UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC

Universitat Politècnica de València
Departamento de Sistemas Informáticos y Computación

FitG: Desarrollo ágil de un sistema híbrido para la gestión del ejercicio físico

Trabajo de Fin de Máster

Máster Universitario en Ingeniería y Tecnología de Sistemas Software

Autor
Rubén Moreno Jimeno
Directores
María Carmen Penadés Gramage
Patricio Orlando Letelier Torres

Julio, 2018

Para todas esas personas que creen que el deporte tiene el poder de cambiar las vidas.

AGRADECIMIENTOS

A mis tutores por darme la oportunidad de realizar este TFM y por sus comentarios y ayuda.

A todos los participantes que han accedido a realizar los experimentos y han tenido la paciencia y el tiempo para probar y validar el sistema.

A mis amigos, a mi hermano y a Andrea, por todo el apoyo incondicional que me han estado dando a lo largo de todo este año pese a la distancia.

A mis padres por todo el apoyo y por el esfuerzo y sacrificio de permitirme una vez más seguir continuando mis estudios, incluso estando fuera de casa.

RESUMEN

Cada vez es más frecuente ver que la gente se interesa por su estado físico. Sin embargo, la implantación de la tecnología en el mundo del deporte, aunque es importante, sino siendo mejorable en muchos aspectos, por ejemplo, sigue habiendo una carencia considerable de aplicaciones que ayuden a ver la progresión del estado físico de la persona. Muchas de las aplicaciones existentes, no aportan la capacidad de mostrar unas estadísticas/gráficas globales, para el análisis del progreso de dichos ejercicios a lo largo del tiempo de una forma intuitiva. Asimismo, suelen centrarse únicamente en un tipo de deporte, lo que incomoda la tarea para quienes practican más de uno. Además, suelen ser poco flexibles a la hora de ofrecer tipos de ejercicios, haciendo que uno se tenga que adaptar a los que contiene la aplicación. Por otro lado, actualmente hay pocas aplicaciones que permitan la recogida de datos acerca de la salud física, para que se puedan analizar y así comprobar si uno se mantiene en unos niveles saludables o no.

El objetivo de este TFM es crear una solución web para dichas carencias dentro del marco de un proyecto de emprendimiento. En esta solución, los usuarios cuentan con una aplicación web para móviles donde se da la posibilidad de gestionar nuevos tipos de ejercicios, la posibilidad de añadir marcas a dichos ejercicios realizados, añadir análisis físicos sobre la salud de los usuarios, y búsqueda rápida de los ejercicios. Por otro lado, se cuenta con una aplicación web para escritorio con la posibilidad de ver los históricos de las marcas y los análisis, tanto de forma detallada como en forma de gráficas, que a su vez puedan ser mínimamente personalizadas.

En la gestión del proyecto se ha aplicado la metodología ágil SCRUM. Además, puesto que se trata como ya se ha dicho de un proyecto de emprendimiento, también se han seguido las recomendaciones que ofrece el marco y metodología de Lean Startup. Se ha realizado una amplia evaluación de la idea de negocio, analizando su viabilidad como producto de mercado, y el trabajo se ha dividido en tres iteraciones. Además, se ha reducido el riesgo y la incertidumbre de la idea, gracias al uso de conceptos clave como los productos mínimos viables, y el uso de los pasos y herramientas que proporciona para validar la idea de negocio, tanto antes de su puesta en marcha, como al poco tiempo de iniciar el proyecto. Estas metodologías, sin embargo, se han adaptado a las características del proyecto ya que éste cuenta con algunas peculiaridades dada su naturaleza de TFM.

El sistema web desarrollado cuenta con una arquitectura Modelo-Vista-Controlador, implementando el sistema en capas de abstracción (capa de presentación, de servicios web y controladores, y de persistencia y búsqueda de datos). Dicho sistema se ha implementado con las tecnologías del *stack* MEAN al completo, y el *framework* Ionic 2 para los clientes móviles.

Dada el tipo de proyecto que se ha desarrollado y la metodología que se ha seguido, se han realizado dos validaciones con usuarios reales. Esto ha consistido en dos experimentos supervisados en los que los usuarios probaban la aplicación y daban retroalimentación de esta para así poder comprobar si el producto era o no viable y se estaba siguiendo la estrategia adecuada.

Palabras clave: gestión del deporte, metodologías ágiles, lean startup, *stack* MEAN, *framework* Ionic 2

SUMMARY

It is increasingly common to see people care about their physical condition. Nevertheless, the implantation of technology in the world of sports, although it is important, but being improvable in many aspects, for example, there is still a considerable lack of applications that help to see the progress of the physical state of the person. Many of the existing applications do not provide the capacity to display a global statistics/graphs, to analyze the progress of these exercises over time in an intuitive way. Besides, they usually focus only on one type of sport, which makes the task more difficult for those who practice more than one. Moreover, they usually are inflexible when it comes to offering types of exercises, making one have to adapt to those contained in the application. On the other hand, there are currently few applications that allow the data collection about physical health, in order to analyze them and check whether one stays at a healthy level or not.

The aim of this Master's Thesis is to create a web solution for these deficiencies within the framework of an entrepreneurship project. In this solution, users have a mobile web application where they can manage new exercise types, the possibility to add marks to these exercises, add physical analysis about the user's health, and quickly search for exercises. On the other hand, there is a desktop web application with the ability to view historical marks and analyzes, both in detail and in the form of graphics, which they can be minimally customized.

In the management of the project, the agile SCRUM methodology has been applied. In addition, since it is an entrepreneurship project, as it has already been said, the recommendations offered by the Lean Startup framework and methodology have also been followed. It has been made a broad evaluation of the business idea, analyzing its viability as a market product, and the whole development has been divided into three iterations. Also, the risk and uncertainty of the idea has been reduced, thanks to the use of key concepts such as viable minimum products, and the use of the steps and tools that it provides to validate the business idea, both before its implementation and at the beginning of the project. These methodologies, however, have been adapted to the project's features since it has some peculiarities given its nature of Master's Thesis.

The developed web system has a Model-View-Controller architecture, implementing the system in abstraction layers (presentation, web system and controllers, and persistence and data search layers). This system has been implemented with the full stack MEAN technologies, and the Ionic 2 framework for the mobile clients.

Given the type of project that has been developed and the methodology that has been followed, two validations have been carried out with real users. This consisted of two supervised experiments in which the users tested the application and gave feedback on it in order to check if the product was viable or not and the appropriate strategy was being followed.

Keywords: sports management, agile methodologies, lean startup, stack MEAN, framework Ionic 2

RESUM

Cada vegada és més freqüent veure que la gent s'interessa pel seu estat físic. No obstant això, la implantació de la tecnologia en el món de l'esport, encara que és important, es pot millorar en molts aspectes, per exemple, continua havent-hi una manca considerable d'aplicacions que ajuden a veure la progressió de l'estat físic de la persona. Moltes de les aplicacions existents, no aporten la capacitat de mostrar unes estadístiques/gràfiques globals, per a l'anàlisi del progrés d'aquests exercicis al llarg del temps d'una forma intuïtiva. Així mateix, solen centrar-se únicament en un tipus d'esport, la qual cosa incomoda la tasca per als qui practiquen més d'un. A més, solen ser poc flexibles a l'hora d'oferir tipus d'exercicis, fent que un s'haja d'adaptar als quals conté l'aplicació. D'altra banda, actualment hi ha poques aplicacions que permeten la recollida de dades sobre la salut física, perquè es puguen analitzar i així comprovar si un es manté en uns nivells saludables o no.

L'objectiu d'aquest TFM és crear una solució web per a aquestes manques dins del marc d'un projecte d'emprenedoria. En aquesta solució, els usuaris compten amb una aplicació web per a mòbils on es dóna la possibilitat de gestionar nous tipus d'exercicis, la possibilitat d'afegir marques a aquests exercicis realitzats, afegir anàlisis físiques sobre la salut dels usuaris, i cerca ràpida dels exercicis. D'altra banda, es compta amb una aplicació web per a escriptori amb la possibilitat de veure els històrics de les marques i les anàlisis, tant de forma detallada com en forma de gràfiques, que al seu torn puguen ser mínimament personalitzades.

En la gestió del projecte s'ha aplicat la metodologia àgil SCRUM. A més, ja que es tracta com ja s'ha dit d'un projecte d'emprenedoria, també s'han seguit les recomanacions que ofereix el marc i metodologia de LEAN Startup. S'ha realitzat una àmplia avaluació de la idea de negoci, analitzant la seua viabilitat com a producte de mercat, i el treball s'ha dividit en tres iteracions. A més, s'ha reduït el risc i la incertesa de la idea, gràcies a l'ús de conceptes clau com els productes mínims viables, i l'ús dels passos i eines que proporciona per a validar la idea de negoci, tant abans de la seua posada en marxa, com al poc temps d'iniciar el projecte. Aquestes metodologies, no obstant això, s'han adaptat a les característiques del projecte ja que aquest compta amb algunes peculiaritats donada la seua naturalesa de TFM.

El sistema web desenvolupat compta amb una arquitectura Model-Vista-Controlador, implementant el sistema en capes d'abstracció (capa de presentació, de serveis web i controladors, i de persistència i cerca de dades). Aquest sistema s'ha implementat amb les tecnologies del stack MEAN al complet, i el framework Ionic 2 per als clients mòbils.

Pel tipus de projecte que s'ha desenvolupat i la metodologia que s'ha seguit, s'han realitzat dues validacions amb usuaris reals. Això ha consistit en dos experiments supervisats en els quals els usuaris provaven l'aplicació i donaven retroalimentació d'aquesta per a així poder comprovar si el producte era o no viable i s'estava seguint l'estratègia adequada.

Paraules clau: gestió del deport, metodologies àgils, lean startup, *stack* MEAN, *framework*Ionic 2

TABLA DE CONTENIDOS

Lista de Figuras

Lista de Tablas

1. Introducción

Cada vez es más frecuente ver que la gente se interesa por su estado físico (CMD Sport, 2016). La gente intenta mantener algún tipo de control sobre su alimentación, a la vez que intenta realizar algún tipo de deporte. Sin embargo, pese a los grandes avances tecnológicos que existen hoy en día en muchas áreas, o una gran cantidad de aplicaciones que te ayudan y complementan en la realización de algunas tareas, en el mundo del deporte todavía no se han integrado al completo, y hay una carencia considerable de las mismas.

Para una persona que le gusta realizar dichas tareas, es de gran ayuda disponer de una serie de controles para conseguir el máximo de rendimiento en los entrenamientos diarios. En la actualidad existen algunas soluciones para intentar resolver dichas carencias, sin embargo, no llegan a cubrir todo el espectro de posibilidades que pueda necesitar un deportista, ciñéndose únicamente a un solo deporte. Incluso algunas de ellas se vuelven innecesariamente complejas o recargadas de funcionalidades que no son realmente interesantes para el entrenamiento que se desea realizar.

La motivación inicial para crear un sistema que ayude a solucionar esa falta de soporte nace de mi propia experiencia personal. Muchas veces cuando mantienes un entrenamiento prolongado en el tiempo en un gimnasio te van surgiendo algunas preguntas como, por ejemplo:

- ¿Estoy progresando adecuadamente en cada uno de mis ejercicios o me estoy estancando?
- ¿He hecho alguna vez este tipo de ejercicio? O en caso de llevar mucho tiempo sin hacerlo, ¿cuáles eran mis últimas marcas para no empezar desde más abajo?
- ¿Qué días y porqué me encuentro con más energía o con menos?
- ¿Cómo me está repercutiendo esto en mi salud física? ¿La estoy mejorando? ¿Cuánto? ¿Mantiene unos niveles saludables o la tengo que mejorar?

Cuando en vez de realizar únicamente un tipo de deporte, se realizan dos o más, el problema empieza a crecer y multiplicarse. Uno acaba teniendo que instalarse más de una aplicación, una para cada tipo de deporte que está practicando, sin ofrecer todas ellas funcionalidades similares, haciendo que haya que adaptarse a cada una de ellas por separado.

Es aquí donde nace *FitG*, tratando de resolver dichos problemas intentando crear una solución general y de uso intuitivo para un amplio sector de usuarios, yendo desde los más expertos en tecnologías hasta los menos familiarizados con las mismas.

Por último, ya que este proyecto se plantea como un producto para una empresa real, lo óptimo es que se llevara a cabo por un equipo de desarrolladores. Pero por la propia naturaleza del TFM, cada vez que se haga referencias a “los desarrolladores” o al “equipo de desarrollo” en el documento, éste está integrado únicamente por el autor de este TFM.

1.1 Objetivos

Este TFM tiene como propósito realizar un proyecto de emprendimiento centrado en *FitG*, un sistema web híbrido en varias plataformas que ayuda a la gestión de varios tipos de deportes, de integrados bajo un mismo sistema. Esta idea de producto cuenta con los siguientes subobjetivos:

- Llegar a un público amplio y solucionar los problemas (que más adelante se tratarán en detalle) que los usuarios tienen a la hora de practicar varios deportes, apoyándose en el uso de las tecnologías.
- Ofrecer una solución que junto con la arquitectura y la tecnología ofrezcan un sistema que sea mínimamente escalable.
- Ofrecer una portabilidad en las diversas plataformas existentes, ofreciendo la aplicación en un número alto de sistemas (p.ej., navegadores de escritorio, dispositivos Android, iOS…).

- Conseguir validar financieramente la idea con una mínima proyección económica para poder comercializar el producto de cara al futuro cercano.
- Validar el sistema con usuarios reales, realizando experimentos, antes de ponerla en una primera fase beta o de producción.

Como objetivos personales del autor del TFM, y creador y desarrollador de la idea, se tienen los siguientes:

- Ahondar y coger conocimientos de las últimas versiones que se utilizan en el desarrollo web en el *stack* completo.
- Familiarizarse con el proceso completo del desarrollo de un producto software de emprendimiento de principio a fin siguiendo todas las fases (análisis, diseño, implementación…) que ofrecen las metodologías ágiles y Lean Startup (Letelier, 2018a).

1.2 Estructura del documento

El documento recoge la documentación del proyecto propia de sistemas y desarrollos software en el contexto de emprendimiento. A continuación, se realiza una descripción de lo que contiene cada capítulo:

- En el capítulo 2 se presenta el producto en sí como idea de negocio. También se habla de la materialización de la idea, así como de su modelo de negocio, dando los aspectos claves de la dirección que se tomará en él de cara a un futuro a corto plazo, incluyendo su estrategia de mercado, su competencia y su viabilidad.
- En el capítulo 3 se habla acerca de la metodología de desarrollo empleada, y de cómo se va a gestionar y organizar el proyecto. Además, se establecen una serie estándares que se han de seguir a la hora de desarrollar el proyecto, así como los pasos a seguir para cada fase del desarrollo de la aplicación y el sistema.
- En el capítulo 4 se citan los aspectos iniciales del sistema antes de su desarrollo en profundo. Asimismo, se ven los aspectos de análisis y diseño de la arquitectura que tendrá el sistema en alto nivel, un estudio de las alternativas y la viabilidad de algunas decisiones tecnológicas, la integración de dichas tecnologías en la arquitectura presentada, y algunos aspectos de usabilidad y documentación del sistema.
- En el capítulo 5 se detalla la descripción de cada una de las tres iteraciones en las que se ha desarrollado el sistema. Se incluye tanto el alcance de la iteración, como su desarrollo y sus resultados. Además, se incluye unos resultados finales de la calidad del código y los desafíos técnicos más relevantes durante el desarrollo del sistema.
- En el capítulo 6 se presenta el estado final del sistema, y se detalla una estrategia a seguir para la puesta en marcha del sistema en la nube.
- En el capítulo 7 se presentan las conclusiones y trabajo futuros.

A continuación, se enumeran las referencias bibliográficas consultadas para la realización del presente TFM. Por último, el documento cuenta con un apartado de anexos que son referenciados durante el resto de la memoria para ampliar algunas secciones. En concreto hay 3 anexos. El anexo A contiene el análisis y diseño del sistema. El anexo B describe los detalles de implementación. El anexo C enumera las instrucciones para el despliegue del sistema. El anexo D presenta la API RESTful del sistema.

2. Evaluación de la idea de negocio FitG

2.1 Resumen de FitG

FitG debería ser un sistema centrado en la gestión de las marcas que realizas en los diferentes tipos de ejercicios (como por ejemplo podrían ser de musculación, correr, nada, etc.) y orientado a usuarios que no tengan por qué ser expertos o asiduos en el uso de tecnologías. La aplicación debería contar con dos aplicaciones cliente distintas orientados a funcionalidades y finalidades diferentes. Ambos clientes deberían de tener una interfaz intuitiva y fácil de usar.

Una de las aplicaciones cliente debería de ser para dispositivos móviles, dado que su principal objetivo sería el de introducir las marcas del entrenamiento, y debe ser algo que se pueda hacer durante el mismo ejercicio de una forma cómoda. Se debería de disponer de funcionalidades tales como la gestión de los distintos tipos de ejercicios, entre los cuales se tendrían un conjunto ya predefinidos, además de poder crear nuevos. Dentro de cada uno de los ejercicios existentes, el usuario debería poder añadir sus marcas registradas durante el entrenamiento conteniendo una amplia información que le caracterizase para poder analizarlos posteriormente. También debería de poder buscar un ejercicio de forma rápida y sencilla para poder anotar una nueva marca mientras se realiza el entrenamiento sin perder mucho tiempo. Por último, debería de ofrecer la posibilidad de introducir un análisis con datos acerca de tu salud física (p.ej., el peso, el porcentaje de grasa, la masa visceral, etc.).

La otra aplicación cliente debería de ser para dispositivos de sobremesa, dado que su principal objetivo sería el de analizar los datos recogidos durante los entrenamientos, de forma que se puede realizar más cómodamente en este tipo de dispositivos que en los dispositivos móviles. Este cliente debería disponer de la capacidad para listar todos los ejercicios predefinidos más los que ha creado el usuario. Además, se debería de poder hacer un listado histórico de todas las marcas de cada uno de los ejercicios que ha ido introduciendo el usuario a lo largo del tiempo. Con los datos del listado histórico se deberían de mostrar unas gráficas para que el usuario pueda ver el progreso de una forma más visual sin tener que ir al detalle de cada una de las marcas. Dichas gráficas, junto con el listado histórico, deberían poderse personalizar mínimamente para que el usuario pueda visualizar los datos que le sean de interés. Asimismo, se debería poder mostrar el mismo listado, las mismas gráficas y la misma personalización de éstas, pero en vez de para las marcas de los ejercicios, para los análisis de la salud física. Por último, también debería poderse buscar el ejercicio del cual se desean ver sus datos de una forma rápida y cómoda.

2.2 Modelo de negocio

El modelo de negocio nos da una visión de conjunto del proyecto en la que la solución resultante de este TFM sería una pieza más. De esta forma le damos un marco de referencia al análisis, a los requisitos capturados, a algunas decisiones de diseño, y a aspectos como la valoración del coste de este TFM. La elaboración del modelo de negocio del producto se ha basado en la herramienta desarrollada por *Alex Osterwalder* denominada *Business Model Canvas* (Bernardo, 2013). El modelo consta de 9 campos o bloques que permiten tener una visión integrada del modelo de negocio y se ha representado en la Tabla 1. El número que aparece entre paréntesis en los títulos de cada bloque representa el orden de trabajo en el que se han completado los módulos. Gracias a ese orden es como se consigue refinar y conceptualizar el modelo de negocio.

Tabla 1. Business Model Canvas.

<table>
<tr>
<td rowspan="2">PROBLEMA (2)

Existen hoy en día diversas aplicaciones para ayudar a una gestión para el deporte que se practica. Sin embargo, muchas de ellas no aportan la capacidad de mostrar unas estadísticas/gráficas globales para el análisis del progreso de dichos ejercicios a lo largo del tiempo de una forma intuitiva.
Asimismo, suelen centrarse únicamente en un tipo de deporte o plataforma, lo que incomoda la tarea para quienes practican más de uno teniendo que instalarse varias.
Además, suelen ser poco flexibles a la hora de ofrecer tipos de ejercicios, haciendo que uno se tenga que adaptar a los que contiene la aplicación, muchas veces no encajando en lo que uno practica.
Por otro lado, el progreso de la salud física de una persona es algo que va de la mano cuando alguien realiza deporte. Actualmente hay pocas aplicaciones que permitan la recogida de datos acerca de tu salud física para que puedas analizarlos y comprobar si se mantiene en unos niveles saludables o no.</td>
<td>SOLUCIÓN (4)

Un sistema web híbrido en varias plataformas que ayuda a la gestión de varios tipos de deportes:
Posibilidad de añadir marcas de ejercicios realizados y análisis de estados físicos generando históricos de los mismos.
Consulta de los históricos de forma gráfica para ver su progreso.
Gestión de nuevos tipos de ejercicios por parte del usuario y una cantidad predeterminada ya creada en el sistema.
Búsqueda rápida de los ejercicios a los que se desea añadir una marca.</td>
<td rowspan="2">PROPUESTA DE VALOR (3)

Añadir el progreso de tu actividad de deportiva o tus niveles de salud físicos en unos pocos segundos.
Crea tus propias actividades deportivas si no se te adapta ninguna existente.
Visualiza y controla tu progreso de una forma cómoda e intuitiva.</td>
<td>VENTAJA COMPETITIVA (9)

Ser el primero en el mercado en intentar unificar los problemas bajo un solo sistema.</td>
<td rowspan="2">SEGMENTOS DE CLIENTES (1)

Personas a las que les gusta realizar más de un tipo de deporte, ya sea de forma profesional o de forma amateur, y les gusta mantener un control sobre sus progresos.</td>
</tr>
<tr>
<td>MÉTRICAS (8)

Número de usuarios activos.
Número de descargas de la aplicación.</td>
<td>CANALES (5)
Redes sociales.
Sitio web.</td>
</tr>
<tr>
<td colspan="3">ESTRUCTURA DE COSTES (7)

Unos costes totales aproximados a 5 años de 692.234€ repartidos entre:
Infraestructura Cloud: 10.000€
Oficina y gastos asociados (alquiler, ordenadores, muebles, internet, electricidad…): 25.000€
Marketing: 192.234€
Sueldos de personal (desarrolladores, técnicos soporte, CEO, etc.): 465.000€</td>
<td colspan="3">FLUJO DE INGRESOS (6)

Modelo de negocio basado en el tipo freemium.
Cuentas de usuarios Premium a 1,99€/mes.
Objetivo a 5 años: 115.000 usuarios Premium dando un total anual de 2.746.200€</td>
</tr>
</table>

2.3 Análisis DAFO

En este apartado se realiza un análisis acerca de factores internos o externos que puedan influir y estar relacionados con el proyecto. Para ello se ha utilizado la técnica conocida como la matriz DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) en la que se identifican como factores internos las debilidades y las fortalezas, y como factores externos las amenazas y las oportunidades.

A partir de la información recogida en la Tabla 2 se han de tomar las decisiones estratégicas correspondientes para conseguir aprovechar al máximo los puntos favorables y que los que no lo sean no conduzcan el proyecto al fracaso.

Tabla 2. Matriz DAFO.

Debilidades	**Fortalezas**
• Falta de experiencia en gestión. • Recursos económicos y de personal muy limitados. • Incapacidad de financiar el proyecto sin recurrir a terceros. • Especialización del producto.	• El producto tiene disponibilidad en amplio número de plataformas. • El producto cuenta con una rápida integración en nuevas modalidades de deportes pudiendo personalizar las actividades. • El equipo cuenta con flexibilidad organizativa.
Amenazas	**Oportunidades**
• Aumento de una tendencia hacia la integración de más de un deporte bajo la misma aplicación entre los competidores. • Cambios en los hábitos, necesidades y gustos de los consumidores.	• El aumento del interés en actividades deportivas.

2.4 Estudio de mercado

Dado que se trata de una idea que trata de solucionar un problema e innovar sobre él, es muy importante conocer el mercado, determinando qué competidores y productos están disponibles y comparar sus características con las de *FitG*.

Se ha realizado un análisis de esos posibles competidores. En la siguiente Tabla 3 se encuentra una comparativa de las características de todos ellos, junto con las de *FitG*. Además, a continuación, se hace una breve descripción de todos ellos:

MySSI[1] (SSI Scuba Schools International): Aplicación que permite mantener un control del progreso en las actividades relacionadas con el submarinismo. Además, cuenta con un componente social para que puedas comparar tus progresos con los de otras personas.

Ski Tracker[2] (EXA Tools): Aplicación centrada en el sector de los amantes de los deportes de invierno y la nieve. En concreto para los esquiadores y los *snowboarders*. Realiza unas métricas precisas de los descensos a las pistas, así como un seguimiento en el mapa del recorrido realizado.

[1] https://play.google.com/store/apps/details?id=com.divessi.ssi&hl=es
[2] https://play.google.com/store/apps/details?id=com.exatools.skitracker

CycleDroid[3] (Michal Marschall, 2014): Aplicación centrada en el seguimiento en vivo de las rutas y sesiones de ciclismo. Pensado para poder compartir en redes sociales las rutas y mantener un seguimiento en vivo por el mapa durante el transcurso de la sesión.

Runtastic[4] (Runtastic): Aplicación que permite registrar un seguimiento en vivo principalmente de actividades aeróbicas como correr, caminar, ciclismo, etc. Tiene un fuerte componente de interacción social permitiendo que la gente te apoye y te dé ánimos durante tu sesión, además de poder ver tu desplazamiento en vivo.

Finess & Bodybuilding[5] (VGFIT LLC): Aplicación que ayuda al seguimiento y control de ejercicios musculares. Ofrece visualización temporal del progreso, así como pequeñas descripciones de los distintos ejercicios.

Tabla 3. Comparativa entre los posibles competidores.

Características	**MySSI**	**Ski Tracker**	**CycleDroid**	**Runtastic**	**Fitness & Boodybuilding**	**FitG**
Guardado histórico de marcas del entrenamiento realizado	Sí	Sí	Sí	Sí	Sí	Sí
Añadir comentarios en las marcas del entrenamiento realizado	No	No	No	Sí	No	Sí
Guardado histórico de análisis físico	No	No	No	No	No	Sí
Gestionar nuevos tipos de ejercicios	No	No	No	No	No	Sí
Poder realizar una búsqueda rápida de un ejercicio determinado	No	No	No	No	No	Sí

[3] https://play.google.com/store/apps/details?id=com.maral.cycledroid
[4] https://play.google.com/store/apps/details?id=com.runtastic.android
[5] https://play.google.com/store/apps/details?id=softin.my.fast.fitness

Características	MySSI	Ski Tracker	CycleDroid	Runtastic	Fitness & Boodybuilding	FitG
Gráficas para el análisis del progreso de las marcas	No	Sí	Sí	No (pero información detallada de gráficas por cada marca sí)	Sí	Sí
Personalizar la forma de visualizar los históricos de las marcas	No	No	No	No	No	Sí
Gráficas para el análisis del progreso de los análisis	No	No	No	No	No	Sí
Personalizar la forma de visualizar los históricos de los análisis	No	No	No	No	No	Sí
Consejos e información acerca de cada métrica de los análisis	No	No	No	No	No	Sí
Seguimiento en vivo del entrenamiento	Sí	Sí	Sí	Sí	No	No
Noticias relacionadas con el deporte	Sí	No	No	Sí	No	No
Interacción social	Sí	Sí	Sí	Sí	No	Sí
Sistema de niveles/logros para medir el progreso	Sí	No	No	No	No	Sí
Tutoriales acerca del entrenamiento	Sí	No	No	Sí	Sí	No

Características	MySSI	Ski Tracker	CycleDroid	Runtastic	Fitness & Boodybuilding	FitG
Exportar/Importar datos a formatos externos de la aplicación	No	No	Sí	No	No	Sí
Deportes	Submarinismo	Esquí/*Snowboard, y patinaje.*	Ciclismo	Ciclismo, correr, patinaje, y nadar.	Entrenamiento muscular	Ejercicios aeróbicos y anaeróbicos.
Idiomas	30+	1	14	15	Desconocido	2
Plataforma	Android	Android	Android	Android e iOS.	Android e iOS.	Android, iOS, Windows Phone, y Web.
Modelo de negocio	Necesidad de pertenecer al colectivo.	Publicidad, y opción premium (1,99€)	Donación	Publicidad, y opción premium (9.99€/mes)	Publicidad, y opción premium (3.99€/mes)	Opción premium (1,99€/mes)
Satisfacción de los clientes: 0: Nada satisfecho 5: Muy satisfecho	3,2	4,4	4,4	4,5	4,6	N/A
Descargas actuales (En dispositivos móviles)	10.000+	100.000+	500.000+	10.000.000+	5.000.000+	N/A

Para obtener el análisis de estas características que tienen los competidores se ha realizado una búsqueda de aplicaciones relacionadas en la App Store de Android. Una vez se han encontrado un número considerable de las mismas, se han analizado una a una descargándolas, probándolas, buscando comentarios acerca de ellas e información en sus páginas web.

Para analizar el valor que aporta cada característica al producto se ha utilizado el modelo Kano (Zacarias, s.f.). Este modelo proporciona una visión de las características del producto ayudando a posicionarlas en función de la satisfacción y el rendimiento del servicio que percibirán los usuarios siguiendo el esquema de la Figura 1.

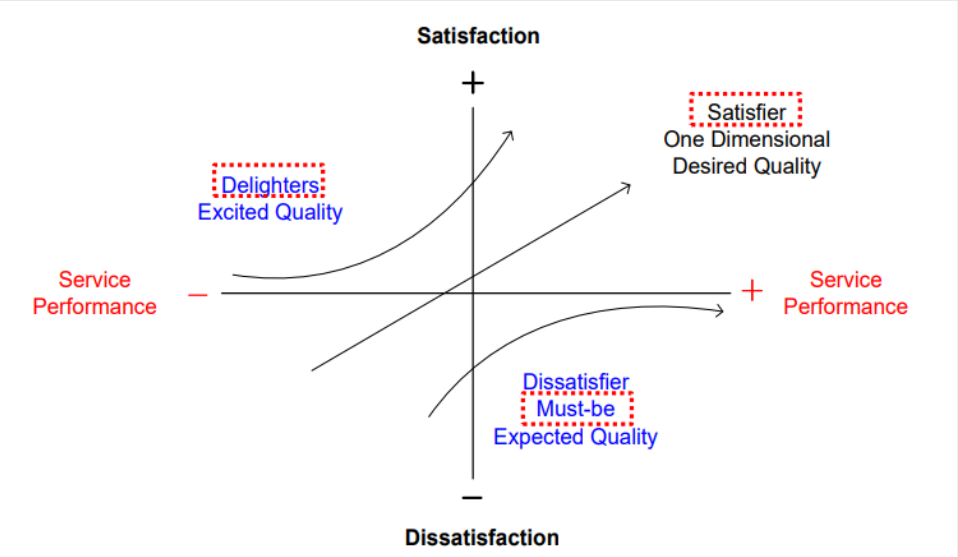

Figura 1. Modelo Kano (consultado en (Letelier, 2018b)).

Utilizando este modelo, se ha podido observar como todas las características relacionadas con la búsqueda, la personalización de ejercicios, la gestión de marcas de los análisis físicos, y personalizar las gráficas de progreso se considerarían en el cuadrante donde tendrían una satisfacción muy positiva, siendo estas características exclusivas. Por otro lado, la característica de interacción social no se tenían pensada incluirla, pero dado que el resto de los competidores sí que la tienen y no hay una justificación para no incluirla en el producto, se ha clasificado como característica del cuadrante "*Must-be*". Sin esta característica se considera que corres el riesgo de decepcionar al cliente si no está incluida en el producto. Por último, el resto de las características se considera que estarían en un equilibrio lineal positivo tanto en satisfacción como en rendimiento.

2.5 Proyección económica a 5 años

Para obtener la proyección económica de aquí a 5 años se han realizado unos cálculos aproximados del presupuesto, gastos e ingresos. Para ello, se ha tenido en cuenta el material y personal que se prevé que se utilizará durante el desarrollo y mantenimiento del producto. Estos gastos se pueden ver recogidos en la Tabla 4 y a continuación se van a detallar los más importantes (que no son genéricos y dependen más de cada proyecto en específico).

En el apartado de ingresos se va a seguir un modelo de negocio tipo *freemium* como se indicaba en la Tabla 1 (Lean Canvas). Se ha establecido que este tipo de modelo de negocio sea de un precio de 1,99€/mes. Este precio sale de intentar bajar el precio que ponen a sus productos los competidores siendo aún rentable el producto. Esto da lugar a 24€ anuales por cada licencia que se mantenga activa los 12 meses del año. Por otro lado, respecto al número de licencias vendidas, se ha estimado que mínimo se debería de llegar al final de los 5 años a un 1% del total de usuarios con los que cuentan las aplicaciones más directas en los competidores, es decir, de un total de 15.500.000 usuarios (10.000.000 de *Runtastic*, 5.000.000 de *Fitness & BodyBuilding*, y 500.000 de *CycleDroid*).

Con respecto al marketing, se han realizado los cálculos aproximados sacados de un estudio (Leone, 2017) en el que se obtiene que para las empresas orientadas a los servicios "*Business-to-Consumer*" se gastan en torno a un 7% de sus ingresos anuales. Además, dan una pista de que actualmente y de cara a los próximos años dicho presupuesto tiene que ser destinado más a los canales digitales de publicidad y marketing como Facebook o Google Ads, que a los canales tradicionales de transmisión.

Por otro lado, los empleados al final de los 5 años serán un total de 11 (1 Chief Executive Officer, Chief Marketing Officer, Chief Financial Officer, Chief Technology Officer y desarrollador senior, 4 técnicos de soporte, y 2 administrativos). Al principio del proyecto y durante el primer año, únicamente se contará con 1 director que hará las veces de CEO y CTO (ya que no se manejan grandes cifras de momento), y 2 desarrolladores senior (para el desarrollo inicial del sistema). En el segundo año se incorporarán dos directores más (obteniendo así 1 CEO, CTO y CMO), un técnico de soporte y un administrativo. Estos cambios se deben a que se empiezan a manejar más cifras respecto a las ventas del producto, ascendiendo a 10 veces más que el año anterior. Durante el tercer año, se incorporará un CFO (puesto que se empiezan a manejar cifras de dinero más altas), se prescindirá de uno de los desarrolladores senior (ya que el sistema estará desarrollado casi al completo y solo serán necesarias labores de mantenimiento y pequeñas mejoras), y se incluirá otro técnico de soporte y otro administrativo más. En los años 4 y 5 se mantienen todos los puestos, únicamente aumentando un técnico de soporte por año dado el incremento en usuarios del sistema.

Por último, se han realizado algunos cálculos aproximados para los aspectos relacionados con la inversión inicial. Según los cálculos realizados, se necesitaría una inversión aproximada de unos 200.000€. De esta forma se cubrirían los 2 primeros años hasta que en el 3º, se recupere la inversión (aproximadamente en el mes de mayo como se puede ver en la Figura 2).

Un inversor, normalmente espera multiplicar por 7-10 veces su inversión inicial en una startup a los 5 años (Patel, 2016), para ello, de forma rápida y aproximada de comprobar si merece o no la pena invertir es calcular con la fórmula *EBITDA* (Lorenzana, 2013) el resultado anual acumulado y multiplicarlo por 10, de esa forma se obtiene un valor aproximado de la empresa para saber si la inversión podría ser o no rentable. En este caso, realizando dichos cálculos se obtendría que la empresa tendría un valor de unos 36.882.660€. Si calculamos que el inversor quiere recuperar 10 veces la inversión inicial, es decir 2.000.000. Esto supondría que, además de que podría recuperarlo, tan solo haría falta darle entre un 5-6% de participación de la empresa.

Figura 2. Resultado anual respecto a los beneficios a 5 años.

Tabla 4. Proyección económica a 5 años detallada.

	Años				
Ingresos	**1**	**2**	**3**	**4**	**5**
Nº de licencias vendidas[6]	1.000	10.000	40.000	80.000	115.000
Total de ingresos (nº de licencias vendidas * precio anual *freemium*):	**23.880€**	**238.800€**	**955.000€**	**1.910.400€**	**2.746.200€**
Gastos anuales					
Infraestructura Cloud[7]	5.000€	5.000€	5.000€	10.000€	10.000€
Ordenadores e impresoras[7]	6.000€	4.000€	3.000€	3.000€	3.000€
Marketing	1.672€	16.716€	66.864€	133.728€	192.234€
Alquiler oficina, muebles oficina e instalaciones.[7]	15.000€	14.000€	14.000€	14.000€	14.000€
Internet, electricidad, agua, teléfono, etc.[7]	3.000€	3.500€	4.000€	4.500€	5.000€
Gestoría[7]	2.000€	2.000€	3.000€	3.000€	3.000€
CEO[7]	40.000€	40.000€	50.000€	60.000€	60.000€
CTO[7]	0€	40.000€	50.000€	60.000€	60.000€
CMO[7]	0€	40.000€	50.000€	60.000€	60.000€
CFO[7]	0€	0€	40.000€	50.000€	60.000€
Desarrolladores Senior[7]	80.000€	80.000€	45.000€	45.000€	45.000€
Técnicos de soporte[7]	0€	25.000€	60.000€	90.000€	120.000€
Administrativos[7]	0€	25.000€	60.000€	60.000€	60.000€
Total Gastos	**152.672€**	**295.216€**	**452.864€**	**593.228€**	**692.234€**
Resultados					
Resultado Anual	**-128.792€**	**-56.416€**	**502.336€**	**1.317.172€**	**2.053.966€**
Resultado Anual Acumulado	**-128.792€**	**-185.208€**	**317.128€**	**1.634.300€**	**3.688.266€**

[6] Se ha seguido una aproximación pesimista a la hora de establecer una hipótesis de cuántos usuarios habrá cada año hasta llegar a los 115.000 en el quinto. Se ha establecido que en el primer año solo se conseguirá cerca del 1% de los usuarios deseados, el segundo año cerca del 10%, el tercer año entre un 30 y 40%, el cuarto alrededor de un 70%, y finalmente el total de los usuarios en el quinto año. Esto se ha hecho debido a que en el contexto de las *startups* hay una alta posibilidad de fracasar, es por eso que se ha querido que la curva de recuperar la inversión sea más lenta, para que de esta forma, si los cálculos y cuentas son positivas y cuadran (como en este caso), se tenga una mayor confianza en el producto a la hora de buscar inversores.

[7] Datos aproximados (de históricos y ejemplos) ofrecidos en la asignatura de Emprendimiento en Productos Software, del Máster Universitario en Ingeniería y Tecnología de Sistemas Software de la Universidad Politécnica de Valencia.

3. Gestión del proyecto

Dado que se trata de un proyecto con el plazo fijado, pero costes y alcance variables, se ha utilizado una metodología con una aproximación ágil, la cual se describe en la sección Metodología de desarrollo. Para garantizar que no se pierde calidad en el producto, se han establecido unas "Definiciones de hecho y estándares del proyecto" que se han de comprobar para cada funcionalidad implementada, así como realizado un esfuerzo extra en técnicas de calidad, pudiéndose ver en la sección "Calidad del producto". En la sección "Planificación del proyecto", se ha establecido una planificación del proyecto basada en iteraciones debido a la aproximación ágil ya comentada. Por último, en la sección "Esfuerzos distribuidos por tiempo y actividad" se recogen sin embargo todos los esfuerzos realizados en el proyecto y distribuidos por tiempo y actividad, para un mejor análisis de la gestión y mejoras en futuras iteraciones a la hora de estimar las tareas.

3.1 Metodología de desarrollo

En este apartado se describe la metodología de desarrollo y el proceso de implementación. En cuanto al proceso de calidad y mantenimiento seguida durante el desarrollo del producto, se han utilizado algunas herramientas, tecnologías y metodologías para poder conseguirlo. Ya que se trata de una aproximación ágil, se ha aplicado la metodología de integración continua (Fowler, 2006) (como se puede ver en la Figura 3), integrando y construyendo todo el código constantemente cada vez que se realiza un *commit*, pasando los tests correspondientes, y permitiendo que cada poco tiempo se puedan hacer entregas con un valor potencial para el cliente teniendo siempre un producto mínimo viable.

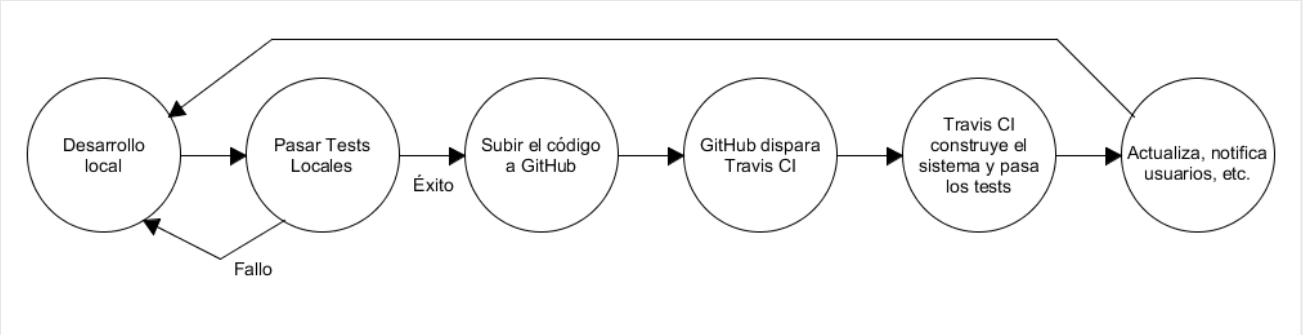

Figura 3. Diagrama de alto nivel del proceso seguido de integración continua.

Dado que la aplicación se ha desarrollado empleando el *framework* MEAN (Raj, 2014), una de las estrategias de construcción automática empleada ha sido el gestor de paquetes npm (npm, Inc, s.f.). Gracias a que se pueden configurar numerosos comandos en el fichero package.json del proyecto, npm facilita la ejecución de la aplicación con un solo comando: 'npm start'.

Además, puesto que su función principal es la gestión de paquetes y dependencias del proyecto, npm cuenta con el comando 'npm install' que descarga automáticamente todas las dependencias necesarias para el proyecto definidas en el package.json. Asimismo, cuenta con el comando 'npm test', el cual ejecuta todos los tests unitarios y de integración de mocha que se encuentren disponibles.

Por otro lado, en la parte del cliente de Ionic 2, dado que este framework cuenta con su propio CLI (Ionic Framework, s.f.), existen algunos comandos que se necesitan saber. 'ionic serve' se utiliza para empezar un servidor local de desarrollo para el desarrollo/testing de la aplicación. 'ionic cordova build [<plataform>] construye (prepara + compila) el proyecto de Ionic para una determinada plataforma, pudiendo ser iOS, Android o Windows. 'ionic cordova run [<plataform>]' ejecuta el proyecto de Ionic en un dispositivo conectado. 'ionic cordova emulate [<plataform>]' emula el proyecto de Ionic en un emulador o simulador. Estos son los comandos más importantes que se han utilizado durante el proyecto para la construcción automática del cliente para dispositivos móviles, ya que como se ha explicado con anterioridad, toda la dependencia de paquetes se ha instalado previamente.

La importancia de agilizar el proceso de desarrollo del proyecto es fundamental siguiendo este tipo de metodología. No obstante, no hay que dejar de lado un buen control de versiones. Se ha decidido utilizar la herramienta GitHub creando un repositorio llamado "FitG-TFM" (https://github.com/nebur395/FitG-TFM) y siguiendo un flujo de trabajo del tipo "*Integration-manager workflow*" (Git) modificado. De esta forma se posee un repositorio central al cual no se puede realizar un *push* directo (salvo en excepciones y situaciones extremas) hasta haber pasado localmente todos los test correspondientes. Los cambios del proyecto se van realizando en local. Cuando se quiere integrar algo lo suficientemente interesante y estable al repositorio central, se realiza el *push* correspondiente. Una vez hecho, se comprueba que se han pasado con éxito los tests automáticos de TravisCI (TravisCI, s.f.) y todas las definiciones de hecho establecidas.

Así se consigue mantener una frecuencia de trabajo rápida, una rama en el repositorio central lo más estable posible, y un control mínimo de que todo el trabajo integrado en la misma ha pasado una revisión y posee un grado de calidad aceptable.

Para la organización de los documentos y de la gestión del proyecto se ha usado la plataforma de Google Drive, en la cual se ha mantenido un seguimiento de los documentos que se han ido actualizando. Se ha creado una organización en el directorio para que se pueda realizar una gestión adecuada del mismo siguiendo la siguiente estructura:

Tabla 5. Estructura del directorio creado en Google Drive.

1_Gestión	Temas relacionados con la gestión del proyecto.
1.01_Hoja de esfuerzos	Hojas de esfuerzos y control de horas del TFM.
1.02_Planificación	Planificación del proyecto.
2_Análisis y diseño	Temas relacionados con todo el análisis y diseño del proyecto.
2.01_Diagramas	Diagramas del análisis y diseño del sistema.
2.01_Mapas de navegación	Mapas de navegación de la UI del sistema.
3_Documentación	Temas relacionados con toda la documentación del proyecto, incluidas las entregas finales.
4_Presentaciones	Presentaciones que se van a realizar (en este caso la defensa del TFM).
5_Multimedia	Contenidos varios del TFM, mayormente multimedia.
Z_Cosas	Otros.

El proceso de implementación de las funcionalidades de la solución se ha realizado siguiendo un proceso determinado (el cual se puede ver en la Figura 4):

- Una vez introducidas las funcionalidades en el mapa de características, se creaba una *issue* para cada una de ellas en el *Version Control System* de GitHub (ver Figura 5). A dicha *issue* se le asignaba un tamaño estimable medido en puntos de historia (para dicha medición se ha utilizado la escala más común, una secuencia de Fibonacci modificada: 1,2,3,5,8,13,20,40,100, o tallas de camiseta si de momento es lejana y no se quiere estimar en detalle). Una vez creada, se escribían los criterios de aceptación/satisfacción en su descripción.
- Antes de cada sprint y una vez se tenían introducidas las funcionalidades al *VCS*, se establecía en el mapa de características cuáles de ellas formarían parte de cada producto mínimo viable (*MVP*), con sus dependencias incluidas. Se creaba una *milestone* en el *VCS* de GitHub correspondiente a ese *MVP* y se le asignaban las *issue* correspondientes. Además, se creaba un tablero *Kanban* correspondiente a dicha *milestone*/*MVP* y se añadían todas las *issues* asociadas a la columna de TODO.
- Una vez asignadas y añadidas al tablero *Kanban* de GitHub, se ponían ordenadas por prioridad en la columna TODO hasta que se empezaba a trabajar en ella y se movía a la columna de DOING (como se puede ver en la Figura 6).

Además, cada uno de los dos procesos, contaba con unos pasos comunes:

- Cada vez que se finalizaba el desarrollo de una funcionalidad se establecía una batería de tests para ella, que a partir de entonces se tendría que integrar y pasar cada vez que se realizara cualquier cambio siguiendo la práctica de integración continua, para comprobar en el futuro que no hay nada que pueda introducir fallos en dicha funcionalidad.
- Habiendo completado la batería de tests correspondientes a la funcionalidad, se hacía un escaneo de la herramienta de SonarQube (SonarQube, s.f.) para comprobar si los niveles de calidad y mantenibilidad del código pasaban las reglas establecidas, y en caso de que no los pasaran corregir los problemas añadidos.

Una vez finalizado todo el proceso anterior, se daba por concluida la funcionalidad correspondiente cerrando la *issue* y moviéndola a la columna de DONE en el tablero Kanban.

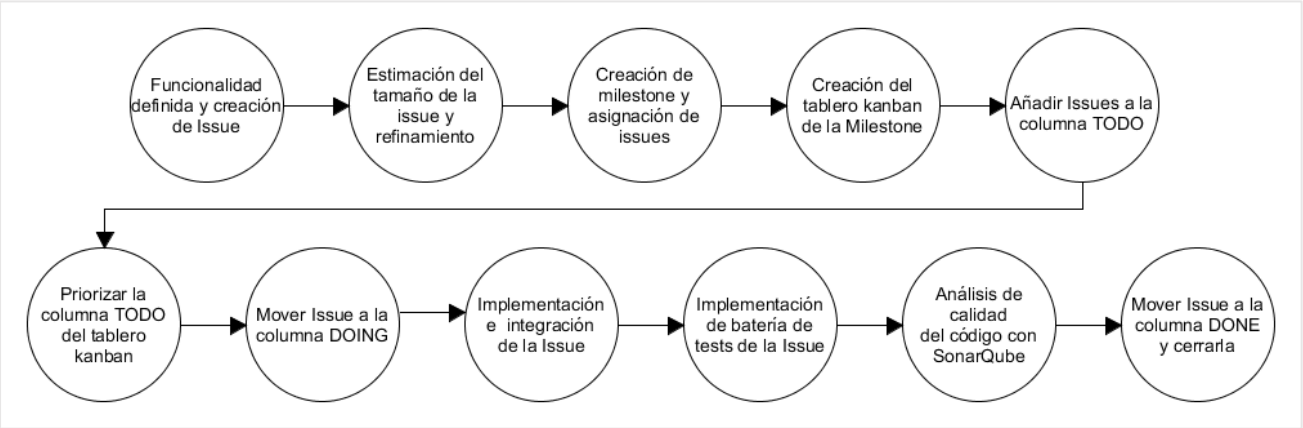

Figura 4. Proceso seguido durante la implementación del sistema.

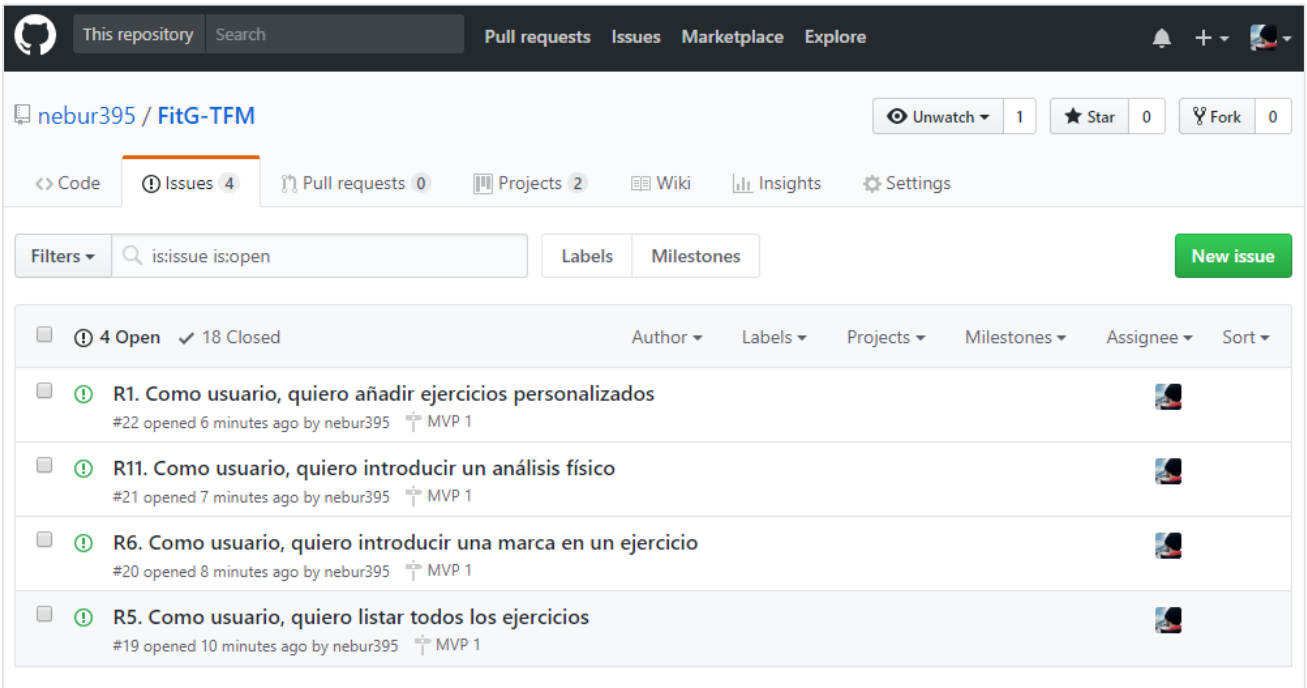

Figura 5. Pila del producto representado con Issues usando Github.

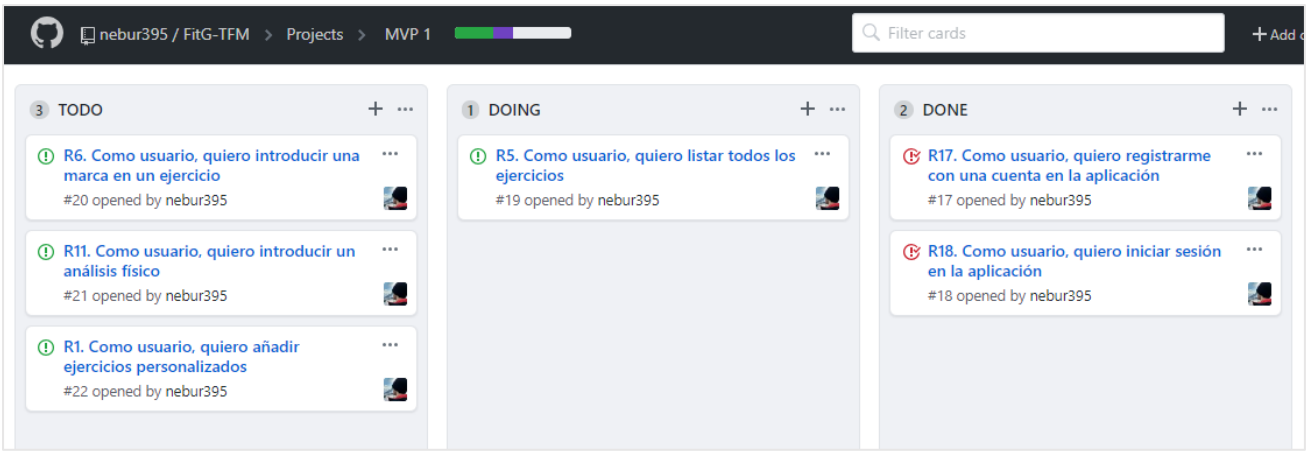

Figura 6. Kanban del proyecto usando GitHub.

3.2 Definiciones de hecho y estándares del proyecto

En este apartado se definen las definiciones de hecho y los estándares usados en la aplicación *FitG*. Para la codificación, las guías de mejores prácticas se han basado en los estándares de HTML y CSS (Google, s.f.a), y los de JavaScript (Google, s.f.b). Además, todas las funcionalidades del sistema tienen que cumplir las siguientes definiciones de hecho para que se consideren completadas correctamente

- Se pre-verifican los criterios de satisfacción.
- Se verifica que el sistema adaptará la GUI a la pantalla en la que se visualice la aplicación.
- El producto ha sido incluido en la rama master del repositorio del *owner* y supera todos los test unitarios y de integración automáticos con TravisCI.
- El código del producto sigue los estándares de codificación basados en las guías de mejores prácticas de HTML, CSS y JavaScript.
- Los nombres de las variables, métodos y clases tienen que ser mínimamente descriptivos, así como guardar una consistencia semántica entre las distintas capas de implementación si se refieren al mismo concepto.
- La documentación del proyecto debe estar redactada en inglés.
- Toda función o método que no sea un método de acceso a atributos debe tener una breve explicación de la descripción de su funcionamiento o ser tan sencilla que baste con un título autodescriptivo.
- Los métodos de acceso a atributos tipo "*getAtributo*" o "*setAtributo*" pueden prescindir de una explicación si poseen un nombre intuitivo.

3.3 Calidad del producto

Para asegurar que el producto tiene una calidad adecuada, se estableció una batería de tests automáticos. (los cuales se disparan en cada integración del software gracias a la herramienta de TravisCI siguiendo las pautas de la integración continua), con la herramienta SonarQube, que es un analizador estático de código, para ayudar a hacer un software de mayor calidad que consiga tener unos niveles altos de mantenimiento, y con la tecnología de EditorConfig (EditorConfig, s.f.), una herramienta que automatiza el estilo de la codificación para conseguir un código consistente y uniforme.

En la verificación y validación de la aplicación se han realizado tests unitarios, y de integración. Para ellos, se ha empleado la herramienta Mocha (Mocha, s.f.), un *framework* de testing para Node.js (Node.js Foundation, s.f.). Junto con Mocha, a fin de contar con aserciones para realizar los mencionados tests, se ha empleado también Chai (ChaiJS, s.f.), una librería de aserciones, también para Node.js. Con la integración de ambas

herramientas se han conseguido implementar varias suites de tests para las diferentes funcionalidades de la aplicación.

En relación con la cobertura de los tests automáticos que se han mencionado, se ha decidido integrar en el proyecto una nueva herramienta que permite, mediante un análisis del código, mostrar en cada estado del proyecto la cobertura del código en ese momento. Dicha herramienta es Codecov (Codecov, s.f.), la cual se ha integrado con GitHub, dándole acceso al repositorio para realizar el análisis tras cada *commit* y *pull request*. De esta forma, Codecov muestra si la cobertura del código ha aumentado, descendido o si sigue igual tras añadir el contenido que se incluya en dicho *commit* o *pull request*.

Por último, como se comentaba al comienzo de esta sección, se ha decidido integrar la herramienta de *Quality Assurance* y análisis estático del código SonarQube con el proyecto, para mejorar la misma y la mantenibilidad del código. Para la configuración de la herramienta se ha hecho uso principalmente de dos módulos llamados Sonar Way: uno para los perfiles de calidad de Web, y otro para los perfiles de calidad de JavaScript. Dado que ya se cuenta con una herramienta de análisis de cobertura de código, se ha desactivado esa sección de la herramienta.

Para el análisis del código se han activado las reglas que recomienda SonarQube en su página para cada uno de los módulos utilizados, estas se pueden ver en los siguientes enlaces:

- **JavaScript**: *https://sonarqube.com/coding_rules#qprofile=js-sonar-way-56838|activation=true*
- **Web**: *https://sonarqube.com/coding_rules#qprofile=web-sonar-way-50375|activation=true*

3.4 Planificación del proyecto

El proyecto ha seguido una aproximación ágil. Además, dado que se trata de un proyecto de emprendimiento se ha situado y guiado también por el marco que ofrece Lean Startup. Esto ha hecho que se haya dividido el desarrollo en iteraciones para poder conseguir cuanto antes un producto mínimo viable, pudiendo realizar experimentos con usuarios, y tener siempre algo de valor para poder trabajar sobre él, ampliarlo poco a poco y en un caso extremo, incluso tener algo que poder entregar como producto final recortando alcance y/o calidad.

Además, esta aproximación que se ha seguido se adapta a la situación actual, ya que el desconocimiento de las tecnologías o poca experiencia trabajando con ellas, además de tener los plazos fijados, pero el alcance y calidad variables, ha ayudado a que el análisis de requisitos y diseño del sistema vaya evolucionando a la vez que el proyecto, refinándose así los anteriores continuamente hasta el final de la aplicación.

A lo largo del desarrollo del producto se han planteado dos primeras iteraciones destinadas a realizar los dos primeros productos mínimos viables o *MVPs*. Cada una de las dos iteraciones ha consistido en el desarrollo de un grupo de funcionalidades que se considera que necesitan ser validadas por usuarios de forma prioritaria, seguido de un experimento con personas reales para recoger la retroalimentación y validación necesaria. Dado que se trata de procesos que tienen como prioridad la validación de la idea, en ellos se han prescindido de algunas de las guías establecidas en los puntos anteriores, en concreto se ha invertido poco esfuerzo a la estimación y planificación de las características del mapa de producto, y en los test de regresión realizados durante la integración continua.

Después de estas dos primeras iteraciones, se ha realizado una tercera, en la cual se ha volcado el mapa de características restante a implementar en una pila de producto y aplicado ya todos los procesos definidos, para realizar un primer acercamiento a una normalización en *sprints* de cara al lanzamiento y mantenimiento del producto. En concreto, este sprint se ha focalizado más en la corrección de fallos y detalles, que en la creación de nuevas características. También se ha invertido esfuerzo como se decía, en dotar de la calidad deficiente de las iteraciones anteriores para todo el sistema.

Para poder representar de forma gráfica las funcionalidades del sistema y poder establecer mejor una planificación del proyecto con respecto a las iteraciones y los *MVP*, se ha realizado un mapa de características (Figura 7). Este mapa nos permite detectar dependencias entre las funcionalidades y priorizar sobre qué

funcionalidades deben implementarse en cada iteración. Estas prioridades van ordenadas según lo que un usuario potencial querría que la aplicación tuviera y no con respecto a lo que el sistema necesita. Además, dichas características se han agrupado en función de a qué grupo global pertenecerían.

A continuación, se va a describir el esquema temporal que ha tenido el proyecto. En la Figura 8, se puede ver el diagrama temporal resultante del proyecto, mientras que en la Tabla 6 se tiene una descripción un poco más detallada.

Tabla 6. Planificación a priori: Lanzamientos e iteraciones del proyecto.

Lanzamiento del proyecto	**2018/01/01 - 2018/03/31**
Reuniones de lanzamiento del proyecto en las que se decide la motivación de la idea, una investigación del trabajo existente, un análisis del producto con respecto al mercado, una proyección económica, una planificación, y una investigación de las diferentes tecnologías a usar. Análisis y diseño inicial del proyecto, incluyendo la maduración del plan de producto, la captura inicial de requisitos, y la implementación de la infraestructura del proyecto.	
Primer lanzamiento – Primera iteración	**2017/04/01- 2017/04/29**
Desarrollo del 1º *MVP*	
Primer lanzamiento – Segunda iteración	**2017/04/30 - 2017/05/20**
Desarrollo del 2º *MVP.*	
Primer lanzamiento – Tercera iteración	**2017/05/21 - 2017/06/10**
Desarrollo del 1º *sprint.*	
Finalización del proyecto TFM	**2017/06/11 - 2017/07/01**
Finalización de redacción de la memoria y el proyecto, dando los toques finales al mismo.	

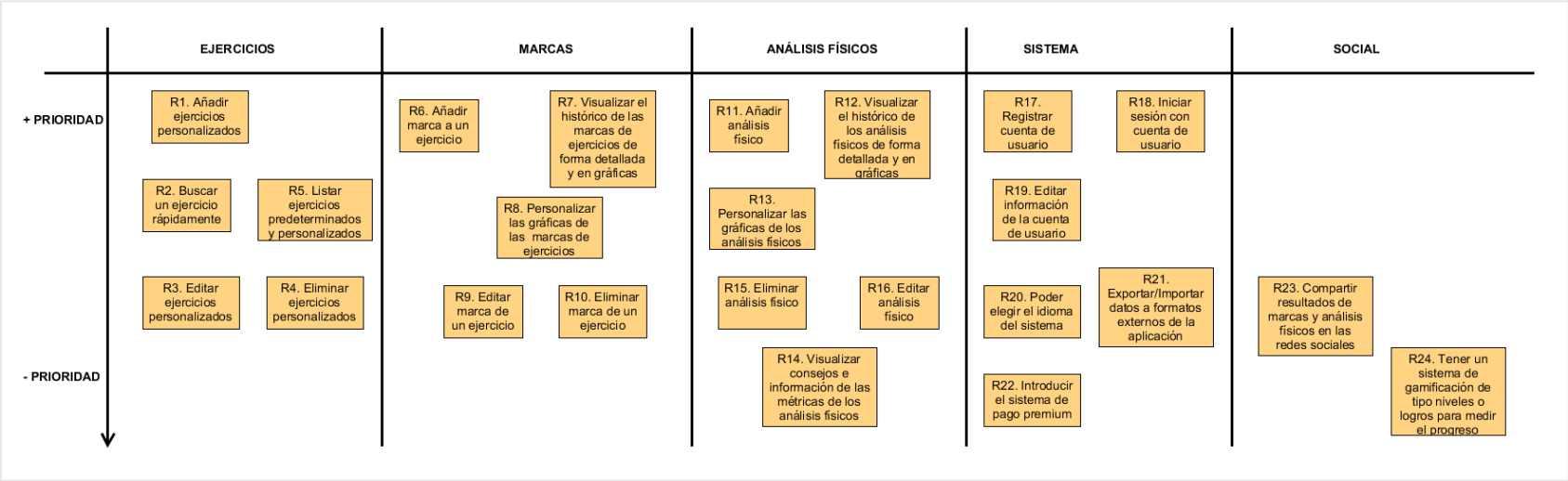

Figura 7. Mapa de características inicial.

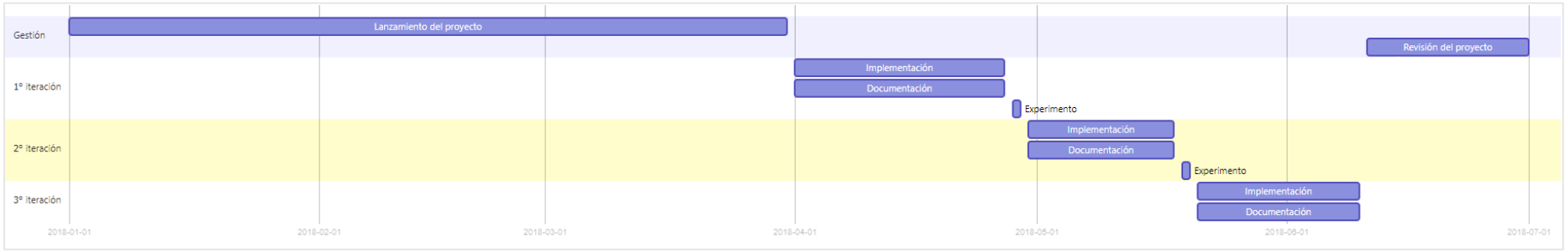

Figura 8. Diagrama Gantt del TFM.

3.5 Esfuerzos distribuidos por tiempo y actividad

El proyecto desarrollado ha tenido una duración de 401 horas, 366 de ellas de desarrollo, y 35 de formación en nuevas tecnologías. Se ha llevado a cabo una recopilación de los esfuerzos realizados en horas en cada una de las actividades distribuidos a lo largo de la duración del proyecto. Dicha recopilación se puede ver en la Tabla 7 y en la

Tabla 8 que contienen el desgranado de las horas por cada mes y actividad respectivamente.

Tabla 7. Distribución de horas por meses.

	Actividades	Formación	TOTAL
Febrero	0	29	**29**
Marzo	45	0	**45**
Abril	133	6	**139**
Mayo	101	0	**101**
Junio	75	0	**75**
Julio	14	0	**14**
TOTAL	**368**	**35**	**403**

Tabla 8. Distribución de horas por actividades

Análisis y diseño del sistema	21
Desarrollo	125
Pruebas	17
Documentación y gestión del proyecto	186
Gestión de configuraciones	5
Quality Assurance	14

4. Diseño del producto

El desarrollo del producto ha sido orientado pensando en qué características eran las más importantes de cara a los usuarios objetivos, y el funcionamiento del sistema. Además, se han intentado llevar unos estándares de calidad con respecto al código y el proyecto, de forma que el proceso de desarrollo fuera lo más automático posible, además de reproducible y con unos niveles de mantenimiento y calidad adecuados.

Los objetivos mencionados en el párrafo anterior han sido determinantes a la hora de los estudios de viabilidad tecnológica de la sección Estudio de alternativas y viabilidad, y del desarrollo de las tecnologías y la arquitectura escogidas, así como de la solución final. Estos objetivos principalmente eran la usabilidad, la escalabilidad, y el rendimiento. En la sección Arquitectura propuesta, se tiene una visión gráfica y descriptiva de alto nivel de la arquitectura del sistema. En Solución tecnológica, se hace una descripción de los aspectos de escalabilidad y rendimiento, que han dado lugar a la solución actual con sus diferentes tecnologías. En Usabilidad de la solución se detallará la usabilidad, su proceso y los principios de diseño en los que se ha basado la solución. Por último, en Documentación de la API se detalla cómo se ha documentado la API durante el desarrollo del proyecto y cómo se pretende abrir hacia el público.

4.1 Arquitectura propuesta

Como resultado de los requisitos se tiene que el proyecto tiene que implementar una aplicación web con dos clientes, uno para dispositivos de escritorio y otro para dispositivos móviles. Ya que se trata de una aplicación que no trata ningún problema en específico ni necesita de una arquitectura especial, se ha aplicado el patrón de diseño arquitectural Modelo-Vista-Controlador, implementando el sistema en capas de abstracción, con todas las ventajas que ello supone, pudiéndose ver una aproximación de muy alto nivel en la Figura 9. Además, se ha intentado simplificar el modelo de datos al máximo posible, a la vez que intentando emularlo con el dominio real del problema como refleja la Figura 10 en un nivel muy alto. Toda la información detallada de la arquitectura del sistema se encuentra en el anexo A (Análisis y Diseño del sistema).

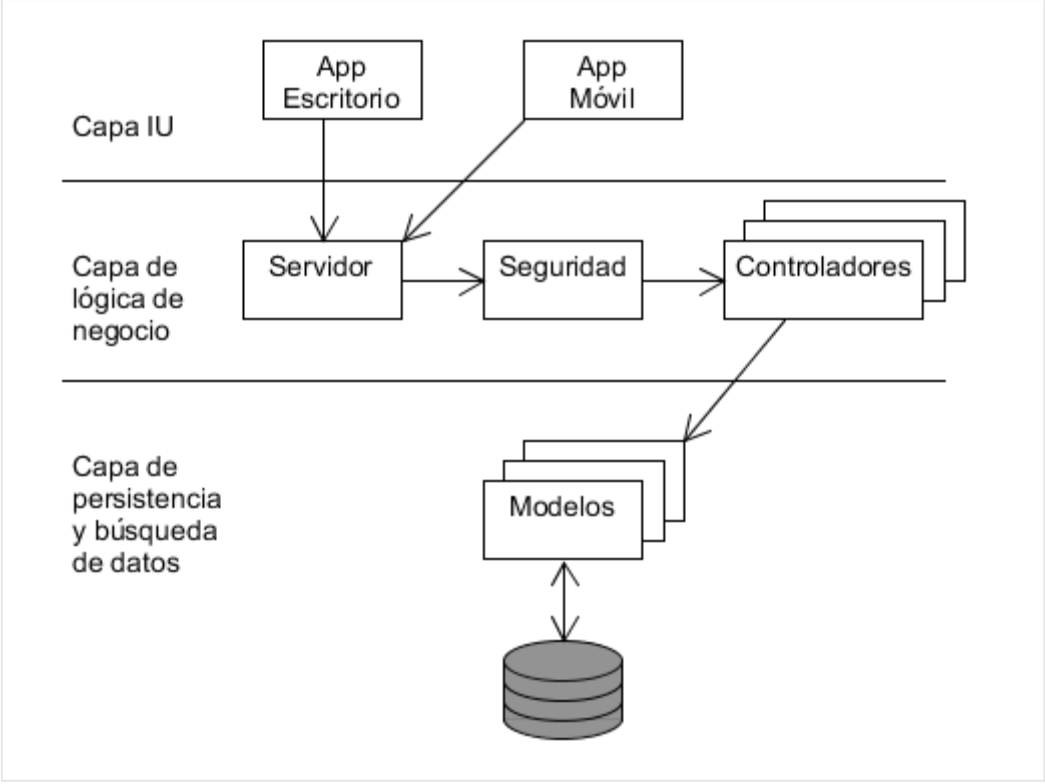

Figura 9. Arquitectura de alto nivel de la solución.

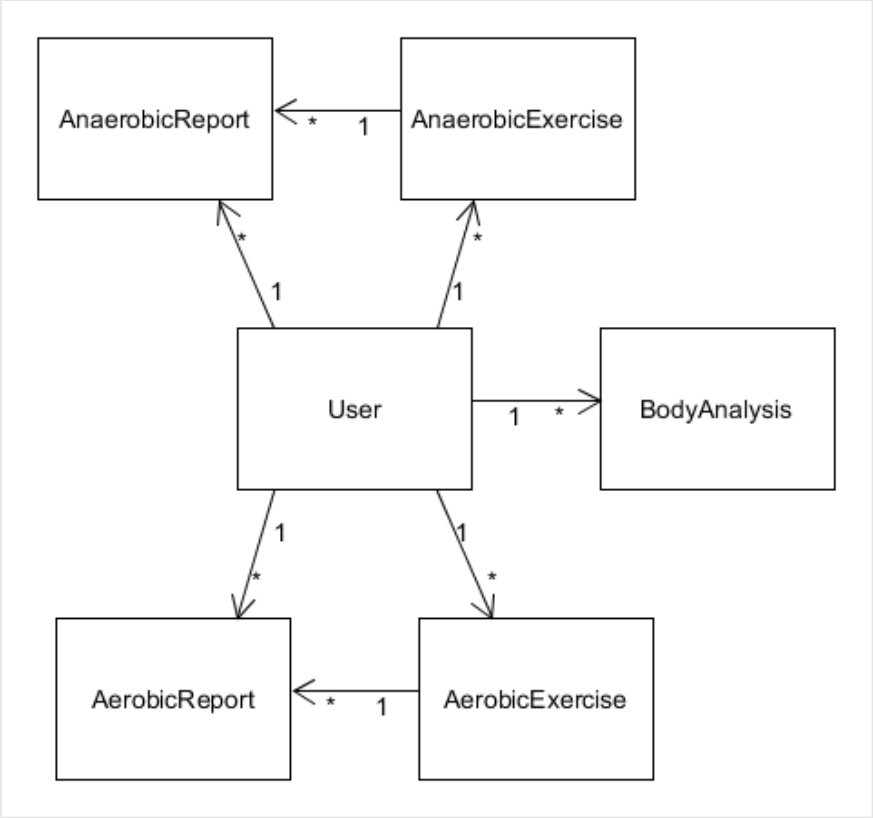

Figura 10. Modelo de datos inicial de alto nivel de la solución.

4.2 Estudio de alternativas y viabilidad

Dado que se trata de un proyecto de desarrollo de software, es muy importante que haya un estudio de alternativas. Para poder tomar correctamente las decisiones, primero se han de resumir los aspectos más relevantes del entorno del proyecto.

El proyecto tenía unas restricciones importantes en cuanto a plazos y esfuerzos, los cuales idealmente no deberían pasar de 300 horas siendo su fecha límite el 23 de julio. Junto a estas restricciones, se tenía sólo un desarrollador, es por eso que cuando se presentaron alternativas de viabilidad, se cogieron aquellas que ya eran dominadas por él o cuyo plazo de aprendizaje eran asumibles. Además, puesto que se trata de un producto completamente nuevo y en evolución, se necesitaba un sistema flexible que pudiese evolucionar a la vez que lo hacía el alcance del proyecto.

A continuación, se van a describir las decisiones tomadas en el proyecto, las cuales se estructuran, como se puede ver en la Figura 11, en las decisiones tomadas para dispositivos clientes móviles y sobremesa, servidor web, comunicaciones en tiempo real, y base de datos. Al final de la sección se encuentra la Tabla 9 con el resumen de alternativas y opción elegida para cada uno de los apartados.

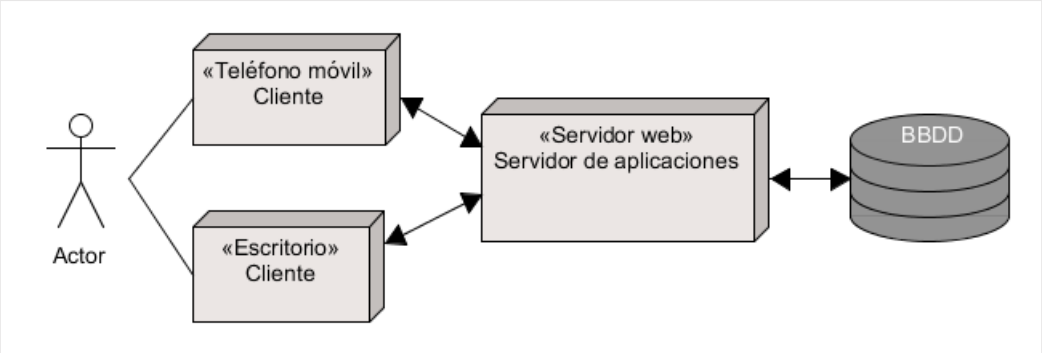

Figura 11. Diagrama de alto nivel de los principales componentes del sistema.

Dispositivo clientes para móviles: Aparecieron dos alternativas viables, la primera de ellas desarrollar la aplicación en lenguaje Android (Android, s.f.) e iOS (Apple Inc., s.f.) nativo, y la segunda en una tecnología híbrida como el *framework* Ionic 2 (Ionic Framework, s.f.). Las decisiones que llevaron a finalmente, desarrollarlo con el segundo, fueron las siguientes:

- La tecnología del *framework* Ionic 2, permitía desarrollar una sola aplicación, que gracias a ser híbrida funcionando sobre el navegador web Cordova, funcionaba independientemente de la plataforma móvil, es decir, con el desarrollo de un mismo cliente móvil, se cubrían las necesidades de Android, iOS y Windows Phone, mientras que, con Android o iOS, solo se cubría uno de los dispositivos móviles.
- Mientras que para desarrollar Android o iOS necesitas desarrolladores especializada únicamente en dichos lenguajes, con el *framework* Ionic 2 únicamente necesitas desarrolladores que sepan implementar aplicaciones Web independientemente de si es para móvil o para dispositivos de escritorio, lo que hace más fácil ampliar el equipo de desarrolladores con vistas al futuro.

Dispositivo clientes de sobremesa: Las alternativas que surgieron fueron entre las diversas aplicaciones con capacidad de implementar tecnología AJAX y así hacer el sistema más escalable, descartando por completo tecnologías de clientes web basadas en *templates* como JSP. Aquí se plantearon cuatro tipos de alternativas distintas: JavaScript plano (DesarrolloWeb, s.f.), jQuery (jQuery, s.f.), React (React, s.f.), y AngularJS[8] (AngularJS, s.f.):

- El primero de ellos se descartó debido a que implementar una aplicación web cliente hoy en día sin utilizar ningún *framework* y utilizando únicamente JavaScript, es bastante contraproducente y se ralentiza el ritmo del trabajo, además de que el sistema pueda acabar teniendo serios problemas de seguridad.
- El segundo quedó descartado puesto que únicamente es una librería del lenguaje JavaScript para manipular el DOM (w3schools, s.f.), haciendo que sea más difícil también implementar algunas funcionalidades que con un *framework* moderno sería más rápido y fácil.
- Por último, las razones de peso para elegir entre React o AngularJS fueron que, además de que el desarrollador del producto dominaba AngularJS con antelación (pudiendo implementar el sistema sin problemas, mientras que no se tenían conocimientos acerca de React), existen unas cláusulas de la empresa Facebook (Jorgé, 2016) que pueden revocar la licencia de React y caer en asuntos judiciales si existe algún tipo de competencia con dicha empresa. Puesto que es una aplicación joven y no se sabe a ciencia cierta hacia dónde podría evolucionar en un futuro, es una razón de peso para tener en cuenta.

[8] **:** En este documento cuando se hace referencia a AngularJS es a las versiones del *framework* 1.X, y cuando se hace a Angular, es a las versiones del *framework* 2.0 en adelante

Servidor Web: Se presentaban dos opciones claras en la capa de la lógica de controladores. Dado que se buscaba un *framework* para desarrollar el back-end de una aplicación web, para evitar los problemas de trabajar sin ellos en un determinado lenguaje, se presentaban dos opciones viables por conocimiento del desarrollador: Spring Framework (Spring Framework, s.f.) y Express Framework (Express Framework, s.f.). El lenguaje del primero es Java, mientras que del segundo es JavaScript.

La decisión aquí fue casi inmediata, se contaba ya con un sistema casi puro escrito en JavaScript y JSON, orientado a recursos tipo *RESTful*, y, además, se tenía más conocimiento de NodeJS y Express que de Spring en cuanto a desarrollo back-end de aplicaciones web. Es por eso que se tomó la decisión de implementarlo con Express y NodeJS, para que, de esta forma, agregando Mongoose como driver de base de datos, se pudiera integrar todo el sistema sin problemas dando lugar a lo que se conoce como el *framework* MEAN (Raj, 2014).

Base de datos: La primera decisión aquí ha sido entre bases de datos SQL y clásicas, NoSQL (Sadalage, 2014). La decisión resultante tomada ha sido escoger una base de datos no relacional debido a las siguientes razones:

- Ya que se trata de un proyecto nuevo, con una metodología ágil, en el cual aún no se tiene clara la dirección y evolución del mismo, esto otorga una mayor flexibilidad, adaptándose a necesidades nuevas que surjan pudiendo hacer cambios de los esquemas sin tener que parar bases de datos.
- En lo referente al rendimiento y escalabilidad, las bases de datos relacionales necesitan un mayor procesamiento en recursos y rendimiento cuanto más compleja sea la base de datos y sus relaciones (principalmente debido a la atomicidad). Esto es un problema ya que se busca una escalabilidad horizontal, ampliando el número de máquinas en lugar de máquinas más potentes. Esto permite que se pueda empezar el proyecto con un menor presupuesto y posteriormente poder integrarlo con más facilidad en servicios como Amazon Web Services y hacer que el sistema posea una alta escalabilidad.
- Por último, en lo relacionado a la atomicidad, las NoSQL salen perdiendo con la consistencia eventual que las caracteriza (por ejemplo, en MongoDB a nivel de documento y no de colección). Sin embargo, solo existiría una operación que incluiría varios documentos a la vez (borrado de ejercicios), que en caso de que fallara, no supondría un daño crítico al sistema y que se podría resolver dejando un proceso en segundo plano que purgara las marcas huérfanas de ejercicios. Es por eso que se asume dicho riesgo ya que no supondría un problema en el sistema.

Por último y debido a restricciones temporales del proyecto, dado que el desarrollador cuenta con conocimientos suficientes como para desarrollar este proyecto a tiempo en MongoDB, pero no en otra tecnología NoSQL, se ha decidido desarrollar la capa de datos en MongoDB.

Tabla 9. Posibles alternativas junto con su opción elegida.

	Opción elegida	Alternativas
Cliente móvil	*Framework* Ionic 2	Android, e iOS
Cliente escritorio	AngularJS	JavaScript, jQuery, y React
Base de datos	MongoDB con Mongoose	Base de datos relacional
Servidor web	NodeJS con Express	Spring Framework

En la Tabla 9 se recogen todas las posibles alternativas que se podían haber escogido para el proyecto, junto con las decisiones elegidas finalmente. Remarcar que dichas decisiones no tienen por qué ser mejores tecnologías que otras, sino que, en el contexto del proyecto de emprendimiento presentado, encajan de una forma mejor que otras.

4.3 Solución tecnológica

Al haberse implementado el sistema con el *framework* MEAN, se ha conseguido un sistema como el de la Figura 12. Todo ello ha otorgado algunas características esenciales en el sistema, las cuales se pueden ver en la Tabla 10.

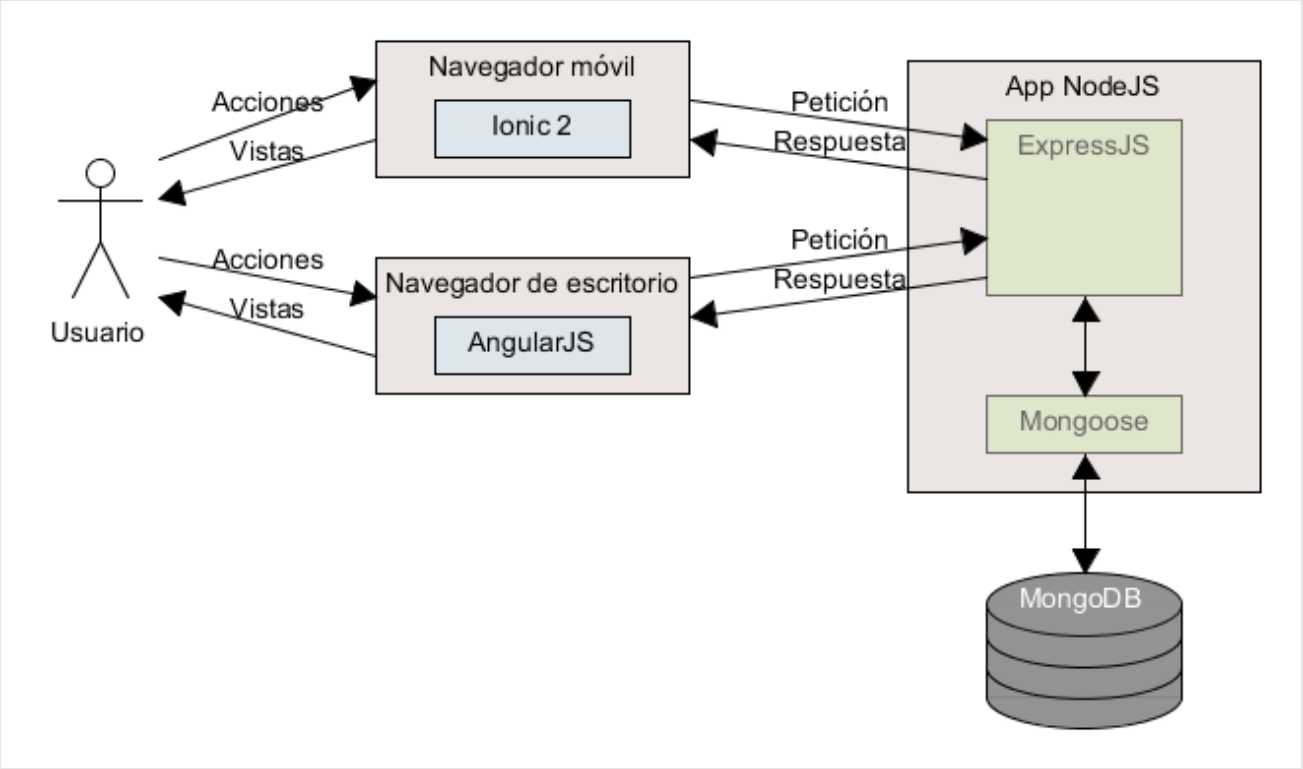

Figura 12. Visión gráfica conceptual de alto nivel de la solución.

Tabla 10. Características esenciales del sistema con el framework MEAN.

Capa de presentación	Desarrollada con las tecnologías AngularJS e Ionic 2, siguiendo un estilo SPA que permite establecer peticiones HTTP con la tecnología AJAX. Esto influye en reducir la carga de los servidores, y ofrecer una mejor experiencia de usuario, pudiendo realizar consultas asíncronas al servidor sin tener que recargar la página completamente.
Capa de servicios web y controladores	El diseño que se ha seguido es utilizando el patrón *API-First*, construyendo la capa a partir del diseño de la API. Puesto que se partía de un sistema desde cero y completamente nuevo, era la mejor forma de empezar a diseñarlo. Esto permite tener divisiones más claras entre la capa de servicios y la capa de *front-end*, consiguiendo menor acoplamiento en el sistema, repitiendo menos código en los controladores y manteniéndolos unificados. Además, dado que se trata de una aplicación claramente orientada a recursos, y haciendo el sistema lo más escalable posible, se ha utilizado el paradigma *RESTful* para diseñar la API y el comportamiento del sistema. De esta forma, se aprovecha al máximo el estándar HTTP, protocolo que, al no tener estado, y además ser fácilmente cacheable, da la ventaja a la hora de aumentar la escalabilidad. Por otro lado, la tecnología usada de NodeJS permite muy fácilmente replicar el sistema horizontalmente, ya que está pensado para generar peticiones no bloqueantes.
Capa de persistencia y búsqueda	Al haberse utilizado la tecnología de MongoDB, su principal ventaja es el uso de la técnica de *sharding* (MongoDB, s.f.). Esta técnica permite a la base de datos particionar los documentos en árboles para poder replicarlos fácil e independientemente. Además, dicha técnica MongoDB la implementa de forma transparente gracias a su tecnología, proporcionando mecanismos internos para realizarla sin que sea necesario implementarla a nivel de aplicación.

Por último, en lo que respecta a seguridad, se ha implementado en el sistema la tecnología de los JSON Web Tokens (Auth0, s.f.). Esta tecnología permite crear unos tokens en el momento de la autenticación, para su posterior envío en cada una de las peticiones HTTP que se realizan en el sistema. Esta tecnología permite gracias a eso cifrar la comunicación dentro del token, y además simular una sesión activa (ya que se les puede otorgar un tiempo de expiración a los mismos). Además, puesto que únicamente se trata del cifrado de un token, esto permite tener una sesión en el sistema, y añadirle un nivel de seguridad, consumiendo un número mínimo de recursos y sin que influya en la escalabilidad del sistema. Asimismo, se ha cifrado la comunicación cliente-servidor dentro de un canal HTTPS.

Lo comentado hasta ahora es lo que el sistema puede hacer actualmente, en el anexo B (Detalles de implementación del sistema), se tiene una visión más a bajo nivel de la solución tecnológica, y en el anexo C (Instrucciones y despliegue del sistema), el procedimiento y los pasos correspondientes para el arranque del sistema. Sin embargo, se ha diseñado pensando también en aspectos futuros, para poder ampliar la escalabilidad sin tener problemas de integración. La propuesta tecnológica que se plantea para la siguiente iteración consistiría en lo siguiente:

- En los temas de seguridad, se puede integrar fácilmente el uso de la tecnología de reCAPTCHA de Google (Google's reCaptcha, s.f.) para evitar un uso de BOTs y SPAM en el sistema en el requisito de registrar cuenta de usuario.
- En los temas de escalabilidad, resiliencia, y recuperación, se puede integrar de una forma rápida los servicios de Amazon S3 (*Amazon Simple Storage Service*) (Amazon Web Services, s.f.). Utilizando dichos servicios se externalizarían estos riesgos (con lo que ello conlleva). Dichos servicios garantizan un 99.99% de disponibilidad en la nube (lo que se traduce en que el servicio está inactivo para los usuarios únicamente durante 48 minutos al año), y un 99,999999999% de durabilidad de los objetos en la nube (lo que se traduce en que, si se guardan 10.000 objetos, de media se pierde 1 de ellos cada 10 millones de años). Además, el servicio *cloud* de Amazon permite el escalado horizontal a demanda del sistema (integrándose con la escalabilidad actual del *framework* MEAN), de forma que el presupuesto es bastante flexible y se amoldaría a las necesidades de la aplicación en cada momento determinado de demanda. Esto permite tener un sistema altamente escalable y disponible a un precio flexible y ajustado.

4.4 Usabilidad de la solución

Se ha realizado un gran esfuerzo para conseguir una aplicación usable en el sistema, ya que principalmente está orientado a un público muy amplio que no siempre tiene porqué tener experiencia con la tecnología, y además se trata de una aplicación que usualmente se usará durante un entrenamiento (con lo que no tiene que requerir de mucho esfuerzo cognitivo ni de mucho tiempo). Para conseguirlo, se han aplicado algunos principios de diseño generales en la capa de presentación y la UI los cuales se pueden ver en la Tabla 11, dando como resultado las vistas de la aplicación.

Tabla 11. Principios de diseño generales de la UI.

Consistencia	Se ha intentado que toda la aplicación funcione con los mismos estilos, iconos y formas de manipulación de interfaz, para que el usuario una vez sepa hacer cualquier acción dentro de ella, el resto le sean fácilmente recordables y la curva de aprendizaje sea más rápida. Además, se mantiene la consistencia gracias a la tecnología de Ionic 2, en el uso de iconos y distribución de la UI, con los estilos generales dependiendo del dispositivo móvil que lo esté ejecutando (iOS, Android, o Windows Phone), de forma que para ellos algunos componentes les resulten familiares (p.ej.: El botón del *navbar*).
Retroalimentación	Todas las acciones que realiza el usuario tienen su propia retroalimentación con mensajes llamativos y textos explicativos que permiten al usuario saber lo que está pasando gracias a sus acciones en todo momento, así como el estado de la aplicación.

Número mágico	Gracias al estudio de George A. Miller (Miller, 1956) se sabe que la memoria de trabajo de las personas puede almacenar una cantidad máxima de 7 (+/- 2) elementos o de unidades de información que se pueda recordar, se ha seguido este principio también para no sobrecargar al usuario en ninguna pantalla con más acciones de las que pueda recordar.

4.5 Documentación de la API

Para que la *API RESTful* pública del sistema sea accesible por cualquier entidad externa, y esté correctamente documentada, se ha utilizado la herramienta de Swagger (Swagger, s.f.), la cual sincroniza automáticamente la documentación del código para transformarla en formato JSON y posteriormente, con el servidor en ejecución, la presenta por pantalla en las direcciones "localhost:8080/swagger.json" sin formatear y "localhost:8080/api-docs/" ya formateada. Esto hace que la API tenga un formato de presentación intuitivo y fácilmente accesible por entidades de terceros, y además se mantenga sincronizada la documentación de la misma con el código, dando un resultado como en la Figura 13 y la Figura 14 . Toda la API en detalle del sistema, con un total de 28 operaciones, se encuentra en el anexo D (*API RESTful* del sistema).

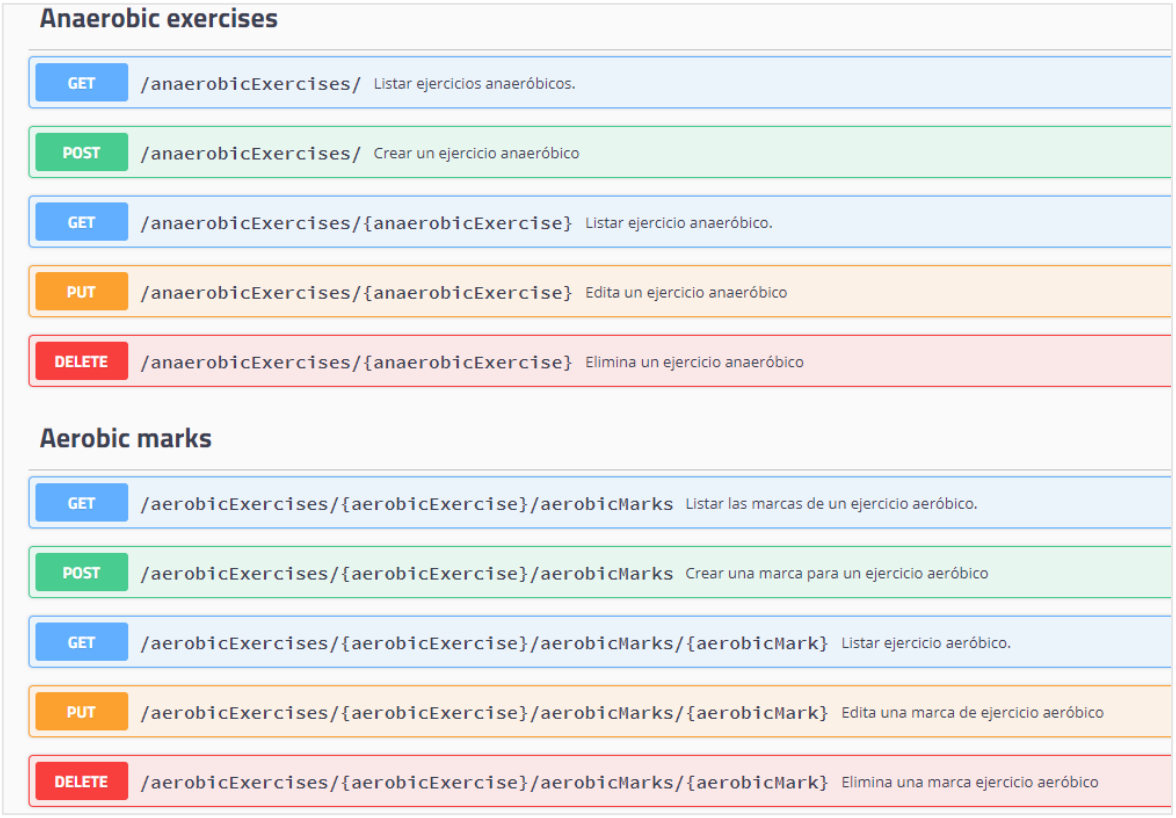

Figura 13. Ejemplo de Swagger: Lista de operaciones.

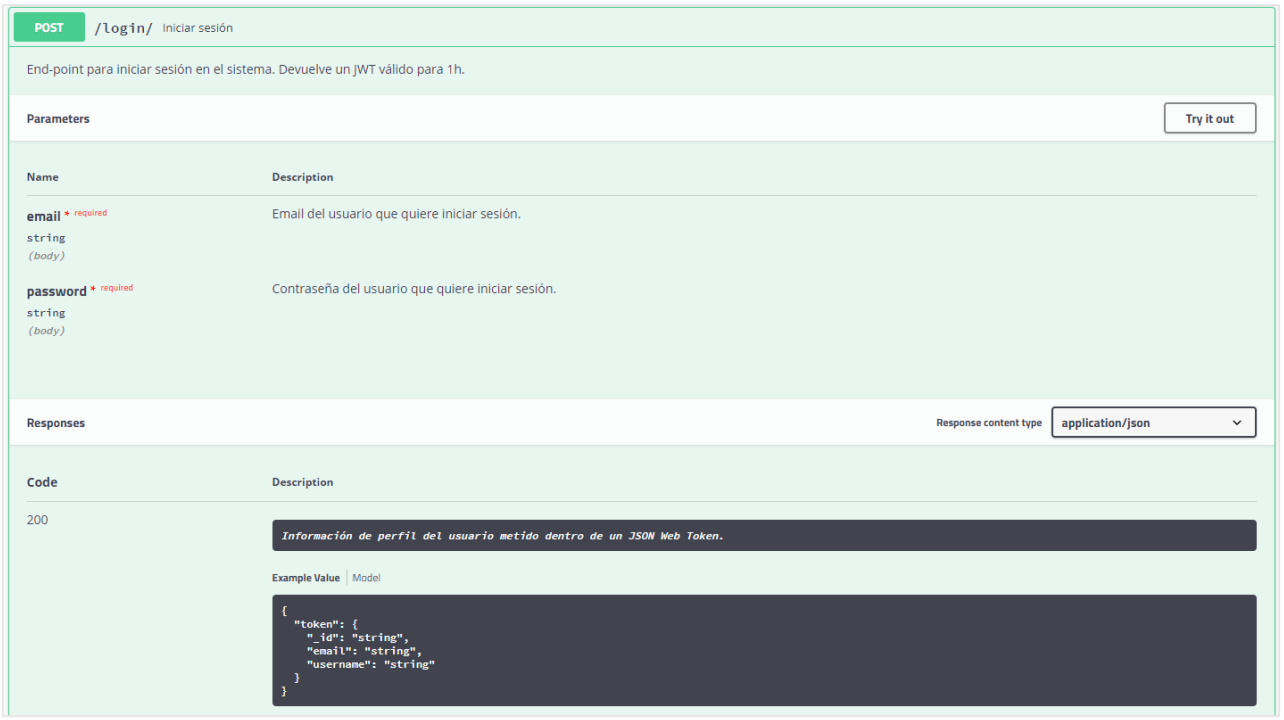

Figura 14. Ejemplo de Swagger: Operación ampliada.

5. Proceso de desarrollo

5.1 MVP 1

5.1.1 MVP 1: Definición

En este primer *MVP* se ha decidido invertir el esfuerzo en validar si las funcionalidades núcleo de la aplicación estaban siguiendo un buen camino. Éstas principalmente son las relacionadas con añadir ejercicios personalizados, marcas de un ejercicio, y análisis físicos.

Se han decidido priorizar estas tareas con respecto a otras como la visualización de históricos y gráficas (pese a que éstas son más llamativas y se encuentran en ese sector del modelo Kano de características exclusivas), porque incluyen algunas decisiones importantes de diseño. Estas decisiones se basan sobre todo en que, al querer integrar muchos deportes a la vez, se tenía que validar con los usuarios si se estaba representando correctamente a la variedad de dichos deportes, tanto a la hora de crearlos como a la hora de añadirles marcas.

Además, como la adición de análisis físicos tampoco se ha encontrado en otros competidores/aplicaciones, se tenía que validar si realmente era algo que interesaba, y en ese caso, si los atributos añadidos eran los correctos, si eran demasiados, o innecesarios, o si había alguna carencia de algún otro.

Para que el uso de la aplicación sea medianamente fluido durante el experimento para los usuarios (ya que también se quiere realizar un primer examen a la usabilidad de la aplicación), se han tenido que desarrollar otras características del sistema de las que dependen las anteriores, como puedan ser la de listar los ejercicios predeterminados y personalizados, y la de iniciar sesión con una cuenta de usuario. Aquí en este caso, se ha decidido dejar fuera la característica de registrar cuenta de usuario porque no era realmente necesaria. Para solventar esto, el sistema ya contaba con unos usuarios predeterminados en el sistema, que se les suministraba a los usuarios durante el experimento para que no tuvieran que preocuparse de ello.

Por último, la característica de buscar un ejercicio rápidamente se ha desarrollado también por las razones de que además de ser una característica exclusiva e interesante para los usuarios, tenía una complejidad reducida y podía incluirse en el primer *MVP* sin retrasar su planificación.

A continuación, se puede ver en la Figura 15 el mapa de características con las funcionalidades resaltadas que han entrado en el primer *MVP*. También se pueden ver en la Figura 16 y en la Figura 17 los bocetos relevantes de la interfaz de usuario referentes al *MVP* 1.

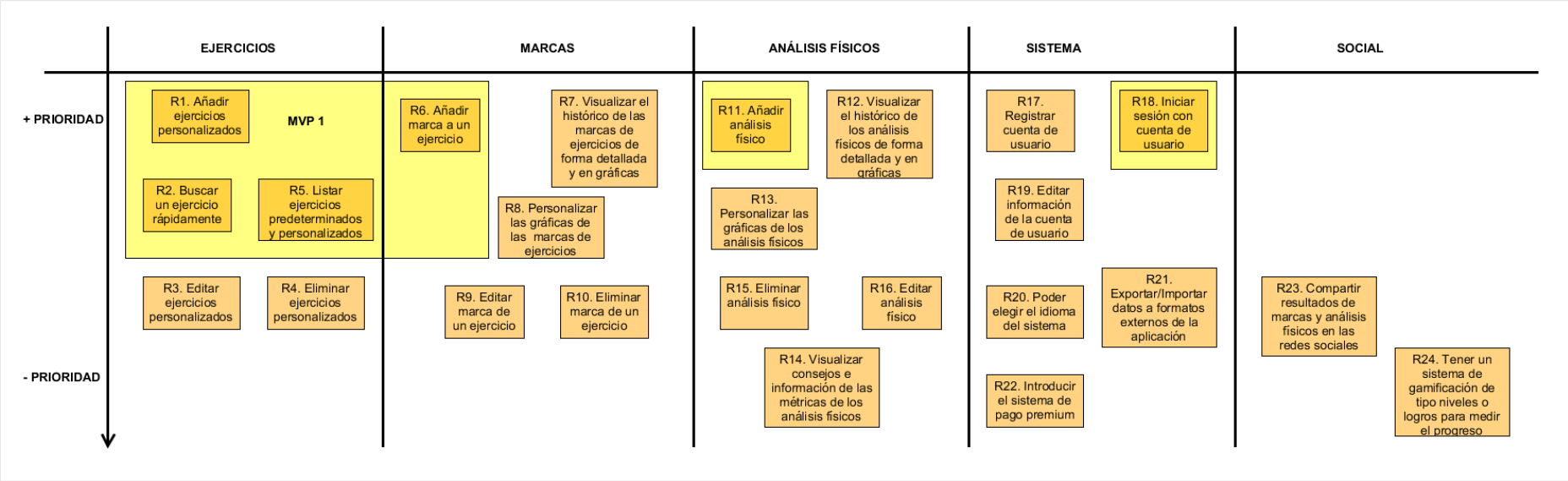

Figura 15. Mapa de producto del MVP 1.

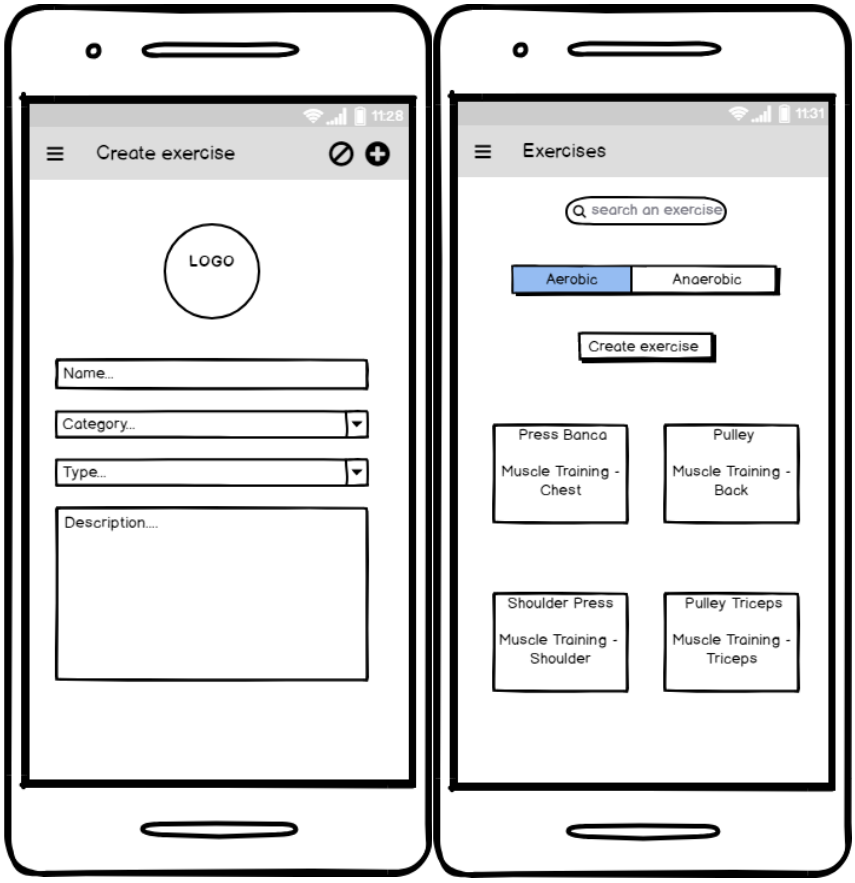

Figura 16. Bocetos de la interfaz de usuario del dispositivo móvil del MVP 1.

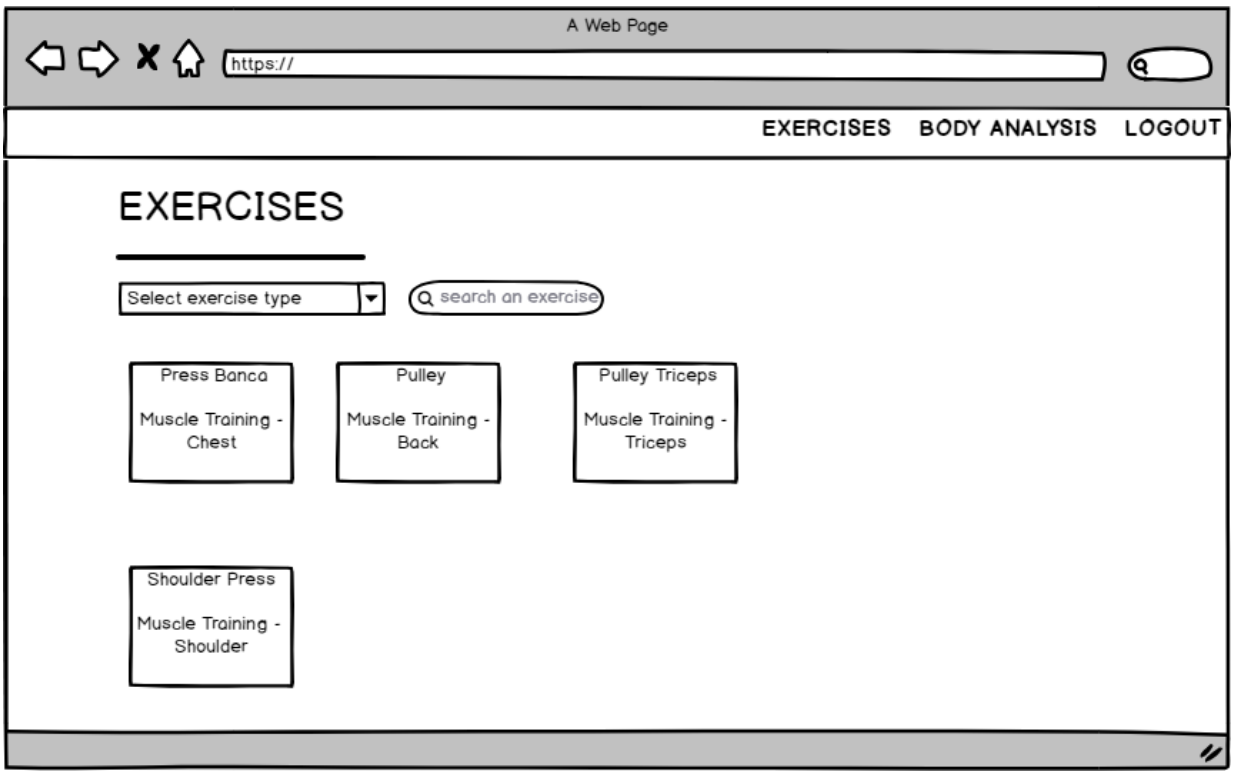

Figura 17. Bocetos de la interfaz de usuario del dispositivo de escritorio del MVP 1.

5.1.2 Preparación del experimento

Al acabar el MVP 1, se ha realizado el primer experimento asociado al mismo. Este experimento se ha realizado con el fin de obtener retroalimentación de los usuarios y saber si se está siguiendo un buen camino o no, o si hay algún tipo de característica o comodidad que no se haya pensado en el planteamiento inicial. Además, también han servido para realizar test de usabilidad supervisados.

El proceso del experimento ha consistido en una sala que contaba con un dispositivo móvil y un ordenador con las dos aplicaciones lanzadas. Antes de empezar el experimento los involucrados tenían preparado un pequeño formulario para introducir algunos datos de carácter personal que son de interés para el producto. Una vez rellenado, se les hacia una breve introducción sobre la motivación de la idea, y funcionalidades que se iban a poder realizar. Después se les explicaba qué funcionalidades de ellas estaban ya implementadas y cuales no lo estaban, pero sí que lo iban a estar antes del lanzamiento para que no se preocuparan de ellas. Una vez aclarados esos detalles, se les dejaba a los usuarios vía libre para que realizaran con la aplicación lo que quisieran durante un máximo de 20 minutos. Por último, se les pasaba un pequeño cuestionario para la evaluación de la usabilidad y se les preguntaba acerca de su experiencia con la aplicación, así como posibles sugerencias.

En el formulario inicial que los participantes tenían que contestar acerca de sus datos personales, se ha utilizado una escala de frecuencia de 1-7 (Vagias, 2006) con el siguiente significado: **1**- Nunca, **2**- Raramente, aproximadamente menos del 10%, **3**- Ocasionalmente, aproximadamente un 30%, **4**- A veces, aproximadamente un 50%, **5**- Frecuentemente, aproximadamente un 70%, **6**- Por lo general, aproximadamente un 90%, **7**- Siempre. Este formulario consistía en las siguientes 8 preguntas:

1. **Género (H/M)**
2. **Edad**
3. **Uso Frecuente de ordenadores (1-7)**
4. **Uso frecuente de móviles (1-7)**
5. **Uso frecuente de *tablets* (1-7)**
6. **¿Utiliza alguna aplicación deportiva? (S/N)**
7. **Frecuencia con la que utiliza dichas apps (1-7)**
8. **Deporte/s que practica**

Para el cuestionario final que los participantes tenían que contestar para el análisis de la usabilidad supervisada del sistema, se ha escogido la técnica de las heurísticas de Nielsen (Latorre, Baldassarri, & Cerezo, 2015). Esta técnica sirve para analizar globalmente la interfaz, es un método de evaluación que se basan en principios reconocidos de usabilidad. El método consiste en 10 reglas que son evaluadas por participantes para validar la interfaz. Cada evaluador puntúa del 0-7 (siendo 0 NS/NC, 1- Totalmente en desacuerdo, 2- No estoy de acuerdo, 3- Algo en desacuerdo, 4- Ni de acuerdo ni en desacuerdo, 5- Algo de acuerdo, 6- De acuerdo, 7- Totalmente de acuerdo) cada una de las heurísticas. Las 10 reglas que se han utilizado para evaluar han sido las siguientes:

1. **Visibilidad del estado del sistema**: El sitio mantiene informados a los usuarios de lo que está ocurriendo a través de retroalimentación apropiada.
2. **Relación entre el sistema y el mundo real**: El sitio habla el lenguaje de los usuarios mediante conceptos que son familiares al usuario.
3. **Control y libertad del usuario:** El sitio permite “salidas de emergencia” claramente marcadas, como “deshacer” y “rehacer”.
4. **Consistencia y estándares:** el sitio sigue las convenciones establecidas. Los usuarios nunca tienen duda de que diferentes elementos significan en realidad la misma cosa.

5. **Prevención de errores:** El sitio tiene un diseño cuidadoso que previene la ocurrencia de problemas.
6. **Reconocimiento antes que recuerdo:** El sitio hace visibles los objetos, acciones y opciones. Las instrucciones para su uso están a la vista o son fácilmente recuperables cuando sea necesario.
7. **Flexibilidad y eficiencia de uso:** El sitio incluye la presencia de aceleradores que no son vistos por los usuarios novatos, pero ofrecen una interacción más rápida a los usuarios expertos.
8. **Estética y diseño minimalista:** El sitio no contiene información que es irrelevante o poco usada.
9. **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores:** el sitio incluye mensajes de error en un lenguaje claro y simple.
10. **Ayuda y documentación:** El sitio ofrece ayuda y documentación. Dicha información es fácil de buscar.

5.1.3 MVP 1: Resultados del experimento

En el primer experimento participaron solo 4 personas con el fin de que fuese más controlado y dirigido. Se buscaron perfiles que practicaban más de un deporte y que solían utilizar aplicaciones parecidas para que puedan valorar y validar mejor la idea. Los resultados se pueden en la Tabla 12.

Tabla 12. Descripción de los participantes del MVP 1.

Participante	Género (H,M)	Edad	Uso frecuente de ordenadores (1-7)	Uso frecuente de móviles (1-7)	Uso frecuente de *tablets* (1-7)	¿Utiliza alguna aplicación deportiva? (S,N)	Frecuencia con la que utiliza dichas apps (1-7)	Deporte/s que practica
1	M	24	7	7	3	S	6	Bici, Gimnasio
2	H	24	7	7	1	S	5	Correr, Gimnasio
3	H	26	7	7	1	S	5	Bicicleta, Natación, Correr, Gimnasio
4	M	21	7	7	1	S	5	Bicicleta, Gimnasio

Los resultados de los formularios de las heurísticas de Nielsen son los que se muestran en la Tabla 13, que se han representado gráficamente (ver Figura 18). Los resultados obtenidos para estas heurísticas son muy satisfactorios, con un 90% de aceptación.

Tabla 13. Heurísticas de Nielsen aplicadas al MVP 1.

Heurísticas	NS/NC	Totalmente en desacuerdo					Totalmente de acuerdo	
	0	1	2	3	4	5	6	7
Visibilidad del estado del sistema							1	3
Relación entre el sistema y el mundo real							1	3
Control y libertad del usuario						3	1	
Consistencia y estándares								4
Prevención de errores							2	2
Reconocimiento antes que recuerdo							2	2
Flexibilidad y eficiencia de uso					4			
Estética y diseño minimalista							1	3
Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de errores							1	3
Ayuda y documentación							1	3

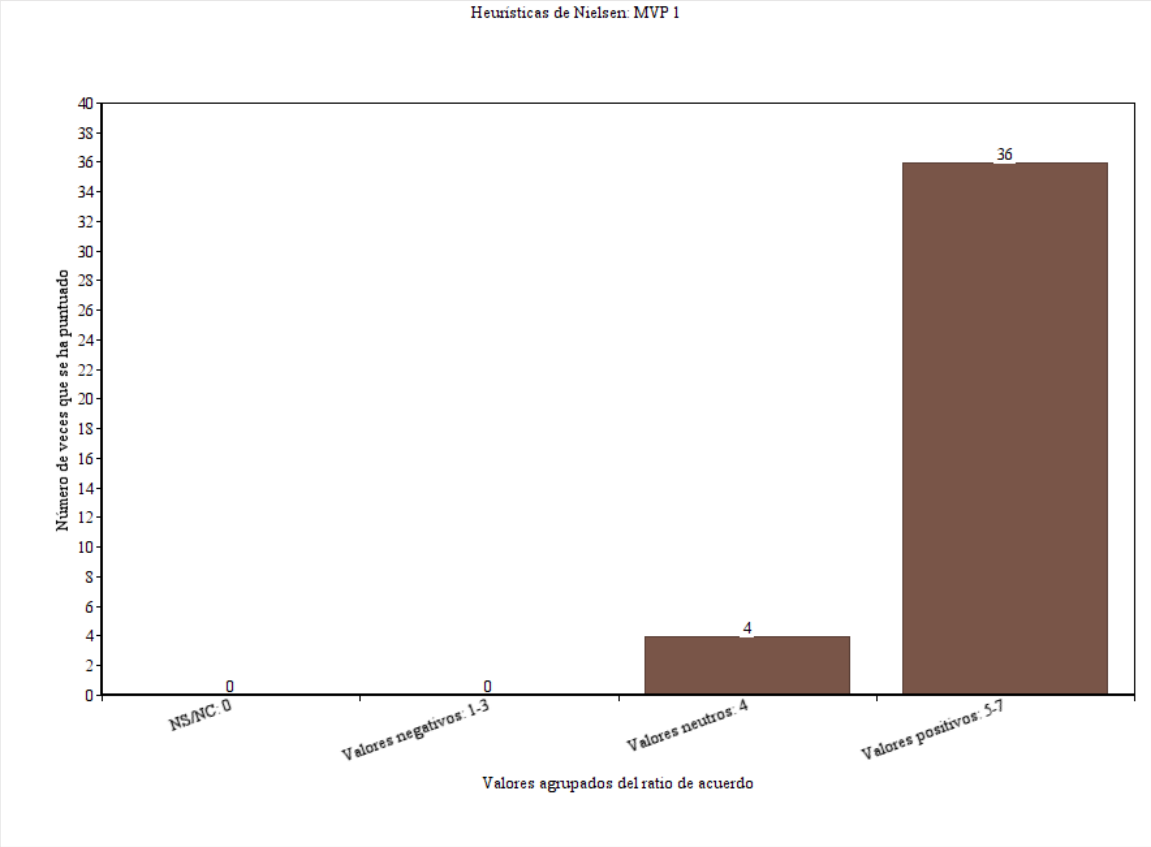

Figura 18. Resultados de las heurísticas de Nielsen del MVP 1 agrupados por puntuación.

Respecto a los comentarios generales de los participantes, centrados en lo que les parece la aplicación, todos coincidían en que les ha parecido una idea interesante y atractiva, que facilita la integración de funcionalidades cuando practicas más de un deporte. Ya en los comentarios más centrados en cómo mejorarían la aplicación o qué les gustaría que se cambiara o añadiera, se destacan dos comentarios interesantes:

- Que se puedan añadir las series de los ejercicios anaeróbicos de una en una, en vez de un único valor. En el momento del experimento, solo se podía introducir un valor numérico para los atributos "tiempo, peso y repeticiones", lo que hacía que no se ajustara a la realidad. Esto hacía que algún participante preguntara si se referían a las marcas medias de las series realizadas, a la más alta, a la última… etc. Para ello, proponían poder introducir dichos atributos una vez por cada serie que has realizado en el ejercicio.
- Que las tarjetas de los ejercicios que aparecen en la pantalla donde se listan, incluyan de alguna forma una pequeña foto representativa de lo que hace el ejercicio.

Por último y como autocrítica para mejorar en los siguientes experimentos se han observado algunos detalles que de cara al siguiente deberían ser cambiados:

- La escala de los formularios tanto de frecuencia como de ratio de acuerdo se tiene que reducir. Ya que los usuarios no son expertos en análisis de usabilidad, causaba un poco de confusión tener un rango tan extenso entre el que elegir. Además, como se puede apreciar en los resultados realmente todos acaban concentrándose en los mismos valores, sin cobrar mucha importancia los valores intermedios.
- Pese a que se ha conseguido el objetivo de conseguir participantes que realizaran más de un deporte y que utilizaran aplicaciones parecidas, de cara al siguiente experimento, además de conseguir más participantes, se ha de hacer un especial esfuerzo en conseguir algunas personas en otros rangos de edades como puedan ser los rangos de [30,40] y [40,50]. Además, hay que intentar también tener una mayor variedad en las frecuencias de uso de ordenadores y móviles.

5.2 MVP 2

5.2.1 MVP 2: Definición

En este segundo *MVP* se ha decidido invertir el esfuerzo en validar las funcionalidades restantes del núcleo de la aplicación y que son de un mayor atractivo, manteniéndose en las clasificadas como exclusivas con respecto a los competidores. Estas principalmente son las relacionadas con las gráficas y los históricos.

En esta ocasión se han realizado menos características durante el desarrollo de esta segunda iteración dado que éstas tienen una mayor complejidad y carga de trabajo que las anteriores. Además, son unas características que se quieren probar lo antes posible porque junto con las de la iteración anterior, forman casi todas las características exclusivas del sistema, pudiendo comprobar mejor si realmente el producto es viable.

Estas características consisten en las de visualizar el histórico de marcas y análisis físicos de forma detallada y en gráficas, además de la posibilidad de la personalización de la información mostrada en las propias gráficas.

Por último, se ha incluido una característica obtenida de la retroalimentación del experimento pasado, que es la de añadir una por una las series de una marca de los ejercicios anaeróbicos. Esta característica se ha decidido implementar en este primer experimento porque fue un comentario de más de un participante del experimento anterior y además se ha considerado que realmente era una carencia y error importante que no se tuvo en cuenta en un inicio y que afecta a la experiencia de usuario en una de las características más base del sistema.

A continuación, se puede ver en la Figura 19 el mapa de características con las funcionalidades resaltadas, que han entrado en el segundo *MVP* junto con las del primer *MVP,* para que hacerse un mapa mental del desarrollo global que se lleva hasta ahora del producto sea más fácil. También se pueden ver en la Figura 20 y en la Figura 21 los bocetos relevantes de la interfaz de usuario referentes al *MVP* 2.

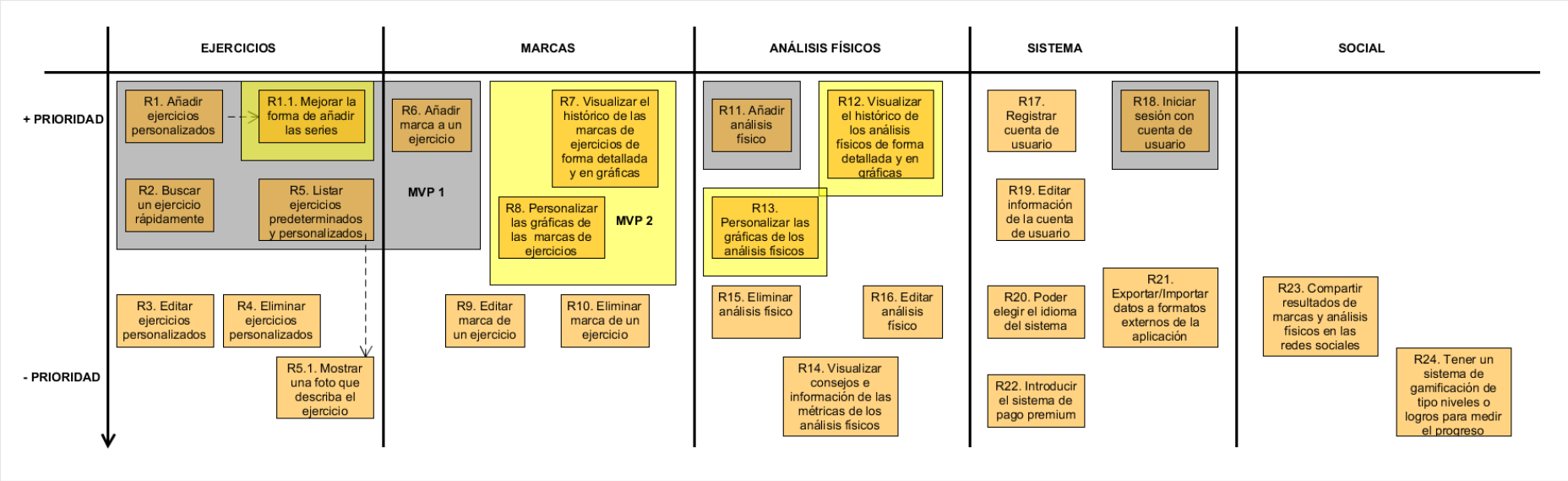

Figura 19. Mapa de producto del MVP 2.

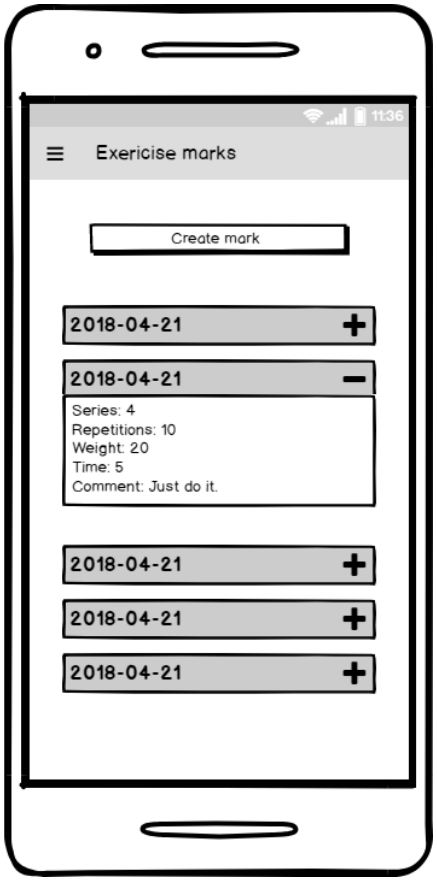

Figura 20. Bocetos de la interfaz de usuario del dispositivo móvil del MVP 2.

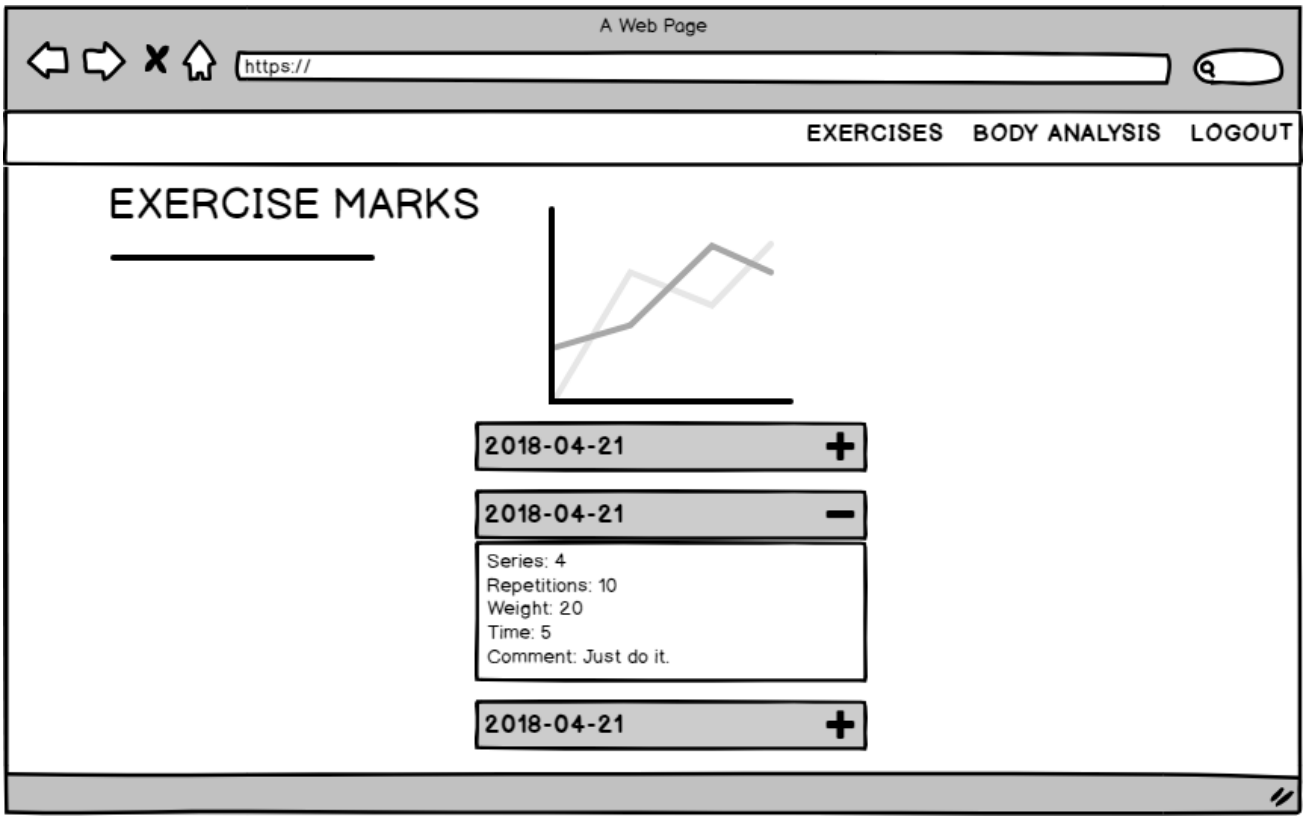

Figura 21. Bocetos de la interfaz de usuario del dispositivo de escritorio del MVP 2.

5.2.2 MVP 2: Preparación del experimento

Al acabar el MVP 2, se ha realizado el primer experimento asociado al mismo. Este experimento se ha realizado con el fin de obtener, como en el experimento anterior, retroalimentación de los usuarios y saber si se está siguiendo un buen camino o no, o si hay algún tipo de característica o comodidad que no se haya pensado en el planteamiento inicial. Además, se han realizado los test de usabilidad supervisados igual que en el MVP 1.

El proceso del experimento ha sido prácticamente como en el del experimento del MVP 1. La única diferencia se encuentra en los formularios, tanto del inicial, con los datos personales de los participantes, como del final, con las heurísticas de Nielsen. Esta diferencia consiste en haber reducido la escala de frecuencia y la de ratio de acuerdo en dichos formularios, que era de 1-7.

La nueva escala que se ha decidido escoger ha sido la escala Likert (SurveyMonkey, s.f.). Esta escala reduce en 2 las posibilidades para valorar del usuario acotando mejor su respuesta. Esta escala 1-5 tiene el siguiente significado para los formularios de frecuencia: 1- Nunca, 2- Casi nunca, 3- Ocasionalmente/A veces, 4- Casi todas las veces 5- Siempre. Para los formularios de ratio de acuerdo tiene el siguiente significado: 1- Totalmente en desacuerdo, 2- En desacuerdo, 3- Ni de acuerdo ni en desacuerdo, 4- De acuerdo, 5- Totalmente de acuerdo.

5.2.3 MVP 2: Resultados del experimento

Para el segundo experimento se ha aumentado el número de participantes para intentar que éste fuera de mayor envergadura y con una mayor retroalimentación de cara ya a un primer lanzamiento. En este caso se ha logrado la participación de hasta 16 usuarios, incluyendo los mismos 4 del experimento primero. Se han seguido focalizando en perfiles que practiquen más de un deporte (aunque también se han buscado unos pocos que solo practicaran 1 para ver la reacción de ese sector de usuarios), pero ahora intentando que hubiera más diversidad en el rango de edad, en los deportes practicados y en la frecuencia de uso de las tecnologías. Los resultados del primer formulario se pueden en la Tabla 14.

Tabla 14. Descripción de los participantes del MVP 2.

Participante	Género (H,M)	Edad	Uso frecuente de ordenadores (1-5)	Uso frecuente de móviles (1-5)	Uso frecuente de *tablets* (1-5)	¿Utiliza alguna aplicación deportiva? (S,N)	Frecuencia con la que utiliza dichas apps (1-5)	Deporte/s que practica
1	M	24	5	5	3	S	5	Bici, Gimnasio
2	H	24	5	5	1	S	4	Correr, Gimnasio
3	H	26	5	5	1	S	4	Bicicleta, Natación, Correr, Gimnasio
4	M	21	5	5	1	S	4	Bicicleta, Gimnasio
5	H	23	5	5	1	N	1	Correr, Gimnasio
6	H	36	2	5	1	N	1	Correr, Gimnasio
7	H	26	5	5	3	N	1	Bicicleta
8	H	24	4	5	1	Y	3	Bicicleta, Gimnasio, Escalada
9	M	24	4	5	1	Y	5	Bicicleta, Gimnasio
10	H	26	5	5	1	N	1	Gimnasio
11	M	24	5	5	1	N	1	Tenis, Gimnasio
12	H	33	3	5	1	N	1	Correr, Gimnasio
13	M	24	4	5	1	Y	3	Deportes extremos, Deportes de invierno, Bicicleta, Gimnasio
14	H	23	5	5	1	N	1	Bicicleta, Gimnasio
15	M	24	5	5	1	N	1	Gimnasio

Participante	Género (H,M)	Edad	Uso frecuente de ordenadores (1-5)	Uso frecuente de móviles (1-5)	Uso frecuente de *tablets* (1-5)	¿Utiliza alguna aplicación deportiva? (S,N)	Frecuencia con la que utiliza dichas apps (1-5)	Deporte/s que practica
16	H	25	5	5	1	N	1	Correr, Gimnasio

Los resultados de los formularios de las heurísticas de Nielsen son los que se muestran en la Tabla 15, que se han representado gráficamente (ver Figura 22). Los resultados obtenidos para estas heurísticas siguen siendo muy satisfactorios, con un 81% de aceptación. Los valores neutros se mantienen proporcionales, mientras que la mayor diferencia es la aparición de un pequeño porcentaje (6%) que los participantes han marcado como "NS/NC".

Tabla 15. Heurísticas de Nielsen aplicadas al MVP 2.

Heurísticas	NS/NC	Totalmente en desacuerdo				Totalmente de acuerdo
	0	**1**	**2**	**3**	**4**	**5**
Visibilidad del estado del sistema					5	11
Relación entre el sistema y el mundo real					2	14
Control y libertad del usuario	2			8	6	
Consistencia y estándares					1	15
Prevención de errores	1				5	10
Reconocimiento antes que recuerdo					5	11
Flexibilidad y eficiencia de uso	6		1	8	1	
Estética y diseño minimalista					1	15
Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de errores					2	14
Ayuda y documentación				4	5	7

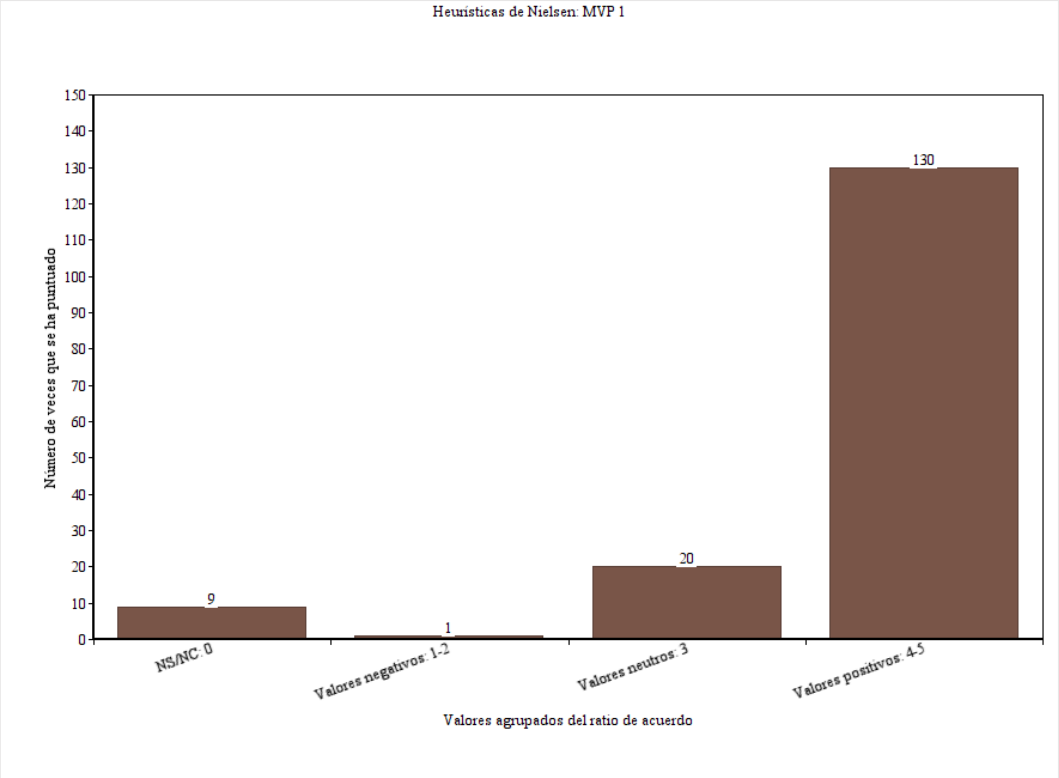

Figura 22. Resultados de las heurísticas de Nielsen del MVP 2 agrupados por puntuación.

Respecto a los comentarios generales de los participantes centrados en lo que les parece la aplicación, todos volvían a coincidir en que les parecía una idea interesante y atractiva, que facilita la integración de funcionalidades cuando practicas más de un deporte. Todos más o menos coincidían también en la facilidad de uso de la aplicación (pudiéndose ver también reflejado en los cuestionarios de las heurísticas de Nielsen), y que todo pareciera bastante intuitivo.

Con lo que respecta a la creación de nuevos ejercicios, marcas de los mismos, y gráficas, todos parecían sentirse cómodos con lo que esas funcionalidades ofrecían, pareciéndoles algo bastante interesante a la hora de guardar y poder visualizar su progreso de una forma clara.

Las personas que la actividad que más practicaban era la de ejercicios anaeróbicos del gimnasio (como, por ejemplo, levantamiento de pesas), parecían especialmente más contentos con la aplicación. Por otro lado, las personas que se centraban más en un solo tipo de deporte aeróbico (como, por ejemplo, el participante 7 que sólo hacía bicicleta), aunque no le disgustaba, prefería que su aplicación estuviera más enfocado a seguimientos en vivo en vez de al almacenaje y explotación de marcas e históricos. Sin embargo, esto no es un problema, ya que el público realmente objetivo, que son las personas que practican más de un deporte, son los que más contentos se mostraban con ella.

Por último, algunos de los comentarios más interesantes y repetidos entre los participantes referentes a mejoras o sugerencias de funcionalidades que quisieran que tuviera la aplicación, se destacan los siguientes:

- Poder visualizar de forma inmediata la última marca de cada ejercicio al entrar en su histórico, ya que es algo que se utiliza con frecuencia.
- Poder realizar las mismas acciones en ambos clientes sin que haya distinciones.
- Poder borrar más de una marca antigua de golpe. Esto viene a raíz de que, con el uso prolongado de la aplicación, el número de marcas de un ejercicio frecuente puede ser muy largo, y te puede interesar borrar por ejemplo las marcas que hiciste durante tu primer año de entrenamiento, que ya no son relevantes para uno mismo.

- Implementar un *login* social con Google, Facebook y/o Twitter.
- Más opciones de personalizar las búsquedas, como por ejemplo poder filtrar por tipo o categoría.
- Que el listado de los ejercicios en vez de ordenarse alfabéticamente, que se ordene por frecuencia de uso para que los más comunes siempre estén más accesibles.
- Opciones para personalizar la visualización de la gráfica no sólo con combinaciones de lo que se quiere mostrar sino también del tiempo. Es decir, que se pueda personalizar que la gráfica solo muestre la progresión entre un intervalo de tiempo.
- Demasiados ejercicios predeterminados. En este aspecto algunos usuarios lo preferían y otros no. Es por eso que alguna solución puede pasar por dar la opción de borrarlos en el futuro, o de que se le dé al usuario a elegir si quiere cargar los ejercicios predeterminados o no.

Por último, y como autocrítica referente al experimento en sí mismo, resaltar que los usuarios que han repetido el experimento han notado una mayor facilidad a la hora de rellenar los formularios con las escalas reducidas. Además, Se ha conseguido una mayor variedad en los deportes practicados, así como en el uso de aplicaciones deportivas y de frecuencia del uso de ordenadores. Sin embargo, ha sido imposible encontrar personas que no estén familiarizadas con el uso de teléfonos móviles. En lo que respecta al rango de edad, pese a que también es algo más variado, no se ha conseguido ninguna persona en el rango de edad de los [40,50] años (siendo estos unos usuarios que no hay que descartar como posibles clientes), superando únicamente la barrera de los 30 con sólo dos participantes.

5.3 Sprint 1

5.3.1 Pila de producto

Como ya se ha comentado anteriormente en la planificación del proyecto, mientras que las dos iteraciones anteriores se centraban en validar el núcleo y la viabilidad de la idea con productos mínimos viables (MVP 1 y MVP 2) y experimentos con usuarios, esta tercera iteración se considera el sprint de lanzamiento del producto, de cara a una normalización de sus iteraciones con vistas al futuro. Esa es la razón por la que este *sprint* se centra más en refinar el producto:

- Dotar al sistema de una calidad en cuanto a verificación y validación se refiere, con tests automáticos del software. Implementar tests de regresión automáticos con las tecnologías y metodologías detalladas en el punto de Calidad del producto, tanto de las funcionalidades de iteraciones anteriores como de la actual.
- Cerrar algunos grupos de funcionalidades para que el usuario pueda ser completamente autónomo en la navegación y utilización del sistema (p.ej.: editar y eliminar las distintas entidades que el usuario puede crear, registrar una cuenta de usuario, así como poder editarla, etc.).

Para ello, se ha volcado todo el mapa de características restante en una pila de producto tipo *SCRUM* y se le han añadido las necesarias para lo anterior citado y las funcionalidades nuevas sacadas de la retroalimentación del experimento anterior.

En la Tabla 16 se recoge el estado de las entradas de la pila con las historias de usuario ordenadas por prioridad. Las funcionalidades marcadas con el carácter '✓' son las que se incluyen en el alcance del sprint. Cada una de ellas contiene una referencia a una tabla que contiene la historia de usuario más detallada. Estas tablas cuentan con su riesgo, estimación (en puntos de historia), prioridad, el cliente al que pertenecen, los y criterios de aceptación.

Las marcadas con el carácter '✗' son las que se dejan para futuros *sprints* o lanzamientos. Cada una de ellas contiene su estimación en lugar de la referencia a la tabla, pues aún no ha sido refinada. Al tratarse de una entrada que se va a incluir en iteraciones posteriores, solo se estima en puntos de historia si se cree que es fácil de estimar y se va a incluir en la siguiente iteración, de lo contrario se estima en tallas de camiseta ya sea porque necesite de un mejor refinamiento o porque se va a incluir en iteraciones a largo plazo.

Tabla 16. Pila de producto del sprint 1.

✓	**R36. Como desarrollador, tengo que implementar tests automáticos y realizar análisis de QA (Tabla 18)**
✓	**R34. Como desarrollador, tengo que unificar los estilos de la interfaz de usuario de los dos clientes (Tabla 18)**
✓	**R17. Como usuario, quiero registrarme con una cuenta en la aplicación (Tabla 19)**
✓	**R19. Como usuario, quiero editar mi cuenta (Tabla 20)**
✓	**R4. Como usuario, quiero eliminar mis ejercicios personalizados (Tabla 21)**
✓	**R10. Como usuario, quiero eliminar una marca en un ejercicio (Tabla 22)**
✓	**R15. Como usuario, quiero eliminar un análisis físico (Tabla 23)**
✓	**R3. Como usuario, quiero editar mis ejercicios personalizados (Tabla 24)**
✓	**R9. Como usuario, quiero editar una marca en un ejercicio (Tabla 25)**
✓	**R16. Como usuario, quiero editar un análisis físico (Tabla 26)**
✗	**R33. Como desarrollador, tengo que configurar la infraestructura para que el sistema pueda funcionar en la nube (13)**
✗	**R22. Como sistema, tengo que ofrecer el sistema de pago premium (13)**
✗	**R5.1. Como usuario, quiero que se me muestre una foto que describa el ejercicio (8)**
✗	**R14. Como usuario, quiero visualizar consejos e información de las métricas de mis análisis físicos (3)**
✗	**R25. Como usuario, quiero visualizar de forma inmediata la última marca de cada ejercicio al entrar en su histórico (3)**
✗	**R28. Como usuario, quiero poder iniciar sesión con mi cuenta de Google, Facebook y/o Twitter (8).**
✗	**R29. Como usuario, quiero buscar ejercicios por categoría o tipo (2)**
✗	**R30. Como sistema, quiero ordenar la lista de ejercicios por frecuencia de uso (2)**
✗	**R27. Como usuario, quiero borrar varias marcas a la vez con algún criterio temporal (M)**
✗	**R31. Como usuario, quiero que pueda seleccionar el intervalo de tiempo de las marcas a visualizar por las gráficas (M)**
✗	**R32. Como usuario, quiero tener alguna opción para gestionar mis ejercicios predeterminados (L)**
✗	**R20. Como sistema, tengo que ofrecer el sistema en el idioma que tenga por defecto el navegador del usuario (L)**
✗	**R23. Como usuario, quiero compartir mis resultados con mis amigos en redes sociales (L)**
✗	**R21. Como usuario, quiero exportar/importar datos a formatos externos de la aplicación (L)**
✗	**R26. Como sistema, quiero ofrecer las mismas funcionalidades en ambos clientes (XL)**
✗	**R24. Como usuario, quiero tener un sistema de gamificación tipo niveles o logros para medir mi progreso (XL)**

Tabla 17. Historia de usuario detallada de R36.

R36. Como desarrollador, tengo que implementar tests automáticos y realizar análisis de QA	
Esfuerzo estimado: 13 puntos	Cliente: Ninguno.
Prioridad: alta	Riesgo: alto
Criterios de aceptación	
Precondiciones: • Pasos a seguir: • Implementar tests automáticos de regresión para el servidor. • Realizar análisis estáticos con la herramienta de SonarQube para mejorar el apartado de QA. Resultado esperado: • Conseguir una cobertura de código mínima de un 80%. • Conseguir un nivel A en todas las métricas de análisis de la herramienta de SonarQube.	

Tabla 18. Historia de usuario detallada de R34.

R34. Como desarrollador, tengo que unificar los estilos de la interfaz de usuario de los dos clientes	
Esfuerzo estimado: 5 puntos	Cliente: móvil
Prioridad: alta	Riesgo: bajo
Criterios de aceptación	
Precondiciones: • No estar con la sesión iniciada. Pasos a seguir: • Iniciar la aplicación y en la pantalla de bienvenida pulsar el botón de registrarse. • Rellenar los campos de nombre, email, contraseña y repetir contraseña. • Hacer clic en el botón de registrarse. Resultado esperado: • Si la contraseña tiene una longitud menor a 5 o mayor a 20, mostrar un mensaje de error. • Si los campos de contraseña y repetir contraseña no coinciden, mostrar un mensaje de error. • Si ya existe en el sistema una cuenta registrada con el email rellenado mostrar, un mensaje de error. • Si alguno de los campos está vacío, mostrar un mensaje de error. • Si no hay ningún error en el proceso crear la cuenta y mostrar un mensaje de éxito.	

Tabla 19. Historia de usuario detallada de R17.

R17. Como usuario, quiero registrarme con una cuenta en la aplicación	
Esfuerzo estimado: 3 puntos	Cliente: móvil
Prioridad: alta	Riesgo: bajo
Criterios de aceptación	
Precondiciones: • No estar con la sesión iniciada. Pasos a seguir: • Iniciar la aplicación y en la pantalla de bienvenida pulsar el botón de registrarse. • Rellenar los campos de nombre, email, contraseña y repetir contraseña. • Hacer clic en el botón de registrarse. Resultado esperado: • Si la contraseña tiene una longitud menor a 5 o mayor a 20, mostrar un mensaje de error. • Si los campos de contraseña y repetir contraseña no coinciden, mostrar un mensaje de error. • Si ya existe en el sistema una cuenta registrada con el email rellenado mostrar, un mensaje de error. • Si alguno de los campos está vacío, mostrar un mensaje de error. • Si no hay ningún error en el proceso crear la cuenta y mostrar un mensaje de éxito.	

Tabla 20. Historia de usuario detallada de R19.

R19. Como usuario, quiero editar mi cuenta	
Esfuerzo estimado: 3 puntos	Cliente: móvil
Prioridad: media	Riesgo: bajo
Criterios de aceptación	
Precondiciones: • Haber iniciado sesión. Pasos a seguir: • Abrir el *navbar* de la aplicación y pulsar el botón de opciones. • Editar los campos que se deseen entre los de nombre, email, y contraseña nueva. • Rellenar el campo de contraseña actual con la contraseña que se tenía en el momento de iniciar sesión. • Hacer clic en el botón de guardar cambios. Resultado esperado: • Si la contraseña tiene una longitud menor a 5 o mayor a 20 mostrar un mensaje de error. • Si ya existe en el sistema una cuenta registrada con el email rellenado mostrar un mensaje de error. • Si alguno de los campos de nombre, email, y/o contraseña actual está vacío, mostrar un mensaje de error. • Si la contraseña actual no coincide con la contraseña que el sistema tiene guardada para ese usuario mostrar un mensaje de error. • Si no hay ningún error en el proceso crear la cuenta y mostrar un mensaje de éxito.	

Tabla 21. Historia de usuario detallada de R4.

R4. Como usuario, quiero eliminar mis ejercicios personalizados	
Esfuerzo estimado: 3 puntos	Cliente: móvil
Prioridad: media	Riesgo: bajo
Criterios de aceptación	
Precondiciones: • Haber iniciado sesión. • Tener un ejercicio personalizado ya creado. Pasos a seguir: • Abrir el *navbar* de la aplicación y pulsar el botón de ejercicios. • Buscar el ejercicio que se desea eliminar. • Hacer clic en el botón de eliminar. Resultado esperado: • Si no hay ningún error, el ejercicio se elimina, desaparece de la lista, y se muestra un mensaje de retroalimentación al usuario.	

Tabla 22. Historia de usuario detallada de R10.

R10. Como usuario, quiero eliminar una marca en un ejercicio	
Esfuerzo estimado: 3 puntos	Cliente: móvil
Prioridad: medio	Riesgo: bajo
Criterios de aceptación	
Precondiciones: • Haber iniciado sesión. • Estar en la pantalla de ejercicios. • Tener una marca ya creada en un ejercicio. Pasos a seguir: • Hacer clic en el ejercicio del que se quiere eliminar una marca. • Buscar y seleccionar la marca que se quiere eliminar. • Hacer clic en el botón de eliminar. Resultado esperado: • Si no hay ningún error, la marca se elimina, desaparece de la lista, y se muestra un mensaje de retroalimentación al usuario.	

Tabla 23. Historia de usuario detallada de R15.

R15. Como usuario, quiero eliminar un análisis físico	
Esfuerzo estimado: 3 puntos	Cliente: móvil
Prioridad: medio	Riesgo: bajo
Criterios de aceptación	
Precondiciones: • Haber iniciado sesión. • Tener un análisis físico ya creado. Pasos a seguir: • Abrir el *navbar* de la aplicación y pulsar el botón de análisis físicos. • Buscar y seleccionar el análisis que se desea eliminar. • Hacer clic en el botón de eliminar. Resultado esperado: • Si no hay ningún error, el análisis corporal se elimina, desaparece de la lista, y se muestra un mensaje de retroalimentación al usuario.	

Tabla 24. Historia de usuario detallada de R3.

R3. Como usuario, quiero editar mis ejercicios personalizados	
Esfuerzo estimado: 3 puntos	Cliente: móvil
Prioridad: medio	Riesgo: bajo
Criterios de aceptación	
Precondiciones: • Haber iniciado sesión. • Tener un ejercicio ya creado. Pasos a seguir: • Abrir el *navbar* de la aplicación y pulsar el botón de ejercicios. • Buscar el ejercicio que se quiere editar. • Hacer clic en el botón de editar. • Modificar los campos del formulario. • Hacer clic en el botón de guardar marca. Resultado esperado: • Si se está modificando un ejercicio aeróbico, y el campo categoría no se encuentra entre [*Running, Swimming*], mostrar un mensaje de error. • Si se está modificando un ejercicio aeróbico, y el campo tipo no se encuentra entre [*Crawl, Butterfly, Trail runing, Cross running, Sprint*], mostrar un mensaje de error. • Si se está modificando un ejercicio anaeróbico, y el campo categoría no se encuentra entre [*Muscle training*], mostrar un mensaje de error. • Si se está modificando un ejercicio anaeróbico, y el campo tipo no se encuentra entre [*Chest, Back, Shoulder, Triceps, Biceps, Legs, Gluteus, abs*], mostrar un mensaje de error. • Si alguno de los campos está vacío, mostrar un mensaje de error. • Si no hay ningún error durante el proceso de modificación, el ejercicio se modificará y se actualizarán sus datos en la lista.	

Tabla 25. Historia de usuario detallada de R9.

R9. Como usuario, quiero editar una marca en un ejercicio	
Esfuerzo estimado: 3 puntos	Cliente: móvil
Prioridad: medio	Riesgo: bajo
Criterios de aceptación	
Precondiciones: • Haber iniciado sesión. • Estar en la pantalla de ejercicios. • Tener una marca ya creada en un ejercicio. Pasos a seguir: • Hacer clic en el ejercicio al que se quiere modificar la marca. • Buscar y seleccionar la marca que se quiere modificar. • Hacer clic en el botón de editar. • Modificar los campos del formulario. • Hacer clic en el botón de guardar marca. Resultado esperado: • Si el campo de distancia no está entre los valores [0,3431], mostrar un mensaje de error. • Si el campo de tiempo no está entre los valores [0,1440], mostrar un mensaje de error. • Si el campo de intensidad no está entre los valores [0,10], mostrar un mensaje de error. • Si el campo de frecuencia cardíaca no está entre los valores [0,225], mostrar un mensaje de error. • Si los campos de distancia y/o tiempo están vacíos, mostrar un mensaje de error. • Si no hay ningún error durante el proceso de modificación, la marca se modificará y se actualizarán sus datos en la lista.	

Tabla 26. Historia de usuario detallada de R16.

R16. Como usuario, quiero editar un análisis físico	
Esfuerzo estimado: 3 puntos	Cliente: móvil
Prioridad: medio	Riesgo: bajo
Criterios de aceptación	
Precondiciones: • Haber iniciado sesión. • Tener un ejercicio físico ya creado. Pasos a seguir: • Abrir el *navbar* de la aplicación y pulsar el botón de análisis físicos. • Buscar y seleccionar el análisis físico que se quiere modificar. • Hacer clic en el botón de editar. • Modificar los campos del formulario. • Hacer clic en el botón de guardar análisis físico. Resultado esperado: • Si el campo de peso no está entre los valores [0,400], mostrar un mensaje de error. • Si el campo de I.M.C. no está entre los valores [0,200], mostrar un mensaje de error. • Si el campo de edad metabólica no está entre los valores [0,150], mostrar un mensaje de error. • Si el campo de metabolismo basal no está entre los valores [0,10000], mostrar un mensaje de error. • Si el campo de grasa corporal no está entre los valores [0,100], mostrar un mensaje de error. • Si el campo de masa muscular no está entre los valores [0,150], mostrar un mensaje de error. • Si el campo de masa ósea no está entre los valores [0,50], mostrar un mensaje de error. • Si el campo de líquidos corporales no está entre los valores [0,100], mostrar un mensaje de error. • Si el campo de adiposidad visceral no está entre los valores [0,59], mostrar un mensaje de error. • Si el campo de aporte calórico diario no está entre los valores [0,50000], mostrar un mensaje de error. • Si ya se ha creado un análisis físico en ese mismo día, mostrar un mensaje de error. • Si no hay ningún error durante el proceso de modificación, el análisis físico se modificará y se actualizarán sus datos en la lista.	

5.3.2 Velocidad de desarrollo

Como muestra el diagrama de *burnup* de la Figura 23, durante la primera semana se completó únicamente la primera entrada de la pila (de 13 puntos), ya que las dos primeras eran demasiado grandes como para completarlas en tan pocos días. En la segunda semana se completaron las 3 entradas de la pila siguientes (de 5, 3 y 3 puntos respectivamente), haciendo un total de 12 puntos. Finalmente, a lo largo de la tercera y última semana, se completaron las otras 6 entradas de la pila restantes (cada una de 3 puntos). Así pues, el último día finaliza con los 42 puntos completados que se tenían previstos para este sprint.

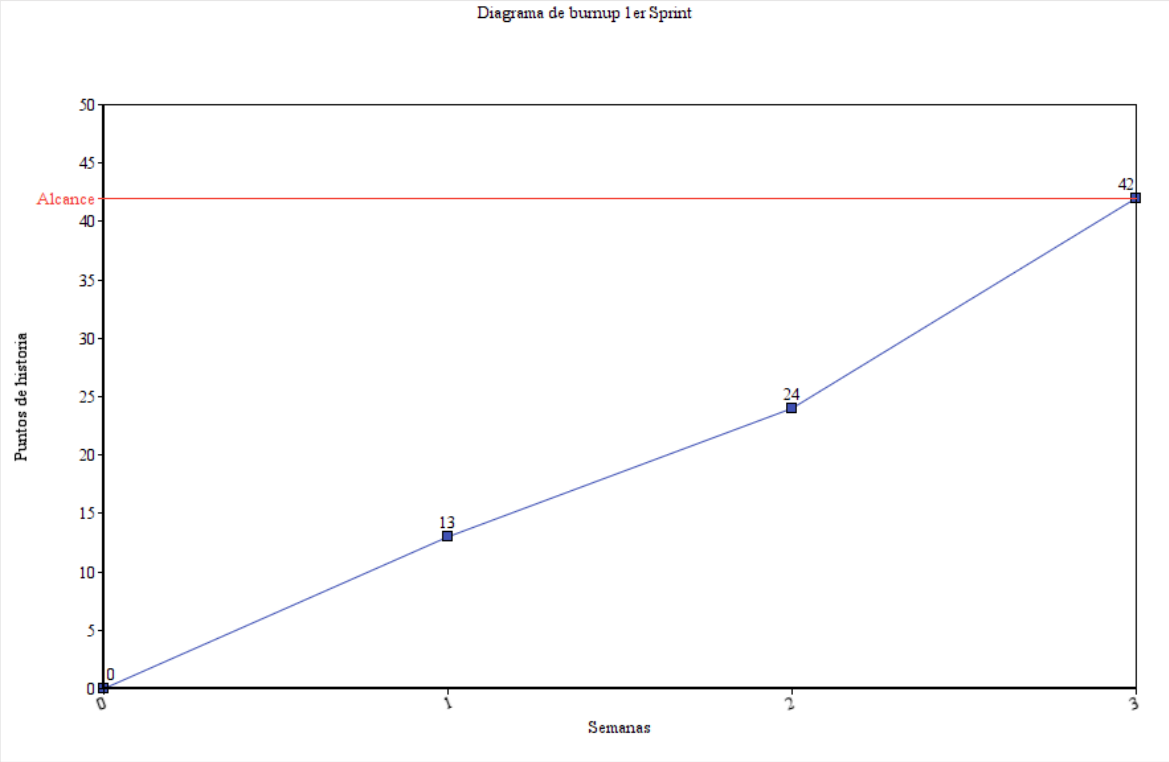

Figura 23. Diagrama de trabajo del sprint 1.

En el desarrollo se han completado un total de 10 entradas de la pila, todas las planificadas para el sprint, lo cual hace un total de 42 puntos de historia según las estimaciones realizadas al comienzo del sprint. Por tanto, la estimación de la velocidad del equipo es de 42 puntos por sprint hasta que se tengan más datos del resto.

5.3.3 Resultado de la retrospectiva

Durante la retrospectiva realizada para este sprint se han analizado algunos aspectos que podrían mejorarse proponiendo algunas mejoras para algunos de ellos, pudiéndose poner en práctica en los próximos sprints. Estos aspectos se han analizado tomando como unidades de medida el 0, 1 y 2 (siendo el 0 un valor negativo y el 2 un valor positivo). Los aspectos analizados han sido: tests automáticos, diseño simple, nombres, refactorización, documentación de diseño, automatización del build, control de versiones, historias de usuario, estimación de entradas de la pila, ritmo uniforme, y herramientas scrum.

De todos ellos, el que ha salido negativo es el de tests automáticos, porque pese a que se tiene un alto porcentaje de cobertura de código y se tienen tests de regresión e integración, no se cuenta con tests automáticos de sistema y aceptación incluyendo la UI. Para solucionar esto, se plantea investigar para el siguiente sprint tecnologías como Protractor (Angular) y Selenium (Selenium).

5.4 Resultados finales de la calidad del código

Los resultados del esfuerzo de esta última iteración en mejorar la robustez y la calidad han hecho que en este punto del TFM, el sistema cuente con un 95.51% de cobertura de código realizados con la herramienta Mocha (mencionada anteriormente) en un total de 136 tests. A continuación, se puede ver el total de porcentaje de cobertura de test distribuidos por ficheros en la Figura 24.

Files	≡	●	●	●	Coverage
config	9	7	0	2	77.77%
controllers	428	409	0	19	95.56%
models	33	33	0	0	100.00%
security	16	14	0	2	87.50%
Folder Totals (4 files)	486	463	0	23	95.26%
Project Totals (26 files)	513	490	0	23	95.51%

Figura 24. Cobertura del sistema al finalizar el TFM.

Además, gracias a los estándares definidos internamente junto con las definiciones de hecho, conseguido mantener una alta calidad y mantenibilidad del código en el proyecto. Los análisis obtenidos se encuentran en la Tabla 27 mientras que una vista general del análisis se puede visualizar en la Figura 25.

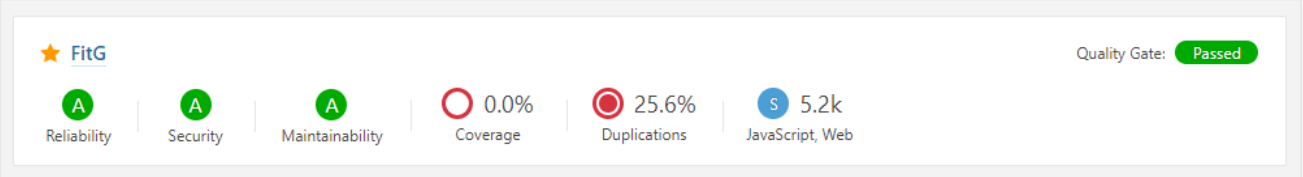

Figura 25. Vista general del análisis de SonarQube.

Tabla 27. Resultados de las métricas obtenidas con SonarQube.

Métrica	Definición	Resultado
Confiabilidad (Reliability)	Número de errores de programación en el código.	**A:** 0 errores
Seguridad (Security)	Número de vulnerabilidades en el código.	**A:** 0 vulnerabilidades
Mantenibilidad (Maintainability)	Número de incidencias de código deficiente. Se entiende por código deficiente cualquier síntoma en el código fuente de un programa que posiblemente indique un problema más profundo. Usualmente no son errores de programación, es decir, no son técnicamente incorrectos y en realidad no impiden que el programa funcione correctamente, sin embargo, indican deficiencias en el diseño que puede ralentizar el desarrollo o aumentan el riesgo de errores o fallos en el futuro.	**A:** 0 mejoras
Duplicaciones (Duplications)	Número de bloques duplicados de líneas.	**25.6%**
Tamaño (Size)	Tamaño de líneas de código.	**5.2K**
Complejidad (Complexity)	Es la complejidad calculada en base al número de rutas a través del código. Siempre que el flujo de control de una función se divide, el contador de complejidad se incrementa en uno. Cada función tiene una complejidad mínima de 1. Este cálculo varía ligeramente según el idioma.	**811**
Incidencias (Issues)	Número de incidencias.	**0** incidencias

Como se puede observar, el proyecto tiene el nivel A en todas las métricas establecidas en el proyecto con las correspondientes reglas. Además, cuenta con un porcentaje bastante bajo de código duplicado, con tan solo un 25.6% en sus 5200 líneas de código totales. Se cuenta actualmente con un total de 0 *issues* que añadan problemas al proyecto. Gracias a todo ello, el proyecto no cuenta con una deuda técnica y consigue de esta forma unos niveles altos de calidad en producción.

5.5 Desafíos técnicos del desarrollo

Los desafíos técnicos más interesantes que han aparecido a lo largo del proyecto son los relacionados con el desarrollo de la aplicación cliente de dispositivos móviles. Esta aplicación cliente involucra un cúmulo de tecnologías, lenguajes y *frameworks* desconocidas por el equipo de desarrolladores que todas se necesitaban desde el primer minuto para iniciar el desarrollo. Además, se ha hecho uso de sus versiones más estables más recientes, las cuales tienen alrededor de 1 año de vida, con lo que la documentación existente es aún escasa en comparación con el resto de las tecnologías existentes.

Este conjunto de *frameworks*, tecnologías y lenguajes nuevos se basa principalmente en 3. El primero es el *framework* de Ionic 2, del que se ha hecho uso de su versión 3, la cual salió en abril de 2017 (Carney, 2017). Este *framework*, a su vez, está desarrollado sobre la última versión de otro, el conocido como Angular, que en el momento de desarrollo estaba en su versión estable 5, que paso a estar disponible en noviembre de 2017 (Fluin, 2017). Por último y por si fuera poco, el *framework* de Angular no está desarrollado sobre JavaScript, si no que lo está sobre otro lenguaje de programación, TypeScript.

Dado que Ionic 2, Angular, y TypeScript, no habían sido utilizadas antes por los desarrolladores, esto hizo que fuera un desafío el tener que aprender las 3 a la vez en el poco tiempo que se tenía disponible (dada la naturaleza académica del proyecto).

Sin embargo, el peso pesado de estos tres era el *framework* de Angular. Esta tecnología cuenta con una abstracción software bastante elevada e introduce muchos conceptos, nuevos y ya existentes en el sector web, que se tienen que combinar en el desarrollo de una aplicación. Esto hace que la curva de aprendizaje sea realmente elevada al principio.

Los conceptos que son utilizados por esta tecnología son muchos y complejos, además de que en pocas líneas de código se pueden juntar fácilmente. Un ejemplo de ello se puede ver en la Figura 26 (código fuente del proyecto capturado en el momento de la redacción de este documento). En ella se ve como, por ejemplo, en una única función y en apenas 20 líneas de código aparecen varios conceptos de alto nivel como objetos observables (Angular, s.f.) con el patrón de *publish-subscriber*, las funciones flecha (Asim, s.f.), las promesas (Yeen, 2016), los *callbacks* (Richard, 2013)¸ los servicios (Simon, 2017) y el paso de información entre diferentes componentes (en este caso con los modales de Ionic 2).

```
addAnaerobicMark(): void {
  let addModal = this.modalCtrl.create( component: 'AnaerobicMarkCreatePage');
  addModal.onDidDismiss( callback: (mark) => {
    if (mark) {
          this.marksService.addAnaerobicMark(this.anaerobicExercise._id, mark)
            .then( onfulfilled: (observable: Observable<any>) => {
              observable.subscribe( next: (resp) => {

                let newMark = resp.mark as AnaerobicMark;
                // User created
                let toast = this.toastCtrl.create({
                  message: "Anaerobic mark successfully created.",
                  position: 'bottom',
                  duration: 3000,
                  cssClass: 'toast-success'
                });
                toast.present();
                this.marksList.unshift(newMark);
                this.marksShowList.set(newMark._id, false);

              }, error: (err) => {
                this.errorHandler(err.status, err.error.message)
              });
            }
          );
    }
  });
  addModal.present();
}
```

Figura 26. Extracto de código de una función del sistema FitG.

6. Lanzamiento del producto

Dados los niveles de robustez que tiene el código en el momento de finalizar el TFM, se considera que el sistema está listo para poder configurarle una infraestructura y así desplegarlo en la nube. Además, como resultado de la última iteración y la unificación de las interfaces, se ha logrado dotar a la interfaz gráfica del usuario final de cierta calidad para que fuera más atractiva y usable. A continuación, se puede ver la aplicación desarrollada en los clientes tanto de escritorio como de dispositivo móvil. Además, después se mostrarán los mapas de navegación del sistema:

6.1 FitG – Cliente de dispositivo móvil

La aplicación cliente de dispositivos móviles recoge las siguientes funcionalidades: Iniciar sesión, Registrar cuenta, Gestión de ejercicios nuevos, Gestión de marcas de ejercicio, Gestión de análisis físicos, y Editar Cuenta de usuario. Las Figuras 27 a 31 muestran las capturas de pantalla de FitG; para el diseño de las interfaces gráficas se han seguido las guías de estilo de Material Design (Google, s.f.c) para Android:

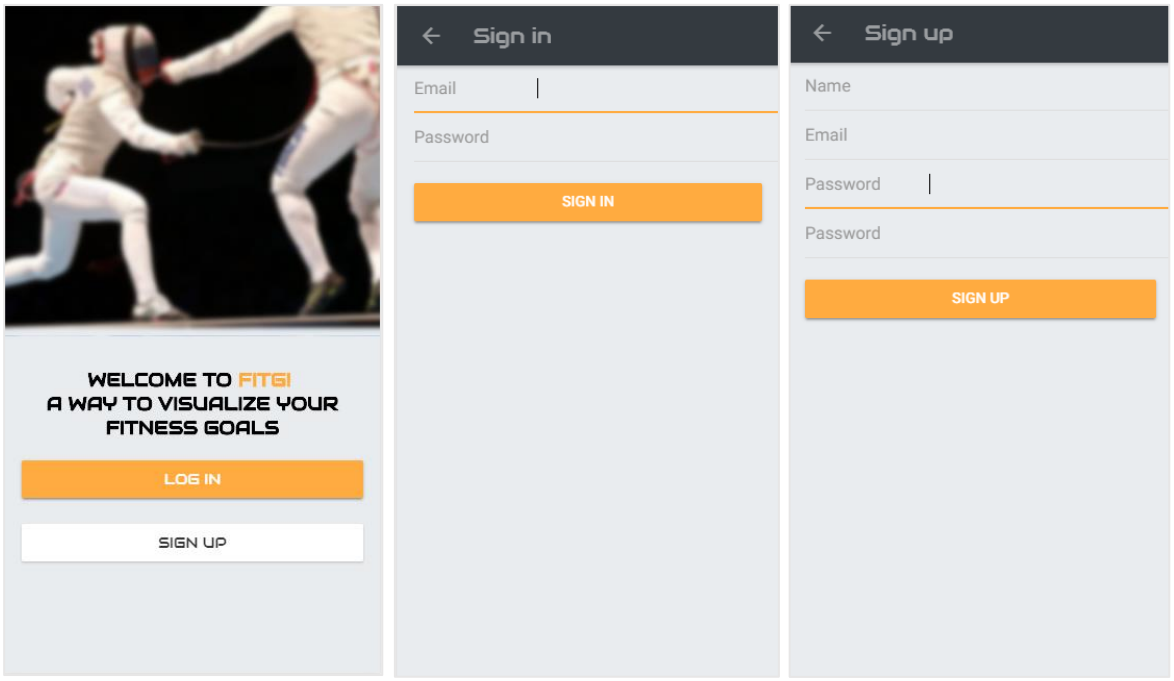

Figura 27. a. *Figura 27. b.* *Figura 27. c.*

Figura 27.Pantallas de dispositivo móvil: a) Bienvenida; b) Iniciar sesión; c) Registrar usuario.

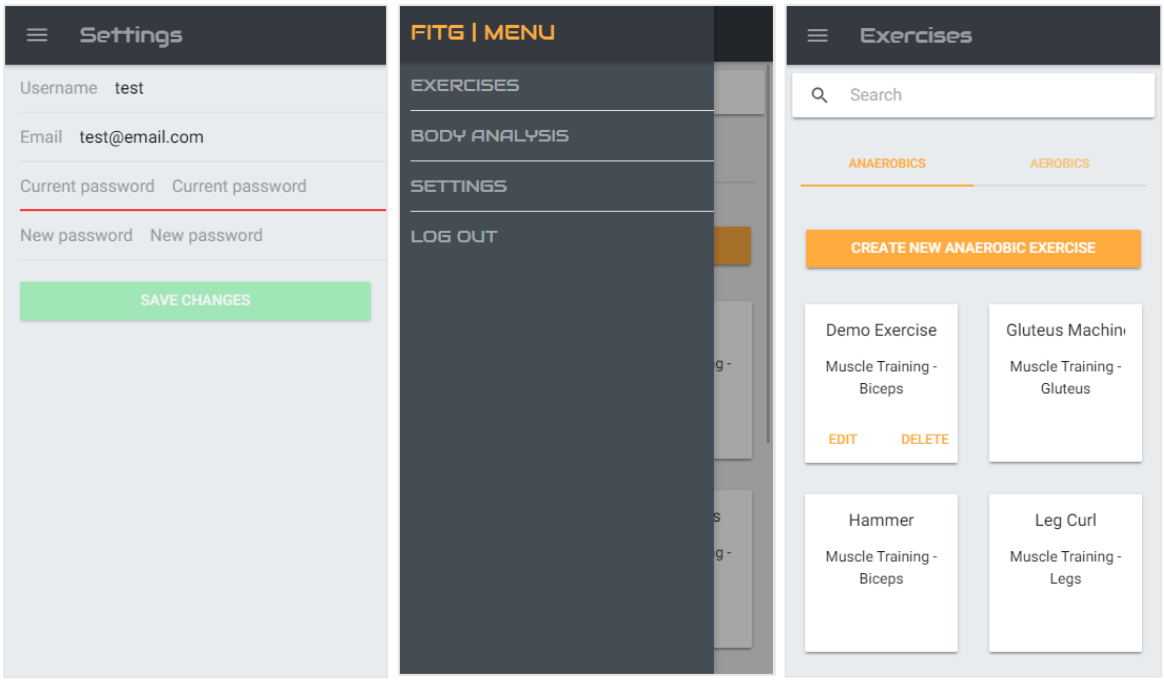

Figura 28. a. *Figura 28. b.* *Figura 28. c.*

Figura 28. Pantallas de dispositivo móvil: a) Editar cuenta de usuario; b) Navbar lateral; c) Listar ejercicios.

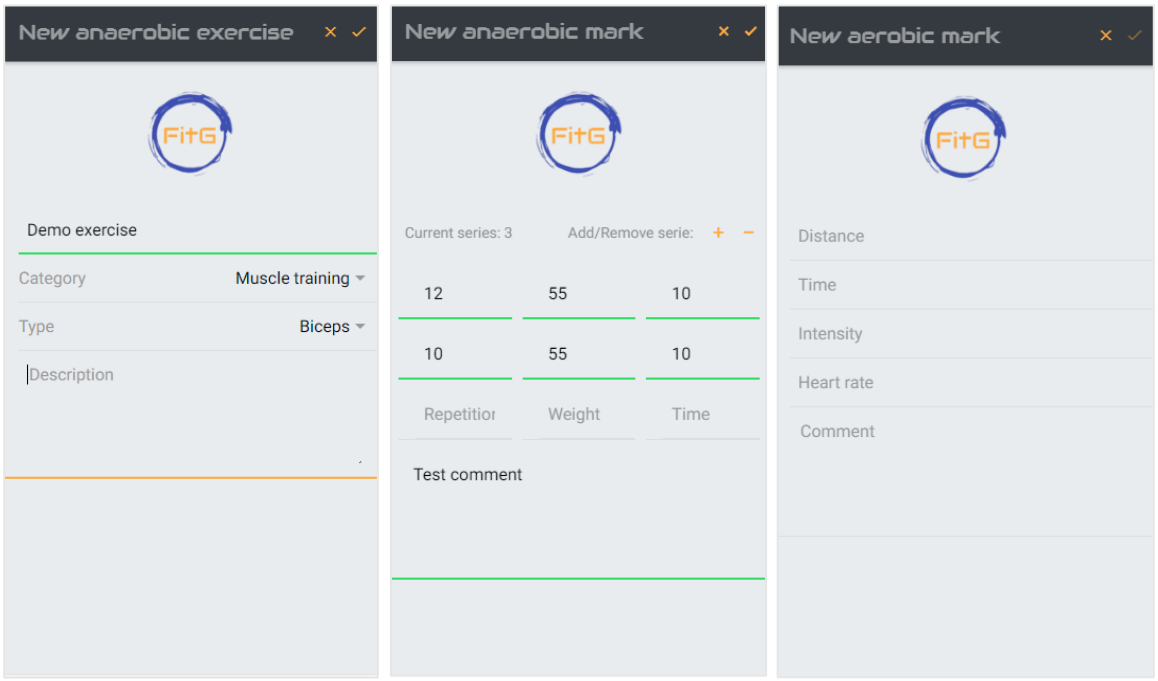

Figura 29. a. *Figura 29. b.* *Figura 29. c.*

Figura 29. Pantallas de dispositivo móvil: a) Crear/Editar ejercicios; b) Crear/Editar marcas de un ejercicio anaeróbico; c) Crear/Editar marcas de un ejercicio aeróbico.

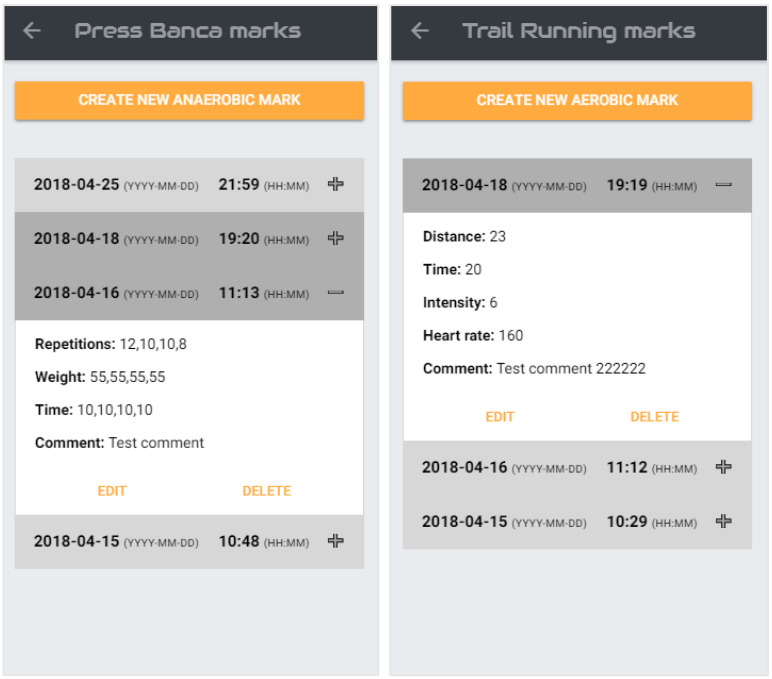

Figura 30. a. *Figura 30. b.*

Figura 30. Pantallas de dispositivo móvil: a) Listar marcas de un ejercicio anaeróbico; b) Listar marcas de un ejercicio aeróbico.

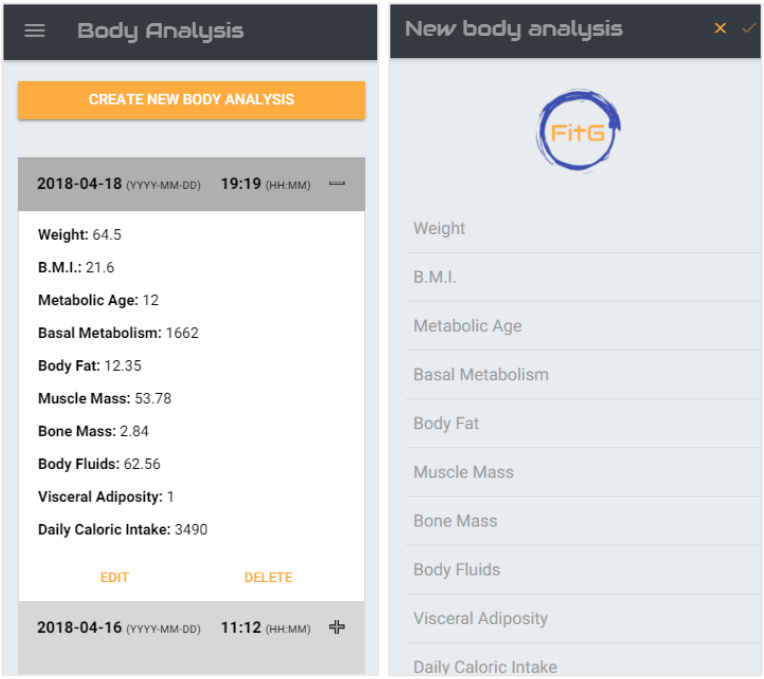

Figura 31. a. *Figura 31. B.*

Figura 31. Pantallas de dispositivo móvil: a) Listar análisis físicos; b) Crear/Editar análisis físicos.

6.2 FitG – Cliente de dispositivo móvil

La aplicación cliente de dispositivos escritorio recoge las siguientes funcionalidades: Iniciar sesión, Visualizar ejercicios, Visualizar marcas de ejercicio en detalle y en gráficas, Visualizar análisis físicos en detalle y en gráficas, y Personalizar las gráficas. A continuación, se pueden ver los resultados finales de las interfaces (ver Figuras 32 a 35), las cuales han seguido las guías de estilo de Material Design para escritorio (además de utilizar la misma temática en cuanto a colores, formas y letras que la aplicación cliente de dispositivos móviles para que tengan una apariencia similar y coherente entre las dos).

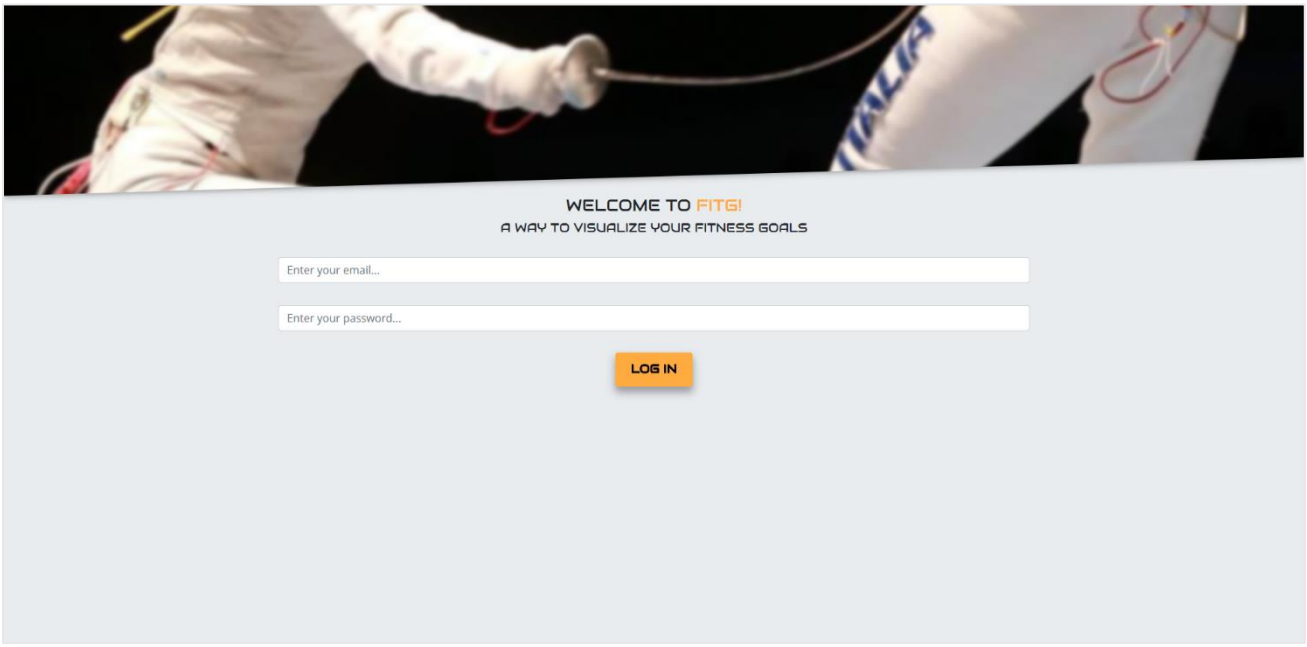

Figura 32. Pantalla de dispositivo de sobremesa: Bienvenida.

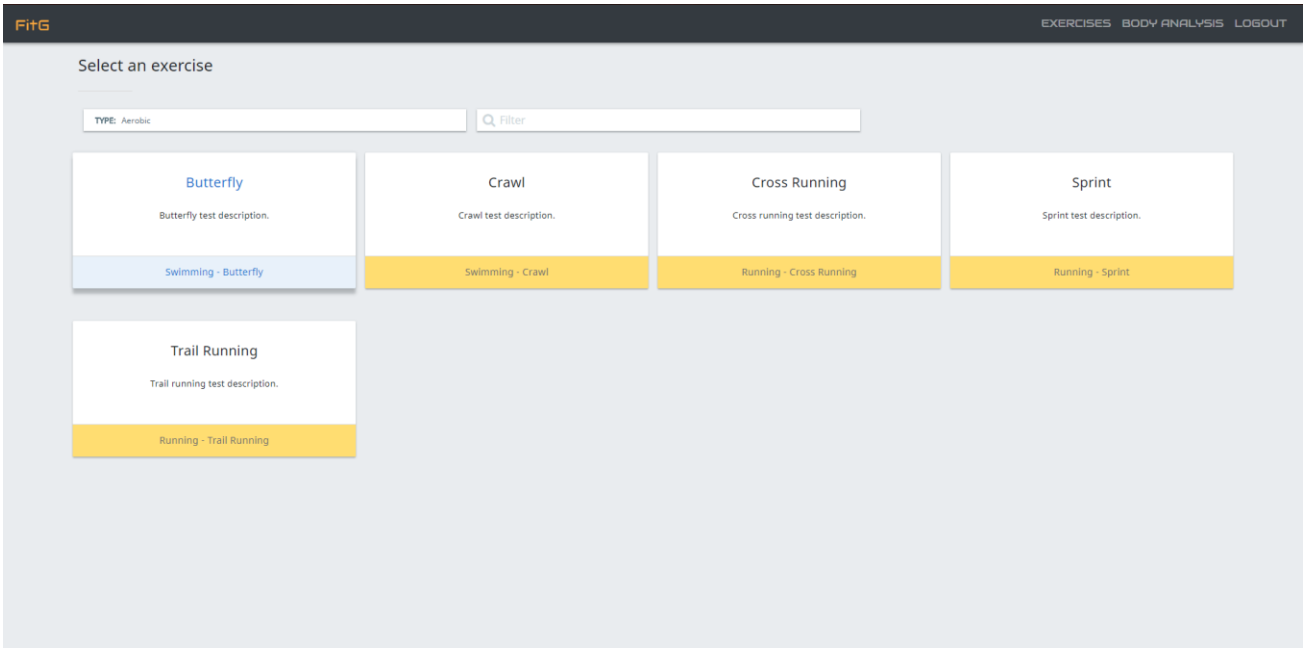

Figura 33. Pantalla de dispositivo de sobremesa: Listar ejercicios.

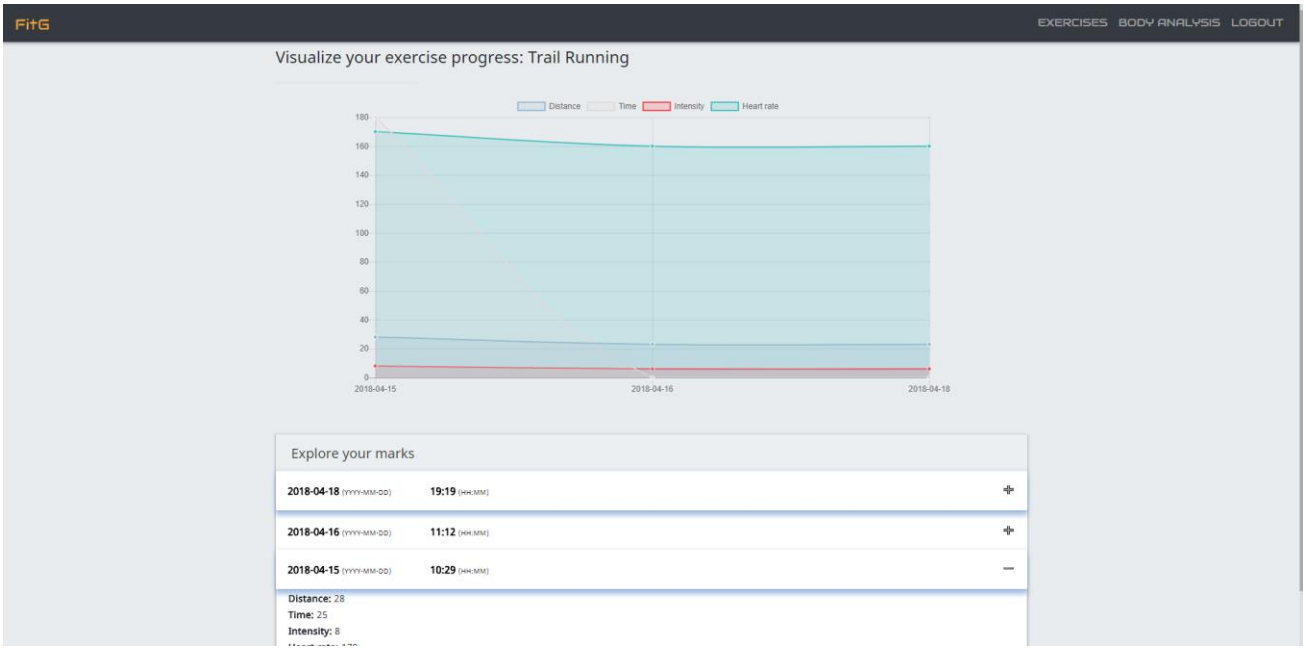

Figura 34. Pantalla de dispositivo de sobremesa: Visualizar progreso de las marcas.

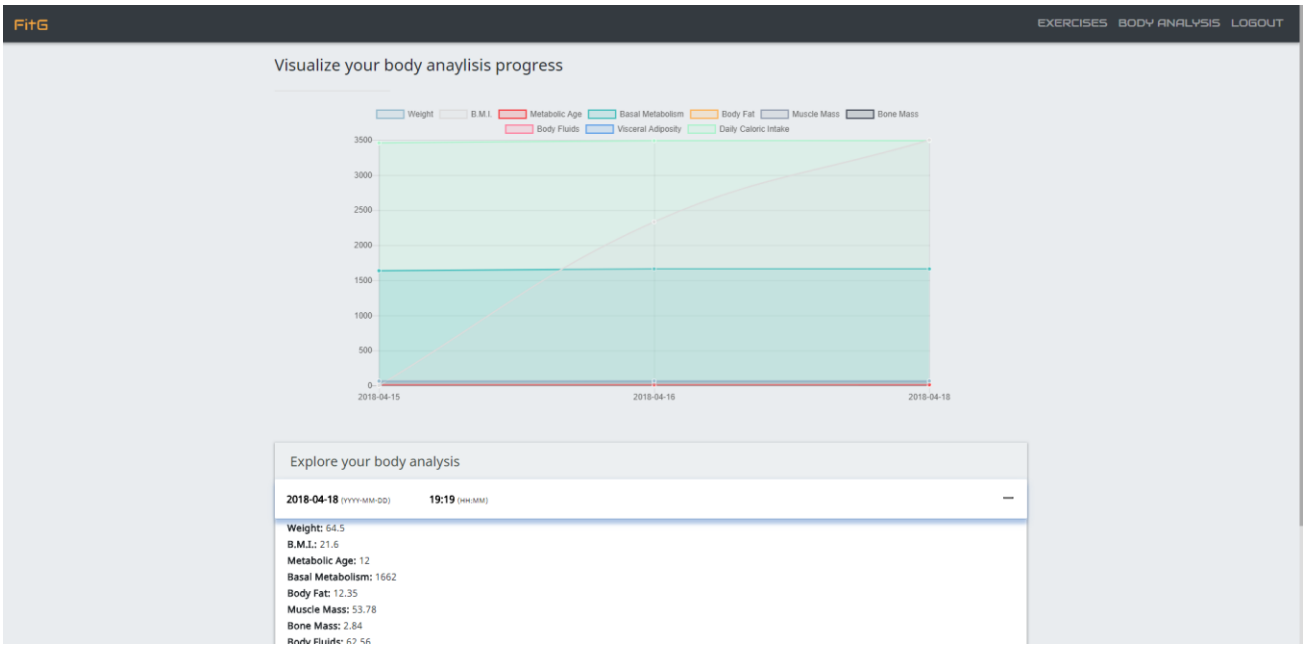

Figura 35: Pantalla de dispositivo de sobremesa: Visualizar progreso de los análisis físicos.

6.3 Mapas de navegación

La aplicación cuenta con dos mapas de navegación. Dado que cuenta con un número elevado de pantallas, se ha simplificado el mapa de navegación para que se puedan visualizar de una forma más legible en el formato actual del documento. Los nombres de las pantallas que aparecen se refieren a las mismas que las del apartado anterior.

Cliente de dispositivo móvil:

En este mapa de navegación existen algunas conexiones que no aparecen, esto se debe a que esas pantallas son accesibles a través del *navbar* lateral, una vez se haya iniciado sesión, desde cualquier otro estado. Estas pantallas se hay representado en el mapa con el borde rojo:

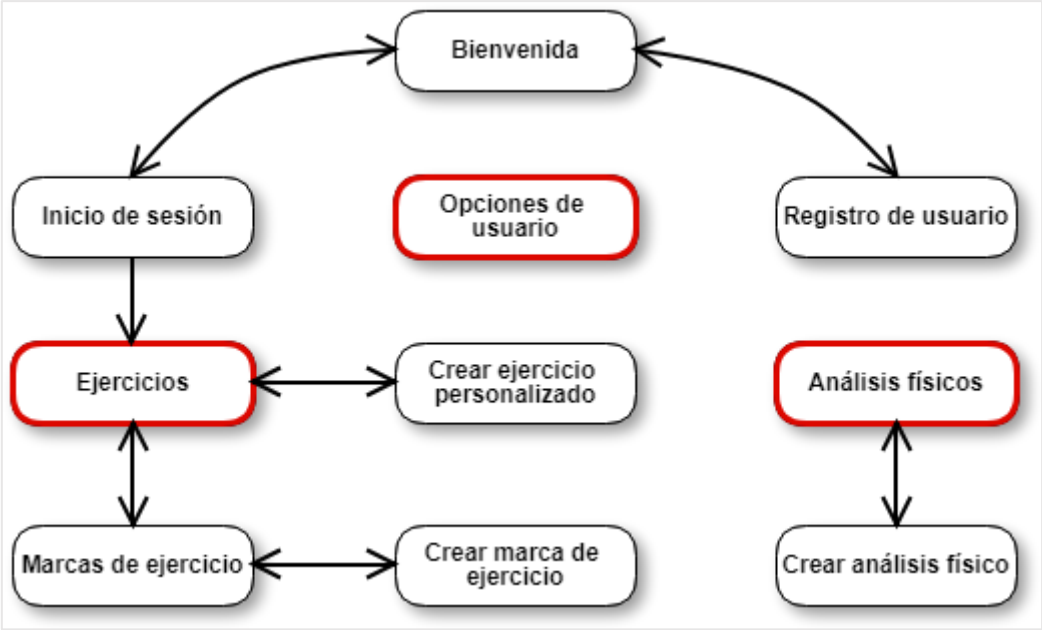

Figura 36. Mapa de navegación del cliente de dispositivos móviles.

Cliente de escritorio:

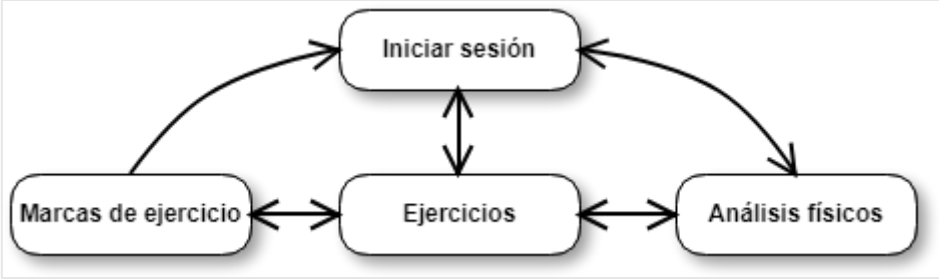

Figura 37. Mapa de navegación del cliente de dispositivos móviles.

6.4 Estrategia a seguir para la puesta en marcha del sistema

En lo que respecta a la puesta en marcha en la nube, y el lanzamiento del producto, se plantea la siguiente estrategia:

- Lo primero que se plantea es conseguir que la construcción del entorno del sistema esté estandarizada, y se obtenga el mismo resultado siempre a la hora de poderse desplegar en un servicio u otro. Esto, se conseguirá gracias a utilizar la tecnología de Docker, la cual permite crear contenedores con dichas características.
- Una vez realizado este paso, como ya se ha comentado en el apartado de "Solución tecnológica", el servidor web, así como la aplicación del cliente de escritorio estarían preparadas para poderse integrar con los servicios de Amazon S3. Esta estrategia, gracias a la flexibilidad de escalado que ofrecen, permitirá adaptarse al número de usuarios con un coste acorde a su uso.
- Realizar la compra del servicio de dominios DNS para "FitG", el cual está disponible en el momento en el que se escribe el documento (basado en terceros como GoDaddy, RaiolaNetwork, 1and1, y Hostalia). De esta forma se podrá acceder a la aplicación cliente de escritorio bajo el dominio www.fitg.com.
- Para el despliegue de la aplicación del cliente móvil, puesto que la tecnología del *framework* Ionic permite tener un sistema multiplataforma entre iOS y Android, se van a lanzar paralelamente en las dos tiendas especializadas (Apple Store y Google Play). Esto permitirá ese efecto deseado de con una sola aplicación, poder abarcar los dos mercados desde el principio de la comercialización. A ella se podrá acceder, buscando en las correspondientes tiendas, bajo el nombre de FitG, el cual también está disponible en el momento en el que se escribe el presente documento.
- Por último, las primeras acciones que se van a llevar a cabo respecto a la publicidad y marketing es la promoción por redes sociales centradas principalmente en Twitter y Facebook. Además, se va a promover la aplicación realizando algún tipo de convenio de descuentos en algunos gimnasios (por el momento en Zaragoza) para poder promocionarla como una herramienta de sus clientes.

Una vez se haya lanzado y puesto en marcha el producto, se realizará un seguimiento activo del progreso con algunas métricas. Estas métricas, como se describe previamente en la Tabla 1 (Lean Canvas) del capítulo 2.2., se van a centrar principalmente en analizar el número de usuarios activos, y en el número de descargas de la aplicación, tanto en la Apple Store como en Google Play. Además, estas tiendas permiten que los usuarios también dejen una retroalimentación, la cual será fundamental a corto plazo para ver si el producto sigue el curso correcto y las expectativas de los usuarios necesarias.

7. Conclusiones y trabajo futuro

7.1 Conclusiones

A lo largo del documento se ha comentado que los principales objetivos del trabajo eran los de crear un sistema web de gestión de las marcas que se realizan en los diferentes tipos de ejercicio y que todo ello estuviera orientado bajo el marco de un proyecto de emprendimiento en productos software. Se considera que dichos objetivos se han completado satisfactoriamente a lo largo del desarrollo, viéndose los resultados en el presente documento, tanto de la aplicación desarrollada como de su documentación y gestión. Así pues, el proyecto ha finalizado con los niveles de calidad, de validación y de robustez necesarios para cualquier sistema como producto de emprendimiento, y ha quedado en disposición para realizar un lanzamiento y puesta en marcha. Además, se ha dejado el proyecto preparado y con una estrategia a seguir de cara a dicho lanzamiento.

Como conclusiones generales, cabe destacar la importancia que ha tenido seguir el proceso de gestión y organización en productos software del marco Lean Startup de principio a fin del proyecto, teniendo en cuenta cada una de las fases documentadas aquí. Esto ha permitido poder evolucionar y desarrollar una idea novedosa como es *FitG*, de la que no se tenía del todo clara su trayectoria:

- Por una parte, gracias a la fase inicial de definición y evaluación del producto, la idea consiguió tomar forma rápidamente, pudiendo dar paso a fases posteriores como el análisis y diseño. Los beneficios que se obtuvieron en esta parte fueron los de dejar claras unas proyecciones y una dirección a seguir por el desarrollo del producto. Esto garantizó que se pudiera realizar una validación del producto, tanto del producto en el mercado con sus actuales competidores, como financieramente hablando a corto plazo, y así, saber si realmente merecía o no la pena iniciar el proyecto.
- Por otro lado, en lo que respecta al desarrollo en iteraciones y su propia gestión, se pudo ver como el utilizar la división especial en iteraciones del marco Lean Startup fueron de gran ayuda a lo largo del desarrollo del producto. De cara a las dos primeras, el utilizar conceptos como los del mínimo producto viable (MVP) fueron realmente útiles y necesarios. Gracias a este sistema, se pudo comprobar de una forma rápida si esto era lo que realmente necesitaban los usuarios reales y si se iba por un buen camino, o hacía falta pivotar algún aspecto del desarrollo, gracias a esa retroalimentación conseguida en los experimentos. De cara a la última iteración, también se ha visto la importancia de realizar esa transición de mapa de características y MVPs a un proceso ya más estricto de metodologías ágiles con *sprints,* pila de producto, estimaciones, criterios de aceptación, calidad, etc. Esto ayuda a que de cara ya al primer lanzamiento del producto, se normalice el desarrollo y el mantenimiento a procesos más regulares siguiendo metodologías ágiles como SCRUM, consiguiendo así mejores estimaciones y un desarrollo más óptimo en producción.

Cabe destacar en estas conclusiones que unos de los problemas más comunes a lo largo de este desarrollo han sido los retrasos de plazos y algunas alternativas descartadas o alcance recortado por los mismos. Estos inconvenientes han surgido precisamente por su propia naturaleza de incertidumbre al tratarse de un proyecto de emprendimiento, y a la vez ser un TFM realizado al mismo tiempo que el curso académico. No obstante, gracias a la tercera iteración, ya normalizada y transformada en *sprint*, se ha conseguido un mínimo de datos e historial de desarrollo con los que de cara al futuro poder realizar unas estimaciones mucho más precisas.

En el apartado de los objetivos personales también se han cumplido las expectativas. Se ha conseguido profundizar y coger los conocimientos necesarios para poder realizar aplicaciones completas pasando desde el *front-end*, por el *back-end* y hasta la capa de persistencia. Todo ello se ha conseguido con tecnologías actuales y recientes. De este punto en concreto, ha sido muy satisfactorio el haberse podido enfrentar a unos desafíos tecnológicos (como los nombrados en la sección Desafíos técnicos del desarrollo) en tan poco tiempo, pudiendo lograr el desarrollo completo y funcional de una aplicación completa. Por último, se ha podido realizar con éxito un proceso completo de desarrollo software, de un producto de emprendimiento siguiendo

las metodologías ágiles y Lean Startup, el cual también ha aportado la experiencia y conocimientos de haber realizado experimentos reales de productos en desarrollo consiguiendo así la validación de los usuarios.

7.2 Trabajo futuro

Con vistas al trabajo futuro ya se han ido dejando posibles mejoras durante el documento, que se resumen a continuación en las siguientes funcionalidades:

- Realizar una puesta en la nube del sistema para el servidor y la aplicación cliente de escritorio, siguiendo la estrategia ya planteada para poder configurar y desplegar la infraestructura necesaria.
- Subir la aplicación cliente de dispositivos móviles a las tiendas correspondientes (Apple Store y Google Play).
- Terminar el desarrollo de las entradas restantes en la pila del producto, apoyándose en el uso de las metodologías ágiles.
- Ampliar el número de deportes (ampliando el tipo y categoría de las entidades existentes y añadiendo las nuevas entidades de ejercicios mixtos aeróbico-anaeróbicos) aceptados por el sistema.
- Determinar de una forma específica la estrategia de marketing y publicidad que se va a seguir en las redes sociales de Twitter y Facebook para captar nuevos usuarios. Asimismo, acordar y especificar los convenios que se quieren tratar con los gimnasios locales.
- Establecer un proceso claro de recogida de retroalimentación de los usuarios para ir refinando y aumentando el producto en forma de nuevas entradas de la pila que se vayan planteando en el futuro.

Bibliografía

Amazon Web Services. (s.f.). *Amazon S3*. (Amazon Web Services Inc) Recuperado el 20 de marzo de 2018, de https://aws.amazon.com/es/s3/

Android. (s.f.). Recuperado el 26 de febrero de 2018, de https://developer.android.com/index.html

Angular. (s.f.). *Observables*. Recuperado el 16 de junio de 2018, de https://angular.io/guide/observables

Angular. (s.f.). *Protractor.* Recuperado el 09 de junio de 2018, de https://www.protractortest.org/#/

AngularJS. (s.f.). Recuperado el 15 de marzo de 2018, de https://angularjs.org/

Apple Inc. (s.f.). *Apple Developer*. Recuperado el 01 de abril de 2018, de https://developer.apple.com/

Asim. (s.f.). *Fat Arrow Functions*. Recuperado el 16 de junio de 2018, de https://codecraft.tv/courses/angular/es6-typescript/arrow/

Auth0. (s.f.). *JSON Web Tokens.* Recuperado el 15 de julio de 2018, de https://jwt.io/

Bernardo, A. (4 de septiembre de 2013). *9 pasos para que tu negocio sea un éxito a través del modelo Canvas.* Recuperado el 18 de marzo de 2018, de http://blogthinkbig.com/modelo-canvas-9-pasos-exito-negocio/

Carney, B. (07 de abril de 2017). *Ionic 3.0 has arrived!* Recuperado el 14 de junio de 2018, de https://blog.ionicframework.com/ionic-3-0-has-arrived/

ChaiJS. (s.f.). *Chai Assertion Library.* Recuperado el 03 de julio de 2018, de http://www.chaijs.com/

CMD Sport. (21 de marzo de 2016). *La práctica deportiva en España crece un 45,9%.* Recuperado el 11 de julio de 2018, de https://www.cmdsport.com/multideporte/actualidad-multideporte/la-practica-deportiva-en-espana-crece-un-459/

Codecov. (s.f.). Recuperado el 23 de marzo de 2018, de https://codecov.io/

DesarrolloWeb. (s.f.). *Javascript a fondo*. Recuperado el 22 de febrero de 2018, de https://desarrolloweb.com/javascript/

EditorConfig. (s.f.). Recuperado el 28 de mayo de 2018, de http://editorconfig.org

EXA Tools. (s.f.). *Ski Tracker.* Recuperado el 20 de marzo de 2018, de https://play.google.com/store/apps/details?id=com.exatools.skitracker

Express Framework. (s.f.). (StrongLoop Inc.) Recuperado el 15 de marzo de 2018, de http://expressjs.com/es/

Fluin, S. (01 de noviembre de 2017). *Version 5.0.0 of Angular Now Available.* Recuperado el 15 de junio de 2018, de https://blog.angular.io/version-5-0-0-of-angular-now-available-37e414935ced

Fowler, M. (01 de mayo de 2006). *Martin Fowler.* Recuperado el 16 de mayo de 2018, de https://www.martinfowler.com/articles/continuousIntegration.html

Git. (s.f.). *Distributed Git - Distributed Workflows.* Recuperado el 04 de marzo de 2018, de https://git-scm.com/book/en/v2/Distributed-Git-Distributed-Workflows

Google. (s.f.). *Google HTML/CSS Style Guide*. Recuperado el 05 de abril de 2018, de https://google.github.io/styleguide/htmlcssguide.html

Google. (s.f.). *Google JavaScript Style Guide*. Recuperado el 05 de abril de 2018, de https://google.github.io/styleguide/javascriptguide.xml

Google. (s.f.). *Google Play: App Store*. Recuperado el 05 de abril de 2018, de https://play.google.com/store

Google. (s.f.). *Material Design.* Recuperado el 13 de julio de 2018, de https://material.io/

Google's reCaptcha. (s.f.). Recuperado el 15 de marzo de 2018, de https://www.google.com/recaptcha/intro/android.html

Ionic Framework. (s.f.). Recuperado el 18 de febrero de 2018, de https://ionicframework.com/

Ionic Framework. (s.f.). *Ionic CLI Guide*. Recuperado el 15 de marzo de 2018, de https://ionicframework.com/docs/cli/

Jorgé. (16 de julio de 2016). *Your license to use React.js can be revoked if you compete with Facebook.* Recuperado el 02 de marzo de 2018, de https://react-etc.net/entry/your-license-to-use-react-js-can-be-revoked-if-you-compete-with-facebook

jQuery. (s.f.). Recuperado el 15 de marzo de 2018, de https://jquery.com/

Latorre, P., Baldassarri, S., & Cerezo, E. (2015). Evaluación sin usuarios: Heurísticas. En *Interacción Persona - Ordenador: Evaluación* (págs. 10-14). Zaragoza: Universidad de Zaragoza.

Leone, C. (31 de octubre de 2017). *How Much Should You Budget For Marketing In 2018?* Recuperado el 29 de abril de 2018, de https://www.webstrategiesinc.com/blog/how-much-budget-for-online-marketing-in-2014

Letelier, P. (2018). *Introducción a emprendimiento.* Valencia: Universidad Politécnica de Valencia.

Letelier, P. (2018). Técnicas para evaluación y desarrollo de ideas de producto. En *Emprendimiento en Productos Software* (págs. 14-15). Valencia: Universidad Politécnica de Valencia.

Lorenzana, D. (08 de octure de 2013). *¿Qué es el EBITDA de una empresa y cómo se calcula?* Recuperado el 29 de abril de 2018, de https://www.pymesyautonomos.com/administracion-finanzas/que-es-el-ebitda-de-una-empresa-y-como-se-calcula

Michal Marschall. (02 de julio de 2014). *CycleDroid.* Recuperado el 20 de Marzo de 2018, de https://play.google.com/store/apps/details?id=com.maral.cycledroid

Miller, G. A. (1956). The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review, 63*, 81-97.

Mocha. (s.f.). Recuperado el 23 de febrero de 2018, de https://mochajs.org/

MongoDB. (s.f.). *Sharding*. Recuperado el 19 de abril de 2018, de https://docs.mongodb.com/manual/sharding/

Node.js Foundation. (s.f.). *NodeJS.* Recuperado el 14 de julio de 2018, de https://nodejs.org/es/

npm, Inc. (s.f.). *npmjs*. Recuperado el 27 de marzo de 2018, de https://www.npmjs.com/

Patel, N. (16 de enero de 2016). *90% of Startups Fail: Here's What You Need to Know About the 10%.* Recuperado el 29 de abril de 2018, de https://www.forbes.com/sites/neilpatel/2015/01/16/90-of-startups-will-fail-heres-what-you-need-to-know-about-the-10/#313cc7166679

Raj, J. (17 de abril de 2014). *An introduction to the MEAN Stack.* Recuperado el 02 de marzo de 2018, de https://www.sitepoint.com/introduction-mean-stack/

React. (s.f.). Recuperado el 15 de marzo de 2018, de https://facebook.github.io/react/

Richard. (04 de marzo de 2013). *Understand JavaScript Callback Functions and Use Them.* Recuperado el 15 de junio de 2018, de http://javascriptissexy.com/understand-javascript-callback-functions-and-use-them/

Runtastic. (s.f.). *Runtastic.* Recuperado el 20 de marzo de 2018, de https://play.google.com/store/apps/details?id=com.runtastic.android

Sadalage, P. (02 de octubre de 2014). *NoSQL Databases: An Overview.* Recuperado el 11 de marzo de 2018, de https://www.thoughtworks.com/insights/blog/nosql-databases-overview

Sans, G. (15 de abril de 2015). *Angular - Introduction to ngNewRouter vs ui-router.* Recuperado el 18 de junio de 2018, de https://medium.com/angularjs-meetup-south-london/angular-just-another-introduction-to-ngnewrouter-vs-ui-router-72bfcb228017

Selenium. (s.f.). *Selenium.* Recuperado el 09 de junio de 2018, de https://www.seleniumhq.org/

Simon, G. (19 de abril de 2017). *Angular 4 Services Tutorial.* Recuperado el 15 de junio de 2018, de https://coursetro.com/posts/code/61/Angular-4-Services-Tutorial

SonarQube. (s.f.). Recuperado el 28 de mayo de 2018, de https://www.sonarqube.org/

Spring Framework. (s.f.). (Pivotal Software Inc.) Recuperado el 15 de marzo de 2018, de https://spring.io/

SSI Scuba Schools International. (s.f.). *MySSI.* Recuperado el 20 de marzo de 2018, de https://play.google.com/store/apps/details?id=com.divessi.ssi&hl=es

SurveyMonkey. (s.f.). *Likert Scale: What It Is & How to Use It*. Recuperado el 27 de mayo de 2018, de https://www.surveymonkey.com/mp/likert-scale/

Swagger. (s.f.). Recuperado el 17 de abril de 2018, de https://swagger.io/

TravisCI. (s.f.). Recuperado el 23 de febrero de 2018, de https://travis-ci.org/

Vagias, W. M. (2006). *Likert-Type Scale Response anchors.* Recuperado el 16 de mayo de 2018, de http://www.marquette.edu/dsa/assessment/documents/Sample-Likert-Scales.pdf

VGFIT LLC. (s.f.). *Fitness & Bodybuilding.* Recuperado el 08 de marzo de 2018, de https://play.google.com/store/apps/details?id=softin.my.fast.fitness

Vitonica. (15 de mayo de 2013). *Spinning: la escala de Borg para medir el esfuerzo.* Recuperado el 12 de mayo de 2018, de https://www.vitonica.com/spinning/spinning-la-escala-de-borg-para-medir-el-esfuerzo

w3schools. (s.f.). *The HTML DOM (Document Object Model)*. Recuperado el 12 de abril de 2018, de https://www.w3schools.com/js/js_htmldom.asp

Yeen, J. (01 de diciembre de 2016). *JavaScript Promises for Dummies.* Recuperado el 15 de junio de 2018, de https://scotch.io/tutorials/javascript-promises-for-dummies

Zacarias, D. (s.f.). *The Complete Guide to the Kano Model*. Recuperado el 29 de abril de 2018, de https://foldingburritos.com/kano-model/

Anexo A: Análisis y Diseño del sistema

A.1 Modelo de datos

Presentación primaria

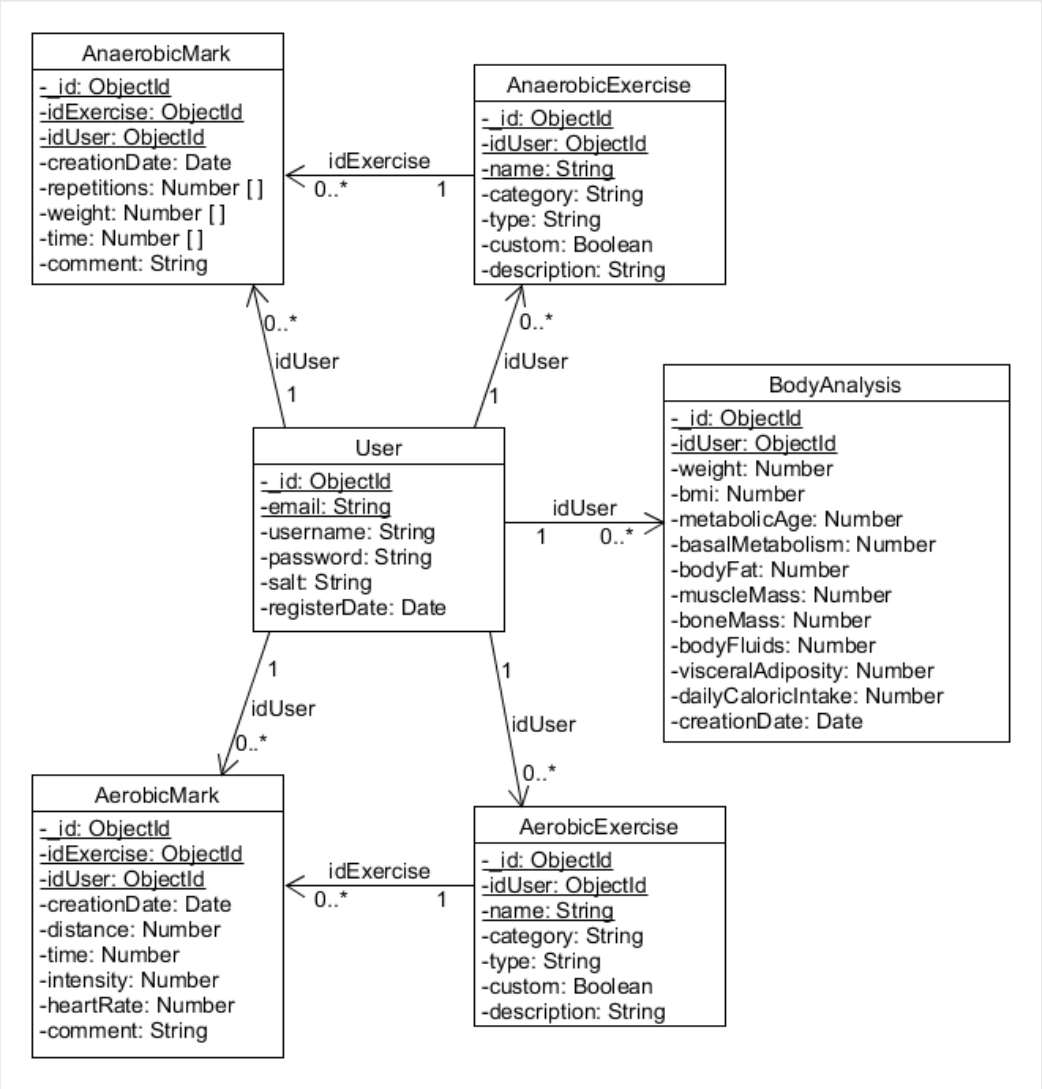

Figura 38. Diagrama de modelo de datos del sistema.

Catálogo de la vista

- **User:** Un usuario registrado en el sistema. Incluye los datos básicos del registro de un usuario y la fecha de creación. La contraseña está cifrada mediante una combinación del atributo *salt* (un *string* generado aleatoriamente por usuario) y el algoritmo sha512, lo que otorga una mayor seguridad.
- **BodyAnalysis:** Se trata de un análisis físico. Incluye todos los datos que se pueden obtener de un análisis físico, ya sean dados por una báscula o calculados con fórmulas matemáticas.
- **AnaerobicMark**: Marcas de un ejercicio anaeróbico. Incluye los factores comunes de posibles datos que se quieren almacenar en ejercicios de este tipo: tiempo de duración de la serie, peso utilizado en la serie, y número de repeticiones de la serie. Incluye un comentario opcional y su fecha de creación.
- **AerobicMark**: Marcas de un ejercicio aeróbico. Incluye los factores comunes de posibles datos que se quieren almacenar en ejercicios de este tipo: distancia recorrida, duración del ejercicio, intensidad del

ejercicio (pensado para una escala tipo Borg (Vitonica, 2013)), y el ritmo cardiaco medio durante el ejercicio. Incluye un comentario opcional y su fecha de creación.

- **AnaerobicExercise**: Ejercicio anaeróbico. Incluye los atributos tipo *enum* "tipo" y "categoría" que ayudan a discretizar y categorizar los distintos ejercicios. Por el momento se tiene restringido a categoría: ejercicio muscular; y tipo: pecho, espalda, hombro, tríceps, bíceps, piernas, glúteos, y abdominales. Además de una descripción del mismo y un booleano para saber si es un ejercicio predeterminado o personalizado.
- **AerobicExercise**: Ejercicio aeróbico. Incluye los atributos tipo *enum* "tipo" y "categoría" que ayudan a discretizar y categorizar los distintos ejercicios. Por el momento se tiene restringido a categoría: correr y nadar; y tipo: crol, mariposa, *trail running*, *cross running*, y sprint. Además de una descripción del mismo y un booleano para saber si es un ejercicio predeterminado o personalizado.

Exposición de razones

Se han tomado numerosas decisiones de diseño a lo largo del desarrollo del proyecto en lo referente al modelo de datos del mismo. En primer lugar, centrándose en la entidad de BodyAnalysis, las decisiones tomadas han sido relacionadas al número de atributos que se deseaban almacenar. En este caso se barajaba la opción de o reducirlo al número básico que la media normal de gente conoce, o expandirlos a los que unos usuarios más expertos podrían conocer. Finalmente se ha optado por incluir un número más amplio, dejando todos como opcional para que la gente solo rellene los que ellos conozcan.

En lo relativo a los ejercicios tanto anaeróbicos como aeróbicos, las principales decisiones estuvieron relacionadas sobre cómo clasificarlos para poder unirlos todos bajo un número razonable de entidades. El primer acercamiento fue decidir que solo iba a haber una entidad por el tipo de ejercicio que se estaba realizando, entendiéndose como tipo a cómo afectan muscularmente al cuerpo. Aquí se vio que principalmente había 3 tipos, lo cual reducía bastante el número de entidades pudiendo clasificar todos los ejercicios bajo ellas. Estos eran: ejercicios anaeróbicos, aeróbicos, y mixtos. Estos últimos mezclan los dos primeros, y suelen ser deportes de equipo tipo fútbol, baloncesto, balonmano, etc. Por el momento no se han incluido, dejándolos para más adelante. Por último, había que encontrar una forma de, una vez clasificados por el tipo de actividad física que requiere el ejercicio, discretizarlos aún más. Esto se consiguió con los dos atributos mostrados. El atributo "categoría" se basa en organizarlos por tipo de deporte "correr, saltar, nadar, gimnasio, bicicleta" etc. El segundo atributo, "tipo", se basa en clasificar las categorías anteriores por sus respectivas disciplinas. Esto da como lugar a una clasificación que, conforme se expanda el producto, consigue clasificar de forma correcta casi cualquier deporte.

Por último, las decisiones tomadas con respecto a las marcas fueron menores. La base fue buscar los atributos comunes que siempre eran de interés en cada tipo de actividad física realizada. En el caso de las marcas aeróbicas eso fue suficiente, sin embargo, en el de las anaeróbicas no. Inicialmente, en el primer MVP todos los atributos eran valores únicos, lo que acabó causando problemas (ya explicados en los resultados de los experimentos) teniendo que tomarse la decisión de convertirlos en una lista de una longitud igual al número de series realizadas.

A.2 Vista de Componentes y Conectores

En el diagrama presentado a continuación pueden diferenciarse las tres capas de la arquitectura Modelo-Vista-Controlador. Después, se van a especificar los componentes pertenecientes a cada una de las capas.

Presentación primaria

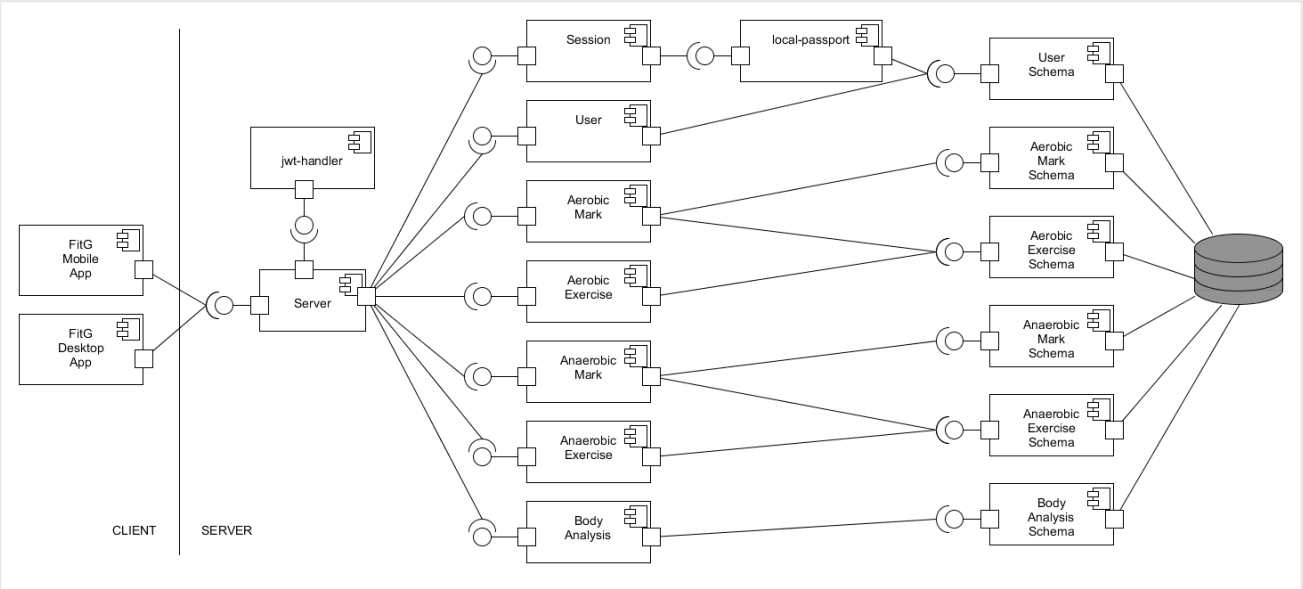

Figura 39. Diagrama de CyC.

Catálogo de la vista

- **Vista**:
  - **FitGDesktopApp**: Es el cliente para dispositivos desktop de la aplicación. Contiene todo el cliente desarrollado sobre AngularJS para los usuarios administradores y se puede comunicar haciendo las conexiones con los *end-points* vía API REST, con todos los componentes de los controladores.
  - **FitGMobileApp**: Es el cliente para dispositivos móviles de la aplicación. Contiene todo el cliente desarrollado sobre Ionic 2 y se puede comunicar haciendo las conexiones con los *end-points* vía API REST, con todos los componentes de los controladores.
- **Controlador**:
  - **Server**: Representación del server.js que inicializa todo el sistema, realiza todo el enrutado a los módulos que gestionan las peticiones, así como del módulo del control de acceso de usuarios. Requiere el componente jwt-handler dado que todas las peticiones pasan primero por el filtro para comprobar que la petición la ha realizado un usuario válido y que ya ha iniciado sesión.
  - **jwt-handler**: Es el encargado de gestionar el control de acceso al sistema. Este control de accesos se va a realizar con la tecnología JSON Web Tokens. Solo se puede hacer uso de los componentes habiendo iniciado sesión.
  - **Session**: Es el encargado de gestionar el inicio de sesión y generar un JSON Web Token válido y con la información del usuario.
  - **local-passport**: Ofrece los distintos tipos de inicio de sesión del sistema. Por el momento solo local al sistema, pero posible expansión con otras estrategias como *login* social.
  - **User**: Ofrece la interfaz para acceder a las funcionalidades tipo CRUD relacionadas con la gestión de usuarios.

- **AerobicMark**: Ofrece la interfaz para acceder a las funcionalidades tipo CRUD relacionadas con la gestión de marcas aeróbicas.
- **AerobicExercise**: Ofrece la interfaz para acceder a las funcionalidades tipo CRUD relacionadas con la gestión de ejercicios aeróbicos.
- **AnaerobicMark**: Ofrece la interfaz para acceder a las funcionalidades tipo CRUD relacionadas con la gestión de marcas anaeróbicas.
- **AnaerobicExercise**: Ofrece la interfaz para acceder a las funcionalidades tipo CRUD relacionadas con la gestión de ejercicios anaeróbicos.
- **BodyAnalysis**: Ofrece la interfaz para acceder a las funcionalidades tipo CRUD relacionadas con la gestión de análisis físicos.

- **Modelo:** Todos los componentes del modelo están conectados directamente con la base de datos MongoDB mediante el driver Mongoose**.**
  - **User Schema**: Componente que representa el esquema del modelo de datos de un usuario.
  - **AerobicMark Schema**: Componente que representa el esquema del modelo de datos de una marca aeróbica.
  - **AerobicExercise Schema**: Componente que representa el esquema del modelo de datos de un ejercicio aeróbico.
  - **AnaerobicMark Schema**: Componente que representa el esquema del modelo de datos de una marca anaeróbica.
  - **AnaerobicExercise Schema**: Componente que representa el esquema del modelo de datos de un ejercicio anaeróbico.
  - **BodyAnalysis Schema**: Componente que representa el esquema del modelo de datos de un análisis físico.

Exposición de razones

El *framework* MEAN lleva de forma casi natural a realizar una implementación siguiendo el patrón de diseño Modelo-Vista-Controlador, ofreciendo servicios RESTful orientados a recursos, por lo que se ha decidido usar dichos patrones para el desarrollo de la aplicación

Es por eso que nos encontramos los controladores divididos en componentes que se identifican como recursos web que pueden ser accedidos e identificados vía URI, que, en este caso, son accedidas desde la vista.

A.3 Vista de módulos

El diagrama de módulos presentado en esta sección representa las cuatro capas del sistema desarrollado: vista, controlador, modelo e infraestructura. Se va a explicar a continuación el contenido de cada capa, sin entrar demasiado en detalle.

Presentación primaria

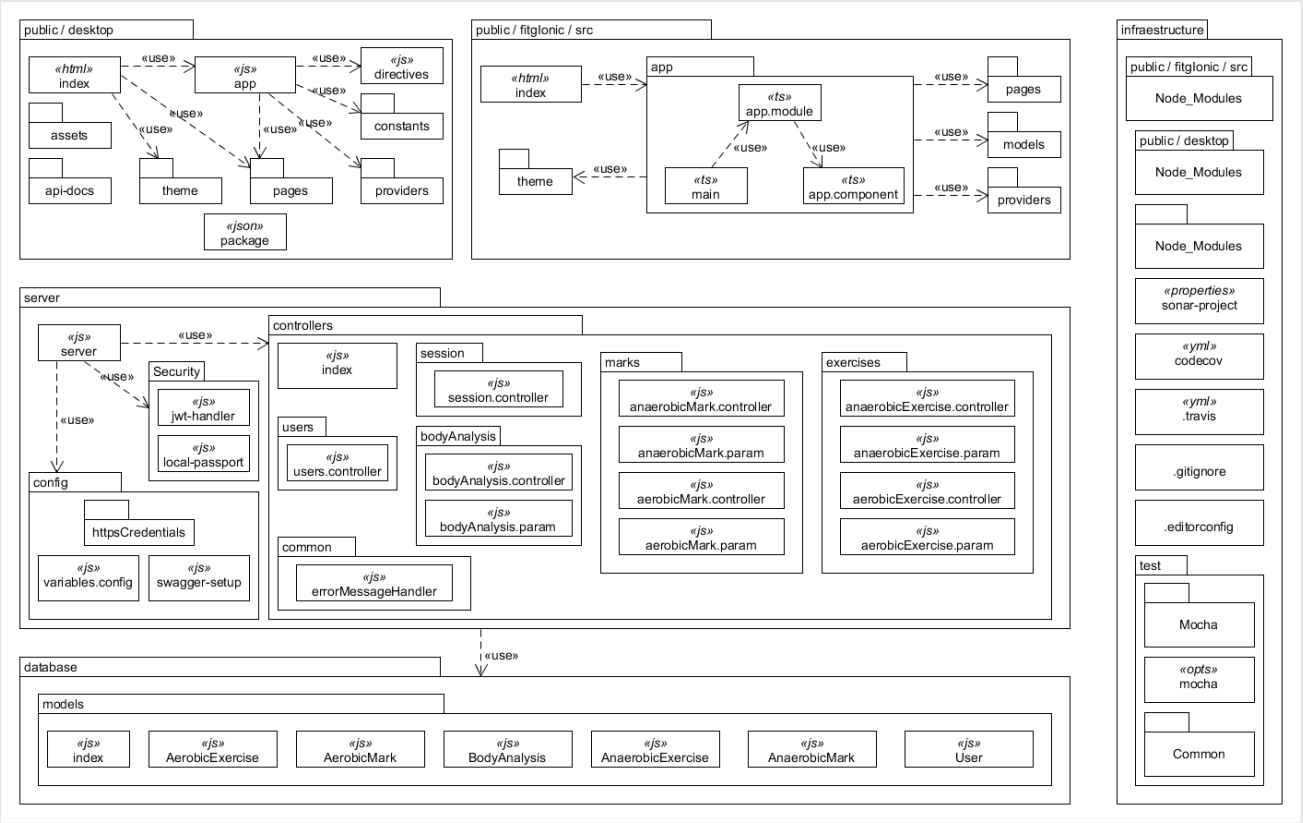

Figura 40. Diagrama de paquetes.

Catálogo de la vista

- **Módulo "public/desktop":** Representa los módulos y ficheros que conforman la parte de la vista de la aplicación del cliente de escritorio.
  - **index**: El fichero index.html de la aplicación web. Usa los módulos de CSS, así como el fichero app.js de la aplicación.
  - **theme**: Módulo que contiene todos los ficheros CSS comunes a las vistas de la aplicación.
  - **app**: El fichero app.js de la vista de la aplicación. Usa las vistas del módulo Pages y los ficheros JavaScript *directives*, *constants* y *providers*.
  - **pages** Módulo que contiene un directorio por cada componente/página de la vista. Cada directorio contiene su fichero de lógica JavaScript, su fichero *template* de HTML y su fichero de estilo CSS.
  - **assets**: Módulo que contiene todos los archivos multimedia utilizados en el cliente.
  - **providers**: Módulo que contiene los servicios con diferentes funciones para ser utilizadas en los componentes de la vista que se requiera.
  - **constants**: Módulo que contiene todas las constantes de AngularJS utilizadas en la aplicación.
  - **directives**: Módulo que contiene todas las directivas de AngularJS utilizadas en la aplicación.

- **api-docs**: Módulo que contiene los ficheros necesarios para la documentación de la API en Swagger.
- **package**: Fichero package.json del módulo Public que contiene las dependencias de módulos de la vista de la aplicación.

- **Módulo "public/fitgIonic/src":** Representa los módulos y ficheros que conforman la parte de la vista de la aplicación del cliente móvil.
  - **index**: El fichero index.html de la aplicación web. Usa el módulo main.ts de la aplicación para inicializarla.
  - **theme**: Módulo que contiene los colores principales de la aplicación para poder ser usados en cualquier componente de la misma.
  - **app**: Este módulo sirve para inicializar y ensamblar el sistema. Crea el componente principal del sistema en el que se desarrollarán los subsiguientes, así como inyecta las dependencias de los servicios del módulo *providers* y de librerías externas si las hay.
  - **pages**: Módulo que contiene un directorio por cada componente/página de la vista. Cada directorio contiene su fichero de lógica TypeScript, su fichero *template* de HTML y su fichero de estilo SCSS.
  - **models**: Módulo que contiene los modelos de objetos utilizados para la vista.
  - **providers**: Módulo que contiene los servicios con diferentes funciones para ser utilizadas en los componentes de la vista que se requiera.
- **Módulo "Server":** Representa los módulos y ficheros que conforman la parte del controlador de la aplicación.
  - **server**: Fichero server.js de la aplicación, encargado de crear el servidor, la conexión con la base de datos y fijar todas las rutas y módulos globales necesarios.
  - **config**: Fichero config.js de la aplicación. Contiene los ficheros *variables.config*, *swagger-setup* y el módulo *httpsCredentials*. En el primer fichero se encuentran las variables constantes del sistema, entre ellas la clave secreta empleada para la firma de los JSON Web Tokens emitidos. En el segundo fichero se encuentra la inicialización del *framework* de *swagger* para la documentación. Por último, en el módulo restante se encuentran las credenciales utilizadas para el cifrado y los certificados para el canal de comunicación del protocolo HTTPS.
  - **security**: Módulo que contiene los ficheros encargados de la gestión de los JSON Web Tokens empleados en la aplicación, así como de las estrategias para los diferentes tipos de inicios de sesión.
  - **controllers**: Módulo que contiene todos los ficheros con los *end-points* y la lógica de la aplicación.
    - **index**: Fichero que contiene la definición de las rutas para cada uno de los ficheros del módulo.
    - **users**: Módulo que cuenta con los ficheros que contienen la lógica de la aplicación en lo que se refiere a las funciones disponibles sobre los usuarios del sistema.
    - **bodyAnalysis**: Módulo que cuenta con los ficheros que contienen la lógica de la aplicación en lo que se refiere a las funciones disponibles sobre los análisis físicos.
    - **marks**: Módulo que cuenta con los ficheros que contienen la lógica de la aplicación en lo que se refiere a las funciones disponibles sobre las marcas de ejercicios.
    - **exercises**: Módulo que cuenta con los ficheros que contienen la lógica de la aplicación en lo que se refiere a las funciones disponibles sobre los ejercicios.
    - **common**: Módulo que contiene la gestión de mensajes de error según el tipo de código HTTP que hay que devolver al cliente.

        - **session**: Módulo que cuenta con los ficheros que contienen la lógica de la aplicación en lo que se refiere a las funciones disponibles sobre el control del inicio de sesión en el sistema.

- **Módulo "database":** Representa los módulos y ficheros que conforman la parte del modelo de la aplicación.
    - **models**: Módulo que contiene los ficheros de definición de modelos de datos de la aplicación.
        - **index**: Fichero que exporta globalmente los modelos de datos de la aplicación.
        - **User**: Fichero que contiene el modelo de datos de un usuario en el sistema.
        - **AerobicMark:** Fichero que contiene el modelo de datos de una marca aeróbica en el sistema.
        - **AerobicExercise:** Fichero que contiene el modelo de datos de un ejercicio aeróbico en el sistema.
        - **AnaerobicMark:** Fichero que contiene el modelo de datos de una marca anaeróbica en el sistema.
        - **AnaerobicExercise:** Fichero que contiene el modelo de datos de un ejercicio anaeróbico en el sistema.
        - **BodyAnalysis:** Fichero que contiene el modelo de datos de un análisis físico en el sistema.
- **Módulo "infraestructure":** Representa los módulos y ficheros que conforman la parte de infraestructura de la aplicación.
    - **./node_modules, public/desktop/node_modules, public/fitgIonic/src/node_modules**: Módulos que contienen las librerías/paquetes de npm empleados en las distintas partes de la aplicación: servidor, cliente de escritorio y cliente de dispositivos móviles respectivamente.
    - **sonar-project:** Fichero de configuración para la tecnología de SonarQube.
    - **codecov.yml:** Fichero de configuración para la tecnología de codecov.
    - **.travis.yml:** Fichero de configuración para la tecnología de travisCI.
    - **.gitignore:** Fichero de configuración para la tecnología de gitignore.
    - **.editorconfig:** Fichero de configuración para la tecnología de editorconfig.
    - **Test**: Módulo que contiene todos los ficheros y submódulos de testing del sistema.

Exposición de razones

Como ya se ha expuesto anteriormente, la aplicación sigue un patrón Modelo-Vista-Controlador, por lo que en el diagrama de módulos se han representado las distintas capas agrupando los módulos que pertenecen a cada una. No obstante, como se puede apreciar en dicho diagrama, la organización que se ha seguido en la aplicación es casi representativa de dicho patrón, agrupando los ficheros de cada capa en sus módulos correspondientes.

Tal y como dicta la arquitectura de capas que se ha representado, se ha respetado que tan sólo las capas superiores pueden utilizar las inferiores y nunca viceversa. La capa de infraestructura, aunque representada a un lateral, se sitúa en la parte más baja y todas las capas pueden acceder a ella.

A.4 Vista de distribución

Presentación primaria

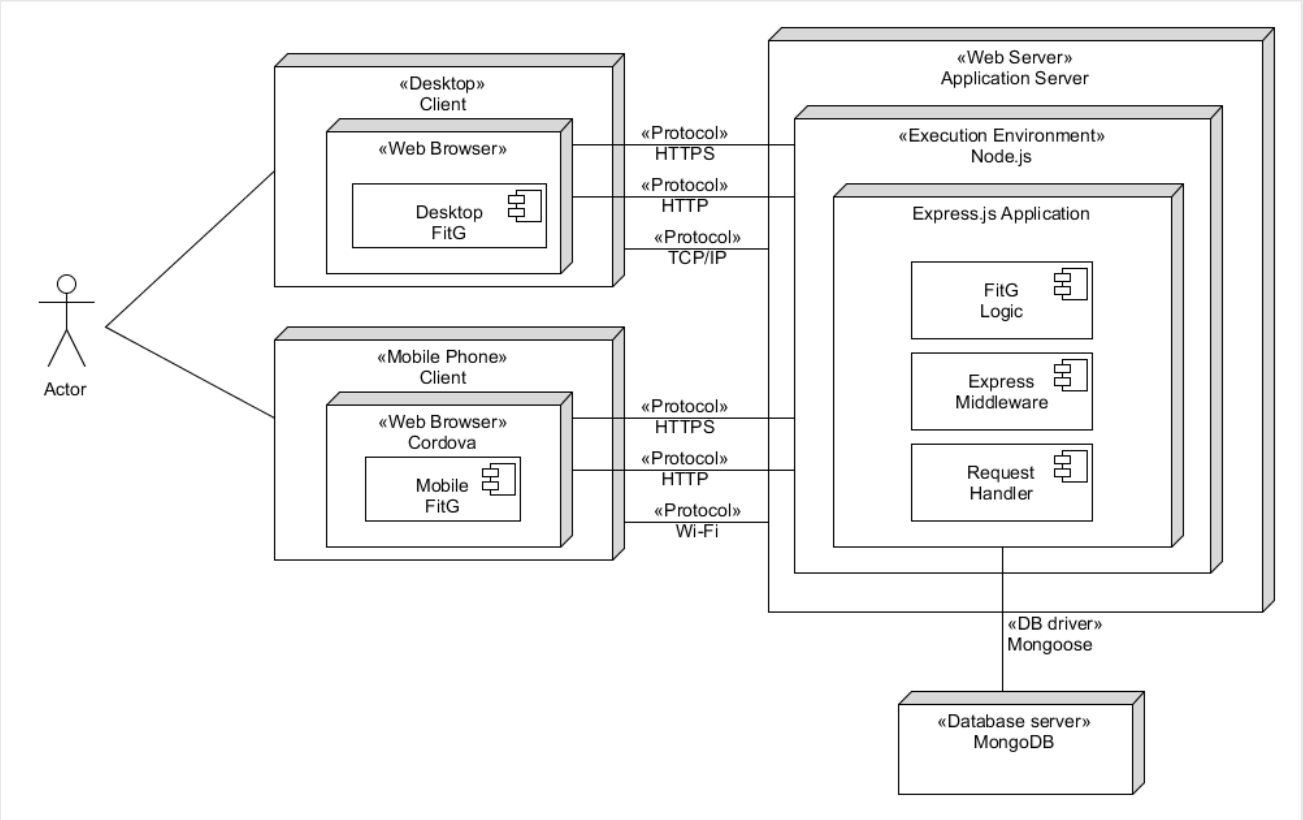

Figura 41. Diagrama de despliegue.

Catálogo de la vista

Siguiendo un orden de flujo de interacción se encuentran los siguientes componentes:

- El cliente *Mobile Phone*, componiéndose de un dispositivo móvil que puede ser iOS, Android, y Windows Phone. Incluye el navegador web Cordova para conseguir el funcionamiento híbrido. Este nodo se encarga de contener el cliente de la aplicación para los usuarios, comunicándose con los protocolos HTTP/s sobre Wi-Fi con el servidor en el puerto 8100.
- El cliente Desktop, que se compone de un dispositivo desktop, en el cual se incluye un navegador web. Este nodo se encarga de contener el cliente para los usuarios comunicándose con los protocolos HTTP/s sobre TCP/IP con el servidor en el puerto 8080 y 8443 respectivamente.
- El servidor web, componiéndose del entorno de ejecución Node.js en el cual se despliega Express, una aplicación *framework* que actúa como middleware, conteniendo a su vez la lógica de la aplicación.
- El servidor de base de datos documental MongoDB, el cual se conecta con la aplicación del servidor mediante un DB Driver, en este caso Mongoose.

Exposición de razones

Dado que la aplicación sigue el despliegue habitual para cualquier aplicación desarrollada con el *framework* MEAN, no se han tomado decisiones relevantes en cuanto a la distribución del sistema.

En la parte del cliente, puesto que la aplicación va a contar con dos tipos de clientes diferentes, se presentan dos nodos independientes actuando como clientes del servidor web con una arquitectura web cliente-servidor. Además, se ha tomado la decisión de dejar abiertos los dos servidores HTTP y HTTPS del cliente de

escritorio para que se pueda usar el que se desee, al menos hasta que el producto salga a producción y se replantee dicha decisión.

Anexo B: Detalles de implementación del sistema

B.1 Front-end – Cliente de escritorio

En lo referente a los detalles de la implementación del *front-end* de escritorio realizado con AngularJS, se van a aclarar algunos detalles y nombrar las librerías de las que se ha hecho uso.

Primero hay que aclarar que se ha seguido uno de los estándares que marca AngularJS para la organización del proyecto del cliente. De esta forma, se ha agrupado las vistas por directorios (dentro de la carpeta **/pages**), incluyendo cada una de ellas su propia *template*, controlador y fichero de estilo. Por otro lado, se han dejado todos los servicios en la carpeta **/providers**, así como los recursos utilizados en **/assets** y las constantes en el directorio **/constants**. Los estilos pertenecientes al sistema en sí (es decir, los más generales y no dependientes de cada vista) en la carpeta **/theme**. Por último, en la raíz de la carpeta se encuentran las directivas, el index.html, el package.json y el app.js.

Se ha decidido usar el gestor de vistas de ui-router en vez del ngRouter (Sans, 2015) puesto que además de ofrecer un mayor potencial y mejora de la flexibilidad gracias a su gestión de los estados, el equipo de desarrollo ya tenía experiencia en ese gestor.

Las librerías que se han utilizado han sido las siguientes:

- **Angular-char.js y chart.js**: Se han utilizado estas dos librerías para poder hacer uso de las gráficas que se muestran en la pantalla de estadísticas.
- **Angular-jwt**: Librería utilizada para codificar y decodificar los JWT utilizados en el sistema.
- **Angular-ui-notification**: Librería para mostrar notificaciones de diversa naturaleza.

Por último, añadir que para el correcto funcionamiento del control de accesos en la aplicación, se han utilizado interceptores de peticiones HTTP. Esto sirve para incluir automáticamente el JWT, si existe, en cada una de las peticiones antes de ser enviada, y comprobar si el servidor ha devuelto un código HTTP 401 (debido a problemas con el JWT). Al hacer la comprobación, se realizan las acciones correspondientes (p.ej.: comprobar que no hay errores en el JWT o si no existe y ha caducado, salir de la sesión…) antes de que la página siga con su flujo de ejecución correspondiente.

B.2 Front-end – Cliente de dispositivo móvil

En lo referente a los detalles de la implementación del *front-end* de dispositivos móviles realizado con Ionic 2 y Angular, se van a aclarar algunos detalles y nombrar las librerías de las que se ha hecho uso.

Primero hay que aclarar que se ha seguido uno de los estándares que marca Angular para la organización del proyecto del cliente, introduciendo todo el contenido no compilado de TypeScript en la carpeta **/src**. De esta forma, se ha agrupado las vistas por directorios (dentro de la carpeta **/pages**), incluyendo cada una de ellas su propia *template*, controlador y fichero de estilo.

En la carpeta **/app** se incluye el fichero de inicialización de la aplicación **main.ts**, el módulo **app.module.**ts para acoplar e inyectar todas las dependencias globales del sistema e indicar como inicializarlo, y el fichero **app.component.ts** que actúa como componente general de la aplicación, en la que van a ejecutar sus ciclos de vida el resto de componentes de la aplicación (además, este componente incluye su propia template actuando como navbar lateral).

Por último, en la carpeta **/assets** se encuentra todo el contenido multimedia utilizado en la aplicación. En la carpeta **/models** todos los modelos de entidades fijas. En la carpeta **/providers** todos los servicios utilizados. Y en la carpeta **/theme** los colores principales de la aplicación.

El resto de directorios y ficheros son los usados para compilar la aplicación y propios de los *frameworks* de Ionic 2 y Angular.

La única librería usada de forma extra a las que vienen por defecto en el package.json de aplicaciones de Ionic 2 ha sido la siguiente:

- **Angular2-jwt**: Librería utilizada para codificar y decodificar los JWT utilizados en el sistema.

B.3 Back-end

A continuación, se van a exponer algunos de las librerías (no pertenecientes al core del framework Express) y decisiones de implementación de las mismas tomadas a lo largo del desarrollo del proyecto en lo referente al *back-end* de la aplicación:

- **cors**: Módulo que actúa como middleware en las peticiones HTTP configurando la activación del CORS (*Cross-origin resource sharing)* , se ha utilizado en este caso para permitir que durante el desarrollo se puedan comunicar el servidor web que contiene la aplicación de Ionic 2 con el servidor web que contiene el back-end de Express sin ningún tipo de problema.
- **crypto**: Módulo que sirve para cifrar cadenas de caracteres. En este caso se ha usado para cifrar todas las contraseñas que se guardan en la base de datos.
- **express-jwt**: Módulo que actúa como middleware interceptando las peticiones que llegan al back-end para comprobar que las rutas que estén protegidas solo puedan ser accedidos por peticiones que contengan JSON Web Tokens válidos.
- **https**: Módulo que sirve para lanzar un servidor HTTPS en Node.js.
- **jsonwebtoken**: Módulo que ofrece varias funciones sobre JSON Web Tokens. Se emplea para firmar los tokens que se envían en el inicio de sesión de un usuario.
- **morgan:** Módulo que actúa como middleware para hacer un seguimiento e imprimir por pantalla las peticiones HTTP del servidor y así ayudar a tareas de depuración.
- **passport:** *Middleware* de autenticación para nodejs.
- **swagger-jsdoc:** Módulo que permite utilizar la herramienta de Swagger.
- **utf8**: La decodificación con base-64 da lugar a una ristra de bytes que más adelante hay que interpretar. Para ello, se usa el módulo utf8 para pasarlos a la codificación deseada.
- **winston**: *Logger* que sirve para crear distintos niveles de información para javascript y ayudar en las tareas de depuración.

B.4 Persistencia y capa de datos

Aunque *back-end* y persistencia van muy unidos en el framework MEAN, sí se pueden destacar algunos detalles de esta parte de la implementación.

El driver empleado para la comunicación con la base de datos MongoDB ha sido Mongoose, que, como se ha mencionado, ha facilitado la creación de esquemas y modelos para las colecciones en la base de datos. La conexión se crea en el fichero server.js, el primero en ejecutarse al lanzar la aplicación. Dicha conexión se crea con el puerto por defecto de MongoDB (27017) y con la base de datos "fitgDb". Además, también en el fichero **server.js**, se han asignado las rutas de los modelos en la aplicación con 'app.models'. De esta forma, resulta mucho más sencillo acceder a ellos desde cualquier parte del back-end de la aplicación.

Además, ha sido necesario modificar la variable "Promise" de Mongoose, fijándola a "global.Promise" dado que, de no hacerlo, se indicaba que las promesas que se estaban usando con ese modelo en algunas partes del código estaban *deprecated*.

Anexo C: Instrucciones y despliegue del sistema

Para realizar el despliegue del sistema, asumiendo que el sistema anfitrión ya tiene instaladas las tecnologías de Node.js y MongoDB, hay que seguir los siguientes pasos en el orden especificado, situándonos de partida en la raíz del proyecto:

1. Abrir una terminal de comandos y ejecutar '**npm install'**. Sólo será necesario la primera vez que se despliegue el sistema para descargar todas las dependencias.
2. Lanzar un servicio de MongoDB con el siguiente comando en la terminal: '**sudo service mongod start**'. Si se prefiere, también puede crearse una carpeta 'db' en la raíz del directorio y ejecutar el comando '**mongod - -dbpath db'**.
3. Añadir los ejercicios predeterminados en la base de datos. Para ello, ejecutar el fichero initialize.js mediante el comando '**node initialize.js'**.
4. Ejecutar en otra terminal el comando '**npm start**' o, alternativamente, el comando **'node server.js**'. Esto hace que se lance el servidor y se despliegue el cliente de escritorio. El servidor se habrá desplegado cuando la consola indique que se encuentra escuchando en los puertos 8080 (HTTP) y 8443 (HTTPS).
    a. Para emplear la aplicación de escritorio, abrir una venta de un navegador (preferiblemente Mozilla Firefox o Google Chrome) e introducir la URL 'http://localhost:8080' o 'http://localhost:8443'.
    b. Para ejecutar el cliente de dispositivos móviles primero hay que situarse en la carpeta correspondiente (con la consola: **'cd public/fitgIonic'**). A continuación, ejecutar el comando **'ionic serve'** para lanzar el servidor web. Esperar a que el servidor indique por consola que se ha inicializado, abrir un navegador web y acceder a la dirección 'http://localhost:8100'.

Por último, si se desea ejecutar los test se han de seguir las siguientes instrucciones (siempre con el servidor sin lanzar):

- Para ejecutar los test de Mocha en el proyecto, únicamente hay que ejecutar el comando '**npm test**' y se lanzarán automáticamente.

Anexo D: API RESTful del sistema

La mayoría de las peticiones de la API RESTful del sistema pueden devolver un código de respuesta HTTP 400, 401, y 500. El primero indica que la petición está mal formada ya que se ha utilizado de forma incorrecta o los parámetros no son los esperados. El segundo indica que el token de la sesión ha caducado, es inválido, o es inexistente. El tercero que ha ocurrido un error interno en el servidor. A continuación, se puede ver el formato que siguen:

HTTP 400 | 401 | 500	{ "message": string }

D.1 Análisis físicos

GET /bodyAnalysis	
Lista todos los análisis físicos.	
Request Headers	{ "Authorization": "Bearer " + JSON Web Token }
Request Body	Ninguno
Responses	
HTTP 200	{ "analysis": [{ "_id": string, "idUser": string, "bmi": number, "weight": number, "metabolicAge": number, "basalMetabolism": number, "bodyFat": number, "muscleMass": number, "boneMass": number, "bodyFluids": number, "visceralAdiposity": number, "dailyCaloricIntake": number, "creationDate": date }] }

GET /bodyAnalysis/{analysis}	
Lista toda la información de un análisis físico.	
Request Headers	{ "Authorization": "Bearer " + JSON Web Token, "analysis": string }
Request Body	Ninguno
Responses	
HTTP 200	{ "analysis": { "_id": string, "idUser": string, "bmi": number, "weight": number, "metabolicAge": number, "basalMetabolism": number, "bodyFat": number, "muscleMass": number, "boneMass": number, "bodyFluids": number, "visceralAdiposity": number, "dailyCaloricIntake": number, "creationDate": date } }

POST /bodyAnalysis	
Crea un análisis físico si no se ha creado ya uno en el mismo día.	
Request Headers	{ "Authorization": "Bearer " + JSON Web Token }
Request Body	{ "bmi": number, "weight": number, "metabolicAge": number, "basalMetabolism": number, "bodyFat": number, "muscleMass": number, "boneMass": number, "bodyFluids": number, "visceralAdiposity": number, "dailyCaloricIntake": number }
Responses	
HTTP 200	{ "analysis": { "_id": string, "idUser": string, "bmi": number, "weight": number, "metabolicAge": number, "basalMetabolism": number, "bodyFat": number, "muscleMass": number, "boneMass": number, "bodyFluids": number, "visceralAdiposity": number, "dailyCaloricIntake": number, "creationDate": date } }

PUT /bodyAnalysis/{analysis}	
Edita la información de un análisis físico ya existente.	
Request Headers	{ "Authorization": "Bearer " + JSON Web Token, "analysis": string }

PUT /bodyAnalysis/{analysis}	
Request Body	{ “bmi”: number, “weight”: number, “metabolicAge”: number, “basalMetabolism”: number, “bodyFat”: number, “muscleMass”: number, “boneMass”: number, “bodyFluids”: number, “visceralAdiposity”: number, “dailyCaloricIntake”: number }
Responses	
HTTP 200	{ “message”: string}

DELETE /bodyAnalysis/{analysis}	
Elimina un análisis físico ya existente.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “analysis”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “message”: string}

D.2 Ejercicios aeróbicos

GET /aerobicExercises	
Lista todos los ejercicios aeróbicos tanto predeterminados como los personalizados del usuario que realiza la petición.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token }
Request Body	Ninguno
Responses	
HTTP 200	{ “exercises”: [{ “_id”: string, “idUser”: string, “name”: string, “category”: string, “type”: string, “custom”: boolean, “description”: string }] }

GET /aerobicExercises/{aerobicExercise}	
Lista toda la información de un ejercicio aeróbico.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “aerobicExercise”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “exercise”: { “_id”: string, “idUser”: string, “name”: string, “category”: string, “type”: string, “custom”: boolean, “description”: string } }

POST /aerobicExercises	
Crea un nuevo ejercicio aeróbico.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token }
Request Body	{ “name”: string, “category”: string, “type”: string, “description”: string }
Responses	
HTTP 200	{ “exercise”: { “_id”: string, “idUser”: string, “name”: string, “category”: string, “type”: string, “custom”: boolean, “description”: string } }

PUT /aerobicExercises/{aerobicExercise}	
Edita la información de un ejercicio aeróbico ya existente únicamente si es personalizado.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “aerobicExercise”: string }
Request Body	{ “name”: string, “category”: string, “type”: string, “description”: string }
Responses	
HTTP 200	{ “message”: string}

DELETE /aerobicExercises/{aerobicExercise}	
Elimina un ejercicio aeróbico ya existente únicamente si es personalizado.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “aerobicExercise”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “message”: string}

D.3 Ejercicios anaeróbicos

GET /anaerobicExercises	
Lista todos los ejercicios anaeróbicos tanto predeterminados como los personalizados del usuario que realiza la petición.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token }
Request Body	Ninguno
Responses	
HTTP 200	{ “exercises”: [{ “_id”: string, “idUser”: string, “name”: string, “category”: string, “type”: string, “custom”: boolean, “description”: string }] }

GET /anaerobicExercises/{anaerobicExercise}	
Lista toda la información de un ejercicio anaeróbico.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “anaerobicExercise”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “exercise”: { “_id”: string, “idUser”: string, “name”: string, “category”: string, “type”: string, “custom”: boolean, “description”: string } }

POST /anaerobicExercises	
Crea un nuevo ejercicio anaeróbico.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token }
Request Body	{ “name”: string, “category”: string, “type”: string, “description”: string }
Responses	
HTTP 200	{ “exercise”: { “_id”: string, “idUser”: string, “name”: string, “category”: string, “type”: string, “custom”: boolean, “description”: string } }

PUT /anaerobicExercises/{anaerobicExercise}	
Edita la información de un ejercicio anaeróbico ya existente únicamente si es personalizado.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “anaerobicExercise”: string }
Request Body	{ “name”: string, “category”: string, “type”: string, “description”: string }
Responses	
HTTP 200	{ “message”: string}

DELETE /anaerobicExercises/{anaerobicExercise}	
Elimina un ejercicio anaeróbico ya existente únicamente si es personalizado.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “anaerobicExercise”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “message”: string}

D.4 Marcas aeróbicas

GET /aerobicExercises/{aerobicExercise}/aerobicMarks	
Lista todas las marcas del ejercicio aeróbico del usuario que realiza la petición.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “aerobicExercise”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “marks”: [{ “_id”: string, “idExercise”: string, “idUser”: string, “distance”: number, “time”: number, “intensity”: number, “heartRate”: number, “comment”: string, “creationDate”: date }] }

GET /aerobicExercises/{aerobicExercise}/aerobicMarks/{aerobicMark}	
Lista toda la información de una marca del ejercicio aeróbico del usuario que realiza la petición.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “aerobicExercise”: string, “aerobicMark”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “mark”: { “_id”: string, “idExercise”: string, “idUser”: string, “distance”: number, “time”: number, “intensity”: number, “heartRate”: number, “comment”: string, “creationDate”: date } }

POST /aerobicExercises/{aerobicExercise}/aerobicMarks	
Crea una nueva marca a un ejercicio aeróbico.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “aerobicExercise”: string }
Request Body	{ “distance”: number, “time”: number, “intensity”: number, “heartRate”: number, “comment”: string }
Responses	
HTTP 200	{ “mark”: { “_id”: string, “idExercise”: string, “idUser”: string, “distance”: number, “time”: number, “intensity”: number, “heartRate”: number, “comment”: string, “creationDate”: date } }

PUT /aerobicExercises/{aerobicExercise}/aerobicMarks/{aerobicMark}	
Edita la información de una marca ya existente de un ejercicio aeróbico.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “aerobicExercise”: string, “aerobicMark”: string }
Request Body	{ “distance”: number, “time”: number, “intensity”: number, “heartRate”: number, “comment”: string }
Responses	
HTTP 200	{ “message”: string}

DELETE /aerobicExercises/{aerobicExercise}/aerobicMarks/{aerobicMark}	
Elimina una marca ya existente de un ejercicio aeróbico.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “aerobicExercise”: string, “aerobicMark”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “message”: string}

D.5 Marcas anaeróbicas

GET /anaerobicExercises/{anaerobicExercise}/anaerobicMarks	
Lista todas las marcas del ejercicio anaeróbico del usuario que realiza la petición.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “anaerobicExercise”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “marks”: [{ “_id”: string, “idExercise”: string, “idUser”: string, “repetitions”: [number], “weight”: [number], “time”: [number], “comment”: string, “creationDate”: date }] }

GET /anaerobicExercises/{anaerobicExercise}/anaerobicMarks/{anaerobicMark}	
Lista toda la información de una marca del ejercicio anaeróbico del usuario que realiza la petición.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “anaerobicExercise”: string, “anaerobicMark”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “mark”: { “_id”: string, “idExercise”: string, “idUser”: string, “repetitions”: [number], “weight”: [number], “time”: [number], “comment”: string, “creationDate”: date } }

POST /anaerobicExercises/{anaerobicExercise}/anaerobicMarks	
Crea una nueva marca a un ejercicio anaeróbico.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “anaerobicExercise”: string }
Request Body	{ “repetitions”: [number], “weight”: [number], “time”: [number], “comment”: string }
Responses	
HTTP 200	{ “mark”: { “_id”: string, “idExercise”: string, “idUser”: string, “repetitions”: [number], “weight”: [number], “time”: [number], “comment”: string, “creationDate”: date } }

PUT /anaerobicExercises/{anaerobicExercise}/anaerobicMarks/{anaerobicMark}	
Edita la información de una marca ya existente de un ejercicio anaeróbico.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “anaerobicExercise”: string, “anaerobicMark”: string }
Request Body	{ “repetitions”: [number], “weight”: [number], “time”: [number], “comment”: string }
Responses	
HTTP 200	{ “message”: string}

DELETE /anaerobicExercises/{anaerobicExercise}/anaerobicMarks/{anaerobicMark}	
Elimina una marca ya existente de un ejercicio anaeróbico.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “anaerobicExercise”: string, “anaerobicMark”: string }
Request Body	Ninguno
Responses	
HTTP 200	{ “message”: string}

D.6 Sesión

POST /login	
Devuelve un JSON Web Token válido para mantener una sesión activa durante al menos 1h en el sistema.	
Request Headers	Ninguno
Request Body	{ “email”: string, “password”: string }
Responses	
HTTP 200	{ “token”: { “_id”: string, “email”: string, “username”: string } }

D.7 Usuarios

POST /users	
Crea un nuevo usuario en el sistema.	
Request Headers	Ninguno
Request Body	{ “username”: string, “password”: string, “rePassword”: string, “email”: string }
Responses	
HTTP 200	{ “message”: string }

PUT /users/{user}	
Edita la información de la cuenta de un usuario ya existente en el sistema únicamente si el usuario que ha realizado la petición se trata del mismo al que se le desean aplicar los cambios.	
Request Headers	{ “Authorization”: “Bearer “ + JSON Web Token, “user”: string }
Request Body	{ “username”: string, “oldPassword”: string, “password”: string, “rePassword”: string, “email”: string }
Responses	
HTTP 200	{ “message”: string}