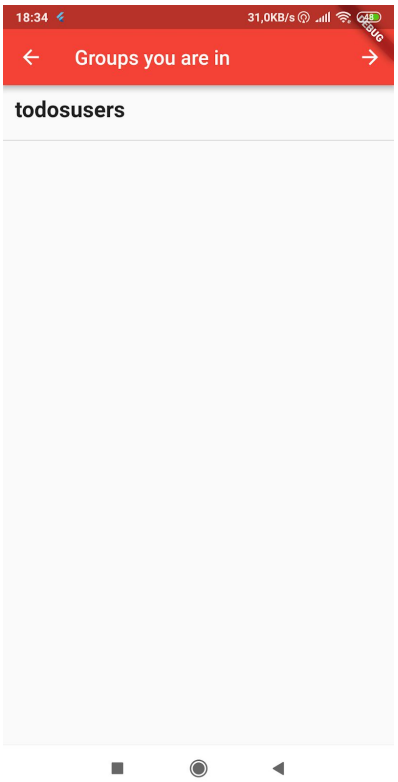

Figura 47 Grupos del usuario comparador

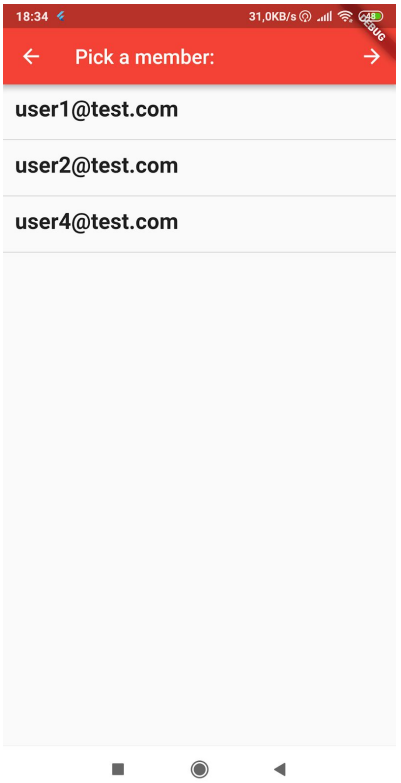

Figura 48 Usuarios del grupo escogido

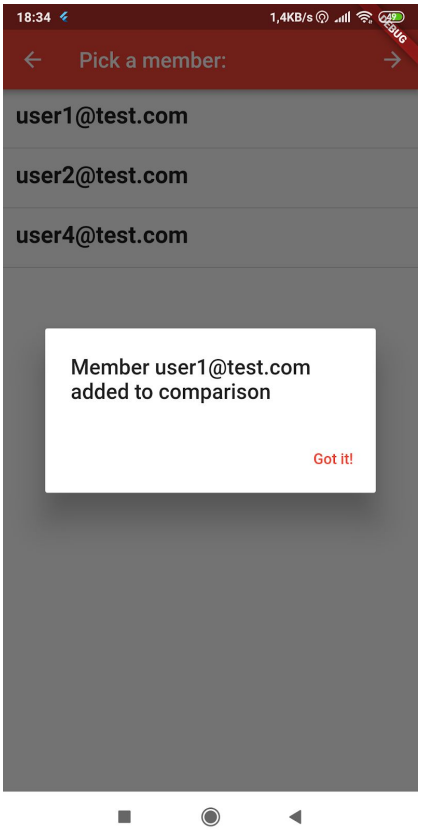

Figura 49 Miembro añadido

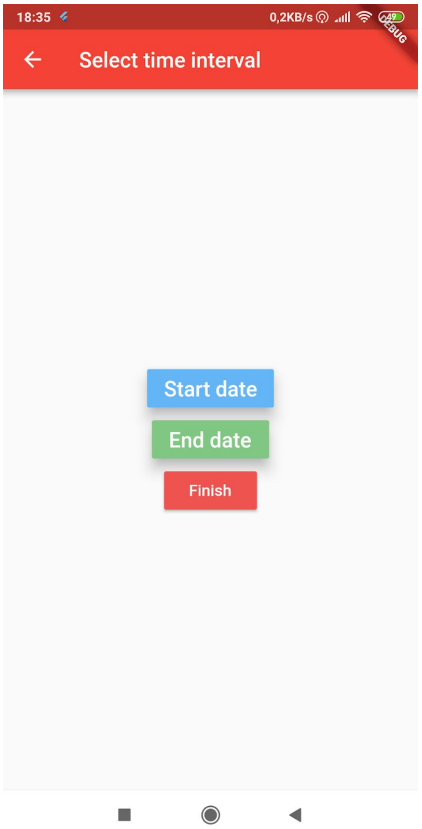

Figura 50 Elección del intervalo de tiempo

Para finalizar nos queda ver una comparativa. Simplemente pulsando en la feed la comparativa que deseamos ver en detalle nos aparecerá la siguiente pantalla:

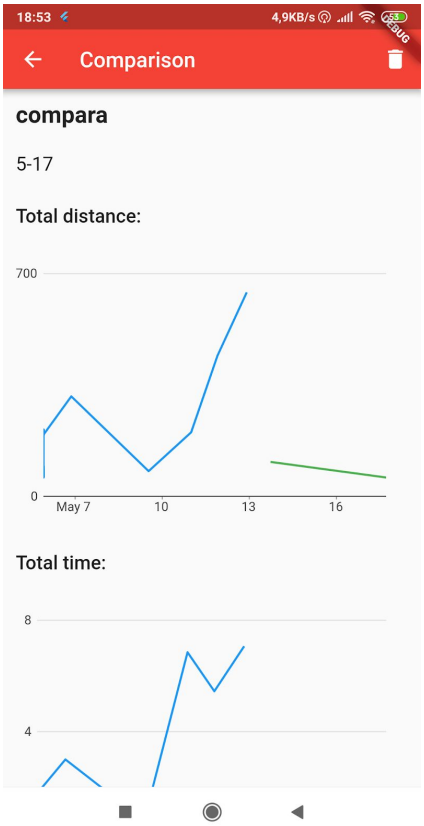

Figura 51 Vista comparativa

En esta vista podremos desplazarnos hacia arriba y hacia abajo para visualizar los gráficos asociados a cada métrica de las actividades. Cada usuario aparecerá con un color distinto sobre el gráfico.

Para borrar una comparativa basta con pulsar en el icono de la parte superior derecha y confirmar que queremos borrarla.