UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Departamento de Sistemas Informáticos y Computación

PLANITNOW!: APLICACIÓN INTELIGENTE PARA LA GESTIÓN Y DESARROLLO DE PLANES

Memoria de Trabajo de fin de Máster

Máster Universitario en Ingeniería y Tecnología de Sistemas de Software

Autor: Eduardo Pertierra Puche
Director: Carlos Herrero Cucó
Codirector: Antonio Molina Marco

Curso: 2020-2021

RESUMEN

Este trabajo de fin de máster se centra en el desarrollo completo de una aplicación móvil que permite a los diferentes usuarios gestionar sus planes y actividades de forma inteligente. El propósito de este proyecto es el de implantar esta aplicación en el mercado existente con el fin de emprender un nuevo producto software.

ABSTRACT

This master final project focuses on the full stack development of a mobile application which allows end users manage their plans and activities in an intelligent way. The purpose of this project is to implant this application in the existing market in order to launch a new software product.

PALABRAS CLAVE

Lenguaje Natural; Emprendimiento; Redes Sociales; Aplicación; Android; Sistemas de Recomendación

TABLA DE CONTENIDO

CONTEXTO

Este documento hace referencia a un Trabajo de Final de Máster de la Universidad Politécnica de Valencia, concretamente del Máster Universitario en Ingeniería y Tecnología de Sistemas Software [1]. Este proyecto está enfocado esencialmente al emprendimiento de un nuevo producto software que idealmente estará a disposición de los diferentes usuarios una vez el proyecto sea finalizado.
A lo largo de este documento se referirá al proyecto en general como ***PlanItNow!*** ya que este es su nombre comercial y abreviación. El nombre formal será *Aplicación Inteligente para la Gestión y Desarrollo de Planes"*, un nombre que trata de ser lo más autodescriptivo posible

1. INTRODUCCIÓN

PlanItNow! nace a través de diferentes trabajos realizados a lo largo del máster en cuestión, comenzando como una idea *freelance* de gestión y manipulación de datos en la asignatura *Data Science* [2], para posteriormente validar la idea de negocio durante la asignatura de *Emprendimiento en Productos Software*, resultando no sólo en una idea de negocio completa, sino también en un trabajo final de máster con diferentes desafíos tecnológicos como pueden ser el desarrollo de una aplicación desde cero tanto a nivel de *front-end*[1] como de *back-end*[2], desafíos que se podrán ver reflejados en posteriores capítulos de este documento.
Por otro lado, emprender siempre ha supuesto un reto tanto para estudiantes como para personas que ya se han desarrollado a lo largo de los años en el mundo empresarial, es por ello que este trabajo trata de ser un nexo entre el mundo académico, el tecnológico y el empresarial, de tal forma que además de resultar idealmente en un exitoso proyecto de software, pueda servir como referencia a que estudiantes con grandes ideas puedan lanzarlas como Trabajo de fin de Grado o fin de Máster.
Respecto a lo que se pretende lograr con ***PlanItNow!*** es facilitar a las personas el hecho de poder organizar de forma efectiva sus planes y quedadas[3], así como estar al día respecto a planes y actividades por su zona, logrando así emplear el teléfono móvil a modo de brújula social. Por otro lado, a los diferentes negocios se les dará la oportunidad de dar visibilidad a sus actividades y ofrecer actividades en las que los usuarios puedan participar.
Para lograr esto, se dispondrá de una aplicación móvil implementada en el sistema operativo Android. Además de las funcionalidades básicas de gestión de planes, se implementará un algoritmo de recomendación que sea capaz de asociar de forma efectiva a los usuarios con los planes y actividades más afines que existan dentro de la aplicación, teniendo en cuenta diversos factores clave de los usuarios para realizar estas recomendaciones.

[1] Parte visual de una aplicación de software. Es aquella que se encarga de interactuar directamente con el usuario y de mostrar los datos.

[2] Parte lógica de una aplicación de software, aquella que corresponde a un servidor. Esta parte se encarga de realizar la gestión de los datos de la aplicación así como las funcionalidades complejas.

[3] En este documento se referirá como planes o quedadas a una actividad realizada en un período concreto de tiempo ya sea en solitario o con un conjunto de personas.

1.1. MOTIVACIÓN

Este proyecto surge a través de diferentes trabajos académicos como se ha mencionado a lo largo de la introducción, sin embargo, sí que existen motivos fundamentales y trascendentales para evaluar la idea de crear una aplicación que trate de gestionar los diferentes planes. Por otro lado, cabe destacar que el desarrollar una aplicación completa desde cero e integrar las diferentes partes, así como los diferentes algoritmos que participarán en todo este proceso, supone un reto a nivel profesional, es por ello que se ha decidido clasificar las diferentes motivaciones en los siguientes apartados.

1.1.1. MOTIVACIÓN PERSONAL

Como autor de este proyecto, siempre he sentido la necesidad de aportar mi grano de arena para poder mejorar la forma en la que el mundo funciona. Eso fue lo que me llevó a estudiar la carrera de Ingeniería del Software y posteriormente centrarme en este proyecto de planes. Podríamos decir que las diferentes profesiones de Ciencias de la Computación son ejes clave de cara al desarrollo futuro de la humanidad y es por ello que debemos crear herramientas que permitan mejorar la calidad de vida de las diferentes personas que habitan a nuestro planeta y tienen acceso a la tecnología.
Es por ello, que tras conocer a referentes en el mundo digital como Jaron Lanier y sus libros *"Ten Arguments for Deleting your Social Media Right Now"* [3] y *"¿Quién Controla el Futuro?"* [4], donde desglosa los diferentes inconvenientes que producen en nuestra sociedad el uso de las diferentes redes sociales y explica detalladamente cuáles eran sus modelos de negocio, vi que existía una necesidad de *nadar a contracorriente* respecto a este mundo.
Esto no es otra cosa que la necesidad de gestar una idea que logre lo contrario que logra una red social: Que nos despeguemos de nuestras pantallas y logremos ver que hay un mundo ahí fuera. Que en lugar de estar constantemente conectados a internet y desconectados de nuestro entorno, se logre una sinergia entre nuestro entorno, internet y nosotros mismos gracias a la tecnología.
Es por ello, que nace la motivación personal de una aplicación como *PlanItNow!,* tratar de lograr que se emplee la tecnología para cubrir de forma real y efectiva la necesidad de *ser social* que tenemos todos como seres humanos

1.1.2. MOTIVACIÓN PROFESIONAL

Realizar un proyecto que contemple desde la parte de ideación, gestación, validación y creación hasta el desarrollo de la idea es un gran desafío a nivel profesional.
Es por ello que el desarrollo de una aplicación como *PlanItNow!* supone un gran reto profesional. Gran parte de los esfuerzos se han centrado en que este proyecto sea en un futuro rentable, que el software desarrollado funcione correctamente y además que cumpla con los requisitos definidos en las fases de emprendimiento para poder crear un MVP.
El proceso a seguir es el mismo que el de creación de una ***startup***[4] *y,* aunque esto pudiera ubicarse fuera del marco de un Trabajo de Fin de Máster, la metodología y el proceso sí que están contemplados dentro del marco.

[4] El concepto Startup se utiliza en el mundo empresarial para referirse a empresas de reciente creación, normalmente fundadas por un emprendedor o varios, sobre una base tecnológica, innovadoras y supuestamente con una elevada capacidad de crecimiento

Por último, el tratar de emplear las tecnologías más avanzadas que están a disposición de un estudiante de Máster para lograr que la aplicación tenga la mayor calidad posible y se adapte a los requisitos supone una gran motivación de cara a desarrollar este proyecto.

1.2. OBJETIVOS

Debido a que se pretende establecer los objetivos de la forma más clara y concisa, se han empleado los objetivos **SMART** [5] para definir el marco de este proyecto. El orden de los objetivos refleja la prioridad con la que estos se pretenden lograr aunque ese pueda no ser el orden final resultante.
De cara a establecer una mejor organización de los objetivos de este proyecto, estos se han dividido en **objetivos técnicos** (aquellos que se pueden medir y evaluar de forma técnica) y **objetivos generales**, que son aquellos más valiosos puesto que deberían ser el resultado de lograr implementar exitosamente los diferentes objetivos técnicos.

1.2.1. OBJETIVOS GENERALES

- **OG1.** Reforzar la sensación de comunidad en las diferentes localidades donde se use la aplicación.
- **OG2.** Facilitar a los usuarios una herramienta para poder desconectar de las redes sociales, conectando en su lugar, con su entorno cercano (amigos, familiares o nuevos amigos).
- **OG3.** Recomendar planes y actividades a los usuarios con el fin de facilitarles participar en aquellos más afines a sus gustos.

1.2.2. OBJETIVOS TÉCNICOS

- **OT1.** Desarrollar una aplicación Android que implemente, al menos, un Producto Mínimo Viable durante el marco de tiempo de desarrollo de este proyecto.
- **OT2.** Lograr que la aplicación desarrollada se conecte e integre a un servidor externo que facilite la persistencia y gestión de datos durante el marco de tiempo de desarrollo de este proyecto.
- **OT3.** Desplegar la aplicación en un servidor externo que permita a los diferentes usuarios usarla al finalizar el desarrollo del proyecto.
- **OT4.** Publicitar la aplicación y mostrarla de tal forma que se obtengan al menos cinco *early-adopters*[5] una vez se finalice el desarrollo.
- **OT5.** Validar la idea y el modelo de negocio de cara a un proyecto de emprendimiento una vez esté terminado el primer MVP con al menos tres clientes finales
- **OT6.** Lograr implementar y pulir un algoritmo de recomendación que trabaje en función de los planes que los usuarios creen y aquellos en los que participen antes de finalizar el desarrollo. Este algoritmo debe recomendar al menos, los 10 planes más afines al usuario.

[5] Early adopters es un conjunto de usuarios (o empresas cliente) que están dispuestos a probar (o incluso comprar) una versión preliminar del producto/servicio, que aún no tiene todas las funcionalidades deseables, ni está totalmente robusto.

1.3. STAKEHOLDERS

Los **stakeholders o partes interesadas** de este proyecto son todas aquellas personas, empresas o instituciones interesadas en que se desarrolle este proyecto o que serán afectados por el desarrollo de este proyecto.

1.3.1. REALIZADOR DEL PROYECTO

Es el principal interesado y responsable de este proyecto. Se encarga del **desarrollo, documentación, gestión y análisis** de las diferentes características de proyecto con el fin de lograr que el proyecto cumpla con los objetivos definidos anteriormente.
Su interés de cara a este proyecto radica en la calificación del Trabajo de fin de Máster, así como **la oportunidad de desarrollar una idea de negocio** empleando los conocimientos del máster cursado.

1.4. IMPACTO ESPERADO

A continuación, se procederá a describir el impacto que tendrá el proyecto en un futuro en el que este resulta exitoso. Para ello, se desglosará el impacto **en usuarios, negocios y sociedad**.

1.4.1. IMPACTO EN USUARIOS

En todos los usuarios se pretende lograr un impacto de movilización, principalmente en el ámbito de la *salud y bienestar* ya que podrán disponer de una nueva forma de socializar y realizar planes con sus amigos. Esto pretende contribuir al cuidado de la salud mental y física logrando concentrar grupos de individuos con intereses comunes.
Por otro lado, se pretende también ayudar a concienciar sobre el cuidado de los entornos naturales ya que se espera que los usuarios usen estos espacios para realizar sus planes y actividades.

1.4.2. IMPACTO EN NEGOCIOS

De cara a los negocios se pretende ofrecer una mayor visibilidad a aquellos negocios que ofrezcan planes y actividades gratificantes. Esto repercutirá en unos mayores beneficios para estos negocios y una mayor capacidad de captación de clientes.

1.4.3. IMPACTO SOCIAL

A través de esta aplicación se pretende lograr un impacto social, mejorando las relaciones personales entre los individuos, facilitando a crear nuevos círculos sociales y esencialmente contribuyendo a desligar a la mayoría de personas de las diferentes redes sociales ofreciendo una nueva perspectiva desde la que disponer de la tecnología.

1.5. METODOLOGÍA

Debido a que este es un proyecto de emprendimiento bajo una envergadura de *Trabajo de fin de Máster*, se ha seguido una metodología que facilita este proceso.
Es por ello por lo que se ha decidido elaborar el proyecto a través de la metodología *Lean Startup* [6]. Esta metodología consta esencialmente de tres fases iterativas: Crear, Medir y Aprender.

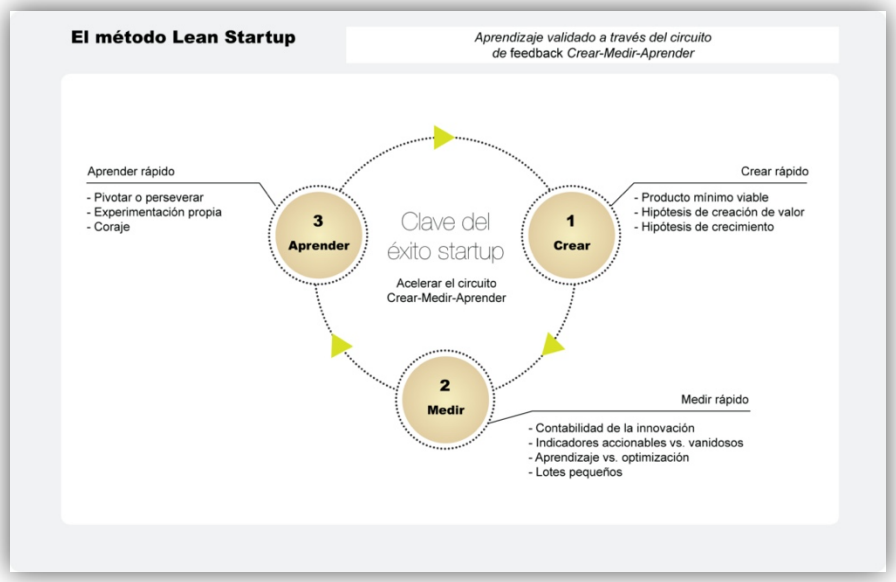


ILUSTRACIÓN 1. MÉTODO LEAN STARTUP [6]

Cabe decir que esta metodología se ha seguido de la forma más práctica posible. Es por ello que ha sido necesario modificar ligeramente este proceso iterativo para que encaje en el marco de este proyecto. Por esta misma razón se han definido a continuación de forma más detallada los pasos a seguir.

1.5.1. FASE 1. CREACIÓN

En la primera iteración de esta fase se ha gestado la idea a nivel de aplicación, así como las características iniciales. También se han elaborado los diferentes diseños de la aplicación y una primera hipótesis de creación de valor y modelo de negocio.
En las iteraciones posteriores se procederá a mejorar el producto hasta que pueda alcanzar una versión completamente comercial y con un modelo de negocio que se ajuste de la mejor forma a las necesidades de mercado.
Esta fase está contenida en los apartados posteriores referentes al desarrollo de la aplicación, así como la gestación de la idea y empleo de técnicas de evaluación.

1.5.2. FASE 2. MEDICIÓN

En la primera iteración de esta fase se ha evaluado la validez de esta idea. Se han empleado diversas técnicas de evaluación de ideas de negocio que se han ido puliendo a lo largo de las siguientes iteraciones (*DAFO, Lean Canvas…*). Durante esta fase se trata de validar la idea de la mejor forma posible.
Esta fase contiene los apartados posteriores referentes a la evaluación y validación de la idea, así como los experimentos con usuarios finales.

1.5.3. FASE 3. APRENDIZAJE

Debido a que esta fase corresponde a un aprendizaje de todo lo anterior, se ha evaluado todo el trabajo desarrollado a lo largo de un marco establecido de tiempo.
En el caso de que se haya detectado algún error, inconsistencia, mejora o que se va por buen camino, se ha empleado esta fase para poder pivotar y modificar los resultados o continuar por el camino esperado.
Esta fase está contenida esencialmente en los diferentes apartados referentes a las conclusiones o incluso a lo largo del desarrollo e implementación.

1.5.4. ACLARACIONES

Cabe destacar que esta metodología se ha seguido de la forma más precisa posible, aunque esto no pueda verse reflejado a lo largo del documento final del proyecto debido a su estructura. Sin embargo, a lo largo de los diferentes apartados mencionados anteriormente se aclara en qué fase e iteración se tomó cada decisión y/o desarrolló cada característica del resultado final. Para ello se ha asumido que cada iteración de esta metodología corresponde a una versión del producto final.

1.6. ESTRUCTURA

Debido a que este documento cuenta con un gran número de páginas, se ha decidido dividir por secciones, subsecciones y capítulos.

Las diferentes secciones y subsecciones de este documento, podemos verlas en el **índice** con un nombre que describe los diferentes aspectos a tratar.

Por otro lado, el documento cuenta con los siguientes capítulos:

- **Contexto.** En este capítulo se menciona el contenido general del proyecto, así como todo el contexto y convenciones necesarias para poder entender el resto del documento.
- **Evaluación de la Idea de Negocio.** En este capítulo se evalúa la idea de negocio relacionada con este proyecto a lo largo de sus diferentes secciones.
- **Desarrollo de la Idea de Negocio.** En este capítulo se muestra cómo se ha desarrollado la idea de negocio, así como los diferentes aspectos y decisiones importantes que se han tomado durante el mismo, así como los resultados de los diferentes MVP.
- **Aspectos Técnicos.** Este capítulo se centra en mostrar los diferentes aspectos técnicos del proyecto referentes al mundo de la ***Ingeniería del Software***, tales como herramientas, tecnologías y diseño empleados para dar lugar a los resultados de los diferentes MVP.
- **Conclusiones.** En este capítulo se realiza una síntesis de todos los aspectos más relevantes logrados durante el proyecto a nivel de emprendimiento, objetivos y experiencia adquirida.

1.7. CONVENCIONES

En este apartado se enumeran las diferentes convenciones empleadas a la hora de redactar este documento con el fin de facilitar su legibilidad:

- Se han marcado en **negrita** aquellas palabras o frases que el autor del documento considere más relevantes.
- Se han marcado en *cursiva* los extranjerismos, tecnicismos, citas y palabras pertenecientes a la jerga tecnológica de este proyecto.
- Se han entrecomillado las citas textuales externas a la obra.
- La primera aparición de palabras clave se ha definido en la parte inferior de la misma página donde aparezcan para facilitar su entendimiento, sin embargo, también pueden ser encontradas en el glosario. Se aplicarán las letras en cursiva para conceptos que requieran ser documentados con bibliografía.
- La bibliografía se realizará en el formato IEEE. Esto significa que, al lado de cada entrada bibliográfica o concepto documentado, aparece un número en corchetes del que se podrá encontrar su correspondencia al final de este documento, en la sección *Bibliografía.*

2. GENERACIÓN DE LA IDEA DE NEGOCIO

Como se ha mencionado a lo largo de la introducción, esta idea de negocio nace a través de diversos proyectos realizado en las diferentes asignaturas del máster universitario *Tecnologías y Sistemas de Software,* ofertado por la Universidad Politécnica de Valencia. En este apartado se procede a documentar cómo fue la generación de esta idea, así como las diferentes fases por las que este proyecto pasó antes de ser formalizado como idea de emprendimiento y *Trabajo de Fin de Máster.*

2.1. GESTACIÓN DE LA IDEA DE NEGOCIO

2.1.1. DATA SCIENCE

*"En **PlanItNow!** se pretende desarrollar una aplicación móvil que facilite a los diversos usuarios del mundo el hecho de realizar planes y actividades afines a ellos mismos. Para poder ofrecer esto, la aplicación proveerá a los usuarios con planes y actividades afines a sus gustos. Además, y únicamente si el usuario lo desea, tendrá la oportunidad de apuntarse a actividades creadas por otras personas con el fin de poder ampliar su círculo social y conocer a personas con sus mismos gustos e inquietudes."* [7]

Estas fueron las primeras palabras redactadas sobre el alcance de un proyecto nacido en el mes de Noviembre del año 2020, durante la asignatura *Data Science*. A través de un proyecto *freelance* de la asignatura se podían gestar las ideas propias de los estudiantes. Inicialmente, la aplicación comenzó como una idea que, probablemente nunca llegaría a implementarse y que se enfocaba en algoritmos de recomendación y gestión de datos de usuarios con el fin de facilitar la creación y gestión de planes entre los mismos.

2.1.2. EMPRENDIMIENTO EN PRODUCTOS SOFTWARE

Meses después, en Febrero de 2021, durante la asignatura de *Emprendimiento en Productos Software* y con un equipo de tres personas en lugar de dos, se comenzó a dar forma a lo que sería una idea real de negocio. A través de la metodología empleada en esta asignatura, se fue iterando sobre la idea, la posible captación de clientes y el modelo de negocio, así como las fases que se debían seguir para materializar un producto que correspondiese a esta aplicación.

En esta fase, se realizó un trazo más preciso de la idea a nivel de negocio, se previó un análisis financiero y se comprobó que efectivamente una idea de esta envergadura podría encajar en un mercado real. Sin embargo, no se había desarrollado aún ningún artefacto de software.

2.1.3. TRABAJO DE FIN DE MÁSTER

Hacia finales de Mayo de 2021, una vez terminada la asignatura de *Emprendimiento en Productos Software* y tras haber adquirido los conocimientos suficientes como para desarrollar una aplicación de esta envergadura, se lanza la propuesta como proyecto de fin de máster en las plataformas de la Universidad Politécnica de Valencia para que finalmente sea aceptado como *"Aplicación Inteligente para Gestión y Desarrollo de Planes"*, produciendo una sinergia entre el emprendimiento del que se habla en los apartados anteriores y el proyecto de fin de máster a realizar.

2.2. CONCURSO IDEAS UPV 2K21

Debido a que este proyecto trata de ser de emprendimiento y fue gestado durante diferentes etapas, el proyecto se formalizó bajo la documentación que se solicitaba en el concurso del departamento de ***IDEAS UPV,*** llamado *"Concurso Emprendedor Universitario 2K21"* [8] en el que el proyecto fue presentado en las modalidades *Proyecto Empresarial y Trabajo de Fin de Máster*, logrando el puesto finalista en ambas.
Esto ayudó tanto al autor de este documento como al equipo que participó en la modalidad de proyecto empresarial a pulir diferentes aspectos del proyecto como el público objetivo y la oportunidad de restablecer este proyecto como un proyecto de triple impacto.

EVALUACIÓN DE LA IDEA DE NEGOCIO

3. ESTUDIO Y ANÁLISIS DE MERCADO

En este apartado se desglosa el estudio y análisis de un mercado de aplicaciones que trabajan en ofrecer planes y experiencias a los diferentes usuarios. Es decir, se ha realizado una investigación principalmente en anchura sobre aplicaciones que están relacionadas o que puedan ser complementadas con *PlanItNow!*, así como los potenciales clientes de la aplicación que se ha desarrollado y su competencia directa.

3.1. POTENCIALES CLIENTES

Tal y como se ha establecido el diseño de la aplicación, podemos encontrar en el mercado diferentes perfiles de clientes. Para **simplificar** este documento, en este apartado **se tratarán los diferentes *usuarios* de la aplicación como potenciales clientes**, aunque en apartados posteriores se aclarará qué tipo de usuario generará beneficios y qué tipo de usuario únicamente usará la aplicación.
A continuación, se ha realizado una clasificación de los diferentes potenciales clientes de esta aplicación.

3.1.1. USUARIO PROMEDIO

Es **el usuario promedio de la aplicación**. Aquel que **participa y crea los planes** de la aplicación.
De acuerdo con la web ***Statista*** [9] *,* únicamente en la red social *Instagram,* hay 20 millones de cuentas creadas sólo en España. De esto, podemos deducir que inicialmente, nuestros potenciales usuarios son aquellas mismas personas que están usando ya otras aplicaciones de redes sociales y a su vez tienen disposición de obtener nuevos planes.
Es por ello que nuestro potencial usuario promedio serán personas de entre 18 y 45 años, ya que este es el grupo poblacional que representa la mayor parte de los usuarios activos en redes sociales [10].
Debido a que este proyecto pretende inicialmente ser lanzado en España y en un caso exitoso más adelante, ser lanzado en otras partes del mundo, se ha calculado el número de potenciales clientes gracias a la web ***Instituto Nacional de Estadística*** [11].
De este modo, se suman un total de **16.365.766 de potenciales usuarios** únicamente en España, siendo el grupo con mayor población el de 40 a 44 años.

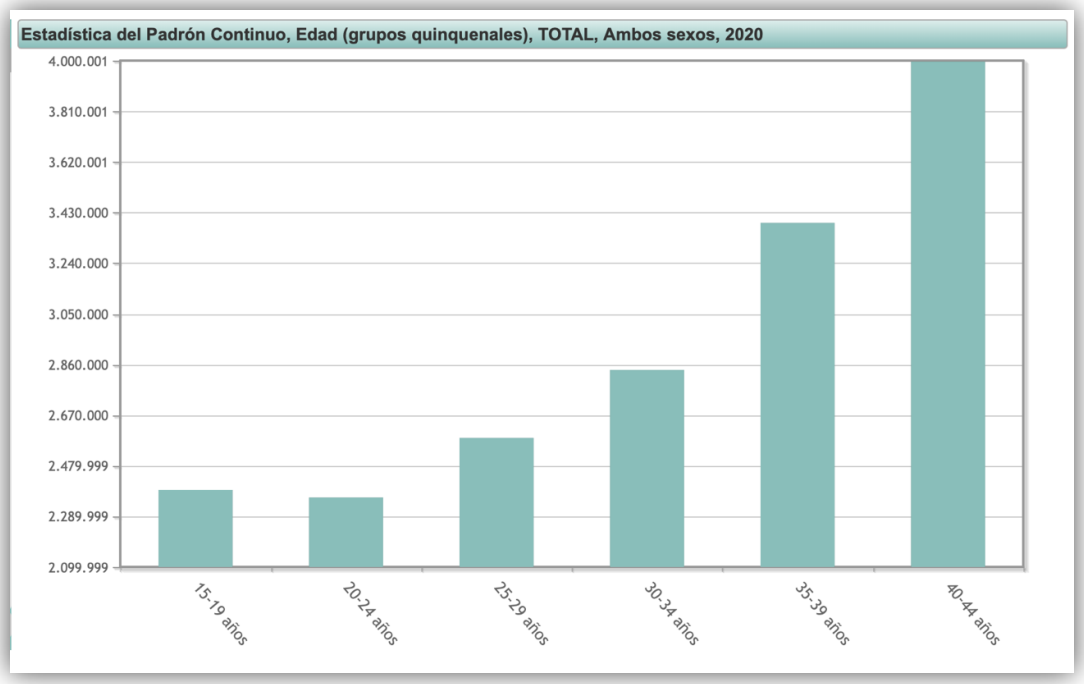


ILUSTRACIÓN 2. GRÁFICO DE POBLACIÓN POR EDAD EN ESPAÑA ENTRE 15 Y 45 AÑOS [11]

De este gráfico podemos deducir que, pese a que probablemente los usuarios entre los 20 y 39 años sean aquellos que tengan mayor tasa de adquisición, se debe considerar realizar estrategias y campañas de marketing enfocadas en el público mayor de 40 años puesto que suponen un grupo muy representativo a tener en cuenta.
Por último, hay que aclarar que el grupo de 18 y 19 años quedaba excluido como rango único de edad, de modo que se ha realizado el promedio del grupo comprendido entre los 15 y 19 años para extraer el valor de los potenciales usuarios.

3.1.2. NEGOCIOS

Los negocios corresponden a empresas, individuos o entidades que podrían obtener beneficios gracias al uso de la aplicación. Estos beneficios se generarían gracias a la oportunidad de publicar y promocionar sus diferentes planes, logrando que los diferentes *usuarios* de la aplicación puedan descubrir y asistir estos planes
Estos negocios podrían publicar planes, actividades y eventos de tal forma que la única responsabilidad del usuario sea apuntarse y decidir si va o no. También se les facilitará gestionar de forma eficiente estas actividades.
De cara a realizar un análisis de mercado de los potenciales negocios que podrían participar en el flujo de la aplicación, se tendrán en cuenta los siguientes:

- Restaurantes, bares
- Centros/salas de ocio
- Salones de videojuegos
- Cines y teatros
- Agencias de turismo
- Negocios de turismo
- Gimnasios
- Promotores de conciertos
- Centros deportivos

- Centro de eventos
- Otros

Se ha realizado una estimación total del número de negocios de esta envergadura en España y se ha concluido que existen **aproximadamente** 1.000.000 de negocios de estas características. Esto ha sido contrastado mediante la web del instituto nacional de estadística.
A continuación, se muestra una gráfica con algunos de los negocios que se tendrían en cuenta:

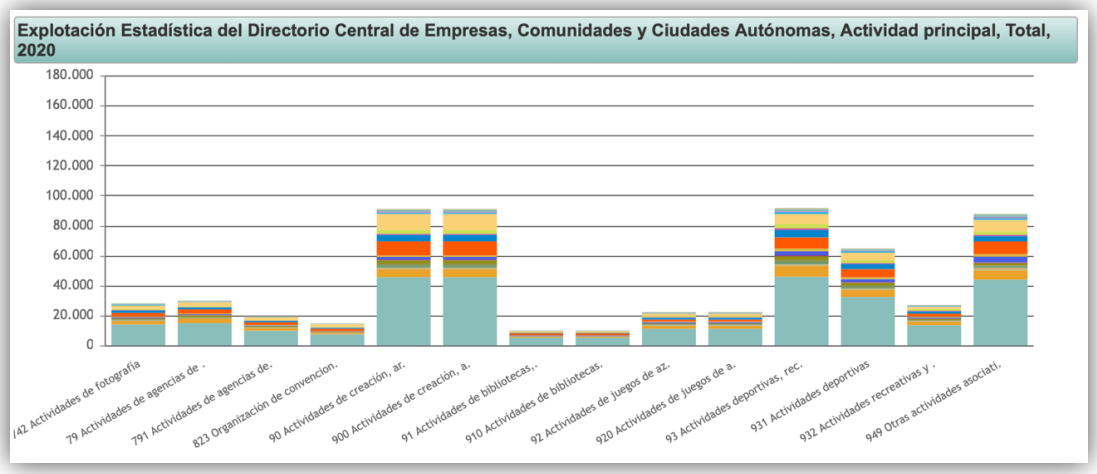


ILUSTRACIÓN 3. ESTADÍSTICAS DE TIPOS DE NEGOCIO EN ESPAÑA [11]

3.2. POTENCIALES COMPETIDORES

PlanItNow! cuenta con múltiples competidores. Esto se debe a que ya existen diversas aplicaciones que permiten crear eventos. Para poder mostrar el entorno de competencia de la forma más visual posible, se han enumerado las principales aplicaciones que puedan tener relación directa o indirecta con la filosofía de este proyecto.
De cara a establecer esta tabla competitiva, se han tenido en cuenta las diferentes características tanto técnicas como no técnicas ya que lo que se pretende es el generar una idea totalmente innovadora para posteriormente establecer el modelo de negocio.

TABLA 1. ANÁLISIS DE COMPETIDORES.

	Conecta Gente Nueva	Ofrece Actividades	Recomienda Actividades	Permite Crear Actividades Personalizadas	Tiene en Cuenta la Localización	Tiene ya establecida una Amplia red de contactos	Uso completo de la aplicación gratuito	Proporciona un servicio de mensajería
Mitmi	~	✓	✗	✓	✓	✗	✓	✗
Qhaceshoy?	✗	✓	✗	✗	✓	✗	✗	✗
FEVER	✗	✓	~	✗	✓	✓	✗	✗
BlaBlaCar	✓	~	✗	~	✓	✓	~	✓
Facebook	✓	✓	✗	✗	~	✓	✓	✓
Google Calendar	✗	✗	✗	✓	✓	✓	✓	✓
PlanItNow!	✓	✓	✓	✓	✓	✗	✓	✓
MeetUp	✓	✓	✓	✓	✓	✓	✗	✓
Skout	✓	✗	✗	✗	✓	✓	✓	✓
Happn	✓	✗	✗	✗	✓	✓	✓	✓
Timpik	✓	✓	✓	✗	✓	✓	✓	✓

Como se puede ver en la tabla, se aprecia que existen diferentes aplicaciones que son potenciales competidores del proyecto que se ha desarrollado.
Debido a que este proyecto consta de una aplicación que nace desde cero, esto permite enfocarse en las características en las que se destaca respecto a la competencia, ofreciendo innovación al mercado.
A continuación, se describe brevemente a los competidores, así como los puntos fuertes que estos puedan tener respecto al proyecto que se ha desarrollado.

3.2.1. MITMI

Mitmi [12] es una aplicación que hoy en día no está en el mercado, sin embargo, casualmente es aquella que más similaridades tiene a lo que se pretende desarrollar.
Como se puede ver en la *Ilustración 4,* existen diferentes similitudes a nivel de concepto, pero esta aplicación no es otra cosa que lo que sería un primer prototipo de este proyecto, ya que permite a los usuarios obtener la localización y horario de un plan.

ILUSTRACIÓN 4. INTERFAZ DE MITMI [12]

Sin embargo, **no cuenta con un algoritmo de recomendación**, especialmente de cara a los planes públicos, sino que únicamente permite realizar filtros de búsqueda. Se deduce que, al hacer que el usuario trabaje tanto de cara a *cazar* una oferta, finalmente cese su uso de la aplicación.

3.2.2. QHACESHOY?

Qhaceshoy? [13] selecciona las mejores ofertas de espectáculos en directo en la zona que se busque. Su banco de datos abarca las principales ciudades Valencia, Barcelona y Madrid. Su característica principal es la de permitir ver precios y reservar en establecimientos de ocio.

Hoy en día **esta aplicación ha sido descontinuada**, ya que **su única función era la de vender entradas a último minuto**.

3.2.3. FEVER

Fever [14] muestra ofertas de ocio en tu ciudad o comunidad local. Al igual que **Qhaceshoy?** nos permite comprar desde la misma app. En este caso el banco de datos abarca Madrid, Barcelona, Valencia, Sevilla, Málaga, Bilbao, Ibiza y muchas otras ciudades de Europa y de los EE. UU.

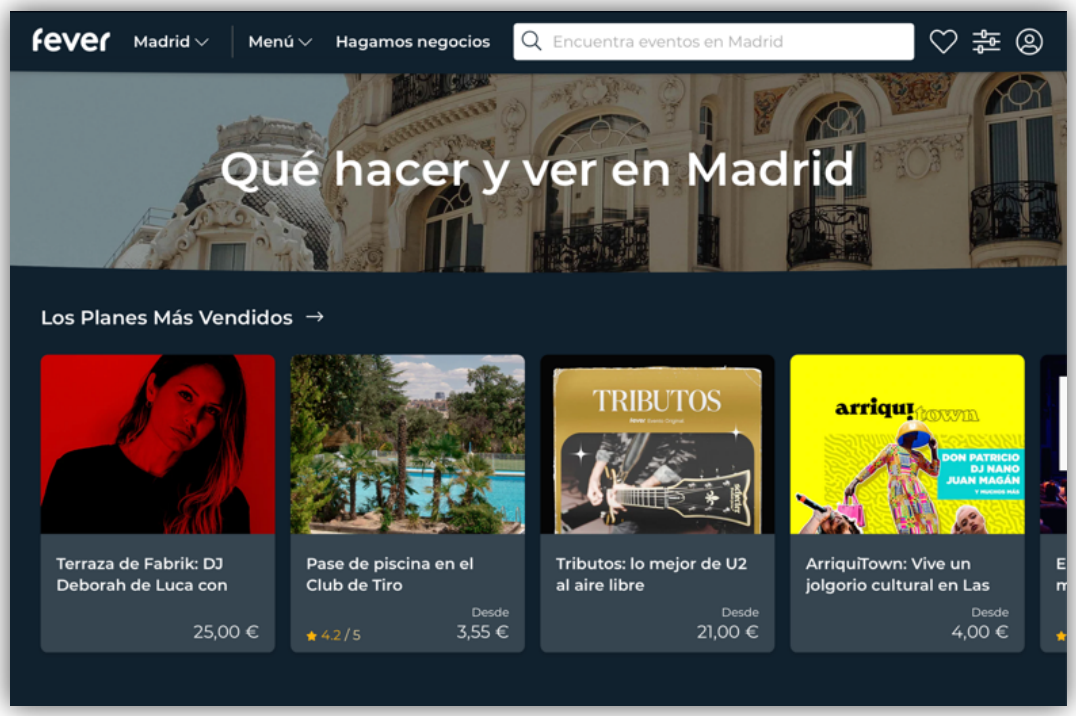


ILUSTRACIÓN 5.INTERFAZ WEB DE FEVER [14]

Algunas de sus características son:

- Permite ver precios y reservar en establecimientos de ocio.
- Muestra planes en base a los gustos de los usuarios.

Pese a que esta aplicación se aproxima bastante a lo que se pretende hacer en PlanItNow!, únicamente gestiona planes de pago, lo que reduce enormemente el flujo de usuarios. Además, no permite conocer gente nueva a través de la aplicación ni *apuntarse* a los diferentes eventos.

3.2.4. BLABLACAR

BlaBlaCar [15] ayuda a los usuarios a desplazarse de un lugar a otro en coche, abaratando los costes de transporte. Su principal característica es la de unir gente que tienen un mismo destino e intereses.

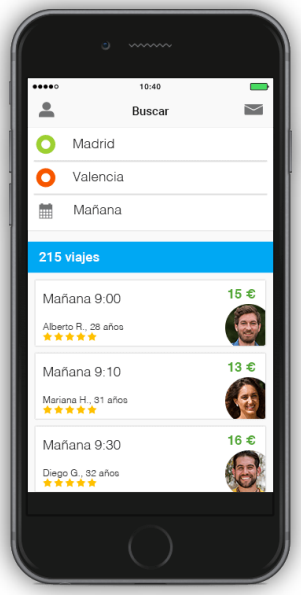


ILUSTRACIÓN 6. APLICACIÓN BLABLACAR PROGRAMANDO UN VIAJE MADRID-VALENCIA [15]

Pese a que la filosofía de *BlaBlaCar* pueda ser muy similar a la de PlanItNow!, únicamente engloba los viajes con destino común. Sin embargo, son un ejemplo a seguir de cara a establecer las distancias entre los diferentes planes y actividades e incluso llegar a ser una potencial integración.

3.2.5. FACEBOOK (EVENTS)

Actualmente Facebook [16] es una gran empresa tecnológica que cuenta con miles de millones de usuarios, es por ello que es uno de nuestros principales competidores, ya que cuenta con una gran base de usuarios y la posibilidad de crear casi cualquier tipo de sub-aplicación.
La comparación con su sistema de eventos es especialmente interesante ya que, pese a que cuentan con un sistema de eventos muy completo que permite la participación a miles de usuarios e informarse de los diferentes eventos.
Este sistema nos permite diferenciar los usuarios *interesados y* los *asistentes,* sin embargo, no es una herramienta muy usada entre los diferentes usuarios de Facebook y carece de un sistema de recomendación más allá de los *destacados*, así como las dificultades de filtrados por distancia y otros factores relevantes para una aplicación como PlanItNow!.
Sin embargo, sí que son un ejemplo a seguir de cara al sistema de organización de planes y eventos y una posible integración futura de que los planes que se creen en la aplicación, se creen como eventos en Facebook para lograr un mayor número de asistentes.

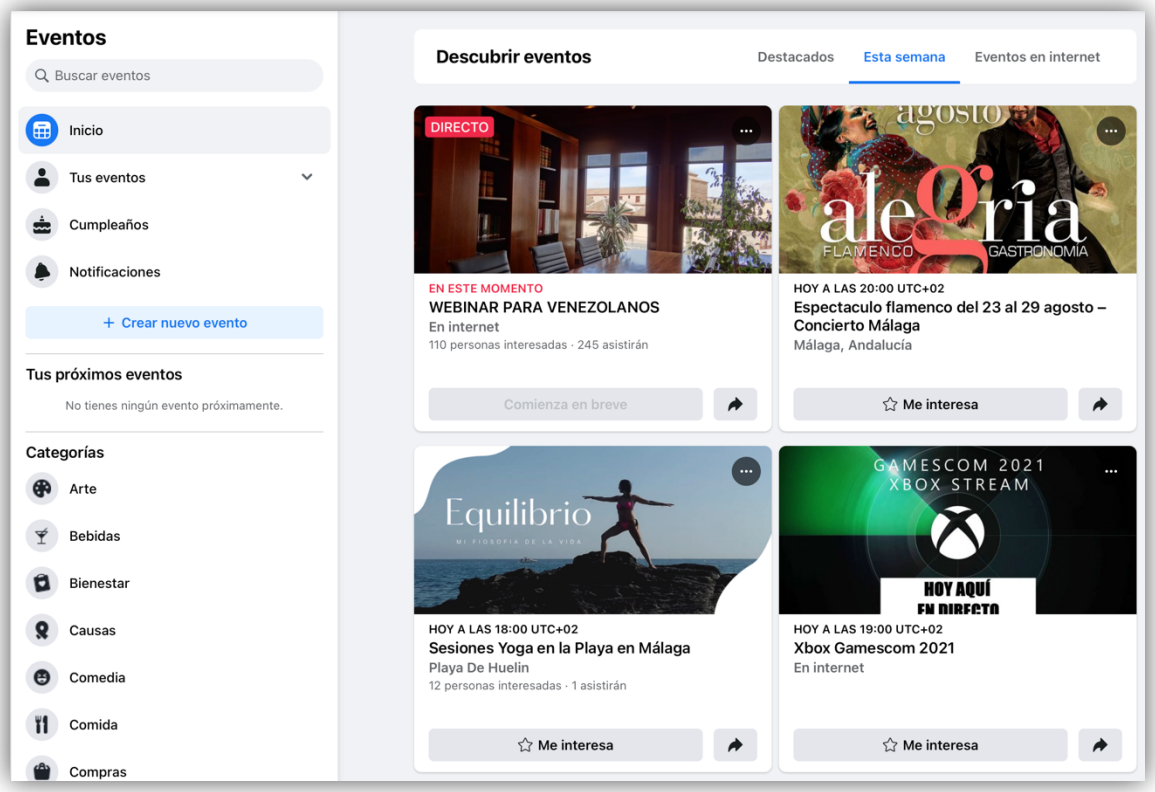


ILUSTRACIÓN 7. EVENTOS DE FACEBOOK [16]

Otro de los puntos fuertes de Facebook es su gran número de usuarios, es por ello que gracias a esta plataforma podrían alcanzarse potenciales clientes y usuarios.
Como resumen, podemos decir que la mayor ventaja competitiva de *PlanItNow!* respecto Facebook es el algoritmo de recomendación junto con la posibilidad de conocer nuevas personas y así crear una comunidad más orientada a planificar sus vidas que a compartirlas a través de una red social.

3.2.6. GOOGLE CALENDAR

Pese a que Google es una empresa que actualmente no gestiona redes sociales, sí que cuenta con una herramienta que podría servir de referencia y competencia a lo que se desea hacer. Esta herramienta es *Google Calendar* [17], una herramienta que permite organizar actividades y reuniones entre diferentes usuarios ofreciéndoles una video llamada o incluso pudiendo establecer lugar y ubicación.
Sus puntos fuertes con la capacidad de establecer y clasificar los diferentes planes y actividades además de buscar entre ellos, sin embargo, al no ser una plataforma orientada a reunirse con diferentes personas no supone una gran amenaza para *PlanItNow!*. Sin embargo, sí que sería ideal en un futuro poder integrar los planes y actividades de esta aplicación con un calendario de esta envergadura para facilitar una mayor gestión, además de inspirarnos en su formulario de creación de entrada en el calendario para facilitar el crear los planes de este proyecto.

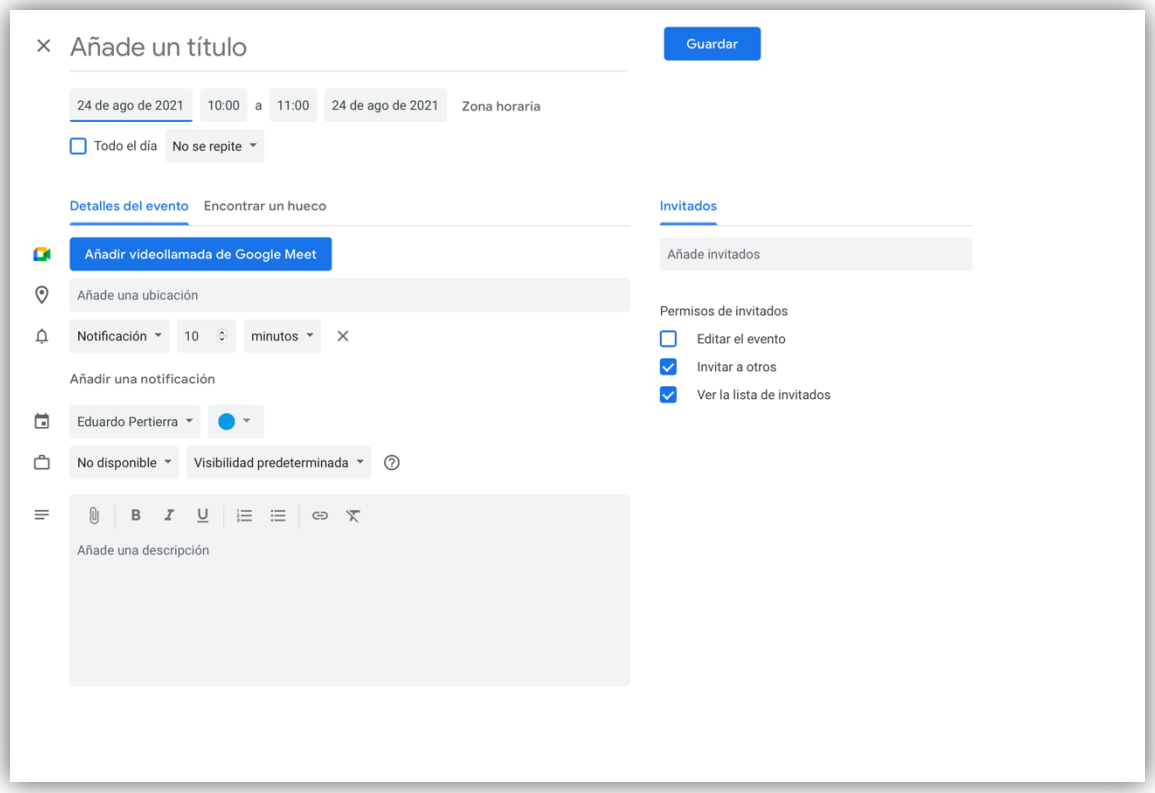


ILUSTRACIÓN 8. CREACIÓN DE UN EVENTO EN GOOGLE CALENDAR [17]

3.2.7. MEETUP

MeetUp [18] es una plataforma con más de 3 millones de usuarios activos que permite a las diferentes personas encontrar a otras que disfruten las mismas actividades.
Su modelo de actividades se basa en la creación y gestión de grupos de cara a que los usuarios participen en estos. Esta es una debilidad respecto a *PlanItNow!*, ya que lo que se pretende es que descubras planes a los que poder ir con tus amigos o con gente nueva, incluso planes en los que únicamente puedas participar con tus amigos.

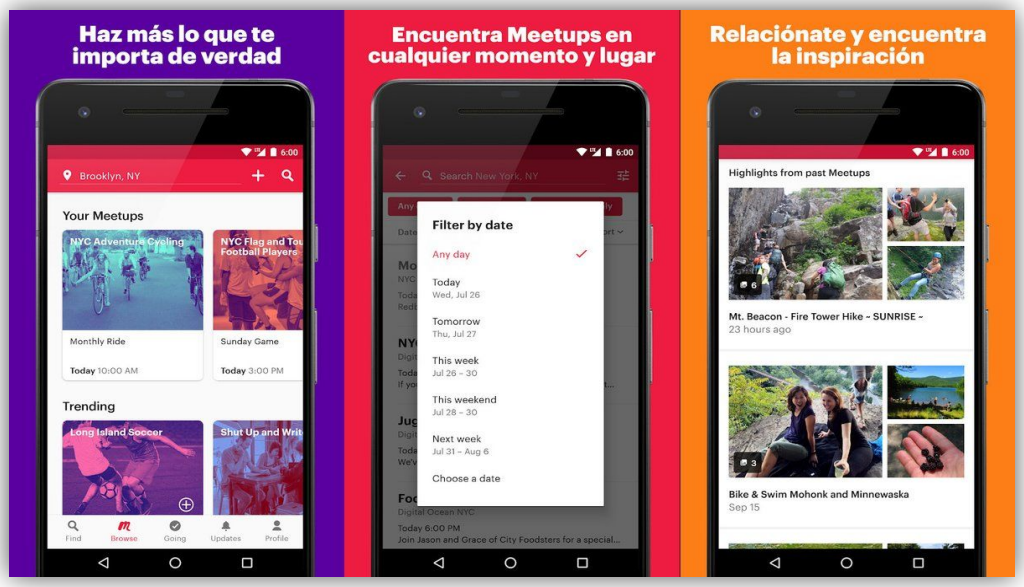


ILUSTRACIÓN 9. INTERFAZ MÓVIL DE MEETUP [18]

Además, el modelo de negocio de *MeetUp* se basa en cobrar a los organizadores de planes y grupos, resultando en una menor oferta de planes. Esto difiere radicalmente de la filosofía de este proyecto, donde se pretende que el uso de la aplicación sea totalmente gratuito.

3.2.8. SKOUT

Skout [19] es una aplicación que permite a los usuarios conectar con usuarios a lo largo del mundo, ya sea personas que están cerca o lejos.
Pese a que esta filosofía calza parcialmente con lo que se desea lograr en este proyecto, para *PlanItNow!* es necesario que se conecten principalmente usuarios que compartan entorno e intereses, es decir, aquellas personas que estén cerca puesto que lo que se desea es fomentar la sensación de comunidad.
Sin embargo, una aplicación como *Skout* podría ser un gran referente en el caso de que se desease en un futuro implementar un sistema para realizar nuevas amistades con las que ir a planes dentro de la app. También nos orienta mucho de cara a implementar un sistema de perfiles privados/públicos de usuarios.

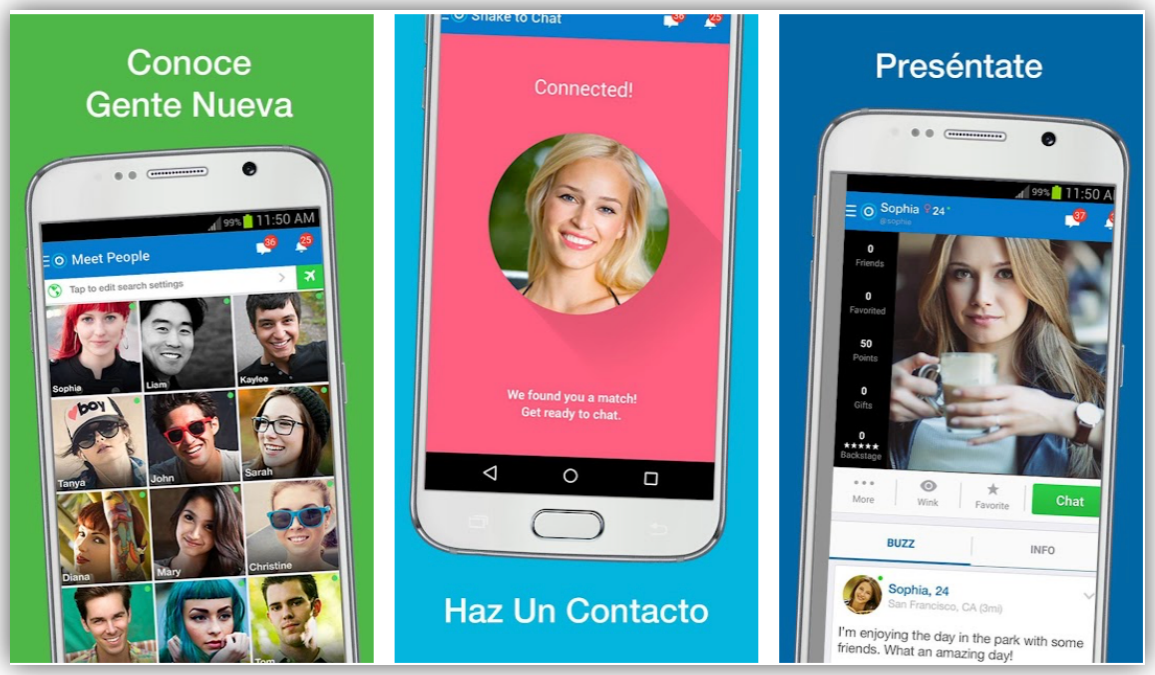


ILUSTRACIÓN 10. IMÁGENES DE LA INTERFAZ DE SKOUT [19]

3.2.9. HAPPN

"Conocer a la persona que te gusta ocurre a menudo en el momento en el que menos te lo esperas. Happn es la aplicación que te conecta con todas las personas con las que te cruzas a diario, aquellas que forman parte de tu día a día."

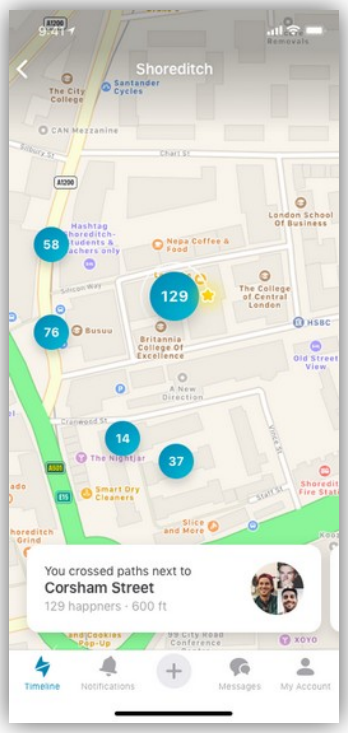


ILUSTRACIÓN 11. MAPA DE HAPPN [20]

Este enérgico mensaje es el mensaje promocional de *Happn* [20]. Pese a que sí que es ideal la forma que tiene de mostrarte las personas con las que te vas cruzando, en el caso de *PlanItNow!,* se desea priorizar los planes y actividades a realizar con estas personas.

Es por ello que aunque son aplicaciones distantes, pueden hacerse competencia entre sí en el ámbito de conocer personas, ya que se extrema la cercanía.

Como plan de futuro, podría implementarse una idea como la de *Happn* de que te vaya notificando qué planes te has cruzado o cuáles tienes extremadamente cerca.

Por último, destacar que, pese a que *Happn* es probablemente una aplicación para encontrar pareja, *PlanItNow!* no pretende ser de esta categoría, pese a que pueda haber ciertos aspectos en los que sí exista una competencia más directa.

3.2.10. TIMPIK

Timpik [21] es una aplicación orientada a encontrar jugadores casuales para diferentes actividades deportivas. Su filosofía es la más parecida a *PlanItNow!* de todas las diferentes aplicaciones que se han enunciado a lo largo de este documento.
Su modelo de negocio se basa en ayudar a las diferentes entidades deportivas a gestionar los diferentes eventos deportivos que puedan existir, de tal modo que los usuarios puedan apuntarse a estas actividades.
En el ámbito de los deportes, sin lugar a duda *Timpik* es la mayor competencia de este proyecto, es por ello que se tomará como referencia en este aspecto.
Afortunadamente *PlanItNow!* cuenta con infinidad de planes y eventos diferentes, pudiendo contener también los planes deportivos, y con ayuda del algoritmo de recomendación se podría neutralizar esta competencia.

ILUSTRACIÓN 12. APLICACIÓN TIMPIK EN IOS [21]

3.2.11. CONCLUSIÓN

Como se ha podido ver en el análisis de mercado, existen multitud de aplicaciones que contienen características similares, especialmente *MeetUp*, *Mitmi* y *Facebook Events*, sin embargo, *PlanItNow!* trata de ofrecer aspectos innovadores respecto a estas mismas así como un nuevo concepto y filosofía que pretende reflejarse en el diseño de la aplicación.
Es por ello que se considera que PlanItNow! tiene cabida en un mercado tan saturado de aplicaciones, gracias a los puntos fuertes que se han identificado a lo largo de este estudio:

- Prioridad de **planes** sobre otros factores. Es decir, se priorizará el poder realizar actividades con las personas.
- Sistema **social** que permitirá a los diferentes usuarios establecer sus amigos de toda la vida y hacer nuevos amigos a través de esta aplicación.
- Sistema de **recomendación** que resultará en que los diferentes usuarios no tengan que invertir su tiempo a la hora de encontrar planes gratificantes.
- Sistema de **cercanía** que permitirá a los usuarios reforzar su entorno y sensación de comunidad.
- Aplicación **totalmente gratuita**, lo cual permitirá tener un gran flujo de usuarios respecto a otras aplicaciones que únicamente te permiten usar las funcionalidades básicas de forma gratuita.

4. ANÁLISIS DAFO

De cara a validar la idea de negocio de este proyecto, se ha realizado un análisis ***DAFO*** [22]. Para realizar este análisis, se han tenido en cuenta el ámbito competitivo elaborado anteriormente además del esbozado de características que contendrá la aplicación en un caso ideal.
A continuación, se muestra este análisis en formato de tabla:

TABLA 2. ANÁLISIS DAFO. ELABORACIÓN PROPIA

DEBILIDADES	**AMENAZAS**
• Se **depende** plenamente de **internet**. • **No se tiene experiencia** desarrollando este tipo de aplicaciones. • Todo el **contenido de la aplicación es generado por los usuarios.** • No existe **presencia física** de la compañía o aplicación. • Alta **dependencia** de que los **negocios** quieran publicitarse. • Aplicación **fácilmente clonable.** • **Equipo muy pequeño** • Requiere un **gran número de usuarios activos para atraer clientes**. • Requiere concentrar a los usuarios en las mismas **localizaciones** para que la aplicación funcione correctamente.	• **Cambio en la legislación** de protección de datos. • Que con la crisis sanitaria este tipo de aplicaciones **no tengan cabida** en el mercado. • Reviews y experiencias negativas que creen una **mala reputación** a PlanItNow. • Existen otras plataformas como **Facebook Events y MeetUp** que permiten realizar funcionalidades muy similares. • Que no haya una cantidad suficiente de usuarios de pago como para financiar el modelo de negocio. • Que los negocios **no consideren el sistema de publicidad** como una manera de promocionarse. • La aplicación **no resulta atractiva a los potenciales clientes**.

FORTALEZAS	OPORTUNIDADES
• Única aplicación en el mercado con un **sistema de recomendación** de planes. • Sistema de **ubicación** que permite **geolocalizar** el uso de la aplicación de cara a crear eventos. • **Cualquier persona es potencial usuario de la app.** • Beneficios **proporcionales al número de clientes y usuarios.** • Se tiene la capacidad de ofrecer **publicidad** a **todo tipo de diferentes negocios e instituciones**. • **Coste de mantenimiento del servicio proporcional** al **número de usuarios.** • Los propios **datos recopilados de los usuarios** sirven para mejorar el sistema de **recomendaciones**. • Proyecto cuyo **mayor coste es capital humano.** • **Sistema que fomenta usarlo entre amigos**, creando mayor sensación de unidad. • **Producto software** que puede ser utilizado en cualquier lugar. • **Completamente gratuito** para los usuarios promedio facilitando la captación de clientes. • **Altamente escalable.**	• Plena oportunidad de **negocio y promoción tras resolver la crisis sanitaria**. • A nivel conceptual, una idea así podría **lanzarse a nivel mundial.** • La aplicación puede **darse a conocer en eventos** y otros lugares siendo de las pocas aplicaciones con esta funcionalidad. • Es de las **pocas aplicaciones especializadas en eventos** y planes en el mercado actual. • El **boca a boca** podría hacer crecer la aplicación exponencialmente. • Se puede ampliar la aplicación con **nuevas funcionalidades para abarcar más mercado.** • De cara a ofrecer visibilidad, **aliarse con productoras de eventos** para conseguir un negocio win-win. • **Integración con plataformas ya existentes** para abarcar mayor mercado. • **Venta de datos** a empresas de marketing. • **Oportunidad de viralizarse** al ser creada esta aplicación como Trabajo de fin de Máster (comunidad Universitaria).

Tal y como se puede ver en este *análisis DAFO*, se puede apreciar que las mayores debilidades son referentes a la inexperiencia y captación de clientes, mientras que las fortalezas están principalmente relacionadas con las proporciones de usuarios e ingresos obtenidos, además de la posibilidad de generar comunidades en cualquier parte del mundo.
Por otro lado, vemos que las mayores amenazas es que no se logren captar usuarios y clientes o que una compañía mayor, clone la idea. Sin embargo, se tienen grandes oportunidades debido a que existen múltiples formas de dar a conocer la aplicación y de generar mayores ingresos.
Como conclusión de este análisis podemos ver que este proyecto es altamente escalable y pese a que cuenta con diversas debilidades y amenazas, en el caso de que resultase exitoso, podría resultar en un proyecto altamente rentable.

5. MODELO DE NEGOCIO

Tal y como se ha mencionado en los apartados 3 y 4, se pretende desarrollar una aplicación que resulte completamente gratuita para el usuario final, sin embargo, de cara a un proyecto de emprendimiento es fundamental mantener una fuente de ingresos e idealmente, beneficios para poder sostener toda la infraestructura y personal que colabora en este proyecto.
Es por ello se ha definido modelo de negocio gratuito para los diferentes usuarios finales pero que sí tendrá determinados costos para aquellas empresas y negocios de los mencionados en el apartado **3.1.2.**
A modo de resumen, se pretende establecer un sistema **de suscripciones Premium** para los diferentes **negocios** que participen en el sistema.
Sin embargo, y debido a que existen diversas **debilidades a la hora de localizar a los diferentes usuarios** se ofrecen también **funcionalidades extras de pago a los usuarios** con el fin de obtener una mayor cantidad de ingresos de los usuarios que deseen emplear estas funcionalidades.

5.1. SUSCRIPCIONES A NEGOCIOS

El eje principal del modelo de negocio de este proyecto se basa en ofertar una **suscripción mensual** a los diferentes negocios que deseen participar en la aplicación.
A cambio de esta suscripción, las diferentes entidades pueden darse de alta en la aplicación como ***Empresa o Negocio***, y así registrar su marca en la aplicación. Esta suscripción les permite el uso de funcionalidades extras como el **poder promocionar sus planes**, **estadísticas de visitas**, **uso de sus actividades** y otros factores que contribuyen de forma directa al **crecimiento de los negocios suscritos**.
Además, a través de esta suscripción se pretende enviar mensualmente una guía de usuario con recomendaciones de cara a la creación de planes para captar un mayor número de usuarios, las tendencias de planes durante el último período, así como el priorizar sus planes en el algoritmo de recomendación respecto a otros planes públicos.
Debido a que el proyecto se encuentra ahora mismo en fase de desarrollo, se ha establecido un precio de suscripción inicial de **10 euros mensuales,** siendo gratuito para aquellos negocios que deseen ser ***early-adopters.***
Sin embargo, este precio podría incrementar en un futuro para ajustarse a la demanda de los servicios ofertados.

5.2. FUNCIONALIDADES EXTRAS A USUARIOS

La filosofía de este proyecto es la de que todos los usuarios puedan usar la aplicación completamente de forma gratuita, de modo que es **necesario definir claramente cuáles son estas funcionalidades extras.**
Supongamos que somos un usuario promedio que crea planes con sus amigos, participa en planes públicos de *voleibol en la playa* y *va al gimnasio* con miembros de la aplicación pero que **eventualmente desea crear un plan público que obtenga la misma promoción que aquellos planes de los negocios suscritos**, o **quiere encontrar planes fuera de su localidad.**

- El usuario puede **pagar una pequeña cantidad** para **promocionar su plan o evento dentro de los diferentes algoritmos** de la aplicación, habilitándole también las

funcionalidades pertenecientes a los negocios. De este modo, **un usuario común puede promocionar un plan** en el que desee asistencia por parte de los usuarios de la aplicación.

- Tengamos en cuenta a los diferentes usuarios que quieren planear un viaje. Debido al diseño y filosofía de la aplicación, **únicamente se muestran planes cercanos al entorno del usuario**, sin embargo, **si el usuario desea poder cambiar su ubicación para conocer qué planes y actividades hay en otros lugares** y reservar su plaza, puede lograrlo como funcionalidad extra pagando una pequeña **cuota o suscripción**.

Debido a que la variación de precios puede resultar confusa, se cobrará inicialmente un precio fijo de **2 euros por cada acción**, ya sea por **buscar planes en otra localización o promocionar un plan.**
Sin embargo, debido a que se ha tenido en cuenta el sector de usuarios que **desean viajar**, se ofrece también una suscripción de **10 euros mensuales** aquellos usuarios que deseen estar constantemente buscando los mejores planes de diferentes localidades.
Cabe decir que, de nuevo, y debido a que este proyecto está en desarrollo, estas funcionalidades son gratuitas para todos los diferentes ***early-adopters***.

5.3. JUSTIFICACIÓN DEL MODELO DE NEGOCIO

Es importante tener en cuenta la filosofía de este proyecto para entender cómo se ha elaborado este modelo de negocio. Inicialmente no se pretende realizar publicidad o vender datos tal y como hacen *otras plataformas y aplicaciones*[6]. En todo momento se pretende que el uso de la aplicación sea legítimo para el usuario.
De este modo, se ha establecido un modelo de negocio en el que salen beneficiados tanto los usuarios como los diferentes negocios, así como los diferentes participantes en este proyecto:

- Se ayuda al crecimiento de los **diferentes negocios** ofertando oportunidades y generando la posibilidad de mostrar sus actividades y participación dentro de un grupo de usuarios.
- Se oferta a los usuarios la **posibilidad de realizar planes ricos y enriquecedores**, además de funcionalidades extras en el caso de que éstos lo deseen. Además, se les ofrece de forma totalmente gratuita la posibilidad de gestionar sus planes con sus amigos y crear planes públicos, siendo el usuario el mayor beneficiado de todo este proceso.
- Como *empresa* se obtienen beneficios a través de resolver esta necesidad social tan vigente hoy en día.

De este modo, se considera que además de crear una **nueva oferta enriquecedora y no intrusiva para los usuarios**, se **prevé** el estar creando una **nueva oportunidad de mercado para los diferentes negocios**, estableciendo un modelo de negocio que **cumple íntegramente con la filosofía de este proyecto** y sus objetivos; aumentar la satisfacción social reforzando la sensación de comunidad.
Además, el modelo de negocio planteado se considera altamente rentable y escalable.

[6] Facebook, Instagram y Google son ejemplos de modelos de negocio basados en procesar sus datos para generar ingresos a través del flujo de usuarios.

5.4. TABLA RESUMEN DE MODELO DE NEGOCIO

Con el fin de aclarar cómo funcionará en primera instancia el modelo de negocio de ***PlanItNow!*** se ha elaborado la siguiente tabla a modo de resumen:

TABLA 3. TABLA RESUMEN DEL MODELO DE NEGOCIO DE PLANITNOW!. ELABORACIÓN PROPIA

	Promocionar Plan	*Modificar visibilidad planes*	*Suscripción Cambio Ubicación*	*Otras acciones*
Usuario	2 € / plan	2 € / cambio	10 € mensuales	Gratuito
Negocio	10€ mensuales	2 € / cambio	10 € mensuales	10€ mensuales

En las filas podemos ver las diferentes **entidades** que participan en el modelo de negocio, mientras que en las columnas podemos ver las diferentes **acciones** que pueden realizar gracias a las suscripciones.

6. ANÁLISIS FINANCIERO

La primera fase del lanzamiento de ***PlanItNow!*** está planificada para iniciar localmente en la ciudad de Valencia, la cual cuenta con un ambiente social y universitario que se ajusta a la perfección con el perfil de usuarios objetivo que tiene la aplicación.
Inicialmente se ofrece un servicio para cualquier persona que quiera promocionar un plan para conocer gente o para organizarse con amigos. A su vez, este servicio permite que varias empresas, las cuales son consideradas como los clientes, puedan promocionar y visibilizar sus servicios.
Teniendo en cuenta estos factores, se ha realizado un análisis financiero que contempla tanto los gastos como los potenciales ingresos de este proyecto en el marco de 5 años.
De cara facilitar la lectura de esta sección, se ha dividido en **costos**, **ingresos**, **gráficos** y **totales**.
Por último, hay que destacar que este análisis financiero se ha desarrollado en una etapa ideal tras ser finalizado este proyecto de fin de máster, asumiendo una inversión inicial.

6.1. COSTOS

En este apartado se muestra el costo total del proyecto dividido en diferentes categorías.

6.1.1. COSTO DE PERSONAL

En cuanto a los empleados que conformarán ***PlantItNow*!**, se deben tomar en cuenta las siguientes posiciones:

- **Empleado de Marketing**.

  Hay que contar con un departamento de marketing digital con 1 empleado encargado de gestionar la campaña publicitaria del servicio. Este empleado de marketing cobrará un salario de **25.000 euros brutos anuales**. Esto supone un coste para la empresa aproximado de 33.375,00 € anuales.

- **Empleado Comercial**.

  Se debe contar con un comercial a partir a finales del primer año, es decir, una vez el producto alcance mayor repercusión y se obtenga una mayor cantidad de potenciales clientes. Este comercial cobrará un salario de **18.000 € brutos anuales**. Esto supone un coste aproximado para la empresa de 24.030,00 € anuales.

- **Desarrollador**

  Será necesario contar, además, con un desarrollador (o varios a tiempo parcial) que se contabilizará como uno a tiempo completo cada año y que será quien se encargue del mantenimiento y de la gestión. Su sueldo **se estima en 25.000 euros brutos anuales** y para la estimación financiera se considera que será un **miembro de la plantilla desde el año 0, y contratado desde el primer momento.** Esto supone un coste aproximado para la empresa de 33.375 € anuales.

Hay que destacar que este coste aproximado se ha estimado a través de los **impuestos para la empresa** junto con el sueldo bruto [23].

Esto supone un coste mínimo anual de **90.780 €.**

6.1.1.1. ESTRATEGIA DE CONTRATACIÓN

A continuación, se procederá a enumerar la estrategia de contratación:

- **Si la cuota de mercado inicial es decreciente,** entonces no se contratarán más miembros.
- En el caso de que el **crecimiento sea moderado**, se contratará un segundo comercial al año 3 y un tercero para el año 5.
- Finalmente, si la cuota de **mercado crece rápidamente** será necesario un nuevo comercial cada año para mantener el estándar de calidad de atención al cliente.
- **En cualquier caso,** a partir del primer año se contratará otro desarrollador que permita mejorar la calidad del producto.

Respecto al caso de **desarrolladores** y **empleados de marketing**, así como otros roles, se irán contratando según la demanda que haya en el proyecto a lo largo de los años que éste se esté desarrollando. Inicialmente se deseará haber contratado **en total 3 desarrolladores y 2 agentes de marketing a partir del año 3.** Sin embargo, esto podría variar en un futuro según la demanda existente de tales roles.

6.1.2. COSTE INICIAL

De cara a comenzar a desarrollar el proyecto, será necesario realizar una inversión inicial. Esta inversión se basa en todos los productos tecnológicos que es necesario adquirir:

- **Servidores de Gama Media (máximo según presupuesto):** 4000€
- **Portátiles de Trabajo (inicialmente 3):** 4000€

Este es un presupuesto aproximado basado en los precios de mercado de los diferentes productos tecnológicos que cumplen con las características necesarias para el desarrollo del proyecto.
Por otro lado, se estima una **vida media de 5 años** para el servidor, así como los portátiles que se tratarán de **vender a partir del año 5** por **un 20% de su precio inicial**, es decir, recuperar una inversión de 1600 €, resultando en una **cuota de amortización anual** de 160 €.
Respecto al **software** a emplear, se pretende que **el primer año se emplee únicamente software gratuito** ya que este será clave en el desarrollo. A continuación, se evaluará el uso de software licenciado que aumente la productividad e ingresos de la empresa.
Todo esto supondrá un coste inicial de **8.000€**

6.1.3. OTROS COSTES

Es necesario también evaluar los costes de aspectos del proyecto referentes a luz, hosting, conexión a internet y otros factores que el proyecto pueda tener.
Teniendo en cuenta todos los factores mencionados, se estima un coste total de **3.650 €** de esta categoría.

6.2. INGRESOS

Teniendo en cuenta el modelo de negocio establecido en el **apartado 5**, se han enumerado los ingresos que la aplicación puede obtener en función de cada uno de los métodos de pago, así como su proyección y captación de clientes en **cinco años**.
Para desarrollar un volumen de los ingresos preciso, es necesario evaluar los siguientes escenarios:

- **Pesimista.** 25 % de variación anual[7].
- **Moderado.** 50% de variación anual.
- **Optimista.** 75% de variación anual.

Por otro lado, hay que decir que para desarrollar este análisis se ha empleado una **cuota inicial de mercado** de entre **0,1% y el 0,001%** según el caso, puesto que esta cuota depende de a quién vaya dirigido el marketing de la aplicación, así como la facilidad de captación de clientes para los diferentes agentes del departamento de ventas.

6.2.1. INGRESOS POR SUSCRIPCIÓN DE NEGOCIOS

Recordemos que durante la **sección 3** se realiza una estimación de los potenciales negocios que podrían suscribirse a este proyecto. Eso significa que contamos con aproximadamente 1.000.000 de potenciales negocios que podrían suscribirse. De este modo, se partirá con **1.000 negocios a partir del año 1,** es decir, el 0,1% de todos los potenciales clientes respecto a los negocios.
Partiendo de nuestra **cuota inicial establecida**, y de que el **año 0** no se tendrá ningún cliente, debido a que se está desarrollando y captando clientes, obtenemos la siguiente tabla para cada uno de los casos mencionados anteriormente:

TABLA 4. INGRESOS POR CLIENTES EN FUNCIÓN DE AÑO Y ESCENARIO. ELABORACIÓN PROPIA

	Año 0	*Año 1*	*Año 2*	*Año 3*	*Año 4*	*Año 5*
Pesimista	0 €	120.000 €	150.000 €	187.500 €	234.375 €	292.968 €
Esperado	0 €	120.000 €	180.000 €	270.000 €	405.000 €	607.500 €
Optimista	0 €	120.000 €	210.000 €	367.500 €	643.125 €	1.125.468 €

En la tabla podemos ver que el año 0 no se obtiene ningún ingreso debido a que será un año de desarrollo y captación de clientes, mientras que en los años posteriores se puede ver el crecimiento económico en el caso de que la aplicación resulte exitosa.
Por otro lado, para calcular el **número de clientes necesario**, sólo deberemos realizar la siguiente fórmula:
Clientes Suscritos = (Ingresos Generados) / (Precio Por Suscripción)

6.2.2. INGRESOS POR SUSCRIPCIÓN DE USUARIOS

Tal y como se ha mencionado durante la **sección 3**, se estima una cantidad de 16.365.766 de potenciales usuarios en España, sin embargo, sería imprudente asumir que el 0,1% de todos ellos van a suscribirse a cambio de la funcionalidad de ubicación, es por ello, que,

[7] La variación anual es el cambio de porcentaje entre dos valores. Para el caso de este análisis financiero hace referencia al porcentaje de potenciales clientes que se han adquirido/perdido. Por ejemplo, un 25% de variación anual significará que se adquieren un 25% de clientes más respecto al año anterior.

para realizar una estimación más precisa, se ha empleado para este caso el **0,001%** de cuota inicial de mercado para realizar las estimaciones.
Este análisis de ingresos es análogo al del apartado anterior tanto en la fórmula como en los cálculos realizados para predecir los diferentes casos. A continuación, se muestra la tabla que refleja los ingresos en esta modalidad:

TABLA 5. INGRESOS POR USUARIOS SUSCRITOS EN FUNCIÓN DE AÑO Y ESCENARIO. ELABORACIÓN PROPIA

	Año 0	*Año 1*	*Año 2*	*Año 3*	*Año 4*	*Año 5*
Pesimista	0 €	19.638,92 €	24.548,65 €	30.685,81 €	38.357,26 €	47.946,58 €
Esperado	0 €	19.638,92 €	29.458,38 €	44.187,57 €	66.281,35 €	99.422,03 €
Optimista	0 €	19.638,92 €	34.368,11 €	60.144,19 €	105.252,33 €	181.191,58 €

Tal y como ocurre en el caso anterior, para calcular el número de **clientes necesario,** se empleará la siguiente fórmula:
Usuarios Suscritos = (Ingresos Generados) / (Precio Por Suscripción)

6.2.3. INGRESOS POR FUNCIONALIDADES EXTRAS DE USUARIOS

En este caso, usaremos el mismo número estimado sobre la cantidad de potenciales usuarios. Para realizar estos cálculos, se empleará el **0,01**% de cuota de mercado. Esto quiere decir que, el **0,01%** de los potenciales usuarios efectuará pagos, al menos, una vez al año por funcionalidades extras o en su defecto, que el mismo usuario realizará varios pagos supliendo esta cantidad.
Recordemos que el precio por ejecutar una funcionalidad extra en la aplicación será de **2€,** de este modo, realizando de nuevo el análisis para los diferentes escenarios podemos obtener la tabla mostrada a continuación:

TABLA 6. INGRESOS POR ESCENARIO Y AÑO GENERADOS CON FUNCIONALIDADES EXTRAS DE USUARIOS. ELABORACIÓN PROPIA

	Año 0	*Año 1*	*Año 2*	*Año 3*	*Año 4*	*Año 5*
Pesimista	0 €	39.277,84 €	49.097,30 €	61.371,62 €	76.714,53 €	95.893,16 €
Esperado	0 €	39.277,84 €	58.916,76 €	88.375,14 €	132.562,70 €	198.844,06 €
Optimista	0 €	39.277,84 €	68.736,22 €	120.288,38 €	210.504,67 €	368.383,16 €

En este caso, se empleará la siguiente fórmula para calcular el **número de acciones** que es necesario que los usuarios ejecuten para lograr tal cantidad de ingresos:
NºUso de Funcionalidades = (Ingresos Generados) / (Precio por Uso)

6.3. TOTALES

En este apartado se procederá a mostrar los costos e ingresos totales en conjunto de cara a entender el análisis financiero de este proyecto y validar su rentabilidad.

6.3.1. COSTE TOTAL POR AÑO

A continuación, muestra una tabla conteniendo los diferentes costes totales por año desglosados, así como el coste total según los diferentes escenarios.

TABLA 7. COSTES FIJOS TOTALES DESGLOSADOS POR TIPO, ESCENARIO Y AÑO. ELABORACIÓN PROPIA

		Año 0	Año 1	Año 2
Sueldos Comerciales				
Mercado Creciente		24.030,00 €	48.060,00 €	72.090,00 €
Mercado Moderado		24.030,00 €	48.060,00 €	48.060,00 €
Mercado Decreciente		24.030,00 €	24.030,00 €	24.030,00 €
Otros Costes		3.650,00 €	3.650,00 €	3.650,00 €
Personal Marketing		33.375,00 €	33.375,00 €	33.375,00 €
Coste Inicial		8.000 €	- €	- €
Personal Desarrollo		33.375,11 €	66.750,21 €	66.750,21 €
Coste total escenario	Pesimista	102.430,11 €	127.805,21 €	127.805,21 €
	Esperado	102.430,11 €	151.835,21 €	151.835,21 €
	Optimista	102.430,11 €	151.835,21 €	175.865,21 €
		Año 3	**Año 4**	**Año 5**
Sueldos Comerciales				
Mercado Creciente		96.120,00 €	120.150,00 €	144.180,00 €
Mercado Moderado		72.090,00 €	72.090,00 €	96.120,00 €
Mercado Decreciente		24.030,00 €	24.030,00 €	24.030,00 €
Otros Costes		3.650,00 €	3.650,00 €	3.650,00 €
Personal Marketing		66.750,00 €	66.750,00 €	66.750,00 €
Coste Inicial		- €	- €	- €
Personal Desarrollo		100.125,32 €	100.125,32 €	100.125,32 €

Coste total escenario	Pesimista	194.555,32 €	194.555,32 €	194.555,32 €
	Esperado	242.615,32 €	242.615,32 €	266.645,32 €
	Optimista	266.645,32 €	290.675,32 €	314.705,32 €

Podemos observar cómo el mayor coste se encuentra en los escenarios Esperado y Optimista, ya que también son los escenarios en los que se prevén un mayor número de ingresos.

6.3.2. INGRESOS TOTALES POR AÑO

A continuación, se mostrarán las diferentes tablas que contendrán los ingresos totales por año, según los diferentes escenarios. Para ello, se tomarán como referencia las diferentes tablas mostradas en el apartado **6.2.**

6.3.2.1. INGRESOS TOTALES POR AÑO EN ESCENARIO PESIMISTA

A continuación, se mostrará la tabla con los ingresos totales para el escenario pesimista:

TABLA 8. DESGLOSE DE INGRESOS TOTALES EN EL CASO PESIMISTA. ELABORACIÓN PROPIA

	Año 0	*Año 1*	*Año 2*	*Año 3*	*Año 4*	*Año 5*
Suscripción de Negocios	0 €	120.000 €	150.000 €	187.500 €	234.375 €	292.968 €
Suscripción de usuarios	0 €	19.638,92 €	24.548,65 €	30.685,81 €	38.357,26€	47.946,58 €
Funcionalidades Extras de usuarios	0 €	39.277,84 €	49.097,30 €	61.371,62 €	76.714,53 €	95.893,16 €
Total	0€	178.916,76 €	223.646 €	279.557 €	349.447 €	436.807,74 €

6.3.2.2. INGRESOS TOTALES POR AÑO EN ESCENARIO ESPERADO

A continuación, se mostrará la tabla con los ingresos totales para el escenario esperado:

TABLA 9. DESGLOSE DE INGRESOS TOTALES EN EL CASO ESPERADO. ELABORACIÓN PROPIA

	Año 0	*Año 1*	*Año 2*	*Año 3*	*Año 4*	*Año 5*
Suscripción de Negocios	0 €	120.000 €	180.000 €	270.000 €	405.000 €	607.500 €
Suscripción de usuarios	0 €	19.638,90€	29.458,38 €	44.187,57 €	66.281,35 €	99.422,03 €
Funcionalidades Extras de usuarios	0 €	39.277,84 €	58.916,76 €	88.375,14 €	88.375,14 €	198.844,06 €
Total	0€	178.917 €	268.375 €	402.562,71 €	559.656,49€	905.766,09 €

6.3.2.3. INGRESOS TOTALES POR AÑO EN ESCENARIO OPTIMISTA

A continuación, se mostrará la tabla con los ingresos totales para el escenario optimista:

TABLA 10. DESGLOSE DE INGRESOS TOTALES EN EL CASO OPTIMISTA. ELABORACIÓN PROPIA

	Año 0	*Año 1*	*Año 2*	*Año 3*	*Año 4*	*Año 5*
Suscripción de Negocios	0 €	120.000 €	210.000 €	367.500 €	643.125 €	1.125.468 €
Suscripción de usuarios	0 €	19.638,92 €	34.368,11 €	60.144,19 €	105.252,33 €	181.191,58 €
Funcionalidades Extras de usuarios	0 €	39.277,84 €	68.736,22 €	120.288,38 €	210.504,67 €	368.383,16 €
Total	0€	178.916,76 €	313.104 €	547.933 €	958.882 €	1.675.043 €

6.3.3. BENEFICIOS TOTALES

Una vez se han calculado los costes e ingresos para los diferentes escenarios, se puede dar pie al cálculo de los diferentes potenciales beneficios para cada uno de los escenarios. En este apartado se evaluará si los diferentes escenarios son (o no) rentables, y se realizará una validación del **modelo de negocio** en el caso de que el proyecto resulte rentable.
Para realizar el cálculo de los beneficios de la forma más precisa posible, se han empleado los diferentes resultados del apartado **6.2,** del que ya conocemos el desglose de los diferentes ingresos y costes de la aplicación para cada uno de los diferentes escenarios.
Para ello, se aplica una fórmula simple, siendo ésta la resta de los ingresos respecto a los costes.
Por último, hay que aclarar que a la hora de realizar estos cálculos **no se aplicará el impuesto de sociedades** (25% de los beneficios totales).

6.3.3.1. BENEFICIOS TOTALES ESCENARIO PESIMISTA

TABLA 11. CÁLCULO DE BENEFICIOS TOTALES PARA EL ESCENARIO PESIMISTA. ELABORACIÓN PROPIA

	Año 0	*Año 1*	*Año 2*	*Año 3*	*Año 4*	*Año 5*
Total Ingresos	0€	178.916,76 €	223.646 €	279.557 €	349.447 €	436.807,74 €
Total Costes	102.430,11 €	127.805,21 €	127.805,21 €	194.555,32 €	194.555,32 €	194.555,32 €
Total Beneficios	-102.430,11€	51.111,55 €	95.840,79 €	85.001,68 €	154.891,68 €	242.252,42 €

Tal y como se puede apreciar, el único año en el que se generarían pérdidas sería en el año 0 puesto que el proyecto estaría en desarrollo y hasta comenzar el año 2 no se suplirían estas pérdidas con los beneficios generados, haciendo que el proyecto sea rentable.

6.3.3.2. BENEFICIOS TOTALES ESCENARIO ESPERADO

TABLA 12. CÁLCULO DE BENEFICIOS TOTALES PARA EL ESCENARIO ESPERADO. ELABORACIÓN PROPIA

	Año 0	*Año 1*	*Año 2*	*Año 3*	*Año 4*	*Año 5*
Total Ingresos	0€	178.917 €	268.375 €	402.562,71 €	559.656,49 €	905.766,09 €
Total Costes	102.430,11 €	151.835,21 €	151.835,21 €	242.615,32 €	242.615,32 €	266.645,32 €
Total Beneficios	-102.430,11€	27.081,79 €	116.539,79 €	159.947,39 €	317.041,17 €	639.120,77 €

Tal y como se puede apreciar en la tabla, ocurre igual que en el escenario anterior, hasta el año 2 no se suplen las pérdidas haciendo que el proyecto sea rentable, sin embargo, sí que se adquieren unos beneficios significativamente mayores gracias a la adquisición de un nuevo comercial durante el **año 1.**

6.3.3.3. BENEFICIOS TOTALES ESCENARIO OPTIMISTA

TABLA 13. CÁLCULO DE BENEFICIOS TOTALES PARA EL ESCENARIO OPTIMISTA. ELABORACIÓN PROPIA

	Año 0	*Año 1*	*Año 2*	*Año 3*	*Año 4*	*Año 5*
Total Ingresos	0€	178.916,76€	313.104 €	547.933 €	958.882 €	1.675.043€
Total Costes	102.430,11€	151.835,21€	175.865,21€	266.645,32€	290.675,32€	314.705,32€
Total Beneficios	-102.430,11€	27.081,55 €	137.238,79 €	281.287,68 €	668.206,68 €	1.360.337,68 €

Tal y como se puede apreciar en la tabla, ocurre igual que en los dos escenarios anteriores, y los dos primeros años son muy similares, sin embargo, podemos ver que a partir del tercer año el proyecto cuenta con un crecimiento exponencial resultando en un proyecto altamente rentable con un gran número de empleados comerciales.

6.3.3.4. GRÁFICO DE BENEFICIOS

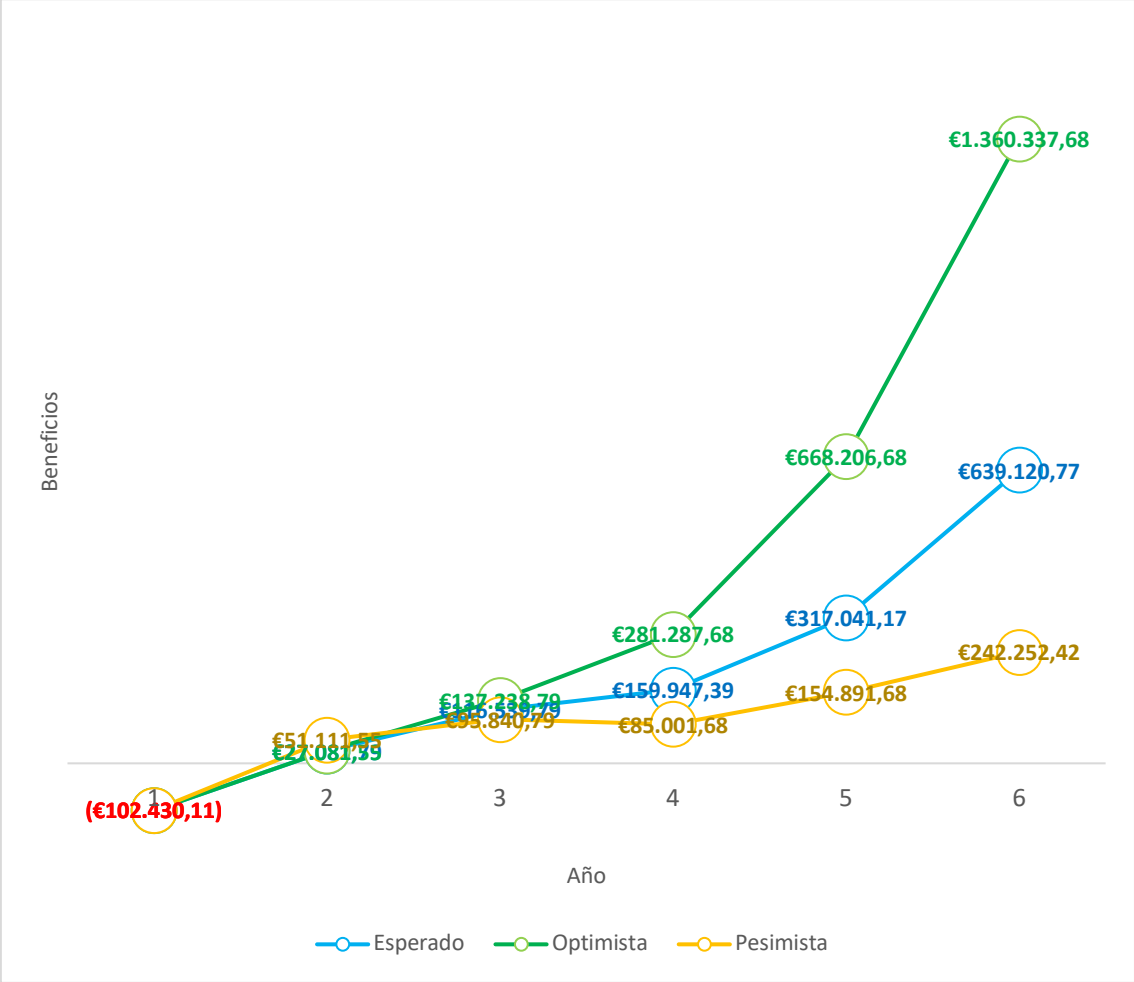


ILUSTRACIÓN 13. GRÁFICO DE BENEFICIOS POR AÑO SEGÚN ESCENARIO. ELABORACIÓN PROPIA

Como se puede ver en el gráfico superior, en todos los escenarios existe una pequeña caída de los beneficios entre los años 2 y 4 salvo en el escenario optimista. Además, se parte del caso base de que hasta el 2º año no se comienza a generar ningún tipo de beneficio relevante. De este modo podemos concluir que el proyecto, en cualquiera de los casos **comenzará a ser rentable a partir del año 2.**

Por otro lado, también se puede prever que **una posible inversión inicial** para comenzar al menos el primer año **rondará los 102.430,11€,** sea cual sea el escenario.

Viendo este gráfico se puede concluir que el proyecto a nivel financiero es viable teniendo en cuenta todos los factores que se han mencionado a lo largo de este capítulo. También podemos observar que, en un caso de rotundo éxito, podría llegar a generar grandes beneficios puesto que el modelo de negocio resulta una escalada exponencial de los diferentes ingresos, ya que estos varían en función del número de suscriptores con los que cuente la aplicación.

7. LEAN CANVAS

El *Lean Canvas* [6] es una herramienta que nos permitirá visualizar en una única tabla el modelo de negocio de este proyecto, la idea, así como aquello que se desea lograr mediante el método *Lean Startup*. Dicho de otro modo, podríamos definirlo como un **lienzo de negocio ligero,** permitiendo entender a modo de resumen posible el cómo se desea desarrollar y gestionar este proyecto. Para ello, este formato cuenta con varias secciones que se pueden ver a continuación.
Este apartado trata de resumir todo lo mencionado a lo largo de los apartados anteriores.

2 PROBLEMA

Hoy en día existe **un desconocimiento de planes sociales**, así como una **dificultad para relacionarse con gente cercana**.
Esto se debe a **que las redes sociales no contribuyen a relacionarse físicamente** o realizar actividades con las personas que están cerca de ti o incluso tus propios amigos.
A pesar de que sitios **como Facebook cuentan con un gestor de eventos** bastante efectivo, no deja de ser **una funcionalidad totalmente secundaria** de la aplicación que muchos usuarios pasan por alto y no se le saca todo el partido que se debiera.
En ***PlanItNow!*** ya se consideran incluir todas las características de este sistema y ampliarlo para hacer que **los usuarios tengan una mayor facilidad para interconectarse**.

4 SOLUCIÓN

PlanItNow! ofrece **visualizar los diferentes planes i**dealmente en modo de mapa para ver aquellos cerca de ti. Además, **recomendará** planes en función de los gustos del usuario a través de un sistema de gestión de planes. Por otro lado, cada plan dispondrá de un chat en el que únicamente se pueda participar en zonas cercanas al evento para el caso de los eventos públicos.
Finalmente, **todos los usuarios serán capaces de acceder a cualquier plan público** de manera inmediata y sencilla, donde también podrán visualizar qué amigos asisten a cada evento.

3 PROPOSICIÓN DE VALOR

PlanItNow! invita a sus usuarios a participar en **una experiencia colaborativa donde sus usuarios pueden crear y participar en eventos dependiendo de su localización** y lista de amigos en la aplicación.
Sus funcionalidades innovadoras proporcionan **una solución eficaz y dinámica para encontrar eventos personalizados a los gustos de sus usuarios** de tal forma que consigan propuestas atractivas para conocer a nuevas personas y participar en actividades con sus amigos.
Finalmente, es un producto que ofrece una solución en un mercado poco desarrollado ofreciendo una experiencia social en todo momento.

9 VENTAJA COMPETITIVA

Entre las características principales que diferencian la aplicación de sus competidores se **encuentra la sensación de inmediatez**, donde los usuarios tienen la opción de participar en **eventos cercanos a su ubicación y que están transcurriendo tanto en tiempo real cómo en horas/días próximos**.
Esto ofrece **una manera innovadora de ofrecer actividades** afines con los usuarios, así como también ayuda a encontrar personas con gustos similares.
Otra diferencia fundamental es la filosofía que invita a los usuarios a **disminuir la cantidad de horas que invierten en las redes sociales**, con la alternativa de ofrecer planes sociales en su entorno.

1 CLIENTES

Público principalmente joven (18 – 45 años).
Los usuarios pueden tener cualquier rango de ingresos ya que los planes pueden tener algún coste (eventos con entrada, conciertos) o no (salir a caminar, hacer deporte).
Empresas con eventos que quieran obtener visibilidad en el mercado o negocios que quieran promocionarse en la aplicación (bares, restaurantes, etc.).
Hay dos tipos de usuarios:

- Usuarios registrados
- Empresas

Geográficamente, se prevé que en un futuro la aplicación estará disponible en todo el mundo, aunque inicialmente se tratará únicamente el escenario de España de cara a focalizar los diferentes esfuerzos de *marketing y promoción* de la aplicación.

8 MÉTRICAS

- Cantidad de eventos creados por zonas.
- Cantidad de usuarios registrados.
- Cantidad de interacciones.
- Cantidad y calidad de evaluaciones y reviews.
- Cantidad de funcionalidades extras empleadas por usuarios.
- Cantidad de usuarios activos.
- Cantidad de suscriptores.

5 CANALES

- Redes sociales
- Boca a boca
- Reviews de *influencers* y personalidades de internet.
- Eventos donde se publicite la app

7 COSTOS

- Departamento de Marketing
- Sueldos de los Desarrolladores
- Hardware para la infraestructura
- Adquisición de servidores

6 INGRESOS

- Cobro por suscripciones *premium* para empresas
- Ingresos por funcionalidades extras
- Ingresos por suscripciones de usuarios

8. CONCLUSIONES DE LA EVALUACIÓN

En este apartado se valora la oportunidad y la viabilidad de la idea de negocio propuesta a lo largo de los capítulos anteriores.
Para comenzar, hay que destacar que, pese a que se ha establecido un modelo de negocio inicial teniendo en cuenta análisis anterior, en un futuro tal y como se menciona en el ***Lean Canvas*** de la **sección 7**, en un caso ideal se pretende realizar un lanzamiento de la aplicación a nivel mundial.
También se considera la posibilidad de *pivotar* el negocio en el caso de que en situaciones futuras se detecten mayores oportunidades.

8.1. OPORTUNIDAD DE NEGOCIO

A lo largo de la **sección 3,** se ha realizado un análisis de mercado en anchura. En este análisis podemos ver dos principales factores:

- **Un nicho de mercado** muy amplio, aún por explotar.
- **Una competencia férrea**, pero que está enfocada a **otros factores** de la explotación del *mercado de eventos y planes.*

Partiendo de esta base, se podría llegar fácilmente a la conclusión de que una idea como ***PlanItNow!*** **podría encajar perfectamente en un mercado como el actual**, ofertando una *alternativa* a las diferentes plataformas sociales, ya que gran parte de ellas tienen otros fines más allá del conectar personas a lo largo de su entorno. Dicho de otro modo, **lo que más puede hacer destacar un proyecto como este es la filosofía que hay tras él**, que llevada bien a la práctica podría lograr una gran aportación social, así como una elevada generación de ingresos.
Sin embargo, tal y como prevé en el *análisis DAFO*, una de las grandes amenazas de este proyecto es el hecho de **ser clonado** o **la escasez de usuarios dispuestos a usar una aplicación de esta envergadura.** De este modo, han sido aspectos completamente clave de cara al desarrollo y comercialización de la plataforma pues se ha trabajado principalmente en aquello que diferencia a la aplicación de su competencia.
Vistos los diferentes puntos clave de cara a una oportunidad de negocio y viendo las fortalezas de la competencia, **se ha identificado que la mayor oportunidad de negocio con la que cuenta un proyecto como *PlanItNow!*** es la de tratar de **crear un *mercado* de planes y actividades localizado por diferentes zonas**, así como el gestar una base de usuarios con los mismos objetivos que los de este proyecto. Este mercado de planes se pretende respaldar por la aplicación, logrando **una sinergia entre usuarios, negocios y tecnología.**
Por último, y para poder satisfacer esta oportunidad de negocio, será necesario realizar un gran esfuerzo a nivel de *Marketing* y *Desarrollo*.

8.2. VIABILIDAD DE LA IDEA DE NEGOCIO

Tal y como se ha podido ver en las **secciones 5 y 6,** se ha elaborado un modelo de negocio del cual se ha realizado un análisis financiero.

De cara a establecer el modelo de negocio, se ha tenido en cuenta quiénes serán los mayores beneficiados del uso de esta aplicación, y es por ello por lo que se ha decidido obtener los ingresos de aquellos que puedan resultar beneficiados a nivel económico del uso de esta aplicación.

Una de las mayores debilidades que se pueden encontrar respecto al modelo de negocio escogido es que es completamente necesario el tener una gran base de usuarios para comenzar a atraer potenciales clientes del mundo de los negocios. Sin embargo, una vez esté el producto establecido se ha demostrado a través de **análisis financiero** que se puede lograr una **alta rentabilidad**.

Es por ello que es necesario mitigar esta debilidad, especialmente en las primeras etapas del negocio.

Por esta misma razón, si a partir del **año 1** se observa que no se obtienen el número de suscriptores mínimos para poder dar lugar al crecimiento económico del proyecto, será necesario volver a evaluar el modelo de negocio.

Sin embargo, con el modelo de negocio que se ha establecido y teniendo en cuenta el ***personal cualificado*** que, idealmente trabajaría en este proyecto, se puede observar a lo largo del **análisis financiero** que en todos los casos se **cesan las pérdidas tras el año 2 y se comienzan a obtener beneficios** lo cual es un gran indicador de viabilidad sobre la idea y el modelo de negocio propuesto.

De esto se concluye que la viabilidad del negocio se verá de forma real a partir de este año, y será necesaria una inversión inicial que permita al personal continuar trabajando en la idea durante este marco de tiempo, así como la adquisición de los diferentes productos tecnológicos que permitan este desarrollo.

DESARROLLO DE LA IDEA DE NEGOCIO

9. MAPA DE CARACTERÍSTICAS

El mapa de características consiste en una representación de las principales características del producto a desarrollar. De este modo, se pueden obtener las características principales, así como la posibilidad de agruparlas, jerarquizarlas y establecer prioridades. Esta categorización se puede realizar de diversas formas, ya sea por tiempo, urgencia, características o tecnología.
Debido a que esto es un proyecto final de máster a la par que un proyecto emprendedor, se han establecido las diferentes agrupaciones en función de las necesidades de los usuarios, así como el interés tecnológico que tienen las diferentes características de la aplicación. Por otro lado, se han desarrollado ***MVPs*** que, si bien no podrían ser lanzados al mercado al carecer de las funcionalidades de pago, contribuyen a validar la idea de forma agresiva a través del uso de la aplicación.
Por último, a raíz de este mapa de características se han deducido las diferentes tareas tecnológicas y no tecnológicas necesarias de cara a desarrollar este producto software.

9.1. MÉTODO MOSCOW

De cara a establecer prioridades en este mapa de características, se ha decidido emplear el método **MoSCoW** [24] [25], este método consiste en agrupar las diferentes características en las siguientes clasificaciones:

- **Must have**
  Aquellas características etiquetadas con ***must***, son aquellas críticas de cara a entregar la aplicación. Si estas características no se logran implementar en el marco de tiempo acordado, el despliegue se considerará un fracaso.

- **Should have**
  Aquellas características etiquetadas con **should,** son aquellas importantes, pero no necesarias de cara a entregar la aplicación. En el caso de estas características, no es crítico el entregarlas en el marco de tiempo acordado, aunque sí que es importante implementarlas en un futuro cercano.

- **Could have**
  Aquellas características etiquetadas con **could**, son aquellas que sería ideal de tener, pero no son ni importantes ni necesarias. Estas características deberían únicamente mejorar la experiencia del usuario final a cambio de un pequeño coste de desarrollo, pero no serán claves de cara a marcar la viabilidad del producto. Típicamente estas características se implementan si el tiempo y los recursos lo permiten.

- **Won't have**
  Aquellas características etiquetadas como **won't**, son aquellas menos críticas, es decir, aquellas que aportan menos al producto, resultan secundarias de cara a marcar el ciclo de vida del producto o simplemente no son apropiadas para el momento en el que se establecen. De este modo, las características etiquetadas como **won't** no se planean ser implementadas en un futuro cercano, pero se pueden tener en cuenta de cara al largo plazo del producto.

9.2. REPRESENTACIÓN DEL MAPA DE CARACTERÍSTICAS

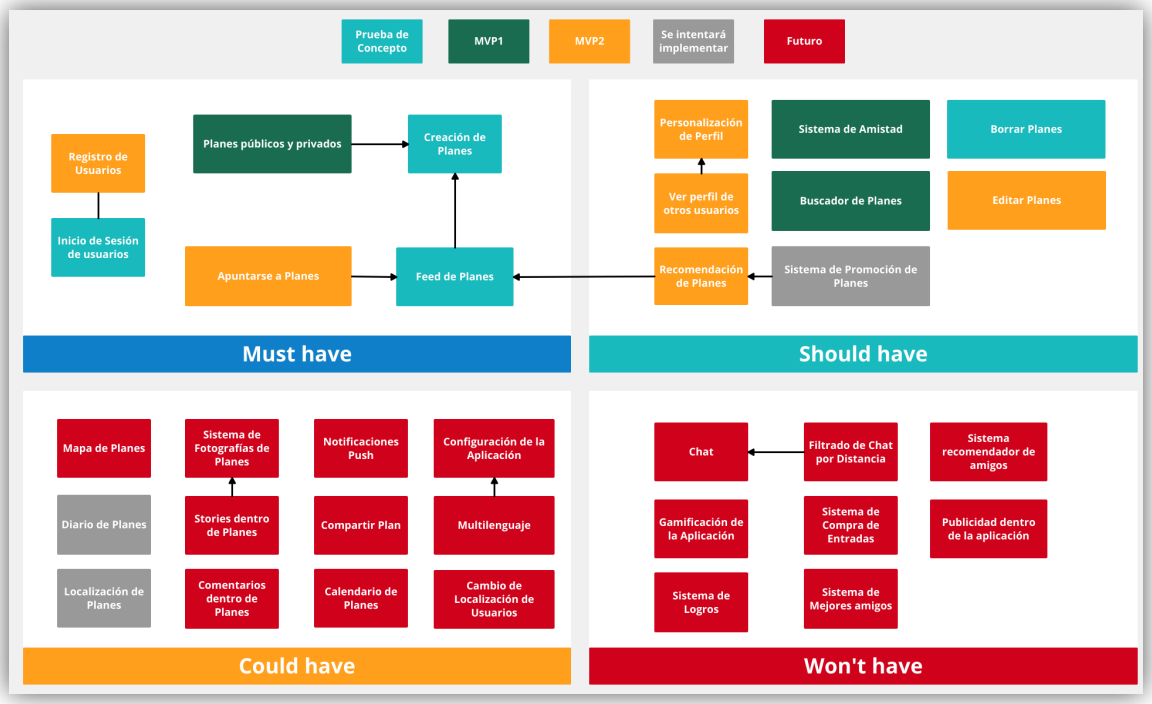


ILUSTRACIÓN 14. MAPA DE CARACTERÍSTICAS DE LA APLICACIÓN. ELABORACIÓN PROPIA

Como se puede apreciar en el mapa de características, ***los Must y Should*** se han implementado a lo largo de los dos primeros MVPs. Cabe decir que el registro no ha sido implementado hasta el 2º MVP ya que se pretende realizar de forma manual hasta esta fase. A partir de este 2º MVP se permite el registro de usuarios de forma más amplia para que estas características puedan ser usadas en un grupo mayor.
También pueden verse las diferentes fases por las que pasará el proyecto. Para cada una de las fases es necesario establecer un *backlog*[8] [26] del que nacen de las diferentes tareas a implementar en cada una de las fases.

[8] Término empleado en la Ingeniería del Software para referirse a la reserva de tareas a desarrollar a lo largo de un proyecto. Estas tareas serían las que quedan pendientes por hacer.

10. BACKLOG DEL PROYECTO

En esta sección se muestra y desglosa el *backlog* resultante, obtenido a través de la observación del *Mapa de Características*.
Las diferentes tareas están ordenadas en función de los siguientes parámetros:

- Validación de la idea
- Valor para el usuario
- Riesgo de desarrollo
- Oportunidad de negocio

10.1. CLASIFICACIÓN POR TIPO DE TAREA

Cabe decir que, ya que este proyecto es un proyecto de desarrollo de software, gran parte de estas tareas contendrán las **siguientes etiquetas**:

- **FRONT.** Se hace referencia a aquellas tareas visuales, las que se verán finalmente representadas en la aplicación móvil.
- **BACK.** Se hace referencia a aquellas tareas lógicas, conceptuales y no visuales. Este tipo de tareas tratarán de encapsular los diferentes mecanismos de la aplicación.
- **INTEGRACIÓN.** Se hace referencia a las diferentes tareas encargadas de unir las tareas de FRONT y BACK para poder dar lugar a una aplicación completamente funcional.
- **DISEÑO.** Se hace referencia a aquellas tareas de diseño que tratan de aplicar patrones y estructuras para que el proyecto resulte exitoso.

10.2. CLASIFICACIÓN POR ÉPICA

Por otro lado, de cara a organizar el proyecto al largo plazo se han creado las siguientes ***épicas***[9]:

- **PLANES.** Son todas aquellas tareas relacionadas con los diferentes planes.
- **MODELO.** Son todas aquellas tareas relacionadas con el modelo del sistema, ya sea a nivel conceptual, de diseño o que requiera algún cambio en el modelo original.
- **DOCUMENTACIÓN.** Son todas aquellas tareas en las que es necesaria la redacción de documentación.
- **USUARIOS.** Son todas aquellas tareas relacionadas con la gestión y creación de usuarios, así como sus relaciones con el sistema.
- **PERSISTENCIA.** Son todas aquellas tareas centradas en almacenar los datos en alguna estructura de almacenamiento de cara a que perduren a lo largo del tiempo.
- **RECOMENDACIÓN.** Son todas aquellas tareas centradas en dar lugar al algoritmo de recomendación de planes.
- **PRUEBAS.** Son todas aquellas tareas orientadas a probar algún aspecto del sistema, sea éste funcional o no funcional.
- **CALIDAD.** Son todas aquellas tareas excepcionales orientadas a mejorar la calidad del producto, ya sea a través de mejoras visuales de rendimiento u otros factores.

Estas dos categorías de etiquetas se han usado a lo largo del proyecto para clasificar las diferentes tareas y aprovechar al máximo la posibilidad de trabajar las diferentes tareas en paralelo.

[9] Tecnicismo utilizado en el desarrollo de software *agile* para referirse a una gran tarea que es posteriormente segmentada en diferentes subtareas.

10.3. REPRESENTACIÓN DEL BACKLOG OBTENIDO

De cara a facilitar la legibilidad de este documento, se estructurará el *backlog* resultante a modo de tabla. El orden de las tareas tratará de corresponder al mismo orden en el que inicialmente pretenden ser implementadas.

TABLA 14. BACKLOG OBTENIDO A PARTIR DEL MAPA DE CARACTERÍSTICAS Y SU FASE CORRESPONDIENTE. ELABORACIÓN PROPIA

Tarea	Descripción	Etiquetas	Fase
Toma de contacto con Android Front-end	Debido a la inexperiencia del desarrollador principal, será necesario establecer una primera toma de contacto con las tecnologías a emplear.	**FRONT, CALIDAD**	Prueba de Concepto
Django Backend	Preparación del proyecto a nivel de backend para poder comenzar a conectar los datos lo antes posible.	**BACK, PERSISTENCIA**	Prueba de Concepto
Pulir Requisitos y Casos de Uso	Gestión de los requisitos y casos de uso iniciales del proyecto, de cara a realizar un análisis en profundidad sobre la viabilidad del proyecto a nivel de desarrollo.	**CALIDAD, DOCUMENTACIÓN**	Prueba de Concepto
Integrar API GraphQL	Estudio sobre cómo realizar la integración de la API GraphQL generada en Django con la aplicación resultante.	**BACK, FRONT, DISEÑO**	Prueba de Concepto y MVP1
Solicitar Licencia MagicDraw	Solicitud de una licencia de la herramienta de diseño MagicDraw.	**DOCUMENTACIÓN**	Prueba de Concepto
Dashboard View Model	Desarrollo de la primera interfaz gráfica del proyecto Android empleando la librería para crear un Dashboard View Model.	**CALIDAD**	Prueba de Concepto
Definir Modelo Conceptual	Diseño del modelo conceptual de la aplicación que será usado a lo largo de todas las iteraciones del producto.	**MODELO, DOCUMENTACIÓN**	Prueba de Concepto
Implementar Navegación	Desarrollo de la navegación de actividades y ventanas dentro de la aplicación Android.	**FRONT**	Prueba de Concepto y MVP1
Estructura de Proyecto	Estructuración de las clases del proyecto de cara a facilitar el desarrollo.	**FRONT**	Prueba de Concepto y MVP1

Tarea	Descripción	Etiquetas	Fase
Estructura de Carpetas	Similar a la anterior, pero realizando la correcta estructuración de carpetas y paquetes del proyecto	**FRONT**	Prueba de Concepto y MVP1
Crear Plan	Implementación del caso de uso de crear plan	**PLANES, FRONT**	Prueba de Concepto
Feed de Planes	Implementación del caso de uso de ver feed de planes	**PLANES, FRONT**	Prueba de Concepto
Borrar Plan	Implementación del caso de uso borrar plan	**PLANES, FRONT**	Prueba de Concepto
Ver Plan	Implementación del caso de uso ver plan	**PLANES, FRONT**	Prueba de Concepto
Implementar Login	Implementación del caso de uso login	**USUARIOS, FRONT**	Prueba de Concepto
Implementar Modelo Conceptual	Implementación de las clases establecidas en el modelo conceptual	**MODELO, FRONT**	Prueba de Concepto
Persistir Datos en App Android	Almacenar los diferentes datos cruciales de la aplicación en una base de datos	**PERSISTENCIA, FRONT, BACK**	Prueba de Concepto
Persistir Planes	Almacenar los planes en una base de datos	**PERSISTENCIA, FRONT**	Prueba de Concepto
Añadir Imágenes a Planes	Dar a los planes la posibilidad de contener imágenes	**PLANES, FRONT**	Prueba de Concepto
Mostrar Planes Ordenados	A la hora de mostrar diversos planes, se deberán de mostrar ordenados según su fecha	**PLANES, FRONT**	Prueba de Concepto

Tarea	Descripción	Etiquetas	Fase
Despliegue del Proyecto	Despliegue del proyecto en un hosting de aplicaciones web	**INTEGRACIÓN**	MVP1
Visualizar Planes Contenidos en el Back	Posibilidad de ver los planes que están contenidos en el backend a través de la API	**FRONT, INTEGRACIÓN**	MVP1
Crear Planes en el Back	Creación de planes a través de la API	**FRONT, BACK, INTEGRACIÓN**	MVP1
Borrar Planes en el Back	Posibilidad de borrar planes a través de la API	**FRONT, BACK, INTEGRACIÓN**	MVP1
Refactorizar toda la App a Kotlin	Tras un tiempo de desarrollo, se ha visto la necesidad de modificar todo el código de la aplicación de JAVA a KOTLIN para facilitar el desarrollo.	**FRONT, CALIDAD**	MVP1
Añadir Validación a las consultas y mutaciones importantes	Es necesario validar las consultas y mutaciones que se realizan a la API de cara a otorgar mayor seguridad al sistema.	**FRONT, BACK, INTEGRACIÓN**	MVP1
Ver Plan	Integración de la tarea ver plan	**FRONT, BACK, INTEGRACIÓN**	MVP1
Añadir URL de FOTO	Modificación del back-end para añadir la posibilidad de cargar imágenes desde un enlace	**BACK**	MVP1
Obtener Amigos de un Usuario	Operaciones internas en el back-end para obtener los diferentes amigos de un usuario	**USUARIOS, BACK**	MVP1
Mostrar Amigos de un Usuario	Representación de los amigos de un usuario en el front-end	**USUARIOS, FRONT**	MVP1
Agregar Amigo	Creación de las operaciones necesarias para agregar amigos	**USUARIOS, BACK**	MVP1

Tarea	Descripción	Etiquetas	Fase
Crear Agregar Amigo	Creación del caso de uso de agregar amigo	**USUARIOS, FRONT**	MVP1
Establecer Sistema de Amigos	Integración de las diferentes tareas de amistad creadas previamente	**INTEGRACIÓN, FRONT, BACK**	MVP1
Refactorizar Ver Plan para Mostrar Correctamente los Datos del Creador	Se modificará el diseño de la interfaz de planes para otorgar una interfaz más limpia e intuitiva.	**PLANES, CALIDAD, FRONT**	MVP1
Crear Menu Toolbar	Creación del menú superior de la aplicación con el fin de otorgar una mayor consistencia	**CALIDAD, FRONT**	MVP1
Establecer Petición de Planes Recomendados	Se creará la petición, así como la actividad que contendrán en un futuro los planes recomendados.	**PLANES, FRONT, BACK, INTEGRACIÓN**	MVP1
Implementar Búsqueda	Se implementará el caso de uso que permitirá al usuario buscar entre los diferentes planes	**PLANES, FRONT, BACK, INTEGRACIÓN**	MVP1
Mejora Estética de la Aplicación	Se mejora la estética de la aplicación con el fin de mejorar la experiencia de usuario	**FRONT**	MVP2
Algoritmo de Recomendación Backend	Preparar la infraestructura del back-end de cara a proveer un algoritmo de recomendación	**RECOMENDACIÓN, BACK**	MVP2
Implementar Registro	Implementación del caso de uso registrarse	**USUARIOS, FRONT**	MVP2
Ver Perfil	Implementación del caso de uso ver perfil y editar perfil	**USUARIOS, FRONT**	MVP2
Definir Matriz de Planes/Lemas para facilitar la recomendación	Implementación y definición de una matriz que relacione los planes con sus palabras claves para generar recomendación	**RECOMENDACIÓN, BACK**	MVP2

Tarea	Descripción	Etiquetas	Fase
Apuntarse a Planes	Implementación de la posibilidad de apuntarse a los diferentes planes	**PLANES, FRONT, BACK, INTEGRACIÓN**	MVP2
Editar Plan	Implementación del caso de uso editar plan	**PLANES, INTEGRACIÓN**	MVP2
Crear Diario de Planes	Implementación del caso de uso ver diario de planes	**PLANES, INTEGRACIÓN**	FUTURO
Menú Lateral	Implementación de un menú lateral para mejorar la visibilidad	**FRONT, CALIDAD, MODELO**	FUTURO
Documentar Código	Documentación de todo el código desarrollado en el proyecto	**DOCUMENTACIÓN**	FUTURO

De este modo, obtenemos un *backlog* inicial con **46** tareas, de las cuáles podríamos decir que 2 de ellas serían *extras,* aquellas referentes a la refactorización. Es necesario mostrarlas en esta tabla debido a que han sido un eje clave a lo largo de este proyecto y se consideró desde un inicio la posibilidad de tener que realizar estos cambios.

10.4. PRIORIZACIÓN DE TAREAS

Debido a que este es un proyecto enmarcado en un trabajo de fin de máster, es necesario ser consciente de las diferentes **restricciones temporales que esto conlleva.** Por esta misma razón es necesario establecer un sistema de priorización de tareas de cara a entregar el mayor posible a lo largo del tiempo de desarrollo de las diferentes versiones del producto.
Además, de cara a priorizar las tareas es necesario lograr, tal y como se ha mencionado en otros apartados alcanzar la mayor sinergia entre el producto resultante y aquellas características que puedan ser relevantes de cara a un trabajo de fin de máster.
Es por ello que se ha definido un sistema de priorización de tareas que se ha reflejado a lo largo del desarrollo de los MVP:

- **Urgente**. Aquella tarea que es bloqueante o imprescindible para la validación del producto o el desarrollo de otras características críticas.
- **Importante**. Aquella tarea que debe ser desarrollada al largo plazo, previo a la finalización del marco del proyecto, pero que no ha de ser implementada inmediatamente.
- **Normal.** Aquella tarea que ha de ser desarrollada en algún momento, pero no es crítica a la hora del uso de la aplicación.
- **Baja.** Aquella tarea que se encuentra posicionada en el *backlog*, pero que, en el caso de no desarrollarse, el producto continuaría teniendo un funcionamiento completo.

Partiendo con este sistema de priorización, se procederán a desarrollar las diversas iteraciones sobre el producto, de acorde a la metodología escogida en los apartados iniciales.

11. DESARROLLO DEL PRODUCTO

En este apartado se desglosan las diferentes iteraciones sobre el desarrollo del producto, el aprendizaje respecto a cada iteración y las decisiones tomadas en función de este aprendizaje.
Para ello se trata cada una de las fases mencionadas en el apartado de mapa de características como una fase única sobre la que se toman diversas iteraciones, es decir, se ha comenzado en la fase de ***Prueba de Concepto*** y se termina en el ***MVP2***.
De cara a estructurar este apartado, cada fase tendrá los siguientes apartados:

- Justificación
- Tareas Implementadas
- *MockUps*[10]
- Dificultades y desafíos
- Problemas
- Resultados

11.1. PRUEBA DE CONCEPTO

En este apartado se desglosa el desarrollo de la prueba de concepto.

11.1.1. JUSTIFICACIÓN DE PRUEBA DE CONCEPTO

Debido a que este proyecto trata de crear una aplicación comercial, es necesario realizar una prueba de concepto del producto para ver que éste es, efectivamente viable. Para ello, en lugar de realizar un *MVP* con toda la infraestructura necesaria se ha simplificado radicalmente el desarrollo de éste en una prueba de concepto que permite a los diferentes usuarios el crear sus planes y borrarlos, de cara a gestionarlos individualmente.
También se ha empleado esta prueba de concepto para ver las limitaciones de la plataforma móvil escogida, ***Android*** y así lograr una mayor calidad de desarrollo.

11.1.2. TAREAS IMPLEMENTADAS

En esta fase del proyecto, se trató de elaborar el mínimo número de tareas correspondientes a los planes, así como a su persistencia de forma local en el sistema, tal y como se puede ver en el *backlog.*
Es por ello, que finalmente han implementado las diferentes tareas del sistema:

[10] Es una maqueta del diseño de una aplicación a tamaño completo.

TABLA 15. TAREAS IMPLEMENTADAS DURANTE LA FASE PRUEBA DE CONCEPTO. ELABORACIÓN PROPIA

Tarea	Implementada
Toma de contacto con Android Front-end	✓
Django Back-end	✓
Pulir Requisitos y Casos de Uso	✓
Integrar API GraphQL	**Movida a MVP1**
Solicitar Licencia MagicDraw	✓
Dashboard View Model	✓
Definir Modelo Conceptual	✓
Implementar Navegación	✓
Estructura de Proyecto	✓
Estructura de Carpetas	✓ **Modificada en MVP1**
Crear Plan	✓
Feed de Planes	✓
Borrar Plan	✓
Ver Plan	✓
Implementar Login	**Movida a MVP1**
Implementar Modelo Conceptual	✓
Persistir Datos en App Android	✓
Persistir Planes	✓
Añadir Imágenes a Planes	✓
Mostrar Planes Ordenados	✓

Tal y como se puede ver, se aplazaron alguna de las tareas al MVP1. Esto se debe a la necesidad de mover estas tareas al existir dificultades técnicas y temporales. También se encontraron diversas dificultades de cara a establecer la integración debido a la estructura del proyecto.

11.1.3. MOCKUPS

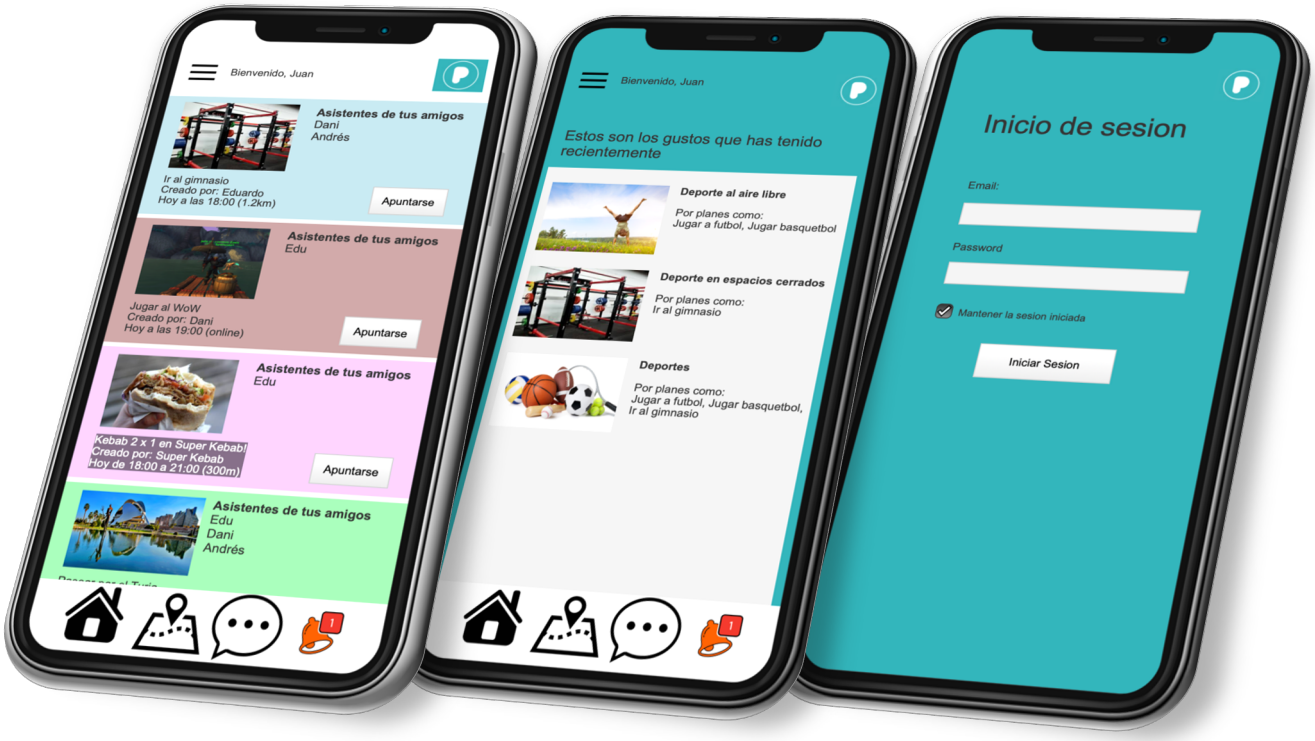


ILUSTRACIÓN 15. MOCKUPS EMPLEADOS DURANTE LA PRUEBA DE CONCEPTO. ELABORACIÓN PROPIA

En este apartado se pueden ver los diferentes *MockUps* en los que se ha basado el diseño de la prueba de concepto, así como los diferentes apartados contenidos en ésta.
En esta fase, existe la peculiaridad de que se elaboraron en una etapa muy temprana del proyecto, es por ello que resultan muy diferentes a los resultados obtenidos.
Tal y como se puede ver en los diferentes *MockUps*, el concepto inicial comprendía un sistema de ***feed de planes***, así como la posibilidad **de apuntarse a estos** y conocer la distancia y quiénes iban de tus amigos.
También, al estar realizado en una etapa tan temprana, se previó el uso de un mapa de planes, así como un chat, que fueron descartados de las primeras fases de desarrollo al percatarse de que supondrían un gran riesgo de cara al desarrollo y las limitaciones temporales de este proyecto.

11.1.4. DIFICULTADES Y DESAFÍOS

En este apartado se enumeran las diferentes dificultades y desafíos que se han encontrado a lo largo del desarrollo de la prueba de concepto de esta aplicación.

- Desconocimiento del desarrollo *Android* a nivel profesional. Se detecta la necesidad de comprender la documentación oficial [27] a una alta velocidad.
- Necesidad de integrar todos los aspectos del ciclo de vida de desarrollo de software en una única aplicación. Es decir, desde el apartado en el que se crea una tarea hasta que ésta resulta completada.
- Establecer las bases respecto a una conexión a un *back-end* desde dispositivos móviles. Cabe decir que este enfoque dista radicalmente del comúnmente usado en el desarrollo web, ya que es necesario tener en cuenta factores como consumo de batería y evitar el bloqueo del dispositivo.
- *Java* [28] como lenguaje de desarrollo de *Android* genera muchas complicaciones que afectaban a la velocidad del desarrollo.
- Especial complejidad a la hora de diseñar un sistema *Android* desde 0, esto se debe a que la arquitectura y el diseño nativo de esta aplicación requiere de la comprensión de patrones arquitectónicos avanzados.
- Integración de una base de datos **interna** en el sistema de la aplicación que permite cargar el estado de todos los planes.

Tal y como se puede ver, gran parte de los desafíos durante esta fase son referentes al desconocimiento y las diferentes tareas complejas como una debida integración.

11.1.5. PROBLEMAS

En este apartado se enumeran los diferentes problemas encontrados a la hora de desarrollar esta primera prueba de concepto.

- Elevada complejidad de desarrollo de código en *Java*. Esto se debe a que gran parte de las **librerías** necesarias para desarrollar en este lenguaje, estaban **obsoletas** o no están actualizadas a una última versión que permitiese desarrollar fácilmente.
- Por otro lado, los dispositivos Android requieren de una gran gestión de las *corrutinas de procesamiento*[11], siendo Java un lenguaje de nuevo limitante para la gestión de estas, requiriendo el desarrollo de código extra y dificultando su mantenibilidad.
- Por último, se detecta una gran cantidad de errores de sistema en la aplicación debido a la integración todo lo mencionado anteriormente.

Como se puede observar, todos los problemas encontrados durante esta fase fueron relacionados con el lenguaje de desarrollo. Es por ello, que tal y como se puede observar en el *backlog*, se decide reestructurar el código de la aplicación al lenguaje llamado ***Kotlin*** [29]**.**
Por último, hay que destacar que los conocimientos previos en Java por parte del equipo de desarrollo son avanzados, por lo cual se toma esta decisión debido a la aportación que ofrece ***Kotlin*** de cara al *multiprocesado* esencial en dispositivos Android, así como las diferentes facilidades de diseño que éste ofrece de cara a implementar las diferentes actividades en el sistema.

[11] Una corrutina es un proceso ligero que ejecuta funcionalidades de forma paralela, evitando el colapso del sistema cuando se trabajan en procesos delicados o que requieran de un tiempo de espera.

11.1.6. RESULTADOS

Al finalizar la fase de desarrollo, se obtuvo un producto que permite a los diferentes usuarios el crear planes, borrarlos y gestionarlos a nivel local, así como *mantener la gestión local de sus planes*.
Cabe decir que, debido a los inconvenientes de desarrollo, no ha sido posible el integrar el sistema con el *back-end* pese a que este estuviese parcialmente desarrollado.
A continuación se muestran algunas capturas de pantalla de la aplicación resultante:

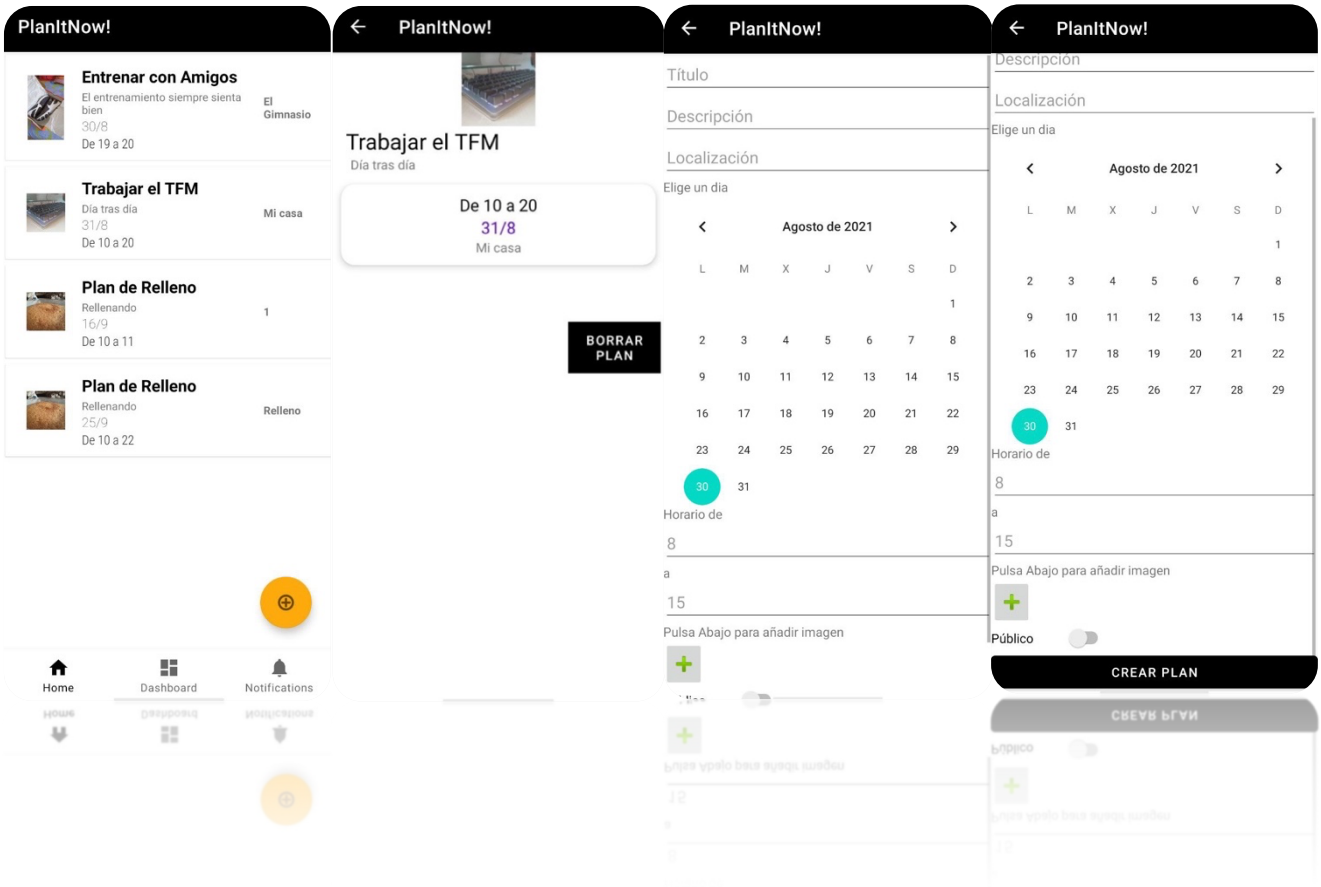


ILUSTRACIÓN 16. CAPTURAS DE PANTALLA DE PRUEBA DE CONCEPTO. ELABORACIÓN PROPIA

En estas ilustraciones podemos ver de izquierda a derecha la implementación de las siguientes características:

- **Feed de planes** ordenado. Nos permite ver cuándo tenemos cada plan, así como una descripción.
- **Visualización** en profundidad de un plan. Esto nos permite ver atributos que inicialmente no están tan a la vista en el *feed*, también se nos permite borrar el plan en el caso de que ya no deseemos verlo.
- **Creación de planes** para poder añadir nuevos planes a nuestra plataforma y guardarlos en una base de datos, con el fin de poder usar esta aplicación indefinidamente.
- **Navegación básica** que nos permite acceder fácilmente a las diferentes funcionalidades del sistema.
- **Dashboard** que permite trabajar con diferentes fragmentos de Android para ofrecer diferentes vistas dentro de una misma actividad.

Esto permite crear una primera prueba de concepto para ver qué es posible añadir en las siguientes fases. También se establece una primera toma de contacto con el desarrollo *Android*.
De este primer producto, se observa la importancia y dificultad de ser conectado a una base de datos externa, ya que a nivel de desarrollo resulta muy costoso almacenar toda la información con un servidor externo y que ambos estén sincronizados. También se **pivota** el lenguaje de desarrollo del producto, tal y como se ha mencionado a ***Kotlin***.

11.2. PRIMER MVP

En este apartado se desglosa todo lo desarrollado durante el primer *MVP*. En este caso, ya se cuenta con un *back-end* establecido, de modo que todas las características implementadas cuentan con el respaldo tecnológico y almacenamiento en una base de datos. Además, este MVP ya cuenta con características funcionales que se desarrollaron con el fin de poder ser probado en usuarios finales.
En este MVP, podemos ver que se desarrollan características principales, y, como bien se ha mencionado antes, se refactoriza todo el código de Java a *Kotlin*, permitiendo desarrollar a mayor velocidad e integrar el *back-end* dentro de la aplicación móvil. El *back-end* inicialmente se prueba de forma local durante toda esta fase del proyecto.

11.2.1. TAREAS IMPLEMENTADAS

En el apartado anterior de prueba de concepto, podemos ver que muchas de las tareas establecidas inicialmente se posponen a este *MVP*, tal y como se dice anteriormente, debido a su facilidad para integrar con el sistema back-end, sin embargo, se pierde la base de datos local y toda la información contenida en la aplicación figura en un servidor externo a la aplicación.
A continuación, se muestra una tabla con todas las tareas implementadas durante esta fase:

TABLA 16. TAREAS IMPLEMENTADAS DURANTE EL DESARROLLO DEL MVP1. ELABORACIÓN PROPIA

Tarea	Implementada
Integrar API GraphQL	✓
Implementar Navegación	✓
Estructura de Proyecto	✓
Estructura de Carpetas	✓
Visualizar Planes Contenidos en el Back	✓
Crear Planes en el Back	✓
Borrar Planes en el Back	✓
Refactorizar toda la App a Kotlin	✓
Añadir Validación a las consultas y mutaciones importantes	✓
Ver Plan	✓
Añadir URL de FOTO	✓
Obtener Amigos de un Usuario	✓
Mostrar Amigos de un Usuario	✓
Agregar Amigo	✓
Crear Agregar Amigo	✓
Establecer Sistema de Amigos	✓
Refactorizar Ver Plan para Mostrar Correctamente los Datos del Creador	✓
Crear Menu Toolbar	✓
Establecer Petición de Planes Recomendados	✓
Implementar Búsqueda	✓

Tal y como se puede apreciar, en esta fase se desarrollaron las diferentes tareas de refactorización hacia proveer de una aplicación completamente integrada con el *back-end*, siendo las primeras tareas aquellas referentes a la integración del proyecto con un servidor externo.
En este primer *MVP* se implementó el *login* pero no el registro, ya que se pretendían hacer pruebas con grupos cerrados de usuarios. Sin embargo, como se puede observar, se despliega al iniciar el proyecto, de este modo, el realizar pruebas en un entorno real ha sido más sencillo.
Por otro lado, pese a que la fase de implementación de este *MVP* fuese muy diferente al anterior al cambiar de lenguaje, el desarrollo fue satisfactorio debido al previo conocimiento del sistema ***Android***, así como el uso del lenguaje al que se decidió pivotar, ***Kotlin***. Esto resulta en que todas las tareas propuestas se hayan completado con éxito.

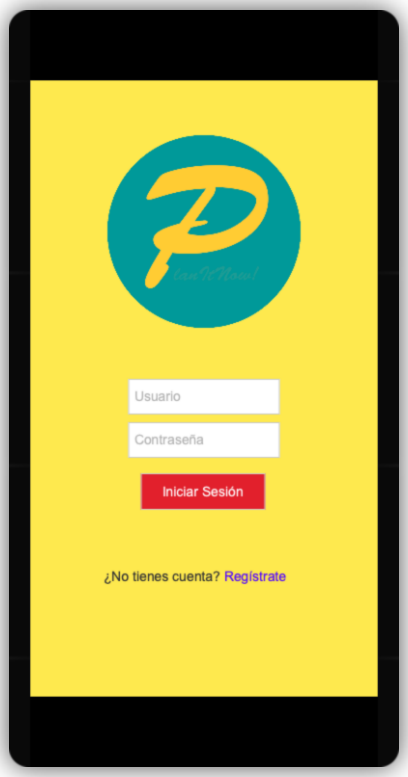
11.2.2. MOCKUPS

Tal y como se puede observar en la fase anterior, los *MockUps* de la prueba de concepto distaban radicalmente de lo que se va a encontrar a lo largo de los que se muestran a continuación. Esto se debe a que, tras la fase previa, se adquirieron los conocimientos suficientes respecto al **desarrollo en sistemas *Android*** como para establecer mejores diseños de interfaz y unos más acordes con lo que realmente se puede desarrollar en este sistema.

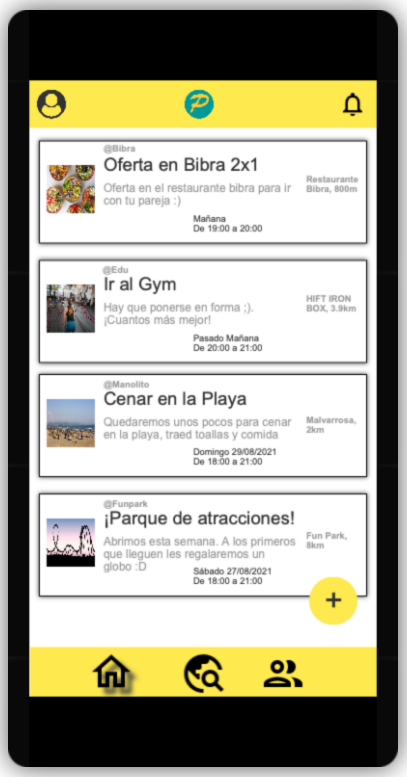
Además, se conocen ya de mejor manera los datos que son necesarios de mostrar e implementar, con lo que estos *MockUps* son mucho más parecidos a los resultados finales.

TABLA 17. MOCKUPS MVP1. ELABORACIÓN PROPIA

Iniciar sesión

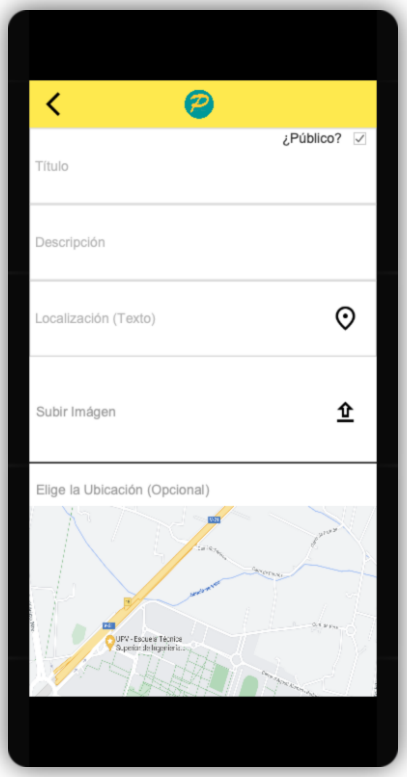


Feed de planes



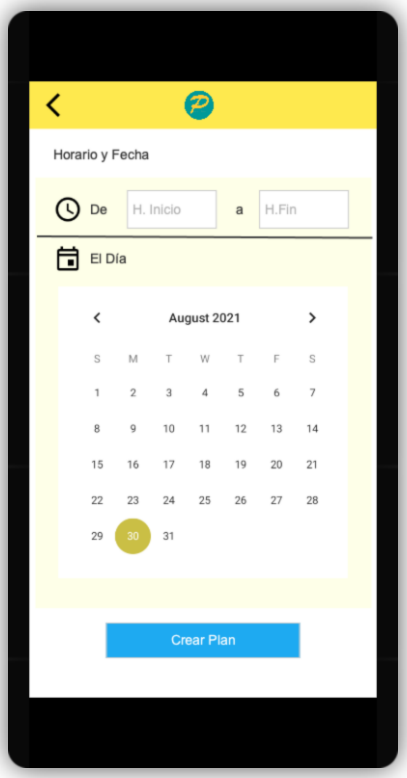
Ver Plan

Crear Plan 1/2



Buscar Plan

Crear Plan 2/2



Ver Amigos

Añadir Amigo

Ver Notificaciones

Tal y como se puede observar, estos *MockUps* mejoran sustancialmente el diseño respecto a la fase anterior, así como que ahora respetan pautas de diseño como el espaciado, el minimalismo y así como un diseño más centrado en el sistema operativo *Android*.

También se ha establecido una paleta de colores amarillo y azul basado en los colores de *Material Design* [30], así como la posibilidad de ver el logo de la aplicación en este proyecto.

Por último, podemos ver que se emplea la barra de acciones de Android como un elemento clave para la navegación.

11.2.3. DIFICULTADES Y DESAFÍOS

En este apartado se enumeran las diferentes dificultades y desafíos encontrados durante el desarrollo de este *MVP*. Cabe decir que esta fase de desarrollo ha sido más precisa, ya que en las fases de diseño se habían establecido los diferentes diseños de la aplicación con mayor precisión.

Entre las dificultades y desafíos podemos encontrar los siguientes:

- La **integración** de un sistema ya existente con un *back-end* establecido. Esto resulta en un gran desafío pues es necesario comprender ampliamente cómo funcionan ambas partes del sistema, así como establecer canales de datos que permitan el correcto intercambio de información.

- La **gestión de sesiones de usuarios** ha supuesto una gran dificultad debido a que ha sido necesario gestionar por un lado que el *back-end* sea consciente de qué usuario hay conectado en cada momento, así como qué, y por otro lado, en el *front-end* ha sido necesario implementar diversos atributos sobre el usuario que tiene su sesión iniciada. Además, ha sido necesario asegurarse de que ambos sistemas funcionen correctamente y libres de fallos.
- A nivel de **sistema**, ha sido necesario optimizar las diferentes actividades de la aplicación para evitar bloqueos de sistema cuando esta se ejecuta.
- El **despliegue del proyecto** supone un reto total, pues, por un lado, desplegar una aplicación es una tarea compleja, y, por otro lado, ha sido necesario verificar que la aplicación funciona correctamente una vez esta es desplegada.
- El paso de código ***Java*** a código ***Kotlin*** ha supuesto un gran desafío, al tener que reestructurar todo el código ya existente en el sistema y adaptarlo a esta nueva versión.

11.2.4. PROBLEMAS

En este apartado se enumeran los diferentes problemas que se han detectado a lo largo del desarrollo de este **primer *MVP***.
A lo largo de esta fase no ha habido grandes problemas ya que se partía de una base previa en la que se había creado la estructura de la aplicación. Sin embargo, algunos de los problemas encontrados durante esta fase son:

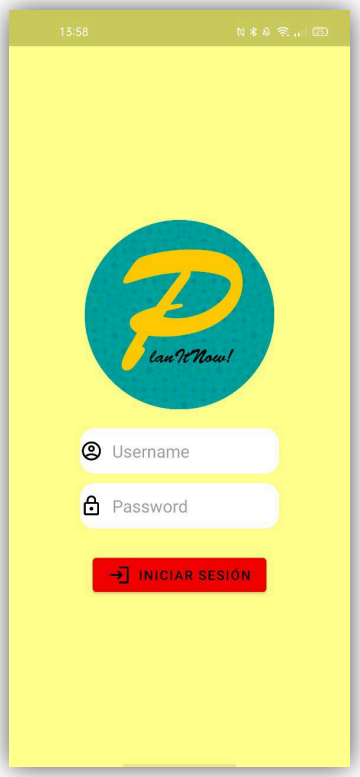
- La librería de desarrollo escogida para realizar la **integración** del sistema con el *back-end* no implementaba algunas características esenciales para el uso de la aplicación, por lo que fue necesario actualizarla durante el desarrollo, teniendo así que realizar significativos cambios en el código.
- Debido a la **nueva estructura de la aplicación**, no ha sido posible mantener el **almacenamiento interno del dispositivo** ya desarrollado previamente con lo que este trabajo realizado durante la fase anterior fue perdido.
- Ha sido necesario realizar más cambios de los esperados en el *back-end* para poder dar lugar a la funcionalidad completa de la aplicación.

11.2.5. RESULTADOS

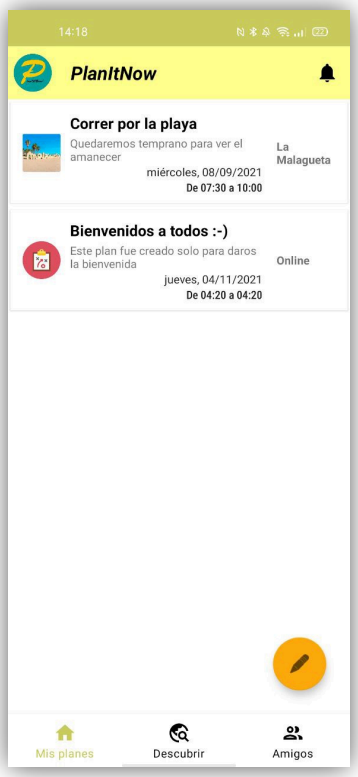
Durante esta fase se ha desarrollado una primera versión del producto aún por finalizar que corresponde a los *MockUps* mostrados anteriormente. Esta versión ya cuenta con la posibilidad de enlazar y gestionar los datos a través de un servidor externo, y, en el caso de que se desease rehacer la aplicación de forma completa, no se perderían datos debido a que se cuenta ya con un *back-end* bien establecido.
Todo el código del *back-end* ha sido desplegado, y tras finalizar esta fase se cuenta con la posibilidad de usar de forma completa la aplicación, crear y gestionar los planes, visualizar aquellos eventos públicos y añadir amigos. De este modo se concluye que el desarrollo de esta fase ha sido exitoso.
A continuación, se muestran capturas de pantalla de los resultados de esta fase:

TABLA 18. CAPTURAS DE PANTALLA MVP1. ELABORACIÓN PROPIA

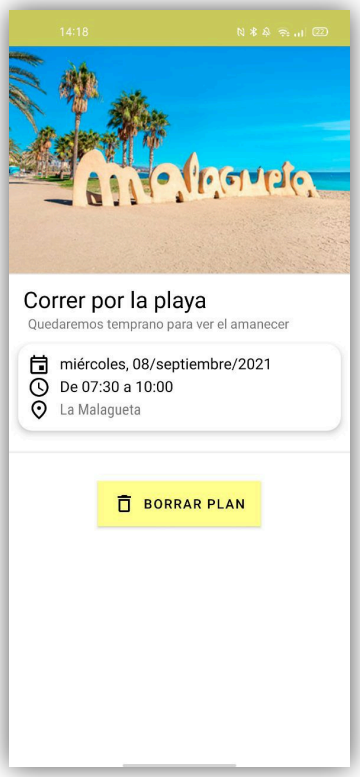
Iniciar Sesión



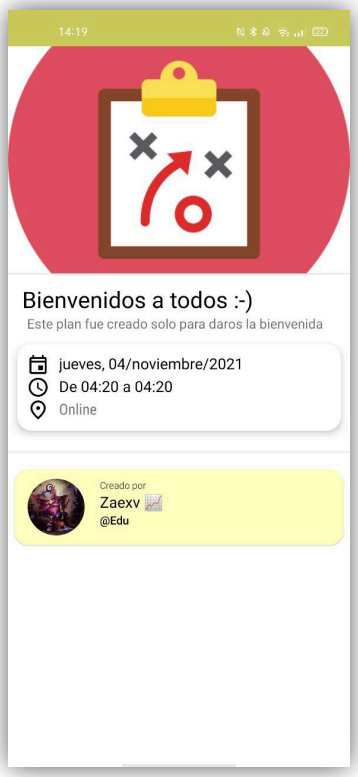
Feed de Planes



Ver Plan (Dueño)



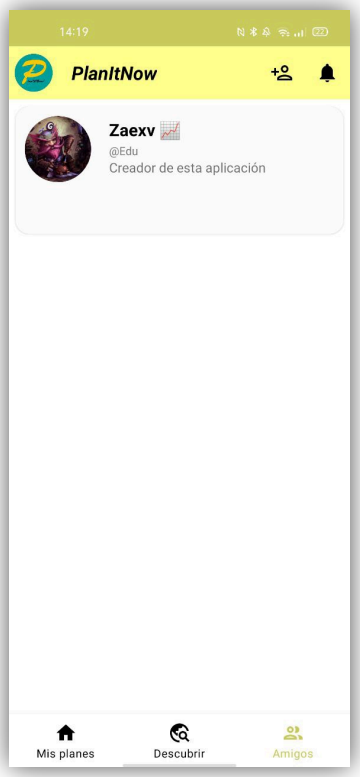
Ver Plan (Participante)



Crear Plan 1/2

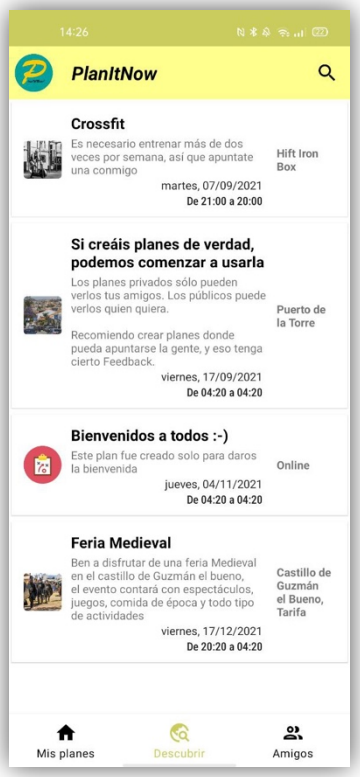
Crear Plan 2/2

Ver Amigos

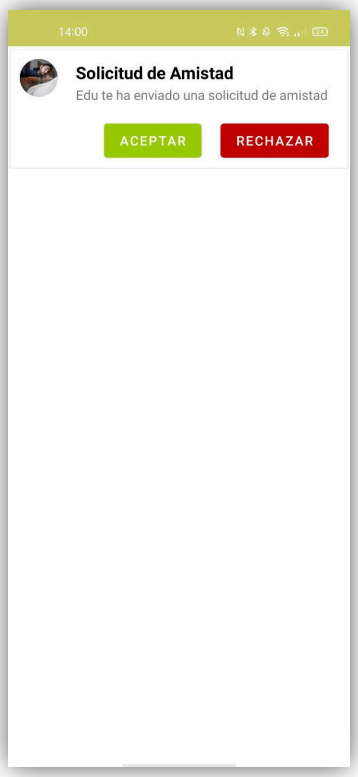


Añadir Amigo

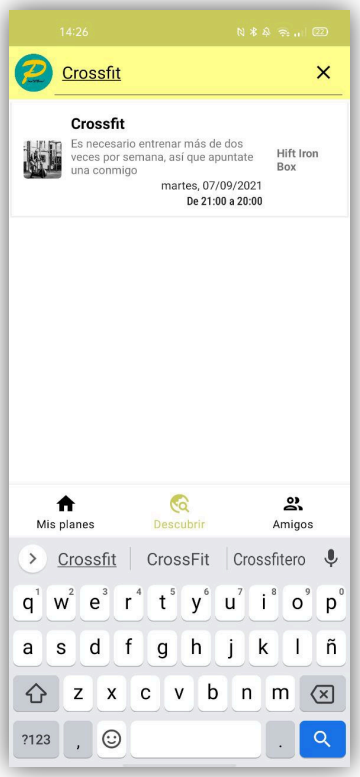
Buscar/Recomendación Plan Feed



Ver Notificaciones



Buscar Plan



En las capturas de pantalla de la tabla anterior, podemos ver que se han implementado diversas características que formalizan una aplicación más completa que la desarrollada en la fase anterior. Ahora la aplicación permite crear planes, eliminarlos y hacer amigos entre otras características. A continuación, se va a realizar una breve descripción de cada una de las características implementadas en este *MVP*:

- **Integración de Datos.** La aplicación almacena los datos en un servidor externo en lugar de almacenar estos datos de forma local. Esto supone un gran cambio del uso de la aplicación.
- **Inicio de sesión.** La aplicación permite a los diferentes usuarios iniciar sesión en el sistema y así poder tener su propia entidad dentro de esta.
- **Feed de Planes**. La aplicación permite visualizar qué planes son los que un usuario en concreto ha creado, así como los diferentes planes a los que está apuntado. Aunque en esta versión la aplicación no permite apuntarse a planes, sí que se ha elaborado la infraestructura necesaria para poder hacerlo.
- **Ver Plan.** La aplicación permite visualizar los planes creados, así como los que otros usuarios han creado. En cada escenario, la vista de visualización de plan cambia, ya que en el caso de que el usuario que ha iniciado sesión sea el creador, se le permitirá borrar el plan.
- **Crear Plan.** La aplicación permite crear planes públicos y privados, además se permite añadir el enlace a una imagen para que esta se muestre, así como características secundarias como la posibilidad de escoger la fecha en un calendario del sistema y escoger la hora a través del reloj implementado por el sistema.
- **Ver Amigos.** La aplicación permite visualizar todos los amigos que hemos hecho a través de la plataforma.

- **Añadir Amigo.** La aplicación permite añadir un nuevo amigo únicamente introduciendo su nombre de usuario.
- **Buscar y Feed de Recomendaciones.** La aplicación cuenta con un segundo *feed* que permite al usuario visualizar los planes que han creado sus amigos o son públicos. Además, se ha implementado la búsqueda de planes a través de este menú. Cabe destacar que se llama *feed* de recomendación pues la infraestructura interna se ha preparado para mostrar la recomendación en la siguiente versión.
- **Ver Notificaciones.** Se permite al usuario ver las solicitudes de amistad en la vista de notificaciones. Además, se han implementado también las acciones de aceptar y rechazar solicitud.
- **Despliegue del Proyecto.** El proyecto se despliega en un servidor web que permite la conexión de múltiples usuarios de forma simultánea.

Gracias a la implementación de las características mencionadas se obtiene una aplicación completamente funcional en la que se pueden registrar diversos usuarios. Durante esta fase se han registrado algunos *early adopters* manualmente.
El resultado de esta fase ha sido más que satisfactorio pues se obtiene la primera versión completamente funcional de ***PlanItNow!***, además de la preparación del proyecto para la siguiente fase.

11.3. SEGUNDO MVP

En este apartado se desglosa todo lo desarrollado durante el segundo *MVP*.
Durante esta fase, ya se cuenta con un **primer *MVP*** totalmente funcional, de modo que esta fase se centra en ampliar y pulir las características ya existentes, haciendo especial énfasis en el **algoritmo de recomendación y apuntarse a los planes.** Además, debido a que durante esta fase se pretende lanzar la aplicación a un mayor número de usuarios, también se establece la posibilidad de que estos se registren.
En este *MVP* también se han realizado mejoras sobre las características implementados en el anterior, así como de diseño. Gracias a haber hecho especial énfasis en el diseño de la aplicación, cabe decir que en esta fase ya se tiene completamente desacoplado el funcionamiento de la aplicación *front-end* respecto al *back-end* de tal modo que únicamente cambiando el comportamiento del *back-end*, cambiará también en el *front-end*, pudiendo realizar cambios dentro del sistema sin que el usuario final se percate de estos.

11.3.1. TAREAS IMPLEMENTADAS

Durante la fase anterior, se crearon gran parte de las características funcionales de la aplicación y se establecieron las bases sobre el diseño del proyecto. Es por ello que, durante esta fase, se hace especial énfasis en lograr una versión *más pulida* del sistema.
Cabe decir que la mayor parte del *back-end* fue implementada en la fase previa, de tal modo que durante esta fase únicamente ha sido necesario realizar varias modificaciones y adaptarlo a la demanda de los usuarios. Por otro lado, el **algoritmo de recomendación** se ha desarrollado de forma completamente independiente a lo que se puede visualizar en el proyecto ya que se pretende que este actúe tal y como se ha mencionado en otros apartados de manera no intrusiva.

TABLA 19. TAREAS IMPLEMENTADAS DURANTE EL DESARROLLO DEL MVP2. ELABORACIÓN PROPIA

Tarea	Implementada
Mejora Estética de la Aplicación	✓
Algoritmo de Recomendación Back-end	✓
Implementar Registro	✓
Ver Perfil	✓
Definir Matriz de Planes/Lemas para facilitar la recomendación	✓
Apuntarse a Planes	✓
Editar Plan	✓
Crear Diario de Planes	✓
Menú Lateral	FUTURO
Documentar Código	FUTURO

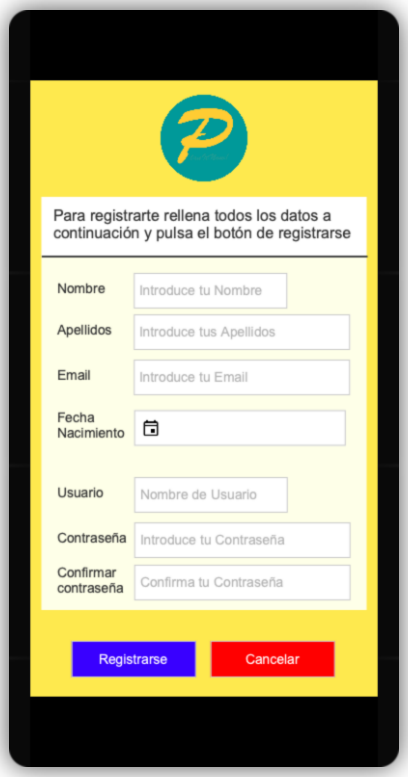
Tal y como se puede apreciar, en esta fase se desarrollan algunas nuevas características, especialmente aquellas referentes a pulir y completar la aplicación. Cabe decir que buena parte del desarrollo durante esta fase consistió en mejorar el rendimiento de la aplicación y otras tareas no funcionales, de modo que carecen de interés de cara a mostrarlas como tareas para el *MVP*, sin embargo, se pueden ver reflejadas en los apartados de aspectos técnicos.

Respecto a las tareas completadas, se puede deducir que esta fase resulta exitosa puesto que logran completarse todas las tareas propuestas para el *MVP2*. Sin embargo, sí que se mantienen en el futuro las tareas que se realizan con esta proyección. Esto se debe al marco de tiempo en el que se desarrolla el proyecto y las prioridades establecidas durante el mismo.

11.3.2. MOCKUPS

Durante esta fase, se emplean los mismos *MockUps* que durante el primer *MVP* con la diferencia de que durante esta fase se añaden nuevos casos relacionados con las nuevas características. Por esta misma razón, durante esta sección se van a mostrar únicamente aquellos que resulten diferentes o implementen nuevas características del producto.

TABLA 20. MOCKUPS SEGUNDO MVP. ELABORACIÓN PROPIA

Registro	Ver / Editar Perfil
	

Tal y como se puede observar, únicamente se han ampliado los MockUps para permitir al usuario registrarse, editar y visualizar perfil. En el caso de **Ver / Editar perfil**, este formulario se rellena automáticamente con los datos del usuario para que pueda realizar una correcta edición y visualizar sus atributos.
Respecto a las tareas de **diario de planes** y **editar plan**, **se han reutilizado los *MockUps* del *MVP1 Feed de Planes*** y **Crear Plan** respectivamente, ya que se consideran tareas muy similares a nivel de interfaz, aunque puedan resultar diferentes a nivel lógico. Tal y como se ha mencionado en el apartado anterior, se ha trabajado para poder desacoplar al máximo la lógica del diseño de la aplicación, creando así un diseño sólido mejorando su usabilidad.

11.3.3. DIFICULTADES Y DESAFÍOS

A lo largo de este apartado se enumeran las dificultades y desafíos enfrentados durante el desarrollo de este segundo *MVP*. Aunque no se han implementado tantas características como en la fase anterior, sí que se han encontrado diversas dificultades, especialmente aquellas referentes al **algoritmo de recomendación,** pues la arquitectura del sistema, así como las herramientas de procesados han supuesto un gran esfuerzo de ser desarrolladas.

A continuación, se pueden encontrar las diferentes dificultades y desafíos de esta fase:

- **Integración del módulo de recomendaciones.** Ya que se pretende realizar un módulo de recomendaciones que no interfiera durante el uso de la aplicación, el integrar este módulo con los componentes ya establecidos ha supuesto un gran desafío tanto a nivel de base de datos como a nivel de código.
- **Traducción y procesado del texto de los diferentes planes.** Esto ha supuesto un desafío ya que debido a la infraestructura establecida para que el algoritmo funcione correctamente es necesario traducir el texto completo de los planes en primera instancia, así como el almacenarlo en la base de datos para su post procesado.
- **Obtención de *lemas.*** Esto ha supuesto una gran dificultad pues a partir del texto traducido previamente, ha sido necesario obtener los diferentes *lemas* de cada uno de los planes, así como establecer las correctas relaciones en la base de datos. La mayor dificultad en este aspecto ha sido la de verificar que los *lemas* son correctos y no añaden ningún tipo de ruido a la base de datos.
- **Algoritmo de Recomendación.** Una vez se han obtenido todos los datos para ejecutar el algoritmo de recomendación, este ha supuesto un gran desafío a la hora de ser desarrollado. No sólo porque provee recomendaciones a tiempo real, sino por factores secundarios como la automatización de esta tarea y la gestión de recursos que este algoritmo implica.
- **Optimización y Diseño de la aplicación.** Ha supuesto un completo desafío el lograr que la aplicación funcione correctamente y mejorar la usabilidad y el rendimiento respecto al primer *MVP*.

Tal y como se puede apreciar, la mayor parte de los desafíos corresponden al apartado de recomendaciones, ya que éste es el punto fuerte durante esta fase de desarrollo.

11.3.4. PROBLEMAS

A lo largo del desarrollo de este segundo *MVP* se han encontrado un menor número de problemas respecto a las fases anteriores pues gran parte de estos se han solucionado en las fases previas.
Sin embargo, se han encontrado los siguientes problemas:

- Necesidad de **reiniciar la base de datos desplegada**, debido a los cambios de modelo necesarios durante esta fase. Esto produce la pérdida completa de los datos del primer *MVP*. Este problema se debe a la corrupción de diferentes archivos de migración en la base de datos, así como la infraestructura del servidor externo.
- **Algoritmo de Recomendación** muy complejo. La complejidad del algoritmo de recomendación resulta muy elevada y ha ocupado gran parte del tiempo de desarrollo de este *MVP*, haciendo que determinadas tareas planeadas para un futuro u opcionales no hayan podido desarrollarse.

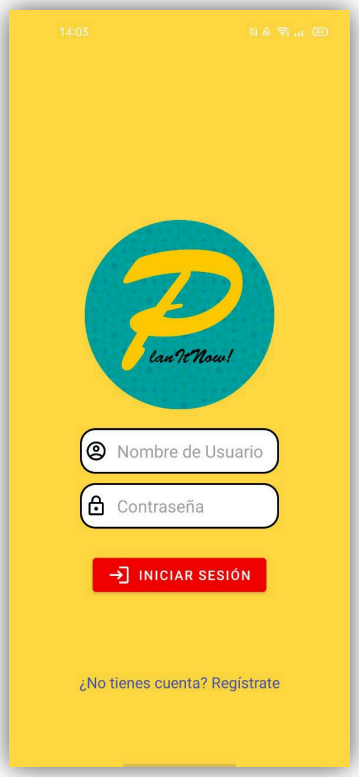
Tal y como se puede observar, únicamente existen dos problemas durante el desarrollo de esta fase. Respecto a la base de datos, afortunadamente no se perdió una gran cantidad de información ya que la información desplegada únicamente contenía el propósito de ser probada por usuarios finales.
Por otro lado, respecto al problema acontecido durante el algoritmo de recomendación se podría decir que se esperaba que esto ocurriese, de tal modo que no afectó al desarrollo de las funcionalidades de la aplicación.

11.3.5. RESULTADOS

Durante los apartados anteriores se puede observar que esta versión de la aplicación trata de mejorar lo desarrollado durante la primera fase, ampliar la aplicación con algunas nuevas características y principalmente, el recomendar los planes a los diferentes usuarios dentro de la pestaña de búsqueda.
Puesto que ya se cuenta con una estructura sólida del proyecto, se han podido desarrollar todas estas tareas con éxito, logrando una casi primera versión comercial del producto desarrollado.

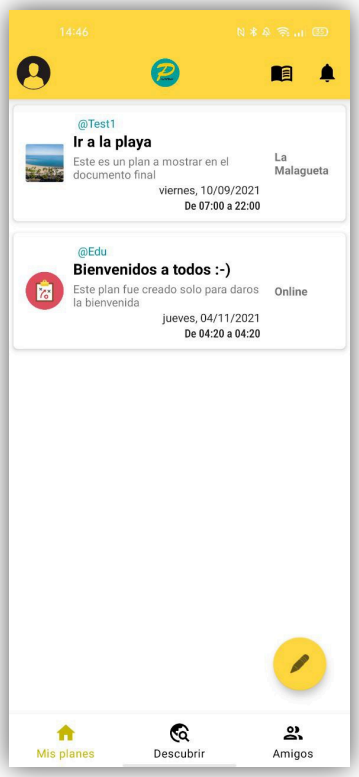
A continuación, se muestran capturas de pantalla de los resultados de esta fase:

TABLA 21. CAPTURAS DE PANTALLA DEL MVP2. ELABORACIÓN PROPIA

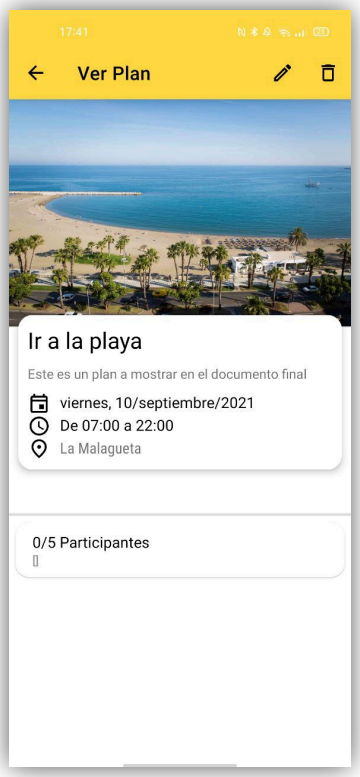
Pantalla de Carga	Login
	

Registro de Usuario

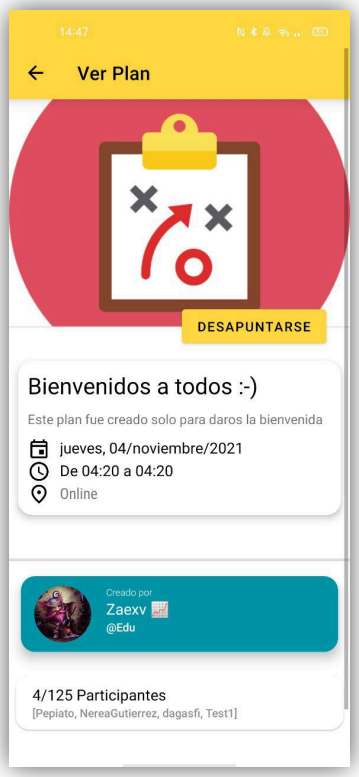
Feed de Planes



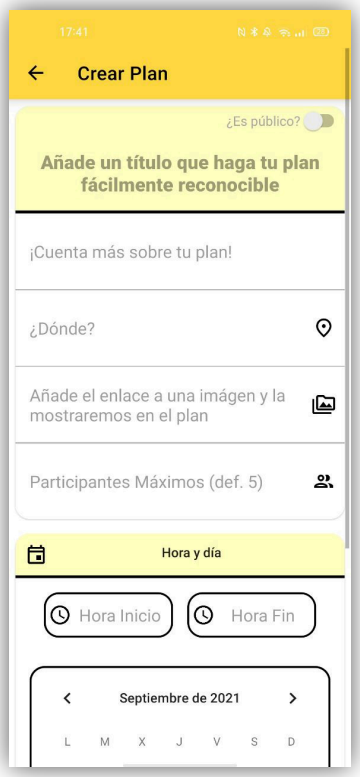
Ver Plan (Dueño)



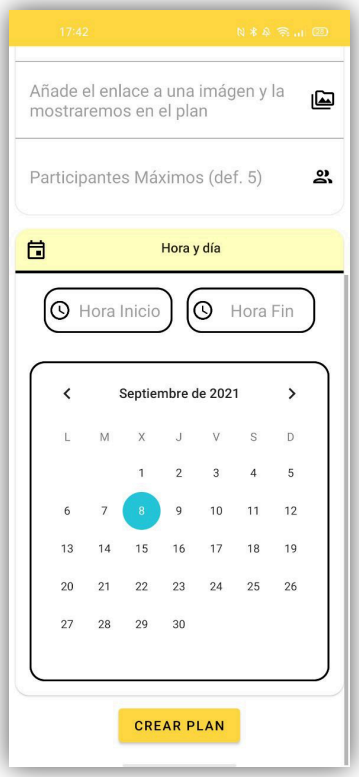
Ver Plan (Participante)



Crear Plan 1/2



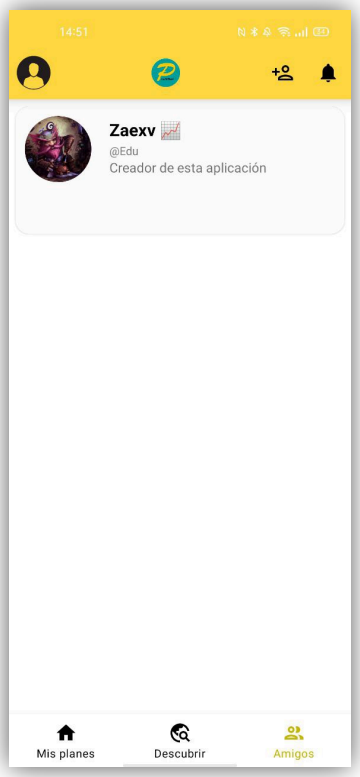
Crear Plan 2/2



Ver / Modificar Perfil

Ver Notificaciones

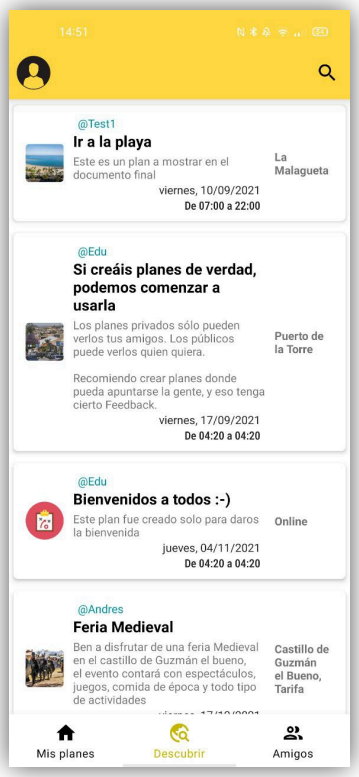
Ver Amigos



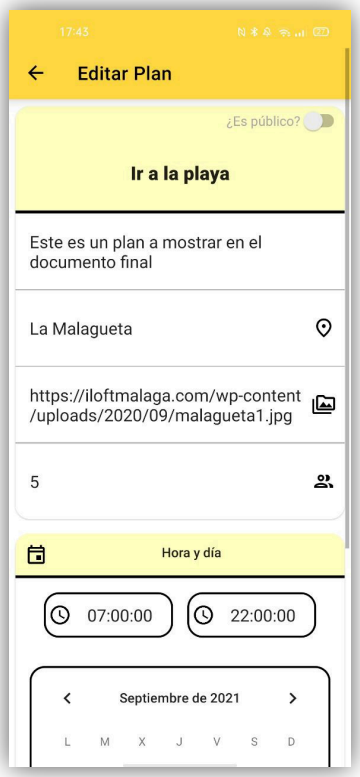
Añadir Amigos

Diario de Planes

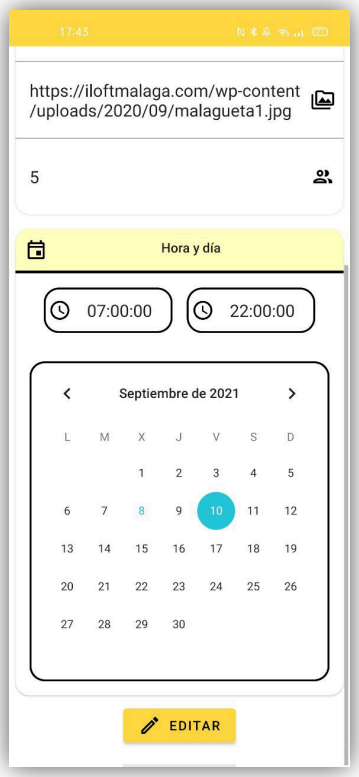
Búsqueda y Recomendación



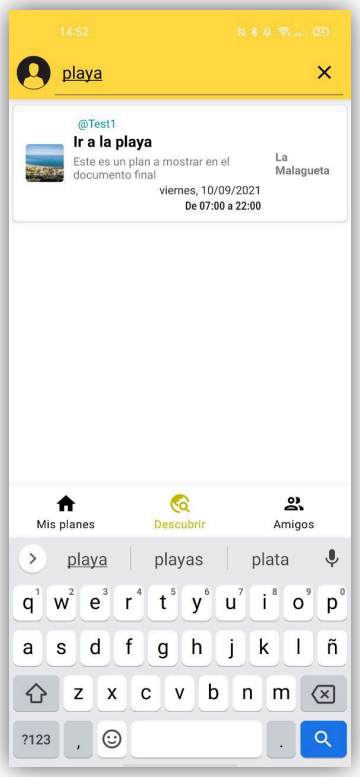
Editar Plan 1/2



Editar Plan 2/2



Buscar Plan



Tal y como se puede apreciar en las diferentes capturas, se ha rediseñado parte de la aplicación, especialmente aquella referente a la barra superior para otorgarle un mayor estilo y funcionalidad. Además, se han modificado los colores para mejorar su usabilidad y mejorado el diseño de las diferentes vistas de edición y creación.
A continuación, se va a realizar una breve descripción de las nuevas características funcionales desarrolladas durante este *MVP*:

- **Pantalla de carga a inicio automático de sesión.** Cuando se inicia la aplicación en lugar de mostrar el inicio de sesión se muestra una pantalla de carga que, en el caso de que exista un usuario que previamente haya iniciado sesión, automáticamente iniciará sesión con este usuario. En el caso de que se desee cambiar de usuario, será necesario cerrar sesión.
- **Editar y ver perfil.** La aplicación permite ver el perfil del usuario que ha iniciado sesión y modificarlo a través de la misma interfaz. Esto ayuda a tener una mayor personalización de los perfiles de usuario.
- **Registro de usuario.** La aplicación permite a los diferentes usuarios registrarse requiriéndoles diversos atributos y realizando una comprobación de que tanto la contraseña como los diferentes datos introducidos son correctos.
- **Apuntarse a Plan.** Ahora los usuarios pueden apuntarse a los planes de sus amigos, así como ver los diferentes usuarios que hay apuntados en los diferentes planes. Cuando un usuario se apunte a un plan, este plan le aparecerá automáticamente en la sección *Mis Planes*. Esta característica no sólo ayuda a saber quién participa en un plan, sino a los diferentes usuarios a gestionar aquellos planes en los que quieren participar.
- **Editar Plan.** La aplicación permite a los usuarios editar aquellos planes que hayan creado. Para esto, la aplicación carga la información del plan en el formulario y actualizará aquellos que el usuario haya decidido modificar.
- **Diario de planes.** Los planes que el usuario haya creado o en los que participe se almacenarán en esta vista una vez hayan caducado temporalmente. Esto permite a los usuarios qué planes realizaron en el pasado.
- **Navegación mejorada.** Ahora la aplicación permite al usuario navegar mediante la barra de acción, así como realizar las diferentes acciones pertinentes según la vista en la que se encuentre. Esto contribuye a un diseño más limpio, evitando botones de acción innecesarios.
- **Recomendación personalizada.** Gracias al algoritmo de recomendación que se ha implementado durante esta fase, el usuario puede ver en la sección de *Descubrir*, aquellos planes que el sistema le ha recomendado seguido por la lista de planes disponibles dentro de la aplicación.

Tal y como se puede ver, esta versión trae nuevas características, aunque gran parte del esfuerzo se ha centrado en pulir algunas de las ya existentes en la versión anterior.
Se puede concluir que los resultados de esta fase de desarrollo han sido exitosos debido a que se han obtenido los resultados esperados para un segundo *MVP* de la aplicación.

12. EXPERIMENTOS CON USUARIOS

Tal y como se menciona durante la sección de desarrollo, el proyecto fue desplegado desde la realización del primer *MVP* en un servidor externo. Esto ha permitido el realizar diferentes experimentos con los *early adopters* de cara a validar el producto, así como la idea de negocio.

La **metodología de este experimento** ha consistido en lanzar la aplicación y permitir a diferentes conjuntos de usuarios usarla con todas las funcionalidades, para que éstos finalmente respondan a una encuesta donde deben evaluar cada una de las características del producto, decidir sobre si pagarían o no pagarían por las funcionalidades extras mencionadas durante los apartados de modelo de negocio y sugerir o criticar las diferentes funcionalidades de la aplicación.

12.1. RESULTADOS PRIMER MVP

Los resultados del primer *MVP* mostraron una satisfacción general por parte de los usuarios.

El **100% de los encuestados** afirmaron que les gustaría que existiese una aplicación para poder hacer planes con amigos.

El **80% de los encuestados** dijeron que usarían su aplicación para realizar sus planes y quedadas, mientras que el **20%** de los encuestados dijeron que tal vez la usarían con este fin.

Por otro lado, respecto al modelo de negocio el **60%** de los encuestados dijeron que tal vez pagarían por promocionar sus planes o modificar su ubicación, el **20%** dijeron que no pagarían mientras que un **20%** afirmaron que sí que pagarían por promocionarse en una aplicación de esta envergadura.

Por último, se les hizo evaluar las diferentes vistas de la aplicación resultante del 1 al 5. La mayoría de los usuarios marcaron valores entre el 4 y 5 respecto a la satisfacción de las diferentes vistas salvo en la vista de **descubrir,** la cual obtuvo un promedio de 3 sobre 5.

Además, se permitió a los usuarios proponer mejoras para la aplicación, las más relevantes fueron **la necesidad de poder añadir fotos desde el dispositivo** y una opción de **restablecimiento de contraseña.**

12.2. RESULTADOS SEGUNDO MVP

Los resultados del segundo *MVP* mostraron una mayor satisfacción general por parte de los usuarios al haber mejorado sustancialmente el diseño de las interfaces.

El **100% de los encuestados** afirmaron que les gustaría que existiese una aplicación para poder hacer planes con amigos.

De nuevo, el **80% de los encuestados** dijeron que usarían su aplicación para realizar sus planes y quedadas, mientras que el **20%** de los encuestados dijeron que tal vez la usarían con este fin.

El **100% de usuarios** se mostró satisfecho con la opción de poder apuntarse a planes.

El **100% de usuarios** indicó que le gustaría contar con una aplicación donde se recomienden planes en función de los gustos del usuario o a los que vayan sus amigos.

Respecto al modelo de negocio, en este caso el **50% de los usuarios** marcó que sí que pagarían por las funcionalidades extras.

Por último, respecto a la evaluación de las vistas en este caso todos los usuarios marcaron su satisfacción entre **4 y 5** debido a las mejoras de diseño de la aplicación.

13. CRONOLOGÍA DEL PROYECTO

En este apartado se trata de representar la cronología que se ha seguido durante el proyecto de cara a alcanzar los objetivos y desarrollar la idea de negocio.
Con el fin de ilustrar la cronología de este proyecto, se ha realizado un ***diagrama de Gantt*** [31] con las diferentes iteraciones del proyecto, así como cada una de sus fases.

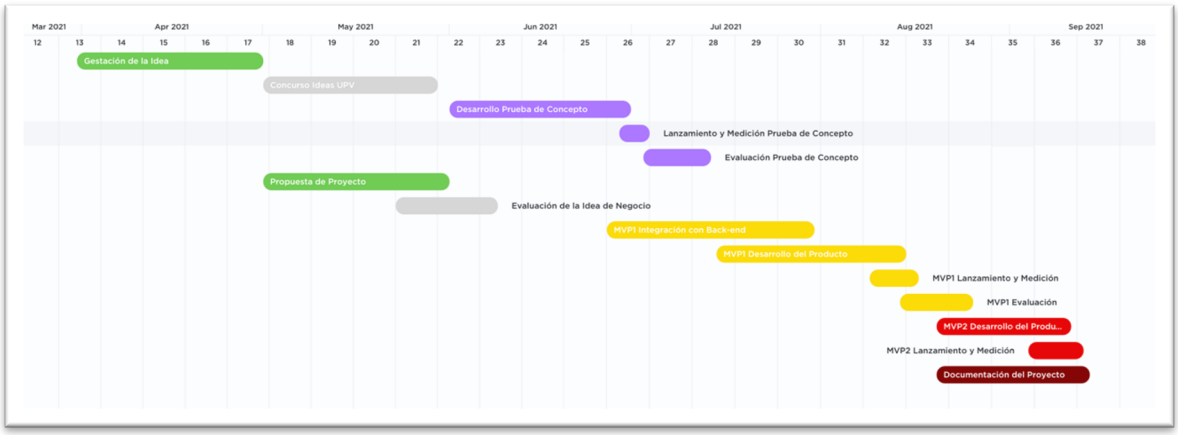


ILUSTRACIÓN 17. CRONOGRAMA DEL PROYECTO. ELABORACIÓN PROPIA

Tal y como podemos ver, en el eje horizontal se aprecian los meses y semanas respecto al **año 2021,** siendo este proyecto desarrollado entre principios de **Abril y** mediados de **Septiembre**. Cabe decir que no todas estas etapas corresponden a este proyecto de fin de Máster, pero sí que se ha considerado importante enunciarlas puesto que han contribuido a la validación y desarrollo de este proyecto.

Por otro lado, se puede apreciar que la fase más duradera del proyecto ha sido el desarrollo del primer MVP. Esto se debe a la elevada complejidad de integrar una aplicación **Android** con un servidor, por esta misma razón, el desarrollo de esta fase contempla un hito que es el de integrar este sistema.

Por último, hay que mencionar que se pueden ver en color verde aquellas tareas que dieron lugar a la forma del proyecto y en gris aquellas que permitieron validar la idea. También se ha asignado un color para cada fase de desarrollo de cada MVP.

A continuación se realiza una breve descripción de cada una de las fases visibles en el cronograma:

- **Gestación de la Idea**. Durante esta fase la idea fue gestada a través de varios proyectos en grupo y en solitario del máster para el que se desarrolla este trabajo.
- **Concurso Ideas UPV.** Concurso de emprendimiento que permitió validar la idea, así como realizar diferentes análisis financieros, de marketing etc.
- **Propuesta de Proyecto.** Proposición y elaboración de este proyecto como trabajo de fin de máster.
- **Evaluación Idea de Negocio.** Evaluación de la idea acontecida a nivel financiero y de cantidad de potenciales clientes, así como la validación del modelo de negocio establecido.
- **Desarrollo Prueba de Concepto.** Fase de desarrollo de la prueba de concepto.

- **Lanzamiento Prueba de Concepto.** Lanzamiento de la prueba de concepto a diferentes usuarios para que puedan probarla y obtener *feedback.*
- **Evaluación Prueba de Concepto.** Evaluación de qué características puede tener o no la aplicación, así como aprendizaje a través de la prueba de concepto.
- **MVP1 Integración con Back-end.** Integración del sistema elaborado durante la prueba de concepto dentro del servidor creado durante esta misma fase.
- **MVP1 Desarrollo del Producto.** Desarrollo de las características relacionadas con la primera versión del producto.
- **MVP1 Lanzamiento y Medición.** Durante esta fase se despliega el MVP1 en un servidor externo facilitando a diferentes usuarios el usar la aplicación y obtener *feedback* de estos, así como gestionar los fallos.
- **MVP1 Evaluación.** Evaluación de los resultados obtenidos durante la fase anterior de cara a decidir qué es importante y qué no de cara al desarrollo de la siguiente fase.
- **MVP2 Desarrollo del Producto.** Desarrollo de las características relacionadas con la segunda versión del producto.
- **MVP2 Lanzamiento y Medición.** Durante esta fase se despliega el MVP2, al igual que en el lanzamiento del primero, con la diferencia de que en esta ocasión se permite el registro de usuarios, abriendo la aplicación para cualquier *early adopter* que desee usarla.
- **Documentación del Proyecto.** Documentación pertinente al proyecto, este documento y otros aspectos de documentación pertenecen a esta fase.

ASPECTOS TÉCNICOS

14. HERRAMIENTAS EMPLEADAS

Durante este apartado se muestran las diferentes herramientas empleadas durante el desarrollo técnico del producto, así como la justificación del uso de dichas herramientas. Todas estas herramientas permiten analizar, implementar o integrar el sistema que se ha desarrollado.

14.1. MAGICDRAW 2021X

MagicDraw [32] es una herramienta que está diseñada para que **Analistas de Negocio, de Software y Programadores** puedan desarrollar y analizar los diferentes modelos de los sistemas informáticos que se desean implementar. Para ello, cuenta con el lenguaje de modelado *UML* [33] el cual permite modelar diferentes aspectos formales del software, así como ***Modelos Conceptuales, de Implementación, Diagramas de Casos de Uso y Diagramas de Secuencia.***

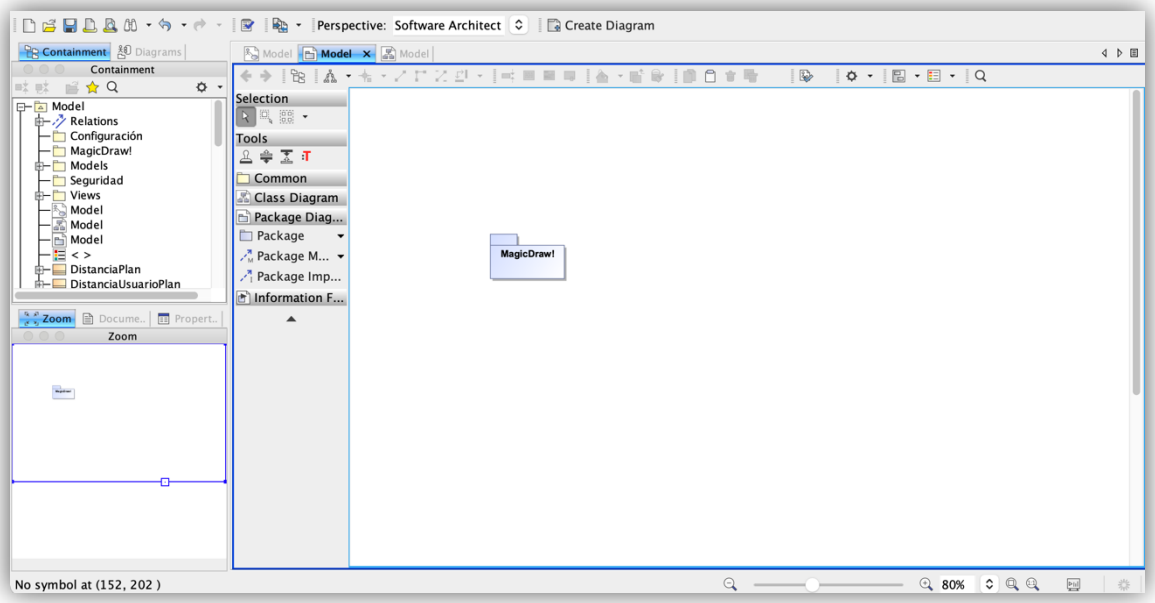


ILUSTRACIÓN 18. CAPTURA DE PANTALLA DE LA INTERFAZ DE MAGICDRAW. ELABORACIÓN PROPIA

14.2. ANDROID STUDIO ARCTIC FOX

Android Studio [34] es una herramienta desarrollada por Google que provee las herramientas para el mayor y mejor desarrollo de aplicaciones en cualquier tipo de **dispositivo Android.**

Entre algunas de estas características se pueden destacar:

- **Emulador Android.** Permite instalar y ejecutar las aplicaciones desarrolladas fácilmente tanto en un dispositivo físico como en un emulador proveído por la propia aplicación.

- **Editor de Código Inteligente.** Esta herramienta permite mejorar la productividad con un editor de código inteligente que provee compleción de código para los lenguajes de programación *Kotlin, Java y C/C++*.
- **Sistema Flexible de Compilación.** Gracias al gestor de paquetes *Gradle* [35], *Android* Studio permite personalizar la compilación y las librerías empleadas de forma completamente sencilla.
- **Interfaz Gráfica de Control de Versiones.** *Android Studio* implementa mediante su interfaz gráfica un control de versiones a través de las diferentes plataformas que emplean *Git* [36].

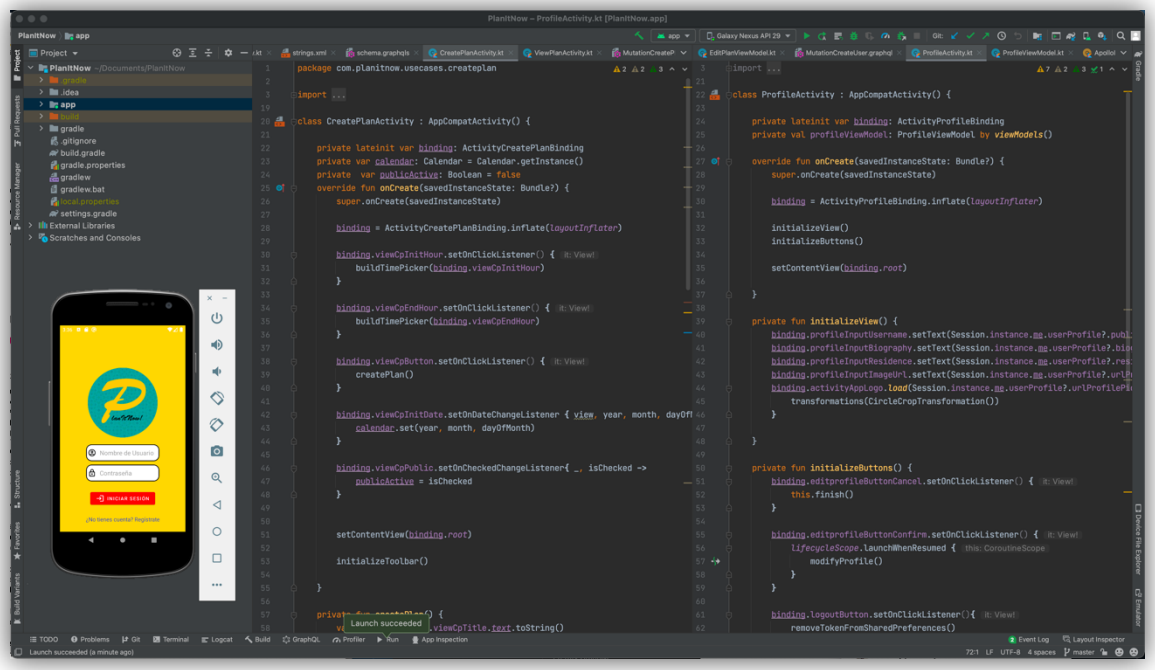


ILUSTRACIÓN 19. INTERFAZ GRÁFICA DE ANDROID STUDIO. ELABORACIÓN PROPIA

14.3. PYCHARM COMMUNITY EDITION

PyCharm [37] es una herramienta desarrollada por *JetBrains* que permite agilizar todas las rutinas del desarrollo del código ***Python***, permitiendo al desarrollador centrarse únicamente en los aspectos más importantes de su código. También facilita la integración con diferentes ***frameworks***[12] para este lenguaje de programación.
Entre algunas de sus características se pueden destacar:

- **Terminal y Entorno Virtual Integrados.** Esta herramienta permite integrar un entorno virtual, lo cual facilita en gran medida el desarrollar código. Además, al tener la terminal integrada no es necesario disponer de una terminal externa.
- **Editor de Código Inteligente.** Esta herramienta autocompleta código y realiza sugerencias y cambios en el código que contribuyen a evitar errores, así como mejorar la estructura visual del código escrito.

[12] Es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática en particular. Para el caso del desarrollo de software, es una estructura tecnológica que facilita el desarrollo de software en diversos aspectos.

- **Interfaz Gráfica de Control de Versiones.** Tal y como ocurre en el caso de **Android Studio, PyCharm** también cuenta con una interfaz gráfica de control de versiones similar.

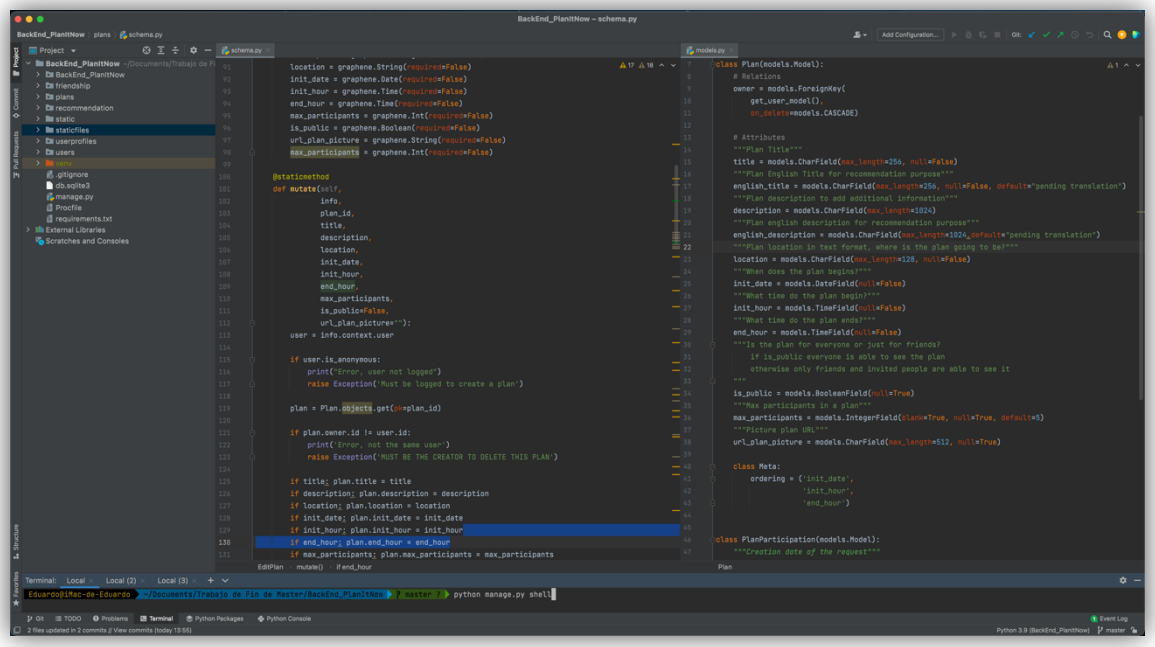
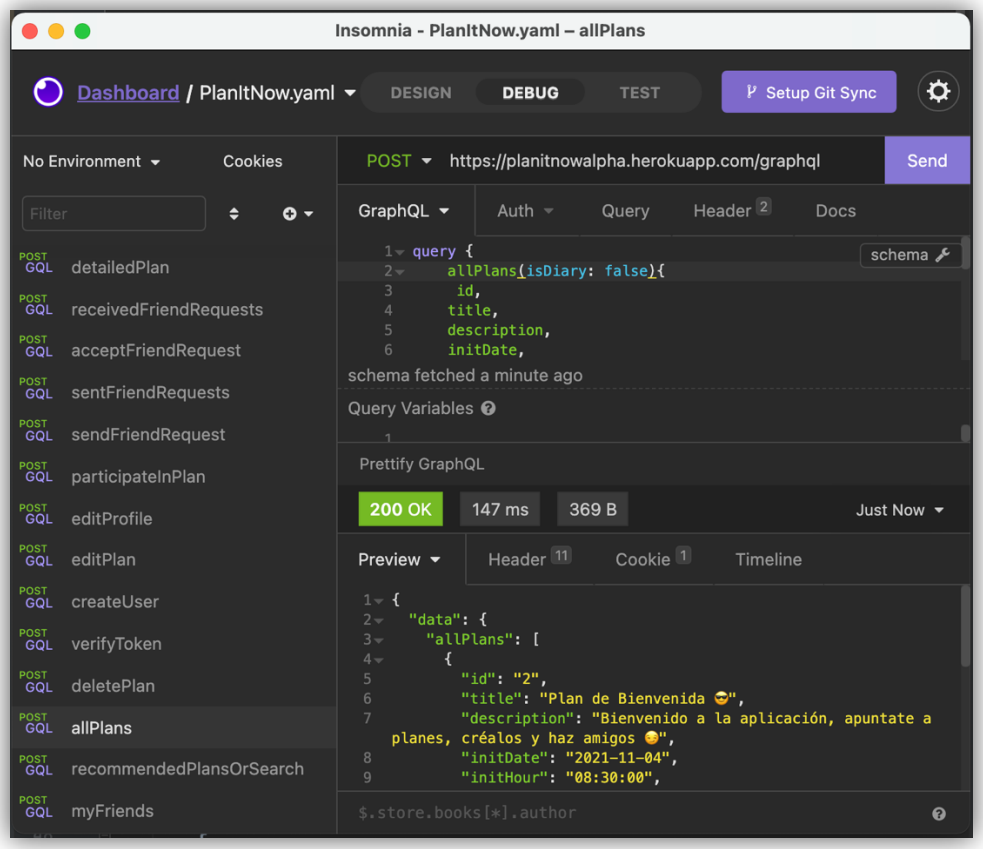


ILUSTRACIÓN 20. INTERFAZ GRÁFICA DE PYCHARM. ELABORACIÓN PROPIA

14.4. INSOMNIA

Insomnia [38] es un **cliente *Open Source*** [39]**de *API*** [40] que permite realizar llamadas a las diferentes *API* creadas, así como realizar un buen diseño de estas *API* mediante estándares como *OpenApi* o *Swagger* [41], automatizar pruebas y otras tareas relacionadas con el desarrollo de estas.

ILUSTRACIÓN 21. INTERFAZ GRÁFICA DE INSOMNIA. ELABORACIÓN PROPIA



15. TECNOLOGÍAS EMPLEADAS

Durante este apartado se muestran las diferentes tecnologías, librerías o *frameworks* utilizados durante el desarrollo del producto para dar lugar a un mayor entendimiento del diseño del sistema. Las diferentes tecnologías que se muestran a continuación se han escogido debido a su facilidad de desarrollo, necesidad o innovación.

15.1. FRONT-END

A continuación, se procede a mostrar las diferentes tecnologías empleadas durante el desarrollo del *front-end* de la aplicación.

15.1.1. JAVA

Java [42] es una tecnología que se usa para el desarrollo de aplicaciones que convierten la Web en un elemento más interesante y útil. Permite jugar, cargar fotografías, chatear en línea, realizar visitas virtuales y utilizar servicios como, por ejemplo, cursos en línea, servicios bancarios en línea y mapas interactivos.
Inicialmente se usó esta tecnología en el desarrollo de la prueba de concepto pues inicialmente el desarrollo **Android** se puede realizar a través de esta.

15.1.2. KOTLIN

Kotlin [29] es un lenguaje de programación moderno a la par que maduro cuyo principal objetivo es hacer a los desarrolladores felices. Es conciso, seguro, interoperable con Java y otros lenguajes y provee diferentes maneras de reutilizar código entre plataformas para mejorar la productividad.
En el caso de este proyecto, *Kotlin* ha sido un lenguaje clave durante el desarrollo Android al seguir esta filosofía mencionada anteriormente. Por esta misma razón se decidió cambiar el lenguaje principal de programación de la plataforma durante las fases de desarrollo.
Toda la aplicación que se puede apreciar durante las fases finales **está desarrollada mediante este lenguaje.**

15.1.3. ANDROID SDK

Android SDK [34] es un conjunto de herramientas y tecnologías que permiten desarrollar código para ser implementado en el sistema operativo Android. Esto permite no sólo realizar un desarrollo más rápido, sino el tener integradas ya las diferentes librerías de desarrollo.

15.1.4. ROOM

Room [43] es un *framework* que proporciona una capa de abstracción sobre *SQLite* que permite acceder a la base de datos sin problemas, y al mismo tiempo, aprovechar toda la potencia de *SQLite*.
En el caso de este proyecto, se empleó *Room* durante la prueba de concepto para almacenar los planes y atributos del sistema de forma local, sin embargo, **tras la**

transición de código a *Kotlin* e integración con el *back-end* se decidió descartar esta librería.

15.1.5. COIL

Coil [44] es una librería de carga de imágenes respaldada por las *corrutinas de Kotlin*, resultando en una implementación **rápida, ligera, fácil de usar y moderna.**
Esta ha sido la librería empleada para cargar de forma eficiente las diferentes imágenes de la aplicación.

15.1.6. APOLLO ANDROID GRAPHQL CLIENT V3

Apollo [45] es un cliente fuertemente tipado y *cacheado* de *GraphQL* [46] para la máquina virtual de Java, permitiendo su desarrollo en Java y *Kotlin*.
Este cliente genera el modelo necesario en Java y *Kotlin* únicamente a partir del esquema de la API en cuestión. Estos modelos contienen tipos seguros que permiten trabajar fácilmente con la *API* del servidor *GraphQL*. *Apollo* también ayuda estructurar de forma correcta y de fácil acceso las diferentes consultas que se realizan en la API.
En el caso de este proyecto, el uso del cliente *Apollo* ha permitido calzar fácilmente la integración del *front-end* con el *back-end*, ya que por un lado permite la **generación automática de modelos** y por otro lado su implementación de las *corrutinas* de Kotlin contribuye a mejorar el rendimiento de la aplicación.
Esta ha sido la librería decisiva de cara a refactorizar todo el código y cambiarlo de lenguaje, a continuación, se muestran dos capturas de pantalla del mismo código realizado en *Java* y *Kotlin* respectivamente:

Java Copy

```java
UpvotePostMutation upvotePostMutation = UpvotePostMutation.builder()
    .votes(3)
    .build();

apolloClient
    .mutate(upvotePostMutation)
    .enqueue(
        new ApolloCallback<>(new ApolloCall.Callback<UpvotePost.Data>() {
          @Override public void onResponse(@NotNull Response<UpvotePost.Data
              Log.i(TAG, response.toString());
          }

          @Override public void onFailure(@NotNull ApolloException e) {
              Log.e(TAG, e.getMessage(), e);
          }
        });
    );
```

ILUSTRACIÓN 22. CÓDIGO DE MUTACIÓN APOLLO ESCRITO EN JAVA [45]

ILUSTRACIÓN 23. CÓDIGO DE MUTACIÓN APOLLO ESCRITO EN KOTLIN [45]

Tal y como se puede observar, para realizar la misma funcionalidad, el código ocupa tres líneas más en *Java* que en *Kotlin*. Hay que tener en cuenta que este es un ejemplo sencillo, sin embargo, para arquitecturas más complejas, el código se vuelve más incomprensible.

Para facilitar el desarrollo de código mediante esta librería, se decidió utilizar la última versión la cual es significativamente más simple que la del ejemplo anterior, con la contrariedad de que está diseñada únicamente para ejecutar código *Kotlin*.

15.2. BACK-END

A continuación, se procede a mostrar las diferentes tecnologías durante el desarrollo *back-end* de la aplicación.

15.2.1. PYTHON

Python [47] es un lenguaje de programación fácil de aprender con una gran cantidad de librerías. En el caso de este proyecto se ha escogido debido a su facilidad para tratar datos, desarrollar y ser integrado con una base de datos.

15.2.2. DJANGO

Django [48] es un *framework* de alto nivel de *Python* que facilita el desarrollo limpio rápido y pragmático. Se encarga principalmente de la mayor parte de los aspectos del desarrollo web, así como las conexiones a la base de datos.
Algunas de las características de este *framework* son las siguientes:

- **Rápido.** Django fue diseñado para ayudar a los desarrolladores a crear aplicaciones tan rápido como sea posible.
- **Seguro**. Django se encarga de los apartados de seguridad de cara a que los desarrolladores no cometan fallos de seguridad.
- **Escalable.** Gran parte de las webs más usadas emplean *Django* por esta misma razón. Es completamente escalable tanto a nivel de usuarios como a nivel de nuevas características.

Por estas razones se ha escogido Django como *framework* principal del *back-end* del sistema.

15.2.3. GRAPHENE

Graphene [49] es una librería para construir *APIs GraphQL* en *Python* fácilmente. Su principal objetivo es proveer una API simple a la par que extensible, facilitando la vida de los desarrolladores.
Esta librería ofrece una completa integración con *Django* mediante la extensión *Graphene-Django* [50], librería la cual provee de abstracciones para añadir funcionalidad *GraphQL* al proyecto *Django.*
A continuación, se muestra un ejemplo básico de su funcionamiento:

```python
# my_app/schema.py

import graphene
from graphene_django import DjangoObjectType

from .models import Question

class QuestionType(DjangoObjectType):
    class Meta:
        model = Question
        fields = ("id", "question_text")

class Query(graphene.ObjectType):
    questions = graphene.List(QuestionType)
    question_by_id = graphene.Field(QuestionType, id=graphene.String())

    def resolve_questions(root, info, **kwargs):
        # Querying a list
        return Question.objects.all()

    def resolve_question_by_id(root, info, id):
        # Querying a single question
        return Question.objects.get(pk=id)
```

ILUSTRACIÓN 24. EJEMPLO DE ESQUEMA Y QUERY EN GRAPHENE [50]

Se puede apreciar que únicamente hay que definir una **clase *Query*** así como un **tipo** de *GraphQL* y *Graphene-django* se encarga de retornar los diferentes objetos pertenecientes a los *modelos de Django.*

15.2.4. DEEP TRANSLATOR

Deep translator [51] es una herramienta de *Python* que permite traducir entre diferentes idiomas de una forma simple usando diferentes traductores.
Se ha escogido esta tecnología de traducción debido a que al poder realizar traducciones mediante diversas *API*, se reduce la carga por parte del sistema.

15.2.5. NATURAL LANGUAGE TOOLKIT

NLTK [52] es una plataforma líder para construir programas *Python* que permiten interactuar con humanos a través de proveer interfaces de uso simple junto con múltiples recursos léxicos. Esto permite el procesado de texto a diferentes niveles, pudiendo obtener información extra de todo el texto insertado.
Actualmente esta plataforma únicamente funciona en inglés, y es por esta misma razón por la que ha sido necesario incluir la librería anterior. Por otro lado, esta librería se ha escogido al ser considerada como la mejor para extraer información del texto existente en la aplicación.

16. ESPECIFICACIÓN DEL SISTEMA

En esta sección se realiza la especificación del sistema que se ha implementado. Ya que esto es una aplicación para el **sistema operativo *Android***, se asociará cada uno de los requisitos funcionales a un caso de uso, mientras que aquellos que resulten no funcionales estarán ligados a características del sistema que no están relacionadas con estos casos.

16.1. REQUISITOS FUNCIONALES

En esta sección se definen los diferentes requisitos funcionales. Puesto que todos los requisitos funcionales están relacionados con los casos de uso de esta aplicación, estos tratan de satisfacer las diferentes necesidades que los usuarios puedan tener a la hora de usar la aplicación.

16.1.1. RF PERSISTENCIA

El sistema debe almacenar la información de cada uno de los usuarios y requisitos que se enuncian a continuación.

16.1.2. RF INICIAR SESIÓN

El sistema debe permitir iniciar sesión con unas credenciales de usuario. Es decir, mediante un usuario y contraseña.

16.1.3. RF CERRAR SESIÓN

El sistema debe permitir cerrar sesión.

16.1.4. RF INICIAR SESIÓN AUTOMÁTICAMENTE

El sistema debe permitir iniciar sesión automáticamente cuando el usuario ha iniciado sesión previamente.

16.1.5. RF REGISTRO

El sistema debe permitir el registro de un usuario, así como de los datos requeridos para usar la aplicación, creando así una cuenta de usuario para poder iniciar sesión.

16.1.6. RF EDITAR PERFIL

El sistema debe permitir editar el perfil de un usuario.

16.1.7. RF VER PLAN

El sistema debe permitir a los diferentes usuarios visualizar los planes que están contenidos dentro del mismo, así como sus diferentes atributos.

16.1.8. RF APUNTARSE A PLAN

El sistema debe permitir a los diferentes usuarios apuntarse a los planes públicos y a los de sus amigos.

16.1.9. RF BORRAR PLAN

El sistema debe permitir al usuario creador de un plan borrarlo del sistema.

16.1.10. RF FEED DE PLANES

El sistema debe permitir a los diferentes usuarios visualizar en una lista todos los planes con fecha futura, siendo estos planes aquellos que el usuario ha creado o se ha apuntado.

16.1.11. RF DIARIO DE PLANES

El sistema debe permitir a los diferentes usuarios visualizar en una lista todos los planes con fecha pasada, siendo estos planes aquellos que el usuario ha creado o se ha apuntado.

16.1.12. RF FEED DESCUBRIMIENTO DE PLANES

El sistema debe permitir a los diferentes usuarios visualizar en una lista todos los planes con fecha futura, siendo estos aquellos que hayan creado sus amigos o que son públicos dentro del sistema.

16.1.13. RF BUSCAR PLANES

El sistema debe permitir buscar planes.

16.1.14. RF VER RECOMENDACIONES

El sistema debe permitir a los diferentes usuarios visualizar recomendaciones personalizadas.

16.1.15. RF VER AMIGOS

El sistema debe permitir a los diferentes usuarios visualizar a sus amigos.

16.1.16. RF ENVIAR PETICIÓN DE AMISTAD

El sistema debe permitir enviar peticiones de amistad.

16.1.17. RF ACEPTAR PETICIÓN DE AMISTAD

El sistema debe permitir aceptar peticiones de amistad.

16.1.18. RF VER NOTIFICACIONES

El sistema debe permitir visualizar las diferentes notificaciones.

16.1.19. RF CREAR PLAN

El sistema debe permitir crear planes.

16.1.20. RF EDITAR PLAN

El sistema debe permitir editar planes.

16.2. REQUISITOS NO FUNCIONALES

En este apartado se tienen en cuenta todos aquellos requisitos que no sean funcionales pero que deban tenerse en cuenta de cara a desarrollar ***PlanItNow!*** con la mayor calidad posible.

16.2.1. REQUISITOS DE APARIENCIA

16.2.1.1. RNF DISEÑO MATERIAL DESIGN

El diseño del sistema debe seguir en la medida de lo posible las pautas de diseño establecidas por *Material Design* [30].

16.2.1.2. RNF FAMILIARIDAD CON OTRAS APLICACIONES

El sistema debe mantener un diseño que resulte familiar con otras aplicaciones comerciales.

16.2.2. REQUISITOS DE USABILIDAD

16.2.2.1. RNF APRENDIZAJE SENCILLO

La curva de aprendizaje del sistema ha de ser sencilla.

16.2.2.2. RNF GUIAR ERRORES

El sistema debe hacer tan complejo como sea posible cometer errores por parte de los usuarios. En el caso de que ocurra, el sistema debe guiar el usuario para solucionar estos errores.

16.2.3. REQUISITOS DE ESCALABILIDAD

16.2.3.1. RNF SISTEMA AMPLIABLE

El sistema debe permitir ampliar o modificar sencillamente cualquiera de sus funcionalidades.

16.2.3.2. RNF SISTEMA ESCALABLE

El sistema debe funcionar correctamente sea cual sea el número de usuarios que usen la aplicación.

16.3. CASOS DE USO

De cara a definir la implementación de los diversos requisitos funcionales del a aplicación, se ha optado por desarrollarlos a través de casos de uso. Esta decisión se ha tomado debido a que, dentro de los diversos modelos de especificación, es el que mejor cumple con la descripción completa de las funcionalidades de la aplicación.

16.3.1. DIAGRAMA DE CASOS DE USO

Con el fin de poder mostrar cómo los diferentes casos de uso se relacionan entre sí, se ha optado por realizar un diagrama *de Casos de Uso* [53].
En el diagrama se puede observar que los casos de uso corresponden de forma directa con los requisitos mencionados anteriormente.

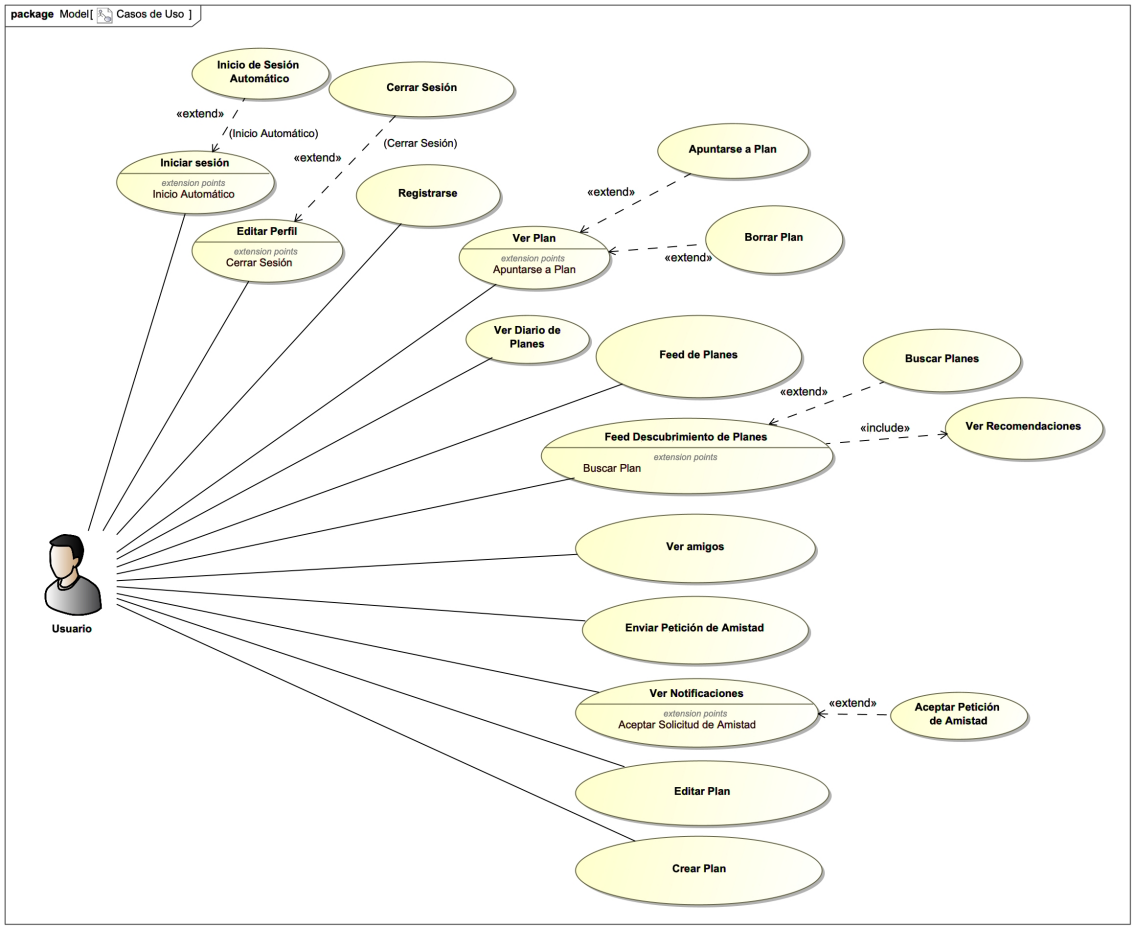


ILUSTRACIÓN 25. CASOS DE USO DE LA APLICACIÓN. ELABORACIÓN PROPIA

A continuación, se procede a realizar la especificación de cada caso de uso. De cara a evitar duplicar la especificación de casos de uso, sólo se especificarán los casos de uso que están directamente ligados al usuario. Las extensiones e inclusiones que se pueden apreciar en el diagrama se especifican en el caso de uso padre.
Por ejemplo, para el caso de **Iniciar Sesión** se especifica también el caso de **Inicio de Sesión Automático.**

CU1. INICIAR SESIÓN

Título	**Iniciar Sesión**
Descripción	El sistema debe permitir iniciar sesión con unas credenciales de usuario. Es decir, mediante un usuario y contraseña.
Pre-Condición	El usuario se encuentra en la vista de *Inicio de Sesión*
Post-Condiciones	El usuario inicia sesión
Autor	Usuario
Escenario Principal	1. El usuario rellena el usuario y contraseña. 2. El usuario selecciona la opción *Iniciar Sesión.* 3. El sistema comprueba si las credenciales obtenidas son válidas. 4. El sistema permite avanzar hacia la siguiente vista.
Escenario Alternativo	1.a. El usuario ya había iniciado sesión previamente. 1.a.1 El sistema comprueba si la sesión sigue siendo válida. 1.a.2 El sistema permite avanzar hacia la siguiente vista. 3.a.1 El sistema detecta que las credenciales obtenidas no son válidas. 3.a.2 El sistema muestra el mensaje "*Tu nombre de usuario o contraseña son incorrectos"* 3.b.1 El sistema detecta que el usuario ha introducido erróneamente más de 3 veces las credenciales. 3b.2 El sistema devuelve el mensaje de error: *Máximo Alcanzado, has alcanzado el número máximo de intentos para el login. Por favor inténtalo más tarde*"

CU2. REGISTRARSE

Título	**Registrarse**
Descripción	El sistema debe permitir registrarse a los diferentes usuarios, creando así sus credenciales de usuario.
Pre-Condición	El usuario se encuentra en la vista de *Registro*
Post-Condiciones	El usuario crea sus credenciales de Usuario
Autor	Usuario
Escenario Principal	1. El usuario rellena los campos *Nombre, Apellidos, Email, Fecha de Nacimiento, Nombre de Usuario, Contraseña y Confirmar Contraseña.* 2. El usuario selecciona la acción *Registrarse.* 3. El sistema crea las credenciales de usuario con la información introducida. 4. El sistema muestra el mensaje *"Te has registrado con éxito"*
Escenario Alternativo	3.a.1 La contraseña no coincide con la confirmación de contraseña. 3.a.2 El sistema muestra el mensaje de error *"La contraseña no coincide. Revísala"* 3.b.1 Faltan datos obligatorios por introducir. 3b.2 El sistema muestra un mensaje de error especificando los campos que faltan por ser introducidos.

CU3. EDITAR PERFIL

Título	**Editar Perfil**
Descripción	El sistema debe permitir editar el perfil de un usuario.
Pre-Condición	El usuario se encuentra en la vista de *Editar Perfil*
Post-Condiciones	El usuario edita su perfil
Autor	Usuario
Escenario Principal	1. El usuario modifica alguno de los atributos de su perfil *Nombre Público, Biografía, Residencia o Enlace a Tu Foto de Perfil.* 2. El usuario selecciona la opción *Modificar Perfil* 3. El sistema modifica el perfil del usuario con los campos actualizados. 4. El sistema muestra el mensaje *"Se ha editado tu perfil correctamente".*
Escenario Alternativo	1.a.1 El usuario selecciona la acción *Cerrar Sesión.* 1.a.2 El sistema muestra el mensaje *"Se ha cerrado sesión. Surtirá efecto cuando reinicies la app".*

CU3. VER PLAN

Título	**Ver Plan**
Descripción	El sistema debe permitir a los diferentes usuarios visualizar los planes que están contenidos dentro del mismo, así como sus diferentes atributos.
Pre-Condición	El usuario se encuentra en la vista de Ver Plan
Post-Condiciones	
Autor	Usuario
Escenario Principal	1. El sistema carga los atributos de un plan en específico. 2. El sistema valida si el usuario que está observando el plan es el creador. 3. El sistema muestra los atributos de un plan en específico.
Escenario Alternativo	3.a.1 El sistema verifica que el usuario es el creador del plan. 3.a.2 El sistema muestra los campos de borrado y edición. 3.a.3 El usuario selecciona la opción *"Borrar"*. 3.a.4 El sistema muestra un mensaje de confirmación. 3.a.5 El usuario selecciona *Aceptar.* 3.a.6 El sistema elimina el plan de forma interna. 3.a.7 El sistema muestra el mensaje *"Plan borrado correctamente".* 3.b.1 El sistema verifica que el usuario no es el creador del plan. 3.b.2 El sistema muestra la acción *Apuntarse/Desapuntarse.* 3.b.3 El usuario selecciona la opción *Apuntarse/Desapuntarse.* 3.b.4 El sistema añade/elimina el usuario a los participantes del plan.

CU4. VER DIARIO DE PLANES

Título	**Ver Diario de Planes**
Descripción	El sistema debe permitir a los diferentes usuarios visualizar en una lista todos los planes con fecha pasada, siendo estos planes aquellos que el usuario ha creado o se ha apuntado.
Pre-Condición	El usuario se encuentra en la vista de *Diario de Planes*
Post-Condiciones	El sistema muestra el diario de planes
Autor	Usuario
Escenario Principal	1. El sistema carga los planes que el usuario ha creado o se ha apuntado con fecha pasada al día actual. 2. El sistema ordena los planes por fecha de más reciente a más antiguo. 3. El sistema muestra los planes.

CU5. FEED DE PLANES

Título	**Feed de Planes**
Descripción	El sistema debe permitir a los diferentes usuarios visualizar en una lista todos los planes con fecha futura, siendo estos planes aquellos que el usuario ha creado o se ha apuntado.
Pre-Condición	El usuario se encuentra en la vista de *Mis Planes*
Post-Condiciones	El sistema muestra los planes activos del usuario
Autor	Usuario
Escenario Principal	1. El sistema carga los planes que el usuario ha creado o se ha apuntado con fecha futura. 2. El sistema ordena los planes por fecha de más actual a más lejano. 3. El sistema muestra los planes.

CU6. FEED DESCUBRIMIENTO DE PLANES

Título	**Feed Descubrimiento de Planes**
Descripción	El sistema debe permitir a los diferentes usuarios visualizar en una lista todos los planes con fecha futura, siendo estos aquellos que hayan creado sus amigos o que son públicos dentro del sistema.
Pre-Condición	El usuario se encuentra en la vista de *Descubrir*
Post-Condiciones	El sistema muestra los planes visibles y recomendados de la aplicación.
Autor	Usuario
Escenario Principal	1. El sistema carga todos los planes públicos, y de los privados aquellos que ha creado el usuario o sus amigos. 2. El sistema ejecuta el algoritmo de recomendación para calcular las recomendaciones del usuario. 3. El sistema obtiene las recomendaciones. 4. El sistema muestra todos los planes, siendo primeros aquellos recomendados.

CU7. BUSCAR PLANES

Título	**Buscar Planes**
Descripción	El sistema debe permitir buscar planes.
Pre-Condición	El usuario se encuentra en la vista de *Descubrir*
Post-Condiciones	El sistema muestra los planes visibles en función de una cadena de texto.
Autor	Usuario
Escenario Principal	1. El usuario selecciona la opción *Buscar*. 2. El usuario introduce texto de búsqueda. 3. El usuario pulsa la acción *buscar*. 4. El sistema carga los planes que contienen el texto de búsqueda. 5. El sistema muestra los planes que contienen el texto de búsqueda.

CU8. VER AMIGOS

Título	**Ver Amigos**
Descripción	El sistema debe permitir a los diferentes usuarios visualizar a sus amigos.
Pre-Condición	El usuario se encuentra en la vista de *Amigos.*
Post-Condiciones	El sistema muestra los amigos del usuario.
Autor	Usuario
Escenario Principal	1. El sistema carga la lista de amigos de un usuario. 2. El sistema muestra la lista de amigos de un usuario.

CU9. ENVIAR PETICIÓN DE AMISTAD

Título	**Enviar Petición de Amistad**
Descripción	El sistema debe permitir enviar peticiones de amistad.
Pre-Condición	El usuario se encuentra en la vista de *Amigos*.
Post-Condiciones	El sistema envía una solicitud de amistad.
Autor	Usuario
Escenario Principal	1. El usuario selecciona la opción *Añadir Amigo*. 2. El sistema muestra un menú donde se puede rellenar un nombre de usuario. 3. El usuario rellena el nombre de usuario de la persona con la que desea establecer amistad. 4. El usuario seleccionar *Enviar*. 5. El sistema crea la petición de amistad. 6. El sistema muestra el mensaje de éxito *"Tu solicitud de amistad se ha enviado correctamente"*.
Escenario Alternativo	5.a.1 El sistema detecta que el usuario no existe o ya hay una petición previa. 5.a.2 El sistema muestra el mensaje de error *"El usuario no existe o ya has enviado una solicitud de amistad"*.

CU10. VER NOTIFICACIONES

Título	**Ver Notificaciones**
Descripción	El sistema debe permitir visualizar las diferentes notificaciones.
Pre-Condición	El usuario se encuentra en la vista de *Notificaciones*.
Post-Condiciones	El sistema muestra las notificaciones del usuario.
Autor	Usuario
Escenario Principal	1. El sistema carga las notificaciones del usuario. 2. El sistema ordena las notificaciones por fecha de más reciente a más antigua. 3. El sistema carga las opciones *Aceptar* y *Rechazar* para las notificaciones que requieran esta acción. 4. El sistema muestra todas las notificaciones.
Escenario Alternativo	3.a.1 El usuario selecciona la opción *Aceptar* para una solicitud de amistad. 3.a.2 El sistema acepta la solicitud de amistad. 3.a.3 El sistema añade mutuamente como amigos a los usuarios involucrados. 3.a.4 El sistema elimina las opciones *Aceptar y Rechazar* para esa notificación.

CU11. CREAR PLAN

Título	**Crear Plan**
Descripción	El sistema debe permitir crear planes.
Pre-Condición	El usuario se encuentra en la vista de *Crear Plan.*
Post-Condiciones	El sistema crea un plan
Autor	Usuario
Escenario Principal	1. El usuario rellena los campos correspondientes al plan. 2. El usuario selecciona la acción *Crear*. 3. El sistema verifica los campos introducidos. 4. El sistema crea el plan dentro de la base de datos. 5. El sistema muestra el mensaje de éxito *"El plan se ha creado correctamente"*.
Escenario Alternativo	3.a.1 El sistema detecta un error en los campos introducidos. 3.a.2 El sistema muestra un mensaje de error con el mensaje correspondiente.

CU12. EDITAR PLAN

Título	**Editar Plan**
Descripción	El sistema debe permitir editar planes.
Pre-Condición	El usuario se encuentra en la vista de *Editar Plan.*
Post-Condiciones	El sistema edita los campos del plan modificados.
Autor	Usuario
Escenario Principal	1. El sistema carga los atributos almacenados en la base de datos del plan. 2. El sistema rellena previamente los campos correspondientes al plan. 3. El usuario rellena los campos que desea modificar. 4. El usuario realiza la acción *Editar.* 5. El sistema verifica los campos modificados. 6. El sistema edita los campos del plan modificados. 7. El sistema muestra el mensaje de éxito *"Tu plan se ha editado correctamente."* 8. El sistema carga la visualización del plan con los campos actualizados.
Escenario Alternativo	5.a.1 El sistema detecta un error en los campos introducidos. 5.a.2 El sistema muestra un mensaje de error con el mensaje correspondiente.

16.4. MODELO CONCEPTUAL

Con el fin de mantener un modelo común que permita implementar los diferentes requisitos y casos de uso, se ha creado un modelo conceptual con las principales entidades de la aplicación.

A continuación, se muestra el diagrama del modelo conceptual:

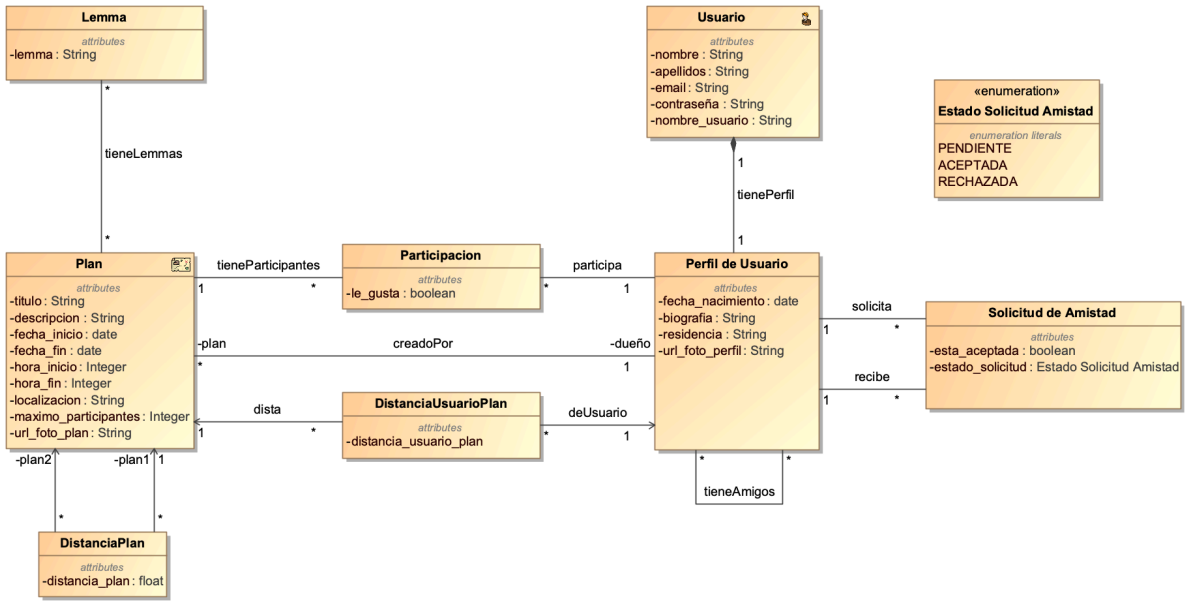


ILUSTRACIÓN 26. DIAGRAMA CONCEPTUAL DE LA APLICACIÓN. ELABORACIÓN PROPIA

A lo largo de este modelo conceptual, podemos ver todas las entidades que se pretenden implementar posteriormente tanto en el *front-end* como el *back-end.* Este diagrama se ha elaborado con la principal funcionalidad de mantener el concepto de la aplicación a través de ambos sistemas.
En este diagrama podemos observar que existen las siguientes entidades:

- **Plan.** Es la entidad principal de la aplicación, esta entidad alberga todos los atributos necesarios para formular un plan.
- **Usuario.** Es la entidad relacionada con el concepto de usuario en sí. Esta entidad se pretende usar principalmente de cara al inicio de sesión y gestión de la misma.
- **Perfil de Usuario.** Es la entidad que muestra todos los atributos del perfil del usuario que no son esenciales para el uso de la plataforma. Se ha creado esta entidad para permitir desacoplar toda la lógica de la aplicación de la entidad **usuario.**
- **Participación.** Es la entidad que gestiona las diferentes participaciones en los planes. Esta entidad pretende realizar las gestiones entre las diferentes participaciones de los planes.
- **Solicitud de Amistad.** Esta entidad pretende gestionar las solicitudes de amistad, así como los conjuntos de amigos de los diferentes usuarios.
- **Lemma.** Esta entidad pretende almacenar los diferentes *lemas* de los planes para dar pie a la recomendación.
- **DistanciaPlan.** Esta entidad pretende almacenar las distancias entre cada par de planes para dar pie a la recomendación.
- **DistanciaUsuarioPlan**. Esta entidad pretende almacenar las distancias entre cada usuario y los diferentes planes para dar pie a la recomendación.

17. DISEÑO DEL SISTEMA

En este apartado se explica cómo se ha desarrollado el diseño de la solución de la aplicación ***PlanItNow!.*** A lo largo de este apartado se tratan a fondo los aspectos técnicos relacionados con la especificación, así como la justificación de cada decisión de diseño.

17.1. ARQUITECTURA DEL SISTEMA

Debido a que este proyecto está compuesto por un *front-end* y un *back-end* ha sido necesario establecer una arquitectura del sistema sólida que permita diferentes características, como la **integración de ambas plataformas, la mantenibilidad y escalabilidad** en caso de que esto fuese necesario.
La arquitectura del sistema está diseñada en forma de ***servicio web***, de la que se dispone un ***cliente*** que puede ejecutar diversas funcionalidades dentro de este servicio.

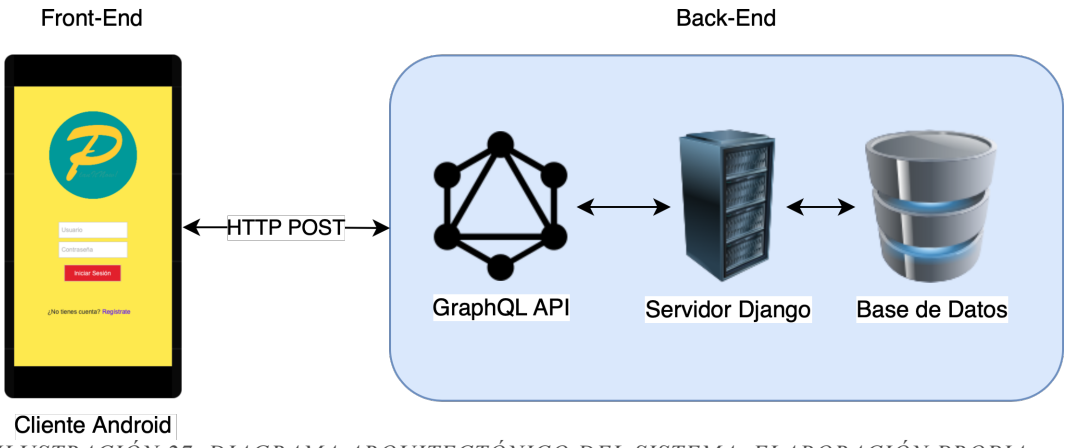


ILUSTRACIÓN 27. DIAGRAMA ARQUITECTÓNICO DEL SISTEMA. ELABORACIÓN PROPIA

En el diagrama mostrado se puede observar que el sistema dispone de un **cliente Android** como *front-end*, mientras que en el lado del *back-end* se ha integrado una ***API GraphQL*** conectada a un **servidor *Django*** que finalmente se conecta con una ***base de datos*** del propio servidor.
Por otro lado, el cliente y el servidor se comunican a través de mensajes ***POST*** que intercambian datos en el formato *GraphQL*.
Idealmente se ha planteado esta arquitectura, ya que facilita evolucionarla en un futuro a una ***arquitectura de microservicios*** [54].

17.2. DISEÑO DEL BACK-END

Este apartado desglosa las diferentes decisiones de diseño tomadas a lo largo del desarrollo del *back-end.* En este apartado se muestran los diferentes paquetes creados, su contenido y el diseño de cada una de las clases implementadas.

17.2.1. DIAGRAMA DE PAQUETES

A continuación, se muestra el diagrama de paquetes de la solución final. Es importante tener en cuenta que almacenar las diferentes funcionalidades del sistema en paquetes independientes entre sí permite poder ampliar y modificar las funcionalidades de forma sencilla, así como estructurar correctamente la implementación del proyecto.

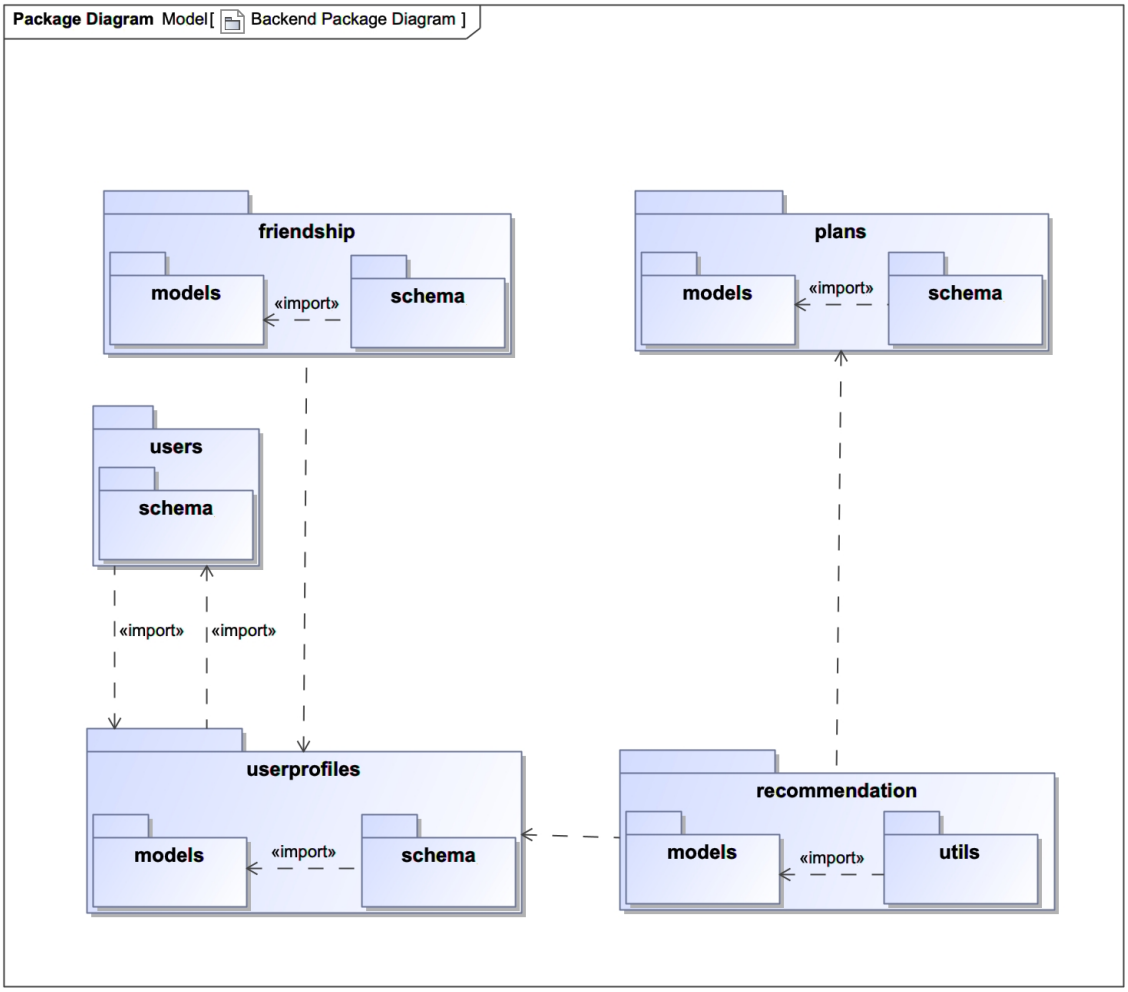


ILUSTRACIÓN 28.DIAGRAMA DE PAQUETES DEL BACK-END. ELABORACIÓN PROPIA

En este diagrama se observa un elemento común entre varios paquetes. Este elemento son los subpaquetes **models y schema:**

- El paquete **models** almacena las diferentes entidades que se persisten finalmente en base de datos, así como las diferentes funciones necesarias para poder gestionar estas entidades.
- El paquete **schema** define las diferentes funciones que se pueden ejecutar a través de la *API GraphQL*, así como la lógica de cada una de las operaciones. Este paquete también define los diferentes submodelos que se pueden mostrar en cada una de las llamadas.

Por otro lado, podemos observar los diferentes paquetes principales:

- **Users** se encarga de almacenar toda la lógica y modelos necesarios para dar lugar a tareas como la autenticación, creación y gestión de usuarios.
- **UserProfiles** se encarga de almacenar toda la lógica y modelos necesarios para poder albergar perfiles de usuario, así como otorga una capa extra para aplicar lógica de planes y otros factores con el fin de desacoplar a los usuarios de sus perfiles.
- **Plans** se encarga de almacenar toda la lógica y modelos necesarios para dar lugar a los planes y sus participaciones.
- **Friendship** se encarga de almacenar toda la lógica y modelos necesarios para dar lugar a la amistad entre usuarios.
- **Recommendation** se encarga de almacenar toda la lógica y modelos necesarios para poder recomendar planes a los diferentes usuarios.

17.2.2. DIAGRAMA DE MODELOS

A continuación, se muestra el diagrama de clases del diseño final. Cabe decir que este diagrama se ha generado automáticamente a través de la herramienta *django-extensions* [55] ya que la herramienta UML seleccionada no permitía establecer los atributos relacionados con el diseño de aplicaciones de *django.* Por otro lado, destacar que cada uno de los modelos que se muestran en este diagrama heredan de los ***modelos de django*** [56], permitiendo también el realizar cada una de las llamadas pertinentes a esta clase padre.

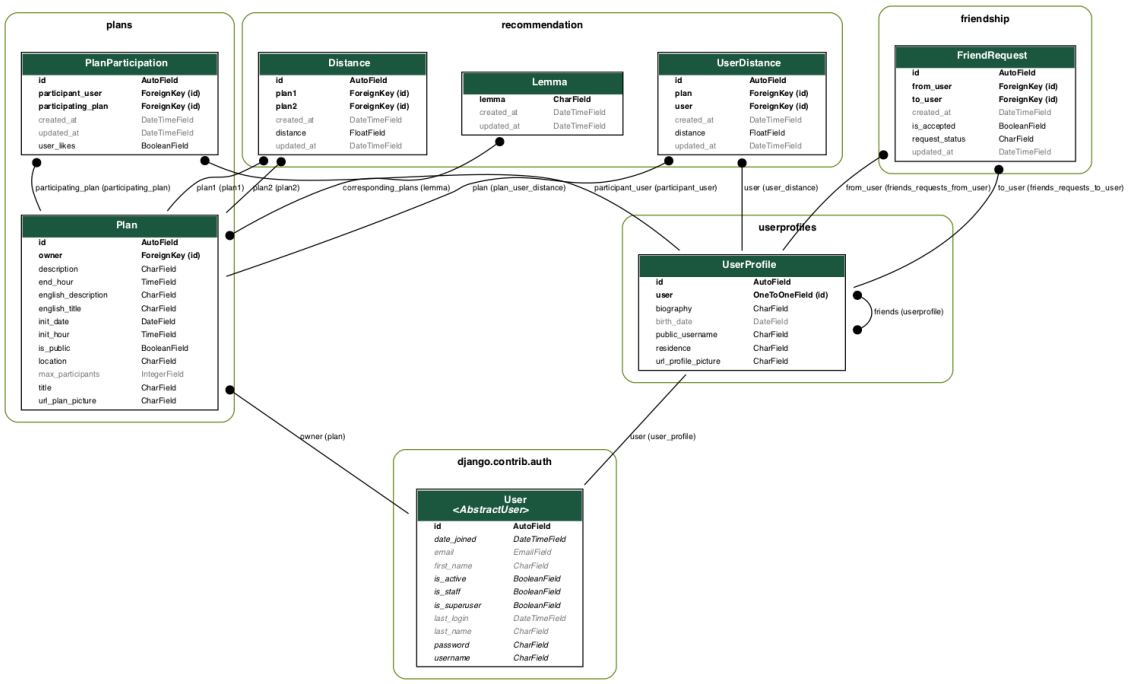


ILUSTRACIÓN 29. DIAGRAMA DE MODELOS IMPLEMENTADOS EN DJANGO. ELABORACIÓN PROPIA

En este diagrama podemos ver en una caja verde los diferentes modelos implementados a lo largo del proyecto, cabe decir que todos **estos modelos implementados corresponden a los definidos previamente en el diagrama conceptual de la aplicación**. Todos estos modelos pueden almacenarse y cargarse de una base de datos externa gracias a la integración que ***Django*** ofrece con muchos de los proveedores de bases de datos.

Por otro lado, podemos ver las relaciones. **Los extremos marcados** son aquellos que tienen la relación como atributo interno de la clase. Sin embargo, ***Django*** permite la navegación inversa entre estos elementos.
Además, podemos ver que los nombres de las relaciones y entre paréntesis, el nombre de la relación inversa, si es que existe.
También se han añadido diversas funciones en las diferentes clases de cara a facilitar el desarrollo de la plataforma que no pueden observarse en este esquema.
Por último, el diseño de estas clases se ha ido desarrollando a lo largo del proyecto según se implementase un caso de uso u otro, tratando de seguir la estructura de paquetes definida inicialmente.

17.2.3. DISEÑO DE API GRAPHQL

En primer lugar, **para entender el diseño de la *API GraphQL*** es necesario mencionar algunos conceptos básicos sobre este tipo de *API*.

- Estas *API* cuentan con un único *endpoint* el cual se encarga de realizar las llamadas a las correspondientes partes del código.
- Las diferentes llamadas se dividen en ***Querys y Mutations.*** Las primeras únicamente obtienen datos del sistema, mientras que las segundas permiten también modificar y persistir datos del sistema.
- Los ***Type*** definen los diferentes atributos que se pueden retornar en cada ***Query*** **y** ***Mutation.***
- Un ***Schema*** alberga un conjunto determinado de ***Querys*** **y** ***Mutations.***
- La librería escogida, *Graphene* permite crear una relación directa entre el **modelo de *Django* y *GraphQL*.**

Por esta misma razón, el diseño de esta *API* se ha centrado en proveer todos los ***Type*** necesarios para el *front-end*, así como las diversas operaciones de ***Query*** **y** ***Mutations*** que se han considerado necesarias de cara a ejecutar los diferentes casos de uso del sistema. Debido a que el sistema es únicamente un servicio web, toda la lógica del sistema se alberga dentro de estas operaciones.
Todas estas operaciones y tipos se encuentran dentro del subpaquete ***schema*** contenido dentro de cada uno de los paquetes principales del *back-end*.
A continuación, se muestran los diagramas de clases correspondientes al diseño de los diferentes ***Type*****,** ***Query*** **y** ***Mutations.***

ILUSTRACIÓN 30. DIAGRAMA DE ENTIDADES DE LA API GRAPHQL 1/2. ELABORACIÓN PROPIA

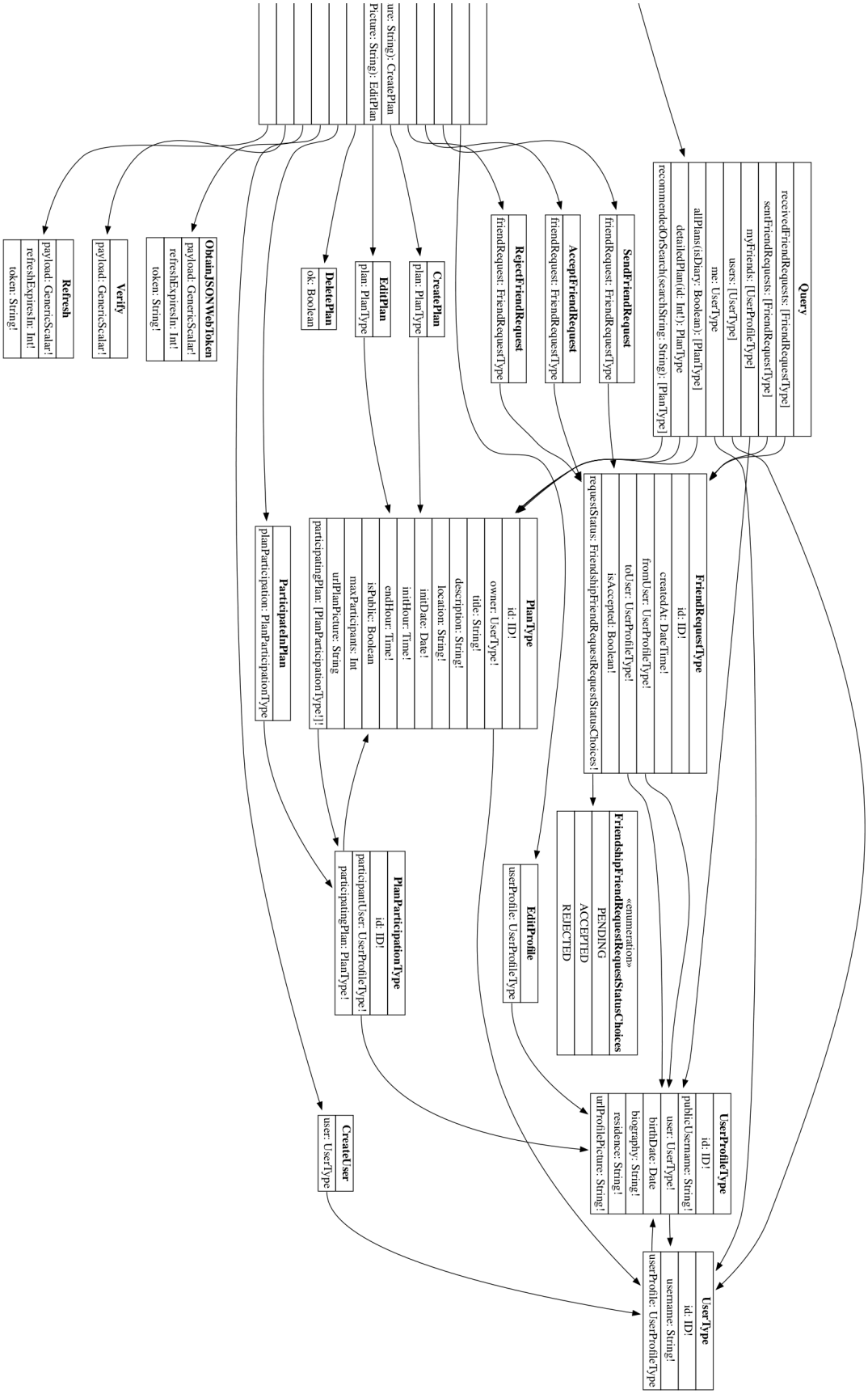


ILUSTRACIÓN 31. DIAGRAMA DE ENTIDADES DE LA API GRAPHQL 2/2. ELABORACIÓN PROPIA

En el caso de este diagrama, podemos ver agrupadas las ***query*** **y** ***mutations*** dentro de un ***schema*****,** y a su vez los distintos ***type*** dentro de cada ***Query*** **y** ***Mutation*****.** Las relaciones direccionales que se aprecian en el esquema simbolizan el uso de una entidad dentro de otra.
Se puede apreciar que existe un *type* por cada uno de los modelos ajenos a la recomendación que se han especificado en apartados previos, salvo por la diferencia de que algunos de sus atributos no se muestran en estos tipos. Esto significa que esos atributos quedan ocultos para los clientes externos al *back-end*.

17.2.3.1. QUERY

A lo largo del diagrama mostrado anteriormente, se han establecido las siguientes query:

- **receivedFriendRequests**. Devuelve una lista de **FriendRequestType** mostrando las solicitudes de amistad recibidas del usuario.
- **sentFriendRequests.** Devuelve una lista de **FriendRequestType** mostrando las solicitudes de amistad enviadas del usuario.
- **myFriends**. Obtiene los amigos del usuario.
- **users.** Devuelve todos los **UserType** creados en la plataforma.
- **me.** Devuelve el **UserType** de la actual sesión.
- **allPlans.** Devuelve una lista de los **PlanType** futuros visibles del sistema. En el caso de que sea diario, devuelve los pasados.
- **detailedPlan.** Devuelve la información de único **PlanType**.
- **recommendedOrSearch.** Devuelve una lista de los **PlanType** recomendados o aquellos que coincidan con la cadena de texto buscada.

17.2.3.2. MUTATIONS

A lo largo del diagrama mostrado anteriormente, se han establecido las siguientes *mutations*:

- **editProfile.** Permite editar el perfil de usuario.
- **sendFriendRequest.** Envía una solicitud de amistad al usuario introducido.
- **acceptFriendRequest.** Acepta una solicitud de amistad previamente creada.
- **rejectFriendRequest.** Rechaza una solicitud de amistad previamente creada.
- **createPlan.** Crea un plan, marcando como dueño al usuario que lo crea.
- **editPlan.** Edita un plan, actualizando todos los atributos modificados.
- **deletePlan.** Edita un plan únicamente si la sesión activa es el dueño del plan.
- **participateInPlan.** Apunta al usuario a un plan.
- **createUser.** Registra un nuevo usuario en el sistema.
- **tokenAuth.** Obtención del *token* de sesión a través de las credenciales de usuario.
- **verifyToken.** Verifica si el *token* proveído sigue activo.
- **refreshToken.** Actualiza el *token* de sesión, generando uno nuevo.

17.3. DISEÑO DEL FRONT-END

En este apartado se desglosan las diferentes decisiones de diseño tomadas a lo largo del desarrollo del *front-end* de la aplicación. Es decir, el ***cliente Android*** mencionado anteriormente.
La estructuración de este proyecto consta de un módulo encargado de gestionar las ***sesiones,*** así como las **diferentes peticiones hacia el *back-end*** y, por otro lado, la implementación de los **diferentes *casos de uso*.**

17.3.1. COMPONENTES ANDROID

A continuación, se muestran los diferentes componentes del sistema operativo *Android* utilizados a lo largo de este proyecto. Es necesaria la comprensión de los principales componentes de este sistema para entender el diseño que se muestra en los siguientes apartados.

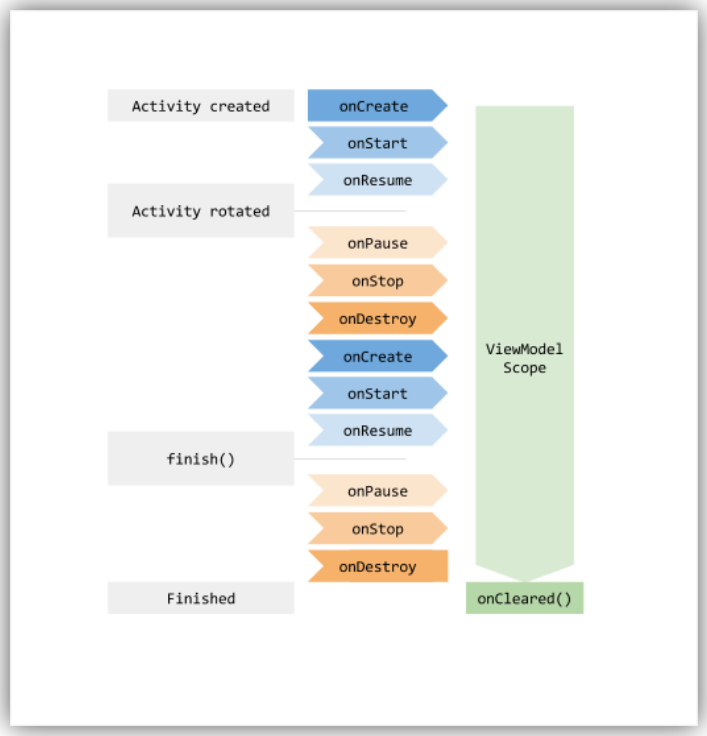


ILUSTRACIÓN 32. CICLO DE VIDA DE ACTIVIDADES Y VIEWMODEL [57]

- En la imagen anterior, se pueden apreciar las ***actividades*,** que se encargan de **mostrar las *vistas*** y gestionar los diferentes recursos de esta, así como interactuar con el **ViewModel.**
- El **ViewModel,** por otro lado, almacenar la información de la interfaz de tal forma que se persista a lo largo del ciclo de vida de las diferentes ***actividades*.**
- Además, se cuenta con los ***fragmentos*** que cumplen la misma funcionalidad que las ***actividades*,** sólo que se pueden agregar varios ***fragmentos*** a una única actividad.
- Por último, contamos con los archivos ***layout*** que permiten *renderizar* las diferentes vistas de la aplicación y enlazar sus atributos con las diferentes *actividades y fragmentos*.

17.3.2. DIAGRAMA DE PAQUETES

A continuación, se muestra el diagrama de paquetes empleado por la aplicación, así como sus dependencias.

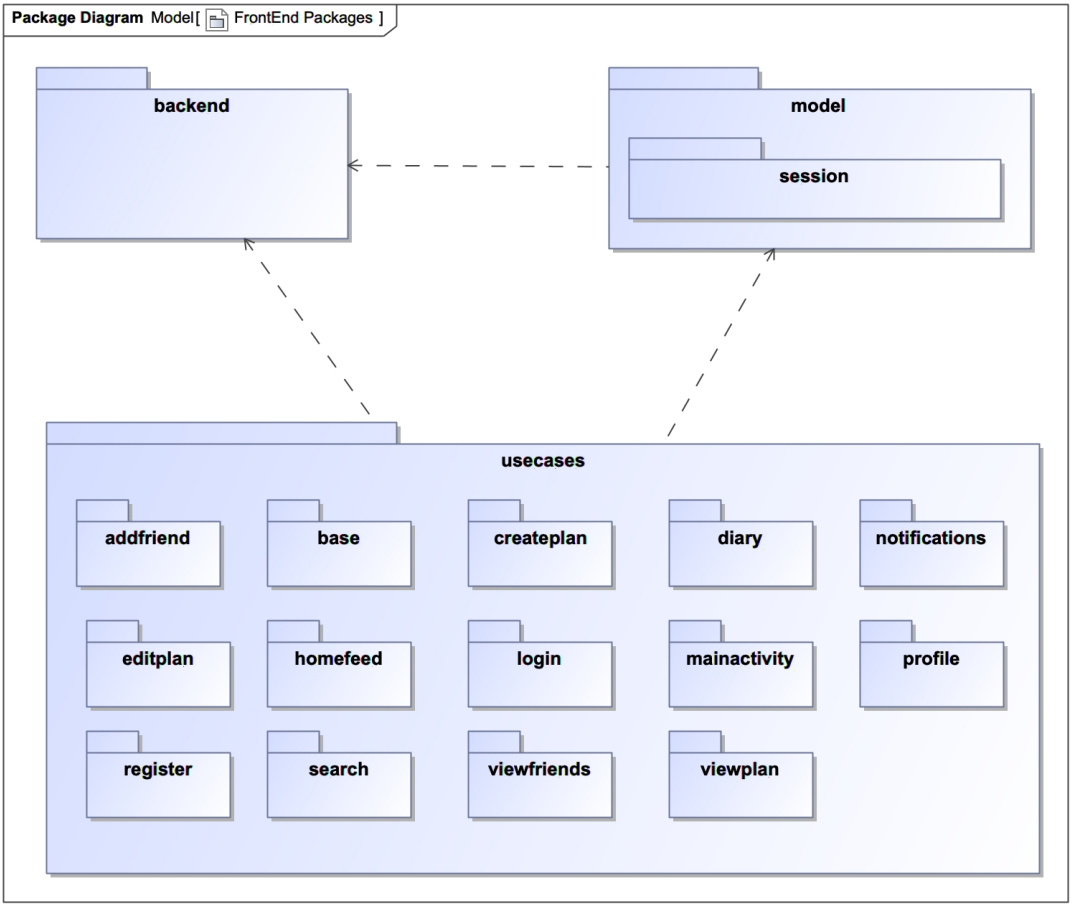


ILUSTRACIÓN 33. DIAGRAMA DE PAQUETES DEL FRONT-END. ELABORACIÓN PROPIA

Tal y como se puede apreciar, el sistema consta de tres paquetes principales, **backend, model y usecases.**

- El paquete **backend** se encarga de albergar las diferentes clases que se conectan con el servidor de la aplicación, así como la gestión de los modelos generados automáticamente por el *cliente Apollo.*
- El paquete **model.session** se encarga de albergar la sesión activa del usuario así como las diferentes funcionalidades. También se encargará de realizar el inicio de sesión dentro de la aplicación.
- El paquete **usecases** se encarga de implementar por separado y de forma independiente cada uno de los casos de uso. La única dependencia que tienen los casos de uso es con los paquetes de **backend** y **sesión,** al ser necesarias las gestiones de estas funcionalidades para la implementación de los casos de uso. Por otro lado, dentro de cada caso de uso podemos encontrar:
    - La ***actividad*** o ***fragmento*** donde se ejecutará cada caso de uso.
    - El ***viewmodel*** correspondiente a cada actividad o fragmento para almacenar la lógica.
    - Un ***adapter*** en el caso de que se esté mostrando una lista en la ***actividad.***
    - Un ***launcher*** que permite lanzar la actividad para el caso de las ***actividades.***

17.3.3. DIAGRAMAS DE CLASES

A lo largo de este apartado se muestran los diferentes diagramas de clases contenidos dentro de cada uno de los paquetes mencionados anteriormente. Cabe decir que debido al elevado número de clases no se mostrarán las relaciones entre las clases de diferentes paquetes.

17.3.3.1. DIAGRAMA DE CLASES BACKEND

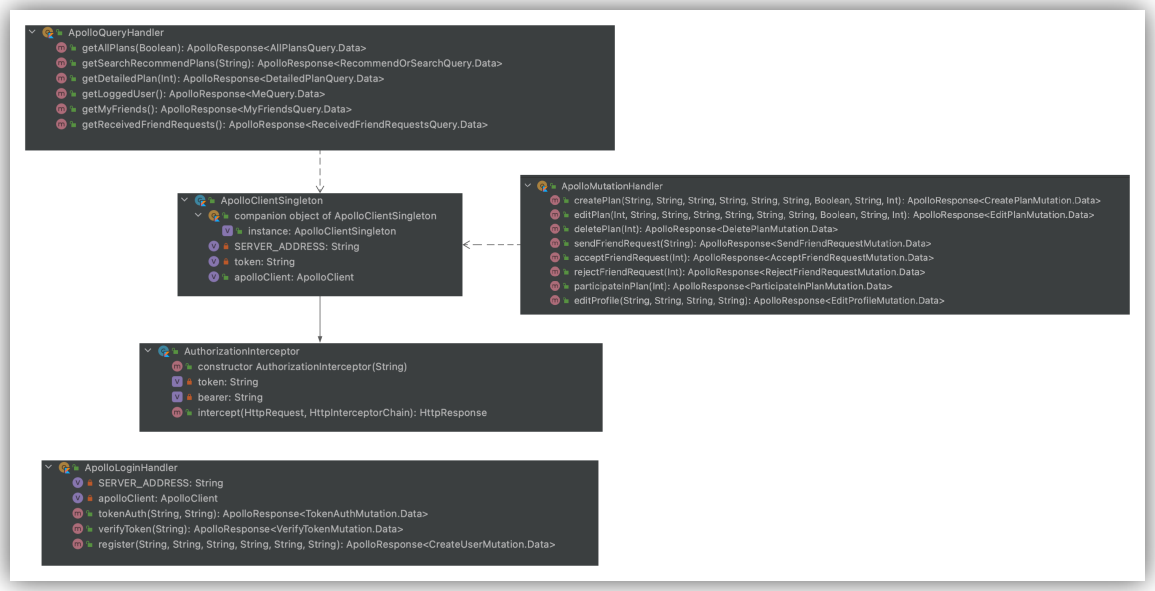


ILUSTRACIÓN 34. DIAGRAMA DE CLASES DEL PAQUETE BACKEND. ELABORACIÓN PROPIA

En este diagrama de clases podemos observar dos ***handler*** para las ***Query*** **y** ***Mutations*** del *back-end.* Estas clases se encargan de realizar las llamadas de cada una de estas operaciones a través del **ApolloClientSingleton**, el cual se encarga de mantener un **único cliente apolo** de cara a realizar las diferentes llamadas al servidor.

Además, se ha creado un **ApolloLoginHandler** para poder realizar el inicio de sesión antes de crearse la instancia ***singleton*** [58] del cliente.

17.3.3.2. DIAGRAMA DE CLASES SESSION

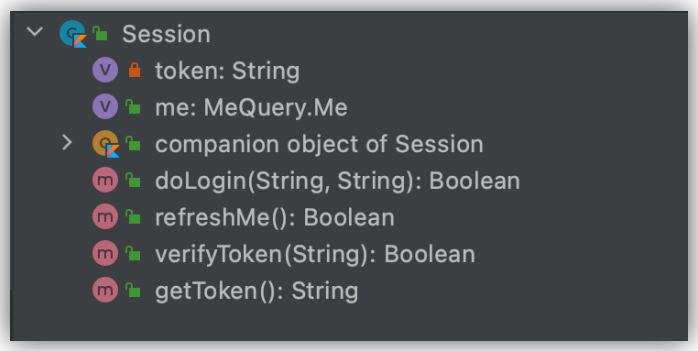


ILUSTRACIÓN 35. DIAGRAMA DE CLASES DEL PAQUETE SESSION. ELABORACIÓN PROPIA

Como se puede apreciar se tiene una única clase sesión con un objeto interno que permite ejecutar el patrón ***singleton*** de nuevo en *Kotlin*.

Esta clase se encarga de gestionar los diferentes aspectos de las sesiones, así como almacenar el estado del usuario.

17.3.3.3. DIAGRAMA DE CLASES BASE

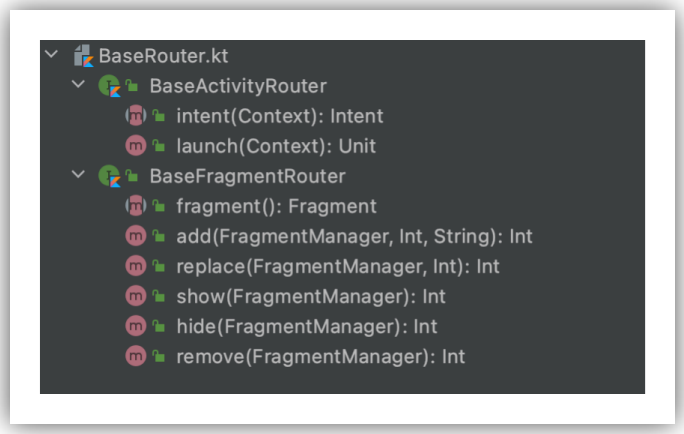


ILUSTRACIÓN 36. INTERFACES DE LOS ROUTERS DEL PAQUETE BASE. ELABORACIÓN PROPIA

En este paquete se han creado las diferentes interfaces de los ***router*** de actividades y fragmentos. Estos ***routers*** permiten lanzar estos componentes de forma sencilla evitando la duplicidad de código.

17.3.3.4. DIAGRAMA DE CLASES MAIN ACTIVITY

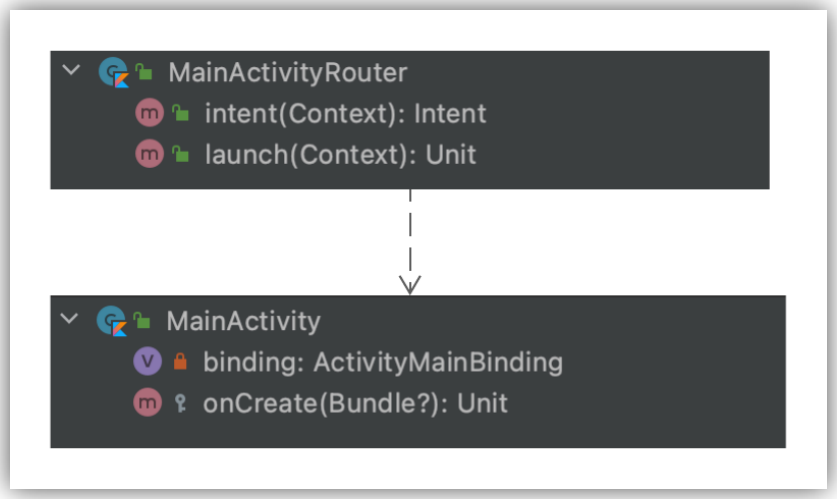


ILUSTRACIÓN 37. DIAGRAMA DE CLASES DEL PAQUETE MAINACTIVITY. ELABORACIÓN PROPIA

Este paquete se encarga de mostrar la navegación ***bottomview*** que incluye los casos de uso de *feed* de planes, amigos y búsqueda.

17.3.3.5. DIAGRAMA DE CLASES REGISTER

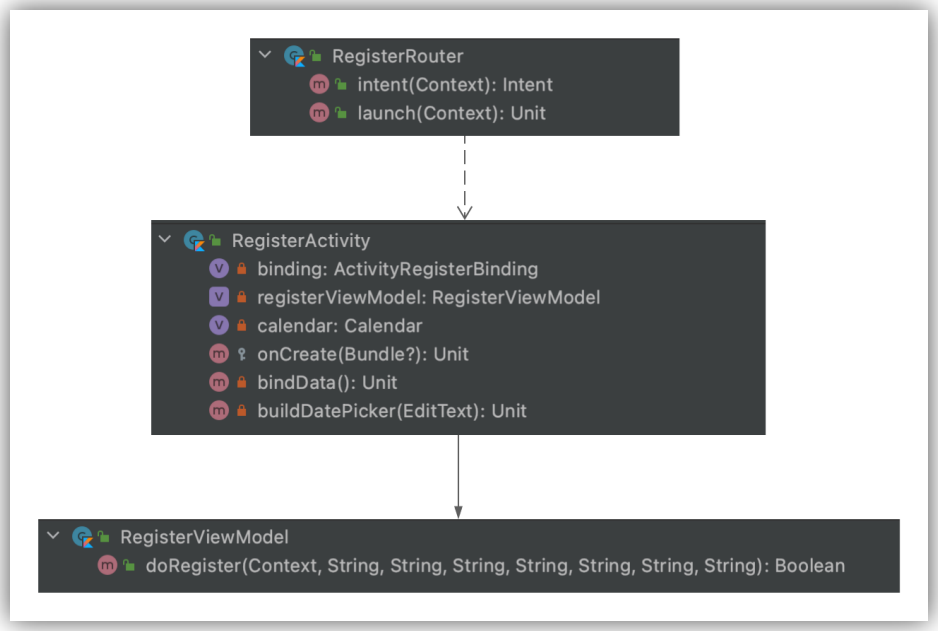


ILUSTRACIÓN 38. DIAGRAMA DE CLASES DEL PAQUETE REGISTER. ELABORACIÓN PROPIA

En este diagrama se puede observar la **actividad de registro** la cual tiene un ***viewmodel*** para realizar la función de **registro.**

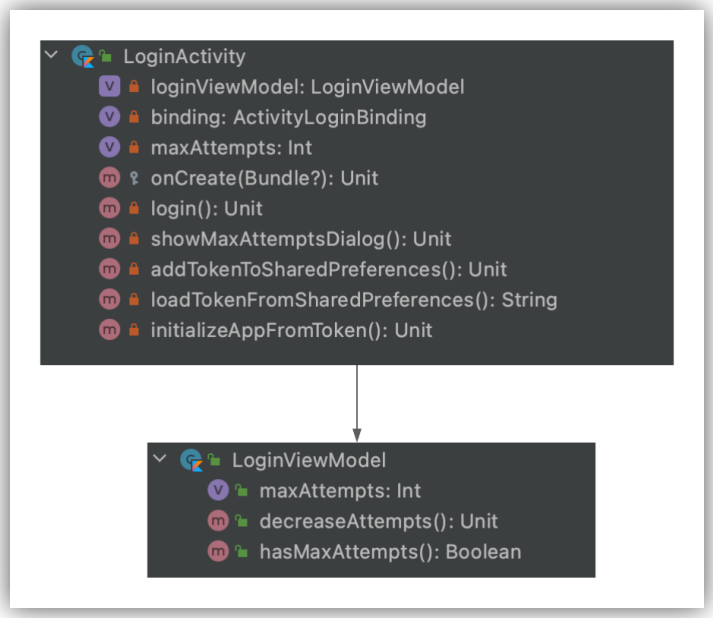


ILUSTRACIÓN 39. DIAGRAMA DE CLASES DEL PAQUETE LOGIN. ELABORACIÓN PROPIA

En este diagrama se puede apreciar que el ***viewmodel*** únicamente se encargará de la gestión de los intentos del usuario. Esto se ha diseñado de esta manera para facilitar el inicio de sesión automático.

Por otro lado, este paquete no dispone de ***router*** debido a que es la actividad principal del sistema.

17.3.3.7. DIAGRAMA DE CLASES HOMEFEED

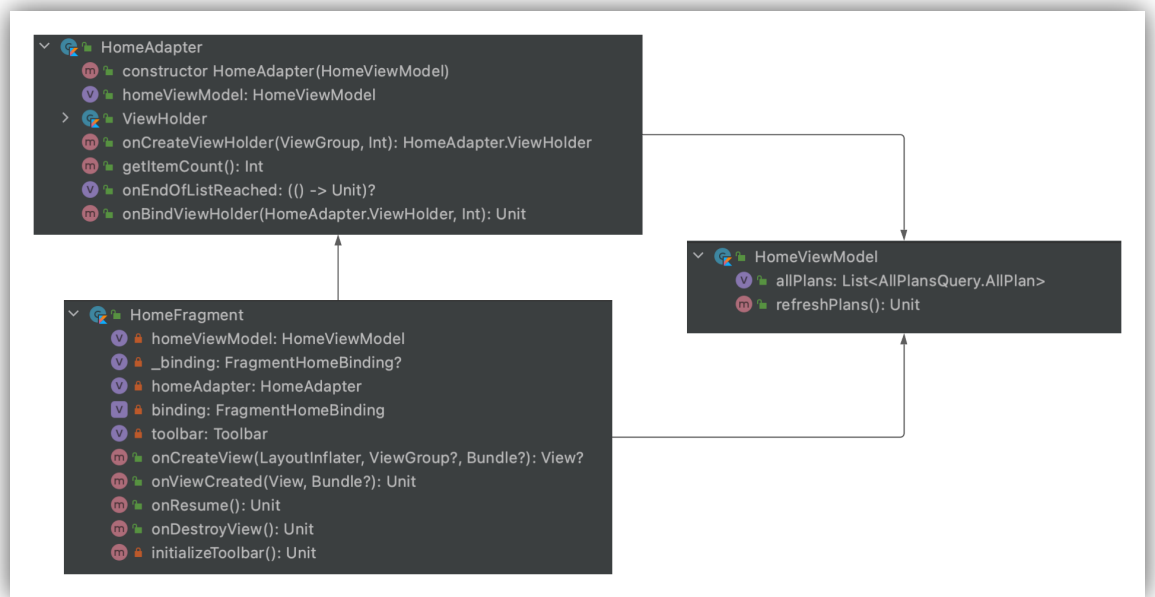


ILUSTRACIÓN 40. DIAGRAMA DE CLASES DEL PAQUETE HOMEFEED. ELABORACIÓN PROPIA

En este diagrama, a diferencia de los anteriores, podemos ver que se crea un fragmento que tiene un ***adapter*** de cara a mostrar los diferentes **planes** en forma de lista **a través del HomeViewModel.**

17.3.3.8. DIAGRAMA DE CLASES SEARCH

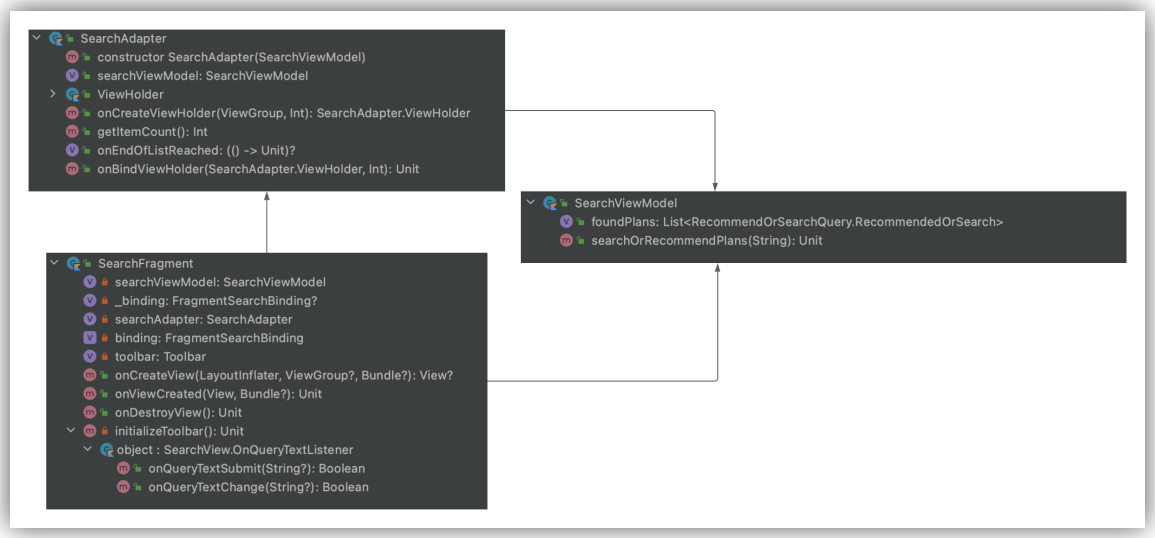


ILUSTRACIÓN 41. DIAGRAMA DE CLASES DEL PAQUETE SEARCH. ELABORACIÓN PROPIA

Se puede observar que este diagrama es similar al anterior, sin embargo, existe la diferencia de que se ha añadido la posibilidad de realizar ***querys*** en el ***SearchFragment.***

17.3.3.9. DIAGRAMA DE CLASES VIEWFRIENDS

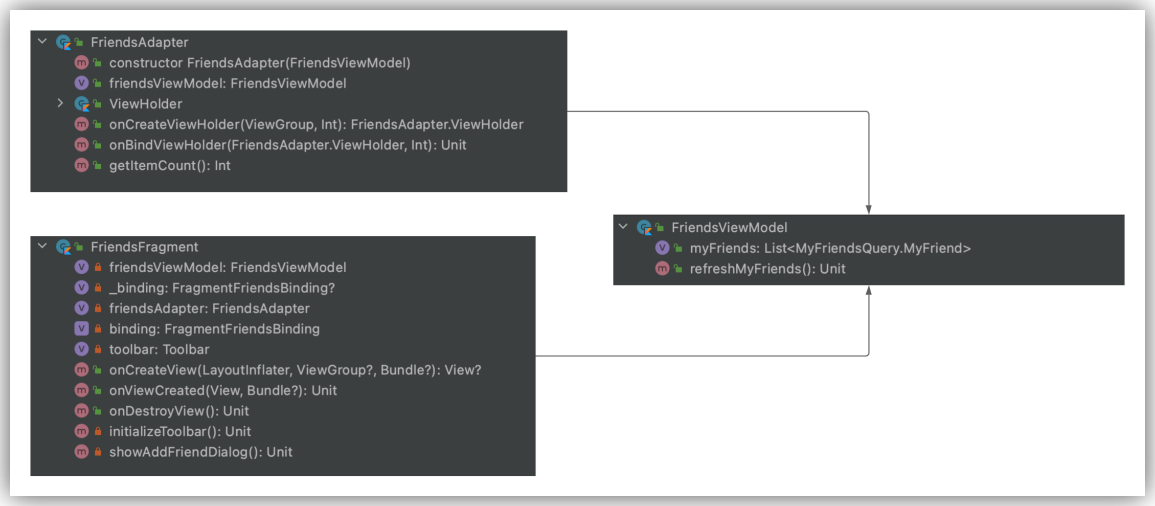


ILUSTRACIÓN 42. DIAGRAMA DE CLASES DEL PAQUETE VIEWFRIENDS

Se puede observar que este diagrama es similar a los dos anteriores, con la peculiaridad de que este alberga la estructura de los diferentes amigos.

17.3.3.10. DIAGRAMA DE CLASES VIEWPLAN

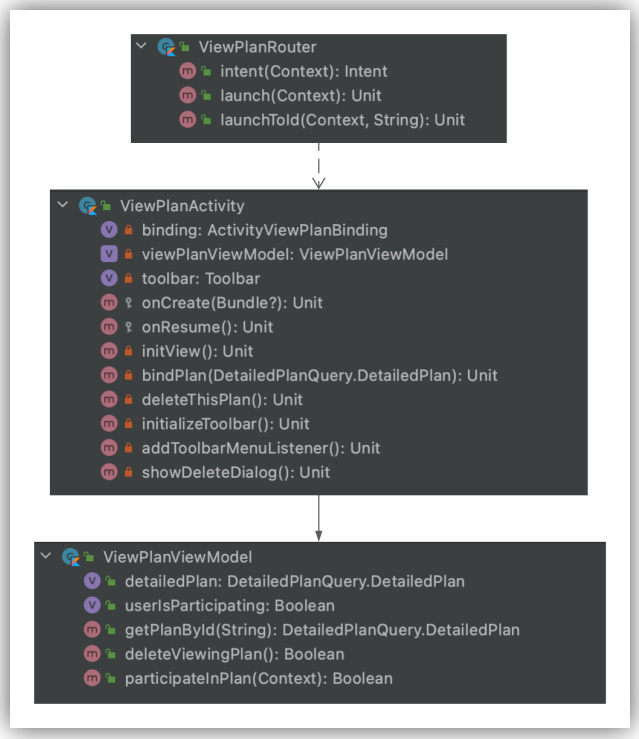


ILUSTRACIÓN 43. DIAGRAMA DE CLASES DEL PAQUETE VIEWPLAN. ELABORACIÓN PROPIA

Se puede observar en el diagrama que el ***router*** contiene una función extra *launchToId* al ser requerida la ***ID*** de un plan en específico para poder inicializar esta actividad.

17.3.3.11. DIAGRAMA DE CLASES CREATEPLAN

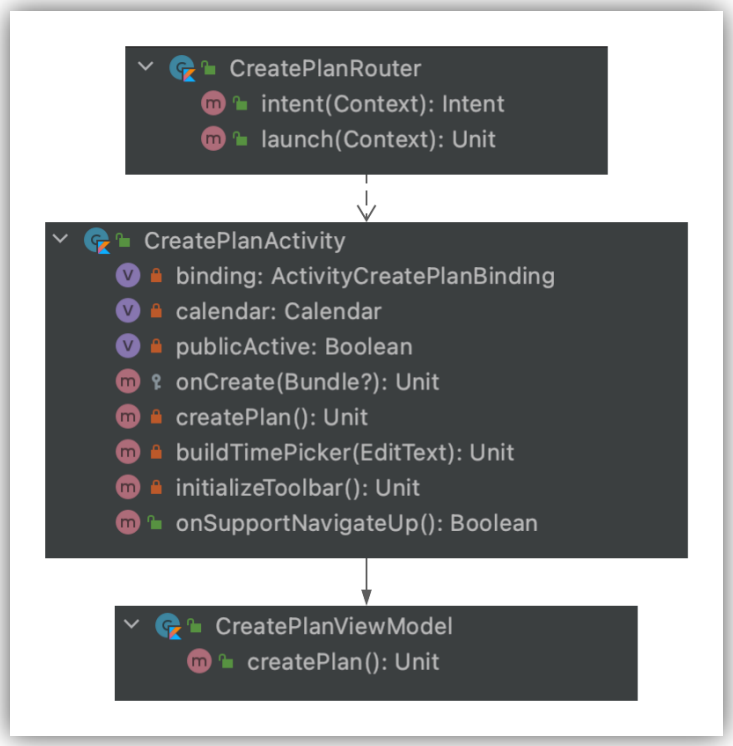


ILUSTRACIÓN 44. DIAGRAMA DE CLASES DEL PAQUETE CREATEPLAN. ELABORACIÓN PROPIA

Se puede observar que mantiene un diseño similar a las actividades anteriores, permitiendo crear un nuevo plan.

17.3.3.12. DIAGRAMA DE CLASES EDITPLAN

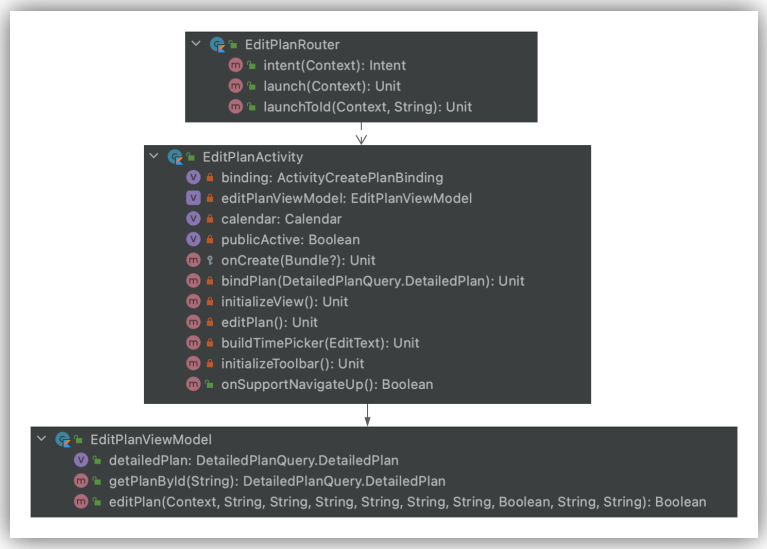


ILUSTRACIÓN 45. DIAGRAMA DE CLASES DEL PAQUETE EDITPLAN. ELABORACIÓN PROPIA

Se puede observar que este diagrama es una mezcla del diagrama de **viewplan y createplan.** Añadiendo la función *launchToID* al ***router***, así como permitiendo al **viewmodel** precargar el plan sobre el que se va a realizar la edición.

17.3.3.13. DIAGRAMA DE CLASES ADDFRIEND

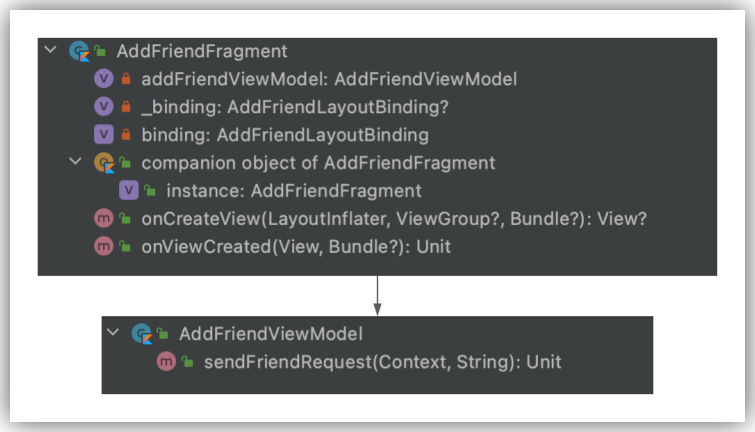


ILUSTRACIÓN 46. DIAGRAMA DE CLASES DEL PAQUETE ADDFRIEND. ELABORACIÓN PROPIA

Se puede observar que este diagrama no dispone de ***router***, esto se debe a que el diseño de estas clases está integrado dentro del caso de uso **viewfriends.**

17.3.3.14. DIAGRAMA DE CLASES NOTIFICATIONS

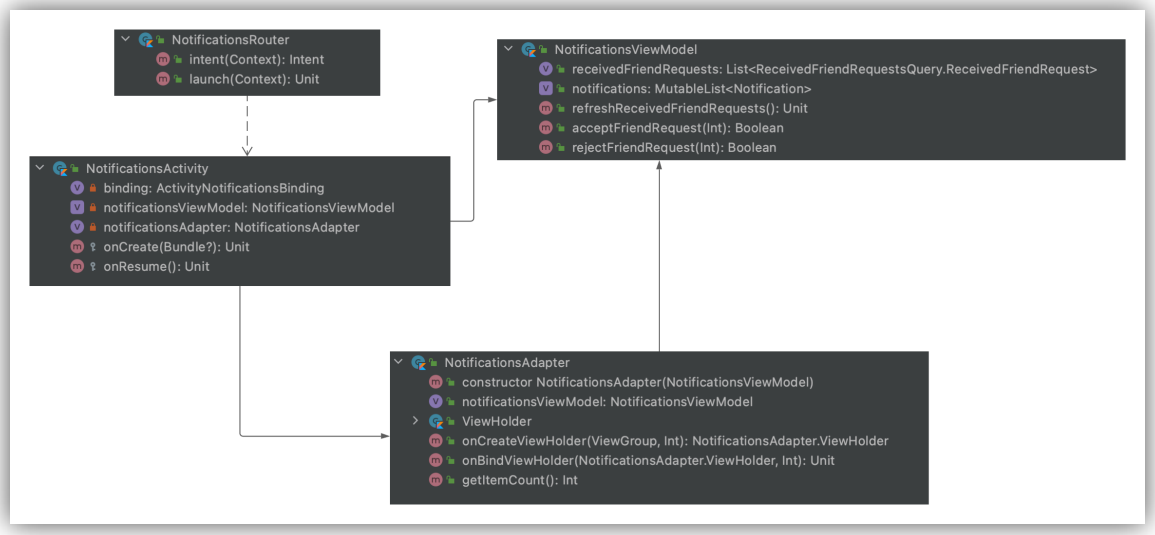


ILUSTRACIÓN 47. DIAGRAMA DE CLASES DEL PAQUETE NOTIFICATIONS. ELABORACIÓN PROPIA

En este caso, podemos ver que el diseño es similar a los ***diferentes feeds*** con la peculiaridad de que se incluye un ***router*** para poder navegar hacia esta actividad.

17.3.3.15. DIAGRAMA DE CLASES DIARY

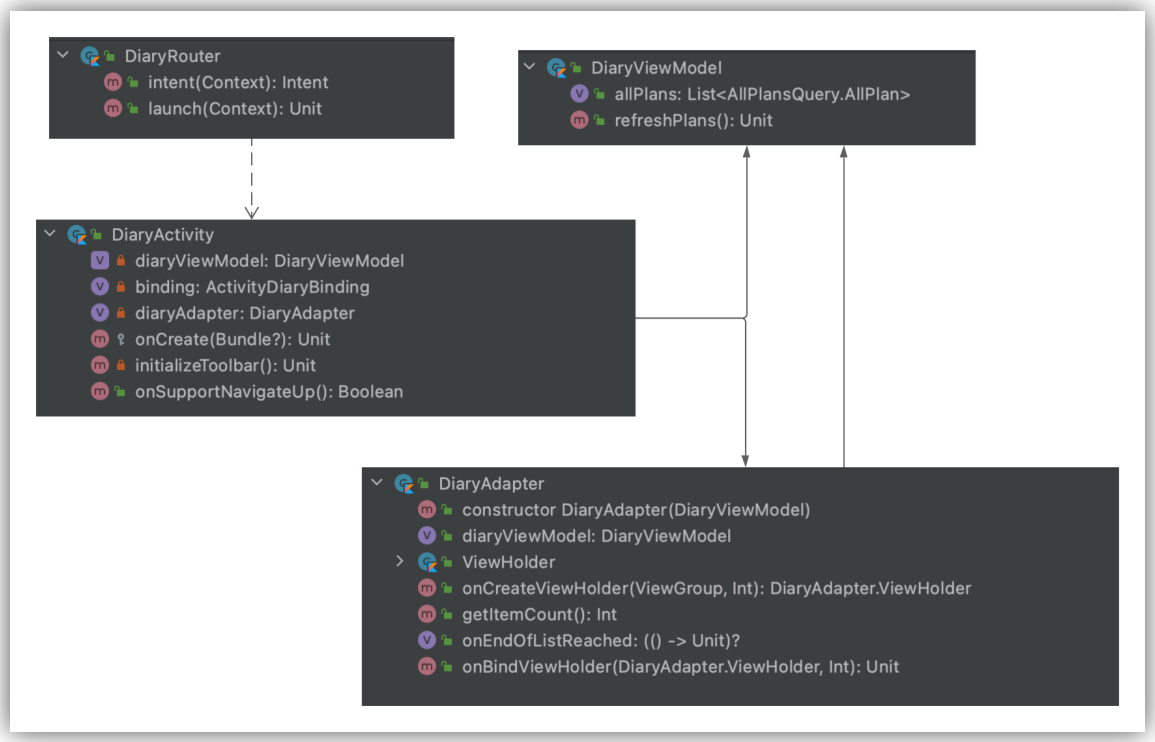


ILUSTRACIÓN 48. DIAGRAMA DE CLASES DEL PAQUETE DIARY. ELABORACIÓN PROPIA

En el diagrama mostrado se puede apreciar que este diagrama de clases es similar al del **homefeed** con la diferencia de que alberga una ***activity*** en lugar de un ***fragment.***

17.3.3.16. DIAGRAMA DE CLASES PROFILE

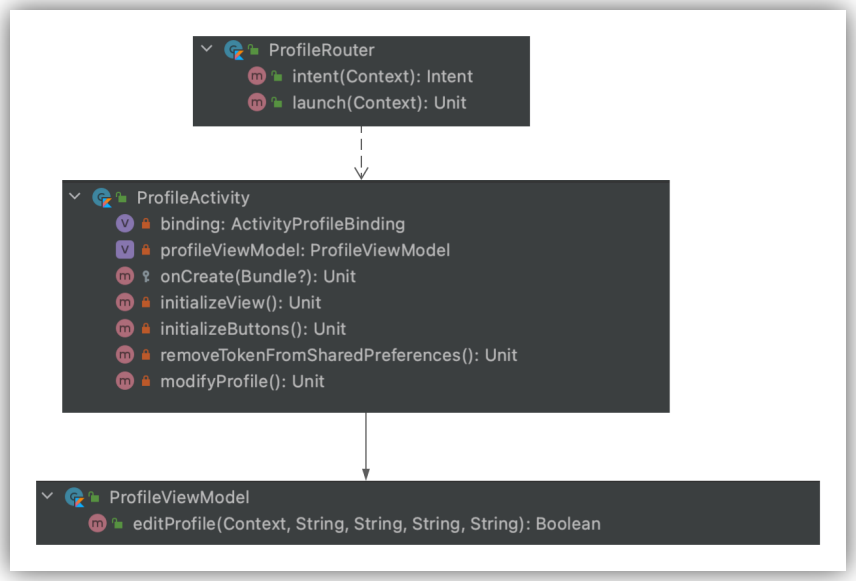


ILUSTRACIÓN 49. DIAGRAMA DE CLASES DEL PARQUETE PROFILE. ELABORACIÓN PROPIA

En el diagrama mostrado no se puede apreciar la precarga del usuario pues éste esta almacenado en el ***singleton sesión.*** Por otro lado, el ***viewmodel*** permite **editar el perfil.**

17.3.4. DIAGRAMA DE NAVEGACIÓN

La navegación del sistema se ha diseñado en función de los ***casos de uso***, tratando de seguir la misma estructura definidas en los mismos con el fin de guiar al usuario.

A continuación, se muestra el diagrama de navegación del ***cliente Android*:**

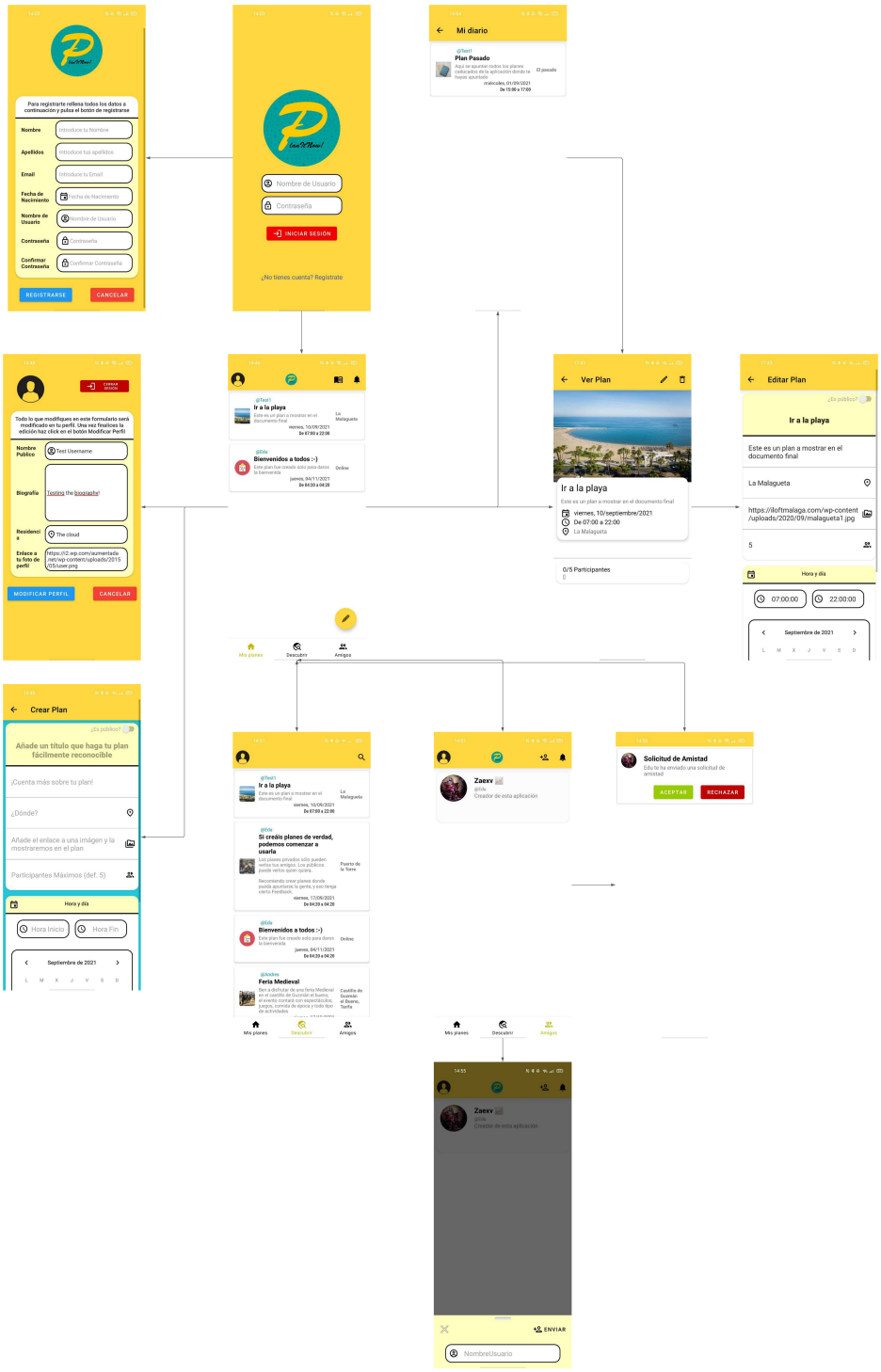


ILUSTRACIÓN 50. DIAGRAMA DE NAVEGACIÓN FRONT-END. ELABORACIÓN PROPIA

17.4. DISEÑO DE LA INTEGRACIÓN

Tal y como se ha mencionado en anteriores apartados, **la librería *Apollo*** facilita sustancialmente la integración de las distintas llamadas en el lenguaje ***GraphQL*** dentro del cliente ***Android*.**

Por esta misma razón, se puede resumir toda la integración del sistema a través del siguiente diagrama de secuencia:

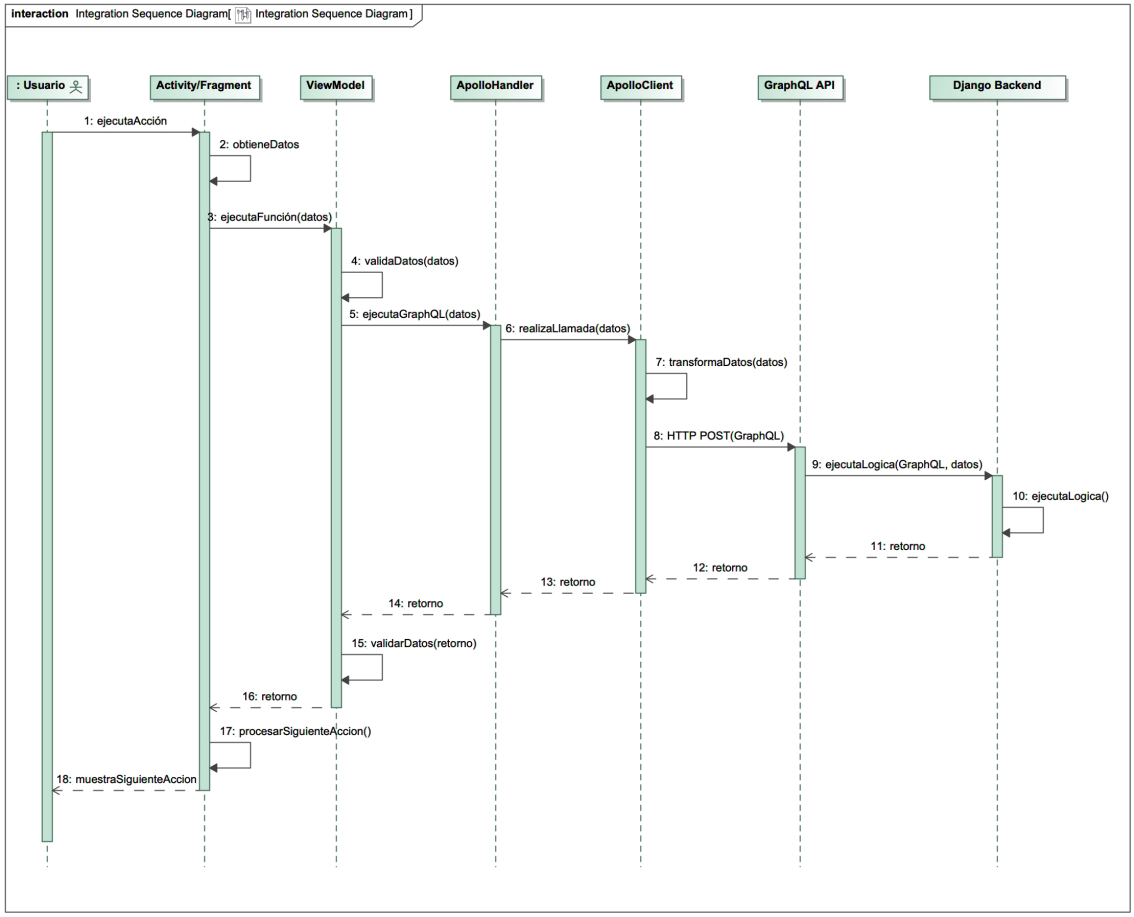


ILUSTRACIÓN 51. DIAGRAMA DE SECUENCIADE LA INTEGRACIÓN. ELABORACIÓN PROPIA

Como se puede observar, en el caso del ***front-end*, son el *viewmodel*, el *handler* y el *client*** los que realizan la llamada, mientras que el ***back-end*** procesa los diferentes datos. Una vez estos datos están procesados, el ***viewmodel*** validará esos datos y se los devolverá a la actividad, la cual **procesará la siguiente acción** según los datos devueltos, para finalmente mostrar esa información al usuario.

17.5. DISEÑO DEL ALGORITMO DE RECOMENDACIÓN

A lo largo de este apartado se desglosa el diseño del **algoritmo recomendación.** Pese a que está contenido dentro del ***back-end***, se considera muy importante el documentar el diseño de manera externa a este, puesto que es un algoritmo independiente.

De cara a entender la elección del algoritmo de recomendación que se muestra a continuación, es importante conocer los tipos de *algoritmos de recomendación que existen* [59]:

- **Métodos de Filtrado Colaborativo.** Los métodos colaborativos para sistemas de recomendación son aquellos que están basados únicamente en las interacciones registradas entre usuarios y objetos así una **matriz de interacciones usuario-objeto.** La idea principal de este tipo de algoritmos de recomendación es que las interacciones entre usuarios y objetos son suficientes para predecir usuarios similares, recomendando el contenido común. Además, este tipo de algoritmos cuenta con dos subcategorías:
    - **Basados en Memoria.** Trabajan directamente con los valores de las interacciones registradas sin necesidad de crear un modelo previo, centrándose en buscar únicamente el vecino más cercano. Por ejemplo, encontrar el usuario más cercano al que nos interesa y recomendarle aquello que le ha gustado.
    - **Basados en Modelos.** Se asume que existe un modelo el cual explica cómo los usuarios interaccionan con los objetos para finalmente crear las predicciones.
- **Métodos basados en Contenido.** Por el contrario de los métodos anteriores, además de utilizar las interacciones de los usuarios, este tipo de métodos utilizan las características de los objetos para realizar estas recomendaciones. Por ejemplo, si a un usuario le gusta una película, se le recomendará aquella que tenga un género similar. La mayor ventaja de estos algoritmos es que requieren de menos información para funcionar correctamente respecto aquellos basados en filtrado colaborativo.
- **Métodos Híbridos.** Son aquellos métodos que mezclan los métodos basados en contenido y basados en un filtrado colaborativo.

El algoritmo de recomendación desarrollado es un **algoritmo de recomendación basado en contenido** [60] es decir, este algoritmo recomienda a los usuarios basándose únicamente en el contenido de la aplicación.

Este algoritmo de recomendación está basado en la ***Ingeniería del Lenguaje Natural***, con lo cual el sistema está diseñado en función de las diferentes palabras que se han introducido en los diferentes planes.

Se ha escogido algoritmo ha sido por diversas causas. La principal ha sido la de explorar **técnicas de procesamiento del lenguaje natural** y **aplicarlas a algoritmos de recomendación** mediante una aproximación básica basada en *lemas*. Por otro lado, se ha escogido por la gran fortaleza de poder comenzar a recomendar sin necesidad de tener una gran cantidad de datos como sería en el caso del **filtrado colaborativo**, pudiendo recomendar planes desde el primer momento en el que la aplicación ha sido lanzada.

Este algoritmo cuenta con dos fases:

- **Fase de Obtención de Datos.** Esta fase se centra en obtener los diferentes ***lemas***[13] de los diferentes planes, así como traducirlos, para finalmente establecer las distancias entre los diferentes planes y usuarios.
- **Fase de Recomendación.** En esta fase se asume que ya están todos los atributos necesarios calculados y se encarga de mostrar las recomendaciones personalizadas al usuario.

17.5.1. FASE DE OBTENCIÓN DE DATOS

Como se ha mencionado previamente, el algoritmo desarrollado está **basado en los diferentes contenidos de la aplicación.** Se han escogido **el título y la descripción de los planes** como el contenido sobre el que recomendar planes debido a que contienen toda la información relevante sobre estos.

Además, se han empleado diferentes técnicas como la ***lematización*** **y** ***eliminación de palabras vacías*** **(*stopwords*)** para reducir el vocabulario requerido por este algoritmo para funcionar, mejorando así su rendimiento y precisión.

Por último, ha sido necesario traducir los diferentes planes de cara **a permitir el procesado por las principales librerías de lenguaje natural**, así como para **internacionalizar el algoritmo de recomendación de la aplicación**.

A continuación, se enumeran y describen los diferentes pasos de cara a completar la fase de obtención de datos:

1. Se traducen todos los planes que no estén traducidos dentro del sistema y que tengan fecha futura al día de hoy.
2. Se almacenan las traducciones dentro de los atributos ***english_description*** y ***english_title*** de cada plan.
3. Partiendo de esta traducción se obtienen los ***lemas*** de cada uno de estos planes**.**
   3.1. Se establece todo el texto en minúscula.
   3.2. Se ***tokeniza*** el texto mediante la librería ***nltk*****.**
   3.3. Se eliminan las palabras redundantes **(*stopwords*[14]**) mediante la librería ***nltk*****.**
   3.4. Se ***etiquetan*** las diferentes palabras gracias a la librería ***nltk*****.**
   3.5. Se obtienen los diferentes ***sustantivos*** **y** ***verbos***.
   3.6. Se obtienen los ***lemas de estos sustantivos y verbos.***
4. Una vez se han obtenido los ***lemas***, se asignan a cada uno de los planes.
5. Se calcula la distancia entre **todos los planes del sistema**, siendo esta la siguiente:
   5.1. ***(número de lemas en común / número de lemas en total)***
6. Se almacena la distancia entre **todos los planes**.
7. Se calcula para **cada usuario del sistema** la distancia hacia **cada plan futuro visible por el usuario.** Para realizar el cálculo de esta distancia, se obtendrán los lemas almacenados de los planes que **el usuario ha creado** y de ahí se obtendrá el valor de la distancia. La fórmula es la siguiente:

[13] El lema se corresponde con la palabra que encabeza una entrada léxica en un diccionario. En el caso de los nombres y adjetivos, corresponde a su forma masculina singular; en el caso de los verbos, corresponde a la forma en infinitivo

[14] Es el nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc. que son filtradas antes o después del procesamiento de datos en lenguaje natural.

7.1. ***(número de lemas en común plan-usuario / número de lemas en total del usuario)***

8. Se almacena la distancia entre **los usuarios y los diferentes planes.**

17.5.2. FASE DE RECOMENDACIÓN

Una vez se obtiene la distancia entre los **usuarios y los diferentes planes**, recomendar al usuario significará **mostrar los planes ordenados en función de su distancia al usuario.** Debido a que se desea recomendar, únicamente se emplearán los planes futuros.

De este modo el usuario puede visualizar rápidamente los planes que se le han recomendado.

17.5.3. DISEÑO DE FUNCIONES

Tal y como se puede ver en el **diagrama de paquetes del *back-end***, el sistema de recomendación ha sido diseñado en un módulo aparte. De esta manera no solamente se pueden cambiar fácilmente las diferentes funciones que calculan las distancias y otros valores en función de los datos almacenados en el sistema, sino que, si se desea modificar íntegramente el algoritmo y los datos almacenados en un futuro, esto resulta posible sin necesidad de modificar el ***núcleo del sistema*.**

A continuación, se puede ver el diseño de la clase que encapsula todas las funciones de recomendación:

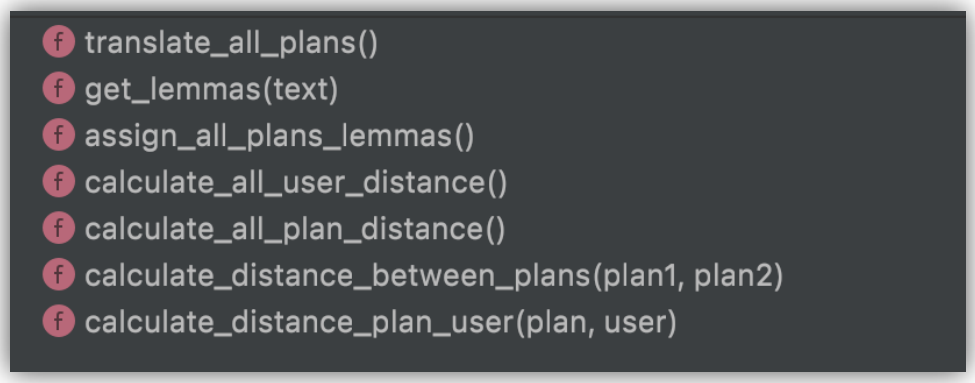


ILUSTRACIÓN 52.DISEÑO DE FUNCIONES DEL ALGORITMO DE RECOMENDACIÓN. ELABORACIÓN PROPIA

Se puede apreciar que se ha diseñado una función para cada uno de los pasos importantes dentro del algoritmo.

Estas funciones se han diseñado de cara a ser **ejecutadas por un *proceso automático*** durante las horas en las que haya el menor número de usuarios conectados. Por esa misma razón las diferentes funciones carecen de una clase padre puesto que se ejecutan individualmente, usando como parámetro de entrada los diferentes datos almacenados en el sistema.

18. DESAFÍOS TÉCNICOS

En este apartado se muestran las diferentes tareas que han supuesto un reto de cara a desarrollar el sistema. Estas tareas pueden haber supuesto un reto por diferentes motivos como la necesidad de probar diversas alternativas antes de encontrar una solución válida, así como la complejidad que el código pueda tener.

18.1. REFACTORIZACIÓN DE CÓDIGO A KOTLIN

Esta tarea ha supuesto un completo desafío debido a que fue necesario *refactorizar* todo el código de la aplicación a ***Kotlin.*** Especialmente supuso un gran desafío debido a que no solamente era necesario desarrollar estas modificaciones de código sino también su integración con la librería ***Apollo.***

A continuación, se muestran fragmentos del código en ***Kotlin*** que fue *refactorizado*.

```
import ...

public class HomeFragment extends Fragment {


    private static HomeViewModel homeViewModel;
    private static PlanAdapter planAdapter;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        homeViewModel =
                new ViewModelProvider(this).get(HomeViewModel.class);

        View root = inflater.inflate(R.layout.fragment_home, container,  attachToRoot: false);

        FloatingActionButton fab = root.findViewById(R.id.goto_createplanbutton);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = new Intent(root.getContext(), CreatePlanActivity.class);
                root.getContext().startActivity(i);
            }
        });

        homeViewModel.loadPlansFromGraphQL();

        initializePlanAdapter(root);
        return root;
    }
```

ILUSTRACIÓN 53. CÓDIGO JAVA PARA CREAR LA VISTA HOMEFRAGMENT. ELABORACIÓN PROPIA

```kotlin
class HomeFragment : Fragment() {

    private lateinit var homeViewModel: HomeViewModel
    private var _binding: FragmentHomeBinding? = null
    private lateinit var homeAdapter: HomeAdapter

    private val binding get() = _binding!!
    private lateinit var toolbar: Toolbar


    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        homeViewModel =
            ViewModelProvider( owner: this).get(HomeViewModel::class.java)


        homeAdapter = HomeAdapter(homeViewModel)

        _binding = FragmentHomeBinding.inflate(inflater, container,  attachToParent: false)
        val root: View = binding.root

        binding.listPlanRecyclerView.layoutManager = LinearLayoutManager(requireContext())
        binding.listPlanRecyclerView.adapter = homeAdapter
        binding.gotoCreateplanbutton.setOnClickListener() {  it: View!
            CreatePlanRouter().launch(requireContext())
        }

        return root
    }
```

ILUSTRACIÓN 54. CÓDIGO KOTLIN DE CARA A MOSTRAR LA VISTA HOMEFRAGMENT. ELABORACIÓN PROPIA

Tal y como se puede apreciar, el código ***Kotlin*** queda mucho más limpio, sin embargo, resulta un gran desafío el modificar todo el código sin perder funcionalidades ni trabajo desarrollado.

18.2. COMPRENSIÓN DE LAS CORRUTINAS DE KOTLIN

A lo largo del sistema ***Android*****,** es muy importante tratar de entender cómo funcionan las diferentes *corrutinas* de tal manera que se puedan cargar datos de servidores externos sin perjudicar al rendimiento de la aplicación. Inicialmente el código no se ejecutaba en estas *corrutinas* resultando en diversos bloqueos de la aplicación.

A continuación, se muestra un fragmento de código ejecutado dentro de una de estas corrutinas, concretamente una *corrutina* que se ejecuta **al menos si la actividad está en estado *resumed*.**

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    lifecycleScope.launchWhenResumed {  this: CoroutineScope
        initializeToolbar()
        homeViewModel.refreshPlans()
        homeAdapter.notifyDataSetChanged()
    }
}
```

ILUSTRACIÓN 55. EJEMPLO DE EJECUCIÓN DE UNA CORRUTINA. ELABORACIÓN PROPIA

18.3. DISEÑO DE LAS LLAMADAS AL BACK-END

El diseño de las diferentes llamadas al *back-end* supuso todo un desafío debido a que era necesario estructurarlas de tal forma que permitiesen validar el *token del usuario* y a su vez permitiesen mantener esta sesión siempre que el usuario estuviese *loggeado*. El diseño que se puede apreciar en apartados anteriores se concluyó tras cometer diversos errores que no sólo impedían el correcto desarrollo sino también generaban fallos en el sistema.

```
suspend fun editPlan(
    planID: Int,
    title: String,
    description: String,
    location: String,
    initHour: String,
    endHour: String,
    initDate: String,
    isPublic: Boolean,
    urlPlanPicture: String,
    maxParticipants: Int,
): ApolloResponse<EditPlanMutation.Data> {
    return ApolloClientSingleton.instance.apolloClient.mutate(
        EditPlanMutation(
            planId = planID,
            description = Optional.Present(description),
            title = Optional.Present(title),
            location = Optional.Present(location),
            initHour = Optional.Present(initHour),
            endHour = Optional.Present(endHour),
            initDate = Optional.Present(initDate),
            maxParticipants = Optional.Present(maxParticipants),
            isPublic = Optional.Present(isPublic),
            urlPlanPicture = Optional.Present(urlPlanPicture)
        )
    )
}
```

ILUSTRACIÓN 56. DISEÑO DE LA LLAMADA EDITPLAN. ELABORACIÓN PROPIA

18.4. INICIO DE SESIÓN AUTOMÁTICO

Pese a que implementar un inicio de sesión automático dentro de una aplicación móvil pueda parecer simple, esto ha supuesto un gran desafío técnico puesto que debido al diseño del *back-end* resultó muy complejo el decidir bajo qué situaciones se podía realizar este inicio de sesión automático.

```
private fun login() {
    var logged = false;
    lifecycleScope.launchWhenResumed {  this: CoroutineScope
        val user = binding.loginEmail.text.toString()
        val password = binding.loginPassword.text.toString()
        try {
            logged = Session.instance.doLogin(user, password)
        } catch (e: ApolloException) {
            Toast.makeText( context: this@LoginActivity, e.message, Toast.LENGTH_SHORT).show()
        }
        loginViewModel.decreaseAttempts()
        if (logged) {
            addTokenToSharedPreferences()
            MainActivityRouter().launch( activity: this@LoginActivity)
        } else {
            Toast.makeText( context: this@LoginActivity, "Tu nombre de usuario o contraseña son incorrectos",
                .show()
        }
    }
}
```

ILUSTRACIÓN 57. FUNCIÓN QUE REALIZA EL INICIO DE SESIÓN AUTOMÁTICO DENTRO DE LA APLICACIÓN MÓVIL. ELABORACIÓN PROPIA

18.5. DISEÑO DE LA LÓGICA DEL BACK-END

Debido a que inicialmente en este proyecto se pretendía delegar toda la lógica en las diferentes estructuras de ***Graphene*** para implementar ***GraphQL***, no se contempló la idea de delegar parte de esta funcionalidad en los propios modelos de ***Django.*** Por esta misma razón, realizar tareas como añadir amigos resultaron un completo desafío.

```python
class SendFriendRequest(graphene.Mutation):
    friend_request = graphene.Field(FriendRequestType)

    class Arguments:
        to_username = graphene.String(required=True)

    @staticmethod
    def mutate(self, info, to_username):
        logged_user = info.context.user
        if logged_user.is_anonymous:
            raise Exception("Not logged in!")
        to_user = UserProfile.objects.get(user__username=to_username)
        from_user = UserProfile.objects.get(pk=logged_user.id)

        if not to_user or not from_user:
            raise Exception("Error: Cannot find user to request")

        if FriendRequest.objects.filter(to_user=to_user, from_user=from_user) \
                or FriendRequest.objects.filter(to_user=from_user, from_user=to_user):
            raise Exception("Error: Friend Request Already Exists")

        friend_request = FriendRequest(to_user=to_user, from_user=from_user)
        friend_request.save()

        return SendFriendRequest(friend_request=friend_request)
```

ILUSTRACIÓN 58. MUTATION DE ENVÍO DE SOLICITUD DE AMISTAD. ELABORACIÓN PROPIA

18.6. DISEÑO VISUAL DE LA APLICACIÓN

Pese a que pueda parecer sencillo diseñar vistas para **una aplicación móvil,** no es una tarea tan sencilla pues influyen una gran cantidad de elementos a lo largo de desarrollar estas vistas. Entre estos elementos, podemos distinguir ***componentes, widgets y layouts.***

Realizar un diseño visual de la aplicación que encajase con la filosofía del proyecto y permitiese ofrecer una interfaz limpia ha supuesto todo un desafío técnico, debido a las limitaciones de estas.

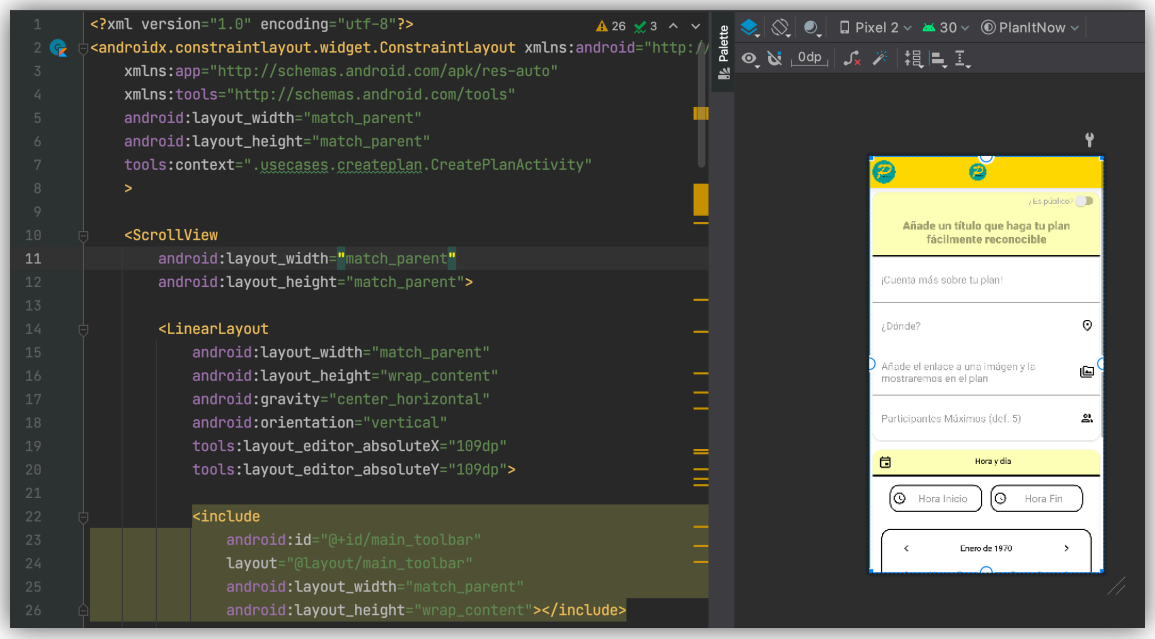


ILUSTRACIÓN 59. DISEÑO VISUAL DEL CASO DE USO CREAR PLAN. ELABORACIÓN PROPIA

Por otro lado, mantener la integridad de este diseño respecto a los ***mockups*** elaborados previamente, ha supuesto otro desafío al tener que respetar las decisiones de diseño tomadas.

18.7. ALGORITMO DE RECOMENDACIÓN

El algoritmo de recomendación ha supuesto un gran desafío técnico tanto a nivel de diseño como de implementación. Realizar un análisis para poder almacenar toda la estructura necesaria para realizar las diferentes recomendaciones, así como elaborar las complejas funciones de traducción y obtención de lemas ha supuesto todo un desafío técnico.

```python
def translate_all_plans():
    for plan in Plan.objects.all():
        print("\t Traduciendo plan...{plan}")
        try:
            english_title = GoogleTranslator(source='auto', target='en').translate(text=plan.title + "")
            english_description = GoogleTranslator(
                source='auto', target='en').translate(text=plan.description)
            print(english_description)
        except NotValidPayload:
            english_title = 'Not valid'
            english_description = 'Not valid'
        print(english_title)

        plan.english_title = english_title
        plan.english_description = english_description
        plan.save()


def get_lemmas(text):
    '''
    Lematiza el texto con la libreria nlp
    '''

    custom_stopwords = [
        'hi', '\n', '\n\n', '&amp;', ' ',
        '.', '-', 'got', "it's", 'it’s',
        "i'm", 'i’m', 'im', 'want', 'like',
        '$', '@', ' !', ' ?', '#', "'", '``',
        "n't", "'m", "'s", ':', '...', '!', '?', ',',
        'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
        'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z',
        'thanks', 'let', 'ta'
    ]
    text = text.lower()
    english_words = words.words()
    all_words = word_tokenize(text)

    is_noun = lambda pos: pos[:2] == 'NN'
    is_verb = lambda pos: pos[:2] == 'VB'
    is_verb_p = lambda pos: pos[:2] == 'VBP'
    nouns = [word for (word, pos) in nltk.pos_tag(all_words) if (is_noun(pos) | is_verb(pos) | is_verb_p(pos))]
```

ILUSTRACIÓN 60. FRAGMENTO DE LAS FUNCIONES DEL ALGORITMO DE RECOMENDACIÓN. ELABORACIÓN PROPIA

19. PRUEBAS

En este apartado se enumeran las diferentes pruebas realizadas al sistema. Cabe decir que las diferentes pruebas realizadas a la aplicación han sido **manuales** al disponer de un marco de tiempo limitado para desarrollar y probar la aplicación.

19.1. PRUEBAS BASADAS EN CASOS DE USO

Para cada caso de uso se ha ejecutado al menos **una prueba manual** una vez ha estado todo el sistema integrado para verificar que funciona correctamente. En este caso se han introducido valores permitidos.

19.2. PRUEBAS DE API

A lo largo del desarrollo se ha realizado una llamada a la *API GraphQL* por cada uno de *las **mutations** y **query*** implementadas.

A continuación, se muestra una imagen con las diferentes pruebas realizadas a través de la herramienta ***Insomnia.***

Mediante estas pruebas se ha verificado que el funcionamiento de la *API* es correcto, de cara a evitar fallos en el *front-end*.

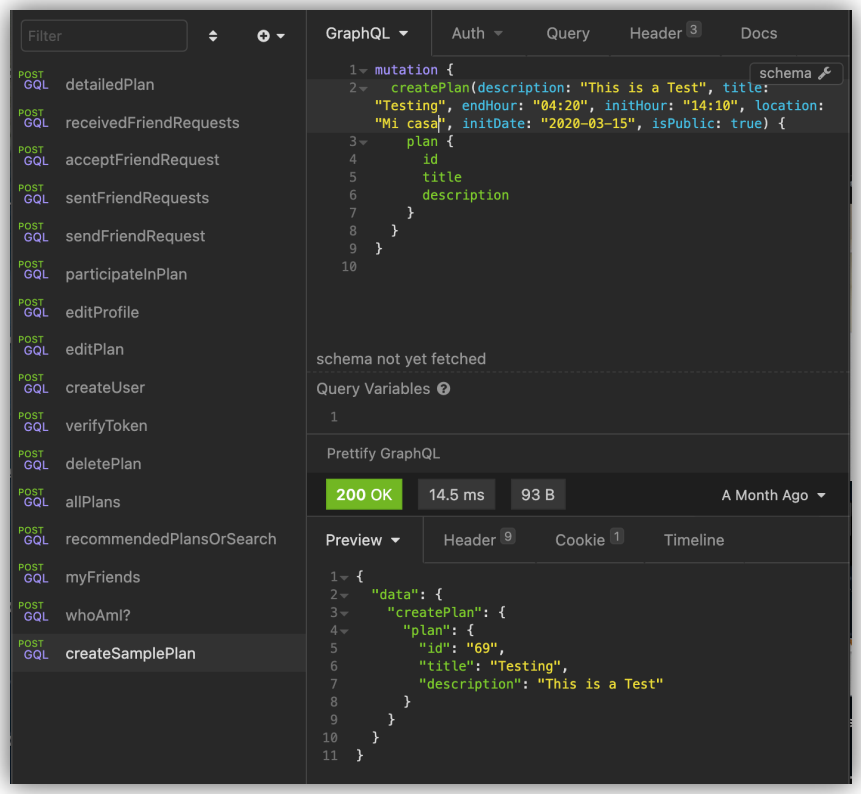


ILUSTRACIÓN 61. PRUEBAS DE API INTRODUCIDAS EN INSOMNIA. ELABORACIÓN PROPIA

19.3. PRUEBAS DE INTEGRACIÓN

Una vez se ha contado con las diferentes funcionalidades del sistema implementadas, se han realizado diferentes pruebas en la aplicación introduciendo esta vez valores erróneos y comprobando si la aplicación devolvía el resultado esperado basándose en los casos de uso.

19.4. PRUEBAS DE USUARIO

Gracias a haber lanzado los diferentes *MVP* en fases tempranas del proyecto, los diferentes usuarios pudieron realizar pruebas a la aplicación, encontrando un gran número de fallos y funcionalidades erróneas. Cada uno de estos fallos ha sido reportado por los diferentes usuarios, permitiendo así tanto encontrar como solucionar errores a una gran velocidad.

19.5. FALLOS DETECTADOS

A continuación, se muestra una lista de todos los fallos que se han detectado durante las etapas de pruebas:

- **Crear plan público.** Se detectó un fallo en el *switch* de crear plan público que no permitía a los usuarios crear planes públicos. Este fallo se ha corregido.
- **Mala visibilidad de planes.** En la vista ver plan resultaba complicado visualizar los diferentes atributos de los planes. Este fallo se ha corregido.
- **Vista de notificaciones bloqueante.** La vista de notificaciones se bloqueaba cada vez que cargaba. Este fallo se ha corregido.
- **Crear plan calendario erróneo.** Las fechas escogidas en el calendario de crear plan no coincidían con las fechas del plan creado. Este fallo se ha corregido
- **Crear plan H. Inicio y H. Fin invertidos.** La hora de inicio y hora de fin de los planes se invertían, creando planes inversos. Este fallo se ha corregido.
- **Mala visibilidad con móviles en modo oscuro.** El modo oscuro de los distintos dispositivos móviles impedía ver correctamente la aplicación. Actualmente el modo oscuro se ha desactivado en la aplicación.
- **URL de foto de plan no actualizándose en editar plan.** En el menú de editar plan, la URL de plan no se actualizaba. Este fallo ha sido corregido.
- **Planes con 0 participantes podían verse.** Los planes con 0 participantes como máximo podían ser vistos por otros usuarios, ahora estos planes son privados y únicamente el creador puede verlos.
- **Visibilidad de planes errónea.** El usuario podía ver más planes de los que le correspondía visualizar por lógica de negocio. Este fallo ha sido corregido.
- **Migraciones erróneas creando problemas de modelo.** El modelo se migró erróneamente creando diferentes problemas a la hora de recomendar. Este fallo ha sido corregido.

20. DIMENSIÓN DEL PRODUCTO

La aplicación desarrollada cuenta con tres repositorios actualmente albergados en ***GitHub*** [61]. Estos repositorios se pueden encontrar en los siguientes enlaces:

https://github.com/Zaexv/PlanItNow_frontend_old
https://github.com/Zaexv/PlanItNow_Android
https://github.com/Zaexv/BackEnd_PlanItNow

De cara a dimensionar el producto, se van a emplear las siguientes métricas:

- **Casos de Uso Implementados**
- **Número de Horas de Análisis y Desarrollo**
- **Número de *Commits***
- **Número de Líneas de Código**

Estas métricas contribuyen a hacerse una idea del esfuerzo que se ha puesto en desarrollo y ejecución de esta aplicación, así como la dimensión y capital humano del que se ha dispuesto.

20.1. CASOS DE USO IMPLEMENTADOS

A lo largo del proyecto se han implementado un total de **47** *casos de uso* que finalmente se traducen como funcionalidades que puede ejecutar el usuario dentro de la aplicación. Es una elevada cantidad de casos de uso si consideramos que este proyecto ha sido realizado por un **único desarrollador.**

20.2. NÚMERO DE HORAS DE ANÁLISIS Y DESARROLLO

Pese a que a lo largo del proyecto no se ha llevado un análisis exhaustivo del número de hora invertidas en el mismo, sí que se puede realizar una estimación precisa de esto.

El proyecto comenzó a ser desarrollado a finales de Mayo para ser finalizado a mediados de Septiembre. Esto son aproximadamente **cuatro meses.**

Aproximadamente, cada uno de estos meses se ha trabajado unos **20 días** en el proyecto, trabajando un promedio de **6 horas diarias** en el desarrollo e implementación de tareas y análisis.

Partiendo de esta base, obtenemos un total de 6 x 20 x 4 **= 480 horas** invertidas en el proyecto, superando sustancialmente el número de horas de un proyecto de fin de máster.

20.3. NÚMERO DE COMMITS

Cada uno de los *commits* en *GitHub* es una modificación en el código. Es importante tener esta métrica en cuenta puesto que, aunque finalmente puedan verse un número concreto de líneas de código, es importante ver cuántas modificaciones se han realizado sobre el mismo.

A continuación, se muestran el número de *commits* por cada uno de los repositorios de código:

- **PlanItNow_Android.** 48 *Commits*.
- **BackEnd_PlanItNow.** 59 *Commits*.
- **PlanItNow_Frontend_old.** 29 *Commits*.

Esto supone un total de **136** *commits* registrados por el control de versiones empleado.

20.4. NÚMERO DE LÍNEAS DE CÓDIGO

El número de líneas de código en la versión más reciente del sistema es sin dudarlo una de las mayores métricas de dimensión del producto pues es, literalmente aquello que define el artefacto final.

Language	files	blank	comment	code
Java	24	307	34	923
XML	31	68	23	921
Bourne Shell	1	21	22	129
Gradle	3	15	7	66
DOS Batch	1	23	2	59
ProGuard	1	3	18	0
SUM:	61	437	106	2098

ILUSTRACIÓN 62. MÉTRICAS DEL REPOSITORIO PLANITNOW FRONTEND OLD. ELABORACIÓN PROPIA

Language	files	blank	comment	code
Python	24	269	137	897
XML	5	0	0	41
SUM:	29	269	137	938

ILUSTRACIÓN 63. MÉTRICAS DEL REPISOTORIO BACKEND PLANITNW. ELABORACIÓN PROPIA

Language	files	blank	comment	code
XML	64	254	19	2687
Kotlin	49	481	24	2092
GraphQL	18	86	0	417
Bourne Shell	1	23	36	126
Gradle	3	11	4	75
DOS Batch	1	21	2	66
ProGuard	1	3	18	0
SUM:	137	879	103	5463

ILUSTRACIÓN 64. MÉTRICAS DEL REPOSITORIO PLANITNOW ANDROID. ELABORACIÓN PROPIA

Sumando el número total de líneas de código para cada uno de los repositorios, se suman un total de **8499 líneas de código** en la versión final del proyecto.

21. DESPLIEGUE

El proyecto ha sido desplegado a lo largo de los diferentes *MVP* en la plataforma ***Heroku*** [62], un servicio de computación en la nube que soporta diferentes lenguajes de programación.

De cara a realizar el despliegue, se ha decidido posicionar el ***back-end*** dentro de ese sistema debido a que esta plataforma es gratuita de cara al desarrollo de aplicaciones básicas y permite realizar las diferentes pruebas de usuario.

Otra de las principales razones para escoger esta plataforma ha sido la facilidad que éste ofrece de cara a realizar una **integración continua** de las diferentes características del sistema.

Por otro lado, **el artefacto *Android*** se ha compilado en formato ***.apk***, es decir, el formato de las diferentes aplicaciones de este sistema operativo para que los diferentes usuarios puedan instalarlo fácilmente en sus dispositivos. Debido a que los diferentes *MVP* de este proyecto han estado en fases de pruebas aún no se puede disponer de la aplicación de forma pública.

CONCLUSIONES

Durante esta sección del proyecto se realiza una síntesis de las conclusiones en los aspectos más relevantes del proyecto, de cara mostrar a modo de resumen los diferentes resultados de este trabajo de fin de máster, así como los diferentes aspectos de futuro a considerar de cara al proyecto emprendedor.

22. OBJETIVOS ALCANZADOS

En las primeras etapas de este proyecto, se definieron una serie de objetivos que resultarían relevantes para el correcto desarrollo y ejecución del mismo. Es por ello por lo que es necesario evaluar cuáles de ellos se han logrado cumplir. Gracias a haber establecido los objetivos mediante ***SMART* en el apartado 2 de este documento**, es sencillo evaluar cuáles de estos se han alcanzado.

Respecto a los **objetivos generales,** se podría decir que se han cumplido los **OG2 y OG3,** creando una herramienta que permita desconectar de las redes sociales de cara a realizar planes y recomendando estos a los usuarios. Por otro lado, el cumplimiento del **OG1** aún no se ha logrado alcanzar **puesto que este proyecto aún no ha sido lanzado de forma pública a los diferentes usuarios.**

Respecto a los **objetivos técnicos** se han logrado alcanzar todos los objetivos. Es decir, se ha creado una **aplicación Android** que está **desplegada en un servidor,** almacena los datos y cuenta **actualmente con más de 5 *early-adopters* activos.** Por otro lado, respecto al **OT5** durante el experimento del segundo MVP se mostró a diferentes dueños de negocios de ocio, obteniendo así la validación de la idea desarrollada.

De este modo, se puede concluir que se han alcanzado la mayoría de los objetivos considerando que el proyecto ha resultado un éxito tanto a nivel de desarrollo como a nivel de emprendimiento.

23. DIFICULTADES Y RETOS ENFRENTADOS

Posiblemente este apartado sea el más ambiguo dentro del marco de las conclusiones debido a que a lo largo de este proyecto se han enfrentado múltiples dificultades y retos ya que no ha consistido únicamente en el desarrollo técnico de una plataforma, sino que ha sido necesario evaluar una idea de negocio, así como tratar de indagar en *qué es lo que la gente quiere, qué es lo que las empresas quieren y qué es lo que se pretende lograr con este proyecto,* lo cual no es una tarea especialmente común entre los ***Ingenieros de Software***.

Por un lado, se ha contado con la gran dificultad de emprender y crear desde cero un producto, desde la gestación de la idea hasta la evaluación de viabilidad para finalmente poder materializar esta idea personal. Además, se ha contado con la especial dificultad de tener que tratar de *hacerse hueco* dentro de un mercado sobresaturado como es el de las aplicaciones móviles y logrando calzar toda la ideación del proyecto con el marco de los diferentes conocimientos adquiridos donde el máster.

Por otro lado, las diferentes restricciones temporales de este proyecto han sido el mayor resto enfrentado dentro de este proyecto. Desde el inicio de este trabajo ha existido la gran ambición de crear una *gran idea* dentro del menor tiempo posible. Esto ha resultado en tener que establecer prioridades de la forma más clara posible, un gran esfuerzo a nivel de trabajo y desarrollo, constantes tutorías con los tutores de este proyecto, así como la capacidad de trabajar bajo un nivel de presión más que común en el ámbito del emprendimiento. Afortunadamente se ha logrado desarrollar todo lo que inicialmente se había propuesto sin tener que cortar *demasiado contenido* respecto al producto final.

Finalmente, una gran dificultad enfrentada ha sido la de desarrollar un producto desde cero a nivel técnico. Lo cierto es que desarrollar una aplicación parece más que sencillo en el **mundo de las ideas de Platón,** sin embargo, cuando hay que tener en cuenta los diferentes requisitos del mercado, de los usuarios y de uno mismo como emprendedor, así como las limitaciones de las tecnologías donde se desea implementar todo esto, la complejidad crece de manera exponencial. Por esta misma razón resulta muy difícil tratar de calzar todas las ideas que se tienen dentro de una aplicación.

Es por ello por lo que realizar la arquitectura, diseño e implementación del sistema combinando todos los *ingredientes* mostrados a lo largo de este documento ha supuesto una gran dificultad.

24. EXPERIENCIA ADQUIRIDA

En aquellos proyectos en los que es el creador el que pone los límites es posiblemente en los que más se aprende. Este proyecto desde el comienzo ha sido completamente ambicioso solicitando un gran aprendizaje *sobre la marcha* del desarrollo de cada una de las fases ya que ha sido necesario aplicar los conocimientos adquiridos sobre el máster, pero además tratar de materializarlos en una idea real.

En primera instancia, al comenzar el proyecto se desconocía prácticamente al completo la arquitectura y diseño de una **aplicación *Android***, por lo que a medida que se ha ido desarrollando dentro de esta plataforma se han ido adquiriendo conocimientos dentro del desarrollo en el mundo de las aplicaciones que han permitido implementar de la manera más correcta posible la aplicación resultante de este proyecto.

En segundo lugar, ha sido el primer proyecto emprendedor real del estudiante que ha realizado este documento. Esto claramente ha requerido el aprendizaje durante todo el proceso aspectos reales del mundo empresarial como el **marketing, el análisis financiero y la evaluación real de una idea.** La necesidad de competir en un mercado real ha requerido realizar tareas más allá de un proyecto de fin de máster.
Esto ha resultado en un gran aprendizaje del funcionamiento de diferentes sistemas software tanto a nivel técnico como a nivel de negocio.

En último lugar está un aspecto posiblemente más subjetivo pero que se considera necesario de enunciar en este apartado y es la autodisciplina que es necesaria mantener cuando uno desea emprender y materializar sus propias ideas. Esta experiencia es crucial pues es en ese momento cuando te das cuenta de que un trabajo bien hecho será un resultado personal, y un trabajo mal hecho también lo será. Es por ello por lo que posiblemente la experiencia más valiosa de haber desarrollado este proyecto es el haber adquirido la capacidad y mentalidad de lograr crear una idea propia y, además, integrar esto con los diferentes conocimientos técnicos adquiridos durante toda la formación universitaria.

25. CONOCIMIENTOS EMPLEADOS

A lo largo de este apartado se muestran los diferentes conocimientos empleados en este proyecto que se han cursados en el máster, así como su justificación.

25.1. EMPRENDIMIENTO EN PRODUCTOS SOFTWARE

Los conocimientos empleados de esta asignatura son cruciales puesto que han sido los que han permitido calzar todo el desarrollo de este proyecto con una idea real de negocio. Los diferentes conocimientos adquiridos durante esta asignatura han permitido **validar la idea**, **crear una idea viable y** realizar **un análisis de los diferentes aspectos de negocio de este proyecto.**

Además, gracias a esta asignatura se ha podido desarrollar este proyecto mediante la **metodología *Lean Startup*** pues se aprendieron los principales conceptos para desarrollar ideas empleando esta metodología.

25.2. INGENIERÍA DEL LENGUAJE NATURAL

El empleo de los conocimientos de esta asignatura ha sido crucial de cara a dar pie al **algoritmo de recomendación,** ya que este funciona de tal manera que pasa completamente desapercibido para el usuario.

Para lograr esto tal y como se ha mencionado en el apartado de diseño, se emplea la ***lematización* de palabras**, lo cual implica el procesado del texto insertado en la aplicación, esto no hubiese sido posible sin el aprendizaje adquirido durante esta asignatura.

25.3. TECNOLOGÍAS DE GESTIÓN DE DATOS

A lo largo de esta asignatura se ha aprendido cómo funcionan los diferentes sistemas que permiten gestionar los datos dentro de una base de datos u otras estructuras. Los conocimientos adquiridos de esta asignatura han ayudado a la gestión de las diferentes ***bases de datos*** empleadas durante esta asignatura, así como la manipulación de estos datos sin corromperlos, así como la realización de ***migraciones*.**

25.4. INGENIERÍA DEL SOFTWARE EXPERIMENTAL

Los conocimientos adquiridos durante esta asignatura de cara a la formalización de documentos, así como la experimentación de software con usuarios finales han contribuido a desarrollar este documento, en particular el apartado de los diferentes experimentos relacionados con los *MVP*.

25.5. MODELOS FORMALES DE COMPUTACIÓN

El aprendizaje de los diferentes **paradigmas de programación** ha contribuido sustancialmente en este proyecto. Especialmente a la hora de aprender el **lenguaje *Kotlin*** y mejorar el **desarrollo en *Python*.**

Pese a que esta asignatura ha sido en gran parte teórica, la comprensión del funcionamiento de los paradigmas **funcionales y orientados a objetos** han sido factores claves de cara a simplificar la curva de aprendizaje de los lenguajes mencionados anteriormente.

25.6. DATA SCIENCE

Los diferentes conceptos adquiridos durante esta asignatura han sido un eje clave dentro de este proyecto. Esto no se debe únicamente a que la primera versión de la idea se gestase durante esta asignatura, sino que las diferentes estructuras de cara a la recomendación, gestión de datos y el entendimiento del **valor real de los datos** dentro de un sistema, así como aspectos legales referentes a la ***anonimización* de datos** han sido el núcleo de este proyecto.

25.7. INTERNET OF THINGS

Los conocimientos adquiridos durante esta asignatura referentes a los diferentes protocolos de intercambio de datos, así como las diferentes arquitecturas han motivado sustancialmente a desarrollar este proyecto en ***GraphQL***, así como a implementar una arquitectura con un **único servicio**, entendiendo ***el potencial que la computación en la nube*** tiene hoy en día.

25.8. HUMAN COMPUTER INTERACTION

Los conocimientos de diseño de interfaces adquiridos a lo largo de esta asignatura han facilitado sustancialmente el diseñar los diferentes ***mockups*** de tal manera que éstos correspondiesen en la mayor medida posible al resultado final, y, además respetando los diferentes estándares de diseño aplicados en el mundo del software.

Además, ser conocedor de la **implementación de *interfaces* basadas en casos de uso** ha hecho que el desarrollo de este proyecto sea aún más rápido de lo esperado.

25.9. EXTRACCIÓN DE INFORMACIÓN DE LAS REDES SOCIALES

Pese a que a lo largo de este proyecto no se extrae información de ninguna red social, la comprensión del funcionamiento de las diferentes *API* de las grandes redes sociales establecidas hoy en día ha contribuido sustancialmente a desarrollar una *API* similar interna en este proyecto.

Por otro lado, durante esta asignatura se aprendieron los principales aspectos de los diferentes algoritmos de recomendación permitiendo la implementación de uno dentro de este trabajo de fin de máster.

25.10. EXPERIENCIAS EN GESTIÓN DE MODELOS

Ampliar el conocimiento sobre lo que es un modelo software y profundizar en éstos, ha permitido no solamente conocer a mayor velocidad el funcionamiento interno de los **dispositivos *Android***, sino realizar un diseño interno del sistema que permitiese todas las funcionalidades esperadas sin necesidad de rediseñar a lo largo de ninguna de las fases, permitiendo como resultado una aplicación **robusta a nivel de diseño.**

26. TRABAJO FUTURO

Debido a que este proyecto ha sido enmarcado dentro del tiempo de un trabajo de fin de máster, hay múltiples aspectos que no se han podido tratar en este período.

Por esta misma razón existe una gran cantidad de trabajo a realizar en un futuro de cara a este proyecto, tanto a nivel técnico como a nivel de negocio.

Respecto a los diferentes apartados técnicos, se desean implementar todas las funcionalidades del mapa de características desarrolladas a lo largo de este proyecto, así como mejorar aún más el diseño y realizar una mayor cantidad de pruebas de tal manera que el sistema funcione a la perfección.

Además, se pretende desplegar la aplicación en un **sistema real** mediante una ***arquitectura basada en microservicios***, para poder escalar a la mayor velocidad las funcionalidades del sistema, así como **integrarla con otras plataformas** para poder crear una aplicación totalmente funcional e integrada en el *mundo de las aplicaciones móviles.*

Por otro lado, a nivel de emprendimiento se desea **elaborar un tercer *MVP*** que resulte ya en una versión pulida de aquello que se ha desarrollado durante **el segundo *MVP*** implementando las funcionalidades importantes que no hayan podido implementarse durante este trabajo de fin de máster.

Una vez se tenga este *MVP*, se desea desplegar en las diferentes tiendas de aplicaciones y comenzar a promocionar el producto con el fin de captar potenciales clientes, y así comenzar un proyecto empresarial. Durante toda esta fase es esencial tratar de captar inversores que permitan realizar **campañas de marketing** y **la adquisición de personal**, siendo esta inversión inicial la mostrada durante el análisis financiero del producto.

GLOSARIO

Front-end: Parte visual de una aplicación de software. Es aquella que se encarga de interactuar directamente con el usuario y de mostrar los datos.
Back-end: Parte lógica de una aplicación de software, aquella que corresponde a un servidor. Esta parte se encarga de realizar la gestión de los datos de la aplicación así como las funcionalidades complejas.
Plan: En este documento se referirá como planes o quedadas a una actividad realizada en un período concreto de tiempo ya sea en solitario o con un conjunto de personas.
Startup: El concepto Startup se utiliza en el mundo empresarial para referirse a empresas de reciente creación, normalmente fundadas por un emprendedor o varios, sobre una base tecnológica, innovadoras y supuestamente con una elevada capacidad de crecimiento
Early Adopters: Early adopters es un conjunto de usuarios (o empresas cliente) que están dispuestos a probar (o incluso comprar) una versión preliminar del producto/servicio, que aún no tiene todas las funcionalidades deseables, ni está totalmente robusto.
Variación Anual: La variación anual es el cambio de porcentaje entre dos valores. Para el caso de este análisis financiero hace referencia al porcentaje de potenciales clientes que se han adquirido/perdido. Por ejemplo, un 25% de variación anual significará que se adquieren un 25% de clientes más respecto al año anterior.
Backlog: Término empleado en la Ingeniería del Software para referirse a la reserva de tareas a desarrollar a lo largo de un proyecto. Estas tareas serían las que quedan pendientes por hacer.
Épica: Término empleado en la Ingeniería del Software para referirse a la reserva de tareas a desarrollar a lo largo de un proyecto. Estas tareas serían las que quedan pendientes por hacer.
MockUp: Es una maqueta del diseño de una aplicación a tamaño completo.
Corrutina: Una corrutina es un proceso ligero que ejecuta funcionalidades de forma paralela, evitando el colapso del sistema cuando se trabajan en procesos delicados o que requieran de un tiempo de espera.
Framework: Es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática en particular. Para el caso del desarrollo de software, es una estructura tecnológica que facilita el desarrollo de software en diversos aspectos.
Lema: El lema se corresponde con la palabra que encabeza una entrada léxica en un diccionario. En el caso de los nombres y adjetivos, corresponde a su forma masculina singular; en el caso de los verbos, corresponde a la forma en infinitivo.
Stopword: Es el nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc. que son filtradas antes o después del procesamiento de datos en lenguaje natural.
Freelance: Persona que trabaja por cuenta propia.
Feedback: Retroalimentación de un producto, argumento o idea.
MVP: Producto mínimo viable. En la metodología *Lean Startup* se compone de las características mínimas que ha de tener un producto funcional.
Token: Clave cifrada y temporal que permite a un cliente comunicarse con un servidor de forma segura.
Sesión: Intercambio de información interactiva semipermanente entre un sistema y un usuario.
Commit: Confirmación de un conjunto de datos provisionales para almacenarlos de forma permanente.
Anonimización: Término empleado en la ciencia de datos en el que los diferentes datos se procesan de cara a que éstos no puedan ser relacionados con ningún usuario final.

Interfaz: Parte de un sistema que muestra información y permite el control al usuario final.

BIBLIOGRAFÍA

[1] UPV, «Máster Universitario en Tecnologías y Sistemas de Software,» [En línea]. Available: https://www.upv.es/titulaciones/MUISMFSI/. [Último acceso: 15 Junio 2021].

[2] UPV, «Asignatura Data Science,» [En línea]. Available: https://www.upv.es/titulaciones/MUISMFSI/menu_1013904c.html.

[3] J. Lanier, Ten Arguments For Deleting Your Social Media Account, PICADOR, 2018.

[4] J. Lanier, ¿Quién Controla el Futuro?, DEBATE, 2013.

[5] J. O'Neill, SMART Goals, Smart Schools, 2000.

[6] E. Ries, El Método Lean Startup, Deusto, 2012.

[7] D. F. Eduardo Pertierra, PlanItNow!, Documento Final de Proyecto, Departamento de Sistemas Informáticos y Computación, 2020.

[8] I. UPV, «Concurso Emprendedor Universitario 2K21,» 2021. [En línea]. Available: https://www.ideas.upv.es/ya-tenemos-finalistas-del-concurso-2k21/. [Último acceso: 20 Julio 2021].

[9] Statists, «Statista,» [En línea]. Available: https://www.statista.com. [Último acceso: 25 Agosto 2021].

[10] M. A. Brooke Auxier, Social Media Use In 2021, www.pewresearch.org, 2021.

[11] I. N. d. Estadística, «Web del Instituto Nacional de Estadística,» [En línea]. Available: https://www.ine.es.

[12] P. E. Español, «elespanol,» [En línea]. Available: https://www.elespanol.com/elandroidelibre/aplicaciones/20120806/mitmi-app-crear-planes-compartir-eventos-publicos/18248401_0.html. [Último acceso: 13 Agosto 2021].

[13] Xataka, «Xataka,» [En línea]. Available: https://www.xatakandroid.com/aplicaciones-android/qhaceshoy-entradas-de-ultimo-minuto-con-descuento. [Último acceso: 4 Agosto 2021].

[14] Feverup, «Feverup,» [En línea]. Available: https://feverup.com. [Último acceso: 21 Julio 2021].

[15] BlaBlaCar, «BlaBlaCar,» [En línea]. Available: https://www.blablacar.es. [Último acceso: 16 Julio 2021].

[16] Facebook, «Facebook,» [En línea]. Available: https://www.facebook.com. [Último acceso: 15 Julio 2021].

[17] Google, «Google Calendar,» [En línea]. Available: https://calendar.google.com/. [Último acceso: 5 Agosto 2021].

[18] MeetUp, «MeetUp,» [En línea]. Available: https://www.meetup.com. [Último acceso: 9 Agosto 2021].

[19] Skout, «Skout,» [En línea]. Available: https://www.skout.com. [Último acceso: 15 Agosto 2021].

[20] Happn, «Happn,» [En línea]. Available: https://www.happn.com. [Último acceso: 17 Agosto 2021].

[21] Timpik, «Timpik,» [En línea]. Available: http://www.timpik.com. [Último acceso: 15 Agosto 2021].

[22] A. FODA-DAFO, «FODA-DAFO,» [En línea]. Available: https://foda-dafo.com. [Último acceso: 16 Agosto 2021].

[23] Emprendedores, «Revista Emprendedores,» 2021. [En línea]. Available: https://www.emprendedores.es.

[24] S. Bradner, RFC Key Words, Harvard University, 1997.

[25] I. I. o. B. Analyst, "MoSCoW Analysys". A Guide to the business Analysis Body of Knowledge, 2009.

[26] P. R. a. C. P. T. Sedano, The Product Backlog, 41st International Conference on Software Engineering (ICSE): IEEE, 2019.

[27] Google, «Android Developers,» [En línea]. Available: https://developer.android.com.

[28] Oracle, «www.java.com,» [En línea]. Available: https://www.java.com.

[29] Kotlin, «Kotlin,» [En línea]. Available: https://kotlinlang.org.

[30] Google, «Material Design Color Tool,» [En línea]. Available: https://material.io/resources/color/. [Último acceso: 5 Agosto 2021].

[31] W. Clark, The Gantt Chart: A Working Tool of Management, New York: Ronald Press, 1922.

[32] Catia, «Dassault Systems,» [En línea]. Available: https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/. [Último acceso: 3 Septiembre 2021].

[33] UML, «Unified Modelling Language,» [En línea]. Available: https://www.uml.org. [Último acceso: 1 Septiembre 2021].

[34] Google, «Android Studio,» [En línea]. Available: https://developer.android.com/studio. [Último acceso: 25 Junio 2021].

[35] Gradle, «Gradle Build Tool,» [En línea]. Available: https://gradle.org. [Último acceso: 5 Septiembre 2021].

[36] Git, «Git --fast-version-control,» [En línea]. Available: https://git-scm.com. [Último acceso: 5 Septiembre 2021].

[37] JetBrains, «PyCharm The Python IDE for Professional Developers,» [En línea]. Available: https://www.jetbrains.com/pycharm/. [Último acceso: 5 Septiembre 2021].

[38] Insomnia, «Insomnia,» [En línea]. Available: https://insomnia.rest. [Último acceso: 5 Septiembre 2021].

[39] Open Source, «Open Source Initiative,» [En línea]. Available: https://opensource.org. [Último acceso: 5 Septiembre 2021].

[40] Red Hat, «Qué son las API y para qué sirven,» [En línea]. Available: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces. [Último acceso: 5 Septiembre 2021].

[41] Swagger, «OpenApi Specification,» [En línea]. Available: https://swagger.io/specification/. [Último acceso: 5 Septiembre 2021].

[42] Oracle, «Java,» [En línea]. Available: https://www.java.com/. [Último acceso: 17 Julio 2021].

[43] Android Developers, «Cómo Guardar Contenido en una base de datos local con Room,» [En línea]. Available: https://developer.android.com/training/data-storage/room. [Último acceso: 23 Junio 2021].

[44] COIL, «COIL,» [En línea]. Available: https://github.com/coil-kt/coil. [Último acceso: 5 Agosto 2021].

[45] Apollo, «Apollo Client Android,» [En línea]. Available: https://www.apollographql.com/docs/android/. [Último acceso: 28 Julio 2021].

[46] GraphQL, «GraphQL,» [En línea]. Available: https://graphql.org. [Último acceso: 27 Mayo 2021].

[47] Python, «Python Website,» [En línea]. Available: https://www.python.org. [Último acceso: 5 Mayo 2021].

[48] Django, «Django Project,» [En línea]. Available: https://www.djangoproject.com. [Último acceso: 24 Mayo 2021].

[49] Graphene, «Graphene-Python,» [En línea]. Available: https://graphene-python.org. [Último acceso: 5 Mayo 2021].

[50] Graphene, «Graphene-Django,» [En línea]. Available: https://docs.graphene-python.org/projects/django/en/latest/. [Último acceso: 5 Mayo 2021].

[51] nidhaloff, «GitHub,» [En línea]. Available: https://github.com/nidhaloff/deep-translator. [Último acceso: 30 Mayo 2021].

[52] NLTK, «nltk,» [En línea]. Available: http://www.nltk.org. [Último acceso: 5 Junio 2021].

[53] I. S. K. B. Dr. Ivar Jacobson, Use-Case 2.0 ebook, Ivar Jacobson International, 2020.

[54] C. Richardson, «Microservices,» [En línea]. Available: https://microservices.io. [Último acceso: 10 Septiembre 2021].

[55] D. Extensions, «django-extensions,» [En línea]. Available: https://django-extensions.readthedocs.io. [Último acceso: 10 Septiembre 2021].

[56] Django, «Django Models,» [En línea]. Available: https://docs.djangoproject.com/en/3.2/topics/db/models/. [Último acceso: 10 Septiembre 2021].

[57] Google, «Descripción general de ViewModel,» [En línea]. Available: https://developer.android.com/topic/libraries/architecture/viewmodel. [Último acceso: 11 Septiembre 2021].

[58] R. H. R. J. J. V. Erich Gamma, Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley, 1994.

[59] B. Rocca, «Introduction to recommender systems,» [En línea]. Available: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada. [Último acceso: 13 Septiembre 2021].

[60] F. R. a. L. R. a. B. Shapira, Introduction to Recommender Systems Handbook, Springer, 2011.

[61] Github, «Github,» [En línea]. Available: https://github.com. [Último acceso: 15 Mayo 2021].

[62] Salesforce, «Heroku,» [En línea]. Available: www.heroku.com. [Último acceso: 15 Julio 2021].

[63] C. Richardson, «Microservices.io,» [En línea]. Available: https://microservices.io/patterns/microservices.html. [Último acceso: 10 Septiembre 2021].