Universitat Oberta de Catalunya

uoc.edu

Desarrollo de una aplicación móvil para la gamificación orientada al fomento de la actividad física entre los adolescentes

Angel González Penella

Máster Universitario de Desarrollo de Aplicaciones para dispositivos móviles

Desarrollo de aplicaciones para dispositivos Android

Consultor/a: Francesc D'Assís Giralt Queralt

Profesor/a responsable de la asignatura: Carles Garrigues Olivella

31/05/2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Desarrollo de una aplicación móvil para la gamificación orientada al fomento de la actividad física entre los adolescentes*
Nombre del autor:	*Angel González Penella*
Nombre del consultor/a:	*Francesc D'Assís Giralt Queralt*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	06/2021
Titulación:	*Máster Universitario en Desarrollo de Aplicaciones para dispositivos móviles*
Área del Trabajo Final:	*Trabajo final de máster DADM*
Idioma del trabajo:	*Castellano*
Palabras clave	*Kotlin, deporte, gamificación, strava*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

El punto de partida del trabajo es la falta de actividad física entre los adolescentes. Se pretende fomentar la práctica de ejercicio entre este colectivo mediante el uso de las nuevas tecnologías. Concretamente, mediante una aplicación móvil para la gamificación.

La finalidad del trabajo es obtener una aplicación móvil, llamada MetroSalud, que permita a los profesores de educación física de un instituto de educación secundaria, proponer a sus alumnos un itinerario de actividades físicas y llevar un control de las mismas. Se plantea como una línea de *Metro* en la que cada estación o parada es una actividad deportiva a completar. Los alumnos obtienen una recompensa al completar cada parada.

La metodología empleada se basa en el modelo en cascada, en el cual partimos de unos requerimientos obtenidos como punto de partida del trabajo y unas necesidades de los profesores de educación física. A continuación se pasa al diseño de la aplicación siguiendo como base el Diseño Centrado en el Usuario que tiene por objeto resolver las necesidades concretas de los usuarios finales. Posteriormente se pasa a implementar la aplicación, realizar las pruebas que verifiquen su funcionamiento y finalmente el mantenimiento de la misma.

Como resultado se ha obtenido una aplicación programada en Kotlin, que utiliza Firebase como backend y la API de una conocida red social de deportes, STRAVA, junto a otras librerías como OkHttp o Jackson. La app es completamente funcional y cumple con los objetivos básicos planteados

inicialmente.

Abstract (in English, 250 words or less):

The starting point of the work is the lack of physical activity among adolescents. It is intended to promote the practice of exercise among this group through the use of new technologies. Specifically, through a mobile application for gamification.

The purpose of the work is to obtain a mobile application, called MetroSalud, that allows physical education teachers from a secondary school to propose to their students an itinerary of physical activities and keep track of them. It is proposed as a Metro line in which each station is a sports activity to be completed. Students get a reward for completing each stop.

The methodology used is based on the cascade model, in which we start from some requirements obtained as a starting point of the work and also some needs of the physical education teachers. Next, the application design is based on User Centered Design, which aims to solve the specific needs of end users. Subsequently, the application is implemented, tests that verify its operation and finally its maintenance.

As a result, an application programmed in Kotlin has been obtained, which uses Firebase as a backend and the API of a well known sports social network, STRAVA, along with other libraries such as OkHttp or Jackson. The app is fully functional and meets the basic objectives initially set.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

El punto de partida de este trabajo final de máster es la evidencia científica de la existencia de un alto grado de sobrepeso en los menores de entre 8 y 16 años en España.

Concretamente un estudio elaborado por la *Gasol Foundation* [1] en colaboración con *Unicef* en nuestro país, que mide el sedentarismo y la obesidad en casi 4000 niños, establece en un 35 por ciento los menores que padecen obesidad. En dicho estudio se evidencia también que solo el 36,4 por ciento de los niños y adolescentes realizan el suficiente ejercicio diario que recomienda la OMS.

Figura 1: Niños que practican actividad física recomendada por la OMS

Especialmente a medida que los niños se van haciendo mayores, se va reduciendo la actividad física e incrementando a su vez el uso de pantallas y otras actividades relacionadas con el sedentarismo. Y como se puede apreciar en la figura siguiente la falta de actividad física entre el alumnado adolescente es alarmante.

Figura 2: Incumplimiento de la recomendación de actividad física de la OMS por sexo y ciclo educativo

Este es un tema de vital importancia puesto que los niños y adolescentes de hoy en día serán la sociedad adulta del mañana y es por eso por lo que pretendemos ser parte de la solución del problema con el desarrollo de esta app.

Este trabajo está pensado como una colaboración con el departamento de educación física de un instituto concreto con estudios de ESO, bachilleratos y ciclos formativos. Los profesores de dicho departamento desean contar con una herramienta que les permita plantear un itinerario de retos deportivos que motiven al alumnado a realizar actividad física fuera del horario lectivo del centro.

Antes de iniciarnos en la aventura de la creación de una app desde cero, analizamos la tienda de aplicaciones de Google con el objetivo de encontrar alguna aplicación que pueda satisfacer nuestras necesidades. En seguida nos daremos cuenta que si bien hay numerosas apps que promueven el deporte y la actividad física, no las hay que pongan el foco en el público adolescente y aún menos que permitan al profesorado llevar un control rápido y efectivo de las actividades realizadas por los alumnos, algo así como un *moodle deportivo*.

Si analizamos unas pocas aplicaciones del Google Play tenemos:

Adidas running by Runtastic – Correr y fitness: Es una aplicación para registrar los entrenamientos con el GPS del teléfono móvil. Da estadísticas sobre el rendimiento, permite compartir actividades en redes sociales, ofrece retos pero no satisface nuestras necesidades.

- Enlace en Google Play: https://play.google.com/store/apps/details?id=com.runtastic.android

Runkeeper - GPS Correr Caminar: Es una aplicación análoga a la anterior pero de la marca de ropa deportiva ASICS.

- Enlace en Google Play: https://play.google.com/store/apps/details?id=com.fitnesskeeper.runkeeper.pro

Nike Run Club: Ésta es otra aplicación análoga a las anteriores de la marca americana de ropa deportiva Nike. Incluye retos y programas de entrenamiento pero no cumple con lo que buscamos.

- Enlace en Google Play: https://play.google.com/store/apps/details?id=com.nike.plusgps

Google Fit: seguimiento de actividad y salud: Es una aplicación de google para llevar un registro de todos nuestros entrenamientos y monitorizar nuestros objetivos. Nos da un historial de todo pero tampoco cumple con nuestras necesidades.

- Enlace en Google Play: https://play.google.com/store/apps/details?id=com.google.android.apps.fitness

GPS de Strava – carreras y ciclismo: Es una red social de deportistas. Registra y comparte con nuestros amigos nuestras actividades físicas y ofrece también retos. La posibilidad de crear clubes permitiría un seguimiento más estrecho de entrenamientos y tiempos en segmentos pero tampoco cumple los requerimientos que se desean.

- Enlace en Google Play: https://play.google.com/store/apps/details?id=com.strava

Toteemi: Es un juego en fase beta en el que participas en un equipo de entre dos y cuyo objetivo es dominar una área geográfica. El ganador se decide a partir de tus actividades deportivas y las de otros deportistas de tu misma zona geográfica. Lo interesante de esta app es que usa la API de Strava y nos da ideas de cara a plantear retos por equipos.

- Enlace en Google Play: https://play.google.com/store/apps/details?id=com.thetacollectivesl.toteemi

Son numerosas las aplicaciones que permiten registrar actividad física como las que hemos nombrado anteriormente. Podríamos señalar aún otras como *Endomondo*, *Sports Tracker*, etc. Pero ninguna de ellas se ajusta a lo que buscamos.

Con esto y considerando que actualmente desde el entorno social y educativo no se aportan suficientes soluciones al problema, con este trabajo final de máster pretendemos aportar una herramienta de uso para el profesorado de educación física que consiga potenciar y fomentar la actividad física entre los menores de 12 a 18 años.

1.2 Objetivos del Trabajo

Los objetivos generales del trabajo se pueden concretar en los siguientes puntos:

- Diseñar y desarrollar una aplicación móvil para la gamificación con el objetivo de fomentar la práctica deportiva y la actividad física entre el alumnado de los centros de educación secundaria obligatoria
- Recoger información de las actividades físicas para a partir de ciertos resultados otorgar recompensas
- Obtener un diseño de interfaz fácil e intuitivo para atraer la atención de los más jóvenes
- Permitir a los profesores de educación física diseñar un itinerario de retos puntuables que además tenga su utilidad para la evaluación de los alumnos
- Profundizar en el conocimiento del desarrollo de aplicaciones móviles para dispositivos Android y concretamente en el lenguaje Kotlin, que será la base del desarrollo del trabajo. Android cuenta con la mayor cuota de mercado, sobre todo en estas edades y al tratarse de una aplicación nativa se va a poder proporcionar un rendimiento mayor de la aplicación.

Más específicamente, el desarrollo de la aplicación debe alcanzar los siguientes requerimientos funcionales:

- Gestión de los usuarios: La app debe permitir registrar nuevos usuarios, iniciar sesión y acceder al perfil. Se debe diferenciar entre profesores y alumnos.
- Gestión de los grupos: se debe permitir al profesor crear grupos, añadir actividades y visualizar el progreso de los alumnos. A los alumnos se les debe permitir apuntarse a un grupo y ver los detalles del mismo y de sus paradas.
- Gestión de los retos, medallas y puntos: la app debe mostrar a los alumnos las estaciones disponibles con los objetivos para lograr la recompensa. Se debe premiar a los alumnos con puntos y medallas después de completar las paradas

Para completar las paradas se empleará una API externa para obtener la información de las actividades de los alumnos y decidir si se cumplen los retos y obtener una puntuación en cada uno de ellos.

1.3 Enfoque y método seguido

Las estrategias para llevar a cabo el trabajo son diversas. La primera de ellas sería la elaboración de una aplicación móvil autónoma que realice todos los requerimientos sin utilizar ninguna tecnología de terceros.

Para ello debería encargarse de tareas tales como:

- El registro de las actividades mediante el uso del GPS del dispositivo móvil y su posterior procesamiento para obtener los datos relevantes que vamos a necesitar para plantear los retos de gamificación de la aplicación que serían: kilómetros efectuados, tiempo de actividad, desnivel acumulado, actividades en grupo, tiempo en segmentos, etc.
- La gestión para la sincronización de datos con otros periféricos que se pueden emplear para capturar actividades físicas como pulseras, relojes smartwatch o ciclocomputadores.

Las tareas enunciadas anteriormente no son triviales y presentan su dificultad. Así por ejemplo, para sincronizar datos con otros dispositivos se requiere el uso de herramientas o APIs de terceros. En algunos casos hay herramientas de código libre, en otros hay que pagar una cantidad importante para adquirir una licencia (como es el caso de Garmin) y en otros simplemente no se ofrecen licencias para uso personal.

Es por todo ello por el que el enfoque seguido será el desarrollo de una aplicación móvil que utilice la API de una conocida red social para deportistas para obtener toda la información que vamos a necesitar.

Figura 3: Logo de la conocida red social de deportistas, STRAVA

Concretamente vamos a emplear Strava, que es una red social que permite registrar y compartir entrenamientos de distinto tipo con nuestros amigos y dejar comentarios en dichas actividades.

Será necesario, por lo tanto, utilizar Strava para registrar las actividades físicas y para iniciar sesión y registrarse en nuestra app.

La metodología aplicada para el desarrollo de la aplicación seguirá un modelo *waterfall* o en cascada. Este modelo es aplicable a proyectos como el nuestro en el que los requisitos están fijados y no van a cambiar durante el ciclo de vida del desarrollo. Como se ve en la figura siguiente, el modelo divide en distintas fases secuenciales el proyecto, interpretándose como el fluir del agua que va cayendo de una fase a la siguiente.

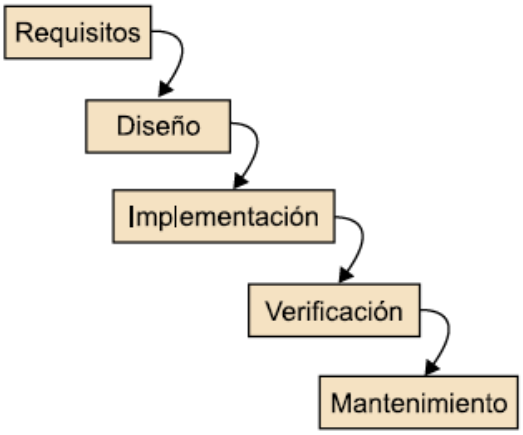


Figura 4: Ejemplo de fases de un proyecto de software en un modelo de desarrollo en cascada

Destacar la importancia que se le da a aspectos como la planificación, los tiempos o las fechas límite.

1.4 Planificación del Trabajo

Los recursos disponibles para realizar este trabajo serán a nivel hardware:

- Un ordenador con un Intel Core i3 7100, 16GB de RAM, SSD Kingston de 500GB, Nvidia GTX 1060 de 3GB con Windows 10.
- Un smartphone BQ Aquaris V con Android 8.1 Oreo

Y a nivel software:

- Android Studio como IDE para el desarrollo de aplicaciones nativas para Android
- Libre Office para el desarrollo de la memoria y la documentación del trabajo
- Firebase como backend
- Gimp para la edición y realización de los recursos gráficos necesarios para la aplicación
- GanttProject para llevar a cabo la planificación del proyecto
- Open Broadcaster Software para la captura del vídeo de presentación del trabajo

Para realizar el trabajo vamos a planificar una dedicación diaria de 2 horas durante los días laborables y de 4 los sábados y domingos. Además

vamos a realizar 4 entregas desde el inicio del trabajo hasta el próximo 2 de junio.

Antes de comenzar el trabajo se redactó una propuesta de trabajo final de máster. Tras la aprobación por parte del profesor colaborador se inicia el trabajo de la PAC1: Plan de trabajo.

En esta primera PAC se van a definir los objetivos y el plan de trabajo que seguiremos para conseguirlos. Concretamente definiremos un contexto para justificar nuestro trabajo, unos objetivos, el enfoque y método elegido y haremos una planificación del proyecto. Esta fase finaliza con la entrega de la PAC1.

La PAC2 contendrá el diseño centrado en el usuario definido en el plan de trabajo. Concretamente vamos a analizar el contexto de uso de la aplicación y sus usuarios. Posteriormente haremos un diseño conceptual y un prototipo y finalmente se realizará una evaluación de ello.

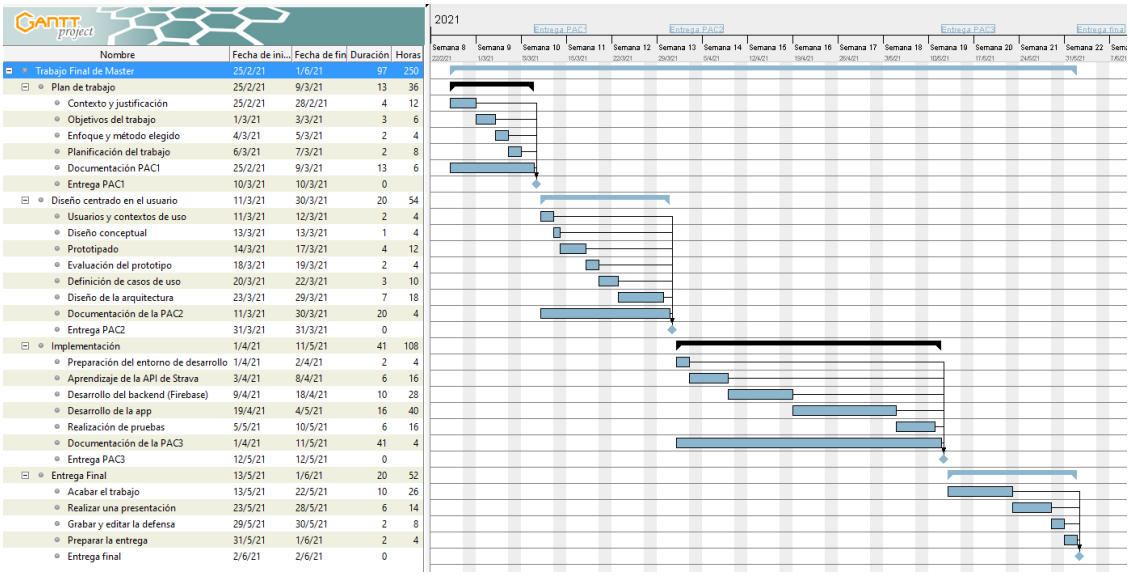
Tras entregar la PAC2 pasaremos a la PAC3: Implementación. En esta fase vamos a implementar el producto definido en el plan de trabajo. Al inicio de esta fase prepararemos el entorno de trabajo configurando todas las herramientas software que vamos a precisar.

A continuación iniciaremos una etapa de aprendizaje de la API de Strava. Ésta es una parte importante puesto que va a ser la clave de nuestro trabajo y debemos aprender a utilizarla correctamente para ser capaces de extraer con eficacia la información requerida en nuestra app.

Después desarrollaremos la parte de backend de la aplicación. Inicialmente la idea pasaba por montar una aplicación web con arquitectura REST pero se considera que es más importante invertir el tiempo disponible en el desarrollo de la app y en su funcionamiento. Por ello emplearemos Firebase como backend.

Finalmente pasaremos a la parte de programación de la aplicación móvil. El público objetivo de la app es fundamentalmente un usuario con teléfono Android. Por ello desarrollaremos la aplicación utilizando Android Studio y utilizaremos el lenguaje Kotlin motivado por nuestro interés en aprender este lenguaje de programación. No debemos olvidarnos de la fase de testeo y pruebas que darán por concluido el desarrollo de la app.

Para concluir, en la fase de la PAC4: Entrega Final, dejaremos un tiempo para concluir el producto que vamos a desarrollar así como para confeccionar la presentación y preparar la posterior defensa del presente trabajo.



Nombre	Fecha de ini...	Fecha de fin	Duración	Horas
Trabajo Final de Master	25/2/21	1/6/21	97	250
Plan de trabajo	25/2/21	9/3/21	13	36
Contexto y justificación	25/2/21	28/2/21	4	12
Objetivos del trabajo	1/3/21	3/3/21	3	6
Enfoque y método elegido	4/3/21	5/3/21	2	4
Planificación del trabajo	6/3/21	7/3/21	2	8
Documentación PAC1	25/2/21	9/3/21	13	6
Entrega PAC1	10/3/21	10/3/21	0	
Diseño centrado en el usuario	11/3/21	30/3/21	20	54
Usuarios y contextos de uso	11/3/21	12/3/21	2	4
Diseño conceptual	13/3/21	13/3/21	1	4
Prototipado	14/3/21	17/3/21	4	12
Evaluación del prototipo	18/3/21	19/3/21	2	4
Definición de casos de uso	20/3/21	22/3/21	3	10
Diseño de la arquitectura	23/3/21	29/3/21	7	18
Documentación de la PAC2	11/3/21	30/3/21	20	4
Entrega PAC2	31/3/21	31/3/21	0	
Implementación	1/4/21	11/5/21	41	108
Preparación del entorno de desarrollo	1/4/21	2/4/21	2	4
Aprendizaje de la API de Strava	3/4/21	8/4/21	6	16
Desarrollo del backend (Firebase)	9/4/21	18/4/21	10	28
Desarrollo de la app	19/4/21	4/5/21	16	40
Realización de pruebas	5/5/21	10/5/21	6	16
Documentación de la PAC3	1/4/21	11/5/21	41	4
Entrega PAC3	12/5/21	12/5/21	0	
Entrega Final	13/5/21	1/6/21	20	52
Acabar el trabajo	13/5/21	22/5/21	10	26
Realizar una presentación	23/5/21	28/5/21	6	14
Grabar y editar la defensa	29/5/21	30/5/21	2	8
Preparar la entrega	31/5/21	1/6/21	2	4
Entrega final	2/6/21	2/6/21	0	

Figura 5: Diagrama de Gantt

Con todo este planteamiento y teniendo en cuenta la dedicación posible al trabajo, se muestra en la imagen anterior la planificación temporal de las tareas a realizar con su estimación en tiempo.

1.5 Breve sumario de productos obtenidos

Con este trabajo se pretende obtener una app, de la cual se entregará el código fuente y el ejecutable apk.

Por otra parte se va a obtener una memoria del trabajo realizado con el mayor nivel de detalle posible. Y finalmente una presentación en vídeo con la defensa del trabajo, donde se hará una completa demostración del funcionamiento de la app.

1.6 Breve descripción de los otros capítulos de la memoria

Los capítulos que describimos a continuación se corresponden con las PACs o hitos propuestos anteriormente.

Por ello tendremos un segundo capítulo dedicado al diseño de la app y compuesto por las tareas comentadas en apartados anteriores y reflejadas también en el diagrama de Gantt.

El tercer capítulo lo dedicaremos a la implementación y al igual que el anterior, vamos a basar sus apartados en las distintas tareas comentadas anteriormente y detalladas también en el diagrama de Gantt.

A continuación tendremos un capítulo dedicado a las conclusiones del trabajo. Reflexionaremos sobre lo realizado en el trabajo y sobre la consecución o no de los objetivos planteados inicialmente, las dificultades encontradas, las soluciones y sobre todo lo que hayamos aprendido.

Finalmente dejaremos para los últimos capítulos, aspectos también importantes sobre el trabajo como un glosario de términos relevantes del trabajo y la bibliografía utilizada.

2. Diseño de la aplicación

En este apartado nos vamos a ocupar de las cuatro partes que siguen las fases del diseño centrado en el usuario (DCU) para el desarrollo de la aplicación móvil.[2]

El diseño centrado en el usuario sitúa a éste en el centro de todo el proceso de diseño de productos y aplicaciones con un claro componente tecnológico. El DCU involucra al usuario en todas las fases de desarrollo del producto, las cuales se realizan de forma iterativa para dar forma a la aplicación. Estas fases como veremos a continuación, consisten en la investigación y el análisis de los usuarios, el diseño y la evaluación.

2.1 Usuarios y contexto de uso

En esta primera fase el objetivo será conocer las características de los usuarios, así como sus necesidades y objetivos con la aplicación y establecer un contexto de uso. Aquí vamos a detectar las funcionalidades que deberá tener la aplicación con el objetivo de satisfacer a sus usuarios.

Para realizar todo esto vamos a utilizar algunos métodos de indagación. Concretamente usaremos dos: entrevistas y encuestas. Dado que desde el inicio ya diferenciamos claramente entre 2 tipos de usuario: profesor y alumno, vamos a utilizar un método diferente para cada uno de ellos. La entrevista nos servirá para comprender las necesidades y preferencias de los profesores de educación física del centro, puesto que son ellos los que demandan una app que satisfaga la necesidad detectada. Los cuestionarios nos servirán para saber qué piensan los alumnos sobre una app dedicada a tales menesteres.

2.1.1 Entrevistas en profundidad

Vamos a emplear este método para obtener información de tipo cualitativo sobre las necesidades y preferencias de un tipo de usuario muy concreto, el profesor.

Para ello hemos preparado un pequeño guión con unas pocas preguntas.

2.1.1.1 Diseño de la entrevista

Conocimiento del usuarios

1. Edad y antigüedad

2. ¿Cuántas horas utilizas el móvil al día?

3. ¿Qué dispositivo móvil tienes? ¿Android o iOS?

4. ¿Tienes algún reloj o pulsera de actividad?

Conocimiento de la práctica deportiva

1. ¿Prácticas algún deporte? ¿Cuál?

2. ¿Cuántas horas de actividad realizas a la semana?

3. ¿Registras tu actividad física con el reloj, móvil o pulsera de actividad?

4. ¿Qué aplicación móvil utilizas para registrar tu activad? ¿Conoces otras? ¿Conoces alguna de estas: Connect, Strava, Runtastic, Runkeeper, Google Fit?

Aplicación

1. ¿Qué actividades quieres que registren los alumnos? ¿Qué datos te interesan de esas actividades?

2. ¿Cómo quieres evaluar las actividades de los alumnos?

3. ¿Quieres diseñar los itinerarios de actividades?

4. ¿Quieres incorporar aspectos “sociales”? (Compartir en redes sociales, fotos, tablas de clasificación, perfiles, etc)

5. ¿Qué características consideras esenciales y deberían incorporarse en la app?

Las entrevistas se han realizado con 2 profesores de educación física, de forma separada. Los apuntes más destacados de cada una de ellas se exponen a continuación.

2.1.1.2 Entrevistas

Entrevista 1: José

José es profesor de educación física en educación secundaria desde hace 2 años y tiene actualmente 26 años de edad. Su uso del teléfono móvil es moderado, entre 1 y 2 horas diarias. Cuenta con un teléfono móvil de la marca china Xiaomi con sistema operativo Android. También dispone de un reloj de la compañía americana Garmin, concretamente un Forerunner 235, que registra su actividad física.

Sus deportes principales son correr tanto por asfalto como por montaña, el senderismo y el ciclismo, del que se ha aficionado recientemente al *Gravel*.

En total practica de 5 a 7 horas de actividad física semanal y registra sus actividades con su reloj Garmin. Las aplicaciones que utiliza son Garmin Connect para sincronizar sus actividades con el teléfono móvil y Strava para

compartir las actividades con sus amigos. Conoce otras aplicaciones, y destaca principalmente Toteemi, la cual ha conocido recientemente y que recomienda a sus amigos. La principal pega que le encuentra es que es demasiado poco popular.

Respecto a la aplicación de este trabajo a José le gustaría promover el deporte entre sus alumnos con una app para móvil y para ello tiene pensado un itinerario de actividades a realizar fuera del centro. Estas actividades se subirían a la aplicación y las tendría en cuenta para la calificación final de su asignatura.

José ha pensado en diferentes tipos de actividades: senderismo, acumular minutos de actividad de cualquier deporte, acumular desnivel y realizar deporte en familia o en grupo entre otras. Según el grado de consecución de estas actividades se obtendría una puntuación.

Como José lleva diferentes grupos de ESO, le gustaría que la aplicación le dejara diseñar los diferentes itinerarios de actividades por cursos, así como llevar a diferentes cursos.

Respecto a la componente social, no le da mayor importancia pero sí es consciente que puede ser un factor determinante a la hora de motivar al alumnado.

Entrevista 2: Eduardo

Eduardo es un profesor de educación física algo más veterano pues lleva 23 años de servicio y tiene 51 años. Eduardo hace un uso del teléfono móvil más reducido, sobre la media hora diaria. Tiene un Samsung Galaxy M11 con sistema operativo Android.

A Eduardo le encantan los deportes y practica casi por igual la natación, el ciclismo y la carrera a pie. De hecho ha participado en algún triatlón y registra sus actividades con un Garmin 735XT. En total practica unas 10 horas de actividad semanal que registra con su reloj Garmin y la aplicación Garmin Connect.

Eduardo conoce otras aplicaciones populares como Strava pero no las utiliza, él es más de planificar sus entrenamientos en una hoja excel.

Respecto a la aplicación del trabajo, le parece muy buena la idea propuesta por su compañero José. Considera que sería una muy interesante poder proponer a los alumnos retos como correr 1km a un ritmo determinado, acumular minutos de actividad semanales o alcanzar un número determinado de piscinas.

A diferencia de José, Eduardo preferiría inicialmente plantear un abanico de actividades a sus alumnos, previamente predefinidas en la aplicación, aunque no le parece mal que la app le permita diseñar esas actividades.

Con respecto a la componente social, Eduardo tampoco le da ninguna importancia a este aspecto.

2.1.1.3 Conclusiones

Después de realizar las entrevistas podemos concluir que la aplicación debería:

- Motivar al alumnado a realizar deporte
- Ofrecer un itinerario de actividades diversas para completar
- Puntuar las actividades realizadas
- 2 perfiles, uno para profesor y otro para el alumno
- El perfil de profesor podría dar la opción de crear actividades y visualizar los grupos

2.1.2 Cuestionarios

Los cuestionarios o encuestas son métodos cuantitativos y por ello se deben llevar a cabo entre una muestra representativa de usuarios. Para ello vamos a diseñar un formulario en Google Docs que pasaremos a un par de grupos de un instituto.

Antes de pasarlo a los grupos, hemos escogido cuatro alumnos a los que pasarles el cuestionario con el fin de asegurarnos que todas las preguntas recogen todas las alternativas posibles.

2.1.2.1. Diseño del cuestionario

La encuesta consta de 16 preguntas y puede consultarse en el siguiente enlace: https://forms.gle/8CX1PARigvFgqVYx6

El cuestionario se compone de cuatro partes. Una primera parte de conocimiento del alumno y del uso que realiza del teléfono móvil. Una segunda parte de conocimiento sobre la actividad física que práctica. Una tercera sobre el conocimiento y el uso de aplicaciones y dispositivos móviles para registrar estas actividades. Y una parte final para conocer la opinión sobre una app orientada a la asignatura de educación física.

El cuestionario se muestra en las imágenes siguientes:

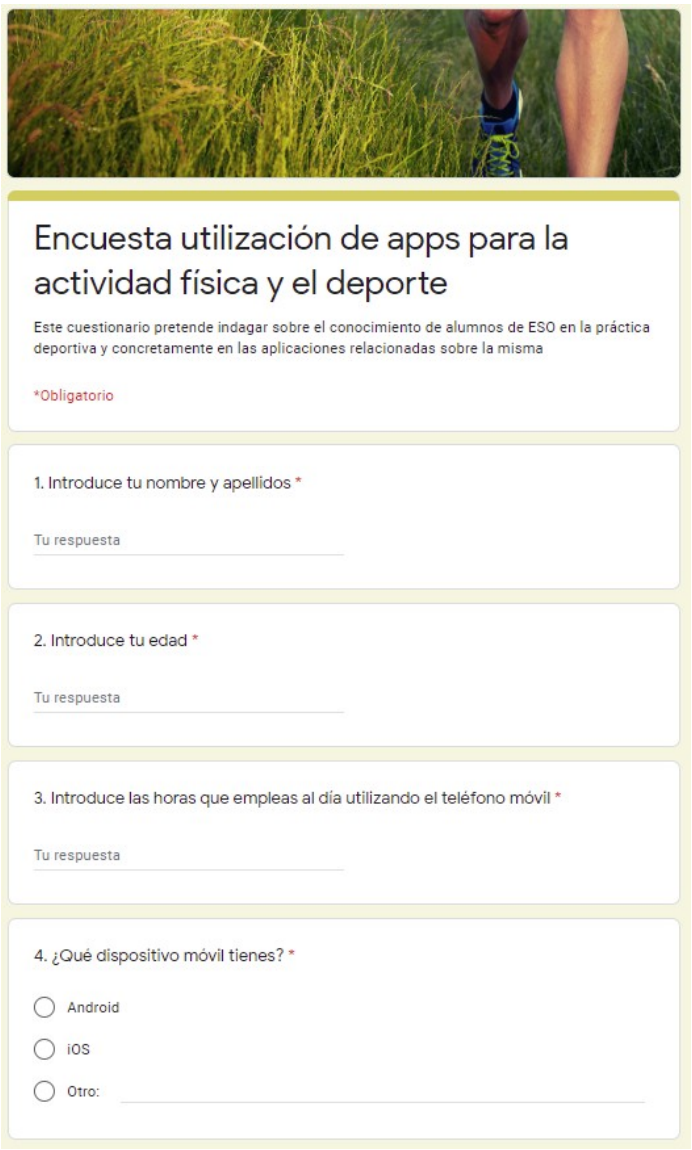


Figura 6: Primera Sección Encuesta a alumnos

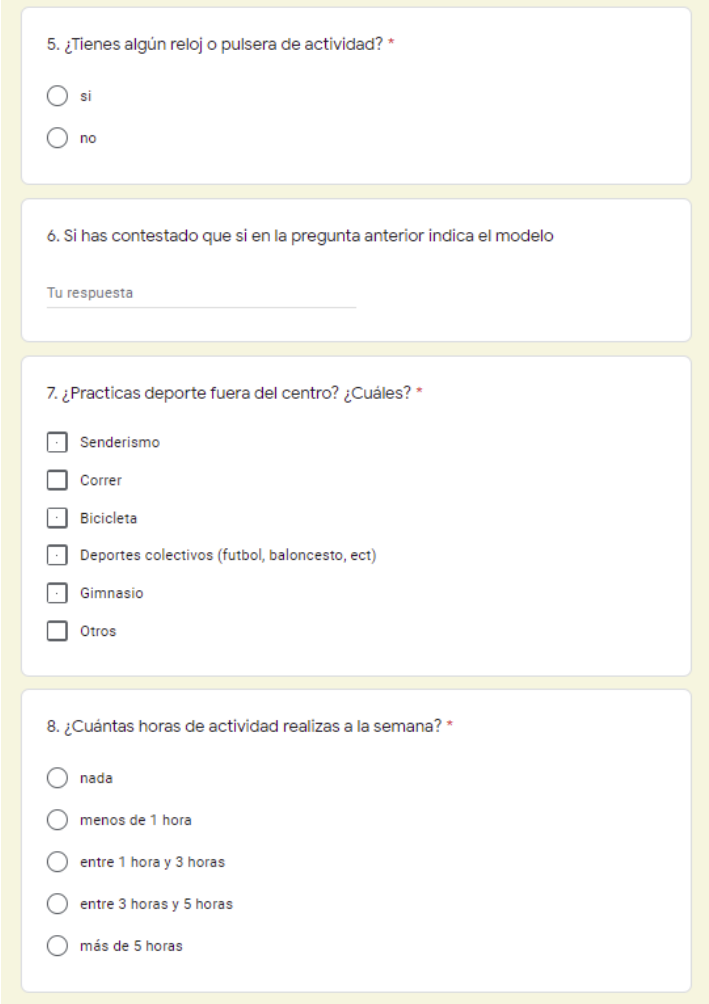


Figura 7: Segunda Sección Encuesta a alumnos

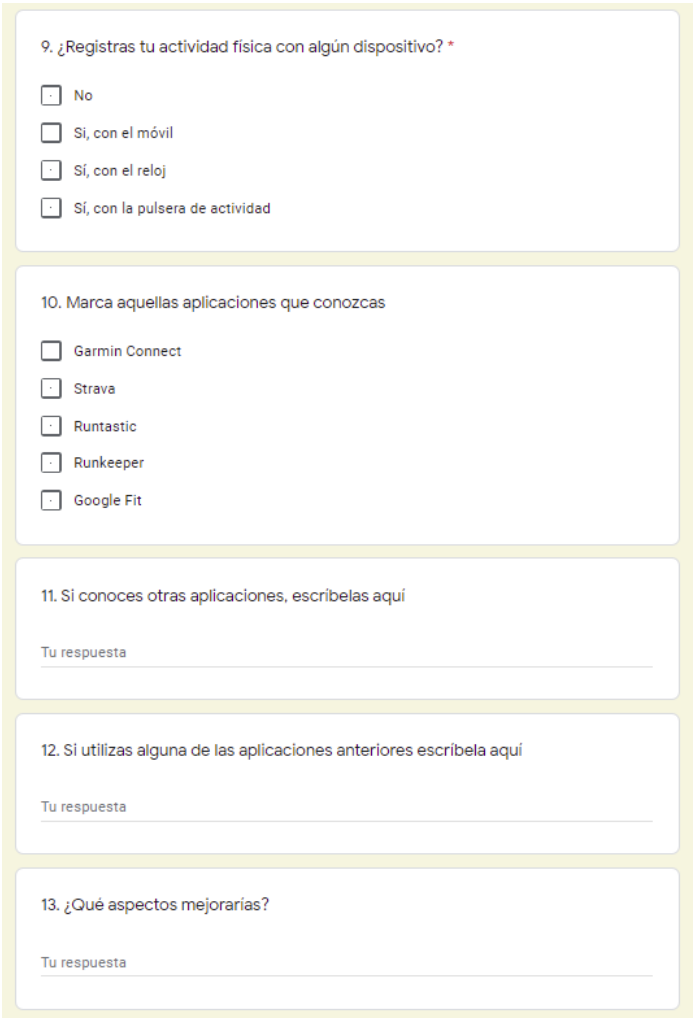


Figura 8: Tercera Sección Encuesta a alumnos

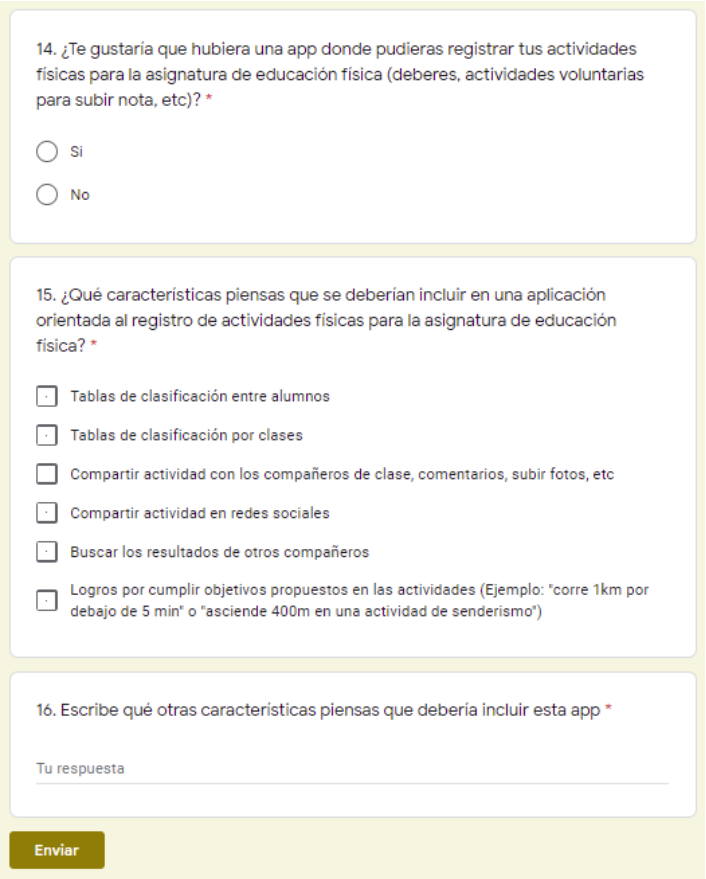


Figura 9: Cuarta Sección Encuesta a alumnos

2.1.2.2. Resultados del cuestionario

Las edades de los alumnos van entre 13 y 17 años puesto que se les pasó el cuestionario a dos grupos diferentes: a un segundo de la ESO y a un primero de bachillerato.

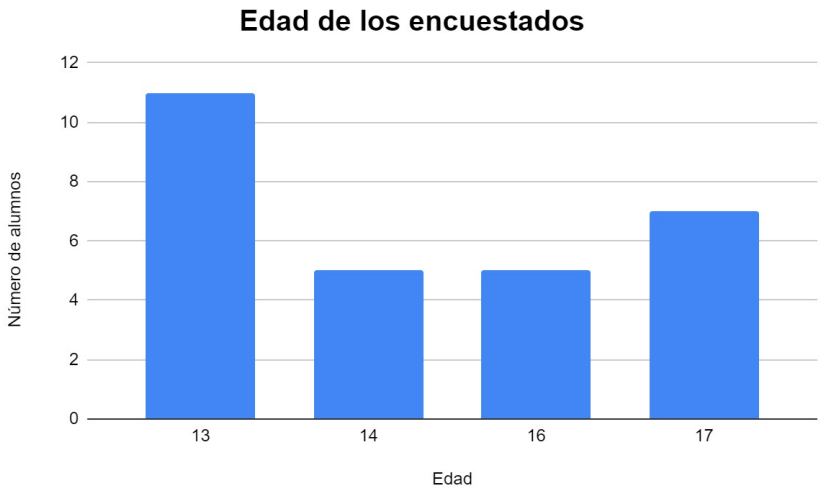


Figura 10: Edad

Respecto a las horas de uso del teléfono móvil vemos como mayoritariamente dedican unas 2 horas al día, habiendo casos extremos con una dedicación de hasta 4 o más horas.

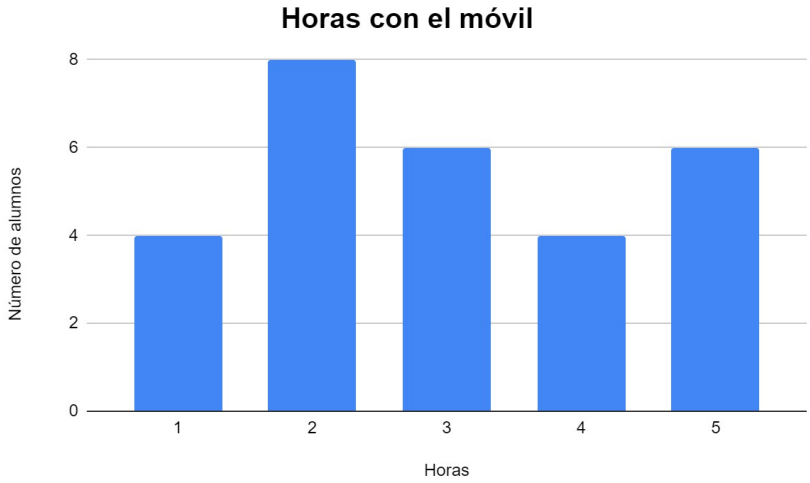


Figura 11: Uso del teléfono móvil

Como ya esperábamos y también podemos observar a continuación, el sistema operativo empleado por la mayoría de alumnos es Android con un 85% de utilización.

Figura 12: Sistema operativo

A continuación preguntábamos por el uso de relojes o pulseras de actividad y aquí podemos ver que más o menos la mitad de ellos utilizan uno.

Figura 13: Uso de relojes o pulseras de actividad

De entre los que han contestado que sí emplean relojes o pulseras de actividad tenemos que la mayoría utilizan una pulsera de la marca china Xiaomi, un amazfit y un par de alumnos tienen un Apple Watch.

Respecto a la práctica deportiva, algo más del 40 por ciento de los encuestados practican deportes colectivos siendo el fútbol el más mayoritario. Y entre un 25 y un 33% practican otros deportes como el senderismo, la carrera a pie y la bicicleta.

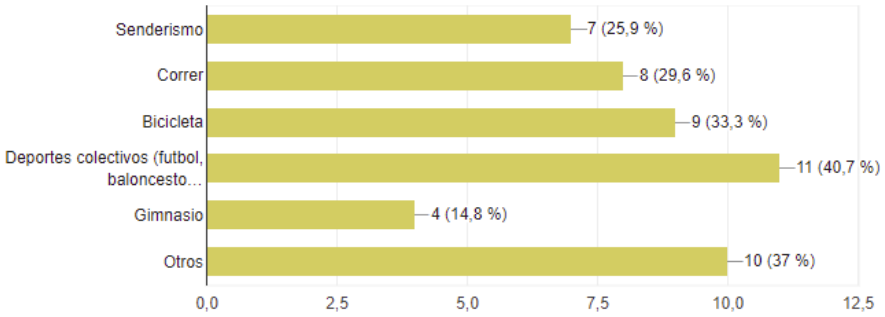


Figura 14: Deportes practicados

A continuación preguntamos a los alumnos por el tiempo que practican deporte a lo largo de la semana y vemos como algo menos del 60% dedican menos de 3 horas semanales a la práctica deportiva. Tan solo un 18% emplean más de 5 horas a la semana a ejercitarse y un 25% de los encuestados dedican entre 3 y 5 horas.

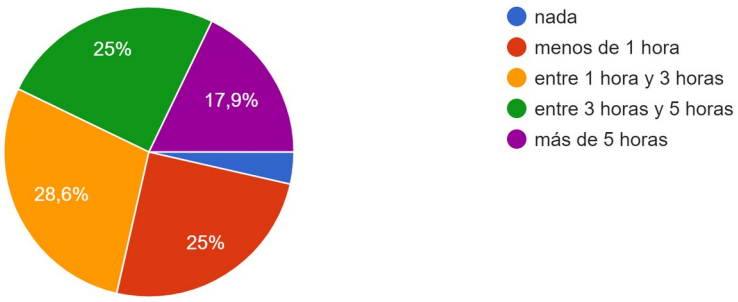


Figura 15: Tiempo de actividad semanal

En cuanto a la utilización de tecnologías para registrar la actividad realizada nos encontramos que la mitad de los encuestados no utilizan ninguna y el resto se reparte entre los que utilizan el teléfono móvil, que es la opción mayoritaria, y los que emplean relojes y pulseras de actividad.

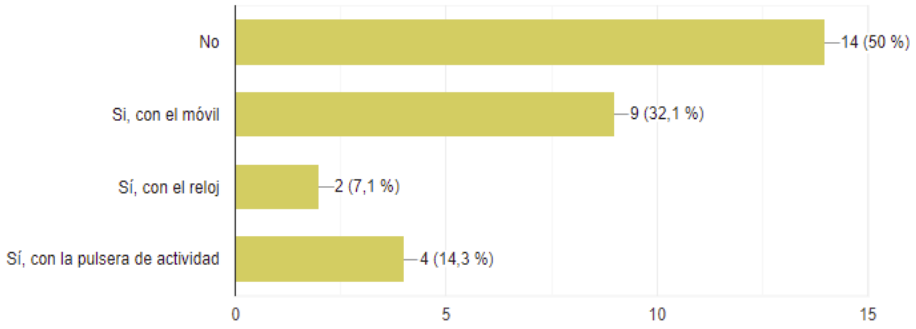


Figura 16: Utilización del reloj o pulsera para registrar la actividad física

Respecto a las aplicaciones conocidas para registrar actividades deportivas, señalamos que tan solo la mitad de los encuestados ha contestado a la pregunta. Además todos ellos han marcado la opción de “Google Fit” pero como podemos observar en las respuestas siguientes, no la usan. Del resto del opciones destacaría Garmin Connect con 5 alumnos que la conocen.

Otras aplicaciones propuestas por los encuestados son “Mi Fit” de Xiaomi, Samsung Health y la aplicación “Salud” de iOS.

Finalmente, respecto a la creación de una app específica para registrar actividades físicas para la asignatura de educación física tenemos que el 75% lo valora positivamente. Además, como podemos observar en el gráfico siguiente, las características de gamificación son las más valoradas.

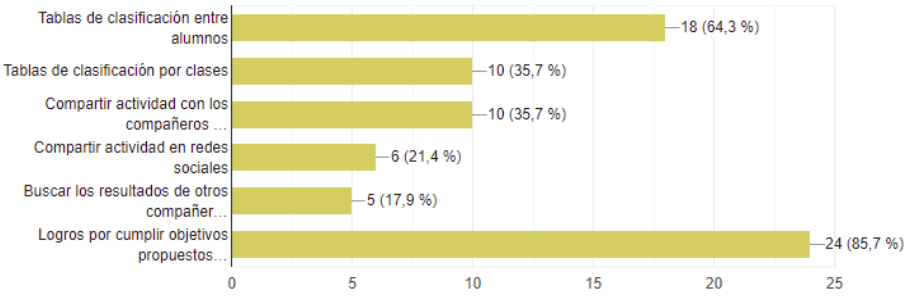


Figura 17: Características deseadas de la app

En la última pregunta del cuestionario, se plantea a los alumnos que sugieran otras características que debería tener la app y las respuestas inciden en por ejemplo, la puntuación para competir con otros compañeros y otras clases o el poder ver estadísticas tuyas y de otros compañeros.

2.1.2.3. Conclusiones del cuestionario

A la vista de los resultados del cuestionario podemos concluir algunas cosas:

- El alumnado cuenta principalmente con dispositivos móviles con sistema operativo Android. Algunos de ellos hacen un uso excesivo del móvil.
- Aproximadamente la mitad utiliza pulseras de actividad aunque en caso de registrar sus actividades lo hacen con el teléfono móvil.
- A penas conocen aplicaciones para registrar actividades físicas y tan sólo 2 han contestado que conocen Strava.
- Un 60% de los encuestados realiza muy poca actividad física semanal. Casi un 30% contesta que dedica menos de 1 hora a la semana a realizar ejercicio.
- Un 75% de los alumnos valoran positivamente la utilización de una app para la asignatura de educación física.
- Como sabemos a partir de nuestra práctica docente y se corrobora en la encuesta, el alumnado valora muy positivamente las opciones de gamificación con logros tablas de clasificación.

De forma más global, podemos concluir que el público objetivo al que se dirige nuestra app no realiza mucha actividad física y tampoco está acostumbrado al uso de aplicaciones móviles para registrar su actividad física aunque muchos de ellos sí utilizan relojes o pulseras de actividad. La opción de usar una app con opciones de gamificación orientada al fomento de la actividad física no parece ser una mala idea después de todo.

2.2. Diseño conceptual

Con toda la información obtenida en el punto anterior vamos a pasar al diseño. Para ello vamos a establecer los perfiles de usuarios, personas y escenarios, los cuales son técnicas que nos sirven para entender y analizar los usuarios y el uso que pensamos harán de la aplicación para ayudarnos en la orientación del diseño.

2.2.1. Perfil de usuario

Nos referimos al perfil de usuario como agrupaciones de los usuarios según sus características. En nuestro caso lo tenemos bastante fácil pues tenemos claramente dos perfiles: el perfil de profesor y el perfil de alumno.

Perfil profesor:

Este perfil está formado por profesores de educación física que quieran premiar a sus alumnos a través de la práctica deportiva fuera del centro.

Serán usuarios de entre 25 y 60 años, con destrezas más o menos elevadas en el uso del teléfono móvil y en la tecnología para el registro de actividades físicas. Hacen un uso normal del teléfono móvil al día con entre 1 y 2 horas.

Estos usuarios utilizarán la aplicación tanto en casa como en el trabajo, durante las horas de trabajo o las complementarias en casa

Las principales tareas que realizarán serán:

- Iniciar sesión en la app
- Crear una clase
- Añadir un itinerario de actividades en la clase o aceptar un itinerario por defecto
- Fijar fecha de comienzo de las actividades
- Acceder a los perfiles de sus alumnos para comprobar el grado de consecución de las diferentes actividades
- Acceder a una vista global con las puntuaciones obtenidas

Por lo tanto, la aplicación debe permitir:

- Registrarse como profesor y poder iniciar sesión
- Crear una clase y añadir alumnos a ella.
- Añadir a una clase las actividades una a una y configurarlas, o bien añadir un itinerario por defecto.
- Fijar una fecha para el comienzo del itinerario

- Obtener una vista resumen de la clase
- Obtener una vista resumen de cada usuario

Perfil alumno

Este perfil está formado por alumnos principalmente de ESO y bachillerato, con edades comprendidas entre los 12 y 18 años. Son usuarios con destrezas medias en la utilización de dispositivos móviles con poca o ninguna experiencia en el empleo de apps relacionadas con las actividades físicas. El uso del teléfono móvil es variado y oscila entre las 2 y las 4 horas diarias.

Este perfil de usuario empleará la aplicación fuera del instituto, normalmente los fines de semana y las tardes que realicen actividad física.

Las principales tareas a realizar serán:

- Iniciar sesión en la app
- Sincronizar su cuenta con Strava
- Si es el primer inicio de sesión apuntarse en la clase de su profesor
- Acceder a cada actividad que quiera realizar para leer la descripción
- Sincronizar cada actividad que considere realizada con Strava
- Ver resumen de actividades
- Ver tablas de clasificación por actividades e itinerario

La aplicación por lo tanto debe permitir:

- Registrarse como alumno e iniciar sesión
- Sincronizar la cuenta de Strava
- Apuntarse a una clase
- Ver la descripción de las diferentes actividades
- Sincronizar cada actividad con Strava para comprobar la realización de la misma
- Ver resumen de puntuación obtenida y/o logros
- Ver tablas de clasificación por actividades e itinerario

2.2.2. Personas y escenarios de uso

A partir de la información obtenida en el apartado anterior vamos a describir una serie de personajes ficticios con sus escenarios de uso para ver qué necesidades consideramos que tendrán estos usuarios.

Persona 1. Tipo: profesor

Nombre: José

Edad: 26 años

Tipo usuario: Profesor

Descripción de la persona

José es un chico soltero que vive con su novia en Requena, una pedanía situada a unos pocos kilómetros del lugar en que trabaja. Hace 2 años aprobó las oposiciones de profesor de educación secundaria en la especialidad de educación física y trabaja en un instituto en Utiel, un pueblo de 10000 habitantes del interior de la provincia de Valencia.

José está muy motivado con su trabajo y no para de pensar en nuevas ideas y proyectos con el objetivo de fomentar la actividad física entre su alumnado. Una app para la gamificación orientada a ello es algo que le motiva y piensa que será una idea bien acogida por los alumnos.

En su tiempo libre es un joven deportista, amante de la montaña y concretamente le encanta salir con su bicicleta de gravel a conocer nuevos caminos.

Descripción de escenario 1

En una hora de preparación de clases decide preparar un grupo en la app con un itinerario de actividades para sus alumnos de 2º de la ESO. Para ellos las tareas que tendrá que realizar serán las siguientes:

1. Iniciar sesión en la app
2. Elegir la opción de crear clase
3. Introducir un nombre para ese grupo
4. Seleccionar itinerario predeterminado o añadir paradas manualmente (segunda opción)
5. Para cada parada, elegir tipo de actividad entre una lista, fijar fechas para la realización y establecer las puntuaciones.

- Ejemplo: Elegir actividad de senderismo, fecha de realización del 01/05/2021 al 09/05/2021 y puntuaciones:
  - Oro (50 pts) Más de 20km,
  - Plata (25 pts) Más de 12 km,
  - Bronce (10 pts) Más de 5 km.

6. Al añadir todas las paradas terminar la creación de la clase. Se obtiene un código que hay que proporcionar al alumnado para que se apunte a su clase
7. Cerrar sesión y salir de la app

Descripción de escenario 2

En casa, durante una hora de preparación de clases, José quiere ver el progreso de sus alumnos de 2º de la ESO en el itinerario de actividades que les ha planteado. Para ello las tareas que realizará son las siguientes:

1. Iniciar sesión en la app
2. Elegir el grupo de 2º de la ESO
3. Verá una lista de todos sus alumnos con la puntuación obtenida hasta la fecha. Quiere ordenar los alumnos por puntuación.
4. Al pulsar sobre uno de ellos verá una nueva lista con la información de cada parada: si está completada, la fecha, la puntuación obtenida, etc.
5. Cerrar sesión y salir de la app

Persona 2. Tipo: profesor

Nombre: Eduardo

Edad: 51 años

Tipo usuario: Profesor

Descripción de la persona

Eduardo es un hombre casado con dos hijos, profesor de educación física de un instituto de Utiel. También reside allí junto a su familia y lleva 23 años ejerciendo como profesor.

Eduardo no pierde nunca su motivación, le encanta su trabajo y promueve entre su alumnado el deporte y las actividades al aire libre. Fuera de su horario de trabajo dirige una escuela de triatlón con alumnos y antiguos alumnos. Si la situación sanitaria lo permite, espera que durante el próximo año puedan competir.

Además de ello, él mismo practica también dicho deporte y dedica entre 12 y 15 horas semanales a su entreno personal. Con el uso de las nuevas tecnologías no es demasiado experimentado, solamente utiliza Garmin Connect para sincronizar sus actividades pues su reloj es de esta marca americana. Aún así todo lo que sea fomentar la práctica deportiva entre su alumnado lo ve positivamente.

Descripción de escenario 1

Durante el patio, Eduardo se dispone a crear un curso en la app para su grupo de 3º de la ESO. Quiere probar la app sin demasiadas complicaciones y por ello decide elegir el itinerario por defecto. Los pasos que realizará serán los siguientes:

1. Iniciar sesión en la app (o registrarse antes si no lo está)
2. Elegir la opción de crear clase
3. Introducir un nombre para el grupo
4. Escoger la opción de itinerario de actividades sugerido por la app
5. Pulsar en terminar y se obtendrá un código que habrá que proporcionar al alumnado par que se apunte

Persona 3. Tipo: Alumno

Nombre: Elena

Edad: 14 años

Tipo usuario: alumna

Descripción de la persona

Elena es una alumna de un instituto en Utiel que actualmente cursa 2º de la ESO. Es muy buena alumna y lo aprueba todo con notas superiores a 8. Su asignatura favorita son las matemáticas donde saca siempre un diez.

Respecto a la asignatura de educación física, no se le da mal pues saca un 8. Elena dedica entre 3 y 5 horas a la semana a la realización de actividad física pues los fines de semana se va con sus padres a practicar rutas de senderismo o salir en bicicleta de montaña.

El uso que realiza del teléfono móvil es moderado, aproximadamente una hora al día. Tiene una pulsera de actividad de la marca Xiaomi, concretamente la “Mi band 5” aunque la usa simplemente para contar los pasos y no conoce otras aplicaciones diferentes a “Mi Fit” para registrar su actividad física.

Descripción de escenario 1

El profesor de educación física de Elena les ha propuesto que para subir nota en la asignatura deben instalarse una aplicación móvil y registrar allí su actividad física. Les ha explicado cómo funciona esta aplicación y que debe estar vinculada necesariamente a otra llamada Strava.

Como la utilización de teléfonos móviles está prohibida en el centro, el profesor les ha dicho que prueben a crearse una cuenta en Strava en casa y después vincularla a la app que van a utilizar llamada *MetroSalud.*

Elena una tarde cualquiera después de merendar se ha creado ya una cuenta en Strava, se ha descargado *MetroSalud* y se dispone a apuntarse a la clase que ha creado su profesor en la aplicación. Para ello realizará los siguientes paso:

1. Registrarse en MetroSalud como alumna Para ello introduce su nombre y apellidos, correo electrónico y una contraseña.
2. Una vez registrada debe apuntarse a su clase. Para ello Elena introduce el código que les ha facilitado su profesor.
3. Al introducir el código aparece el nombre de su clase y el profesor y confirma
4. En la pantalla principal puede ver el itinerario de paradas de su clase y su perfil. Como ya se ha creado una cuenta en Strava, decide sincronizar su cuenta allí.
5. Al sincronizar la cuenta una pantalla de la app le pide introducir sus datos de Strava, le da a confirmar y ya está vinculada la app de MetroSalud con Strava

Descripción de escenario 2

El sábado por la mañana Elena ha salido a la montaña junto dos compañeros de clase a realizar una ruta de senderismo. Tal y como les

comentó su profesor, antes de empezar la ruta Elena ha abierto la aplicación de Strava y ha pulsado en iniciar ruta para registrar la actividad física que van a realizar.

Ya por la tarde mientras descansa sentada en el sofá viendo la televisión, Elena arranca la aplicación MetroSalud para subir la actividad que ha realizado durante la mañana. Para ello realizará las siguientes tareas:

1. Iniciar sesión en la app
2. Seleccionar la actividad de senderismo
3. Tras leer la descripción de la tarea y comprobar que efectivamente su actividad de la mañana cumple con lo que se pide, selecciona sincronizar con Strava
4. Elena comprueba como tras unos pocos segundos, la aplicación puntúa su actividad de la mañana y la marca como completada
5. A continuación decide cotillear un poco para ver cómo la han hecho sus compañeros. Para ello accede a “Ver más” y comprueba en una tabla de clasificaciones las puntuaciones de sus compañeros.
6. Salir de la aplicación.

2.2.3. Flujo de interacción

Una vez definidos los escenarios de uso y antes de pasar al prototipado de la aplicación, vamos a definir de forma general la interacción de los diferentes tipos de usuario con la aplicación.

La realización gráfica del flujo de interacción nos va a permitir detectar todas las funcionalidades necesarias de la app.

En la siguiente imagen tenemos el flujo de interacción del usuario profesor con la aplicación:

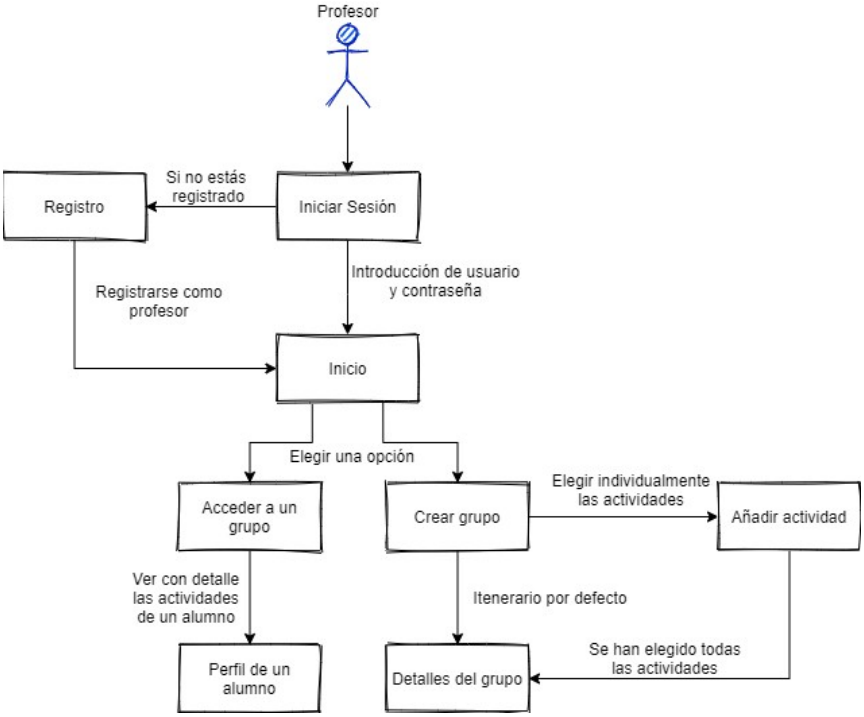


Figura 18: Flujo de interacción del profesor

Y en la siguiente imagen tenemos el flujo de interacción con la aplicación del usuario alumno:

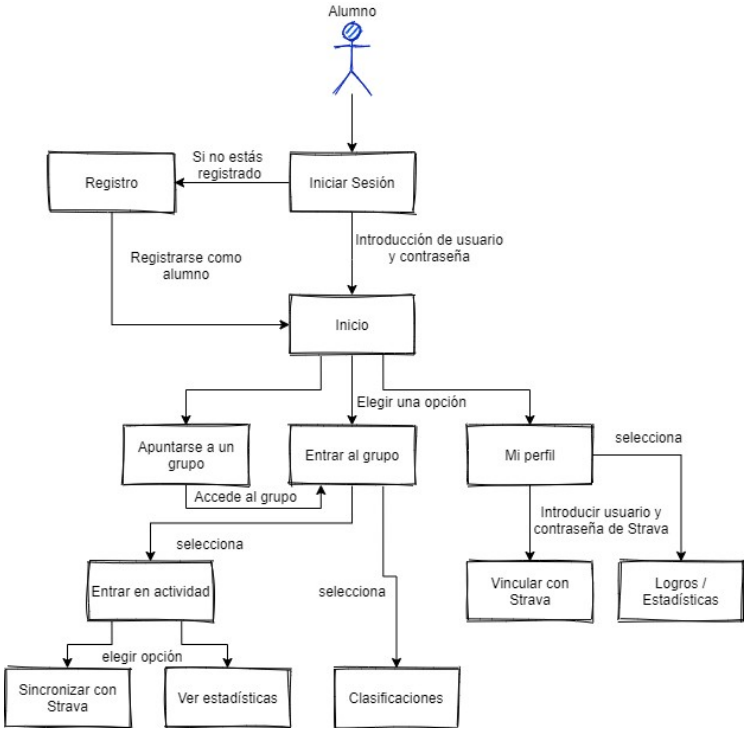


Figura 19: Flujo de interacción del usuario alumno

2.3. Casos de uso

Con toda la información recogida en apartados anteriores, vamos a intentar definir los casos de uso de nuestra aplicación.

En primer lugar vamos a definir los actores del sistema. Nuestra aplicación tendrá dos actores: el usuario alumno y el usuario profesor. El comportamiento y las pantallas de la aplicación serán ligeramente diferentes para cada caso.

Actor	Descripción
Alumno	Representa al tipo de usuario alumno. Como hemos comentado anteriormente su interacción con el sistema será ligeramente diferente al del actor Profesor.
Profesor	Representa al tipo de usuario profesor. Su interacción con el sistema es diferente a la del actor Alumno.

A continuación presentamos la relación de los distintos actores con los paquetes del sistema:

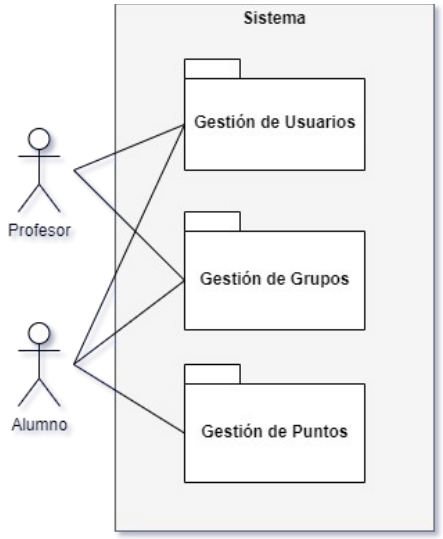


Figura 20: Diagrama de paquetes del sistema

En la imagen siguiente tenemos el diagrama de casos de uso del primer paquete: la gestión de usuarios. La única diferencia entre los actores es que los Alumnos deben vincular su cuenta con Strava:

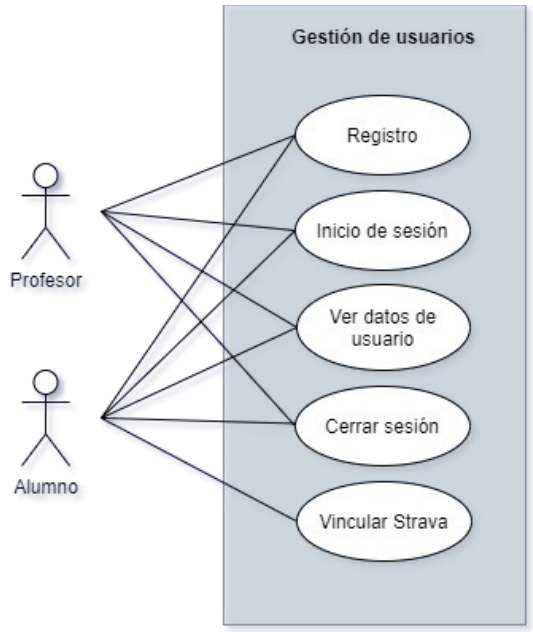


Figura 21: Diagrama de Gestión de usuarios. Casos de uso

En ver *datos de usuario*, el actor alumno verá su perfil y el actor profesor una lista de grupos.

A continuación vemos como interactúan los distintos actores con el paquete de grupos. Como vemos tenemos el caso general *Crear grupos* que se concreta en los casos *crear grupo con las paradas por defecto* y *editando los datos de cada parada*.

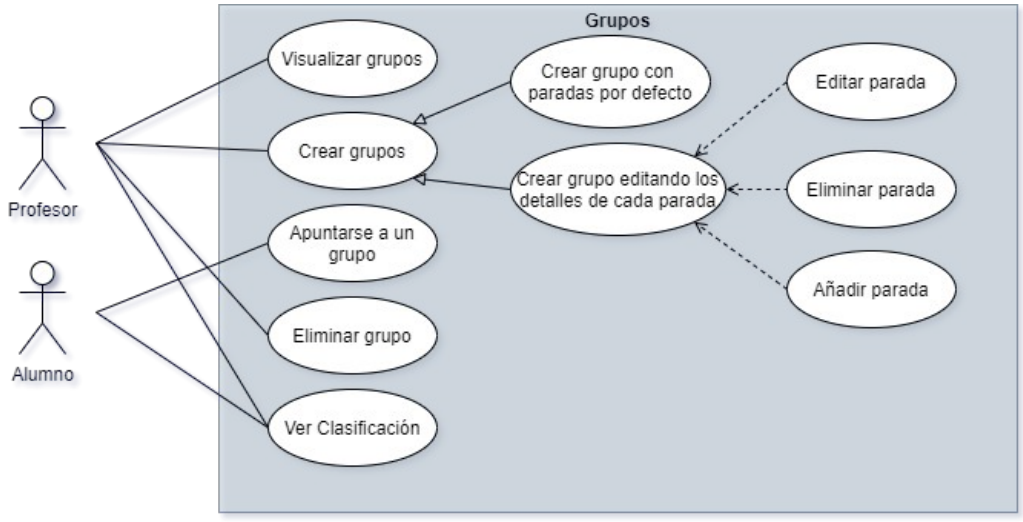


Figura 22: Diagrama de gestión de grupos. Casos de uso

Por otra parte los casos Editar parada, Eliminar parada y Añadir parada sólo se realizarán si el actor profesor desea realizar estas acciones.

Finalmente tenemos el diagrama de gestión de puntos cuyo único actor es el alumno y del cual la función principal es completar las paradas:

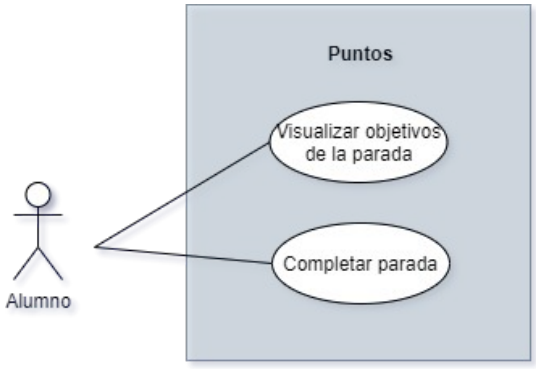


Figura 23: Diagrama de gestión de puntos. Casos de uso

Seguidamente vamos a definir cada uno de lo casos de uso mostrados con anterioridad:

CU-01	Registro
Descripción	El usuario se registra en la aplicación del sistema con un email y un password. A partir de este momento serán sus credenciales de inicio de sesión
Actores	Alumno y profesor. Deben seleccionar el tipo de usuario
Precondiciones	No haberse registrado anteriormente
Secuencia principal	1. Arrancar la aplicación 2. Hacer click en el botón de registro 3. Rellenar los datos de registro, seleccionando el tipo de usuario 4. Hacer click en el botón de confirmar 5. El sistema comprueba que los datos del formulario son válidos y que no está ya registrado 6. Se realiza el registro del usuario y se le redirecciona a la pantalla de inicio
Errores / Alternativas	1. Que los datos del formulario no sean válidos 2. Que falte algún campo por rellenar 3. Que el usuario ya esté registrado
Postcondiciones	El sistema registra al usuario en Firebase Authentication y Firebase Realtime Database

CU-02	Inicio de sesión
Descripción	Inicio de sesión
Actores	Los usuarios registrados acceden a la aplicación
Precondiciones	Alumno y profesor. Deben seleccionar el tipo de usuario
Secuencia principal	1. Arrancar la aplicación 2. Introducir el correo electrónico y la contraseña 3. Hacer click en el botón de Enviar 4. El sistema comprueba mediante Firebase Authentication las credenciales del usuario
Errores / Alternativas	1. Que el correo electrónico o la contraseña no sean válidos 2. Que el usuario no esté registrado
Postcondiciones	Se acceden a los datos de usuario o se le indica al usuario que los datos introducidos no son correctos

CU-03	Ver datos de usuario (Como actor profesor será “Visualizar Grupos”)
Descripción	Acceder a los datos de inicio del usuario
Actores	Alumno y profesor. El profesor verá la pantalla para gestionar sus grupos y el alumno verá la pantalla de su grupo
Precondiciones	Haber iniciado sesión. CU-02
Secuencia principal	1. Hacer click en la opción iniciar sesión 2. Obtener los datos de Firebase para mostrar en la pantalla 3. Mostrar datos en la pantalla
Errores / Alternativas	1. Que el usuario y password introducidos no sean correctos
Postcondiciones	El usuario visualiza sus datos en la pantalla

CU-04	Cerrar sesión
Descripción	Los usuarios cierran la sesión activa
Actores	Alumno y profesor
Precondiciones	Haber iniciado sesión en la aplicación
Secuencia principal	1. Hacer click en la opción cerrar sesión del menú de la barra ActionBar
Errores / Alternativas	
Postcondiciones	Se cerrará la sesión del usuario

CU-05	Vincular Strava
Descripción	Vincular la cuenta de Usuario de MetroSalud con Strava
Actores	Alumno
Precondiciones	• En la pantalla de registro, seleccionando el registro como alumno. CU-01 • Al iniciar sesión se accede a la pantalla de datos de usuario. Siempre que no haya vinculado durante el registro. CU-03
Secuencia principal	1. Hacer click en el botón de vincular con Strava 2. Se abrirá una página web o la propia aplicación de Strava donde iniciar sesión 3. Aceptar la cesión de datos desde Strava hacia MetroSalud 4. El sistema guarda en Firebase los datos necesarios para acceder a Strava
Errores / Alternativas	1. Que el usuario no esté registrado en Strava 2. Que el usuario no introduzca correctamente sus credenciales de strava 3. Que el usuario no acepte la cesión de datos desde Strava hacia MetroSalud
Postcondiciones	

CU-06	Crear grupos
Descripción	Crear un nuevo grupo
Actores	Profesor
Precondiciones	Haber iniciado sesión como profesor. CU-03
Secuencia principal	1. Hacer click en el botón de agregar grupo
Errores / Alternativas	
Postcondiciones	El sistema lleva a la pantalla de creación de grupo

CU-07	Crear grupos con paradas por defecto
Descripción	El usuario profesor crea un grupo con 4 paradas por defecto
Actores	Profesor
Precondiciones	Acceder a la pantalla de creación de grupo. CU-06
Secuencia principal	1. Hacer click en Terminar 2. Se crea una entrada nueva en la base de datos en Firebase con todos los datos del grupo

Errores / Alternativas	1. Se puede editar cada una de las 4 paradas propuestas 2. Se pueden eliminar las paradas 3. Se pueden crear paradas nuevas
Postcondiciones	El sistema le informará del código de inscripción para que lo proporcione a sus alumnos en una ventana de diálogo. Se volvera a la pantalla de grupos

CU-08	**Editar una parada**
Descripción	El profesor puede editar una parada
Actores	Profesor
Precondiciones	Acceder a la pantalla de creación de grupos. CU-06
Secuencia principal	1. Hacer click en el botón de editar una parada 2. Realizar los cambios en la pantalla de edición de datos 3. Aceptar los cambios 4. La parada del grupo se actualiza con los nuevos datos
Errores / Alternativas	1. Que no se rellenen todos los campos
Postcondiciones	Se han editado los datos de la parada y se vuelve a la pantalla de creación del grupo

CU-09	**Eliminar una parada**
Descripción	El profesor puede eliminar una parada del grupo
Actores	Profesor
Precondiciones	Acceder a la pantalla de creación de grupos. CU-06
Secuencia principal	1. Hacer click en el botón de eliminar parada 2. Los datos de la parada se eliminan del grupo que se está creando
Errores / Alternativas	
Postcondiciones	Se elimina la parada del grupo

CU-10	**Añadir una parada**
Descripción	El profesor puede añadir paradas al grupo durante su creación.
Actores	Profesor
Precondiciones	Acceder a la pantalla de creación de grupos. CU-06
Secuencia principal	1. Hacer click en el botón de añadir parada

	2. En una nueva pantalla se rellenan los datos de la parada 3. Hacer click en el botón de confirmar 4. Los datos de la nueva parada se han añadido al grupo que se está creando
Errores / Alternativas	Que no se rellenen todos los datos de la parada
Postcondiciones	Se vuelve a la pantalla de creación de grupo y aparece la parada creada

CU-11	**Eliminar un grupo**
Descripción	El profesor puede borrar grupos
Actores	Profesor
Precondiciones	Acceder a la pantalla de inicio del profesor. CU-03
Secuencia principal	1. Deslizar para borrar el grupo 2. Se solicita confirmación antes de borrar 3. Se elimina dicho grupo de la base de datos en Firebase 4. Los alumnos pertenecientes al grupo se les desvincula del mismo en la base de datos en Firebase
Errores / Alternativas	
Postcondiciones	Se actualiza la pantalla con los grupos que actualmente pertenecen al profesor

CU-12	**Apuntarse a un grupo**
Descripción	Los alumnos se apuntan a un grupo con el código que les proporciona su profesor
Actores	Alumno
Precondiciones	Haber iniciado sesión en la aplicación y acceder a la pantalla de usuario. CU-03
Secuencia principal	1. Introducir el código de inscripción en la caja de texto correspondiente de la pantallas 2. Se busca en la base de datos en Firebase si existe un grupo con ese código de inscripción 3. Se asigna el identificador del grupo al alumno en Firebase
Errores / Alternativas	1. Que el código de inscripción no pertenezca a ningún grupo 2. Se informa al alumno para que introduzca otro código
Postcondiciones	Una vez inscrito en un grupo la pantalla de datos de usuario muestra la información del grupo y sus paradas

CU-13	Ver clasificación 1
Descripción	El profesor visualiza una clasificación por puntos de los alumnos apuntados a su grupo
Actores	Profesor
Precondiciones	Haber creado previamente el grupo. CU-06
Secuencia principal	1. Hacer click en el grupo a visualizar 2. El sistema recogerá los nombres y apellidos de los alumnos apuntados al grupo y su puntuación 3. Se mostrará un listado ordenado por puntos con los alumnos del grupo
Errores / Alternativas	1. Que el grupo no tenga alumnos matriculados 2. En ese caso no se mostrará ningún alumno
Postcondiciones	

CU-14	Ver clasificación 2
Descripción	El Alumno visualiza una clasificación por puntos de los alumnos apuntados a su grupo
Actores	Alumno
Precondiciones	Haberse apuntado a un grupo. CU-12
Secuencia principal	1. En la pantalla de inicio hacer click en el nombre del grupo 2. El sistema recogerá los nombres y apellidos junto con la puntuación, de todos los alumnos de su grupo y la puntuación. 3. Se mostrará un listado ordenado por puntos del alumnado matriculado al grupo
Errores / Alternativas	1. Que sea el único alumno matriculado al grupo y en ese caso solo se mostrará ese alumno
Postcondiciones	

CU-15	Visualizar objetivos de la parada
Descripción	El alumno accede a una descripción de la parada junto con las instrucciones para obtener una puntuación oro, plata o bronce
Actores	Alumno
Precondiciones	Haberse apuntado a un grupo. CU-12
Secuencia principal	1. Hacer click en la parada deseada 2. El sistema carga los datos de la parada junto con la puntuación obtenida hasta la fecha por el alumno

Errores / Alternativas	
Postcondiciones	Si el alumno no ha puntuado aún en la parada, se obtienen simplemente los datos de la parada

CU-16	Completar parada
Descripción	El alumno completa una parada sincronizando datos con Strava
Actores	Alumno
Precondiciones	Visualizar los objetivos de puntuación de la parada. CU-15
Secuencia principal	1. Hacer click en sincronizar datos con Strava 2. El sistema conectará con Strava para obtener las actividades desde la fecha de inicio de la parada hasta la fecha de fin 3. El sistema comprueba en las actividades obtenidas que se cumplan los requisitos de compleción de la parada 4. El sistema otorga una puntuación y guarda la fecha en la que ha sido completada
Errores / Alternativas	1. Que el usuario haya retirado los permisos de conexión con Strava 2. Que el usuario no haya completado la tarea. En cuyo caso no guardará ninguna puntuación
Postcondiciones	Se vuelve a la pantalla de inicio del usuario y se visualiza su nueva puntuación

2.3. Prototipado

En la siguiente fase del Diseño vamos a utilizar la técnica del prototipado para obtener un diseño preliminar de la aplicación. Esto nos permitirá reflexionar sobre cómo debe ser la interfaz de usuario y realizar cambios si procede antes de ponernos a desarrollar la aplicación final.

2.3.1. Prototipo en baja definición

A continuación adjuntamos el prototipo en baja resolución de la aplicación realizado a mano con bolígrafo y que nos servirá como base para la realización del prototipo en alta resolución.

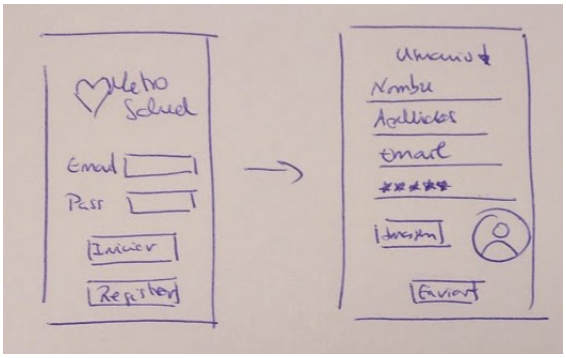


Figura 24: Pantallas de inicio y registro

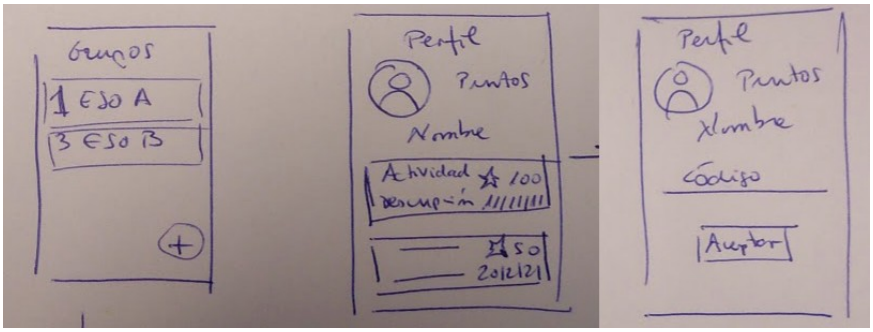


Figura 25: Pantallas de inicio para profesor y alumno (con grupo y sin grupo)

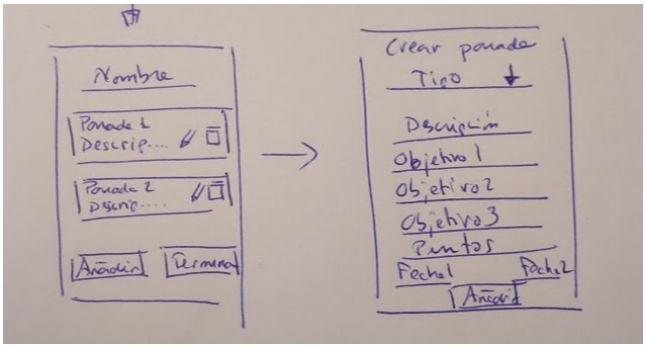


Figura 26: Pantallas de creación de grupo y creación / edición de parada

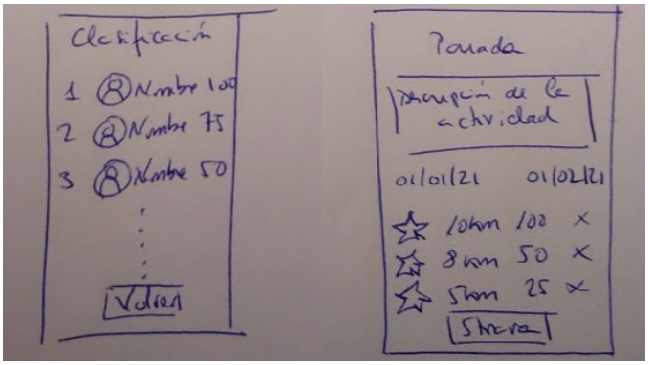
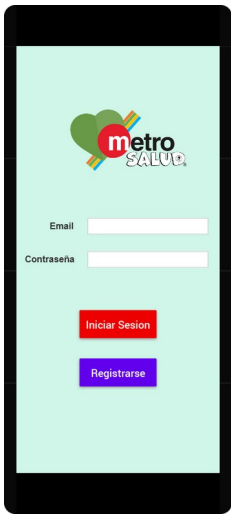


Figura 27: Pantallas de clasificación y descripción de parada

2.3.2. Prototipo en alta resolución

En este apartado vamos a elaborar el prototipo en alta resolución de la aplicación. La herramienta software que utilizaremos será Justinmind. Vamos a intentar que sea lo más fiel posible al resultado final y nos servirá para evaluar el funcionamiento de la app previo a proceder a la implementación del trabajo.

A continuación presentamos cada una de las pantallas de la app y posteriormente presentaremos un árbol de navegación de la misma.

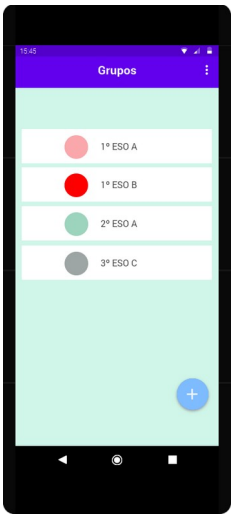


1. Pantalla de inicio de sesión o registro.

Es la pantalla de inicio y el usuario deberá introducir su email y contraseña como credenciales para acceder o bien registrarse.

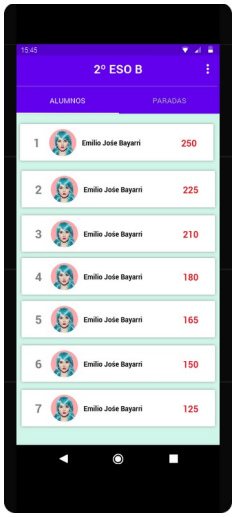
2. Pantalla de registro.

En esta pantalla los usuarios deben escoger tipo de usuario: profesor o alumno. Y a continuación rellenar los datos que se piden: nombre y apellidos, correo electrónico, contraseña y subir una imagen si lo desean



3. Pantalla de grupos.

Esta es la pantalla inicial para el profesor tras iniciar sesión. En ella puede entrar a un grupo para ver los alumnos y las paradas o bien añadir un grupo nuevo pulsando en el botón inferior



4. Pantalla de alumnos del grupo.

En esta pantalla el usuario profesor puede ver el listado de alumnos ordenados por puntuación. Aquí puede pulsar en un alumno para ver su perfil en detalle o puede ver las paradas asociadas al grupo

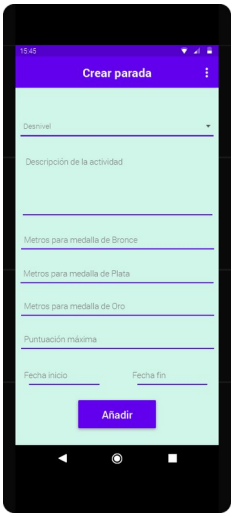
5. Pantalla de paradas del grupo.

Aquí el profesor puede consultar las paradas asociadas a este grupo y verlas en detalle.

6. Pantalla de creación de grupo.

El profesor asigna un nombre al grupo y se le sugieren 4 paradas por defecto. Éste puede darle a terminar o bien modificar y/o eliminar las paradas sugeridas y añadir otras.

Al terminar de crear el grupo una ventana le informará del código de inscripción.

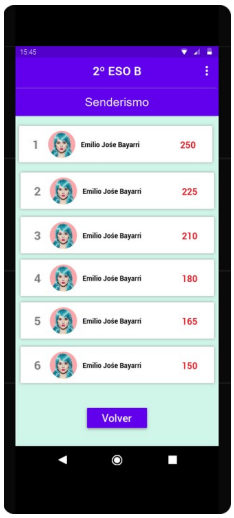


7. Pantalla de creación de parada.

En esta pantalla el profesor elige el tipo de parada, escribe una descripción y establece los objetivos y la puntuación junto con las fechas de inicio y fin.

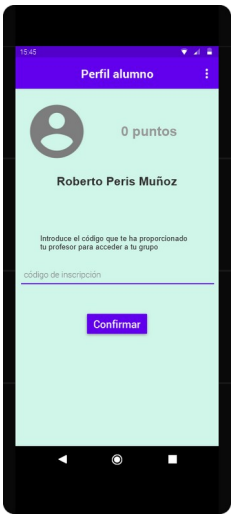
8. Pantalla de perfil del alumno.

Esta pantalla es la pantalla de perfil de un alumno con un resumen de su actividad. En ella aparece su puntuación total y los puntos y medallas obtenidas en cada parada.



9. Pantalla de clasificación de parada

En esta pantalla se puede ver la clasificación por puntos de una parada.

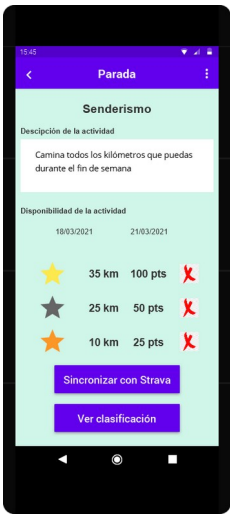


10. Pantalla de perfil de alumno sin grupo asignado.

Esta será la primera pantalla después de iniciar sesión que verá un alumno. Éste debe introducir el código que le ha proporcionado su profesor para acceder a su grupo.

11. Pantalla de perfil con grupo asignado.

En esta pantalla el alumno puede ver la tabla de clasificación pulsando en su curso o ver en detalle cada parada y proceder a completarla



12. Pantalla de detalle de parada.

En esta pantalla se pueden ver una descripción de la actividad y los requisitos para conseguir las medallas de oro, plata y bronce.

Además, el alumno podrá sincronizar la actividad de Strava que cumpla los requisitos para completar la parada. Para ello hay que pulsar el botón de “Sincronizar con Strava” y hacerlo con la app de Strava.

Además se puede consultar una tabla de clasificación para esta parada.

En las imágenes siguientes vemos los árboles de navegación para cada uno de los tipos de usuarios que tenemos: profesor y alumno.

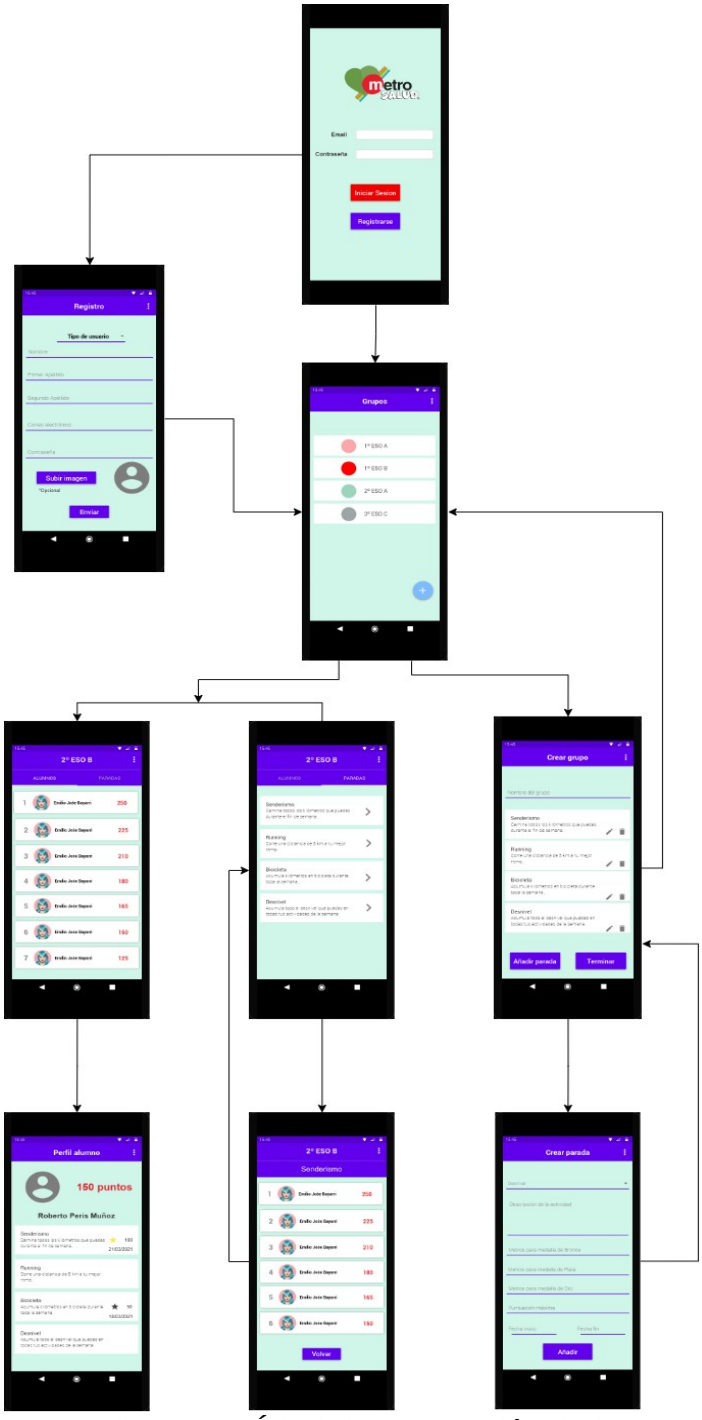


Figura 28: Árbol de navegación de la app para el profesor

Figura 29: Árbol de navegación de la app para el usuario alumno

2.4 Diseño de la arquitectura

En la imagen siguiente se presenta el modelo relacional de la base de datos de la app.

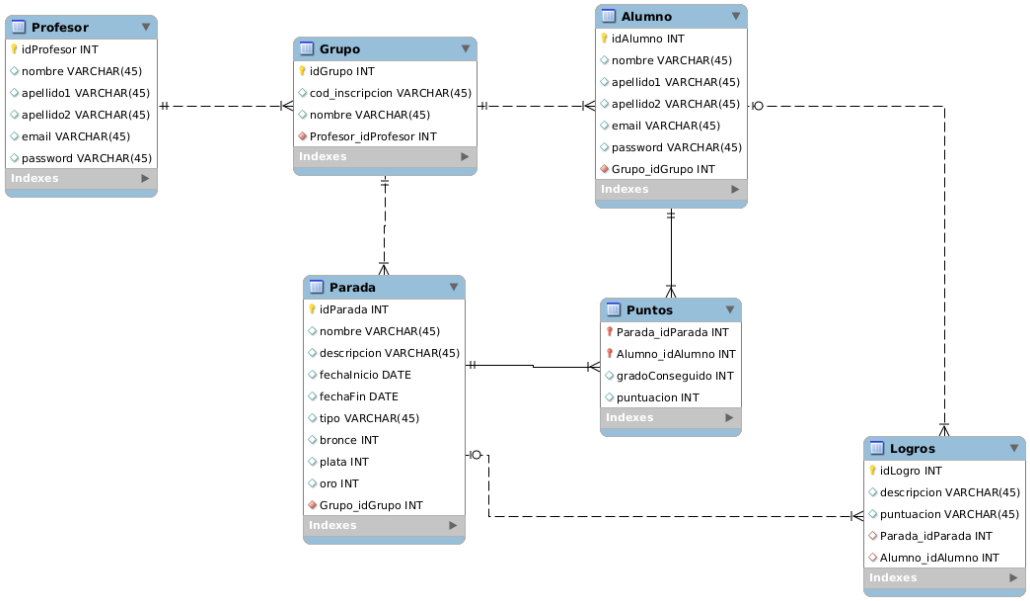


Figura 30: Modelo relacional de la base de datos

Y a continuación vamos a detallar los atributos de cada una de las relaciones

Profesor	
Atributo	Descripción
idProfesor	Identifica de forma unívoca a cada profesor
nombre	Es el nombre del profesor
apellido1	Primer apellido del profesor
apellido2	Segundo apellido del profesor
email	Email del profesor
password	Contraseña del profesor

Grupo	
Atributo	Descripción
idGrupo	Identifica de forma unívoca a cada grupo
cod_inscripción	Es el código de inscripción para que un alumno se apunte a su grupo. Este código debe ser único.
nombre	Es el nombre del grupo
Profesor_idProfesor	Es el identificador del profesor responsable del grupo

Alumno	
Atributo	Descripción
idAlumno	Identifica de forma unívoca a cada alumno
nombre	Es el nombre del alumno
apellido1	Primer apellido del alumno
apellido2	Segundo apellido del alumno
email	Email del alumno
password	Contraseña del alumno
Grupo_idGrupo	Es el identificador del grupo al que pertenece el alumno

Parada	
Atributo	Descripción
idParada	Identifica de forma unívoca a cada parada
nombre	Es el nombre de la parada
descripción	Es la descripción de la parada
tipo	Es el tipo de parada a elegir entre: senderismo, carrera a pie, bicicleta, desnivel, minutos, etc.
bronce	Es el valor mínimo para obtener medalla de bronce
plata	Es el valor mínimo para obtener medalla de plata
oro	Es el valor mínimo para obtener medalla de oro
fechaInicio	Es la fecha de inicio de la parada
fechaFin	Es la fecha final de la parada

Grupo_idGrupo	Es el identificador del grupo al que pertenece la parada

Puntos	
Atributo	Descripción
Parada_idParada	Es el identificador de la parada
Alumno_idAlumno	Es el identificador del alumno
gradoConseguido	Es la medalla conseguida por el alumno en la parada
puntuación	Es la puntuación obtenida por el alumno en la parada

Logros	
Atributo	Descripción
idLogro	Identifica de forma unívoca a cada logro
descripción	Es la descripción del logro
puntuación	Es la puntuación del logro
Parada_idParada	Es el identificador de la parada para este logro
Alumno_idAlumno	Es el identificador del alumno para este logro

La arquitectura del proyecto es Cliente-Servidor pues tenemos una aplicación donde los datos y las tareas se reparten entre los servidores y los clientes de la aplicación.

Los clientes de la app, los terminales móviles, realizarán peticiones a un servidor backend. La tecnología de servidor empleada será Firebase, pues nos va a permitir de formá fácil y sencilla llevar a cabo en un corto espacio de tiempo, nuestro proyecto.

El servidor de Firebase se encargará de autenticar a los usuarios y de proporcionar acceso a los datos almacenados en la base de datos.

Por otra parte nuestra app también utilizará un servidor backend del que obtendrá la información de las actividades de los alumnos que será Strava y del que obtendremos los datos mediante su API.

3. Implementación

3.1. Herramientas software utilizadas

3.1.1. Android

Para el desarrollo del trabajo hemos elegido el Sistema Operativo Android de Google, como hemos comentado anteriormente.

Figura 31: Logo de Android

Android es una pila de software de código abierto basado en Linux. A continuación vamos a describir sus componentes principales.

Kernel de Linux. Android se basa en el kernel de Linux para funcionalidades básicas como la administración de memoria de bajo nivel, generación de subprocesos o aprovechamiento de funciones de seguridad claves.

Capa de abstracción de hardware (HAL). Esta capa de abstracción brinda interfaces estándares del marco de trabajo de la API de Java. Consiste en varias librerías para distintos tipos de componentes de hardware como la cámara o el Bluetooth.

Runtime de Android. A partir de la API 21 de Android (Android 5.0) cada app de Android ejecuta sus propios procesos en instancias propias del Runtime de Android optimizados para ocupar un espacio mínimo en memoria. Se incluyen un conjunto de bibliotecas que proporcionan mayor funcionalidad en el entorno de programación.

Bibliotecas C/C++ nativas. Muchos componentes clave de Android como los comentados anteriormente: el Runtime de Android y la Capa de abstracción de hardware (HAL), se basan en código nativo en C y C++ por lo que es posible acceder directamente a estas bibliotecas si programamos una app que requiera estos lenguajes.

API de Java. Todo el conjunto de funciones del sistema operativo de Android están escritas en lenguaje Java. Todas estas API están disponibles para los desarrolladores de aplicaciones.

Apps del sistema. Con Android se incluyen un conjunto de apps para funcionalidades básicas como la mensajería SMS, navegador, etc. Estas apps de sistema se pueden integrar en las aplicaciones que desarrollemos para no tener que programa dicha funcionalidad.

3.1.2. Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para Android y está basado en IntelliJ IDEA. Es un potente editor con múltiples herramientas que facilitan las tareas al desarrollador. Entre las muchas herramientas que ofrece destacan: el editor visual para el diseño de la interfaz de usuario, el emulador instalar y probar las apps, el editor de código, el sistema de compilación basado en Gradle, la integración con servicios como GitHub, etc.

Figura 32: Logo de Android Studio 4

3.1.3. Kotlin

Para el desarrollo de la app MetroSalud vamos a utilizar el lenguaje de programación Kotlin. Kotlin es un lenguaje orientado a objetos, multiplataforma, de tipado estático y diseñado para interactuar con código Java lo cual permite hacer una migración gradual de Java a Kotlin. Compila principalmente para funcionar sobre la maquina virtual de Java y también compila a Javascript.

Figura 33: Logo de Kotlin

En mayo de 2019, Google anunció que Kotlin sería el lenguaje de programación preferido para el desarrollo de sus aplicaciones para Android. Desde la versión 3.0 de Android Studio, Kotlin se incluye como alternativa a Java para el desarrollo de apps.

Al igual que Swift para iOS, Kotlin distingue entre tipos *nullables* y *no nullables.* Los objetos que pueden ser nulos son declarados con un "?" después del nombre de tipo. Kotlin proporciona operadores seguro de nulidad para evitar excepciones del tipo *NullPointerExceptions*

Otra de las características estrella de Kotlin es que permite escribir los programas con menos líneas de código de manera más sencilla, en contraposición a la excesiva *verbosidad* Java.

Kotlin introduce también el concepto de corrutinas, también presentes en otros lenguajes de programación como C#, en la programación asíncrona muy presente en tareas como consumo de datos de APIs o actualizaciones de bases de datos.

3.1.4. Firebase

Para el backend de la aplicación de MetroSalud, inicialmente pensamos en crearnos el nuestro propio. Por razones de productividad y aprovechamiento del tiempo disponible nos hemos decantado por la utilización de Firebase.

Firebase es una plataforma en la nube de base de datos proporcionada por Google disponible para plataformas móviles y web.

Algunas de las características de Firebase son: la compartición de datos en tiempo real, envío de notificaciones, permite la monetización desde el propio Firebase, engloba la herramienta Analytics especializada en métricas de aplicaciones móviles, es fácilmente escalable, segura y lo que a nosotros más nos interesa: nos va a permitir centrarnos en otros aspectos de la aplicación restándole atención al desarrollo del backend.

Por otra parte y para la envergadura inicial del proyecto, con el plan gratuito pensamos que tenemos suficiente pues nos permite hacer 10000 autenticaciones al mes, 100 conexiones simultáneas a la base de datos, almacenar 1GB de datos y descargar hasta 10GB.

Figura 34: Logo de Firebase

Más adelante, si viéramos viabilidad en llevar MetroSalud a más institutos sería conveniente plantearse una solución diferente para el backend de la app.

Para la autenticación de nuestros usuarios vamos a emplear Firebase Authentication. Esta funcionalidad permite autenticar usuarios con diferentes y populares métodos como Facebook, Twitter, Google, email, etc. Nosotros emplearemos la función de registrarse e iniciar sesión con correo y contraseña.

Como base de datos emplearemos Firebase Realtime Database. Es una base de datos alojada en la nube en formato JSON que se sincroniza en tiempo real con todos los clientes.

3.1.5. Picasso

Picasso es una potente librería para Android que permite la carga rápida y sencilla de imágenes. Es realmente útil dado que cargar una imagen desde una url sin utilizar librerías requiere bastantes líneas de código. Con Picasso es posible hacerlo con una simple línea:

```
Picasso.get().load("http://myurl.png").into(imageView)
```

3.1.6. OkHttp

Para las peticiones POST y GET que hagamos a la API de Strava utilizaremos la librería OkHttp para Android. OkHttp es un cliente de HTTP y SPDY con licencia de Apache muy sencillo de utilizar.

3.1.7. Jackson

Para el intercambio de datos entre la aplicación y los servidores de Backend se emplea JSON. Con el objetivo de facilitar el trabajo con este formato de texto emplearemos la librería Jackson para el trabajo con ficheros JSON.

3.2. Desarrollo de la aplicación

3.2.1. Android Studio

La aplicación ha sido desarrollada para dispositivos Android que tengan una versión de la API igual o superior a la 26, la cual se corresponde con Android 8.0 Oreo. Esto nos asegura una amplia compatibilidad con la mayoría de dispositivos móviles Android en el mercado.

La app se ha desarrollado en Kotlin utilizando la versión 4.1.3 de Android Studio. A continuación vamos a presentar la estructura del proyecto el cual se divide en diferentes paquetes, todos ellos dentro del principal: *com.example.angel.metrosalud.*

Dentro del paquete principal se encuentran todas las clases que representan cada una de las actividades presentes en la app, junto con otros cinco paquetes.

- El paquete *adapters* contienen los adaptadores para los recyclerview que tiene la app.
- El paquete *data* contiene las clases que representan el modelo de datos de la app.
- El paquete *myFirebase* contiene una clase con las funciones que se relacionan con el backend en Firebase.
- El paquete myStrava contiene también una clase con las funciones necesarias para obtener los datos utilizando la API de Strava.
- Y el paquete *myUI* contiene una clase que implementa un diálogo de carga dado que el elemento *ProgressDialog* de Android ha sido desaconsejado o *deprecado.*

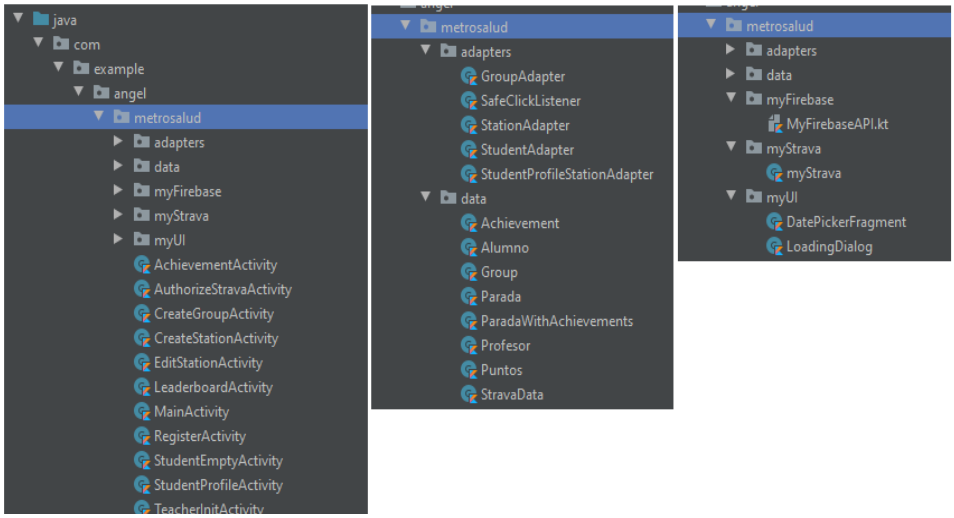


Figura 35: Estructura del proyecto en Android Studio

3.2.2. Data binding

En el trabajo hemos utilizado la librería *Data Binding* la cual nos permite vincular los componentes de la interfaz de usuario de los diseños a las fuentes de datos de la app, sin tener que enlazarlos uno a uno y evitando el excesivo uso de las llamadas findViewById.

Si echamos un vistazo a un fragmento de código de la actividad *StudentEmptyActivity*, podemos observar lo fácil que resulta su utilización. Como vemos nos evitamos todas esas llamadas a la función findViewById que normalmente haríamos para enlazar cada uno de los componentes imageButton, tvName, tvPoints y btnSubmitGroupCode que se ven en la figura siguiente:

```
class StudentEmptyActivity : AppCompatActivity() {
    private lateinit var binding: ActivityStudentEmptyBinding
    private var loading = LoadingDialog( mActivity: this)
    private var alumno: Alumno? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityStudentEmptyBinding.inflate(layoutInflater)
        setContentView(binding.root)
        initialise()
        MyFirebaseAPI( activity: this).getStudent(object : MyCallback {
            override fun onStudent(alu: Alumno) {
                alumno = alu
                updateUI(alu)
            }
        })
    }

    private fun initialise() {
        binding.imageButton.visibility = View.GONE
        binding.tvName.text = "Tu nombre".toString()
        binding.tvPoints.text = "0"
        binding.btnSubmitGroupCode.setOnClickListener {  it: View!
```

Figura 36: Ejemplo de data binding

3.2.3. Firebase Authentication

Como hemos comentado anteriormente, vamos a emplear Firebase Authentication para registrar y autenticar a nuestros usuarios. Vamos a mostrar una parte del código de la actividad *RegisterActivity* donde mostramos la utilización de la función de *createUserWithEmailAndPassword* de Firebase:

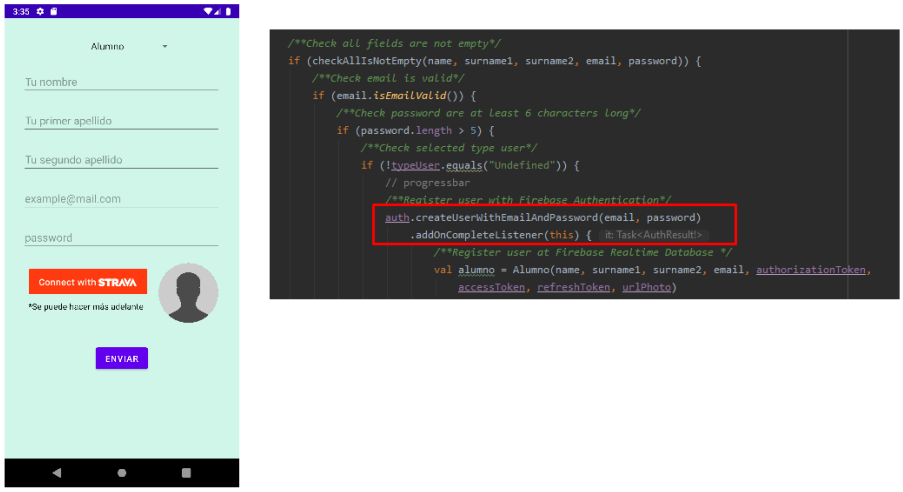


Figura 37: Ejemplo de uso de la función createUserWithEmailAndPassword

Para iniciar sesión emplearemos la función *signInWithEmailAndPassword* de Firebase:

```
private fun login() {
    loading.startLoading()
    /**Check email and password are not empty*/
    if (binding.editTextEmail.text.toString().isNotEmpty()
            && binding.editTextPassword.text.toString().isNotEmpty()
    ) {
        auth.signInWithEmailAndPassword(
            binding.editTextEmail.text.toString(),
            binding.editTextPassword.text.toString()
        )
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    /**if successful sign in check user type*/
                    val user: FirebaseUser = auth.currentUser
```

Figura 38: Ejemplo de uso de la función signInWithEmailAndPassword

3.2.4. Firebase RealTime Database

Para el backend de la app hemos empleado Firebase Realtime Database. En la fase de diseño tuvimos en cuenta una base de datos relacional como backend de la aplicación. En ese diseño obteníamos una serie de relaciones, en total 6, que se correspondían con las tablas de una base de datos relacional.

Pero Firebase no trabaja con bases de datos relacionales y por lo tanto vamos a realizar algunos ajustes al diseño inicial para acomodarlo a lo que sería más idóneo para el trabajo con una base de datos de tipo objeto.

3.2.4.1. Estructura

Los objetos que tendríamos en nuestra base de datos serían los siguientes:

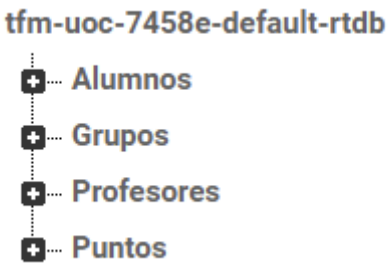


Figura 39: Estructura de la base de datos en Firebase

Como se puede observar, faltan del diseño inicial la relación paradas y la relación logros. A continuación vamos a ver cómo quedan cada uno de ellos y dónde los hemos integrado.

En la figura siguiente tenemos dos alumnos, el primero de ellos simplemente se ha registrado mientras que el segundo se ha apuntado a un grupo y además ha vinculado su cuenta con Strava. De ahí que tenga asignado un idGrupo y los parámetros accessToken, authorizationToken y refreshToken.

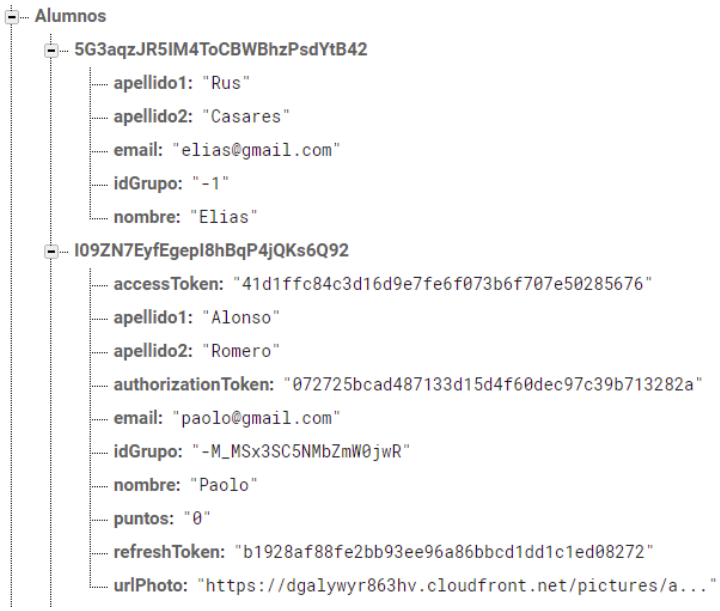


Figura 40: Estructura del objeto Alumnos en Firebase

La forma de acceder a cada alumno será por su key, la cual coincide con el UID de usuario de Firebase Authentication. Esto nos permitirá poder acceder fácilmente y en cualquier momento a los datos del usuario llamando a *Firebase.auth.currentUser.uid*

Ahora pasamos a mostrar los grupos. Entre los grupos y las paradas hay una relación *uno a muchos*, por tanto hemos integrado las paradas como una lista dentro del objeto padre Grupo.

Figura 41: Estructura del objeto Grupos en Firebase

Los grupos también pertenecen a los profesores, pero en este caso se ha optado por introducir un identificador con el uid de cada profesor en el grupo.

A continuación mostramos el objeto profesor, el cual es bastante simple. Al igual que con el objeto Alumno, la key de cada profesor coincide con su uid de usuario de Firebase Authentication. La razón de ello es la misma, poder acceder a los datos del profesor conociendo quien es el usuario que está autenticado en el sistema en ese momento. La key del profesor coincide también con el idGrupo del objeto Grupo para poder asociar los grupos a un profesor.

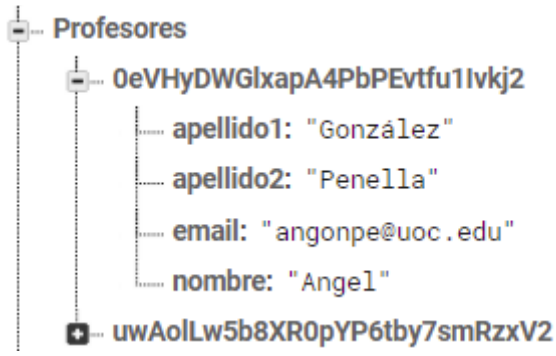


Figura 42: Estructura del objeto Profesores en Firebase

Finalmente tendríamos los puntos. Hemos optado por unificar los puntos y los logros originales en un mismo objeto dado que entre paradas y logros también existe una relación uno a muchos y la estructura de los puntos está pensada como una lista cuyo índice coincide con el de las paradas de los grupos. Lo vemos en la siguiente captura:

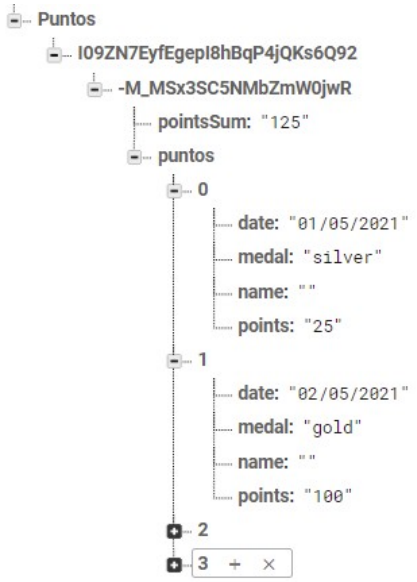


Figura 43: Estructura del objeto Puntos en Firebase

Para acceder a la puntuación y los logros de una parada, hemos de acceder primero mediante el uid del alumno, a continuación mediante el idGrupo y a continuación tendremos la lista de puntos, cuyos índices coinciden con los de la lista de paradas. Está pensado así para que en una futura actualización se pudiera permitir que un alumno perteneciera a diferentes grupos conservando los logros obtenidos en cada uno de ellos.

3.2.4.2. Implementación

En este apartado vamos a ver algunos ejemplos de operaciones habituales que se harán entre nuestra app y la base de datos de Firebase.

Lo primero que hay que destacar es que cuando trabajamos con bases de datos, muchas de las tareas que vamos a realizar son asíncronas. Es decir, no podemos escribir una línea para recuperar los datos de un alumno y de forma secuencial en la línea siguiente actualizar la vista puesto que los datos del alumno es posible que aún no estén disponibles.

Por ejemplo, si queremos mostrar en pantalla los datos de un alumno, debemos leerlos de la base de datos, y cuando recibamos una respuesta, actualizar la vista. Todo ello lo haremos mediante el uso de funciones de callback. Podemos ver un ejemplo en la siguiente captura:

Figura 44: Ejemplo de obtención de datos del Alumno autenticado en Firebase

A la izquierda tenemos la función getStudent de la clase *myFirebaseAPI* y a la derecha tenemos la llamada a la función desde el método *onCreate* de una actividad. De este código vamos a destacar varias cosas:

- Se lee un objeto concreto a través del path y encadenando funciones child para formar la ruta. Así, el alumno que buscamos estará en el path: tfm-uoc-7458e-default-rtdb/Alumnos/uidDelAlumno
- Cuando se obtienen los datos se ejecuta el interior de la función onDataChanged. Los datos obtenidos se encuentran en la variable *snapshot* y son una cadena de texto que puede transformarse a un objeto que implemente el interfaz *Serializable* como es en nuestro caso la clase Alumno
- Dentro de las funciones *onDataChanged* y *onCancelled* no se puede devolver un valor puesto que estas funciones se ejecutan cuando se obtienen los datos, así que lo que se puede hacer es llamar a otra función que realice una acción con los datos obtenidos.

- Cuando se llama a la función *getStudent* de la clase *myFirebaseAPI*, se le pasa la función de callback que comentamos anteriormente.
- Es después de obtenidos los datos del alumno, cuando se actualiza la vista con *updateUI(alu)*

A continuación vamos a mostrar otro ejemplo en el que se obtiene una lista de grupos de un profesor determinado y se muestran en un recyclerview.

Figura 45: Ejemplo de obtención de los grupos del profesor autenticado en Firebase

A la izquierda tenemos la función getGroups de la clase myFirebaseAPI. En esta función se comprueba en la base de datos todos los grupos cuyo idGrupo coincide con uid del usuario que está autenticado en ese momento y se devuelve una lista de grupos.

A la derecha se tienen la llamada a a la función con la función de callback que con los datos obtenidos inicializará el recyclerView para mostrar los grupos.

Para acabar mostramos un ejemplo algo más completo con varias llamadas a la clase myFirebaseAPI para realizar una acción concreta: que un alumno se inscriba a un grupo.

Para ello, con el código de inscripción se deberá obtener el id del grupo para asignarlo al campo idGrupo del alumno y actualizarlo en la base de datos. A continuación se deberá consultar el número de estaciones del grupo para crear un objeto en la base de datos con las puntuaciones (inicialmente vacías). Veámoslo en la siguiente figura:

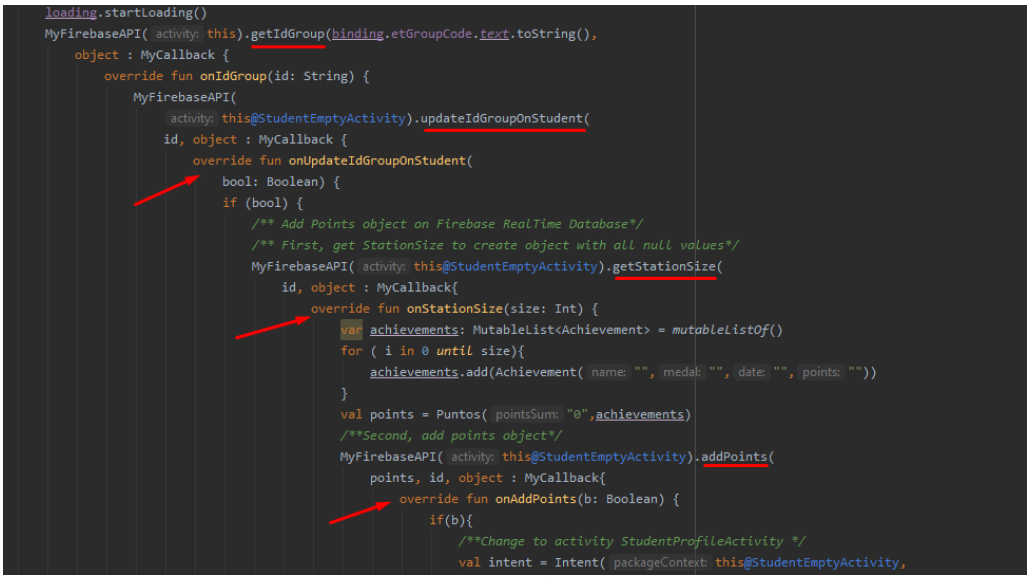


Figura 46: Ejemplo de inscripción de un alumno a un grupo

Como vemos vamos enlazando diferentes llamadas, primero *getIdGroup*, a continuación *updateIdGroupOnStudent*, *getStationSize* y finalmente *addPoints*. Cuando todas ellas se realizan de forma correcta, se procede a cambiar de actividad, donde el alumno verá sus datos actualizados.

3.2.2. Strava

Como comentamos en el punto 1 del trabajo, vamos a emplear la aplicación Strava para realizar el seguimiento de las actividades propuestas en las paradas. Los usuarios de la aplicación, los alumnos, registrarán sus actividades con la aplicación de Strava y en la app de MetroSalud se comprobará si se han completado o no.

Strava es una red social para deportistas nacida en el año 2009 que actualmente alcanza unas cotas de popularidad muy elevadas. Hoy en día se ha convertido en la aplicación de referencia en la que deportistas de todo el mundo llevan el seguimiento de sus actividades y las comparten con amigos y seguidores. A fecha de abril de este mismo año 2021 cuenta con más de 80 millones de usuarios repartidos por todo el mundo [10].

La empresa norteamericana proporciona a los desarrolladores de aplicaciones una API para poder conectar a sus servicios y obtener acceso a la información de los perfiles de usuarios. En MetroSalud vamos a utilizar esta API para que los alumnos puedan vincular su cuenta con la cuenta de Strava y así poder tener acceso a las actividades

3.2.2.1. Primeros pasos con la API de Strava

Vamos a emplear la documentación que proporciona Strava para utilizar su API en nuestra aplicación[6].

En primer lugar nos crearemos una cuenta en Strava o utilizaremos la nuestra personal, y en ajustes configuraremos el apartado “Mi aplicación API”. En este apartado tenemos que anotar el id de nuestra aplicación y el “client secret” que serán los que tendremos que proporcionar en nuestras conexiones a Strava para identificar a nuestra aplicación.

El uso de este API nos permite hacer 100 solicitudes en un intervalo de 15 minutos con un máximo de 1000 diarias que para los objetivos de nuestra aplicación son más que suficientes

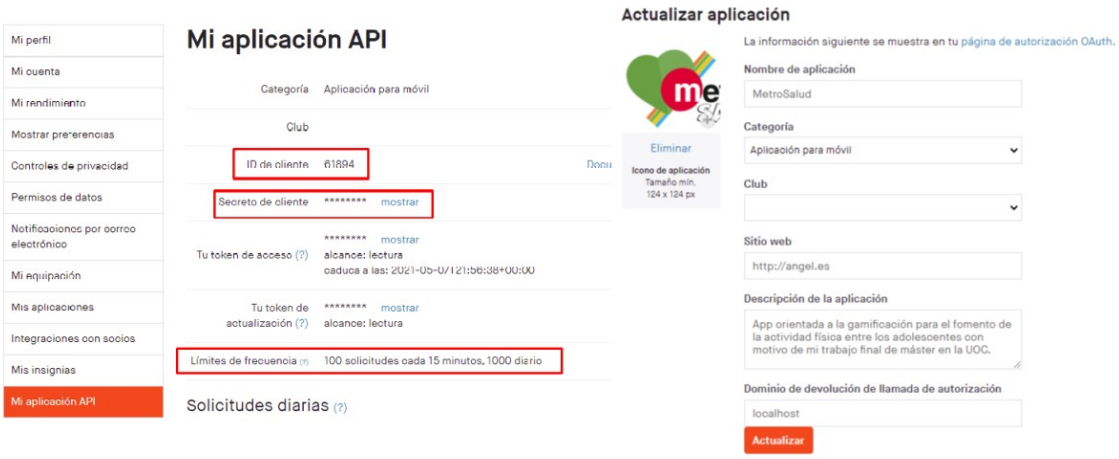


Figura 47: Pasos iniciales con la API de Strava

Para obtener datos de los deportistas vamos tener que pedir a los usuarios que inicien sesión con Strava y acepten unos permisos utilizando OAuth 2.0 [7]. Oauth 2.0 es un estándar abierto para la autenticación segura de APIs y que permite al usuario final otorgar acceso a sus recursos de un sitio en otro, en nuestro caso la vamos a emplear para que nuestros usuarios puedan acceder a sus datos de Strava en MetroSalud.

El proceso de autorización se describe con todo detalle en [7] y consiste básicamente en que nuestra aplicación va a solicitar al usuario que inicie sesión en Strava para aceptar los permisos necesarios. Después de aceptar los permisos, Strava redirigirá a la URL especificada por la aplicación y en la respuesta incluirá un *authorization code* que la aplicación empleará par obtener un *access token* y un *refresh code.* El access token servirá para acceder a los recursos privados del usuario y el refresh code para obtener un nuevo access token cuando éste caduque.

Antes de ponernos a programar en nuestra aplicación el proceso de autenticación con la API de Strava y con el fin de comprenderlo mejor, vamos a realizar todo el proceso mediante la herramienta POSTMAN. Para ello vamos a definir las siguientes 5 peticiones:

Figura 48: Peticiones POST / GET a la API de Strava

La primera de ellas será una petición GET a http://www.strava.com/oauth/authorize con los siguientes parámetros:

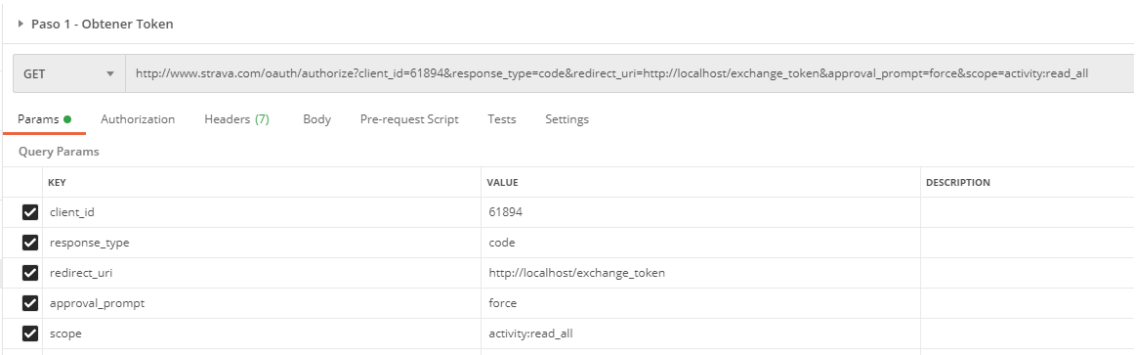


Figura 49: Petición GET para generar access token y refresh token

Después de iniciar sesión y aceptar los permisos obtendremos el authorization code que nos servirá para generar el access token y el refresh token en el paso 2.

Ahora tendremos que hacer una petición POST a http://www.strava.com/oauth/token pasando como parámetros el id de cliente de la aplicación, el secret code, el código de autorización obtenido en el paso anterior y el tipo de permiso:

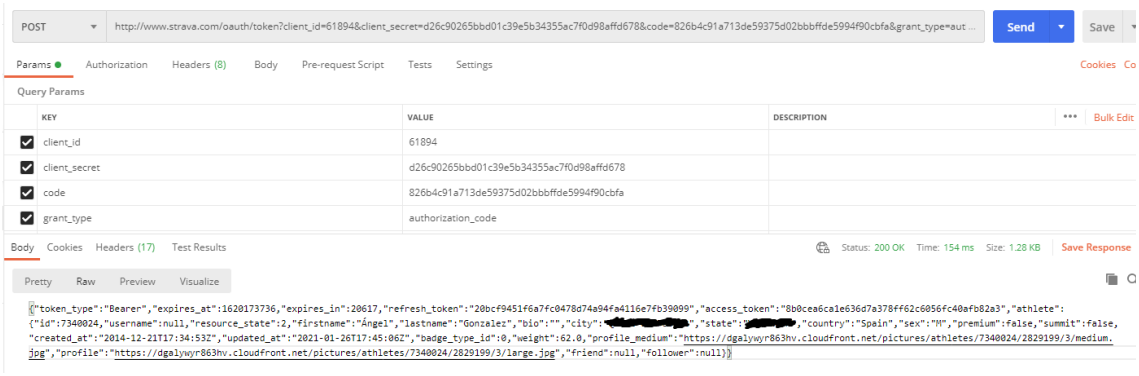


Figura 50: Respuesta de la petición GET

Como podemos observar, la respuesta incluye algunos datos del usuario junto con el access token y el refresh code. El access token será el que deberemos usar para acceder a los datos del usuario hasta que éste expire. Una vez haya expirado deberemos regenerar el access token utilizando el refresh code. Esto vamos a verlo en el paso 3 donde haremos una petición POST a http://www.strava.com/oauth/token con los parámetros siguientes: id de cliente de la aplicación, el secret code, el tipo de permiso y el refresh code:

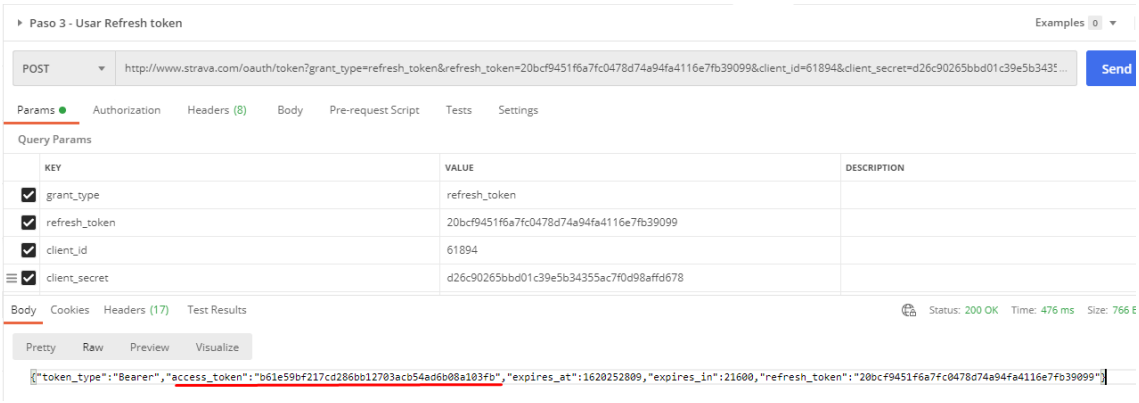


Figura 51: Petición POST y respuesta con el nuevo access token

Como puede observarse, en la respuesta hemos obtenido el nuevo access token junto con el tiempo para que expire.

Finalmente vamos a realizar la petición GET que previsiblemente emplearemos para obtener la información que necesitamos de los usuarios. Se trata de una consulta para obtener un listado de actividades entre una fecha de inicio y una fecha de fin. Y con el id de la actividad, aún podríamos acceder a más información.

Vamos a hacer una petición GET a https://www.strava.com/api/v3/athlete/activities con los siguientes parámetros: before y after que serán las fechas de inicio y fin en formato epoch Timestamp [12], page y per page para filtrar máximo de resultados por página y el access token. Y como podemos observar a continuación la API nos devolverá un array con todas las actividades del usuario en el intervalo de fechas.

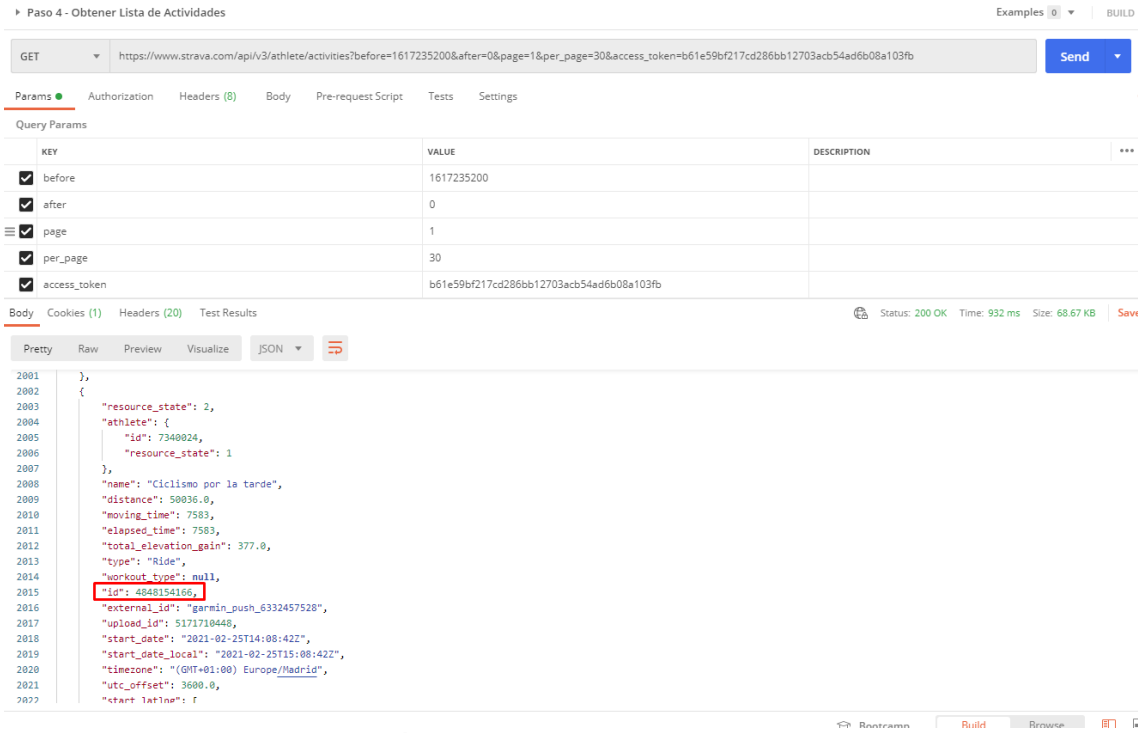


Figura 52: Petición GET y respuesta con datos de usuario

Con los datos obtenidos en esta consulta tendremos bastante para comprobar si los usuarios completan las actividades que proponga el profesor en cada parada. Algunos ejemplos de paradas básicas podrían ser las siguientes:

- Total de minutos acumulados en todas las actividades en un intervalo de tiempo específico. Para ello no hay más que sumar todos los valores de la clave "elapsed_time" de cada actividad
- Total de metros de desnivel acumulados en todas las actividades de bicicleta en un intervalo de tiempo específico. Para ello habrá que sumar todos los valores de la clave “total_elevation_gain” de aquellas actividades cuya clave “type” tenga el valor *Ride*
- Total de kilómetros realizados en una actividad durante un intervalo de tiempo concreto. Para ello habría que comprobar de entre todas las actividades cuya clave “type” sea *Run,* aquella que la clave “distance” sea más elevada.

3.2.2.2. Implementación en Kotlin del proceso para vincular tu cuenta de Strava

Cuando los usuarios hagan click en “conectar con Strava” vamos a realizar una petición GET y otra POST. En la primera de ella vamos a obtener

el authorization code, y en la segunda y utilizando este token, vamos a obtener el refresh code y el access token necesarios para obtener los datos.

En la primera petición, va a ser necesario que el usuario se conecte a Strava y autorize a nuestra aplicación a recopilar los datos necesarios, es por ello que la vamos a implementar mediante una actividad que cargue únicamente un *WebView* [13]. Un *WebView* no es más que una vista de Android que permite cargar páginas web.

En la imagen siguiente se muestra el código de dicha actividad. Como podemos observar construimos la petición con la clase *Uri* y posteriormente se carga sobre el *WebView*. Cuando el usuario introduce sus credenciales en la web y autoriza a nuestra aplicación a recopilar los datos, capturamos la url devuelta sobreescribiendo el método *shouldOverrideUrlLoading*. Aquí nos quedaremos con el valor del parámetro *code.*

```
class AuthorizeStrava : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_authorize_strava)
        val myWebView: WebView = findViewById(R.id.webview)
        var intentUri: Uri = Uri.parse( uriString: "https://www.strava.com/oauth/mobile/authorize")
            .buildUpon()
            .appendQueryParameter("client_id", "61894")
            .appendQueryParameter("redirect_uri", "http://localhost")
            .appendQueryParameter("response_type", "code")
            .appendQueryParameter("approval_prompt", "force")
            .appendQueryParameter("scope", "activity:write,read")
            .build()
        myWebView.webViewClient = object : WebViewClient() {
            override fun shouldOverrideUrlLoading(view: WebView?, url: String?): Boolean {
                val urlUri: Uri = Uri.parse(url)
                if(urlUri.toString().contains( other: "localhost/?state")){ /**We assume response starts with localhost/?state..... */
                    val code = urlUri.getQueryParameter( key: "code")
                    setResult(RESULT_OK, Intent().putExtra( name: "code", code))
                    finish()
                    return true
                }
                return false
            }
        }
        myWebView.loadUrl(intentUri.toString())
    }
}
```

Figura 53: Código de la actividad AuthorizeStrava.kt

Una vez realizada esta primera petición, se retorna a la actividad anterior para realizar la petición POST. En el *onActivityResult* de esta actividad llamaremos a la función siguiente:

```kotlin
private fun postRequest(client_id: String, client_secret: String, code: String, grant_type: String, callback: Callback): Call {
    val client = OkHttpClient()
    val url = URL( spec: "https://www.strava.com/oauth/token")

    /**Use jackson*/
    val mapperAll = ObjectMapper()
    val jacksonObj = mapperAll.createObjectNode()
    jacksonObj.put( fieldName: "client_id", client_id)
    jacksonObj.put( fieldName: "client_secret", client_secret)
    jacksonObj.put( fieldName: "code", code)
    jacksonObj.put( fieldName: "grant_type", grant_type)
    val jacksonString = jacksonObj.toString()

    val mediaType = "application/json; charset=utf-8".toMediaType()
    val body = jacksonString.toRequestBody(mediaType)

    val request = Request.Builder()
        .url(url)
        .post(body)
        .build()

    val call: Call = client.newCall(request)
    call.enqueue(callback)
    return call
}
```

Figura 54: Función postRequest para obtener el refresh token y el access token

En esta función vamos a utilizar 2 librerías externas que nos van a facilitar mucho la comunicación con el servidor de Strava. La primera de ellas es OkHttp [15] la cual nos va a permitir hacer una llamada asíncrona con la petición POST. Y la segunda es Jackson que también de forma muy sencilla nos va a permitir construir objetos en formato JSON y acceder a ellos.

Como podemos observar el uso de OkHttp es sencillo: tenemos que montar la petición request con el *Request.Builder()* y a continuación realizar la llamada con el cliente OkHttp.

En el *onActivityResult* que llama a esta función sobreescribiremos los métodos *onFailure* y *onSucces.* En caso que la petición se realice de forma correcta procederemos a registrar al usuario con los datos necesarios para conectar con Strava así como con su foto de perfil en dicha red social.

3.2.2.3. Implementación en Kotlin de la obtención de datos desde Strava

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    // Step 1: Requested authorization code
    if (requestCode == 1) {
        if (resultCode == RESULT_OK) {
            val token: String? = data?.getStringExtra( name: "code")
            authorizationToken = token
            loading = LoadingDialog( mActivity: this)
            loading.startLoading()
            // Step 2: Requested access token and refresh token
            token?.let {  it: String
                postRequest( client_id: "61894",
                    client_secret: "d26c90265bbd01c39e5b34355ac7f0d98affd678",
                    it,
                    grant_type: "authorization_code", object : Callback {

                        override fun onFailure(call: Call, e: IOException) {
                            loading.isDismiss()
                        }

                        override fun onResponse(call: Call, response: Response) {
                            if (response.isSuccessful) {
                                val responseStr: String = response.body!!.string()
                                val mapperAll = ObjectMapper()
```

Figura 55: Llamada a la función postRequest después de obtener el Authorization code

A continuación, mostramos a modo de ejemplo la pantalla de registro de usuario, donde un alumno introduce sus datos de registro y conecta con Strava para vincular su cuenta (obtener authorization code, access token, refresh token y su foto de perfil):

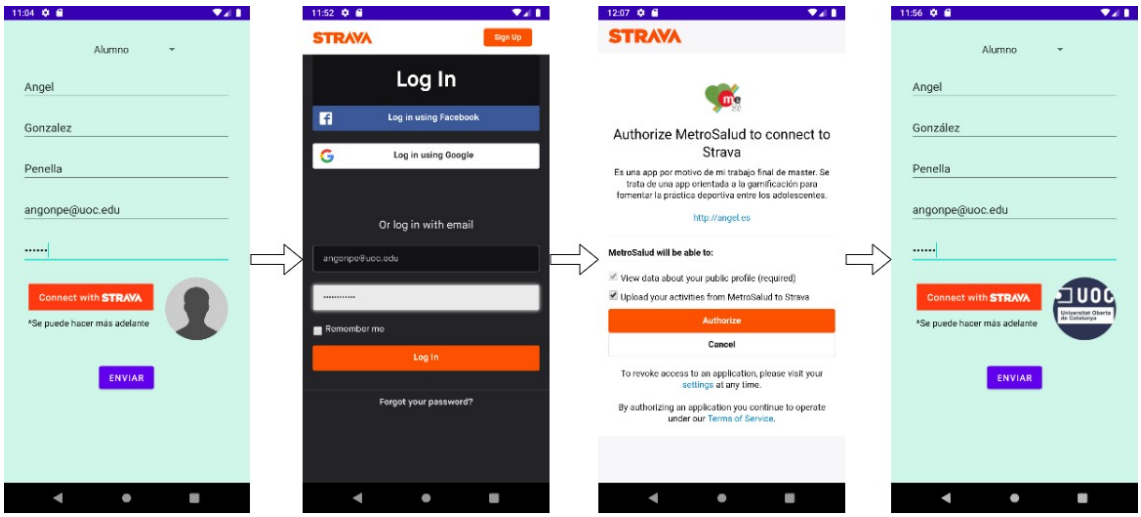
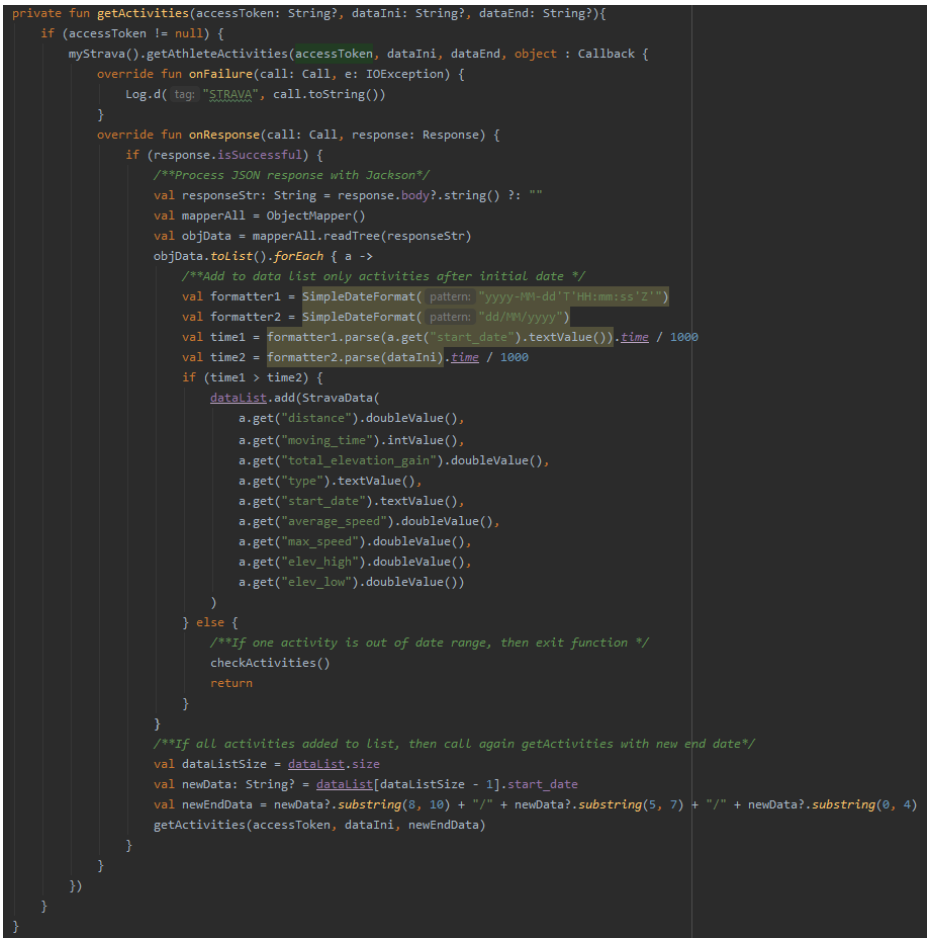


Figura 56: Pantallas para vincular MetroSalud con Strava

3.2.2.4. Implementación en Kotlin de la obtención de actividades de un atleta



```
private fun getActivities(accessToken: String?, dataIni: String?, dataEnd: String?){
    if (accessToken != null) {
        myStrava().getAthleteActivities(accessToken, dataIni, dataEnd, object : Callback {
            override fun onFailure(call: Call, e: IOException) {
                Log.d( tag: "STRAVA", call.toString())
            }
            override fun onResponse(call: Call, response: Response) {
                if (response.isSuccessful) {
                    /**Process JSON response with Jackson*/
                    val responseStr: String = response.body?.string() ?: ""
                    val mapperAll = ObjectMapper()
                    val objData = mapperAll.readTree(responseStr)
                    objData.toList().forEach { a ->
                        /**Add to data list only activities after initial date */
                        val formatter1 = SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:ss'Z'")
                        val formatter2 = SimpleDateFormat( pattern: "dd/MM/yyyy")
                        val time1 = formatter1.parse(a.get("start_date").textValue()).time / 1000
                        val time2 = formatter2.parse(dataIni).time / 1000
                        if (time1 > time2) {
                            dataList.add(StravaData(
                                a.get("distance").doubleValue(),
                                a.get("moving_time").intValue(),
                                a.get("total_elevation_gain").doubleValue(),
                                a.get("type").textValue(),
                                a.get("start_date").textValue(),
                                a.get("average_speed").doubleValue(),
                                a.get("max_speed").doubleValue(),
                                a.get("elev_high").doubleValue(),
                                a.get("elev_low").doubleValue())
                            )
                        } else {
                            /**If one activity is out of date range, then exit function */
                            checkActivities()
                            return
                        }
                    }
                    /**If all activities added to list, then call again getActivities with new end date*/
                    val dataListSize = dataList.size
                    val newData: String? = dataList[dataListSize - 1].start_date
                    val newEndData = newData?.substring(8, 10) + "/" + newData?.substring(5, 7) + "/" + newData?.substring(0, 4)
                    getActivities(accessToken, dataIni, newEndData)
                }
            }
        })
    }
}
```

Figura 57: Función recursiva para obtener todas las actividades de un atleta entre unas fechas dadas

En la captura anterior se ve el código de la función que nos va a permitir obtener los datos relevantes de todas las actividades de un atleta entre unas fechas concretas.

Como se puede observar se hace una llamada a una función que hará una petición GET a Strava. Esta llamada incluye un access token válido junto con un intervalo de fechas para el que se obtendrán las actividades.

La respuesta es un JSON que procesaremos y para cada actividad nos guardaremos los datos que van a ser relevantes en el array *dataList,* de ámbito global para toda la actividad, para posteriormente procesarlo y dar la recompensa correspondiente al alumno.

Como aspecto relevante, destacar que si la última actividad obtenida tiene una fecha superior a la indicada en el intervalo como fecha mínima, volveremos a realizar otra llamada a la misma función pero cambiando los rangos de fechas.

Esto es así porque, aunque de la documentación de la API de Strava se pueda desprender que es posible obtener las actividades entre un rango de fechas de forma directa, en realidad no lo es. Y por ello podemos obtenerlas de 30 en 30 comprobando si la última tiene una fecha inferior a la mínima requerida [25].

3.3. Pruebas

En general, cuando se habla de pruebas de software se distingue entre pruebas manuales y pruebas de software automatizadas. Las primeras son llevadas a cabo por personas que interactúan con la aplicación, mientras que las segundas son realizadas por el propio ordenador que ejecuta un *script* con las pruebas a realizar. Estos test pueden ser sencillos, verificar que una clase o método funcione, hasta asegurar una secuencia de acciones con una interfaz de usuario.

Existen diferentes tipos de pruebas [18]:

- Las pruebas unitarias son de bajo nivel y consisten en probar de forma individual las funciones de las clases y componentes de la aplicación.
- Las pruebas de integración verifican que los diferentes servicios o módulos utilizados por la aplicación funcionen correctamente en conjunto.
- Las pruebas funcionales verifican que la aplicación se comporta según lo esperado cumpliendo con los requerimientos de negocio.
- Las pruebas end-to-end replican el comportamiento de los usuarios con la aplicación y verifican que los flujos que sigue un usuario sean como se espera.
- Las pruebas de humo son pruebas rápidas que verifican la funcionalidad básica de una aplicación.
- Las pruebas de aceptación son pruebas formales que verifican que un sistema satisface los requerimientos de negocio.

En nuestro trabajo nos vamos a centrar en las pruebas manuales, aunque podríamos emplear mockito o espresso para realizar pruebas unitarias de forma sencilla y espresso para realizar pruebas funcionales.

Cabe destacar que durante el proceso de implementación ya se ha verificado que las funciones programadas se comportan correctamente.

A continuación realizaré una serie de pruebas end-to-end o de humo para verificar algunas funcionalidades básicas como: registro de usuario como profesor y como alumno, inicio de sesión como profesor y como alumno,

creación de grupos, edición de paradas, inscripción a un curso, borrar un grupo, listar clasificación o completar una parada. Algunas de ellas las documentaré con capturas paso a paso y otras simplemente anotaré su verificación.

- Prueba 1: Registro de usuario profesor. Pasada correctamente.

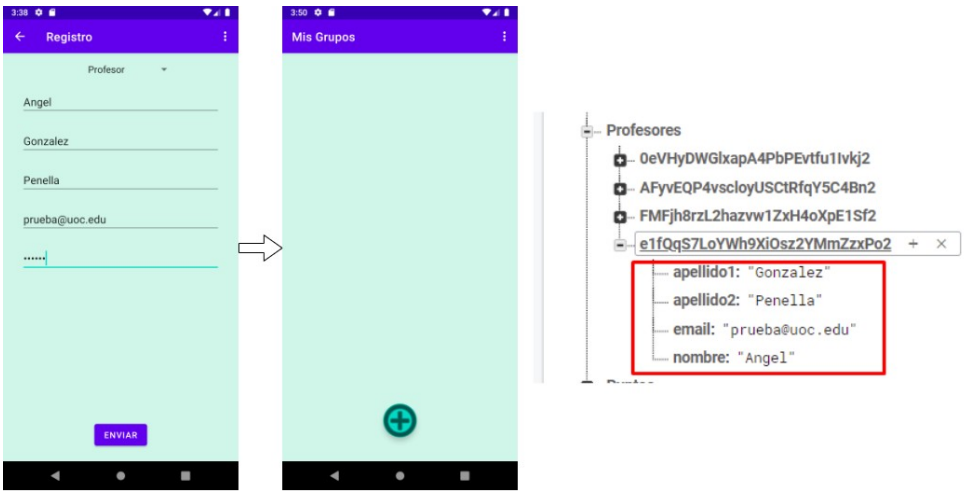


Figura 58: Prueba Registro usuario Profesor

- Prueba 2: Creación de grupo por defecto. Pasada correctamente.

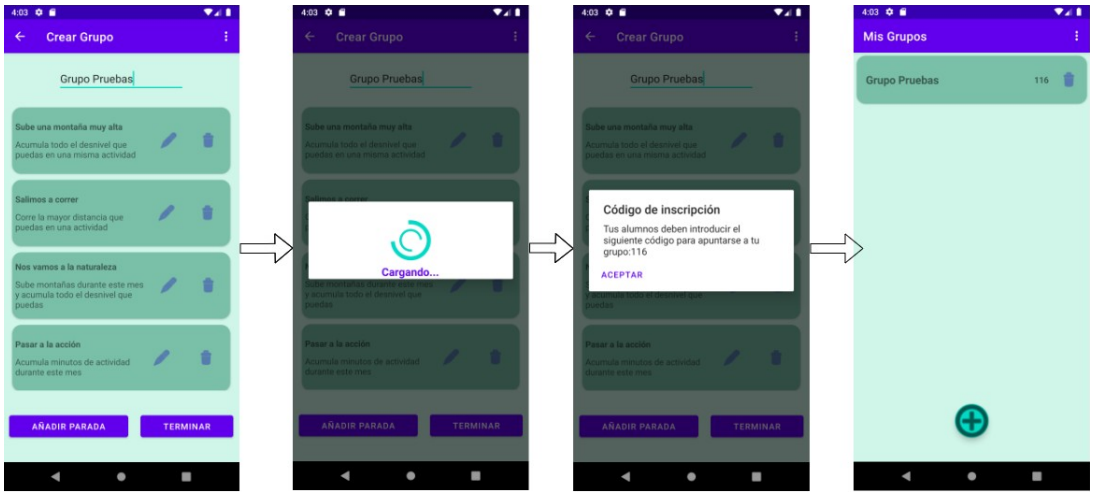


Figura 59: Prueba Creación de grupo por defecto

- Prueba 3: Creación de grupo editando y añadiendo paradas. Pasada correctamente.

  Se comprueba que se deben añadir al menos cuatro paradas y el nombre para poder crease el grupo.

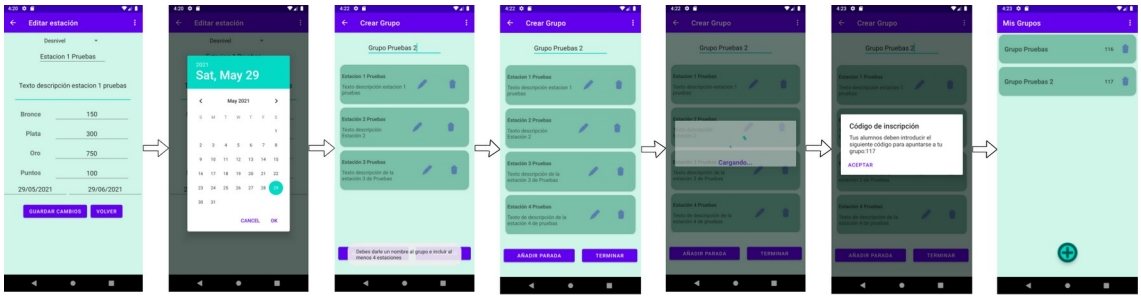


Figura 60: Prueba de Creación de Grupo editanto y añadiendo paradas

- Prueba 4: Eliminación de grupo. Pasada correctamente

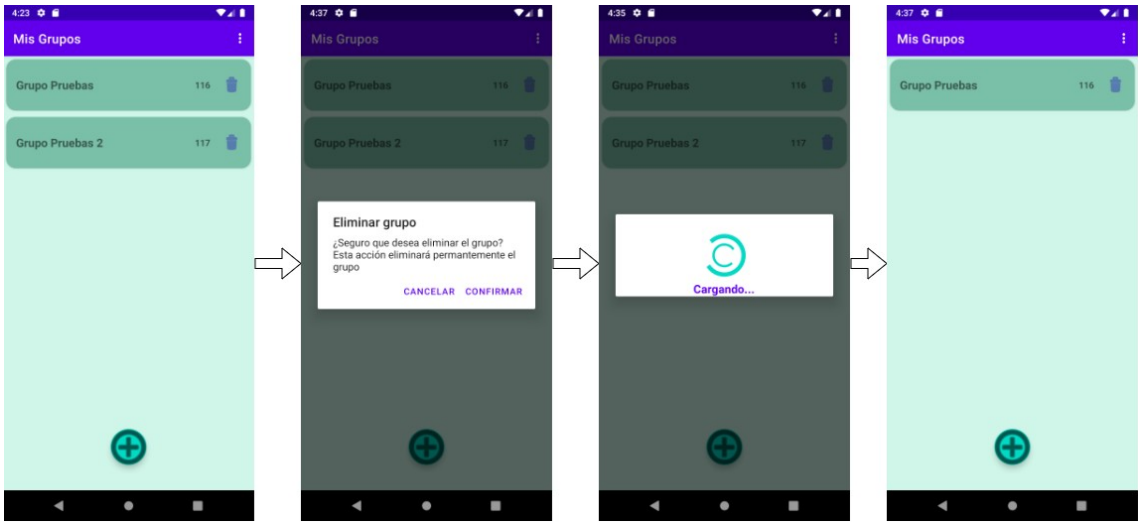


Figura 61: Prueba Eliminación de Grupo

- Prueba 5: Creación de usuario alumno vinculando con Strava. Pasada correctamente. Consultar la figura 56 para ver los detalles
- Prueba 6: Creación de usuario alumno sin vincular con Strava. Pasada correctamente
- Prueba 7: Ver clasificaciones. Pasada correctamente

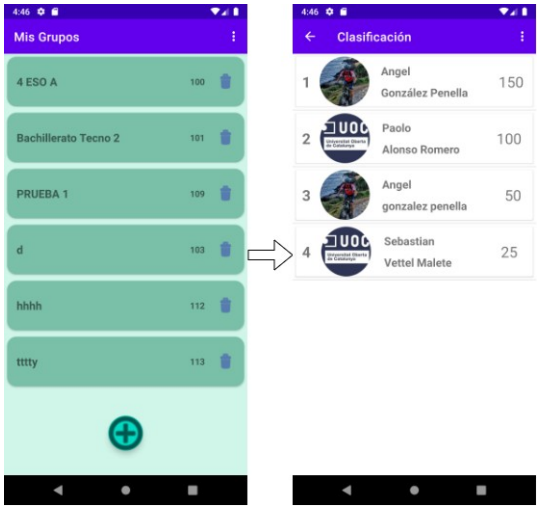


Figura 62: Prueba Ver clasificaciones

- Prueba 8: Inscripción a un curso. Pasada correctamente

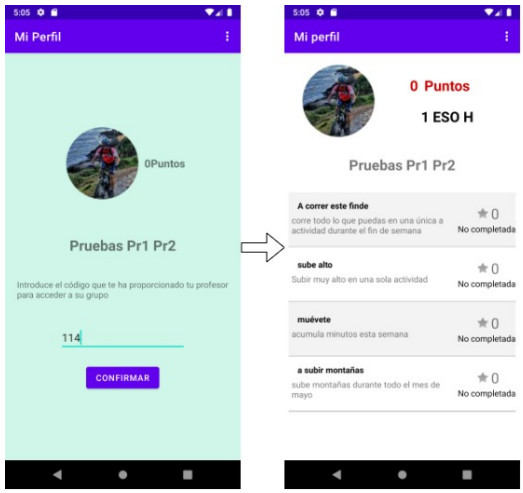


Figura 63: Prueba de inscripción en un grupo

- Prueba 9: No cumplir los requisitos para completar una estación. Pasada correctamente

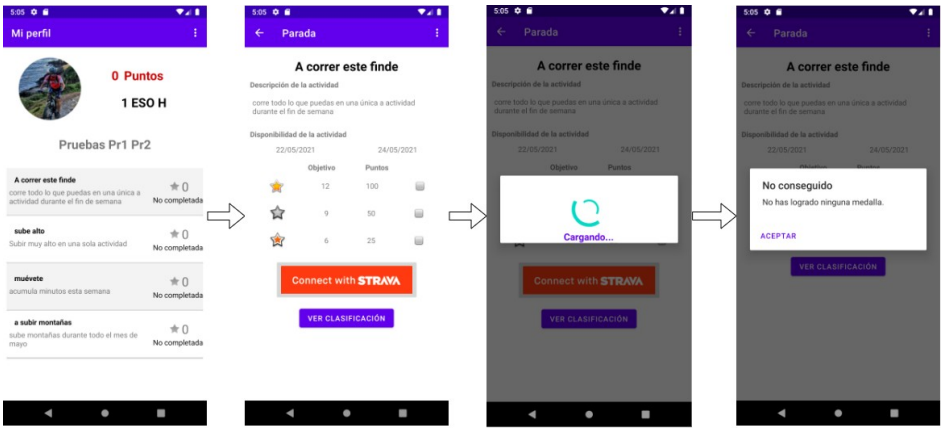


Figura 64: Prueba de no cumplir los requisitos para completar una estación

- Prueba 10: Completar una estación. Pasada correctamente

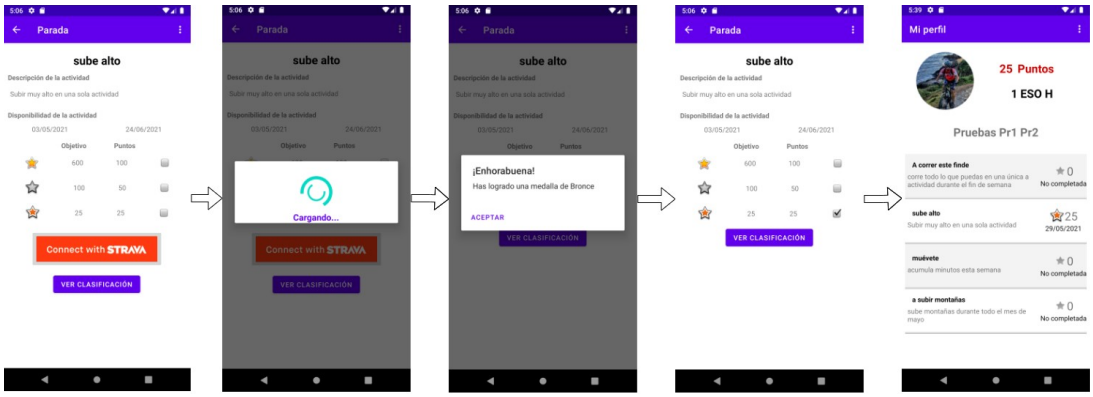


Figura 65: Completar una estación

- Prueba 11: Salir de un grupo. Pasada correctamente
- Prueba 12: Restablecer la contraseña del usuario. Pasada correctamente
- Prueba 13: Cerrar sesión en cualquier momento. Pasada correctamente
- Prueba 14: Salir de la aplicación en cualquier momento. Pasada correctamente.

3.3.1. Errores encontrados durante la fase de testeo

Durante la fase de testeo y pruebas nos hemos encontrado, como es normal, numerosos errores o bugs. Vamos a comentar los más interesantes.

Hacer múltiples clicks sobre un botón:

En primer lugar destaco un error que se produce cuando probamos la aplicación en un terminal real. Cuando se prueba la app en el emulador de android, normalmente nunca hacemos doble click sobre un botón cualquiera. Pero en cambio, en un teléfono móvil sí que puede ser habitual hacer más de una pulsación involuntaria al apretar un botón.

Esta inocente acción puede desencadenar diversos comportamientos no deseados, por ejemplo: puede iniciarse más de una vez una misma actividad o pueden iniciarse más de una misma llamada a un servidor para leer o escribir datos.

Por ello, como solución, hemos implementado la clase SafeClickListener para capturar los eventos setOnClickListener y comprobar si ha pasado más de 1 segundo desde el último click para realizar la acción [20].

Figura 66: Clase SafeClickListener. Ejemplo de uso con setSafeOnClickListener

Lanzar múltiples peticiones a un servidor al hacer múltiples clicks sobre un botón:

Otro error similar al anterior es el que puede producirse cuando hacemos una petición para recuperar o escribir datos al servidor de backend o al servidor de Strava.

Según la petición que hagamos, que puede requerir distintas operaciones sobre la base de datos, y según el estado de la red, una acción puede tardar más de lo previsto.

Así, según las pruebas realizadas, en ocasiones la petición al servidor de Strava para obtener los datos de actividades de un atleta puede tardar algo más de un segundo. Para evitar que el usuario lance más peticiones y

provoque comportamientos indeseados por parte de la app, es conveniente desactivar algunos elementos de la interfaz de usuario.

Por ello, se ha optado por introducir en estos momentos una pantalla de carga.

AddValueEventListener vs addSingleValueEventListener en Firebase Realtime Database

Al programar la parte de comunicación con la base de datos de Firebase, en cada operación que se realiza se añade un *listener* para estar a la escucha de la finalización correcta de la acción.

En algunas de estas operaciones hemos cometido el error de añadir un ValueEventListener en vez de un addSingleValueEventListener. Es fácil cometer este error puesto que el propio IDE de Android te sugiere determinados métodos a medida que escribimos, y por nombre se parecen bastante.

La diferencia entre ambos es que el primero se mantiene a la escucha constantemente mientras que el segundo se mantiene a la escucha hasta que se produce un cambio y después deja de escuchar [23]. Como algunas de nuestras operaciones anidan diferentes llamadas de callback que se van produciendo a medida que se va obteniendo respuestas del servidor, en algunos casos se obtenían comportamientos impredecibles de la aplicación.

Un ejemplo de comportamiento inadecuado es el siguiente: Al crear un grupo se realiza una escritura sobre una referencia en la base de datos, si a continuación se elimina ese grupo se realizará otra escritura sobre la misma referencia. Si la primera operación se mantiene a la escucha de sólo el primer evento, funcionará correctamente, mientras que si se mantiene a la escucha de eventos la operación de añadir grupo también responderá al eliminarse un grupo, obteniéndose así un comportamiento no deseado.

3. Conclusiones

El resultado final del trabajo es una aplicación móvil para dispositivos Android, funcional, que puede utilizarse con los alumnos para fomentar la práctica deportiva planteando retos y otorgando recompensas según los grados de consecución de cada uno de ellos. La app permite a los profesores proponer retos y hacer un seguimiento de los mismos y tiene un diseño sencillo e intuitivo.

Igualmente hemos profundizado en el conocimiento del desarrollo de aplicaciones móviles, concretamente para dispositivos Android y utilizando el lenguaje de programación Kotlin tal y como nos planteamos al inicio.

Es por todo ello que considero que se han alcanzado mayoritariamente los objetivos planteados al inicio del trabajo.

Este trabajo final de máster supone la conclusión de mis estudios en el Máster Universitario de Desarrollo de Aplicaciones móviles. Durante este tiempo me he acercado al desarrollo de aplicaciones móviles desde diferentes tecnologías. Cada asignatura que he cursado me ha permitido experimentar con algo nuevo, así he estudiado programación para dispositivos Android con Java, para dispositivos iOS con Swift, para aplicaciones web con Angular y React, para aplicaciones multiplataforma con Flutter y para videojuegos con C# y Unity.

Como conclusión a mis estudios y tal y como he comentado en la memoria, decidí utilizar también alguna tecnología desconocida para mí y es por ello que elegí programar en Kotlin para dispositivos Android

Una de las primeras conclusiones que extraigo es que tener conocimientos más o menos generales de varias tecnologías te permite una rápida adaptación a tecnologías nuevas. En este sentido la adaptación a Kotlin para este trabajo, ha sido rápida y para nada complicada. Pero por otra parte, tengo también la sensación que para progresar en esta área concreta del desarrollo de aplicaciones se debe elegir una tecnología en la que centrar los esfuerzos para poderle sacar todo el provecho posible.

Considero que con este trabajo final de máster he demostrado que soy capaz de afrontar el diseño y desarrollo de un proyecto con una envergadura y tamaño más o menos considerable como el propuesto.

Respecto a la planificación y metodología empleada a lo largo del trabajo querría comentar que la elaboración de un diagrama de Gantt precisando todas las actividades y el planteamiento de tres entregas parciales ha sido todo un acierto.

En lo relacionado con la planificación he tenido que introducir algún cambio motivado por causas laborales que han supuesto alguna demora en las entregas parciales y a un reajuste en cuanto a la dedicación de horas diarias. Además, la fase de implementación ha requerido también un número superior de horas respecto al planificado inicialmente. Ésto ha sido motivado por una mayor dificultad para esta fase respecto a la estimada en el inicio.

Por todo ello, a partir del mes de abril y hasta prácticamente la entrega del trabajo, la dedicación horaria en algunas jornadas puede haber alcanzado las 8 horas diarias. Sin duda, el hecho de comprometerse a realizar tres entregas y no desviarse muchos días en cada una de ellas, junto con el hecho de detallar todas las actividades con su carga de trabajo estimada, ha ayudado a garantizar el éxito del trabajo.

Respecto a las líneas de trabajo futuro que no se han podido explorar en este trabajo, son muchas y algunas de ellas muy interesantes.

La primera de todas es el despliegue de la aplicación en uno o dos grupos para hacer una evaluación de la aplicación sobre un entorno real. La idea es hacerlo durante el curso escolar 2021-2022. Seguro que surgirán ideas para implementar nuevas funcionalidades y también errores que depurar.

Igualmente, me hubiera gustado profundizar más en la mejora del aspecto visual de la aplicación. La app es perfectamente funcional pero podría haber introducido otros y mejores elementos visuales. Este es un claro ejemplo de lo que comentaba anteriormente sobre centrarse en una tecnología para dominarla y sacarle todo el partido posible.

Otra de las líneas de trabajo donde se podría incidir es en los aspectos de logros y recompensas de la app a los usuarios. En este sentido se podría implementar los *Servicios de juego de Google Play* [25] para otorgar logros e incluir más clasificaciones entre amigos.

Prescindir de la API de Strava o utilizarla como un complemento, sería también otra línea de trabajo. Podría implementarse en la aplicación la posibilidad de que ésta capturara directamente en el teléfono móvil el track GPS con todos los datos relevantes de una actividad deportiva.

El desarrollo de una app multiplataforma o complementarla con una app nativa para dispositivos iOS sería también una línea de futuro evidente para MetroSalud.

Como se observa, las opciones de ampliación del trabajo son muchas y éstas son solo una muestra de todo lo que se puede hacer si se dispone de tiempo.

Además, aunque se aleje un poco del tema del trabajo, desde un punto de vista personal, como un apasionado del deporte, de los datos y de

determinadas métricas deportivas, el uso de la API de Strava abre un abanico de posibilidades enorme para el desarrollo de todo tipo de aplicaciones deportivas.

4. Glosario

- MetroSalud. Nombre que recibe la aplicación del TFM.
- Parada o estación. Nombre que reciben una actividad física en la app de MetroSalud.
- Android Studio. Entorno de desarrollo Integrado (IDE) para el desarrollo de aplicaciones móviles para Android.
- Android. Sistema operativo de Google para teléfonos móviles.
- Backend. Se refiere a la arquitectura interna del sitio web o aplicación móvil que asegura que todo funcione de forma correcta. Normalmente se encarga de llamadas a bases de datos y comunicaciones con el servidor.
- Firebase Realtime Database. Es una base de datos NoSQL alojada en la nube, que permite almacenar y sincronizar datos entre usuarios en tiempo real.
- API. Acrónimo de Application Programming Interfaces. Permite que dos aplicaciones o servicios puedan comunicarse entre sí. Sirve para que una app pueda interactuar con otra.
- Postman. Es una aplicación que permite el envío de peticiones HTTP REST sin necesidad de desarrollar un cliente. Sirve para poder hacer pruebas.
- Kotlin. Es un lenguaje de programación orientado a objetos para el desarrollo de aplicaciones para dispositivos Android.
- DCU. Diseño centrado en el usuario. Forma de diseño que persigue la consecución de un producto con la funcionalidad adecuada para los usuarios a los que se dirige.

5. Bibliografía

[1] https://www.gasolfoundation.org/es/estudio-pasos/ 01/03/2021

[2] Diseño centrado en el usuario. Muriel Garreta Domingo, Enric Mor Pera. Segunda edición: septiembre 2020. FUOC

[3] Métodos para el desarrollo de aplicaciones móviles. Robert Ramírez Vique, Helena Boltà Torrell. PID_00246016. UOC

[4] https://living-sun.com/es/collections/102197-kotlin39s-list-missing-ldquoaddrdquo-ldquoremoverdquo-etc-collections-kotlin.html 27/04/2021

[5] https://www.youtube.com/watch?v=VKePyfdSSoQ&ab_channel=SolutionCodeAndroid 01/05/2021

[6] https://developers.strava.com/docs/getting-started/ 02/05/2021

[7] https://developers.strava.com/docs/authentication/ 02/05/2021

[8] https://www.learn2crack.com/2014/01/android-oauth2-webview.html 03/05/2021

[9] https://johncodeos.com/how-to-make-post-get-put-and-delete-requests-with-retrofit-using-kotlin/ 04/05/2021

[10] https://www.oauth.com/oauth2-servers/making-authenticated-requests/refreshing-an-access-token/#:~:text=To%20use%20the%20refresh%20token,well%20as%20the%20client%20credentials. 05/05/2021

[11] https://apnews.com/article/sports-new-york-north-america-new-york-city-marathon-athlete-health-7beb94a9ca6a0c1b799e593d78ff4926 07/05/2021

[12] https://www.epochconverter.com/ 05/05/2021

[13] https://stackoverflow.com/questions/17774176/android-ui-when-can-i-directly-modify-a-view 07/05/2021

[14] https://developer.android.com/reference/android/webkit/WebView 01/05/2021

[15] https://square.github.io/okhttp/ 01/05/2021

[16] https://www.semicolonworld.com/question/44062/how-to-return-datasnapshot-value-as-a-result-of-a-method 02/05/2021

[17] https://stackoverflow.com/questions/10748796/android-how-to-limit-width-of-textview-and-add-three-dots-at-the-end-of-text 10/05/2021

[18] https://programacionymas.com/blog/tipos-de-testing-en-desarrollo-de-software 13/05/2021

[19] https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_15 17/05/2021

[20] https://medium.com/@simonkarmy2004/solving-android-multiple-clicks-problem-kotlin-b99c06135da0 24/05/2021

[21] https://stackoverflow.com/questions/1555109/how-to-stop-edittext-from-gaining-focus-at-activity-startup-in-android 24/05/2021

[22] https://programacionymas.com/blog/como-pedir-fecha-android-usando-date-picker 23/05/2021

[23] https://stackoverflow.com/questions/41579000/difference-between-addvalueeventlistener-and-addlistenerforsinglevalueevent 23/05/2021

[24] https://groups.google.com/g/strava-api/c/fx_oxzektJo 15/05/2021

[25] https://developer.android.com/distribute/best-practices/engage/games-services?hl=es-419 28/05/2021

[26] https://stackoverflow.com/questions/39144629/how-to-add-sha-1-to-android-applicatio 17/04/2021