Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

AllThenics, aplicación móvil para calistenia: arquitectura y persistencia

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Javier Contrí González

Tutores: David de Andrés Martínez

Juan Carlos Ruiz García

Curso 2019/2020

Resumen

Actualmente, la población empieza a concienciarse de la importancia de la salud, por ello, mucha gente sale a la calle a realizar ejercicio físico ya sea en gimnasios, polideportivos o parques. En esta última opción es donde nace la calistenia, un tipo de entrenamiento basado en el propio peso corporal siguiendo un estilo de vida saludable. Esta disciplina está fundada sobre valores positivos como el respeto, la educación, la igualdad, la integración social y la ayuda entre sus miembros. Sin contextualidades políticas, religiosas o económicas, el *street workout* es para todos, independientemente de la edad, el sexo, la etnia o el nivel físico de las personas.

Esta aplicación pretende cubrir la necesidad de muchos calisténicos con el propósito de llevar un registro de sus actividades y ejercicios con el objetivo de progresar y realizar un seguimiento de su evolución, además, de la incorporación de nuevos usuarios que quieran mejorar su condición física y empezar en este deporte, el cual se encuentra en auge en estos momentos.

Este proyecto es un TFG de emprendimiento realizado en el ámbito de la StartInf con la colaboración del alumno Víctor Cardona Lorenzo el cual se centrará en el desarrollo del *front-end* de la aplicación «AllThenics, aplicación móvil para calistenia: desarrollo del front-end», mientras que este trabajo se focalizará en la implementación de la arquitectura y persistencia de AllThenics.

Palabras clave: aplicación; calistenia; móvil; emprendimiento; entrenamiento; back-end; persistencia; arquitectura;

Abstract

Nowadays, the population is beginning to become aware of the importance of health, so many people go out on the streets to do physical exercise, whether in gyms, sports centres or parks. This last option is where calisthenics are born, a type of training based on one's own body weight following a healthy lifestyle. This discipline is based on positive values such as respect, education, equality, social integration and support among its members. Without political, religious or economic contexts, street workout is for everyone, regardless of age, sex, ethnicity or physical level.

This application aims to cover the needs of many calisthenics with the purpose of keeping track of their activities and exercises in order to progress and monitor their evolution, as well as the incorporation of new users who want to improve their physical condition and start in this sport, which is booming now.

This project is an entrepreneurship TFG carried out in the area of StartInf with the collaboration of the student Víctor Cardona Lorenzo which will focus on the development of the front-end of the application "AllThenics, mobile application for calisthenics: development of the front-end", while this work will focus on the implementation of the architecture and persistence of AllThenics.

Keywords: application; calisthenics; mobile; entrepreneurship; training; back-end; persistence; architecture;

Índice

Anexos

Índice de ilustraciones

Índice de tablas

Índice de gráficas

Índice de diagramas

Índice de fragmentos de código

Capítulo 1

Introducción

1.1. Motivación

El aumento de la actividad física diaria y del cuidado personal en el mundo actual es un hecho que cada año se va asentando más. Según un estudio de The Lancet [1], un 72,5% de la población mundial mantiene una actividad física constante, es decir, que son personas activas realizando deporte contabilizando tanto aquellas que pertenecen a un club o gimnasio como a las que prefieren practicarlo en casa o en algún parque por su cuenta.

A pesar de que parece un dato relativamente agradable para la salud mundial, no hay que olvidar que más de un cuarto de la población sufre sedentarismo y que actualmente se ha visto incrementado ligeramente. Por todo ello, la Organización Mundial de la Salud (OMS) ha establecido en sus planes, plantar cara al sedentarismo y se ha propuesto reducir en un 15% la tasa de sedentarismo entre adultos y adolescentes para el 2030 [2].

Aunque en términos generales el sedentarismo ha ganado terreno, el deporte al aire libre se encuentra en alza. El 40% de los europeos (y el 53% de españoles) [3] prefiere espacios al aire libre o parques, mientras que el 32% practica deporte en casa. De todo ellos, el 77% define su práctica deportiva como espontánea sin pertenecer a ninguna entidad o club deportivo. En este contexto es donde se encuentra la calistenia, cada día más conocida y practicada.

Calistenia viene del griego «*kallos*» (belleza) y «*sthenos*» (fortaleza) y se puede resumir con la frase «hacer ejercicio con tu propio cuerpo». Los recursos de los cuales se nutre la calistenia han sido usados desde nuestros antepasados más lejanos, sin embargo, no se realizaba con el fin de mantenerse en forma o conseguir un físico estético, sino por supervivencia del propio individuo, donde más tarde se preparaba para

la guerra, y que evolucionó adaptándolo a artes marciales o para poder efectuar de manera efectiva labores del día a día. Sin embargo, el origen de la calistenia o s*treet workout*, ya que es conocido por ambos nombres, se remonta a finales de los ochenta y principios de los noventa con partida de nacimiento en el Brooklyn de Nueva York [4]. Fue en los parques de este barrio neoyorquino donde un grupo de amigos con un pasado delictivo se reunía con el fin de ponerse en forma e intentar olvidar su pasado personal además de ayudar a otros vecinos del barrio a no cometer esos errores.

Con esa premisa en mente nacía la calistenia, una práctica de deporte con el propio peso corporal que se podía efectuar de manera gratuita al aire libre con el mínimo equipamiento pero que además promovió los valores del compañerismo, la vida sana, la disciplina, la igualdad y la tolerancia. El primer equipo de calistenia fue fundado en 2002 con el nombre de Ruff Ryders [5], los cuales también llegaron a ofrecer DVDS sobre el entrenamiento, como «Thug Workout, Fitness from the Street». En la Ilustración 1 puede apreciarse a un miembro de este equipo entrenando en lo que parece ser un campo sin un parque acondicionado para esta disciplina. Sin embargo, no fue hasta 2008 con la entrada de YouTube en internet cuando se difundió este deporte. Además, se hicieron populares varios vídeos de un individuo que se convertiría en el máximo referente de la calistenia y el *street workout*, Hannibal Lanham [6] más conocido en el entorno de la calistenia como Hannibal For King, el cual motivó a centenares de personas para salir a las calles a practicar esta disciplina. La Ilustración 2 es una de las más representativas de la calistenia, pues se trata de Hannibal For King, entrenando en un parque en mejores condiciones que en los orígenes y demostrando como se podía alcanzar un físico envidiable únicamente con el propio peso corporal.

Ilustración 1. Miembro de Ruff Ryders entrenando

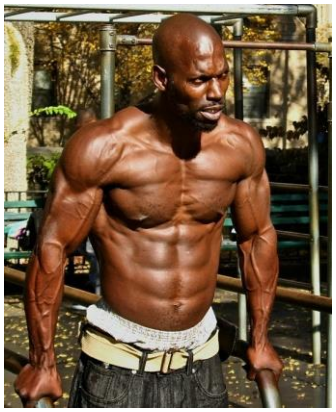
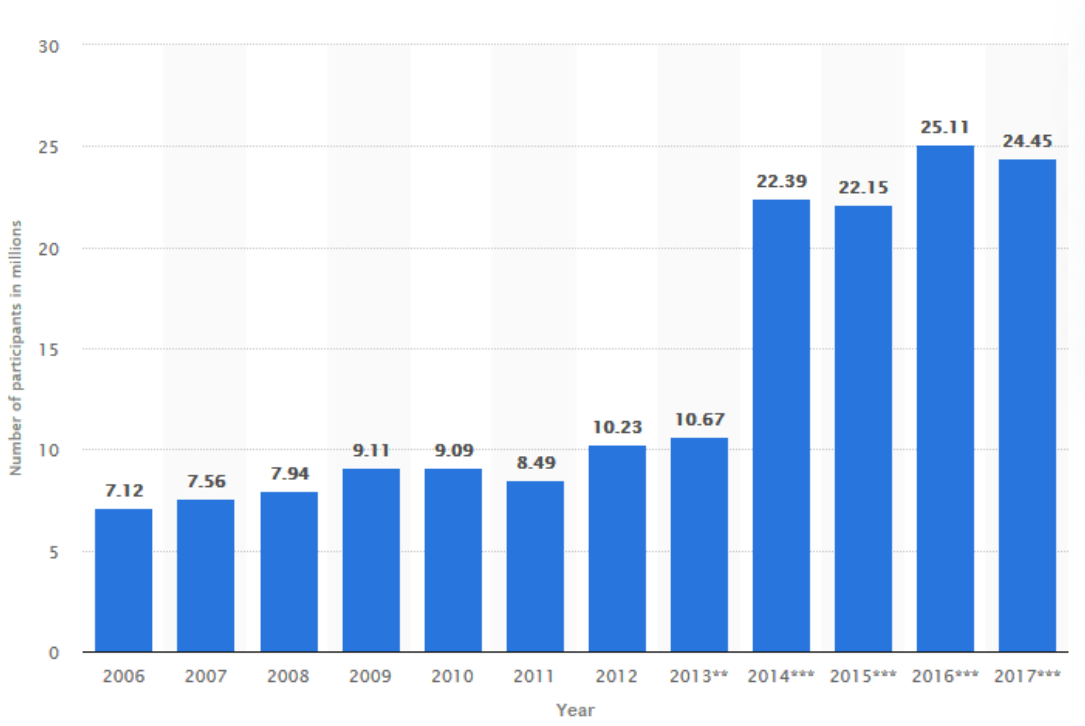


Ilustración 2. Hannibal For King

Fue en 2011 cuando se declaró deporte oficial y se produjo el primer campeonato del mundo en Riga (Letonia) organizado por la World Street Workout and calisthenics Federation (WSWCF[1]) [4]. A partir de entonces y hasta ahora, este deporte ha crecido rápidamente hasta el punto que llegó a convertirse en 2015 en la primera tendencia fitness mundial según el American College of Sports Medicine (ACSM) [5]. A día de hoy cuenta con 122 equipos registrados de 79 países distintos en la federación mundial y más de 30 clubes en la Federación Española de Street Workout y Calistenia (FESWC[2]) [7] [8]. Los últimos datos que se encuentran sobre el porcentaje de población que practica esta disciplina son los que se muestran en la Gráfica 1, la cual se centra en la población de Estados Unidos hasta el año 2017 [9] . Puede observarse claramente la tendencia positiva de la disciplina y como se la augura un futuro prometedor.



Gráfica 1. Millones de personas que practican calistenia en EEUU

En los tiempos que corren se ha producido, por otra parte, una gran revolución en el mundo de las telecomunicaciones mediante los teléfonos inteligentes los cuales permiten realizar infinidad de operaciones gracias a su conexión con internet y la posibilidad de introducir nuevas aplicaciones en él. Actualmente en el mercado de

[1] https://wswcf.org/
[2] https://feswc.org/

aplicaciones de Android se encuentran un número cercano a tres millones de aplicaciones y si realizamos una búsqueda rápida con los términos de calistenia, obtenemos más de 500 resultados, los cuales ofrecen algunas aplicaciones muy completas para facilitar la ejecución de esta disciplina pero que, en algunos casos, la funcionalidad ofrecida resulta limitada [10]. Esta limitación del mercado actual permite conducir a la realización de este proyecto, donde se planteará la realización de un servicio de calistenia para móvil.

Cabe mencionar que se trata de un TFG colaborativo donde se va a realizar el desarrollo completo del mismo, pero centrándose en diferentes ámbitos, por ello, lo que resta de este proyecto será documentado por mi compañero Víctor Cardona Lorenzo con el título de «AllThenics, aplicación móvil para calistenia: desarrollo del *front-end*». Asimismo, indicar que este trabajo se va a desenvolver en el ámbito de StartInf[3], un programa de la Escuela Técnica Superior de Ingenieros Informáticos de la Universidad Politécnica de Valencia para incentivar, asesorar y promover las cualidades de emprendimiento del alumnado.

Por último, comentar el hecho de realizar este proyecto en el ámbito de un TFG de emprendimiento es un desafío más para nosotros, que hemos querido afrontar por el hecho de ser una idea que nos rondaba por la cabeza desde hace un par de años pero que nunca nos habíamos atrevido a llevar a cabo. Este contexto de emprendimiento creemos que va a ser lo más enriquecedor posible y nos ayudará en nuestro futuro, obteniendo mucha experiencia en ámbitos que de otra forma no hubiese sido posible, como por ejemplo, el hecho de realizar un estudio sobre todo el ciclo de vida de una empresa, centrándose no únicamente en el desarrollo del producto y su análisis sino también, en el plan de la empresa, en la evaluación completa de la idea con un presupuesto a medida y teniendo en cuenta a los inversores, así como la puesta en marcha de esta y la adaptabilidad al mercado.

[3] https://startinf.blogs.upv.es/

1.2. Objetivos

El objetivo principal de este trabajo de fin de grado es la creación de un mínimo producto viable (MVP) el cual suponga un punto de partida firme y sólido para poder comercializar y monetizar esta aplicación en un futuro. Para ello se han definido los siguientes objetivos:

- Realizar, al menos, un MVP con su correspondiente evaluación por un conjunto de usuarios.
- Posibilidad de realizar diferentes tipos de rutina.
- Mantener un progreso de todos los entrenamientos realizados por el usuario.
- Acceder al contenido con y sin conexión a internet.
- Conseguir una aceptación de la aplicación de al menos un 70% de los usuarios en el/los experimento/s.
- Obtener un tiempo medio de respuesta de máximo tres segundos.
- Implementar la arquitectura *model-view-viewmodel*.
- Conseguir ofuscar la aplicación mediante R8.

1.3. Estructura de la memoria

Esta memoria se divide en diferentes capítulos haciendo cada uno hincapié en un determinado ámbito.

El segundo punto trata sobre la evaluación de la idea de negocio. En este apartado se desarrollará la idea de la aplicación y su viabilidad, tanto en términos económicos con un modelo de negocio y proyección económica, como a nivel de mercado haciendo un análisis de los competidores. Además, se realiza un análisis DAFO y un Lean Canvas para reforzar nuestra visión de la aplicación y poder abordarla de la mejor forma posible.

En el tercer punto se desarrolla la idea de negocio partiendo de un análisis mediante un mapa de características el cual permite establecer dos MVP que se llevan a cabo y a partir de la implementación de estos, realizar experimentos con *early adopters* para obtener resultados y conclusiones.

A partir cuarto punto hasta el final es donde se diferencian los dos TFG. En este capítulo se especificará el diseño de la base de datos explicando las diferencias entre SQL y NoSQL.

En el quinto punto se tratan todas las tecnologías utilizadas concretamente en la parte desarrollada para este TFG. Además, en algunas de ellas se comparan con herramientas de un corte similar y se explica porque se han utilizado las actuales.

En el sexto punto se explica la arquitectura utilizada en el cliente comparándola con arquitecturas gastadas en muchas otras aplicaciones. También se indica como se ha conseguido implementarla mediante código y se tratan otras implementaciones útiles en la aplicación como la forma utilizada para comunicarse con la base de datos, la autenticación y registro en la aplicación y la manera de enviar notificaciones *push.* Por último, en esta sección se tratan algunos detalles de seguridad como la ofuscación de la aplicación con R8 y las reglas de seguridad utilizadas en Firebase.

En el séptimo punto se discute sobre los resultados obtenidos mediante Crashlytics y Google Analytics en el desarrollo del experimento del segundo MVP y de las pruebas de tiempo de respuesta realizadas.

Por último, los capítulos ocho y nueve sirven para concluir la memoria, resumiendo las conclusiones obtenidas tras el transcurso de todo el trabajo y añadiendo trabajo futuro que se llevará a cabo en la aplicación.

Capítulo 2

Evaluación de la idea de negocio

2.1. Descripción de la idea

Desarrollar un producto software para calistenia, el cual permita facilitar al atleta la gestión de sus rutinas, poder realizar un entrenamiento en tiempo real, establecer diferentes tipos de entrenamiento y disponer de un sistema de gestión de sus entrenamientos dotados de estadísticas de los ejercicios, los grupos musculares trabajados y diferentes parámetros. Esta recopilación de datos permitirá al atleta disponer de una manera sencilla de toda la información necesaria para que pueda modificar su entrenamiento de acuerdo a los resultados que va obteniendo y de esta forma permitirle progresar en su día a día hacia sus objetivos personales.

2.2. Lean Canvas

El Lean Canvas es una adaptación hecha por Ash Maurya y la cual se basa en el Canvas de modelo de negocio [11]. Esta adaptación se centra de manera específica orientada hacia emprendedores que trabajan en su idea de negocio para lanzar su propia *startup*. Se trata de una tabla estructurada en nueve bloques donde su parte derecha refleja el entorno del mercado mientras que la izquierda permite reflexionar sobre el propio producto. A continuación, se muestra el Lean Canvas mediante la Tabla 1.

Tabla 1. Lean Canvas

2 Problema	**4 Solución**	**3 Proposición de valor**	**9 Ventaja competitiva**	**1 Clientes**
En calistenia, a pesar de que puede ser vista como un entrenamiento normal y que recoge todos los aspectos de otras especialidades, no es común que se contrate a un entrenador personal como ocurre en el gimnasio, por ejemplo. Es por ello, que han aparecido las aplicaciones dirigidas a este ámbito, con el fin de proporcionar tanto planes de entrenamiento como ejercicios variados y rutinas con el fin de ayudar a la gente autodidacta. Sin embargo, el mercado actual solamente ofrece un único plan de entrenamiento enfocado al mismo objetivo, sin tener en cuenta que a cada persona puede afectarle de diferente modo e incluso no poder realizarlo.	Una aplicación para el entrenamiento con el propio peso corporal que ofrece: • Gestión de rutinas y ejercicios. • Realizar el entrenamiento en tiempo real con la posibilidad de modificar el número de repeticiones efectuadas y siguiendo el descanso predefinido. • Seguimiento de la evolución del atleta con estadísticas detalladas por ejercicios y grupos musculares. • Varios planes de entrenamiento y rutinas para focalizar en un mismo objetivo. • Diferentes métodos de entrenamiento como piramidal, superseries, EMOM, HIIT y tabata.	¡Empieza a progresar! Entrena donde quieras sin excusas. • Llévate la calistenia contigo y entrena en el parque, en el gimnasio o hasta en tu propia casa. Todos los métodos de entrenamiento a tu alcance. • Mediante los entrenamientos de alta intensidad HIIT, EMOM y tabata conseguirás acelerar tu pulso al máximo y mejorar tu capacidad pulmonar a la vez que reduces la grasa. • Podrás hacer tu rutina convencional con series o en circuito, piramidal para mejorar tu resistencia o si prefieres romper fibras sin descanso, superseries.		Atletas que practiquen cualquier tipo de deporte con el objetivo de mantenerse en forma y realizar un seguimiento de su evolución. En cuanto a la edad se estima entre 16 y 45 años pues es necesario que se sientan cómodos con las nuevas tecnologías. A pesar de que puede estar destinada a cualquier deportista, ya que los ejercicios pueden combinarse con cualquier tipo de disciplina, el público directo al que va dirigido es a los calisténicos, es decir, aquellas personas que practican ejercicio con su propio peso corporal. Respecto a los e*arly adopters* se escogerá un subconjunto de un grupo de atletas calisténicos del grupo Spartans Denia Street Workout.

2 Problema

Sumado a ello, a la hora de realizar un entrenamiento sólo pueden realizarlo, en su mayoría, con un tipo de entrenamiento basado en repeticiones y series sin ninguna otra variedad.

Y en último lugar, no se tiene ningún tipo de gestión sobre el seguimiento de los entrenamientos, y aquellas pocas aplicaciones que lo presentan, lo realizan de una forma mínima y genérica donde un atleta no puede sacar el máximo provecho.

8 Métricas

- Número de clientes que mantienen instalada y usan la aplicación.
- Puntuación de la aplicación en las tiendas.
- Número de suscripciones.

3 Proposición de valor

Seguimiento y estadísticas en detalle.

- Todos tus entrenamientos con cada ejercicio y rutina hecha con infinidad de registros.
- Gráficas, récords personales, grupos musculares trabajados, máximas repeticiones, etc.

5 Canales

- Carteles publicitarios locales en los diversos parques de calistenia.
- Campaña en redes sociales.
- Promoción en la página web de la aplicación.

7 Costos (anuales en el cuarto año)

275.000€ Sueldos de personal (Directores, desarrolladores y otros).	24.500€ Alquiler de oficina y gastos asociados (luz, agua, internet, etc.)
10.000€ Uso de Firebase y derivados	25.000€ Marketing

Total Anual: 334.500€

6 Ingresos (anuales en el cuarto año)

Suscripción por parte de los usuarios para tener acceso a todas las funcionalidades.

Individual. Mensual 2,99€. Trimestral 5,99€/ tres meses. Anual 16,99€/ doce meses.

Se intentará también obtener inversión por parte de concursos de *startups* y de *influencers* famosos en el ámbito de la calistenia que quieran participar en el proyecto.

Objetivo: 30.000 suscripciones anuales → 509.700€ en suscripciones.

Total Anual: 509.700€

2.3. Análisis DAFO

Una de las mejores técnicas para afrontar un desarrollo de una aplicación es realizar un análisis DAFO para saber cómo va a funcionar la empresa al adentrarse en un nuevo terreno teniendo en cuenta las características internas (Debilidades y Fortalezas) y la situación externa (Amenazas y Oportunidades).

Tabla 2. Análisis DAFO

Debilidades	**Fortalezas**
• Inexperiencia en el mercado de las aplicaciones móviles. • Personal limitado. • Tecnologías nuevas con las que nunca se ha trabajado. • Recursos económicos limitados. • Contenido reducido en cuanto a rutinas y planes de entrenamiento.	• Ímpetu y ganas de emprender. • Conocimiento de los usuarios potenciales del producto. • Conocimiento extenso del deporte que cubre la aplicación. • Gran adaptabilidad a entornos cambiantes. • Facilidad de uso. • No precisa de conexión a internet para su funcionamiento.
Amenazas	**Oportunidades**
• Aplicaciones con éxito ya establecidas en el mercado. • Personas influyentes que recomiendan otras aplicaciones. • Cambios en los hábitos de las personas. • Posibilidad de copia de las nuevas funcionalidades por los competidores.	• La gente está comenzando a ser más saludable. • Mercado calisténico en crecimiento. • Precio competitivo. • Mejorar las aplicaciones actuales al conocer sus debilidades. • Carencia de servicios con un seguimiento detallado y fructífero del atleta. • Calisténicos que buscan nuevos métodos de entrenamientos alejados de los clásicos.

Con los datos obtenidos en el análisis DAFO se deberán buscar acciones para potenciar las fortalezas y oportunidades y estrategias para poder hacer frente a las debilidades y amenazas. Lo más relevante obtenido en este análisis es que se va a tener que controlar muy bien a los competidores una vez la aplicación haya salido al

mercado para poder reaccionar, mantener una sólida base de usuarios y atraer a nuevos mediante la introducción de nuevo contenido de manera regular, como añadir más planes de entrenamiento para todo tipo de usuarios.

2.4. Estudio de Mercado

Para poder estimar la viabilidad del proyecto resulta necesario realizar un estudio sobre la situación del mercado actual. El siguiente apartado es aquel que atiende el análisis de aquellas aplicaciones que se encuentran actualmente en el mercado y que pudieran llegar a ser competidoras con ésta, ya sea por el usuario final al que va enfocado, por la gran popularidad entre la comunidad calisténica o por el tipo de funcionalidades que ofrece y resulta necesario conocer.

2.4.1. Calisteniapp

Ilustración 3. Calisteniapp, logo

Calisteniapp[4], que tiene la Ilustración 3 como logo, es, sin ningún tipo de duda, la principal aplicación a batir. Del desarrollo, mantenimiento y evolución de ésta forma parte, junto con otro compañero, Yerai Alonso, más conocido en las redes sociales como Yerai Street Workout[5] el cual se dio a descubrir por YouTube subiendo contenido de calistenia como ejercicios, rutinas y tutoriales de diferentes trucos llegando a posicionarse como uno de los canales más influyentes de habla hispana.

Por lo que a Calisteniapp se refiere fue lanzada en 2016, en una época donde ya eran populares las aplicaciones móviles, pero en el ámbito de la calistenia aún no había llegado a ser explotado, por ello la gran aceptación que tuvo, logrando a día de hoy cientos de miles de usuarios que hacen uso de ella diariamente.

La funcionalidad más destacable es la de rutinas inteligentes, las cuales dependiendo de si llegas a finalizar una serie de ejercicios o no, el siguiente día que afrontes esta rutina ese número de series o repeticiones se verá modificado, ya sea rebajándolo porque aún no estás preparado para el siguiente paso o subiéndolo y

[4] https://calisteniapp.com/es
[5] https://www.youtube.com/channel/UC4ODKpmM3Z309MSL7koj9Jg

permitiéndote continuar progresando. En la Ilustración 4 se muestra la rutina inteligente para la realización de la plancha donde se han añadido nuevos ejercicios diferentes.

Ilustración 4. Calisteniapp, rutinas inteligentes

También presenta la posibilidad de visualizar los ejercicios, creación y recomendación de rutinas, planes enfocados a distintos objetivos y artículos además de presentar una interfaz simple e intuitiva. Son los planes de entrenamiento los que también hacen que esta aplicación se posicione en los primeros puestos para la gente, pues están diseñados de forma que se obtienen muy buenos resultados.

Una de las últimas actualizaciones, implementó la posibilidad de hacer desafíos de 21 días, donde cada uno era una rutina distinta pero enfocada a un mismo fin, con el objetivo de juntar a la comunidad y crear la sensación de que todo el mundo estaba entrenando conjuntamente. La Ilustración 5 es un ejemplo de un desafío, donde consta una breve descripción, el número de personas que lo realizan y un enlace a una diferente rutina por cada día.

Ilustración 5. Calisteniapp, desafío de 21 días

Respecto al método de pago, se realiza mediante suscripción pudiendo elegir entre mensual o anual, con un precio de 3.99€ o 19.99€ respectivamente.

2.4.2. FitNotes

Ilustración 6. FitNotes, logo

Lo primero a comentar sobre esta aplicación es que no trabaja con ejercicios del propio peso corporal, sino que está enfocada a ejercicios de gimnasio con mancuernas y pesas. A pesar de ello y como se ha comentado anteriormente, resulta de interés su análisis por la funcionalidad que ofrece. El logo de FitNotes[6] es el mostrado en la Ilustración 6.

Esta aplicación no nos ofrece ningún tipo de planes de entrenamiento, o la posibilidad de realizar la rutina mientras entrenamos o nada por el estilo. A pesar de ello, la única finalidad de la cual dispone es la de poder llevar un seguimiento de los ejercicios que realizamos a lo largo del tiempo, guardando el tipo de ejercicio, número

[6] http://www.fitnotesapp.com/

de series y repeticiones y la visualización de estos datos en un formato más afable mediante gráficas, como se observa en la Ilustración 7.

Además de que podemos guardar cualquier registro, aunque sea un día anterior gracias a la selección con el calendario que dispone. Pero no solo permite realizar un seguimiento del entrenamiento realizado, sino que también puede realizarlo con las medidas del cuerpo y el peso, de este modo se pueden tomar el tamaño de los músculos e ir viendo su evolución. Relacionado con estas funcionalidades también se encuentra la posibilidad de establecer objetivos ya sea en cuanto a peso, medidas corporales o específicas por algún tipo de ejercicio.

Ilustración 7. FitNotes, estadísticas y gráfica

En cuanto al apartado de ejercicios, no se encuentran muchos sobre calistenia, únicamente los más básicos y generales. No obstante, permite la creación de ejercicios propios por parte del usuario, además de poder visualizar los mayores logros obtenidos en cada ejercicio, como mayor lastre añadido o mayor número de repeticiones.

Por último, mencionar que se trata de una aplicación completamente gratuita y aunque tiene buena acogida entre los usuarios, estos reclaman un rediseño de la interfaz ya que no resulta evidente su funcionamiento.

2.4.3. JeFit

Ilustración 8. JeFit, logo

JeFit[7], con logo en la Ilustración 8, es una de las mejores aplicaciones tanto para entrenar como para realizar un seguimiento de tu progreso centrado en la disciplina de gimnasio. Puede verse como la unificación de las dos aplicaciones anteriores en una sola, ya que ofrece una gran variedad y efectividad de planes de entrenamiento sumado a la posibilidad de registrar los resultados.

Esta aplicación nos ofrece tanto la creación de rutinas como la de ejercicios personalizados, además de rutinas predefinidas y planes de entrenamiento completos enfocados a distintos objetivos. Asimismo, también dispone de un sistema de seguimiento con tal de poder monitorizar los resultados tanto de ejercicios, rutinas, peso, y medidas corporales. Estos datos son presentados mediante gráficas y también permite ver estadísticas afines al rendimiento, en la Ilustración 9 se muestra un ejemplo de cómo se mostrarían estas gráficas.

Ilustración 9. JeFit, estadísticas

Gracias a la comunidad que lo forma, la aplicación ha heredado funciones de una red social donde se permite seguir a otros usuarios, compartir y comentar las rutinas

[7] https://www.jefit.com/

que hayan ejecutado y comunicarse entre ellos gracias al sistema de mensajería interna del cual dispone. Como características reseñables y que hace a esta aplicación única, destaca la integración con relojes inteligentes donde puede realizarse la rutina y establecer el número de repeticiones que ha llevado a cabo en el ejercicio. Asimismo, no se trata de una aplicación móvil únicamente, sino que también tiene una aplicación web dedicada con las mismas funcionalidades. La Ilustración 10 muestra de manera global como se visualizan los datos en los diferentes dispositivos compatibles.

Ilustración 10. JeFit, página web y reloj inteligente

A consecuencia de lo descrito, no resulta descabellado que más de ocho millones de personas hagan uso de estos servicios mediante cualquier método que se les ofrece. El método de monetización del que dispone es mediante suscripción ofreciendo diferentes alternativas, como mensual o anual, con un precio de 6.99€ y 39.99€ respectivamente. Ello conlleva, además de eliminar anuncios y acceder a contenido exclusivo como rutinas, por ejemplo, se añade la funcionalidad de poder realizar un ejercicio alternativo mientras se está entrenando, una característica útil y que no todos la ofrecen, pues resulta normal en un gimnasio tener que abandonar un ejercicio porque no se dispone de la máquina apropiada.

2.4.4. Madbarz, logo

Ilustración 11. Madbarz, logo

Retomando las aplicaciones orientadas únicamente hacia la calistenia, se encuentra Madbarz[8], una aplicación con más de un millón de usuarios conectados a ella que ofrece una parte gratuita, pero para acceder a la mayoría de sus funcionalidades se debe pagar una suscripción, ya sea de 9.99$ mensual o 59.99$ anualmente, respectivamente. Su logo corresponde con la Ilustración 11.

La oferta de Madbarz no destaca por la personalización de ejercicios, o la representación de estadísticas obtenidas mediante entrenamientos. Esta aplicación se ha focalizado en ofrecerle al usuario un alto contenido de artículos mediante libros electrónicos o simples reseñas ya sean de cómo estructurar un entrenamiento, consejos a tener en cuenta, apartado nutricional o hasta recetas de cocina óptimas para el post entreno como se aprecia en la Ilustración 12.

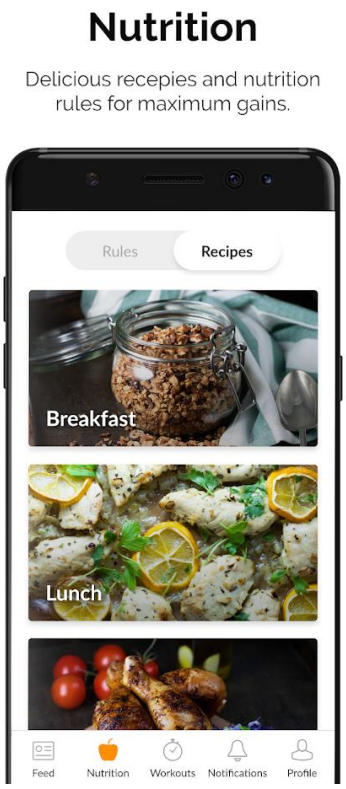


Ilustración 12. Madbarz, artículos de nutrición

A pesar de ello, esto no quiere decir que sus funcionalidades a la hora de ejecutar un entrenamiento sean escasas o pésimas. Ofrece lo básico para ser considerada una

[8] https://www.madbarz.com/

aplicación de deporte con su creación de rutinas propias, el poder ejecutarlas mientras se entrena, poder visualizar los entrenamientos anteriores y, aunque no de manera detallada, incluye una gráfica con la curva obtenida de acuerdo a los ejercicios realizados.

2.4.5. Pro-Gress

Ilustración 13. Pro-Gress, logo

En los análisis anteriores se había comentado que ciertas aplicaciones presentaban una componente social, en cambio, ninguna de ellas puede compararse con Pro-Gress[9], la cual es una aplicación pensada para realizar un entrenamiento básico, en cuanto a funcionalidades se refiere, pero donde se ha invertido mucho para crear una red social para calisténicos. La Ilustración 13 es el logo de la aplicación.

Esta red se compone de la interacción entre usuarios mediante mensajes, publicaciones, sistema de puntuaciones sobre las rutinas realizadas, y también una clasificación global entre todos los usuarios de acuerdo a la persistencia con la que entrenan, la dificultad de los ejercicios y la realización de estos. La Ilustración 14 muestra varias capturas de la aplicación relacionada con este concepto, donde aparece el ranking de mejores usuarios y la posibilidad de buscarlos. Los usuarios, también, pueden seguirse entre ellos, lo cual les permite ver la rutinas o planes de entrenamiento creados o que se encuentran realizando actualmente, es decir, implementa toda la funcionalidad de una red social, pero en el contexto del entrenamiento con el peso corporal.

[9] https://pro-gress.es/

Ilustración 14. Pro-Gress, red social

Añadidas a estas características se encuentra la localización de parques de barras cercanos, los cuales son añadidos por los usuarios y mantenidos por la comunidad. Estos se visualizan en un mapa donde se facilita una breve descripción del material que dispone y alguna que otra foto. Un ejemplo del mapa con nuestra localización actual y varios parques cercanos se encuentra en la Ilustración 15. Esta función permite conocer lugares cercanos donde practicar este deporte, explorar nuevos e incluso realizar concentraciones. El último elemento reseñable es la integración de una tienda online, donde ofrece equipamiento variado específico con la posibilidad de comprarlo directamente.

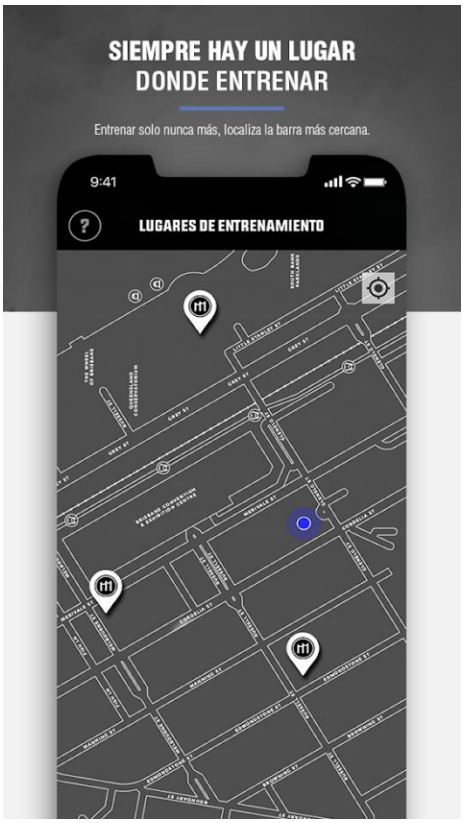


Ilustración 15. Pro-Gress, lugares de entrenamiento

Esta aplicación ha sido desarrollada en parte por Joel Alonso[10], otro creador de contenido de la plataforma YouTube especializado en calistenia. Si bien, no goza de tanta popularidad como su compañero Yeray, mencionado previamente en el apartado 2.4.1, es un gran atleta y la efectividad de sus planes de entrenamiento lo corrobora. El desembolso que debe efectuarse para obtener la totalidad de esta aplicación se realiza de manera mensual o anual, con un precio de 4.99€ o 42.99€, respectivamente.

2.4.6. Strong

Ilustración 16. Strong, logo

Para finalizar este análisis se encuentra Strong[11], cuyo logo corresponde con la Ilustración 16, una gran aplicación orientada al gimnasio pero con magníficas funcionalidades. Por una parte, tenemos lo básico de una aplicación de deporte en este terreno, como es la creación de rutinas, los planes de entrenamiento preestablecidos y la posibilidad de visualizar los ejercicios, con una descripción de cómo realizarlo y los músculos implicados en su ejecución. Además de todo ello, también ofrece una monitorización de cada ejercicio con los registros de los entrenamientos efectuados, los

[10] https://www.youtube.com/channel/UCt5m170NSW-FNAjD9pbtBwQ/featured
[11] https://www.strong.app/

logros alcanzados y diversas gráficas y estadísticas para ayudar a la toma de decisiones para afrontar un seguimiento continuo, controlar el peso y medidas corporales. En la Ilustración 17 tenemos por una parte, la captura de la izquierda con una calculadora de pesos para saber cuál sería el peso óptimo para trabajar y, por otra parte, la captura de la derecha donde se muestran diferentes gráficas sobre el entrenamiento.

Como funcionalidades destacan la posibilidad de modificar y elegir los ejercicios para realizar un calentamiento antes de la rutina, el cual es un punto que no se había encontrado hasta ahora y puede resultar muy interesante para cualquier deportista con el fin de evitar lesiones y empezar a poner en funcionamiento el músculo. Por otro lado, la personalización a la hora de definir una rutina ya que existen diferentes métodos de llevar a cabo un entrenamiento y la mayoría de aplicaciones únicamente incluyen el básico de series. En cambio, Strong nos ofrece la posibilidad de realizar un entrenamiento con superseries, es decir, haciendo varios ejercicios sin descanso entre ellos.

Ilustración 17. Strong, calculadora y estadísticas

Además, también incluye varias calculadoras, una de ellas indica con cuantas pesas resulta conveniente calentar y la otra, calcula el índice de grasa corporal de acuerdo a la estatura y el peso e incluye la posibilidad de sincronizarse con Google Fit o Apple Health.

En último lugar, mencionar que no todas las características son gratuitas pero que ofrece dos métodos de pago distinto, uno por suscripción ya sea mensual o anual con un precio de 5.49€ o 32.99€ respectivamente, o un único pago de 104.99€.

2.4.7. Tabla comparativa

A continuación, se muestran unas tablas, Tabla 3 y Tabla 4, comparando las principales funcionalidades de las seis aplicaciones analizadas anteriormente. Mediante el siguiente método se tiene de una manera fácil y muy visual establecer rápidamente lo que ofrecen o no las principales competidoras. Esta aplicación también se incluye en la tabla comparativa en último lugar y resaltada para poder diferenciarla con objeto de tenerse como referencia respecto a sus competidoras, donde se ha hecho una selección de las características básicas con las que debería contar una aplicación en este ámbito así como aquellas que permitirá diferenciarse de las actuales y ofrecer aquellas características y funcionalidades exclusivas y mejoradas para explotar el mercado actual.

Tabla 3. Comparativa competidores (1/2)

	Nombre	Calisteniapp	FitNotes	JEFIT	Madbarz	Pro-Gress	Strong	AllThenics
Disciplina	Calistenia	✓			✓	✓		✓
	Gimnasio		✓	✓			✓	
Rutina	Planificar semana de entrenamiento	✓	✓	✓		✓		✓
	Planes de entrenamiento	✓		✓	✓	✓		✓
	Diferentes planes y rutinas para un mismo objetivo							✓
	Rutinas por defecto	✓		✓	✓	✓	✓	✓
	Rutinas adaptativas progresivamente	✓						
	Recomendación de rutina por músculos	✓			✓			✓
	Rutinas nuevas diarias							
	Crear rutinas	✓	✓	✓	✓	✓	✓	✓
	Añadir series de calentamiento a la rutina	✓					✓	
	Marcar rutina favorita	✓			✓	✓		✓
	Compartir rutina	✓	✓		✓	✓	✓	✓
Entrenamiento	Visualizar historial de entrenamientos	✓	✓	✓	✓		✓	✓
	Registrar entrenamientos anteriores		✓	✓				✓
	Visualizar estadísticas de la rutina		✓	✓	✓			✓
	Estadísticas de músculos entrenados			✓				✓
	Copiar entrenamientos		✓					
	Visualizar entrenamientos de otros usuarios	✓		✓	✓	✓		
	Seguimiento de peso y medidas del cuerpo		✓	✓			✓	
	Establecer objetivo de peso o medidas del cuerpo		✓	✓				
	Programar recordatorio de entrenamiento			✓				✓
	Entrenamiento con superseries		✓	✓			✓	✓
	Entrenamiento piramidal							✓
	Entrenamiento circuito	✓	✓	✓	✓	✓	✓	✓
	Entrenamiento HIIT	✓						✓
	Entrenamiento TABATA	✓						✓
	Entrenamiento EMOM	✓						✓

Tabla 4. Comparativa competidores (2/2)

	Nombre							
Ejercicio	Visualización de ejercicios	✓	✓	✓			✓	✓
	Crear ejercicios		✓	✓			✓	
	Marcar ejercicio favorito	✓		✓				✓
	Obtener por ejercicio récords, objetivos, gráficas e historial		✓				✓	✓
Varios	Artículos	✓		✓	✓			
	Recetas y guías de nutrición				✓			
	Añadir lugares de entrenamiento			✓		✓		
	Comprar material específico					✓		
	Integración con Smartwatch			✓			✓	
Social	Seguir a otros usuarios			✓	✓	✓		
	Puntuar y comentar entrenamientos de otros usuarios			✓	✓	✓		
	Ránking de usuarios				✓	✓		
Pago	Aplicación gratuita	No todo	✓	No todo	No todo	No todo	No todo	No todo
	Método de pago (suscripción/pago único)	Suscripción	n/a	Suscripción	Suscripción	Suscripción	Ambos	Suscripción
	Precio mensual	3,99 €	n/a	6,99 €	9,99 €	4,99 €	5,49 €	2,99 €
	Precio anual	19,99 €	n/a	39,99 €	59,99 €	42,99 €	32,99 €	16,99 €
	Precio pago único	n/a	n/a	n/a	n/a	n/a	104,99 €	n/a

2.4.8. Clientes potenciales

El cliente al que va dirigida esta aplicación es una persona de una edad comprendida entre 16 a 45 años, aunque el límite superior se encuentra muy condicionado por el manejo de las nuevas tecnologías pues, podría llegar a usarse por personas de una edad más avanzada ya que la forma física y la voluntad por mantenerse en forma es una cualidad muy subjetiva de cada individuo y que con la práctica de la calistenia está asegurada.

Continuando con la descripción del cliente, principalmente se centra en un púbico activo físicamente, al cual le guste y practique deporte con regularidad particularizando en la calistenia como disciplina o, incluso, para aquella persona que nunca ha sido deportista y quiere empezar a ponerse en forma.

2.5. Modelo de negocio y proyección económica

El modelo de negocio que se va a implementar se encuentra basado en suscripciones con ligeros descuentos a mayor tiempo de suscripción. Por lo tanto, habrá una parte gratuita en la aplicación que tendrá una funcionalidad base y una parte *premium* que se obtendrá al suscribirse. Se ha decidido este modelo de negocio porque en el mercado de las aplicaciones de calistenia este es el predominante y el que mejor resultados puede ofrecer.

En cuanto a la proyección económica, como se puede observar en la Tabla 5, se ha puesto en un escenario ideal donde el primer y segundo año no se goza de una gran popularidad, se tienen pérdidas y resulta necesaria una inversión inicial de unos cincuenta mil euros por parte de inversores o de posibles concursos de *startups*.

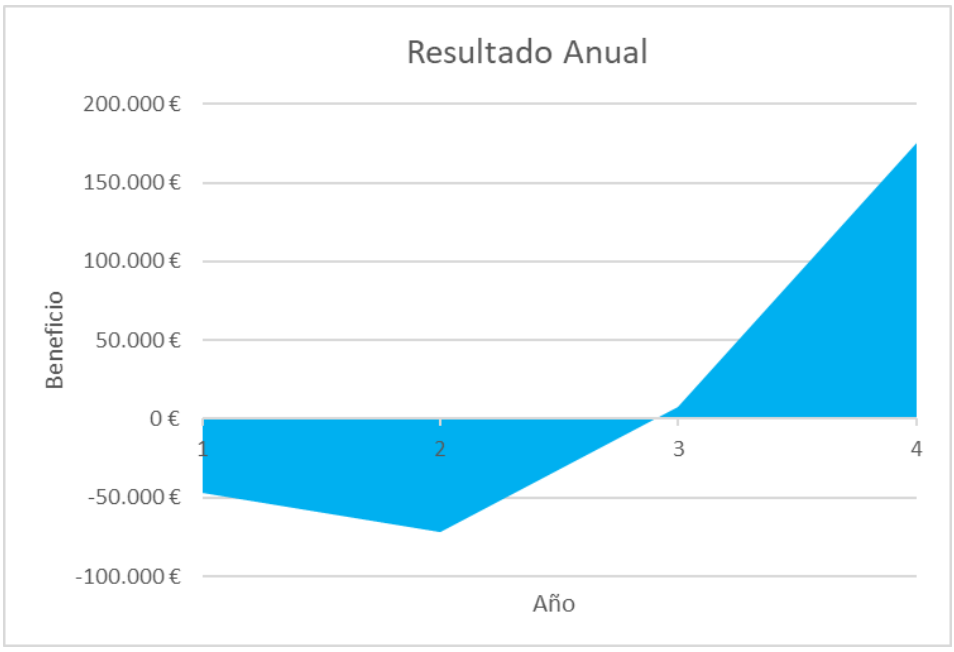
A partir del tercer año, se amplía bastante el capital ya que es cuando se pretende obtener una fiel base de usuarios en España y dar el salto al escenario internacional. Es a partir de este año en el cuál se obtienen unos pocos beneficios y se fuerza la contratación de un director financiero y de un administrador.

Por último, es en el cuarto año donde realmente se percibe la popularidad obtenida y, por tanto, se amplía la cantidad de desarrolladores y de técnicos de soporte para poder hacer frente a la demanda de usuarios.

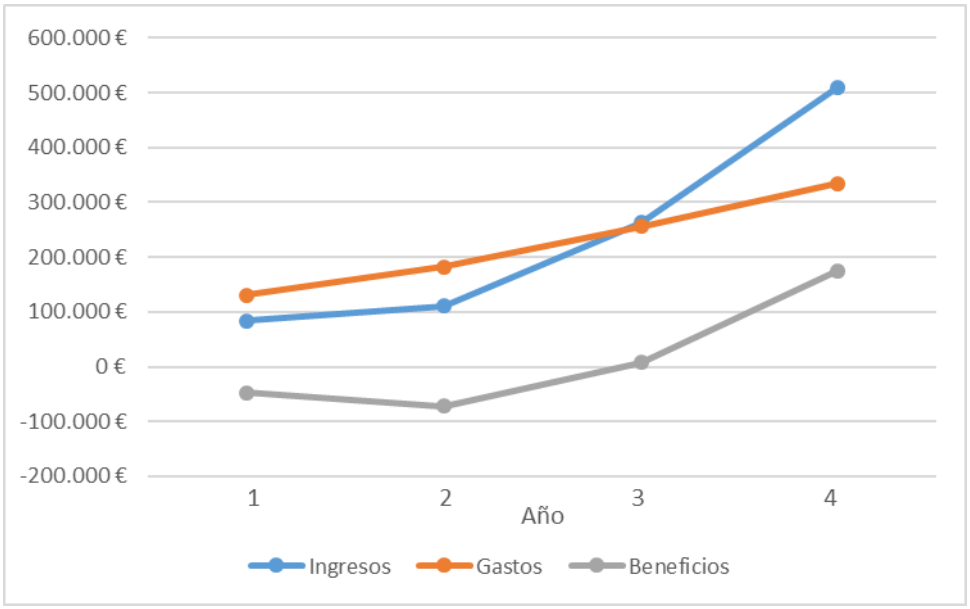
Tabla 5. Proyección económica

Precio de suscripcion anual	16,99 €			
Cantidad de suscripciones	**Año 1**	**Año 2**	**Año 3**	**Año 4**
Suscripciones anuales	2000	6500	15500	30000
Ingresos Anuales				
x Suscripciones	33.980 €	110.435 €	263.345 €	509.700 €
Inversores/concursos	50.000 €	0 €	0 €	0 €
Total Ingresos	83.980 €	110.435 €	263.345 €	509.700 €
Gastos Anuales				
Infraestructura Cloud	2.000 €	3.000 €	5.000 €	10.000 €
Ordenadores e impresoras	4.000 €	2.000 €	5.000 €	3.000 €
Muebles oficina e instalaciones	3.000 €	2.000 €	2.000 €	2.000 €
Marketing	5.000 €	10.000 €	20.000 €	25.000 €
Alquiler oficina	12.000 €	12.000 €	12.000 €	12.000 €
Internet, electricidad, agua, teléfono, etc.	3.000 €	3.500 €	4.000 €	4.500 €
Gestoría	2.000 €	2.000 €	3.000 €	3.000 €
CEO - Director Ejecutivo	30.000 €	30.000 €	35.000 €	40.000 €
CTO - Director Técnico	25.000 €	25.000 €	30.000 €	35.000 €
CMO - Director de Marketing	0 €	22.500 €	25.000 €	30.000 €
CFO - Director Financiero	0 €	0 €	25.000 €	30.000 €
Desarrolladores	45.000 €	50.000 €	50.000 €	75.000 €
Técnicos de soporte	0 €	20.000 €	20.000 €	40.000 €
Administrativos	0 €	0 €	20.000 €	25.000 €
Total Gastos	131.000 €	182.000 €	256.000 €	334.500 €
Resultado Anual	-47.020 €	-71.565 €	7.345 €	175.200 €
Resultado Anual Acumulado	-47.020 €	-118.585 €	-111.240 €	63.960 €
	Personal Año 1	**Personal Año 2**	**Personal Año 3**	**Personal Año 4**
CEO - Director Ejecutivo	1	1	1	1
CTO - Director Técnico	1	1	1	1
CMO - Director de Marketing	0	1	1	1
CFO - Director Financiero	0	0	1	1
Desarrolladores	2	2	2	3
Técnicos de soporte	0	1	1	2
Administrativos	0	0	1	1
Total empleados	4	6	8	10

En las siguientes gráficas se puede ver por una parte el resultado anual en cuanto a beneficios en la Gráfica 2, encontrando el punto de inflexión en el tercer año y por otra, en la Gráfica 3, los beneficios, gastos e ingresos obtenidos en los cuatro años de duración de la empresa. En resumen, se observa que, a mayor tiempo en el mercado, se obtienen más beneficios y gracias a ello permite una mayor inversión la aplicación.



Gráfica 2. Beneficio anual



Gráfica 3. Beneficios, gastos e ingresos anuales

2.6. Conclusiones de la evaluación

En cuanto a toda la evaluación realizada se pueden obtener unas conclusiones bastante claras.

Lo primero y más importante es que el proyecto se va a poder llevar adelante y, si conseguimos cumplir el escenario ideal, sacar beneficio con la aplicación a largo plazo. El punto de inflexión, como se ha podido observar en el apartado 2.5, se encuentra en el cuarto año donde ya se puede devolver la inversión de capital que se recibió por parte de los inversores. Pese a tener esta estimación en mente, no debe olvidarse de que se trata de una estimación y que no puede asegurarse el cumplimiento de este escenario. Por ello, para que la empresa pueda seguir adelante es bastante probable que se necesiten varias aplicaciones en diferentes ámbitos que sumando los beneficios de varias de ellas la empresa consiga mantenerse correctamente a flote.

La siguiente conclusión obtenida es la posibilidad de explotación del mercado de las aplicaciones de calistenia, el cual tiene un gran margen de mejora. Aunque ya existen aplicaciones establecidas y que gozan de gran popularidad, esta aplicación puede implantarse y posicionarse como la próxima gran aplicación favorita de los calisténicos. Con las funcionalidades como la monitorización del entrenamiento realizado a lo largo de los meses y ofreciendo un precio competitivo de salida puede alcanzarse una buena base de usuarios a lo largo de los años.

Relacionado con lo comentado anteriormente se obtiene la última conclusión, la cual hace hincapié sobre la necesidad de hacer partícipes de este proyecto a los usuarios durante todo el ciclo de vida. Debido a ello se hará, al menos, un MVP con una selección de usuarios para obtener la retroalimentación necesaria para mejorar la aplicación antes del lanzamiento oficial e incluso posteriormente estaremos en contacto constante de cualquier sugerencia y/o mejora que los usuarios puedan realizar.

Capítulo 3

Desarrollo de la idea de negocio

Para el desarrollo del producto se ha seguido la metodología de trabajo Lean Startup, dado que esta empresa posee cierto nivel de incertidumbre al ser de nueva creación y de no tener asegurado la cantidad de usuarios prevista esta metodología resulta idónea para tal fin.

La metodología Lean Startup [12] se define como: «Un conjunto de prácticas pensadas para ayudar a los emprendedores a incrementar las probabilidades de crear una *startup* con éxito. No es una fórmula matemática infalible, sino una filosofía empresarial innovadora que ayuda a los emprendedores a escapar de las trampas del pensamiento empresarial tradicional» [13]. Su funcionamiento se encuentra basado en un aprendizaje validado, donde se obtienen opiniones del publico destinatario a lo largo de diferentes fases con el fin de adaptar el proyecto sin margen de error. Sin embargo, este método no garantiza el éxito, sino que logra que el riesgo sea menor probando pequeñas funciones en lugar del producto final al completo. Los prototipos que reciben los clientes para validar estas funciones reciben el nombre producto mínimo viable (MVP) y será mediante la aprobación de los usuarios donde deberán modificarse tareas, incluir nuevas o eliminarlas.

Para llevar un control de las tareas a realizar y mantener un seguimiento continuo se va a utilizar el gestor de proyectos Worki Process, explicado detalladamente en el apartado 5.5. En este gestor se han introducido dos *sprints* de cuatro semanas entre los que se va a dividir el trabajo a realizar correspondiéndose con los dos MVP.

3.1. Mapa de Características

A partir del apartado 2.4 donde se ha analizado los diferentes competidores del mercado actual y se han extraído el conjunto de características que se ofrecen en AllThenics, se ha creado el mapa de características del Diagrama 1 y Diagrama 2. Se entiende como mapa de características a un listado con las tareas a realizar ordenadas por prioridad, así como el tiempo estimado para llevarlas a cabo.

Este mapa, el cual se ha divido en los diferentes módulos por los cuales puede desarrollarse la aplicación, nos facilitará la tarea de elegir aquellas funciones para implementar el primer MVP de acuerdo a la prioridad de estas. Cabe aclarar que existen funciones que se encuentran entre dos módulos, por ejemplo, Añadir rutina a planificación de entrenamiento semanal, lo cual no resulta descabellado pues lo más normal en una aplicación es que las diferentes partes de esta se comuniquen e interactúen entre ellas y que no se trate simplemente de diferentes aplicaciones envueltas en una única que no guardan relación entre sí.

Los módulos más destacables en los cuales se han divido todas las funciones de la aplicación han sido los siguientes:

- Rutinas: Gestión de las rutinas, desde su creación, modificación o eliminación, hasta la visualización de estas. También incluye las características de poder realizar una rutina en tiempo real con el dispositivo, filtrar o buscar y que el sistema devuelva una rutina recomendada de acuerdo a ciertos filtros.
- Ejercicios: Permite visualizar todos los ejercicios, así como sus características y logros con la posibilidad de filtrarlos.
- Progreso: Módulo de principal interés por su contenido exclusivo en el mercado, ya que unifica la realización de las rutinas para la generación y visualización de estadísticas y datos relevantes sobre estas a lo largo del tiempo.
- Planes de entrenamiento: Otro de los módulos interesantes y competidores pues se basa en la ejecución de diferentes rutinas prestablecidas para alcanzar un objetivo fijado.

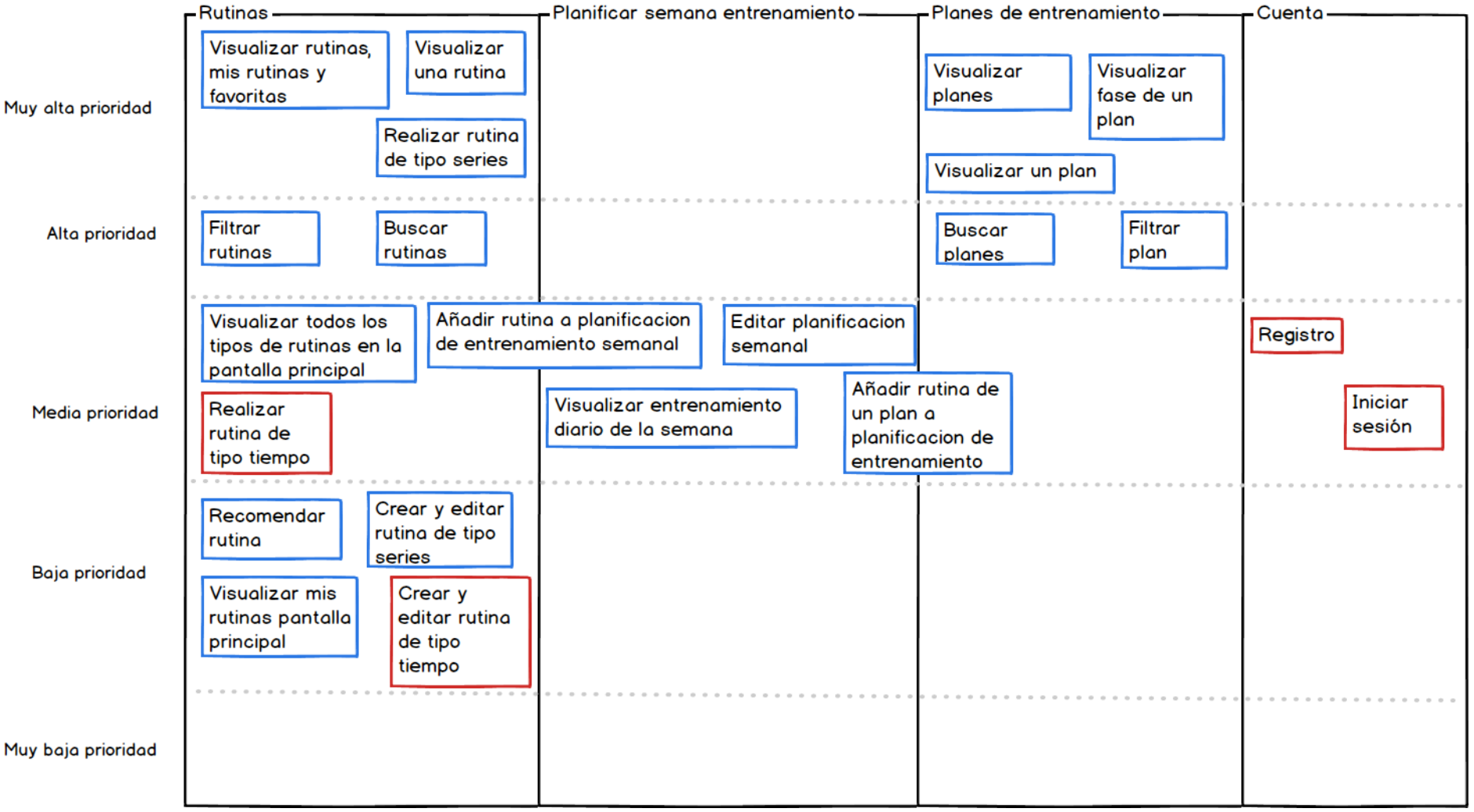


Diagrama 1. Mapa de características (1/2)

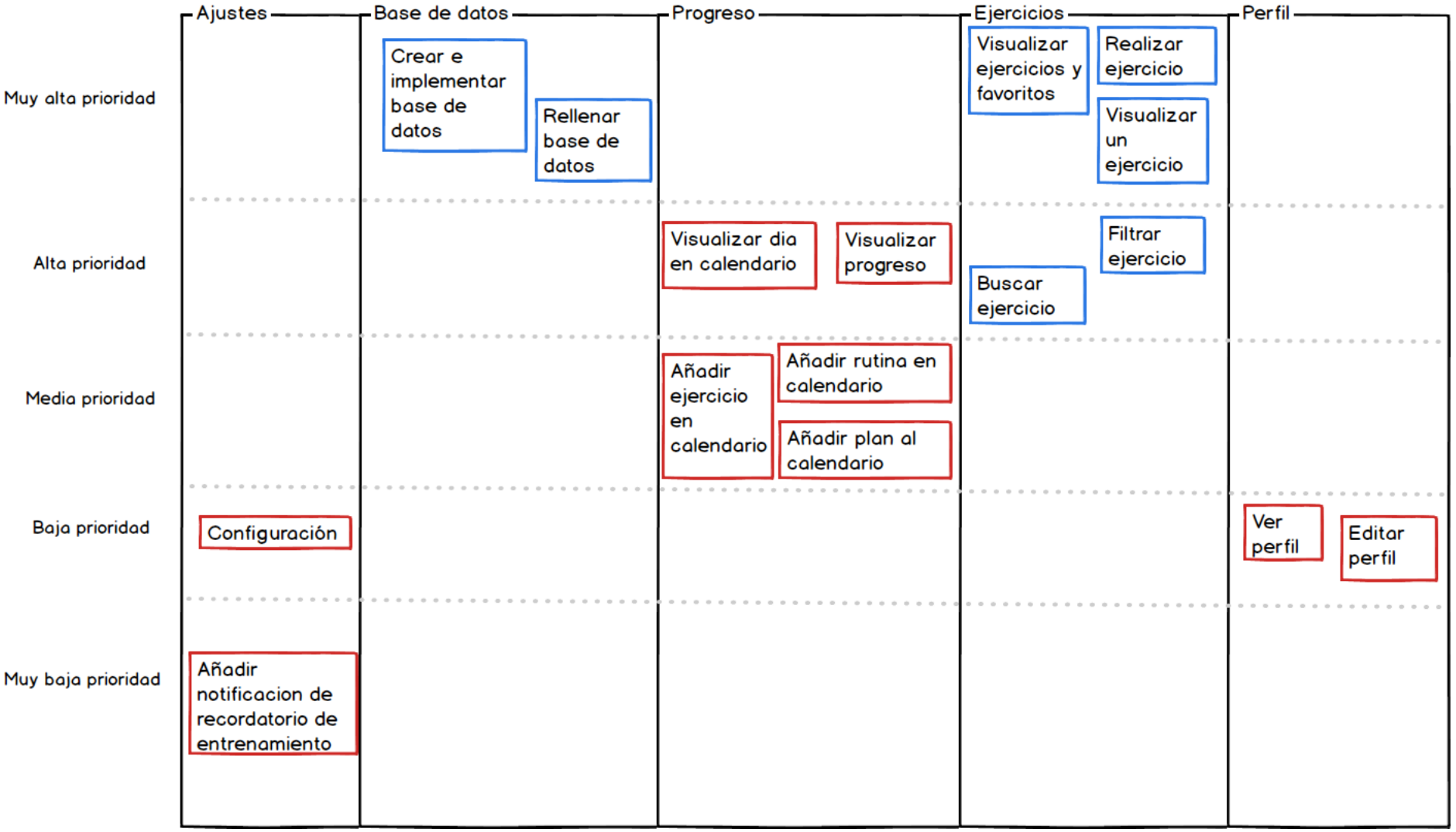


Diagrama 2. Mapa de características (2/2)

3.2. Desarrollo del primer MVP

En este apartado se va a desarrollar todo el primer MVP describiendo el ámbito que abarca y comentado los resultados obtenidos en el experimento con los *early adopters*. La aplicación ha sido subida a Google Play siendo publicada en pruebas internas, consiguiendo establecer una lista de *testers* mediante correos electrónicos que al facilitarles un link puedan descargarse la aplicación desde la propia tienda. Además, la forma de evaluación por parte de estos *early adopters* ha sido utilizando un formulario de Google Forms[12] sin ninguna supervisión.

3.2.1. Descripción del primer MVP

Para el desarrollo del primer MVP, se ha analizado el mapa de características que se obtuvo en la sección 3.1, además de definir una estrategia con el fin de que la validación sea de lo más beneficiosa posible a la hora de identificar el rumbo de la empresa. Esta validación ha sido realizada con *early adopters* cercanos a los desarrolladores y contando con la colaboración del líder del grupo «Spartans Denia Street Workout»[13], un total de diez personas han participado en este primer MVP.

La estrategia a la hora de seleccionar las funciones para este primer MVP ha sido, en primer lugar basarse en la prioridad que está dispuesta en el mapa de características bordeadas con color azul del Diagrama 1 y Diagrama 2. Además, el criterio para establecer las prioridades está centrado en la necesidad de ofrecer un producto que contenga las mínimas funcionalidades para que se considere una aplicación de deporte, calistenia en el caso que nos ocupa, y permita la gestión de rutinas y ejercicios. Comentar que respecto al tipo de entrenamiento en las rutinas existen funcionalidades que son exclusivas de esta aplicación pero que en este producto mínimo viable no se van a llevar a cabo.

Y, en segundo lugar, se implementa una de las principales funciones que permitirán a la empresa recibir ingresos a través de las suscripciones. Los usuarios al suscribirse de manera mensual o anual a la aplicación tendrán acceso ilimitado a esta función. Esta característica es el plan de entrenamiento centrado a diferentes objetivos, los cuales se basan en diferentes fases de dificultad donde cada una de ellas acerca al atleta a su meta. Sin embargo, la diferencia que ofrece AllThenics con estos planes no

[12] https://www.google.es/intl/es/forms/about/
[13] https://www.facebook.com/Spartansdeniabarz

es meramente de calidad y experiencia, sino que también ofrece en cada fase diferentes rutinas para poder adaptarse al mayor número de calisténicos posibles, pues resulta obvio que una rutina no puede afectar del mismo modo a todas las personas.

En este primer MVP también se incluye la realización de las rutinas y ejercicios en tiempo real y características que resultan ocultas para el usuario final como la base de datos o la gestión de los perfiles. Destaca el hecho, que una de las principales características novedosas que impulsan a la realización de esta aplicación como es el seguimiento detallado de estadísticas de acuerdo a entrenamientos posteriores no sea lo primero a validar, pero como se ha justificado en los párrafos anteriores, también hay que comprobar si el producto será viable económicamente y se ha decidido por comprobar la principal fuente de ingresos, y más adelante se validarán el resto de características.

Una vez llevado este planteamiento y empezado a desarrollar el primer MVP, se decidieron pasar del segundo MVP al primero el inicio de sesión y el registro, ya que se necesitaba conocer al usuario para poder vincularlo con los ejercicios o rutinas favoritas y la suscripción a los planes. Por tanto, al mover estas funcionalidades también se movieron las relacionadas con el perfil del usuario. Esta modificación provocó que algunas de las funcionalidades planteadas en un principio como la pantalla principal, la planificación de la semana de entrenamiento y la creación y edición de las rutinas tipo serie fueran relegadas al *backlog*.

Estos cambios fueron posibles gracias al uso de una metodología ágil la cual ha permitido completar los MVP y tener un producto que puede utilizarse perfectamente a pesar de que haya algunas funcionalidades que en un principio iban a ser implementadas pero que finalmente han sido postergadas ya que no eran funcionalidades prioritarias en el desarrollo de este proyecto.

En la Tabla 6 y Tabla 7 están representadas a través de la herramienta Worki Process todas las unidades de trabajo que, tras las modificaciones realizadas en el planteamiento inicial, se han llevado a cabo en el primer MVP. Al ser todas estas unidades nuevos requisitos se representan en la primera columna con una estrella y un más. Cada UT se ha incluido en el módulo correspondiente de acuerdo al apartado 3.1, en la última columna se incluye esta diferenciación.

Tabla 6. Lista UTs MVP1 (1/2)

UT	Estructura - Temas
8 - Visualizar rutinas y favoritas	Rutinas
9 - Buscar rutinas	Rutinas
10 - Filtrar rutinas	Rutinas
11 - Visualizar UNA rutina	Rutinas
14 - Visualizar ejercicios y favoritos	Ejercicios
15 - Buscar ejercicio	Ejercicios
16 - Filtrar ejercicio	Ejercicios
17 - Visualizar un ejercicio	Ejercicios
18 - Realizar ejercicio	Ejercicios
19 - Realizar rutina de tipo series	Rutinas
25 - Editar perfil	Perfil

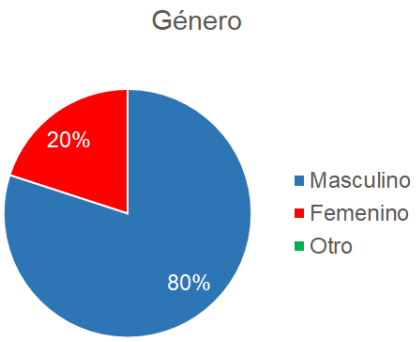
Tabla 7. Lista UTs MVP1 (2/2)

UT	Estructura - Temas
27 - Registro	Login/Registro
28 - Login	Login/Registro
29 - Visualizar planes	Planes
30 - Buscar plan	Planes
31 - Filtrar plan	Planes
32 - Visualizar UN plan	Planes
33 - Visualizar fase de un plan	Planes
37 - Crear e implementar base de datos	Base de datos
38 - Rellenar base de datos	Base de datos

3.2.2. Experimento del primer MVP

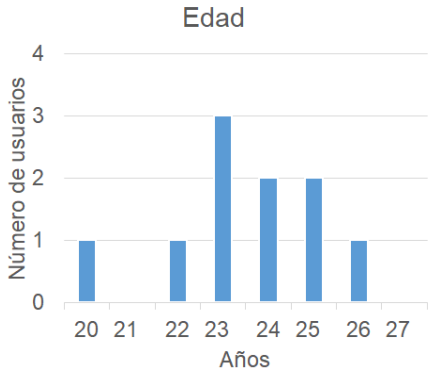
A continuación, se presentan los resultados obtenidos en el primer experimento en diferentes secciones dependiendo del ámbito de la pregunta, tal y como, se preguntó a los usuarios que probaron el sistema. En el Anexo A se encuentran todas las preguntas realizadas para el formulario de este MVP.

La primera sección son los datos personales para obtener un contexto sobre el que se encuentran los *early adopters*, donde la mayor cantidad de público de la aplicación es de género masculino como se puede observar en la Gráfica 4.



Gráfica 4. MVP1, género

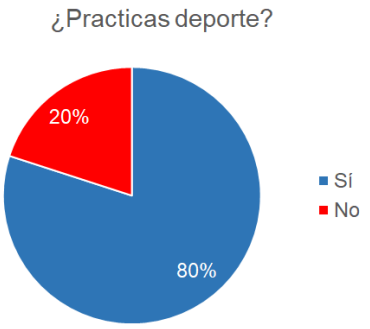
La Gráfica 5 recoge el rango de edades las cuales están comprendidas entre los 20 y 26 años, esto puede deberse a que debido a la cercanía personal de los *testers* presentan un rango de edad parecido al de los desarrolladores. En el segundo MVP se espera poder ampliar el rango para abarcar una mayor población de la cual obtener los resultados.



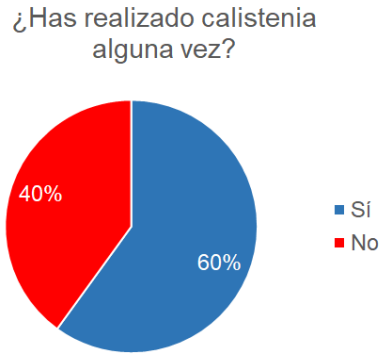
Gráfica 5. MVP1, edades

En cuanto a la actividad física se realizaron las preguntas y obtuvieron los resultados de la Gráfica 7 y Gráfica 6, donde la mayoría de usuarios sí que realizan

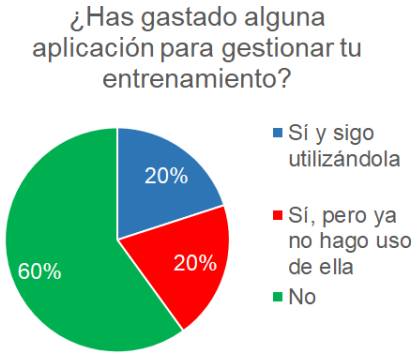
deporte pero de todos ellos un porcentaje cercano a la mitad no practican la disciplina de calistenia. Pese a ello, más de la mitad de usuarios no utilizan o han dejado de utilizar una aplicación de gestión del entrenamiento, como muestra el Gráfica 8.



Gráfica 7. MVP1, deportista



Gráfica 6. MVP1, calistenia



Gráfica 8. MVP1, uso de aplicaciones

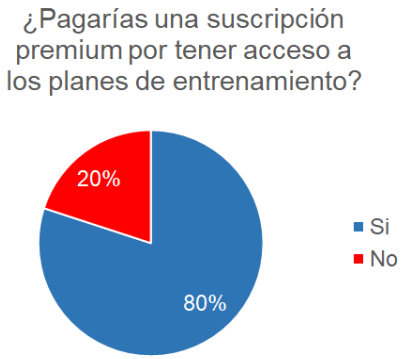
El siguiente sector del formulario guarda relación con la interfaz de la aplicación. En ella los usuarios afirmaron que la navegabilidad es correcta de forma que resulta sencilla e intuitiva. Pese a ello también ha habido algunas críticas respecto a la tipografía utilizada en los listados de ejercicios, rutinas y planes y, aunque los colores han sido aceptados por la mayoría, hay sugerencias de utilizar colores más vivos y utilizarlos también para diferenciar etapas dentro de una rutina, como nos indica este usuario, «Creo que añadiría algún color para diferenciar entre descanso, cambio de series y el propio ejercicio». Por último, destacar que el 100% de los usuarios creen que los iconos de toda la aplicación son lo suficientemente significativos para poder entender de un solo vistazo la funcionalidad a la cual acceden.

La funcionalidad en la aplicación, la cual era la siguiente sección a analizar, también ha sido adecuada en todos los ámbitos, tanto ejercicios, rutinas y planes como a la hora de realizar un entrenamiento. Sin embargo, es en este último punto donde los

usuarios difieren entre ellos. Al 60% de usuarios le ha resultado cómodo y útil realizar una rutina mientras que un 40% cree que habría que realizar algún cambio en cuanto a los sonidos utilizados y la forma de interactuar con los botones, en la Gráfica 9 se encuentran estos resultados. Una sugerencia donde han coincidido estos early adopters ha sido en la introducción de un vídeo o GIF explicativo para la correcta realización de los ejercicios, pese a que esto ya se tenía en cuenta en el ámbito del proyecto no se decidió llevar a cabo para poder realizar pruebas sobre funcionalidades más comprometedoras.

Gráfica 9. MVP1, funcionalidad rutina

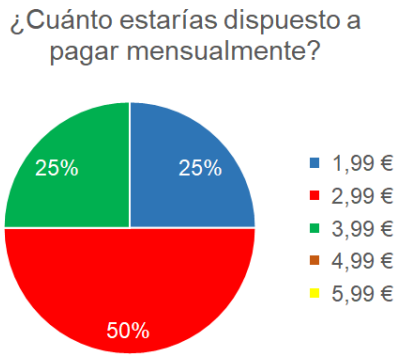
El grupo de monetización ha sido la sección por la que se ha llevado a cabo este MVP, por ello, se ha analizado con un principal interés para conocer si el modelo de negocio propuesto a través de suscripción es el correcto o, por el contrario, si se debería plantear algún que otro cambio. Una de las mayores dudas era saber si la principal funcionalidad *premium* de la aplicación iba a acabar siendo lo suficientemente atractiva para los usuarios y la Gráfica 10 confirma que sí, aunque será en el segundo experimento con una mayor base de usuarios cuando se conocerá realmente.



Gráfica 10. MVP1, suscripción premium

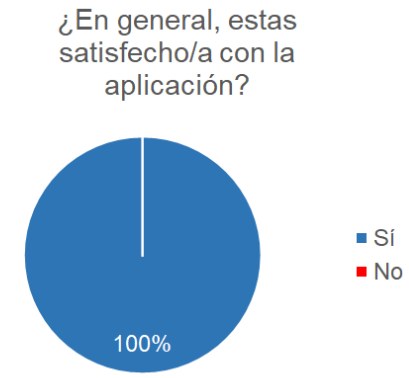
Otra de las grandes dudas es el precio de suscripción mensual adecuado que como la anterior pregunta se conocerá correctamente con una mayor cantidad de

usuarios, pero se puede ir estimando actualmente. Según los primeros *early adopters* el precio adecuado sería de 2,99€, como muestra la Gráfica 11 con el porcentaje de votos a cada precio ofertado, aunque hay usuarios que estarían dispuestos a pagar en un rango muy similar a este.

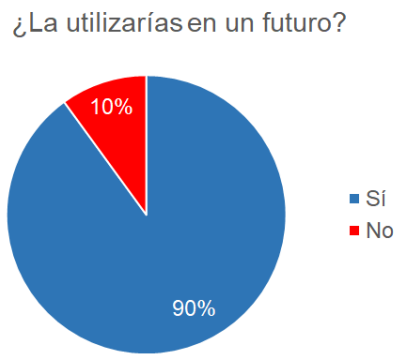


Gráfica 11. MVP1, pago mensual

Para finalizar el cuestionario, se realizaron preguntas sobre la opinión general de AllThenics. A pesar de muchas sugerencias y algún que otro fallo, el 100% de usuarios se encuentran satisfechos con la aplicación y volverían a hacer uso de ella excepto una persona que comentó que pese a estar encantada con AllThenics no cree que en futuro la siga utilizando, estas conclusiones se representan mediante la Gráfica 12 y Gráfica 13. Pese a que no se conoce la identidad de esta persona puede sospecharse que fuera la que escribió el siguiente comentario en el formulario «No cambiaría nada, tan solo esos no son para mí», es decir, que no la vaya a utilizar por los ejercicios propuestos en la aplicación o por la propia práctica de ejercicio con el propio peso corporal.



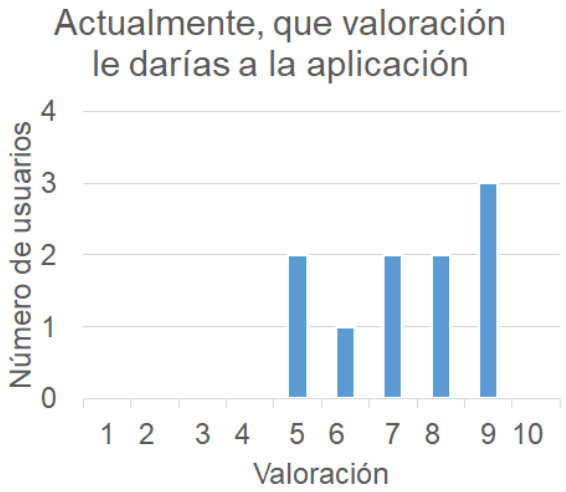
Gráfica 12. MVP1, satisfacción



Gráfica 13. MVP1, uso futuro

Como última pregunta resulta conveniente conocer que puntuación se obtendría con el porcentaje de desarrollo actual, el resumen de estas valoraciones se encuentran en la Gráfica 14. Este primer experimento se corresponde con un notable 7,3 el cual ponderado a la puntuación en Google Play se transforma en un 3,6 sobre 5. A pesar de ser una valoración baja se espera que, al arreglar los fallos, introducir las sugerencias y añadir las funcionalidades más innovadoras haga que mejore esta calificación.

Por último, cabe destacar un comentario que nos ha motivado a continuar con el desarrollo sin desalentarnos, el cual dice lo siguiente: «Siendo el primer testeo el 8 se lo lleva, se pueden mejorar cosas, pero va por buen camino. Hasta yo que no soy de hacer ejercicio estoy por quedármela».



Gráfica 14. MVP1, notas de la aplicación

3.3. Desarrollo del segundo MVP

En este apartado se va a desarrollar todo el segundo MVP describiendo el ámbito que abarca y comentado los resultados obtenidos en el experimento con un conjunto mayor de *early adopters* respecto al primer MVP. Se han gastado las mismas directrices que en el primer MVP haciendo uso de Google Play y realizando la encuesta mediante Google Forms.

3.3.1. Descripción del segundo MVP

Para el desarrollo del segundo MVP, se implementan aquellas características marcadas en color rojo del Diagrama 1 y Diagrama 2. Esta validación se ha realizado por los usuarios del primer MVP que participarán en la totalidad del desarrollo de la

aplicación y con varias personas del grupo Spartans Denia Street Workout gracias a su líder el cual ha quedado encantado con la aplicación y ha decidido compartirla con sus compañeros. También se han añadido nuevos *early adopters* ajenos a estos grupos para tener una mayor población de usuarios.

En primer lugar, se han llevado a cabo algunas de las sugerencias y mejoras propuestas por los primeros *early adopters c*omo son el filtrar rutinas por objetivos, mantener la pantalla activa durante un entrenamiento o el número de resultados obtenidos en el filtrado. En segundo lugar, se implementa todo lo relacionado con la monitorización de entrenamientos, es decir toda la parte de progreso del usuario, que es una de las partes innovadoras de la aplicación. En tercer y último lugar, la posibilidad de realizar diferentes tipos de entrenamiento sin la necesidad de interactuar con el dispositivo móvil de forma que el tiempo de descanso y actividad avance automáticamente como son las rutinas de HIIT, EMOM y tabata y, por otro lado, una rutina de tipo piramidal donde se aumentan y/o disminuyen las series de manera incremental.

En la Tabla 8 y Tabla 9 están representadas a través de la herramienta Worki todas las unidades de trabajo que se han llevado a cabo en el segundo MVP. En la Tabla 8 se encuentran todas las unidades que son nuevos requisitos las cuales se representan en la primera columna con una estrella y un más. Sin embargo, aquellas unidades de trabajo que se tratan de mejoras son las listadas en la Tabla 9, representadas únicamente con una estrella. Cada UT se ha incluido en el módulo correspondiente de acuerdo al apartado 3.1, en la última columna se incluye esta diferenciación.

Tabla 8. Lista UTs MVP2 (1/2)

	UT	Estructura - Temas
	17 - Visualizar un ejercicio	Ejercicios
	19 - Realizar rutina de tipo series	Rutinas
	20 - Visualizar progreso	Progreso
	21 - Visualizar dia en calendario	Progreso
	22 - Añadir rutina en calendario	Progreso
	23 - Añadir ejercicio en calendario	Progreso
	37 - Crear e implementar base de datos	Base de datos
	38 - Rellenar base de datos	Base de datos
	41 - Realizar rutina de tipo tiempo	Rutinas
	42 - Añadir plan al calendario	Progreso
	53 - Rutinas para ti	Rutinas

Tabla 9. Lista UTs MVP2 (2/2)

	UT	Estructura - Temas
	44 - Filtro por objetivo	Rutinas Planes
	45 - Mostrar reps/secs ejercicio en rutinas	Rutinas
	46 - Mejorar menu al iniciar entrenamiento	Rutinas Ejercicios
	47 - Cambiar sonidos tiempo	Rutinas Ejercicios
	48 - Filtro con número de resultados	Rutinas Ejercicios Planes
	49 - Mantener pantalla activa en entrenamiento	Rutinas Ejercicios
	50 - Arreglar fallo sonido al salir	Rutinas Ejercicios
	51 - Cambiar titulos de AppBar	Rutinas Ejercicios Planes
	52 - Cambiar color navegacion menu Android	Ajustes

Como también ocurrió con el primer MVP se han tenido que dejar en el *backlog* funcionalidades que en un principio se iban a llevar a cabo en este *sprint* como son la creación y edición de rutinas de tipo tiempo y la configuración y notificación del recordatorio del entrenamiento de la planificación semanal. Con las funcionalidades que actualmente permanecen en el *backlog* se va a llevar a cabo un tercer MVP fuera del ámbito académico para construir una aplicación completa y poder comercializarla con todas las características que se tenían planificadas en un principio en el apartado 3.1. Por consiguiente, el mapa de características después de haber realizado las modificaciones durante el desarrollo y el añadido de un tercer sprint se establece de la manera mostrada en el Diagrama 3 y Diagrama 4, donde los colores azul, rojo y verde corresponden con el primer, segundo y tercer *sprint* respectivamente.

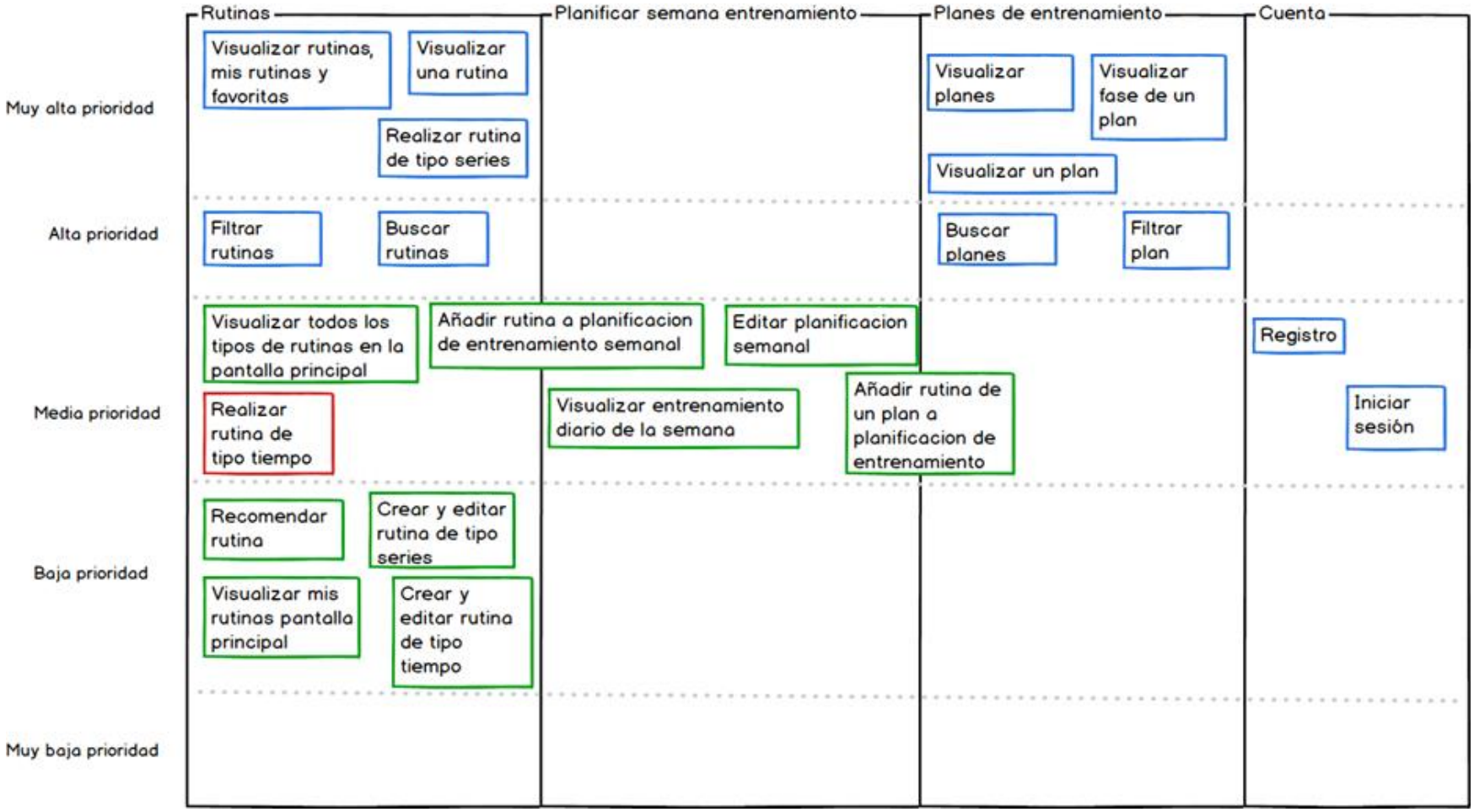


Diagrama 3. Mapa de características final (1/2)

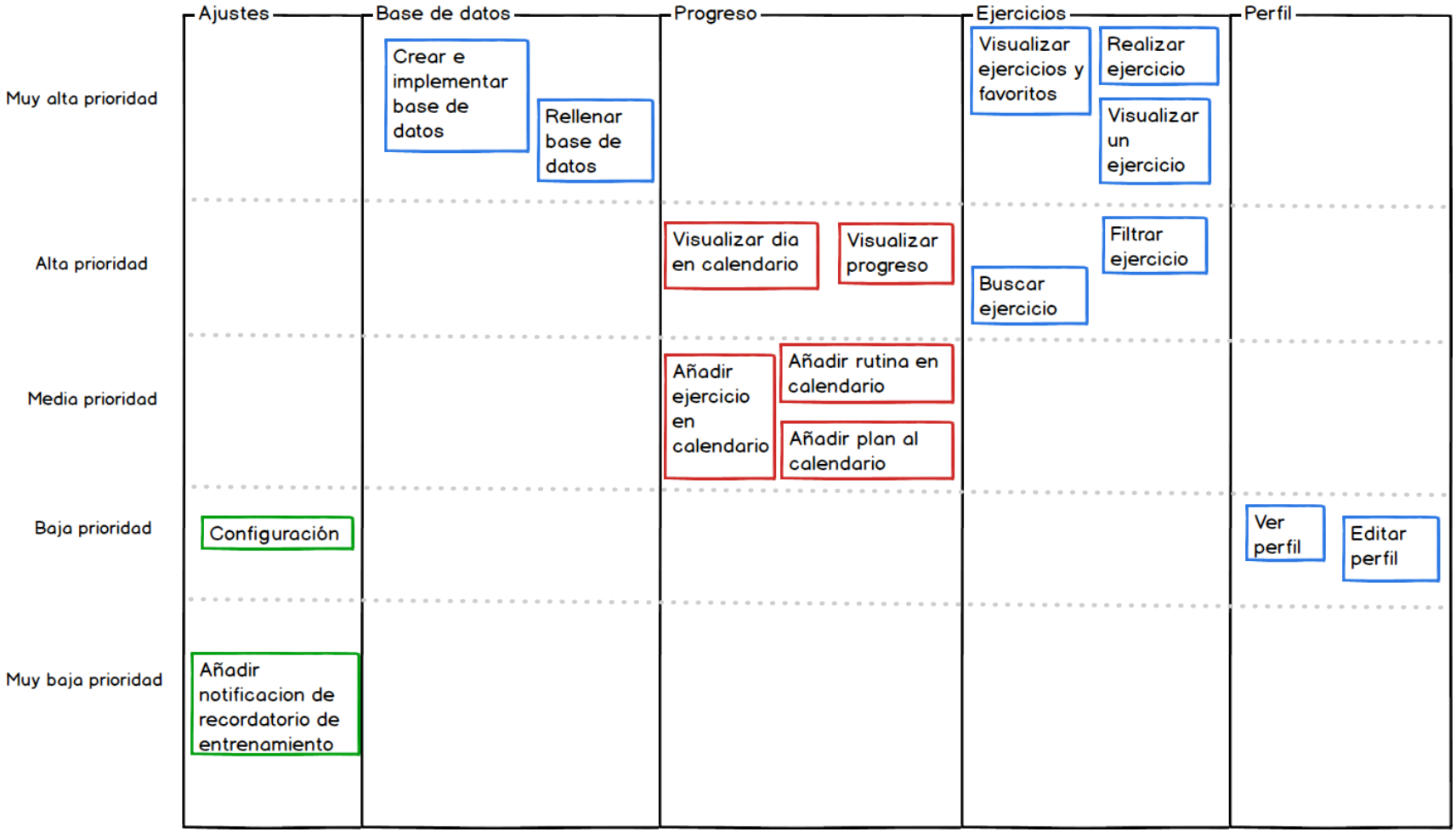
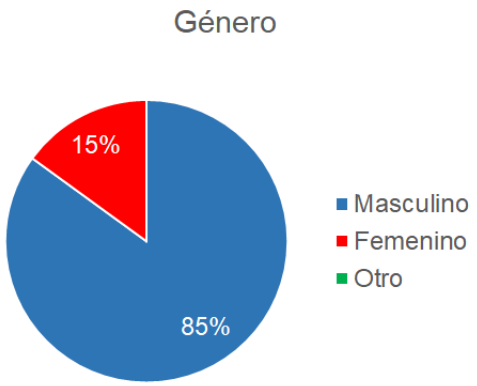


Diagrama 4. Mapa de características final (2/2)

3.3.2. Experimento del segundo MVP

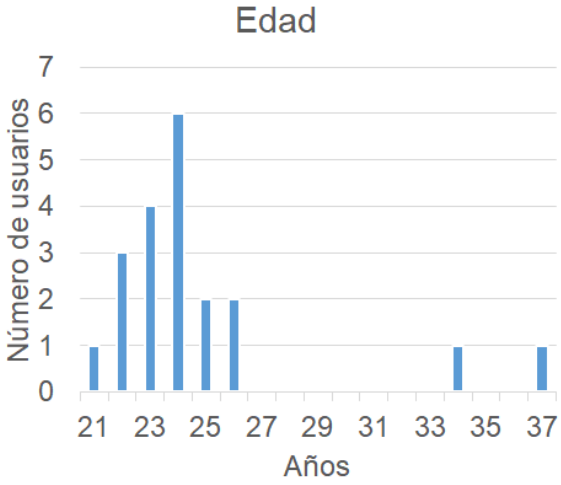
A continuación, se presentan los resultados obtenidos en el segundo experimento en diferentes secciones dependiendo del ámbito de la pregunta, tal y como, se preguntó a los participantes que probaron el sistema. En el Anexo B se encuentran todas las preguntas realizadas para el formulario de este MVP.

La primera sección son los datos personales para obtener un contexto sobre el que se encuentran los *early adopters*, donde la mayor cantidad de público de la aplicación es de género masculino como se puede observar en la Gráfica 15.



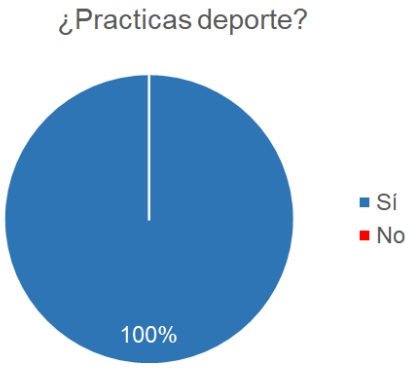
Gráfica 15. MVP2, género

La Gráfica 16 recoge el rango de edades las cuales siguen comprendidas entre los 20 y 26 años debido a, como en el anterior MVP, a la cercanía personal de los *testers* que presentan un rango de edad parecido al de los desarrolladores.

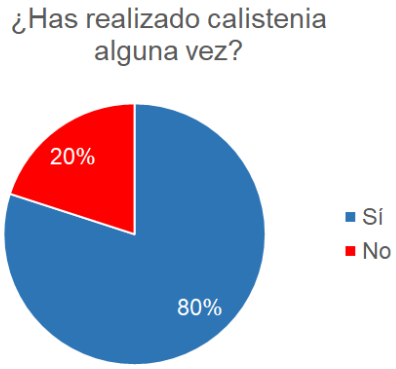


Gráfica 16. MVP2, edades

En cuanto a la actividad física se han mejorado los resultados respecto al primer MVP, donde se obtenía un 20% que no practicaba deporte y solo un 60% que realizaba calistenia. Sin embargo, a través de la Gráfica 17 y Gráfica 18, es notable el crecimiento de deportistas donde todos los usuarios han indicado que realizan deporte y de todos ellos un 20% no practican la disciplina de calistenia, este aumento se encuentra relacionado con la participación de *early adopters* del grupo Spartans Denia Street Workout. Por último, un 95% no utilizan o han dejado de utilizar una aplicación de gestión del entrenamiento, como muestra la Gráfica 19.



Gráfica 17. MVP2, deportista



Gráfica 18. MVP2, calistenia

Gráfica 19. MVP2, uso de aplicaciones

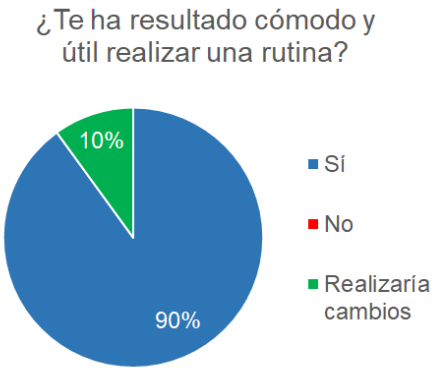
El siguiente sector del formulario guarda relación con la interfaz de la aplicación. En el anterior MVP se criticaron colores y tipografía, por tanto, se puso especial hincapié en estos apartados subsanando estos errores y consiguiendo un 100% de aceptación en ambos.

Además, en este MVP se ha introducido el progreso el cual está formado por la inserción de diferentes gráficas para poder visualizarlo y de nuevas vistas. Con todo esto los usuarios han quedado completamente satisfechos, en la Gráfica 20 se ejemplifica como de útiles les han resultado las diferentes gráficas.

Gráfica 20. MVP2, gráficas progreso

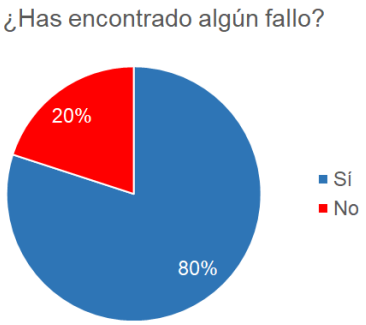
La funcionalidad en la aplicación, la cual era la siguiente sección a analizar, sigue siendo adecuada, como ya fue en el experimento anterior. Incluso las nuevas funcionalidades añadidas como todo el progreso y las diferentes rutinas han sido bien aceptadas. Además, de un 60% que se obtuvo en el primer experimento respecto a la cómodo que era realizar una rutina, se ha conseguido mejorar esta faceta hasta un 90% de los encuestados. En la Gráfica 21 se encuentran estos resultados.

Una sugerencia donde siguen coincidiendo los *early adopters* ha sido en la introducción de un vídeo o GIF explicativo para la correcta realización de los ejercicios, que como ya se comentó, es una idea que se acabará implementando en un futuro.



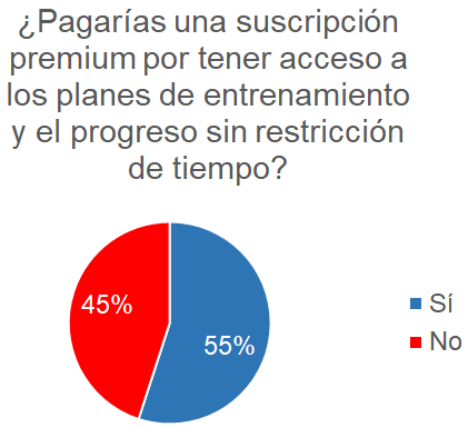
Gráfica 21. MVP2, funcionalidad rutina

Para finalizar la sección de funcionalidad cabe destacar que un 20% de usuarios han encontrado algún fallo en la aplicación, como indica la Gráfica 22. Uno de los errores más comentados es el relacionado con el cronómetro de rutinas como indica este usuario, «Al salir de la aplicación para contestar un WhatsApp o atender una llamada el cronómetro de una rutina deja de funcionar y debes volver a empezar la sesión». Este tipo de errores son bastante complicados de detectar y estos experimentos han sido muy útiles para ello y darles solución.



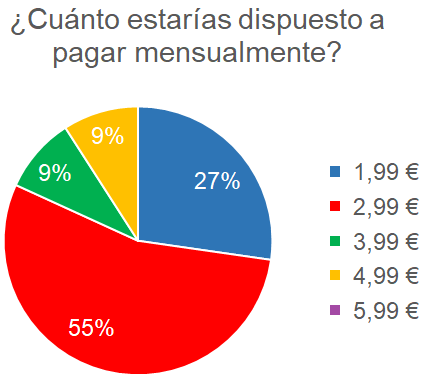
Gráfica 22. MVP2, fallos encontrados

En lo relativo a la monetización, en el anterior MVP no quedaba completamente claro si este modelo de negocio con funcionalidad *premium* iba a ser el adecuado porque se tenía una cantidad muy pequeña de usuarios y en este MVP tampoco se ha podido garantizarlo completamente ya que solo un 55% de los encuestados dicen que sí que pagarían por las funcionalidades *premium* de planes y de mayor tiempo en el progreso, como se observa en la Gráfica 23. Posiblemente se debería replantear el modelo de negocio o mantener el actual con algún tipo de añadido más para hacer más vistosa la funcionalidad *premium* de la aplicación.



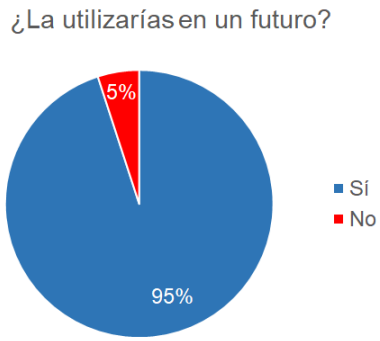
Gráfica 23. MVP2, suscripción premium

Sin embargo, de los que sí que pagarían por la funcionalidad *premium* el resultado en cuanto al coste mensual es mayoritariamente con un 55%, como en el anterior experimento, de 2,99€. En la Gráfica 24 se pueden observar estos resultados.



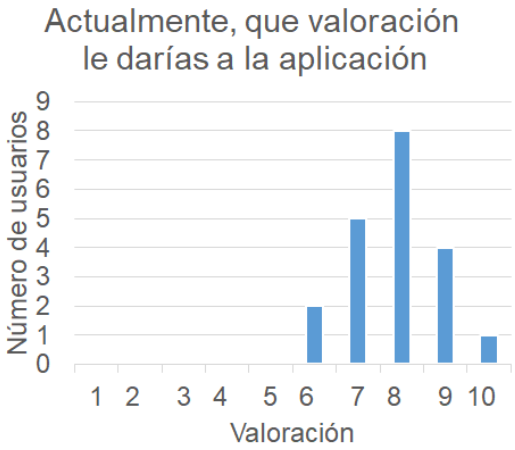
Gráfica 24. MVP2, pago mensual

Para finalizar también este cuestionario se realizaron preguntas sobre la opinión general de AllThenics. El 100% de usuarios se vuelven a encontrar satisfechos con la aplicación y excepto un 5% de los encuestados, que no han dado ninguna razón aparente, volverían a hacer uso de ella. Estas conclusiones se representan mediante la Gráfica 25.



Gráfica 25. MVP2, uso futuro

Otra vez como última pregunta se ha optado por conocer, con el porcentaje de desarrollo actual, la valoración que podría obtener la aplicación. Para variar un poco la forma de puntuar se ha optado por valorarla directamente como si fuese una aplicación en Google Play, es decir, sobre cinco en vez de sobre diez, y el resultado obtenido ha sido de un 3,9 mejorando la marca del anterior experimento que se colocaba en un 3,6 de valoración, lo que significa que la aplicación va por buen camino. En la Gráfica 26 viene representado este resultado escaladas las notas sobre diez en cuyo caso la nota escala a 7,9.



Gráfica 26. MVP2, notas de la aplicación

3.4. Cronología

En este apartado se comentan las fechas más importantes en el desarrollo de la idea de negocio para poder tener una estimación de tiempo invertido en cada etapa, en el Diagrama 5 se puede ver sencilla y visualmente.

La primera etapa empieza el día 1 de junio de 2020 con el desarrollo del primer MVP el cual se extiende hasta el día 5 de julio de 2020. En un principio al ser el desarrollo por *sprints* de cuatro semanas se debería haber acabado una semana antes, pero por problemas de última hora se tuvo que alargar el desarrollo una semana más de lo previsto.

A continuación, el día 6 de julio de 2020 se puso a disposición de Google la aplicación para que fuese subida a la Play Store, pero no fue hasta el día 11 de julio de 2020 cuando la aplicación fue finalmente publicada y desde este día al 14 de julio de 2020 nuestros *early adopters* pudieron probar la aplicación y evaluarla mediante un formulario.

Una vez finalizada la prueba y evaluada, el día 16 de julio de 2020 empezó el desarrollo del segundo MVP el cual se extendió hasta el día 16 de agosto de 2020. Al reorganizar las tareas en tres MVPs este segundo se pudo desarrollar dentro de los plazos planificados sin ningún tipo de inconveniente. Al disponer de esta versión finalizada se realizar los trámites para subirla a la Play Store la cual únicamente precisó de un día para llevarlo a cabo. Por ello, desde el 18 hasta el 20 de agosto los *early adopters* pudieron realizar las pruebas pertinentes sobre la aplicación y proceder a

evaluarla. Después de ello y de realizar el análisis de las respuestas obtenidas sobre esta nueva versión se dio por finalizado el proyecto el día 23 de agosto.

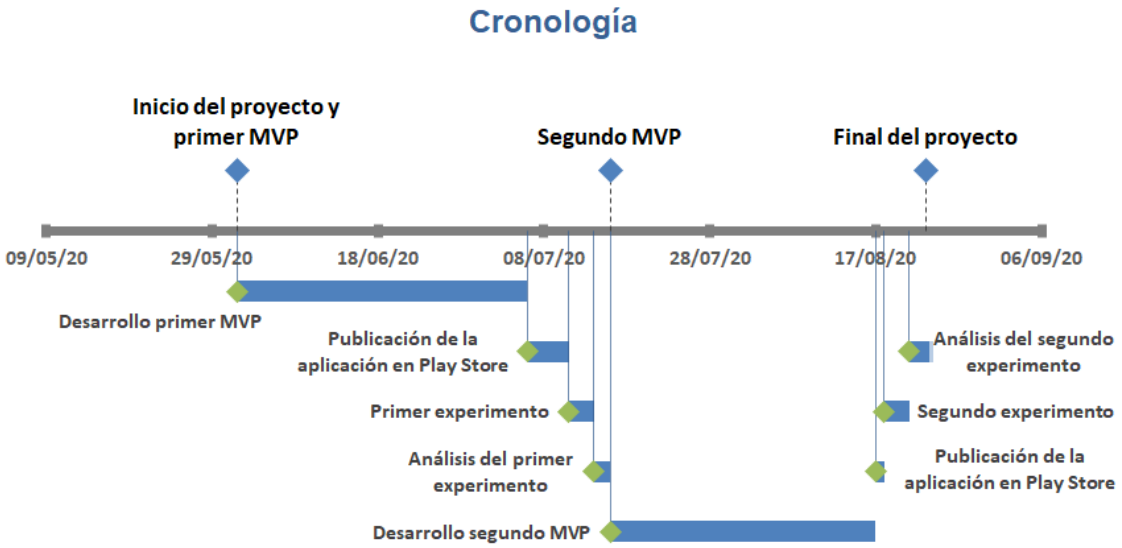


Diagrama 5. Cronología

Capítulo 4

Especificación

A partir del capítulo actual, la memoria difiere del compañero de proyecto Víctor Cardona Lorenzo, ya que para la implementación de la aplicación se ha dividido de tal forma que él se hace cargo de la parte de *front-end* y, por mi parte, el *back-end*. Por lo tanto, en esta memoria se analiza todo lo relacionado con la parte de la arquitectura y de la base de datos. En la memoria «AllThenics, aplicación móvil para calistenia: desarrollo del *front-end*» es donde se encuentran desarrolladas otras partes del proyecto como son los casos de uso o *mock-ups* entre otros.

Una vez descritos los experimentos con sus resultados correspondientes, se describen aspectos más técnicos de la propia aplicación. En primer lugar, se desarrolla la especificación necesaria para diseñar y entender el proceso de creación de la base de datos utilizada.

4.1. Diseño y base de datos

En este apartado se desarrollan las diferencias existentes entre SQL y NoSQL y, además, la explicación de cómo está construida la base de datos NoSQL de la aplicación AllThenics.

Para empezar, la Ilustración 18 sirve como un buen resumen de las diferencias arquitectónicas entre estas dos tecnologías.

4.1.1. SQL vs NoSQL

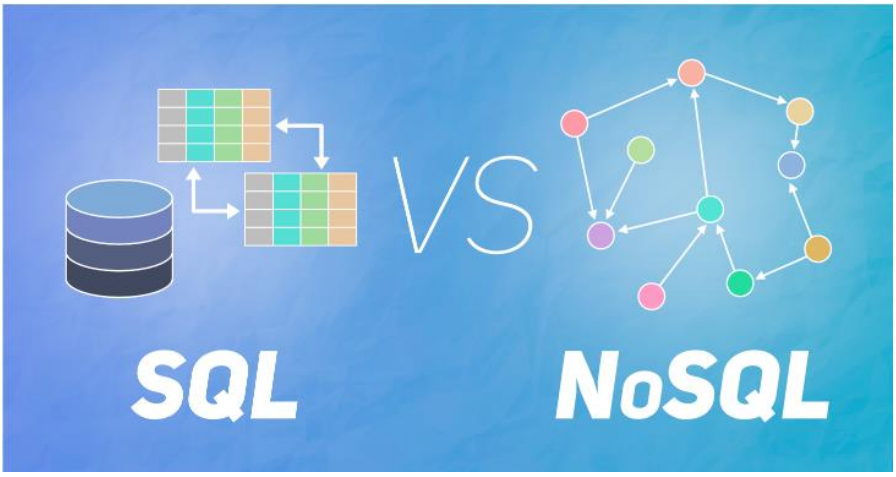


Ilustración 18. Diferencia estructural entre SQL y NoSQL[14]

SQL (*Structured Query Language*) es un lenguaje diseñado para recuperar y administrar datos en los sistemas gestores de bases de datos relacionales y el más extendido debido a su potencia, simplicidad y longevidad.

Su estructura consiste en tablas relacionadas entre ellas que contienen filas con datos. Este tipo de bases de datos estructuradas no son muy flexibles, tampoco muy escalables y en cuestión de rendimiento las NoSQL están bastante por encima de ellas.

Pero sí que existen ventajas, al ser un producto con un largo recorrido, la cantidad de información para poder desarrollar correctamente una base de datos SQL es muy grande. Asimismo, los estándares están perfectamente definidos y su capacidad de entendimiento y sencillez de escritura hacen que muchas personas sin excesivos conocimientos de informática puedan entender.

Por último, destacar la atomicidad de sus operaciones, gracias a ello se puede garantizar que al realizar una operación solo se complete si se ha hecho correctamente al cien por cien y en caso de algún error intermedio no realizar ningún cambio.

Sin embargo, NoSQL ha llegado para quedarse.

NoSQL (*Not Only SQL*) es otro tipo de forma de almacenamiento de los datos y surgió por la necesidad de grandes empresas como Facebook o Twitter que necesitaban dar servicio a millones de usuarios concurrentes haciendo peticiones a las bases de

[14] Imagen obtenida de https://www.geeksforgeeks.org/sql-vs-nosql-which-one-is-better-to-use/

datos constantemente y en tiempo real, por tanto, necesitaban más escalabilidad que consistencia.

En este caso no existen tablas con relaciones entre ellas, simplemente no tenemos un esquema *per se* sino es como una estructura en árbol, y todas las comunicaciones se hacen en notación JSON (*Java Script Object Notation*).

En vez de tablas se tienen colecciones y en vez de filas dentro de la tabla, se almacenan documentos. Estos documentos tienen campos que serían como una especie de atributos, pero siendo en este caso pares clave-valor donde los desarrolladores deben controlar manualmente estas claves para que no haya inconsistencias entre documentos.

Cuando se almacena un objeto en NoSQL se introduce el objeto entero sin la necesidad de separar los datos, por tanto, la recuperación es más rápida mejorando así el rendimiento, ya que con SQL se deberían hacer JOINS mapeando la consulta para, posteriormente, mostrar el resultado a los usuarios.

4.1.2. Estructura SQL

Al principio del desarrollo y antes de conocer Firebase, se estuvo pensando en utilizar dos bases de datos SQL, una interna en el móvil utilizando la tecnología SQLite y otra externa gastando MySQL o similar.

Para ello, se diseñó un boceto que se corresponde con el siguiente diagrama, Diagrama 6, que se ha decidido introducir en la memoria como especificación para entender un poco mejor las relaciones y las tablas pese a que finalmente no se ha hecho exactamente uso de él y, por tanto, no será explicado en profundidad.

En el apartado 4.1.3, se comentará en formato JSON la base de datos que realmente se ha implementado y la cual se está utilizando, explicando ya detalladamente que significa cada atributo. Asimismo, en el apartado 6.2.1, se tratará como se ha llevado a cabo en la aplicación cliente para recibir y enviar los datos a la base datos.

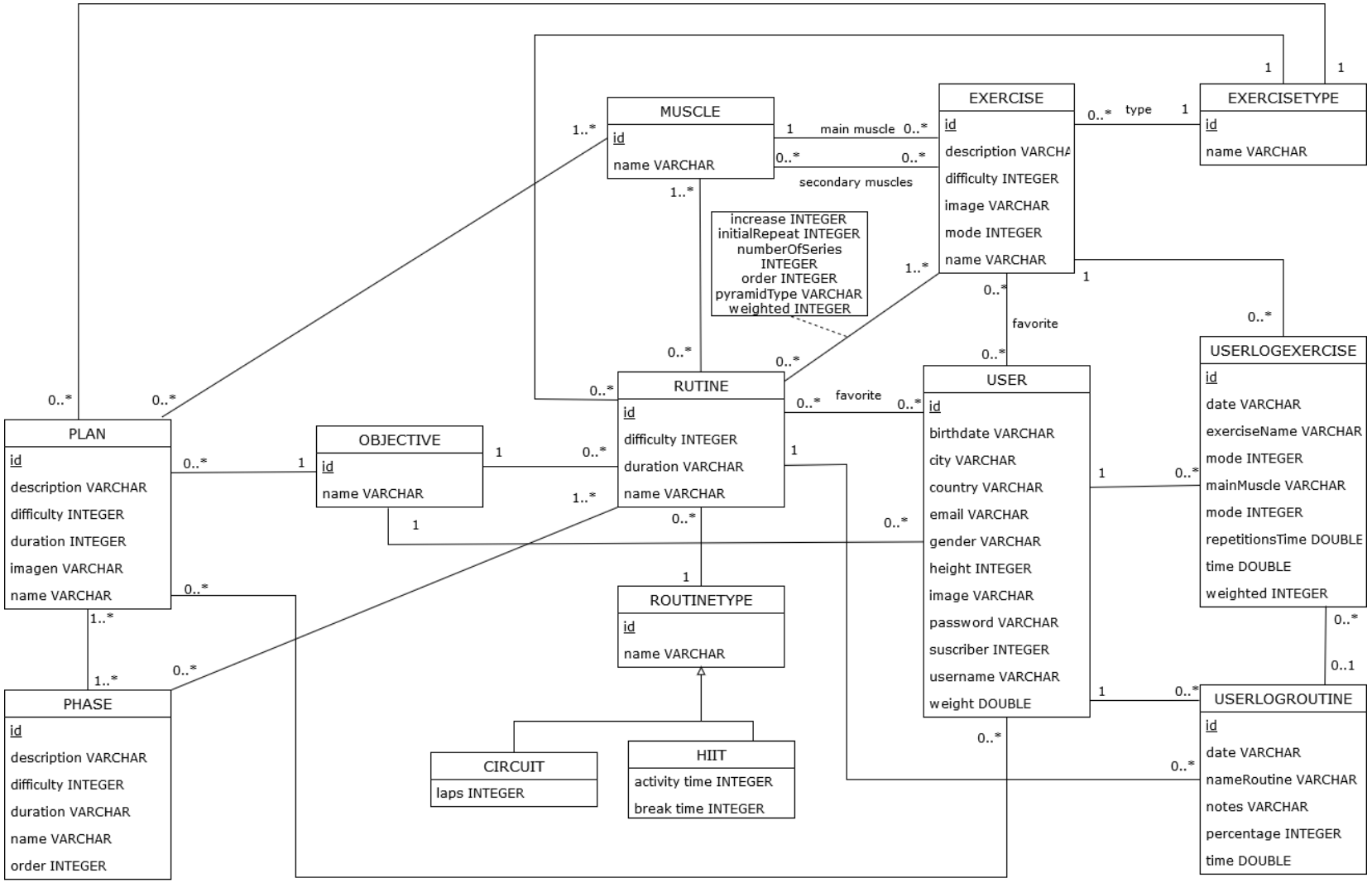


Diagrama 6. Boceto del diagrama de clases UML de la base de datos

- Exercise: tabla con los datos de todos los ejercicios.
- ExerciseType: tabla con el tipo del ejercicio (pese a su nombre, también sirve para rutina y plan).
- Muscle: tabla con los nombres de los diferentes músculos.
- Objective: tabla con los nombres de los diferentes objetivos.
- Plan: tabla con los datos de los planes donde un plan tendrá varias fases.
- Phase: tabla con los datos de las fases donde una fase tendrá varias rutinas.
- Routine: tabla con los datos de las rutinas donde una rutina tendrá varios ejercicios. Destacar que en este caso al hacerse la relación hay que almacenar más datos que surgen de la propia relación como el número de repeticiones o de series.
- RoutineType: tabla con el nombre de los tipos de rutina que existen.
- Circuit: tabla que hereda de RoutineType añadiendo un atributo en caso de que la rutina sea de este tipo.
- Hiit: tabla que hereda de RoutineType añadiendo dos atributos en caso de que la rutina sea de este tipo.
- User: tabla con los datos del usuario, un usuario puede tener varios ejercicios y rutinas como favoritas además de poder suscribirse a algún plan para poder realizarlo.
- UserLogExercise: tabla con los datos de un ejercicio para llevar en cuenta el progreso del usuario haciendo ese ejercicio.
- UserRoutineExercise: tabla con los datos de una rutina para llevar en cuenta el progreso del usuario haciendo esa rutina.

4.1.3. Estructura NoSQL

A continuación, se comenta realmente la estructura que se ha implementado y se describe cada colección junto con sus atributos. Hay que recordar que una colección está compuesta de documentos y un documento está compuesto por pares clave-valor. En la Ilustración 19 podemos ver todas las colecciones que se han creado en Firebase.

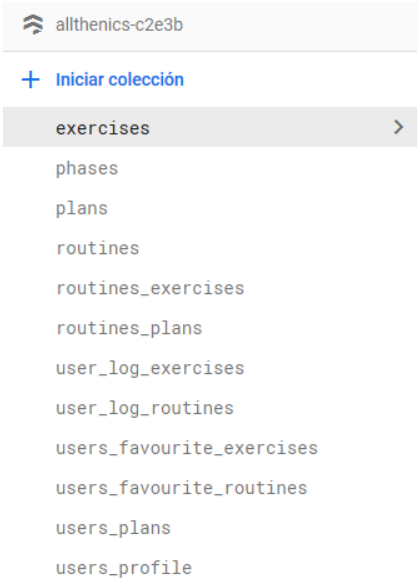


Ilustración 19. Colecciones en Firebase

Seguidamente se desarrollan todas las colecciones con los correspondientes claves-valor que existen en cada una en formato JSON. Como se podrá observar muchas tablas que ya existen y tablas que surgirían de las relaciones del diagrama SQL del Diagrama 6 han desaparecido o no se han implementado porque han pasado formar parte de un par clave-valor de las colecciones.

exercises: colección con todos los ejercicios.

{

"*documentId*": identificador único del documento.

{

"*description*": breve descripción del ejercicio.

"*difficulty*": dificultad que tiene el ejercicio, puede ser 1, 2, 3 o 4. A mayor número, mayor dificultad.

"*exerciseType*": nombre con el tipo de ejercicio.

"*image*": URL de Firebase Storage con la imagen del ejercicio.

"*mainMuscle*": nombre del músculo principal que se ejercita.

“*mode*”: número para identificar si son repeticiones (0) o segundos (1).

“*name*”: nombre del ejercicio.

“*secondaryMuscles*”: vector de strings con más músculos que se ejercitan en ese ejercicio.

}

}

phases: colección con todas las fases.

{

“*documentId*”: identificador único del documento.

{

“*description*”: breve descripción de la fase.

“*difficulty*”: dificultad que tiene el ejercicio, puede ser 1, 2, 3 o 4. A mayor número, mayor dificultad.

“*duration*”: duración, en semanas, de la fase.

“*idPlan*”: identificador único del documento del plan en el cual está incluida esta fase.

“*name*”: nombre de la fase.

“*order*”: número para identificar el orden de la fase dentro del plan.

}

}

En esta colección de fases se observa cómo se han hecho todas las relaciones. Por ejemplo, entre *phases* y *plans* se ha puesto como clave-valor en la colección *phase*, el id único del *plan* al cual hace referencia.

Otra aproximación que se podría haber realizado y que en un futuro se realizará es, en el *plan* haber introducido todo el objeto *phase* porque así al recuperar directamente el *plan* se obtendrían todos los objetos *phase* de ese plan en vez de tener que recoger el id del *plan* y después obtener las *phases* desde su propia colección.

Esto se ha hecho actualmente así para no sobrecargar la base de datos con elementos duplicados y poder mantener el régimen gratuito de Firebase en estas primeras fases de la aplicación, aunque esto es una de las principales ventajas de NoSQL mejorando bastante el rendimiento de las consultas y que como se ha comentado anteriormente se hará uso de esta funcionalidad en un futuro cuando se pase a un régimen de pago al tener ya una fiel base de usuarios.

plans: colección con todos los planes.

{

"*documentId*": identificador único del documento.

{

"*description*": descripción completa del plan.

"*difficulty*": dificultad que tiene el ejercicio, puede ser 1, 2, 3 o 4. A mayor número, mayor dificultad.

"*duration*": duración, en semanas, del plan.

"*exerciseType*": nombre del tipo de ejercicio que se hace con el plan.

"*image*": URL de Firebase Storage con la imagen del plan.

"*muscles*": vector con todos los músculos que se entrenan con el plan.

"*name*": nombre del plan.

"*objective*": nombre del objetivo que se consigue con el plan.

}

}

routines: colección con todas las rutinas no *premium.*

{

"*documentId*": identificador único del documento.

{

"*activityTime*": tiempo realizando la rutina. Solo con tipo *Hiit*.

"*breakTime*": tiempo de descanso de la rutina. Solo con tipo *Hiit*.

"*difficulty*": dificultad que tiene la rutina, puede ser 1, 2, 3 o 4. A mayor número, mayor dificultad.

"*duration*": duración, en minutos, de la rutina.

"*exerciseType*": nombre del tipo de ejercicio que se hace con la rutina.

"*laps*": número de veces que se hace la rutina. Solo con tipo Circuito.

"*muscles*": vector con todos los músculos que se entrenan con la rutina.

"*name*": nombre de la rutina.

"*objective*": nombre del objetivo que se consigue con la rutina.

"*routineType*": nombre del tipo de la rutina.

}

}

routines_exercises: colección relacionando las rutinas con los ejercicios.

{

"*documentId*": identificador único del documento.

{

"*idExercise*": identificador único del documento del ejercicio.

"*idRoutine*": identificador único del documento de la rutina.

"*increase*": número con el incremento del ejercicio en la rutina. Solo para ejercicios con tipo de rutina piramidal.

"*initialRepeat*": número de repeticiones que debe hacer el usuario del ejercicio en cuestión.

"*numberOfSeries*": número de series que debe hacer el usuario del ejercicio en cuestión.

"*order*": orden del ejercicio dentro de la rutina.

"*pyramidType*": tipo de pirámide del ejercicio. Solo para ejercicios con tipo de rutina piramidal.

"*weighted*": cantidad de peso a introducir para llevar a cabo el ejercicio.
}

}

routines_plans: colección relacionando las rutinas con las fases, estas rutinas son las rutinas *premium* que solo aparecerán dentro de las fases de los diferentes planes.

{

"*documentId*": identificador único del documento.

{

"*activityTime*": tiempo realizando la rutina. Solo con tipo *Hiit*.

"*breakTime*": tiempo de descanso de la rutina. Solo con tipo *Hiit*.

"*difficulty*": dificultad que tiene la rutina, puede ser 1, 2, 3 o 4.

"*duration*": duración, en minutos, de la rutina.

"*exerciseType*": nombre del tipo de ejercicio que se hace con la rutina.

"*idPhase*": identificador único del documento de la fase en el cual está incluida esta rutina.

"*laps*": número de veces que se hace la rutina. Solo con tipo Circuito.

"*muscles*": vector con todos los músculos que se entrenan con la rutina.

"*name*": nombre de la rutina.

"*objective*": nombre del objetivo que se consigue con la rutina.

"*routineType*": nombre del tipo de la rutina.

}

}

user_log_exercises: colección relacionando el usuario con los datos de un ejercicio realizado para los datos del progreso.

{

"*documentId*": identificador único del documento.

{

"*date*": fecha en la que se realizó el ejercicio.

"*exerciseName*": nombre del ejercicio realizado.

"*idUser*": identificador único del usuario.

"*idUserLogRoutine*": identificador único de la rutina a la cual pertenece el ejercicio, si tiene.

"*mainMuscle*": músculo principal entrenado.

"*mode*": número para identificar si son repeticiones (0) o segundos (1).

"*repetitionsTime*": vector con los tiempos de cada serie/repetición.

"*time*": duración, en segundos, de la realización del ejercicio.

"*weighted*": peso aplicado al ejercicio.

}

}

user_log_routines: colección relacionando el usuario con los datos de una rutina para los datos del progreso.

{

"*documentId*": identificador único del documento.

{

“*date*”: fecha en la que se realizó el ejercicio.

“*idRoutine*”: identificador único de la rutina.

“*idUser*”: identificador único del usuario.

“*nameRoutine*”: nombre de la rutina.

“*notes*”: anotaciones que puede apuntar el usuario del entreno.

“*percentage*”: porcentaje de la rutina completada por el usuario.

“*time*”: duración, en segundos, de la realización de la rutina.

}

}

user_favourite_exercises: colección relacionando los ejercicios favoritos con el usuario.

{

“*documentId*”: identificador único del documento.

{

“*idUser*”: identificador único del usuario.

“*idsExercises*”: vector de ids de los diferentes ejercicios que el usuario haya puesto como favoritos.

}

}

user_favourite_routines: colección relacionando las rutinas favoritas con el usuario.

{

“*documentId*”: identificador único del documento.

{

“*idUser*”: identificador único del usuario.

“*idsRoutines*”: vector de identificadores de las diferentes rutinas que el usuario haya puesto como favoritas.

}

}

user_plans: colección con los planes a los cuales se ha suscrito el usuario.

{

"*documentId*": identificador único del documento.

{

"*completed*": número identificando si el plan ya ha sido completado.

"*completedPhases*": vector de números con el número de fases que tiene el plan. Si la fase ha sido completada el valor que existe en la posición según el orden de la fase dentro del plan pasará a 1, es decir, si completo la fase 4 la posición del vector 4 pasará a valer 1, por defecto se inicializan todos los valores a 0.

"*idPlan*": identificador único del plan.

"*idUser*": identificador único del usuario.

}

}

users_profile: colección con los datos que almacenamos del usuario.

{

"*documentId*": identificador único del documento.

{

"*birthdate*": texto con la fecha de nacimiento del usuario.

"*city*": ciudad donde vive el usuario.

"*country*": país donde vive el usuario.

"*gender*": género con el cual se identifica el usuario. Puede ser masculino, femenino u otro.

"*height*": altura del usuario.

"*idUser*": identificador único del usuario.

"*image*": URL de Firebase Storage con la imagen que sube el usuario.

"*objective*": objetivo que quiere lograr el usuario utilizando la aplicación.

"*suscriber*": número para saber si el usuario es *premium* o no.

"*username*": nombre del usuario.

"*weight*": peso del usuario.

}

}

Capítulo 5

Tecnologías utilizadas

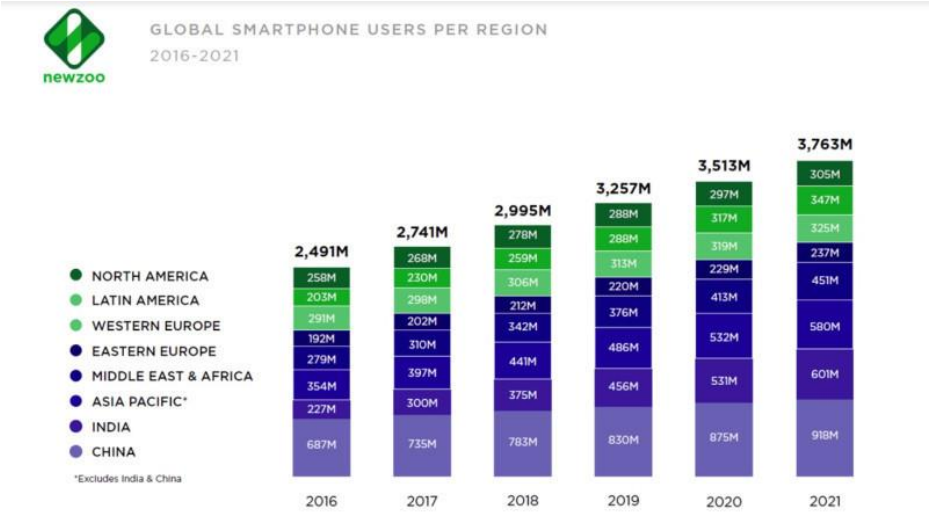
En el siguiente capítulo se comentan las diferentes tecnologías que finalmente han sido utilizadas en la aplicación haciendo en algunas de ellas comparaciones con otras herramientas de corte similar.

5.1. Android

Ilustración 20. Logotipo de Android

La tecnología más importante utilizada ha sido Android[15], cuyo logotipo puede apreciarse en la Ilustración 20. Por el ámbito que desea cubrir la aplicación es obvio que debía ser una aplicación móvil porque la calistenia, aunque puede hacerse en casa, se practica normalmente en exteriores y actualmente un gran porcentaje de la población utiliza *smartphones.* Y según la Gráfica 27, desarrollada por Newzoo[16], el uso de estos dispositivos sigue en aumento.

[15] https://www.android.com/intl/es_es/
[16] https://newzoo.com/



Gráfica 27. Usuarios de smartphone

Además, dentro de este sector de usuarios, la gran mayoría son usuarios de Android como se puede observar en la Ilustración 21 sacada de la web de StatCounter[17].

Ilustración 21. Usuarios según SO en smartphones a junio de 2020

La barrera que proporciona Apple para desarrollar en iOS ha sido lo bastante importante en este momento actual para que el proyecto se decantase por no realizar la aplicación en ese sistema, aunque en un futuro se tiene la intención de sacar la aplicación en este sistema operativo.

Ser desarrollador Android cuesta 25$, pago único, y es para toda la vida. Para ser desarrollador iOS se tienen que pagar 99$ de membresía al año y disponer de un dispositivo Mac y, actualmente, se carece de ellos.

Debido a la intención de tener a los usuarios de iOS en un futuro se estuvo barajando la posibilidad de utilizar Flutter.

[17] https://gs.statcounter.com/os-market-share/mobile/worldwide

Ilustración 22. Logotipo de Flutter

Flutter[18] es un framework de código abierto de desarrollo de aplicaciones móviles creado por Google, la Ilustración 22 representa su logotipo. Con este framework se hubiese podido utilizar el mismo código en Android, iOS y Web, pero tras mucho pensar y, hablar con el tutor, se desechó está idea porque se observó que en cuanto a rendimiento las aplicaciones nativas son más eficientes y que en desarrollos largos las empresas prefieren desarrollar dos aplicaciones nativas, una para cada sistema operativo, en vez de utilizar este tipo de frameworks porque se van agregando nuevas funcionalidades progresivamente.

Por último, otro punto por el cual se decantó el proyecto por realizarse en Android nativo fue que tanto mi compañero como yo habíamos cursado la optativa de desarrollo de aplicaciones para dispositivos móviles en la cual habíamos visto Android y ya conocíamos parte de la tecnología.

5.2. Firebase

Ilustración 23. Logotipo de Firebase

Firebase[19] es una plataforma en la nube actualmente dirigida por Google para desarrollar tanto aplicaciones web como aplicaciones móviles, la Ilustración 23 representa su logotipo.

Esta herramienta proporciona muchas ventajas en el ámbito del desarrollo de cualquier tipo de aplicación. En los siguientes apartados se trata en profundidad las características proporcionadas por Firebase que han sido utilizadas en el proyecto presentando en este apartado funcionalidades alternativas.

[18] https://flutter.dev/
[19] https://firebase.google.com/?hl=es

Por poner un ejemplo, tiene una funcionalidad llamada Admob con la cual se puede ganar dinero en la aplicación mediante anuncios y publicidades. Se descartó esta idea porque los anuncios en móvil acaban siendo demasiado molestos y se puede llegar a perder una buena cantidad de usuarios pese a poder ganar dinero.

Otras funcionalidades interesantes de Firebase que no han sido utilizadas en esta aplicación pueden ser:

- Cloud Functions: es una herramienta que permite reaccionar a cualquier cambio o petición de cualquier usuario de la aplicación mediante una serie de funciones que se pueden programar en la nube.
- Test Lab: es una herramienta que sirve para probar la aplicación en diferentes dispositivos físicos para garantizar el correcto funcionamiento en todos ellos. Para ello dispone de una IA que se encarga de realizar pruebas automáticas recorriendo la aplicación y descubriendo sus funcionalidades sin necesidad de escribir código adicional.
- Remote Config: es una herramienta que sirve para poder realizar cambios en las aplicaciones sin necesidad de sacar una nueva versión, es decir, si existe un valor utilizado en la app que es obtenido de la base de datos de Remote Config se puede modificar directamente este valor aquí. Incluso se puede segmentar ese valor para que solo llegue a un número determinado de usuarios. Esto me enlaza con la siguiente funcionalidad.
- A/B Testing: esta herramienta trabaja con Remote Config, Analytics y Cloud Messaging permitiendo sacar dos tipos de versiones para un grupo de usuarios, obteniendo retroalimentación y decidiendo con que versión te puedes quedar.
- Predictions: esta herramienta ofrece la posibilidad de predecir mediante una IA, y los datos de Analytics, el comportamiento de los usuarios. Con ella se puede detectar, por ejemplo, si un usuario interacciona más con las notificaciones *push* que llegan a la aplicación. Si se observa que no interacciona podemos cancelar esas notificaciones para esos usuarios mejorando así su relación con la aplicación o incluso detectar qué usuarios pueden llegar a realizar alguna compra o quienes no van a hacer nunca ninguna y mostrar algún tipo de anuncios para este segundo clúster.

Todas estas funcionalidades y las que se ven en los siguientes apartados, no existen en otras plataformas de bases de datos NoSQL como puede ser MongoDB.

Ilustración 24. Logotipo de MongoDB

MongoDB[20] es una base de datos distribuida basada en documentos, en otras palabras, NoSQL, la Ilustración 24 representa su logotipo. Además, tiene una consola web, que al igual que Firebase, puedes controlar los accesos a las distintas colecciones de una aplicación, además de crearlas y editarlas. Como cualquier base de datos NoSQL la estructura es la misma, dentro de las colecciones se encuentran los documentos y dentro de estos los campos que serían los atributos.

Pero MongoDB simplemente proporciona eso. Está enfocada completamente a la parte de almacenamiento, por tanto, no es una solución tan completa como si lo es Firebase. Se podría resumir como que MongoDB es una tecnología mientras que Firebase es una plataforma.

A continuación, se pasa a detallar las características y funcionalidades que proporciona Firebase que han sido utilizadas en la aplicación de AllThenics.

5.2.1. Firebase Cloud Firestore

Ilustración 25. Logotipo de Firebase: Cloud Firestore

Cloud Firestore, cuyo logotipo puede apreciarse en la Ilustración 25, es una característica de Firebase que sirve como base de datos de documentos NoSQL. Esta tecnología permite almacenar, sincronizar y consultar datos a escala global.

Igualmente, con Cloud Firestore los usuarios pueden acceder y realizar cambios en los datos de la aplicación en cualquier momento sin necesidad de conexión a internet, una vez accedido a los datos se guardan internamente en caché. Por tanto, no se

[20] https://www.mongodb.com/es

necesita una base de datos local y una externa porque ya se encarga esta tecnología de hacer ambas cosas. Al realizar un cambio en la base de datos, si el usuario no tuviese internet, se almacena localmente y al volver a tener conexión se realiza el cambio en el servidor de la nube.

5.2.3. Firebase Cloud Storage

Ilustración 26. Logotipo de Firebase: Cloud Storage

Cloud Storage, en la Ilustración 26 se puede ver representado su logotipo, es una característica de Firebase la cual permite almacenar imágenes u otros documentos los cuales se pueden ir almacenando mediante carpetas como si fuese una estructura del propio sistema operativo, pero para cualquier aplicación.

También sirve para proteger la privacidad y seguridad de esos documentos ya que existen una serie de reglas que pueden ser aplicadas sobre las carpetas y archivos para que solo puedan acceder, leer y/o escribir determinados usuarios.

5.2.4. Firebase: Authentication

Ilustración 27. Logotipo de Firebase: Authentication

Firebase Authentication, el logotipo de la cual se representa en la Ilustración 27, es una característica de Firebase que permite realizar de forma muy sencilla la autenticación de los usuarios de la aplicación. Se permite registros de diferentes formas como con la cuenta de Google o por correo electrónico, además de proporcionar una interfaz de usuario para la autenticación, proporciona un id único por usuario que sirve para identificarlo y administrar el acceso a las diferentes partes de la aplicación.

5.2.5. Firebase: Cloud Messaging

Ilustración 28. Logotipo de Firebase: Cloud Messaging

Cloud Messaging, cuyo logotipo puede apreciarse en la Ilustración 28, es una característica de Firebase que permite enviar mensajes como notificaciones *push* a los usuarios.

Con esta tecnología se puede suscribir a los usuarios a una serie de canales pudiendo hacer distinciones entre usuarios para que lleguen unos mensajes a un colectivo y otros mensajes a otros, por ejemplo, si son usuarios *premium* o si son usuarios gratuitos.

5.2.6. Crashlytics

Ilustración 29. Logotipo de Firebase: Crashlytics

Crashlytics es una característica de Firebase que permite obtener una especie de informe detallado de cualquier excepción producida en la aplicación en cualquier dispositivo, su logotipo puede apreciarse en la Ilustración 29.

Esta tecnología al producirse una excepción genera log junto con más datos el cual puede ser visto por los desarrolladores. Este informe es bastante útil ya que indica la línea exacta de código donde el programa ha dejado de funcionar, versión de la aplicación e incluso se pueden crear claves o datos adicionales para que sean enviados en el momento del error, por ejemplo, para saber el valor de una variable en el momento del fallo.

5.3. Git

Ilustración 30. Logotipo de Git

Todo buen desarrollo de una aplicación debe tener un sistema de control de versiones y, en este caso, se ha optado por Git[21], cuyo logotipo puede apreciarse en la Ilustración 30.

Git es un software de control de versiones que sirve para mantener una lista de estados o versiones por los cuales va pasando la aplicación, en concreto cada uno de sus ficheros. Con esto se consigue mantener una cierta coordinación entre los diferentes desarrolladores del producto y hace mucho más fácil y llevadero el desarrollo software.

5.4. Google Analytics

Ilustración 31. Logotipo de Google Analytics

Google Analytics[22], cuyo logotipo puede apreciarse en la Ilustración 31, es una de las mayores plataformas de recopilación y análisis de datos para aplicaciones y webs. Con esta herramienta podemos llegar a saber cuántos usuarios tenemos en la aplicación en tiempo real, número de registros, duración de las sesiones, de donde son esos usuarios, incluso en que vista han pasado más tiempo interactuando.

Es una herramienta de lo más útil porque podemos sacar gráficas y estadísticas de muchos eventos y de diferentes índoles que servirán para poder conocer mucho mejor a los usuarios y sus interacciones con la aplicación y, con ello, mejorarla de una manera que hubiese sido imposible sin Analytics.

[21] https://git-scm.com/
[22] https://analytics.google.com/analytics/web/

5.5. Worki Process

Ilustración 32. Logotipo de Worki Process

Worki Process[23], cuyo logotipo puede apreciarse en la Ilustración 32, es una herramienta muy completa enfocada al desarrollo ágil que permite realizar un seguimiento detallado de todo el proceso de desarrollo software.

Nos ofrece una completa lista de funcionalidades, *dashboards*, tableros kanban, *workflows* de trabajo, roles para los colaboradores, mensajes internos, notificaciones y tiempo de trabajo entre otras muchas características.

Al ser una herramienta con la cual hemos trabajado en varias asignaturas de desarrollo software ya se conocía su potencial, por ello se decidió utilizarla por encima de otras más conocidas como podría ser Trello.

5.6. Google Play Developer Console

Ilustración 33. Logotipo de Google Play Console

Google Play Developer Console[24], cuyo logotipo puede apreciarse en la Ilustración 33, es una herramienta para la subida de aplicaciones al mayor mercado de apps de móvil como es la Play Store.

En esta plataforma puedes llevar un control de las versiones publicadas de una aplicación, configurar la ficha de la tienda, ver los comentarios de los usuarios y las valoraciones que hayan hecho. Además, permite lanzar las aplicaciones en diferentes ámbitos como pueden ser de forma interna, alfa, beta o *release* completa.

[23] http://www.tuneupprocess.com/
[24] https://play.google.com/apps/publish/?hl=es

Capítulo 6

Implementación

En este capítulo se trata la arquitectura utilizada en la aplicación y como se ha implementado. Además, también se detallan diferentes implementaciones más concretas de otras partes de la aplicación como pueden ser los *endpoints* definidos para la relación con la base de datos, es decir, los puntos de conexión entre la aplicación y la base de datos, el proceso de autenticación, el uso de Cloud Messaging, el uso de R8 para la ofuscación y las reglas implementadas en Firebase Storage.

6.1. Arquitectura

Cuando se refiere a la arquitectura en un entorno software se está refiriendo a las guías generales que se han seguido para interconectar todas las partes funcionales de la aplicación. En este apartado primero se detalla cómo se ha organizado globalmente la aplicación y posteriormente se estudia la organización dentro de la parte del cliente.

6.1.1. Arquitectura de la aplicación

Al principio del desarrollo cuando solo estaba la idea y se estaba buscando tecnologías se tenía bastante claro que una arquitectura cliente-servidor era lo que se necesitaba. La pregunta era cómo se iba a hacer.

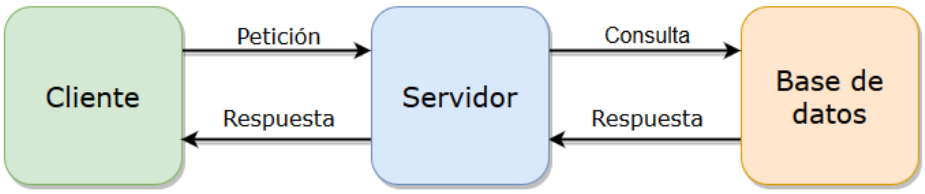


Ilustración 34. Arquitectura cliente-servidor, modelo a 3 capas

Se estuvo pensando en utilizar el típico modelo a tres capas de la Ilustración 34, es una arquitectura fácil de desarrollar, que se conocía bien y que cumplía perfectamente las necesidades de la aplicación.

Esta arquitectura consiste en tener una aplicación cliente la cual será con la que los usuarios interactúen y la única de la que serán conscientes de que existe y esta hará peticiones de consulta, inserción, actualización o borrado de los datos a un servidor.

El servidor se encarga de recoger estas peticiones y transmitírselas a la base de datos la cual responde al servidor y este responde al cliente. Básicamente el servidor es un intermediario que aísla mucha lógica de la aplicación.

Por último, la base de datos almacena todos los datos que serán necesarios para el correcto funcionamiento de la aplicación, recibiendo las consultas u operaciones del servidor y respondiéndole a este.

El problema es que el desarrollo de un servidor que maneje todas estas funcionalidades es un trabajo importante que ocupa mucho tiempo. Además, otro problema que se encontró fue con la comunicación con la base de datos. Al necesitar un servidor conectado a Internet para poder ir recibiendo las peticiones, enviárselas a la base de datos y poder devolver la respuesta se pensó en las veces que un usuario se queda sin Internet en el móvil. Entonces, la solución que se barajó fue utilizar una base de datos local de la aplicación en SQLite aparte de otra base de datos en MySQL o similar que fuese la que se comunica con el servidor.

Ilustración 35. Logotipo de SQLite

SQLite[25] es un sistema de gestión de base de datos relacional muy liviano que se integra directamente con el programa y no es necesario ningún tipo de comunicación utilizando un servidor, la Ilustración 35 representa su logotipo. Gracias a esto las bases de datos que se generan son bastante ligeras pudiendo funcionar perfectamente en dispositivos con pocas prestaciones y, por ello, este sistema es bastante utilizado en las aplicaciones Android debido a su tamaño, estabilidad y rendimiento.

[25] https://www.sqlite.org/index.html

Con esta idea de arquitectura no solo se tenía que hacer un servidor si no que se debía invertir aún más tiempo en hacer dos bases de datos. Fue entonces cuando hablando con el tutor se descubrió la herramienta de Google, Firebase y nos decantamos por esta opción.

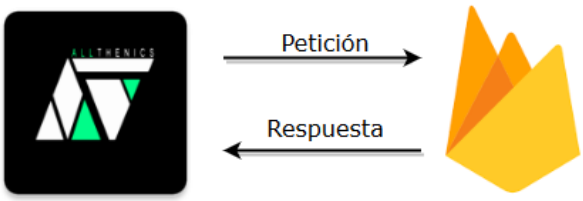


Ilustración 36. Arquitectura AllThenics

Con Firebase, como se puede observar en Ilustración 36, se deshecha la necesidad de hacer un servidor intermedio pero con posibilidad de hacerlo si fuese necesario. También se deshecha la idea de la base de datos local porque ya no es necesario gestionarla manualmente.

Al realizar una petición a Cloud Firestore, la base de datos NoSQL que proporciona Firebase, se utilizan una serie de funciones asíncronas que proporciona la propia biblioteca, en el apartado 6.2.1, se tratará más detalle.

Asimismo, se puede gestionar no solo la base de datos directamente desde el cliente si no también la autenticación y registro usando Authentication sin necesidad de crear una forma propia de almacenar *tokens* de sesión y de registrar usuarios, posteriormente en el apartado 6.2.2, se comentará la forma de hacerlo haciendo uso de la biblioteca FirebaseUI.

Además, con Storage se pueden almacenar las imágenes y documentos que se utilizan en la aplicación y guardar las URL en Firestore para poder acceder directamente.

Por último, todas estas herramientas proporcionan la opción de escribir una serie de reglas para proteger la lectura y escritura haciendo más fiable y seguros los datos almacenados sin la necesidad de administrar códigos de autorización, en la sección 6.2.5 se verán algunas de estas reglas.

6.1.2. Arquitectura del cliente

En la aplicación cliente se ha utilizado una arquitectura basada en el patrón software *model-view-viewmodel* (MVVM) haciendo uso de otra funcionalidad llamada

bindings, este tipo de arquitectura es la recomendada por Google para aplicaciones móviles [14]. En el siguiente diagrama, Diagrama 7, se puede observar cómo funciona esta arquitectura.

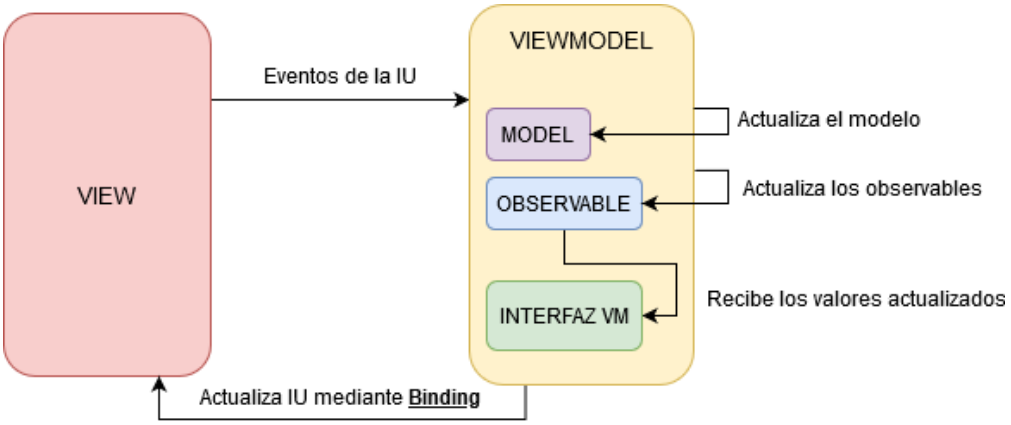


Diagrama 7. Arquitectura MVVM

La vista simplemente se encarga de notificar al *viewmodel* una interacción que haya hecho el usuario con ella. Al recibir esta notificación el *viewmodel* actualiza el modelo, el cual contiene dentro y representa los datos modificados en la IU.

Al hacer estos cambios sobre el modelo también se actualiza el observador. Este observador es un patrón de comportamiento el cual sirve para que cuando haya algún cambio de estado en uno de estos objetos, este cambio se notifique a todos los objetos dependientes.

Por último, al recibir los cambios en el observador, la interfaz viewmodel recibe esta actualización y mediante *binding* se actualizan los componentes de la vista.

El *binding* es una forma de relacionar directamente la capa de presentación con los datos obtenidos en la aplicación. Para ello, en Android se vincula un *pojo* (Plain Old Java Object), que será el contenedor de los datos que se van a mostrar en la vista, con el archivo xml el cual es la vista que va a hacer uso ese de *pojo*. Esta vinculación sería la interfaz viewmodel del diagrama. Posteriormente en el apartado 6.1.3 se explicará detalladamente cómo está implementada esta arquitectura en AllThenics.

Para que quede más claro el uso de esta arquitectura se va a comparar con otras dos arquitecturas más comunes [15] que se suelen utilizar en Android.

La primera arquitectura con la que la vamos a comparar es la *model-view-controller* (MVC), el Diagrama 8 representa su estructura.

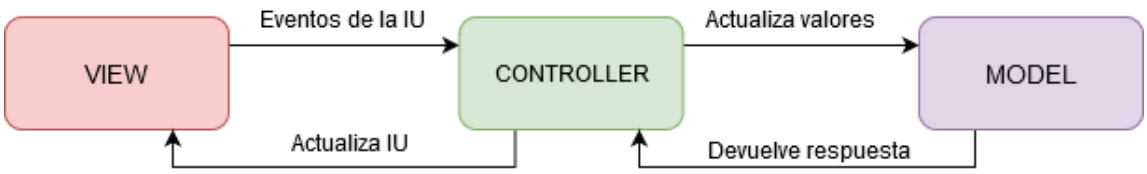


Diagrama 8. Arquitectura MVC

En esta arquitectura, la vista funciona igual que en la *model-view-viewmodel* notificando, en este caso, al controlador de un evento en ella. El controlador recibe esta notificación y modifica el modelo.

A continuación, el modelo realiza su lógica de negocio interna con los datos que le han llegado actualizados del controlador y devuelve el nuevo estado actualizado al controlador. Para finalizar, este controlador al cual le ha llegado el nuevo modelo actualizado, actualiza la interfaz de usuario.

La segunda arquitecta es la *model-view-presenter* (MVP), el Diagrama 9 representa su estructura.

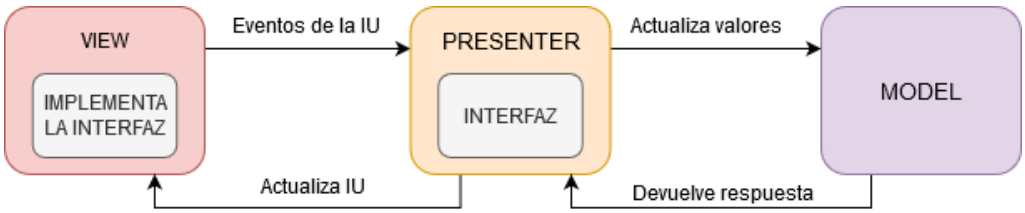


Diagrama 9. Arquitectura MVP

El flujo de la arquitectura *model-view-presenter* se asemeja a la *model-view-controller* pero con una diferencia clara, la forma de interactuar entre la vista y el controlador y la vista y el presentador.

En este caso, la vista debe implementar una interfaz que es la utiliza el presentador para actualizarla. Por ende, al recibir la actualización de los datos en el modelo, que funciona igual que en la arquitectura *model-view-controller*, el presentador hace uso de esa interfaz en la cual existen métodos, que al implementar la interfaz la vista, se utilizan estos métodos para modificarla. Con todo esto se consigue que el presentador no tenga conocimiento de las vistas que está intentando actualizar, por

tanto, no tiene dependencias y es más fácil realizar test unitarios, hecho que no pasa con el controlador que sí tiene conocimiento de la vista que actualiza.

Por último, se adjunta una tabla resumen de las tres arquitecturas, Tabla 10, basada en una realizada por Anmol Sehgal[26], un desarrollador de Android/*backend*, en la web Medium[27] que abrevia perfectamente las diferencias entre ellas.

Tabla 10. Resumen diferencias arquitecturas cliente

PATRÓN	Dependencia de APIS de Android	Complejidad de la vista	Facilidad de pruebas	Modular y SRP
MVC Controller	Alto	Baja	Difícil	No
MVP Presenter	Bajo	Baja	Fácil	Sí
MVVM Viewmodel	Bajo o sin dependencia	Media o Alta	Muy fácil	Sí

6.1.3. Implementación de la arquitectura del cliente

En este apartado se va a explicar cómo se consigue implementar esta arquitectura en una aplicación Android.

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".activities.ProfileActivity">

    <data>
        <variable
            name="userProfile"
            type="vcardona.countero.allthenics.pojos.UserProfile" />
    </data>
```

Fragmento de código 1. Binding de UserProfile

Como se puede observar en el Fragmento de código 1, para poder hacer un binding de una vista con su correspondiente *pojo* hay que envolver todo el diseño de la vista con la etiqueta *layout* y, justo después, utilizar una etiqueta *data* en la cual se pondrá una etiqueta *variable* con los atributos *name*, que será el nombre, y *type*, que será la ubicación del pojo del cual va a coger los datos el *binding* para poder actualizar correctamente la vista.

[26] https://medium.com/@mr.anmolsehgal
[27] https://medium.com/

Por último, para poder relacionar concretamente un *TextView*, que es un componente Android que enseña un texto, es decir, una especie de *label*, con el *binding* y que este sepa qué valor introducir al recibir los datos, hay que poner en el atributo que se quiera ir actualizando de la etiqueta *TextView,* el nombre que se ha puesto en el atributo *name* de la etiqueta *variable* junto con un punto y el nombre del atributo del *pojo* que se va a introducir como valor en esa etiqueta, como se puede observar en el Fragmento de código 2.

```
<TextView
    android:id="@+id/textView6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:text="@{userProfile.birthdate}"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Fragmento de código 2. Gastando binding en TextView

Todo esto se complementa en Android con un tipo de objeto llamado *LiveData,* esta clase es un contenedor de datos observable, y está optimizada para el ciclo de vida de las actividades, fragmentos y servicios, aunque para hacer uso de ella deberemos instanciar los objetos con la clase *MutableLiveData*, ya que *LiveData* es una clase abstracta.

En cada *ViewModel* se hace uso de esta clase para actualizar los valores de la vista mediante el *binding*. A continuación, en el Fragmento de código 3, puede verse un ejemplo de su uso. En este caso se actualiza la lista de todos los ejercicios.

```
// get all exercises and assigned them to the list
// and return the reference where the app will be listening for changes of the list that will be shown
public LiveData<List<Exercise>> fetchAllExercises() {
    MutableLiveData<List<Exercise>> _allExercisesList = new MutableLiveData<>();
    // fetch All exercises from Firestore
    exerciseFirestoreManager.getAllExercises(task -> {
        if (task.isSuccessful()) {
            QuerySnapshot querySnapshot = task.getResult();
            if (querySnapshot != null) {
                _allExercisesList.setValue(querySnapshot.toObjects(Exercise.class));
            }
        }
    });
    return _allExercisesList;
}
```

Fragmento de código 3. Método getAllExercises

6.2. Implementaciones concretas

A continuación, se van a tratar implementaciones hechas en la aplicación para hacer las consultas a la base de datos, es decir, los *endpoints* necesarios para la comunicación, para realizar la autenticación y registro de los usuarios, para enviar notificaciones push mediante Cloud Messaging, el uso de R8 para la ofuscación de código y las reglas de seguridad utilizadas en Firebase.

6.2.1. Relacionando la aplicación cliente con la base de datos

En este apartado se expone cómo se ha conseguido relacionar la base de datos con la aplicación cliente. Para ello habrá que poner en las dependencias del archivo *build.gradle* a nivel de aplicación las bibliotecas que vayamos utilizar, en este caso, las de Fragmento de código 4.

```
//firebase
implementation 'com.google.firebase:firebase-core:17.4.4'
implementation 'com.google.firebase:firebase-firestore:21.5.0'
implementation 'com.google.firebase:firebase-storage:19.1.1'
implementation 'com.google.firebase:firebase-auth:19.3.2'
```

Fragmento de código 4. Bibliotecas de Firebase

Cada una de estas bibliotecas representa diferentes partes necesarias para tratar con la base de datos. La biblioteca Core representa el núcleo de los objetos necesarios para enlazar la aplicación con Firebase, la biblioteca Firestore es concretamente la base de datos en sí donde se van a almacenar todos los datos, la biblioteca Storage representa los archivos que se van a almacenar, en este caso, imágenes y, por último, la biblioteca Auth es la que se encarga de registrar y autenticar a los usuarios.

Lo primero que debe hacerse para comenzar esta relación entre las dos partes es crear un *pojo* que será el objeto al cual se convertirán los valores obtenidos de la base de datos. Las claves del documento serán los atributos de la clase y los valores serán los valores de cada atributo. En el Fragmento de código 5, tenemos un ejemplo de *pojo*, en este caso, de rutinas.

```java
public class Routine implements Serializable {

    public static String ROUTINE_TYPE_CIRCUIT = "Circuito";

    // this @ allows firebase to link automatically the documentID
    @DocumentId
    private String documentId;
    private int activityTime;
    private int breakTime;
    private int difficulty;
    private int duration;
    private String exerciseType;
    private int laps;
    private List<String> muscles;
    private String name;
    private String objective;
    private String routineType;

    public Routine() {
    }

    public Routine(String documentId, int activityTime, int breakTime, int difficulty,
                   int duration, String exerciseType, int laps, List<String> muscles, String name,
                   String objective, String routineType) {
        this.documentId = documentId;
        this.activityTime = activityTime;
        this.breakTime = breakTime;
        this.difficulty = difficulty;
        this.duration = duration;
        this.exerciseType = exerciseType;
        this.laps = laps;
        this.muscles = muscles;
        this.name = name;
        this.objective = objective;
        this.routineType = routineType;
    }

    public String getDocumentId() { return documentId; }
```

Fragmento de código 5. Pojo de rutinas

Cabe destacar el uso de la anotación *@DocumentId* con la cual se consigue vincular directamente el identificador único del documento de la base de datos con el *pojo* una vez se obtienen los datos.

Una vez creado el *pojo* se va a necesitar un *contract*, esta es una clase que sirve para conocer simplemente el nombre de la colección y de los campos que existen en el documento de la base de datos. Para seguir con el mismo ejemplo, en el Fragmento de código 6 se visualiza el *contract* de rutinas.

```java
public class RoutineContract {
    // Root collection name
    public static final String COLLECTION_NAME = "routines";

    // Document ID
    public static final String DOCUMENT_ID = "";

    // Document field names
    public static final String FIELD_ACTIVITY_TIME = "activityTime";
    public static final String FIELD_BREAK_TIME = "breakTime";
    public static final String FIELD_DIFFICULTY = "difficulty";
    public static final String FIELD_DURATION = "duration";
    public static final String FIELD_EXERCISE_TYPE = "exerciseType";
    public static final String FIELD_LAPS = "laps";
    public static final String FIELD_MUSCLES = "muscles";
    public static final String FIELD_NAME = "name";
    public static final String FIELD_OBJECTIVE = "objective";
    public static final String FIELD_ROUTINE_TYPE = "routineType";

    // To prevent from accidentally instantiating the contract class, constructor private
    private RoutineContract() {
    }
}
```

Fragmento de código 6. Contract de rutinas

Para finalizar el enlace entre la base de datos y la aplicación falta una última clase, esta clase es un *manager* que se encarga de realizar las peticiones a la base de datos mediante los objetos *FirebaseFirestore* y *CollectionReference* que proporciona la biblioteca de Cloud Firestore.

En esta clase se hace el CRUD del objeto en la base de datos y las diferentes funciones que sean necesarias en la aplicación, en resumen, aquí se hacen todos los *endpoints*. Por ejemplo, en el caso de las rutinas se tienen las funciones *createDocument*() que sirve para crear una rutina, *getAllRoutines*() que devuelve todas las rutinas, *updateRoutine*() que sirve para actualizar una rutina, *deleteRoutine*() para eliminarla y *getRoutine*() para obtener una rutina en concreto.

Con los diferentes managers se ha hecho uso del patrón de diseño *singleton* el cual sirve para tener una única instancia de este objeto. Para ejemplificar todas estas clases *manager* en el siguiente fragmento de código, Fragmento de código 7, se puede observar el *manager* de rutinas.

```java
// Singleton for making operation with ROUTINES between frontend and backend
public class RoutineFirestoreManager {
    // object singleton, just one instance
    private static RoutineFirestoreManager routineFirestoreManager;
    // firebase object that is a singleton too, comes by default
    private FirebaseFirestore firebaseFirestore;
    // this object give access to a collection from firebase, for that reason is used with the previous
    private CollectionReference routinesCollectionReference;

    private RoutineFirestoreManager() {
        firebaseFirestore = FirebaseFirestore.getInstance();
        routinesCollectionReference = firebaseFirestore.collection(COLLECTION_NAME);
    }

    public static RoutineFirestoreManager newInstance() {
        if (routineFirestoreManager == null) {
            routineFirestoreManager = new RoutineFirestoreManager();
        }
        return routineFirestoreManager;
    }

    public void createDocument(Routine routine) { routinesCollectionReference.add(routine); }

    public void getAllRoutines(OnCompleteListener<QuerySnapshot> onCompleteListener) {
        routinesCollectionReference.get().addOnCompleteListener(onCompleteListener);
    }

    public void updateRoutine(Routine routine) {
        String documentId = routine.getDocumentId();
        DocumentReference documentReference = routinesCollectionReference.document(documentId);
        documentReference.set(routine);
    }

    public void deleteRoutine(String documentId) {
        DocumentReference documentReference = routinesCollectionReference.document(documentId);
        documentReference.delete();
    }

    public void getRoutine(String documentId, OnSuccessListener<DocumentSnapshot> onSuccessListener) {
        routinesCollectionReference.document(documentId).get().addOnSuccessListener(onSuccessListener);
    }
```

Fragmento de código 7. Manager de rutinas

Una de las ventajas de todas las funciones utilizadas sobre la clase *CollectionReference* es que por defecto todas las llamadas son asíncronas en segundo plano haciendo posible que el usuario pueda interactuar con la aplicación sin bloquear el hilo principal de la aplicación.

A continuación, se va a desarrollar la implementación del *listener* que es necesario pasar en el método *getAllRoutines*() ya que en el *manager* no se hace la implementación del resultado obtenido de la base de datos simplemente se realiza la petición.

```
routineFirestoreManager.getAllRoutines(task -> {
    if (task.isSuccessful()) {
        QuerySnapshot querySnapshot = task.getResult();
        if (querySnapshot != null) {
            allRoutinesList = querySnapshot.toObjects(Routine.class);
            // because this method it is called the first and there is no filter/search
            // those list should be all exercises also
            routinesListShow = allRoutinesList;
            filterRoutines = allRoutinesList;
            searchRoutines = allRoutinesList;

            // default title sorting
            Collections.sort(routinesListShow, Routine.nameSort);
            _routinesListMutableLiveData.setValue(routinesListShow);
        }
        binding.progressBarCircular.setVisibility(View.GONE);
    }
});
```

Fragmento de código 8. Listener getAllRoutines

Como se observa en el Fragmento de código 8, se realiza una función lambda implementando el *listener* cuando se lleva a cabo la llamada a *getAllRoutines()* del *manager* de rutinas. Si la petición se ha realizado satisfactoriamente se crea un objeto *QuerySnapshot,* este objeto lo proporciona la biblioteca de Firestore y sirve para guardar el resultado de la petición, y se iguala al resultado de la *task*.

Si no es nulo, se puede aplicar sobre el objeto *QuerySnapshot* un *.toObjects()* que sirve para realizar una conversión directamente del resultado de la petición a un objeto de los creados en la aplicación, en este caso al *pojo* de rutinas, y con esto ya se puede trabajar con los datos obtenidos.

Básicamente todas las llamadas se hacen de una forma parecida, añadiendo el *listener* que corresponda y cambiando sobre el objeto *collectionReference* lo que se quiera hacer. Por ejemplo, en el Fragmento de código 7 se ha podido ver que en *getAllRoutines*() se aplica un .*get*() directamente y en *getRoutine*() primero se realiza un .*document*() pasándole el identificador del documento para buscar en concreto una rutina que coincida con el id introducido y después el .*get*() cambiando el tipo de *listener*.

Por último y para concluir este apartado, se va a comentar la función *getUserPlansCompleted*() del *manager* de los planes del usuario porque en este caso existe un detalle diferente y es que se tienen condiciones.

```
public void getUserPlansCompleted(String idUser, OnCompleteListener<QuerySnapshot> onCompleteListener) {
    usersPlansCollectionReference.whereEqualTo(FIELD_ID_USER, idUser)
            .whereEqualTo(FIELD_COMPLETED,  value: 1)
            .get().addOnCompleteListener(onCompleteListener);
}
```

Fragmento de código 9. Método getUserPlansCompleted

En el Fragmento de código 9 se observa cómo hacer peticiones a la base de datos poniendo condiciones. Con esta función se consigue obtener los planes completados del usuario correspondiente. Para conseguirlo, a la *CollectionReference* se llama a una función, en este caso, *.whereEqualTo*() porque se quiere que sea igual al valor identificador del usuario que pasamos en la función y además, se puede concatenar más de una condición. En este caso, hemos concatenado dos *.whereEqualTo*(), uno para el identificador del usuario y otro para que el plan esté completado.

6.2.2. Autenticación

Para la autenticación en AllThenics se ha hecho uso de la biblioteca FirebaseUI. Esta es una biblioteca de código abierto creada a partir del SDK de Firebase Authentication que proporciona flujos directos de IU para usar en las aplicaciones [16], en la Ilustración 37 se ve un ejemplo visual de su uso.

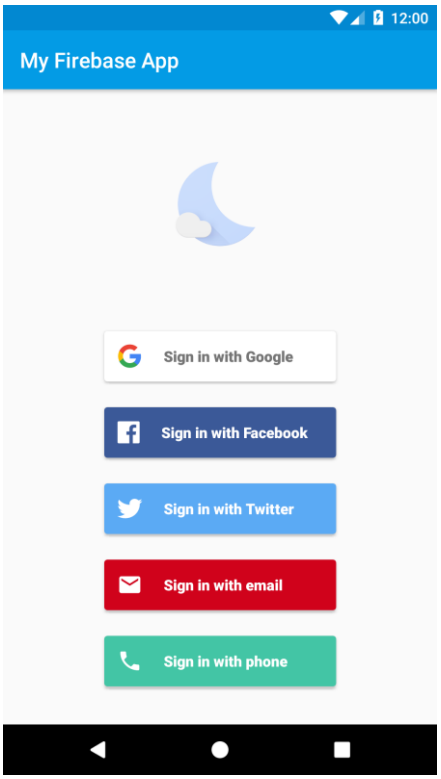


Ilustración 37. Ejemplo de la interfaz de FirebaseUI

Esta biblioteca incluye por defecto muchas funcionalidades que sin ellas se harían más complejas de implementar en la aplicación. Algunas de ellas son:

- **Diversos proveedores de autenticación**: con poco más de un par de líneas de código puedes incluir en la aplicación *login* vía cuenta de Twitter, Facebook, Google, correo electrónico, teléfono y muchos más.
- **Temas personalizados**: pese a que FirebaseUI proporciona una estructura fija en cuanto a la interfaz, puede ser adaptada a la aplicación o, como es de código abierto, hacer una propia bifurcación y personalizarla libremente según las necesidades.
- **Smart Lock**: viene implementada la función Smart Lock la cual sirve que al registrarse con Google se guardan los datos en la propia cuenta y se puede acceder con el mismo usuario directamente sin necesidad de autenticar desde dispositivos diferentes.

Para hacer uso de la biblioteca se va a tener que configurar la aplicación y la consola de Firebase. Primero, se debe añadir en las dependencias del *build.gradle* a nivel de aplicación y, a continuación, en la consola de Firebase en la pestaña de *Sign-in method* de Authentication habilitar los proveedores que se van a utilizar, en este caso correo electrónico y Google como se puede observar en la Ilustración 38.

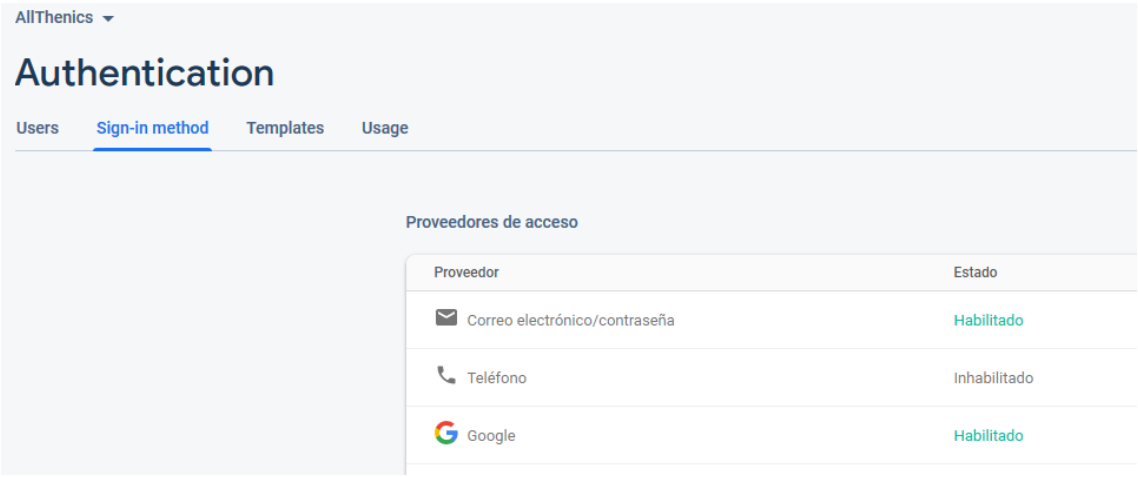


Ilustración 38. Proveedores habilitados en AllThenics

Al habilitar la autenticación con Google se debe especificar la huella digital SHA-1 de la aplicación en la consola de Firebase en la pestaña Configuración. Esta huella sirve para identificar unívocamente la aplicación y poder hacer uso de las API de Google de forma segura.

Una vez configurado, se va a pasar a hacer uso de la biblioteca en la aplicación, pero para ello se debe explicar todo el proceso de autenticación.

Al iniciar la aplicación se entra en RegisterActivity que no tiene vista asociada. Aquí se comprueba si existe ya un usuario autenticado y si existen las preferencias, si ese es el caso se lanza la actividad principal. En el caso de que haya un usuario, pero no tenga las preferencias creadas se mira si en la base de datos existe el usuario y si existe se crean las preferencias, sino se redirige al usuario para que se registre otra vez. Por último, si no se cumplen ninguna de las dos condiciones se muestra la interfaz de FirebaseUI.

```
SharedPreferences sharedPreferences = this.getSharedPreferences("Preferencias
String name = sharedPreferences.getString("Nombre", defValue: "");
// if user is signed in NavigationDrawerActivity will appear, else will apper
if (auth.getCurrentUser() != null && !name.equals("")) {
    startActivity(new Intent( packageContext: this, NavigationDrawerActivity.cla
    finish();
} else if (auth.getCurrentUser() != null) {
    lookInDatabaseIfUserExists();
} else {
    ActionCodeSettings actionCodeSettings = ActionCodeSettings.newBuilder()
            // URL you want to redirect back to. The domain (www.example.com)
            // URL must be whitelisted in the Firebase Console.
            .setUrl("https://google.com")
            // This must be true
            .setHandleCodeInApp(true)
            .setAndroidPackageName( s: "vcardona.countero.allthenics",
                    b: true,
                    s1: null)
            .build();

    // list of providers that will appear on the Auth UI layout
    List<AuthUI.IdpConfig> providers = Arrays.asList(
            new AuthUI.IdpConfig.GoogleBuilder().build(),
            new AuthUI.IdpConfig.EmailBuilder().enableEmailLinkSignIn()
                    .setActionCodeSettings(actionCodeSettings).build());

    startActivityForResult(
            AuthUI.getInstance() AuthUI
                    .createSignInIntentBuilder() AuthUI.SignInIntentBuilder
                    .setAvailableProviders(providers)
                    .setTheme(R.style.AuthUIStyle)
                    .setLogo(R.mipmap.ic_launcher)
                    .setTosAndPrivacyPolicyUrls(
                            tosUrl: "https://example.com/terms.html",
                            privacyPolicyUrl: "https://example.com/privacy.html")
                    .build(),
            RC_SIGN_IN);
}
```

Fragmento de código 10. Uso en RegisterActivity

Para hacer uso de FirebaseUI, en el Fragmento de código 10 se observa que debe crearse un *ActionCodeSettings*, un objeto que proporciona la biblioteca y sirve para indicar las opciones en caso de que usemos el registro/autenticación con correo electrónico. A continuación, se crea una lista con los proveedores que se van a utilizar,

poniendo las opciones que hemos creado anteriormente en el proveedor de correo, y se lanza la FirebaseUI como *activityForResult.* Cuando se lanza una actividad así indicamos que de la actividad lanzada estamos esperando un resultado. El resultado en este caso es el correcto registro en la aplicación mediante la biblioteca en cuyo caso una vez realizado, se lanza una actividad llamada EditProfileActivity que sirve para introducir obligatoriamente los datos del usuario en la aplicación.

Con todo esto ya existe un registro perfecto con Google, pero cabe destacar que para poder hacer el registro por correo aún quedan unos pequeños detalles por hacer.

Primeramente, en el archivo *manifest* hay que indicar una actividad que será lanzada al hacer clic en el hiperenlace que llegará al correo y no puede ser la misma actividad desde la cual se lanza FirebaseUI, esto sirve para poder verificar el correo correctamente. En este caso, en el Fragmento de código 11, se puede comprobar que se ha decidido poner en EditProfileActivity.

```
<activity
    android:name=".activities.EditProfileActivity"
    android:configChanges="orientation"
    android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />

        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data
            android:host="allthenics-c2e3b.firebaseapp.com"
            android:scheme="https" />
    </intent-filter>
</activity>
```

Fragmento de código 11. Manifest de la actividad EditProfile

Al hacer el *intent-filter* con esos valores en las etiquetas *action* y *category* se está haciendo posible que desde el hiperenlace que sale en el correo se lance implícitamente esta actividad. Además, estamos obligando con la etiqueta *data* a que el host sea el servidor de Firebase de AllThenics y el esquema tipo https.

Para finalizar la implementación del registro por correo electrónico, en la actividad que se ha definido para ser lanzada desde el hiperenlace del correo, en este caso EditProfileActivity, se va a tener que iniciar otra *activityForResult* de FirebaseUI con esto se consigue registrar completamente el usuario en Firebase. El Fragmento de código 12 proporciona este trozo de implementación.

```
// thanks to that we know if user has clicked on the email
String emailLink = getIntent().getData().toString();

if (AuthUI.canHandleIntent(getIntent())) {
    if (getIntent().getExtras() == null) {
        return;
    }
    startActivityForResult(
            AuthUI.getInstance() AuthUI
                    .createSignInIntentBuilder() AuthUI.SignInIntentBuilder
                    .setEmailLink(emailLink)
                    .setAvailableProviders(providers)
                    .build(),
            RC_SIGN_IN);
}
```

Fragmento de código 12. Registro del usuario en Firebase por correo

Por último, comentar que este correo de verificación es completamente editable desde la pestaña plantillas de Firebase Authentication, la Ilustración 39 muestra la plantilla de correo utilizada en AllThenics.

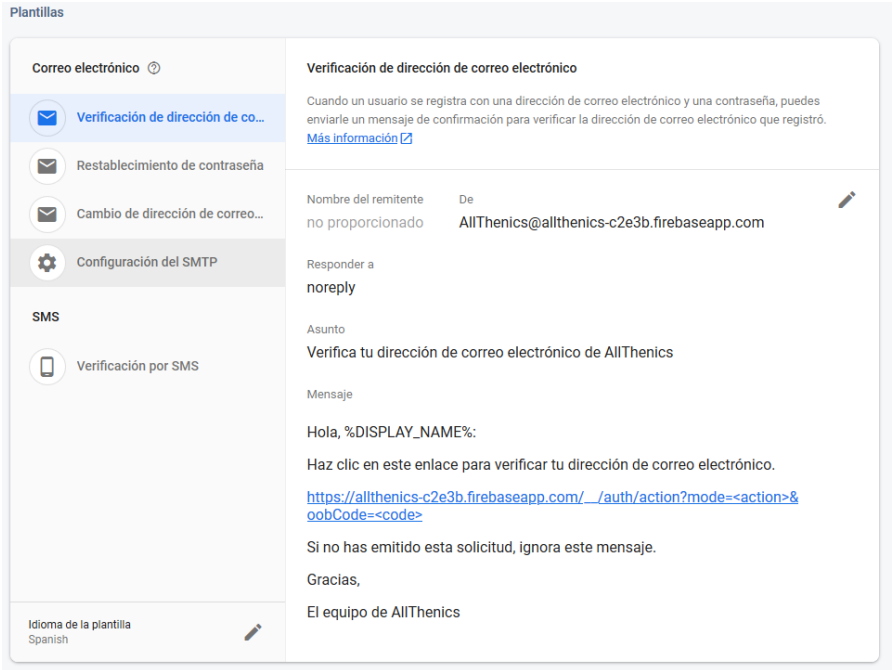


Ilustración 39. Plantilla de correo de verificación

6.2.3. Notificaciones push con Cloud Messaging

Cloud Messaging, como se ha explicado en el apartado 5, es una características de Firebase que permite a los desarrolladores enviar notificaciones *push* a los diferentes usuarios de las aplicaciones pudiendo hacer distinción entre ellos.

Para poder hacer uso de esta funcionalidad primero se debe implementar su dependencia en el archivo *build.gradle* a nivel de aplicación.

Una vez cargada en la aplicación hay que crear un servicio de Android. Un servicio es un componente que puede realizar operaciones en segundo plano sin proporcionar una interfaz. Haciendo esto se consigue dejar la aplicación escuchando pese a no estar en primer plano pudiendo con ello recibir las notificaciones que los desarrolladores quieran enviar.

Este servicio debe extender de la clase *FirebaseMessaginService* y, como todo servicio, debe ser declarado en el archivo *manifest*. Esta declaración debe llevar una etiqueta *exported* con valor a *false*, ya que al introducir una etiqueta *intent-filter* el valor por defecto de *exported* sería *true* y si se quedase con este valor se estaría dejando la posibilidad a otras aplicaciones de hacer uso de este servicio. Esto sería un gran problema de seguridad si no se hubiese controlado.

El *intent-filter* tendrá una etiqueta *action* con el valor *name* a *com.google.firebase.MESSAGING_EVENT* para indicar que ese evento será el que va a recibir el servicio. Todo esto se puede observar en el Fragmento de código 13.

```
<!-- service for cloud messaging function (push notifications)-->
<service
    android:name=".utils.MyFirebaseMessagingService"
    android:exported="false">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Fragmento de código 13. Declaración del servicio de Cloud Messaging

Con todo lo anteriormente comentado, se tendría el servicio funcionando en segundo plano y solo saldría la notificación a los usuarios que no tuviesen en ese momento la aplicación activa. Entonces, en el servicio que se ha creado hay que controlar el mensaje entrante para que salga la notificación cuando la aplicación esté activa.

Para ello se debe sobrescribir los métodos *onNewToken*() en el cual solo habrá una llamada a *super* y *onMessageReceived*() que procesará el mensaje recibido y lo mostrará como notificación. En este último método, aparte de llamar a *super*, se debe crear un *handler* con un *looper*. Un *Handler* es un componente que sirve para procesar y enviar un mensaje recibido y un *Looper* es un componente que permite a un hilo de ejecución crear una cola de mensajes.

En el Fragmento de código 14 se observa como el *looper* es preparado, a continuación, en el *handler* se crea la notificación que saldrá por pantalla. Para acceder a los valores del mensaje se usa el objeto *RemoteMessage* dado en el propio método y una vez creada la notificación, notificamos al manager encargado de gestionar las notificaciones que se pretende mostrarla y al final del todo se llama al método *loop*() de *looper* para que empiece a gestionar la cola de mensajes y vaya mostrándolos uno por uno, parándose al finalizar la cola.

```
@Override
public void onMessageReceived(@NonNull RemoteMessage remoteMessage) {
    super.onMessageReceived(remoteMessage);
    Looper.prepare();
    Handler handler = new Handler();
    handler.post(() -> {
        NotificationCompat.Builder builder = new NotificationCompat.Builder( context: this, NavigationDrawerActivity.CHANNEL_ID)
                .setSmallIcon(R.mipmap.ic_launcher)
                .setContentTitle(remoteMessage.getNotification().getTitle())
                .setContentText(remoteMessage.getNotification().getBody())
                .setPriority(NotificationCompat.PRIORITY_DEFAULT);
        NotificationManagerCompat notificationManagerCompat = NotificationManagerCompat.from(this);
        notificationManagerCompat.notify( id: 1, builder.build());
    });
    Looper.loop();
}
```

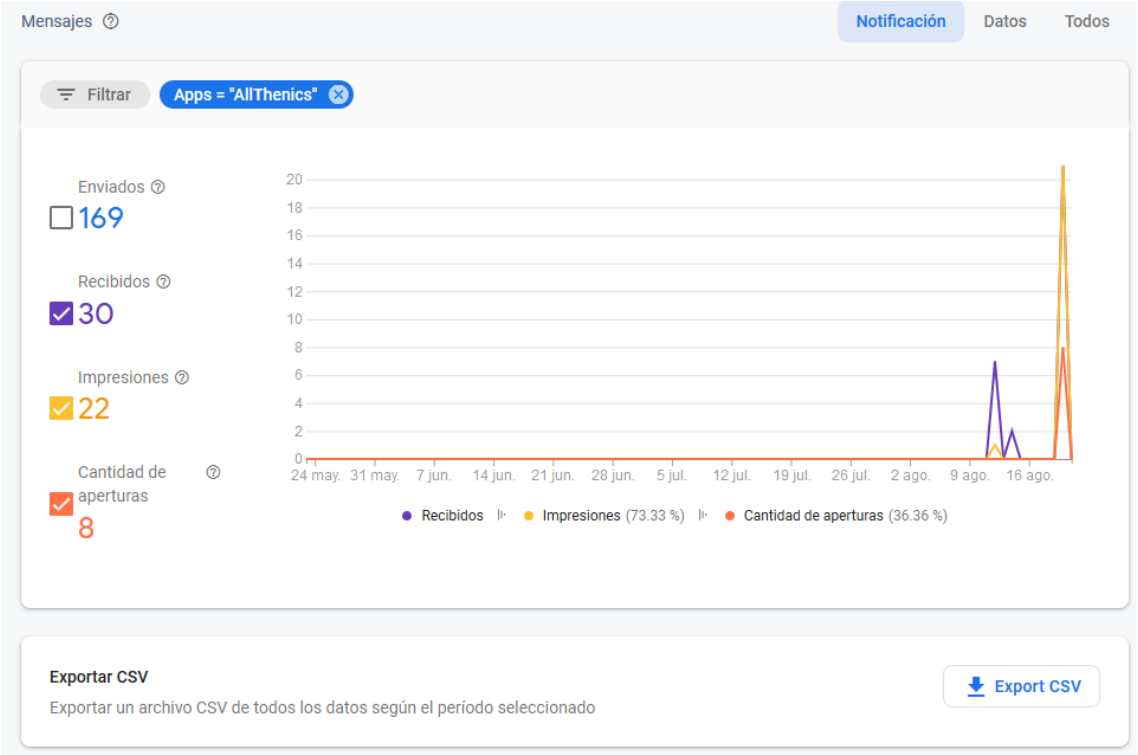
Fragmento de código 14. Servicio MyFirebaseMessasingService

Falta explicar un pequeño detalle que es el *CHANNEL_ID* cuando creamos la notificación en el *handler*. *CHANNEL_ID* es simplemente un atributo de clase estático de tipo String para indicar el identificador del canal. Este campo a partir de la API 26 de Android es obligatorio y no solo eso, sino que para hacer uso de él se debe registrar en la aplicación un canal el cual se encargará de mostrar estas notificaciones. En este caso se ha decidido crear el canal en la actividad principal de la aplicación con un valor de importancia por defecto.

Una vez implementado todo esto solo hace falta ir a la pestaña Cloud Messaging de la plataforma Firebase para crear la notificación que se pretenda enviar. Esta notificación puede ser enviada a un grupo concreto de usuarios, que deberán haber sido suscritos mediante la aplicación, o a todos los usuarios. Esta funcionalidad de suscripción será implementada en un futuro para poder diferenciar entre usuarios

premium o no consiguiendo así una forma de notificarles que se les va a acabar pronto la suscripción o que ha salido algún tipo de plan nuevo. Además, con Cloud Messaging se puede programar una hora y fecha concreta en la cual lanzar la notificación o directamente mandarla en el momento actual.

Por último, destacar como Google Analytics se complementa con esta herramienta proporcionando datos relacionados con las notificaciones enviadas en diferentes lapsos de tiempo incluso con una opción para poder exportar estos datos a un archivo csv y poder manejarlos con comodidad. La Gráfica 28 representa una de estas gráficas con la opción de exportar.



Gráfica 28. Gráfica de Cloud Messaging

6.2.4. Ofuscación con R8

Una de las herramientas más útiles integradas en Android Studio es R8, una herramienta basada en Proguard [17]. Su funcionalidad consiste en quitar el código y los recursos que no se utilizan, reduciendo considerablemente el tamaño de la aplicación y optimizándola. Además, ofusca el código considerablemente añadiendo una buena capa de seguridad a la aplicación por si esta es desempaquetada, ya que al ofuscarla se modifican nombres de clases, métodos y atributos consiguiendo la misma funcionalidad, pero haciendo el código bastante ilegible.

Para activar esta funcionalidad, que no viene por defecto activada, se debe introducir en el archivo *build.gradle* a nivel de aplicación en la *build* de tipo *release* *minifyEnabled* a *true*. Es recomendable activarlo solo cuando se pretenda compilar la aplicación final. En el Fragmento de código 15 se puede observar esta activación.

```
buildTypes {
    release {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
    }
}
```

Fragmento de código 15. Activando R8

También cabe destacar la siguiente línea en el fragmento de código anterior, ya que este archivo de texto es muy importante para el correcto funcionamiento de R8. El archivo proguard-rules.pro es un archivo de texto que permite decirle a R8 algún tipo de restricción para que no elimine referencias o códigos que pese a no ser utilizados de forma estática sí que son accedidos de forma dinámica. Esto pasa mucho en bibliotecas ajenas al código desarrollado. Por ello, muchas de estas bibliotecas tienen en la documentación las reglas explicitas que se necesitan en caso de utilizarlo. Por ejemplo, la biblioteca Glide tiene las siguientes reglas para R8 del Fragmento de código 16.

ProGuard

Depending on your ProGuard (DexGuard) config and usage, you may need to include the following lines in your proguard.cfg (see the Download and Setup docs page for more details):

```
-keep public class * implements com.bumptech.glide.module.GlideModule
-keep class * extends com.bumptech.glide.module.AppGlideModule {
 <init>(...);
}
-keep public enum com.bumptech.glide.load.ImageHeaderParser$** {
  **[] $VALUES;
  public *;
}
-keep class com.bumptech.glide.load.data.ParcelFileDescriptorRewinder$InternalRewinder {
  *** rewind();
}

# for DexGuard only
-keepresourcexmlelements manifest/application/meta-data@value=GlideModule
```

Fragmento de código 16. Reglas Proguard para Glide

El problema ocurre en bibliotecas más pequeñas que carecen de estas reglas en su documentación. Para estas lo más sencillo consiste en decirle a R8 que directamente no toque nada relacionado con ellas.

En la Ilustración 40 se puede observar el método *onCreate*() de la actividad AddTrainingCalendarActivity antes de pasar por R8 en Android Studio y en la Ilustración 41 después de pasar por R8 habiendo decompilado el apk con dex2jar[28] y abierto posteriormente con JD-GUI[29].

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // create and bind the view with data
    binding = DataBindingUtil.setContentView( activity: this, R.layout.activity_add_training_calendar);

    // Replace the default ActionBar (there should be none in order to use menu drawer) by this ToolBar
    setSupportActionBar(binding.includeToolbar.toolBar);
    // Show the user that selecting home will return one level up rather than to the top level
    Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setHomeAsUpIndicator(getDrawable(R.drawable.ic_baseline_arrow_back_24));
    getSupportActionBar().setTitle("Añadir a progreso");

    // get the date which is selected
    day = getIntent().getStringExtra(OneProgressDayActivity.KEY_BUNDLE_DATE);

    // create tabs and set text title tab
    binding.pager2.setAdapter(new AddTrainingCalendarActivity.CustomFragmentStateAdapter( fragmentActivity: this));

    new TabLayoutMediator(binding.tabLayout, binding.pager2, (tab, position) -> {
        switch (position) {
            case 0:
                tab.setText("Ejercicios");
                break;
            case 1:
                tab.setText("Rutinas");
                break;
            case 2:
                tab.setText("Planes");
                break;
        }
    }).attach();
```

Ilustración 40. Código antes de pasar por R8

```
public void onCreate(Bundle paramBundle) {
  super.onCreate(paramBundle);
  j.a.a.e.a a1 = (j.a.a.e.a)e.e((Activity)this, 2131558428);
  this.s = a1;
  I((Toolbar)a1.n.n);
  ((b.b.k.a)Objects.<b.b.k.a>requireNonNull(F())).n(true);
  F().p(getDrawable(2131230879));
  F().r(2131886404);
  this.t = getIntent().getStringExtra("DATE");
  this.s.o.setAdapter((RecyclerView.g)new a((e)this));
  a1 = this.s;
  (new c(a1.p, a1.o, (c.b)new j.a.a.b.a(this))).a();
}
```

Ilustración 41. Código después de pasar por R8

[28] https://github.com/pxb1988/dex2jar
[29] http://java-decompiler.github.io/

6.2.5. Reglas de seguridad en Firebase

Una de las ventajas que proporciona Firebase en cuanto a seguridad para proteger los datos y archivos de una aplicación son las reglas. Estas reglas son aplicables en diferentes funcionalidades como pueden ser Cloud Firestore, Realtime Database o Storage pudiendo tener diferentes reglas en cada una de ellas.

Pese a ello, las reglas introducidas en las diferentes herramientas utilizadas en la aplicación AllThenics son bastante parecidas.

En Storage se tienen creadas tres carpetas de archivos, *exercises* con todas las imágenes pertenecientes a los diferentes ejercicios de la aplicación, *plans* con todas las imágenes pertenecientes a los diferentes planes y *user_images* donde existe una carpeta por cada usuario con las imágenes pertenecientes a cada uno.

Con las reglas introducidas actualmente en Storage lo que se está consiguiendo es que solo los usuarios autenticados puedan acceder a las imágenes de ejercicios y planes. Mientras que para garantizar una mayor seguridad con las imágenes pertenecientes a cada usuario se permite leer y escribir solo al usuario autenticado que tenga el identificador correspondiente a su carpeta.

Todo esto se puede observar en el siguiente Fragmento de código 17 donde se muestran las reglas utilizadas en Storage.

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /exercises/{allPaths=**} {
      allow read: if request.auth != null;
    }
    match /plans/{allPaths=**} {
      allow read: if request.auth != null;
    }
    match /user_images/{userId}/{allPaths=**} {
      allow read: if request.auth.uid == userId;
      allow write: if request.auth.uid == userId;
    }
  }
}
```

Fragmento de código 17. Reglas de seguridad utilizadas en Storage

Como se puede imaginar estas reglas tienen un potencial enorme pudiendo ya no solo controlar si cualquiera lee o escribe, sino más concretamente hacer una actualización, un borrado o una inserción si se cumplen ciertas condiciones y sin tener que realizar manualmente estas comprobaciones.

Capítulo 7

Resultados

En el siguiente capítulo se van a discutir los resultados obtenidos por la característica Firebase Crashlytics al realizar el segundo experimento del Capítulo 3 con los diferentes *early adopters,* algunos datos obtenidos de Google Analytics y el resultado de los test midiendo los tiempos de respuesta con el servidor.

7.1. Resultados de Crashlytics

Como bien se ha comentado en la sección correspondiente a las tecnologías utilizadas Crashlytics sirve para detectar excepciones producidas en las aplicaciones indicando la excepción y la línea en la cual se ha producido, además de datos interesantes como el hardware y software utilizado con todo tipo de detalles y pudiendo añadir datos de interés a nivel de código como pueden ser el identificador del usuario o algún tipo de variable que crea el desarrollador que pueda ser importante conocer su valor en el momento de la fallada.

Esta característica fue implementada para el segundo experimento ya que al tener una mayor cantidad de usuarios se podría obtener una buena retroalimentación para poder detectar cualquier tipo de fallo.

Muchas veces suele ocurrir que gente no experta en el desarrollo de aplicaciones al producirse una excepción, cerrarse y al volver a abrirla ver que todo funciona bien, no creen que sea un fallo de la aplicación que se pueda investigar y por tanto no llegan a notificar a los desarrolladores. Esto nos hemos podido dar cuenta en este experimento al recibir los fallos en Crashlytics porque algunos de los informes recibidos no han sido notificados por el usuario en cuestión y no hubiésemos sido conscientes de ello.

La siguiente Ilustración 42 representa todos los tipos de fallos detectados en el segundo experimento.

Problemas	Detalles	Versiones	Eventos	Usuarios
NavigationDrawerViewModel.java – line 456 vcardona.countero.allthenics.view_model.NavigationDrawerViewModel.getRoutine	Falla	0.2.1 – 0.2.1	2	2
ViewGroup.java – line 3886 android.view.ViewGroup.dispatchRestoreInstanceState	Falla	0.2.1 – 0.2.1	2	2
ViewGroup.java – line 3829 android.view.ViewGroup.dispatchRestoreInstanceState	Falla	0.2.1 – 0.2.1	1	1
CalendarPickerActivity.java – line 48 vcardona.countero.allthenics.activities.CalendarPickerActivity.onCreate	Falla	0.2.1 – 0.2.1	1	1
ProgressFragment.java – line 209 vcardona.countero.allthenics.fragments.ProgressFragment.initFilterDateRange	Falla	0.2.1 – 0.2.1	1	1
RequestManagerRetriever.java – line 317 com.bumptech.glide.manager.RequestManagerRetriever.assertNotDestroyed	Falla	0.2.1 – 0.2.1	1	1

Ilustración 42. Excepciones producidas proporcionadas por Crashlytics

Al pulsar en un problema se accede a una vista más detallada de la excepción con el seguimiento de pila que se obtendría en Android Studio, las claves que el desarrollador pueda haber introducido que hubiese creído necesarias, los registros por los cuales ha ido navegando el usuario, es decir todo el *flow* que ha hecho el usuario hasta producirse la excepción, y los datos hardware y software del dispositivo móvil. Todo esto se puede observar en la Ilustración 43.

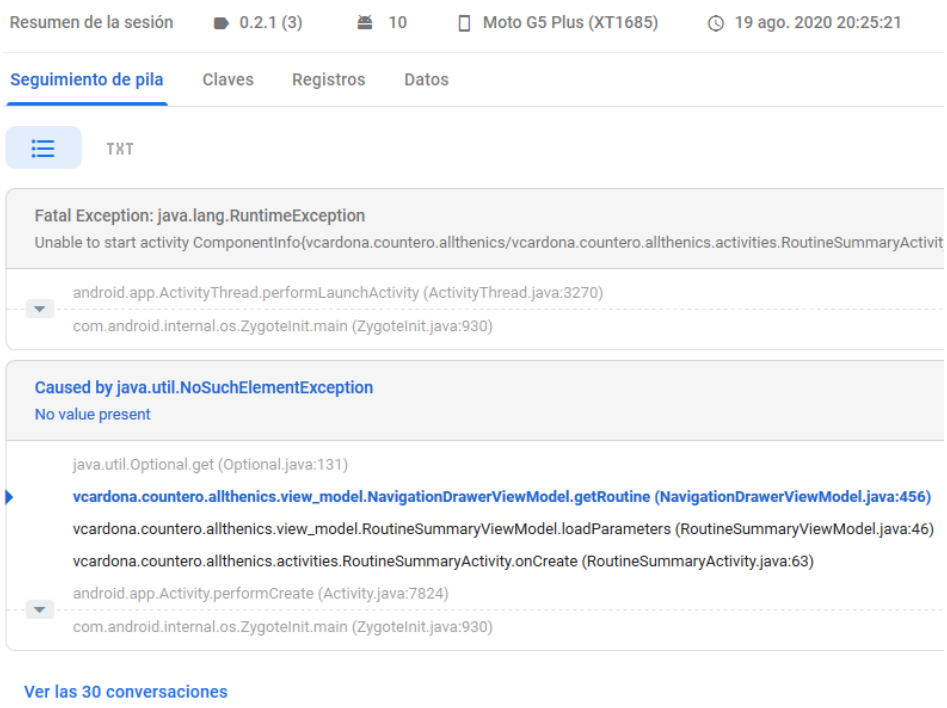
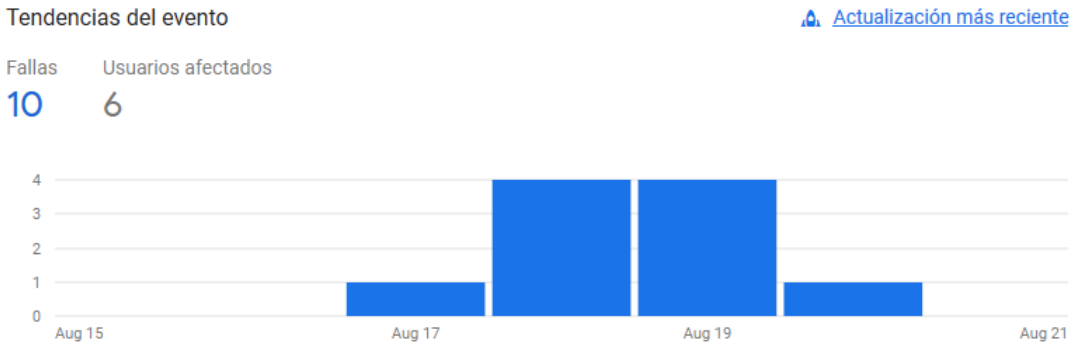


Ilustración 43. Excepción en NavigationDrawerViewModel.java

Como también ha ocurrido con las notificaciones push con Cloud Messaging en el apartado 6.2.3, Analytics también aparece cuando estamos utilizando Crashytics proporcionando gráficas para que de un solo simple vistazo se pueda ver lo que está ocurriendo con las excepciones producidas.

En primera instancia al entrar en la sección Crashlytics de Firebase tenemos dos gráficas, una de líneas que indica el porcentaje de usuarios que no han experimentado ningún bloqueo en función de los días y en una gráfica de columnas el número de fallos producidos con la cantidad de usuarios afectados también en función de los días. La Gráfica 29 y la Gráfica 30 representan estas dos gráficas del segundo experimento.

Gráfica 29. Usuarios sin bloqueos en el segundo experimento



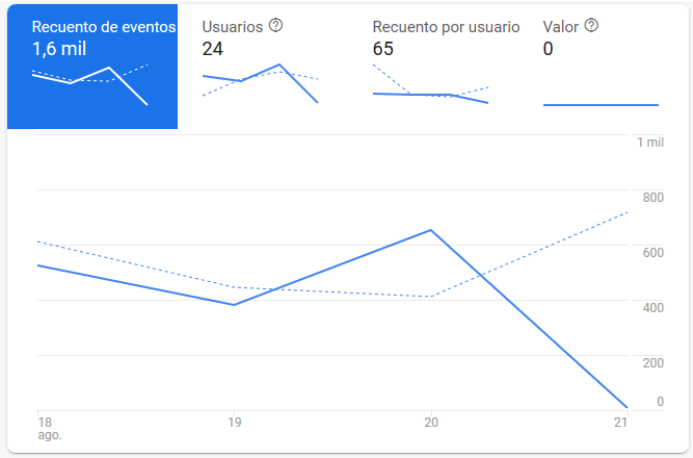
Gráfica 30. Fallas detectados con usuarios afectados en el segundo experimento

7.2. Resultados de Analytics

Analytics proporciona una cantidad de datos ingente bastante útiles para cualquier desarrollo software y no todos pueden ser comentados, por tanto, en esta sección se van a documentar aquellos resultados obtenidos que parecen más interesantes.

Los datos más interesantes vienen dados por los eventos. En esta sección de Analytics se encuentran valores importantes como pueden ser el número de eventos obtenidos en total o el tiempo de interacción de los usuarios en las diferentes vistas.

La Gráfica 31 ejemplifica el número de eventos producidos durante el segundo experimento. Esto es interesante porque con estos datos se puede afirmar el número de usuarios que han participado activamente en el experimento y la cantidad de interacción que han podido tener con la aplicación. En este caso, es un valor bastante correcto y aceptable ya que veinticuatro usuarios han hecho un total de mil seiscientos eventos con la aplicación.



Gráfica 31. Eventos producido en el segundo experimento

La siguiente Gráfica 32 corresponde con la interacción diaria de los usuarios en cada vista. El mayor porcentaje de tiempo con un 56,84% se corresponde con la actividad NavigationDrawerActivity que en la aplicación es la pantalla principal una vez el usuario se encuentra autenticado, por tanto, es lógico que esto sea así con un tiempo medio de cuatro segundos. Le sigue con un 23,13% DoRoutineActivity que es la actividad que engloba todo lo relacionado con realizar una rutina. Se puede observar que este valor es de quince segundos que, respecto a la anterior actividad, es un incremento considerable ya que es casi cuatro veces más.

Pese a ello, el tiempo medio en una actividad no parece un valor excesivamente reseñable porque quince segundos es muy poco para realizar una rutina que suelen ser de cinco minutos o más, pero esto ocurre porque si un usuario entra por probar y no acaba realizando la rutina, el valor medio disminuye considerablemente.

Lo más reseñable es el tiempo medio global de uso en la aplicación que se corresponde a diez minutos y dieciocho segundos. Además, se puede observar por los

porcentajes de aumento en la actividad DoRoutineActivity que, pese a que el experimento ha finalizado y que sería lógico que todo estuviese en decremento, existen algunos *early adopters* que siguen utilizando la aplicación para realizar ejercicio, por tanto, es una estadística muy favorable.

Gráfica 32. Tiempo por vista en el segundo experimento

7.3. Pruebas de tiempo de respuesta

Para poder medir la calidad del servicio prestado en relación con el *back-end* se ha decidido realizar una serie de pruebas de tiempo de respuesta. Al no disponer de una cantidad elevada de dispositivos se ha optado por la opción de realizar desde el mismo móvil diversas llamadas desde varios hilos de ejecución. Al realizarlo de esta forma seguramente los tiempos obtenidos no serán del todo correctos ya que resulta en una mayor carga del dispositivo obteniendo unos tiempos mayores a los reales, así que la diferencia entre un dispositivo y mil puede que realmente no llegue a ser de casi un segundo de diferencia, sino que sean tiempos más parecidos.

Cabe destacar que la mayor carga de información se produce al iniciar la aplicación, ya que al lanzarla se obtienen casi todos los datos necesarios exceptuando las imágenes. Con ello se consigue que el usuario solo tenga que esperar una carga de datos al principio consiguiendo una aplicación mucho más fluida. También se ha diferenciado el tiempo de respuesta obtenido por WIFI o 4G, en la Tabla 11 se exponen los resultados obtenidos en estas pruebas. En el Anexo C se encuentran documentadas.

Tabla 11. Tabla de resultados

	1 dispositivo	100 dispositivos	500 dispositivos	1000 dispositivos
WIFI	1,153 s	1,309 s ± 0,07	1,631 s ± 0,21	1,972 s ± 0,34
4G	1,347 s	1,521 s ± 0,12	1,866 s ± 0,27	2,011 s ± 0,39

Capítulo 8

Conclusiones

Respecto a los objetivos propuestos en el apartado 1.2, todos han sido cumplidos satisfactoriamente. Se han conseguido realizar dos experimentos, se puede acceder a los contenidos de manera online y offline, siempre y cuando se haya accedido ya al menos una vez online, se pueden realizar rutinas de diferentes tipos como HIIT, circuito, EMOM, tabata o piramidal, se ha conseguido en el segundo experimento hacer la funcionalidad de progreso y, posiblemente lo más importante, se ha conseguido más de un 70% de satisfacción en ambos experimentos.

Los tres objetivos restantes son específicos de la parte de back-end y también han sido logrados. Conseguir un tiempo medio de respuesta de tres segundos como máximo, implementar la arquitectura *model-view-viewmodel* y ofuscar la aplicación mediante R8.

Además, gracias a este proyecto de emprendimiento se ha conseguido obtener una experiencia lo más real posible al mundo laboral y, en este caso, con la parte de formar una empresa desde cero. Al poder trabajar mano a mano con un compañero y no tener que realizar el proyecto de manera independiente, se ha trabajado duramente la competencia transversal CT-6 Trabajo en equipo y liderazgo que en el futuro será una de las más importantes. Además, el hecho de que cada uno fuese de una rama diferente de la carrera ha enriquecido el proyecto en su totalidad. Se han tenido sus más y sus menos durante todo el desarrollo, pero la experiencia ha sido increíble, dando apoyo constante, compartiendo ideas, mejorando funciones, distribuyendo el trabajo acordado y haciendo algunas noches señaladas *pair programming* hasta altas horas, sobretodo en fechas cercanas a la conclusión de los *sprints*.

Este aprendizaje ya no solo ha sido a nivel de cercanía al mundo laboral, sino que también ha habido un aprendizaje de tecnologías que no hubiese sido posible de otra forma. Estas tecnologías están en el punto de mira actual y muchas de ellas están

siendo altamente demandadas en diferentes empresas haciendo que todo lo aprendido forme una base de conocimientos muy útil en un futuro. Se han aprendido muchas otras cosas igual de útiles como saber organizar mejor el código o mantener plena consciencia de fechas y organización del calendario. Cabe destacar que también se ha aprendido a documentar correctamente un código, ya que al ser dos personas y que posiblemente un trozo de código sea necesario modificarlo o utilizarlo en un futuro era muy importante que se entendiese cómo funcionaba y para que servía.

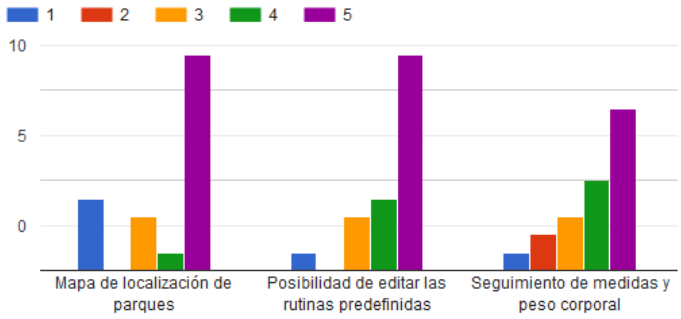
Muchas asignaturas de la carrera han sido útiles en el desarrollo de la aplicación destacando PIN (11554 – Proyecto de ingeniería de software) que ha servido para organizar todo el desarrollo utilizando metodología ágil para poder adaptarse a cualquier cambio, DADM (14093 – Desarrollo de aplicaciones para dispositivos móviles) que ha establecido una base de programación en entornos móviles que pese a ser parecido al desarrollo en otras plataformas, se diferencia lo suficiente, BDA (11548 – Bases de datos y sistemas de información) que pese a haber usado finalmente NoSQL ha servido para conocer como se hace una buena estructura de base de datos y cómo gestionarla correctamente, MES (11569 – Mantenimiento y evolución de software) que ha servido para gestionar el código utilizando buenos estilos de programación y para realizar un buen *refactoring* y, por último, DDS (11565 – Diseño de software) para saber cómo utilizar diferentes patrones tanto de software como arquitectónicos que han resultado útiles en la aplicación.

Para finalizar este capítulo, destacar que el proyecto de emprendimiento aún se encuentra en fase de desarrollo, es decir, que pese a que este trabajo ha sido finalizado, se va a continuar con un tercer *sprint* que finalizará con un tercer experimento, muy probablemente pasando a versión alfa y dejando las pruebas internas atrás, consiguiendo así que más usuarios, ya no solo cercanos a nosotros y relativos a ellos, puedan acceder a participar en el desarrollo de la aplicación desde Google Play.

Capítulo 9

Trabajo futuro

Como bien se ha comentado en diversos apartados anteriormente, el proyecto no se encuentra finalizado. Al acabar este trabajo comenzará un tercer *sprint* para implementar todas las funcionalidades que por tiempo no han podido llegar a implementarse e introducir mejoras recibidas de la retroalimentación del segundo experimento además de nuevas funcionalidades. En la Gráfica 33 se observa las funcionalidades más demandadas por los *early adopters* de este segundo experimento.



Gráfica 33. Funcionalidades futuras más demandadas

Por último, pensando a más largo plazo, se tiene pensando realizar muchas otras mejoras como pueden ser el desarrollo de una API REST para que haga de intermediaria entre la aplicación cliente y Firebase aligerando así el trabajo de la app en los dispositivos móviles, traducción completa de la aplicación a varios idiomas como pueden ser inglés, francés y alemán que viene enlazado con el despliegue de la aplicación en varios países de habla no hispana y el desarrollo de la aplicación para dispositivos iOS y para Smartwatch.

Referencias

[1] R. Guthold, G. A. Stevens, L. M. Riley y F. C. Bull, «Worldwide trends in insufficient physical activity from 2001 to 2016: a pooled analysis of 358 population-based surveys with 1·9 million participants,» *The Lancet,* vol. 6, nº 10, 2018.

[2] P. Garwood, C. Lindmeier y T. Jasarevic, «Organización Mundial de la Salud (OMS),» 4 Junio 2018. [En línea]. Available: https://www.who.int/es/news-room/detail/04-06-2018-who-launches-global-action-plan-on-physical-activity. [Último acceso: 5 Mayo 2020].

[3] EC, «European Commission,» 21 Marzo 2018. [En línea]. Available: https://ec.europa.eu/sport/news/2018/new-eurobarometer-sport-and-physical-activity_en. [Último acceso: 5 Mayo 2020].

[4] U. García, P. Ricard y P. Coll, «Workout temple,» 8 Octubre 2019. [En línea]. Available: https://workout-temple.com/es/calisthenics-articles/what-is-calisthenics. [Último acceso: 6 Mayo 2020].

[5] Y. Alonso y S. Catalán, «Historia de la calistenia y street workout,» de *La calle es tu gimnasio: Guía completa de calistenia y street workout*, Penguin Random House Grupo Editorial, 2020, pp. 12-13.

[6] «Greatest Physiques,» 23 Marzo 2019. [En línea]. Available: https://www.greatestphysiques.com/male-physiques/hannibal-lanham/. [Último acceso: 22 Mayo 2020].

[7] WSWCF, «World Street Workout and Calisthenics Federation,» [En línea]. Available: https://wswcf.org/members/. [Último acceso: 5 Mayo 2020].

[8] FECSW, «Federación española de calistenia y street workout,» [En línea]. Available: https://feswc.org/clubs-deportivos-feswc/. [Último acceso: 5 Mayo 2020].

[9] C. Gough, «Statista,» 24 Junio 2020. [En línea]. Available: https://www.statista.com/statistics/191527/participants-in-calisthenics-in-the-us-since-2006/. [Último acceso: 5 Mayo 2020].

[10] AppBrain, «AppBrain,» 8 Mayo 2020. [En línea]. Available: https://www.appbrain.com/stats/number-of-android-apps. [Último acceso: 9 Mayo 2020].

[11] A. Maurya, «Lean stack,» 27 Febrero 2012. [En línea]. Available: https://blog.leanstack.com/why-lean-canvas-vs-business-model-canvas-af62c0f250f0. [Último acceso: 7 Mayo 2020].

[12] Santalucia Impulsa, «Santalucia Impulsa,» 6 Marzo 2020. [En línea]. Available: https://www.santaluciaimpulsa.es/metodologia-lean-startup/. [Último acceso: 21 Mayo 2020].

[13] E. Ries, El método Lean Startup: Cómo crear empresas de éxito utilizando la innovación continua, Grupo Planeta Spain, 2012.

[14] Google, «Guía de arquitectura de apps,» [En línea]. Available: https://developer.android.com/jetpack/docs/guide. [Último acceso: 29 Mayo 2020].

[15] A. Sehgal, «Medium,» 28 Julio 2018. [En línea]. Available: https://medium.com/@mr.anmolsehgal/common-android-architectures-mvc-vs-mvp-vs-mvvm-afd8461e1fee. [Último acceso: 30 Mayo 2020].

[16] Google, «FirebaseUI,» [En línea]. Available: https://firebase.google.com/docs/auth/android/firebaseui?hl=es-419. [Último acceso: 12 Junio 2020].

[17] Google, «Cómo reducir, ofuscar y optimizar tu app,» [En línea]. Available: https://developer.android.com/studio/build/shrink-code?hl=es. [Último acceso: 15 Agosto 2020].

Anexo A

Formulario MVP1

A.1 Datos personales

- Edad
  - Respuesta abierta numérica
- Género
  - Masculino
  - Femenino
  - Otro

A.2 Experiencia previa

- ¿Practicas deporte?
  - Si
  - No
- ¿Con qué frecuencia lo practicas a la semana?
  - 1 día
  - Entre 2 y 3 días
  - Entre 4 y 5 días
  - Más de 5 días
- ¿Has realizado calistenia alguna vez?
  - Sí
  - No
- ¿Has gastado alguna aplicación para gestionar tu entrenamiento?
  - Sí y sigo utilizándola
  - Sí, pero ya no hago uso de ella
  - No

A.3 Interfaz

- ¿Crees que la manera de desplazarse entre las vistas es correcta?
  - Sí
  - No
  - Creo que se podría mejorar
- ¿De que manera la mejorarías?
  - Respuesta abierta
- ¿Crees que la tipografía utilizada en ejercicios, rutinas y planes es adecuada?
  - Sí
  - No
- ¿Crees que los colores utilizados son los correctos?
  - Sí
  - No
  - Cambiaría colores
- ¿Qué colores utilizarías y dónde?
  - Respuesta abierta
- ¿Has tenido algún problema con el significado de algún icono?
  - Sí
  - No
- ¿Con cuál?
  - Respuesta abierta
- ¿Tienes algún comentario o sugerencia sobre la interfaz?
  - Respuesta abierta

A.4 Funcionalidad

- ¿Te ha resultado cómodo y útil realizar un ejercicio?
  - Sí
  - No
  - Realizaría cambios
- ¿Qué cambios realizarías y por qué?
  - Respuesta abierta
- ¿Te ha resultado cómodo y útil realizar una rutina?
  - Respuesta abierta

- ¿Qué cambios realizarías y por qué?
    - Respuesta abierta
- ¿Has encontrado algún fallo?
    - Sí
    - No
- ¿Cuál?
    - Respuesta abierta
- ¿Tienes algún comentario o sugerencia sobre la funcionalidad?
    - Respuesta abierta

A.5 Modelo de negocio

- ¿Pagarías una suscripción *premium* por tener acceso a los planes de entrenamiento?
    - Sí
    - No
- ¿Cuánto estarías dispuesto/a a pagar mensualmente?
    - 1,99€
    - 2,99€
    - 3,99€
    - 4,99€
    - 5,99€

A.6 Opiniones

- En general, ¿estás satisfecho/a con la aplicación?
    - Sí
    - No
- ¿La utilizarías en un futuro?
    - Sí
    - No
- Actualmente, ¿qué valoración le darías a la aplicación?
    - 1
    - 2
    - 3
    - 4
    - 5
    - 6
    - 7
    - 8
    - 9
    - 10
- ¿Tienes alguna duda, comentario y/o sugerencia sobre la aplicación?
    - Respuesta abierta

Anexo B

Formulario MVP2

B.1 Datos personales

- Edad
  - Respuesta abierta numérica
- Género
  - Masculino
  - Femenino
  - Otro

B.2 Experiencia previa

- ¿Practicas deporte?
  - Si
  - No
- ¿Con qué frecuencia lo practicas a la semana?
  - 1 día
  - Entre 2 y 3 días
  - Entre 4 y 5 días
  - Más de 5 días
- ¿Has realizado calistenia alguna vez?
  - Sí
  - No
- ¿Has gastado alguna aplicación para gestionar tu entrenamiento?
  - Sí y sigo utilizándola
  - Sí, pero ya no hago uso de ella
  - No

B.3 Interfaz

- ¿Crees que la manera de desplazarse entre las vistas es correcta?
  - Sí
  - No
  - Creo que se podría mejorar
- ¿De que manera la mejorarías?
  - Respuesta abierta
- ¿Crees que la tipografía utilizada en ejercicios, rutinas y planes es adecuada?
  - Sí
  - No
- ¿Crees que los colores utilizados son los correctos?
  - Sí
  - No
- ¿Qué colores utilizarías y donde?
  - Respuesta abierta
- ¿Has tenido algún problema con el significado de algún icono?
  - Sí
  - No
- ¿Con cuál?
  - Respuesta abierta
- ¿Te han resultado útiles las diferentes gráficas?
  - Sí
  - No
- ¿Realizarías algún cambio? ¿Añadirías otro tipo de gráfica?
  - Respuesta abierta
- ¿Tienes algún comentario o sugerencia sobre la interfaz.
  - Respuesta abierta

B.4 Funcionalidad

- ¿Te ha resultado cómodo y útil realizar un ejercicio?
  - Sí
  - No
  - Realizaría cambios
- ¿Qué cambios realizarías y por qué?
  - Respuesta abierta

- ¿Te ha resultado cómodo y útil realizar una rutina?
    - Sí
    - No
    - Realizaría cambios
- ¿Qué cambios realizarías y por qué?
    - Respuesta abierta
- ¿Te ha resultado cómodo y útil la manera de introducir las repeticiones al entrenar?
    - Sí
    - No
- ¿Qué cambios realizarías? ¿Por qué?
    - Respuesta abierta
- Valora del 1 al 5 las rutinas que más te gustaría ver en un futuro.
    - Circuito (Escala Likert 1-5)
    - Piramidal (Escala Likert 1-5)
    - EMOM (Escala Likert 1-5)
    - HIIT (Escala Likert 1-5)
    - Tabata (Escala Likert 1-5)
- ¿Te ha resultado útil la pantalla de progreso global y específica de un día para ver el seguimiento de ejercicios y rutinas realizadas?
    - Sí
    - No
- ¿Qué cambios realizarías? ¿Por qué?
    - Respuesta abierta
- Te ha resultado cómodo y útil guardar el entrenamiento de un ejercicio, rutina o plan en cualquier día de la semana?
    - Sí
    - No
- ¿Qué cambios realizarías? ¿Por qué?
    - Respuesta abierta
- ¿Has encontrado algún fallo?
    - Sí
    - No
- ¿Cuál?
    - Respuesta abierta

- ¿Tienes algún comentario o sugerencia sobre la funcionalidad?
    - Respuesta abierta

B.5 Modelo de negocio

- ¿Pagarías una suscripción *premium* por tener acceso a los planes de entrenamiento y poder utilizar todo el progreso sin restricción a tres meses de entrenamientos almacenados?
    - Si
    - No
- ¿Cuánto estarías dispuesto/a a pagar mensualmente?
    - 1,99€
    - 2,99€
    - 3,99€
    - 4,99€
    - 5,99€

B.6 Próximas actualizaciones

- Valora del 1 al 5 cuál de las siguientes características te gustaría ver en la siguiente actualización.
    - Artículos (Escala Likert 1-5)
    - Gráfica comparativa entre dos ejercicios (Escala Likert 1-5)
    - Mapa de localización de parques (Escala Likert 1-5)
    - Posibilidad de editar lar rutinas predefinidas (Escala Likert 1-5)
    - Seguimiento de medidas y peso corporal (Escala Likert 1-5)
    - Compartir el resumen del entrenamiento (Escala Likert 1-5)
    - Minijuegos (Ruleta, ejercicios aleatorios, rutinas de música, …) (Escala Likert 1-5)
- ¿Tienes alguna idea de otra característica para implementarla?
    - Respuesta abierta

B.7 Opiniones

- ¿Qué logo para la aplicación te gusta más?

Ilustración 45. Logo 1

Ilustración 44. Logo 2

Ilustración 46. Logo 3

Ilustración 47. Logo 4

- ¿Crees que esta versión ha solucionado las deficiencias de la primera?
  - Sí
  - No
- En general, ¿estás satisfecho/a con la aplicación?
  - Sí
  - No
- ¿La utilizarías en un futuro?
  - Sí
  - No
- Actualmente, ¿qué valoración le darías a la aplicación?
  - 1
  - 1.5
  - 2
  - 2.5
  - 3
  - 3.5
  - 4
  - 4.5
  - 5
- ¿Tienes alguna duda, comentario y/o sugerencia sobre la aplicación?
  - Respuesta abierta

Anexo C

Pruebas de tiempo de respuesta

Como bien se ha indicado en la sección 7.3, las pruebas necesarias para tener una estimación aproximada de la calidad del servicio prestado han sido de tiempo de respuesta. Las pruebas se han obtenido ejecutando la aplicación de forma normal, con cien hilos de ejecución, quinientos hilos de ejecución y mil hilos de ejecución.

Se ha obtenido el tiempo actual al iniciar el fragmento LoadingData, el cual se encarga de realizar todas las peticiones exceptuando las imágenes que cargará Glide posteriormente en la aplicación, y al final justo al hacer el cambio de fragmento que es cuando se han obtenido ya todos los datos. En el Fragmento de código 18 se observa este código.

```
private void replaceFragment() {
    if (allExerciseDone && favouriteExercisesDone && allRoutinesDone && favouriteRoutinesDone
            && plansDone && userLogExercisesDone && userLogRoutinesDone) {

        long finalTime = System.currentTimeMillis();
        Log.v( tag: "tiempo resultado",  msg: (finalTime - time) + "");
```

Fragmento de código 18. Cálculo del tiempo de respuesta/latencia

Por último, para poder realizar los cálculos con varios dispositivos conectados al mismo tiempo, se han creado diversos hilos de ejecución dependiendo de la prueba en cada *endpoint* que hace la llamada a la base de datos. En el Fragmento de código 19 está realizado en el manager de ejercicios en la función que obtiene todos los ejercicios. El tiempo total de cada hilo se ha sumado y se ha obtenido el tiempo medio, por último, se ha añadido la desviación típica según el hilo que más ha tardado y el que menos.

```
public void getAllExercises(OnCompleteListener<QuerySnapshot> onCompleteListener) {
    for (int i = 0; i < 1000; i++) {
        Thread thread = new Thread() {
            @Override
            public void run() {
                exercisesCollectionReference.get().addOnCompleteListener(onCompleteListener);
            }
        };
        thread.start();
    }
}
```

Fragmento de código 19. Lanzamiento de diversos hilos de ejecución para pruebas