# Universidad San Jorge

# Escuela de Arquitectura y Tecnología

# Grado en Ingeniería Informática

## Proyecto Final

# Aplicación Android de fidelización para centro comercial Puerto Venecia

**Autor del proyecto: Marcos Angel Calvo García**

**Directora del proyecto: Violeta Monasterio Bazán**

**Villanueva de Gállego, 11 de enero de 2021**

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

No doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma: Fecha: 11 de enero de 2021

## Dedicatoria y Agradecimiento

Este trabajo es un esfuerzo en el cual, de manera directa o indirecta, han participado diferentes personas opinando, corrigiendo, aportando, teniendo paciencia, dándome ánimos… Esto me ha permitido aprovechar la competencia y experiencia de muchas personas, a las cuales quiero agradecer en este apartado.

En primer lugar, me gustaría agradecer a todos y cada uno de mis profesores la formación que me han procurado durante estos años, en especial a mi tutora y directora de este proyecto, Violeta Monasterio Bazán, por la inmensa ayuda que me han otorgado para la elaboración de este proyecto de fin de grado.

Una vez dadas las gracias a mi Universidad y tutor me gustaría agradecer a la empresa Hiberus Tecnologías Diferenciales, en especial a todo el equipo de multicanalidad, por el apoyo y confianza depositada en mí durante todo este periodo.

Por último, pero no por ello menos importante, gracias a mi familia y amigos que han supuesto en todo momento un apoyo moral muy importante, ayudándome a superar los posibles obstáculos que un proyecto como este lleva consigo.

# Índice de contenido

# Índice de ilustraciones

# Índice de tablas

## Resumen

Este proyecto tiene como objetivo la creación del Front-End de una aplicación móvil nativa para dispositivos Android. Su desarrollo se realiza en la empresa Hiberus Tecnologías Diferenciales con el fin de satisfacer las necesidades de su cliente, Puerto Venecia, el cual pretende fidelizar y atraer nuevos clientes.

A lo largo de este documento se pretende presentar la aplicación realizada, recopilando el estudio, planificación y desarrollo del proyecto *Aplicación Android de fidelización para centro comercial Puerto Venecia*. Esto incluye una explicación de las principales características y tecnologías utilizadas, así como un análisis por las diferentes fases del proyecto, desde la maquetación hasta las conclusiones finales.

El resultado es una aplicación Front-End para dispositivos Android en la que sus usuarios son premiados por visitar y hacer compras en el centro comercial Puerto Venecia. Todo ello con una interfaz sencilla y común para el usuario, con ciertos toques representativos de la marca.

## Abstract

This project aims to create the Front-End of a native mobile application for Android devices. Its development is carried out at the company Hiberus Tecnologías Diferenciales in order to meet the needs of its client, Puerto Venecia, which aims to build loyalty and attract new customers.

Throughout this document it is intended to present the application made, compiling the study, planning and development of the project *Loyalty Android Application for Puerto Venecia shopping center*. This includes an explanation of the main features and technologies used, as well as an analysis of the different phases of the project, from the layout to the final conclusions.

The result is a Front-End application for Android devices in which users are rewarded for visiting and shopping at the Puerto Venecia shopping center. All this with a simple and common interface for the user, with some representative touches of the brand.

## Palabras clave

Estas son algunas de las palabras que mejor describen este proyecto:

- Aplicación
- Móvil
- Smartphone
- Front-End
- Android
- Cliente
- Usuario
- Fidelización
- Centro comercial
- Compras
- Premios
- Campañas
- Cupones

## Keywords

These are some of the words that best describe this project:

- Application
- Mobile
- Smartphone
- Front-End
- Android
- Client
- User
- Loyalty
- Shopping centre
- Purchases
- Prizes
- Campaigns
- Coupons

# 1. Introducción

En los últimos años, internet ha sido un importante centro de atención para medianas y grandes empresas, y no es de extrañar, ya que es el lugar que más usuarios alberga en el mundo. Tanto es así que muchas empresas han crecido exponencialmente gracias a su inclusión en internet, ya que pueden llegar a muchísimos más usuarios en muy poco tiempo.

Pero internet también ha ido evolucionando a lo largo de los años, así como la forma en la que los usuarios acceden a él. El momento de mayor plenitud fue la introducción al mercado de los teléfonos inteligentes o smartphones, dispositivos económicos y mucho más accesibles para los usuarios, los cuales contaban, entre otras cosas, con conexión a internet.

Y es que, en la actualidad, resulta casi inevitable usar un teléfono móvil o smartphone a diario. El éxito de estos dispositivos es tal, que el número de usuarios no para de crecer. Según Statista [1], el 70% de la población española utiliza activamente un dispositivo móvil, pero dadas las previsiones, se espera que el porcentaje siga aumentando en los próximos años.

El centro comercial Puerto Venecia, conocedor de dicha evolución, manifestó a la empresa Hiberus Tecnologías Diferenciales su deseo de llegar también a sus clientes a través de estos dispositivos. Desde esta empresa se les propuso el desarrollo de una aplicación nativa para sistemas operativos Android e iOS individualmente. Tras varios meses de análisis y definición del proyecto, este es aprobado por las partes implicadas, quienes dan luz verde para empezar con su desarrollo.

En este proyecto de fin de grado se recoge el trabajo realizado por su autor, quien se encargó principalmente del diseño y desarrollo de la aplicación Front-End para dispositivos Android.

Este documento recoge todas las fases mencionadas en el apartado anterior. Además, se incluyen varios anexos con información relevante del proyecto. Todas las tecnologías, herramientas y metodologías utilizadas en el desarrollo de la aplicación quedan explicadas para una mejor comprensión del proyecto.

En la siguiente sección se realiza un estudio del estado del arte, el cual incluye una breve introducción acerca del cliente, una comparación con aplicaciones de la misma índole y una justificación de las tecnologías utilizadas.

En la sección 3 se detallan los objetivos iniciales del proyecto, así como las posibles modificaciones que han tenido a lo largo de su desarrollo.

En la sección 4 "Metodología" se explican las metodologías utilizadas en el proyecto, así como las posibles adaptaciones que se han realizado en base a las necesidades del cliente. Además, se incluye la planificación del proyecto.

La sección 5 "Análisis" recoge un análisis completo de los requisitos y la arquitectura pensada para la aplicación, así como el análisis de la parte Front-End de la aplicación.

En la sección 6 "Desarrollo" se definen primero algunos conceptos sobre el desarrollo en Android y el lenguaje a utilizar, para después describir las características más importantes del desarrollo del proyecto y las decisiones que se llevaron a cabo.

En la sección 7 "Estudio económico" se calcula el coste de los recursos utilizados durante el desarrollo del proyecto.

Por último, se realiza un análisis de resultados comparando los objetivos iniciales con los objetivos finales, todo ello acompañado de unas conclusiones finales sobre el proyecto.

# 2. Estado del arte

En esta sección se va a contextualizar al cliente Puerto Venecia, para posteriormente explicar el marco tecnológico seleccionado para este proyecto, desde las herramientas utilizadas para el desarrollo hasta la tecnología de la aplicación.

## 2.1. Contexto

Puerto Venecia es un shopping resort situado en la ciudad española de Zaragoza, en el distrito de Puerto Venecia. Actualmente es uno de los mayores complejos comerciales y de ocio de Europa. De hecho, fue nombrado mejor centro comercial del mundo tras haber ganado el Premio Mapic 2013 [2].

El perfil de cliente que visita este centro comercial es muy variado, principalmente compuesto por jóvenes y adultos de entre 16 y 60 años con un poder adquisitivo medio, y con ciertos conocimientos de tecnología móvil. Este perfil de usuario estará presente durante todo el desarrollo del proyecto, sobre todo en temas de usabilidad.

Algunas de las funcionalidades implementadas en este proyecto, como el escaneo de tickets o el canje de cupones, ya estaban disponibles en su página web desde hacía un tiempo. El éxito de estas funcionalidades y la necesidad de mejorar la experiencia de usuario fueron algunos condicionantes para el desarrollo de este proyecto.

El cliente expresa la necesidad de usar la gamificación en la aplicación, convirtiendo la experiencia del usuario en un juego. Para ello, se realiza una búsqueda exhaustiva de juegos para dispositivos móviles que puedan concordar con los requisitos de la aplicación. Se decide tomar el juego Pokémon GO como punto de partida.

## 2.2. Proyectos existentes

Esta aplicación tiene el claro objetivo de fidelizar clientes. En este aspecto, no se ha tomado ninguna aplicación como referencia. Pero sí podemos encontrar varias aplicaciones con ciertas similitudes en este aspecto.

- **Mi Carrefour**: Con esta aplicación, el supermercado Carrefour pretende premiar a sus usuarios con cupones y cheques ahorro cuando realicen compras en el establecimiento. Este método para fidelizar clientes es usado también en este proyecto, en el que el usuario recibe cupones al escanear tickets de compra.
- **Club DIA**: Esta aplicación es bastante parecida a la anterior, pero además incluye cierta información adicional acerca de los establecimientos, al igual que la aplicación de este proyecto.

- **Lidl Plus**: Esta aplicación es la que más se acerca a la lógica de este proyecto. Además de premiar al usuario por sus compras, esta aplicación ofrece retos y juegos con los que aumentar el número de recompensas.

Para la parte de gamificación, y tras una exhaustiva búsqueda, se decide usar el videojuego Pokémon GO como principal referencia. A continuación, se explican los conceptos en común entre ambas aplicaciones.

### *2.2.1. ¿Por qué Pokémon GO?*

Pokémon GO es un juego de realidad aumentada diseñado para dispositivos móviles, y que traslada la experiencia de juego al mundo real. Su principal objetivo es que el jugador recorra las calles de su ciudad para descubrir toda clase de Pokémon. A medida que va descubriendo Pokémon, el jugador va aumentando su experiencia lo que le permite conseguir grandes premios.

Es un juego sencillo de utilizar y todo el mundo puede jugar. Además, la mecánica de juego, a pesar de estar presente durante varios años, sigue siendo novedosa pese a lo simple que resulta. El hecho de que un videojuego haga a sus usuarios interactuar con el exterior es novedoso y emocionante para ellos. Y este último punto es clave en este proyecto.

Pero lo más importante de todo es la viralidad de la aplicación, y precisamente por serlo, genera aún más expectación y ganas de probarlo a quien todavía no lo conoce.

Todas estas características encajan a la perfección con los objetivos que el cliente desea cumplir con este proyecto. Es por ello que se decide tomar este videojuego como punto de referencia durante todo el desarrollo.

### *2.2.2. Perfil de usuario*

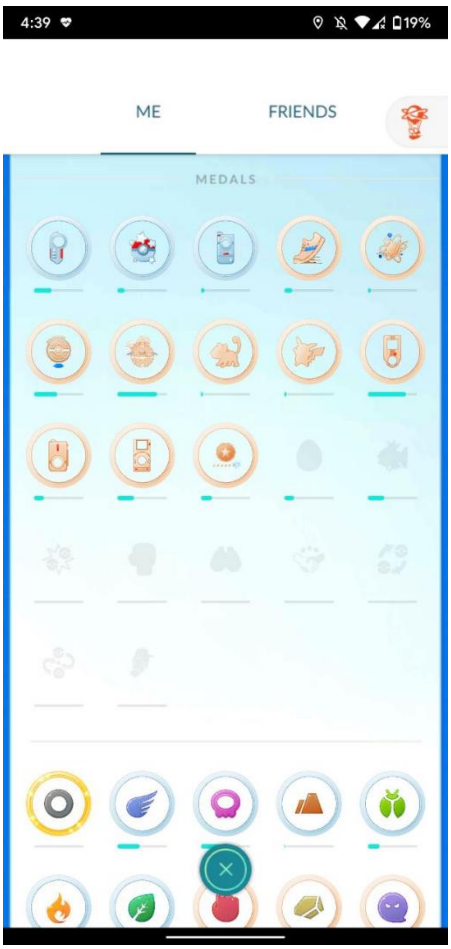



*Ilustración 1 - Medallas Pokémon GO*

En el juego Pokémon GO, el usuario tiene que capturar personajes y competir contra otros usuarios. Cada vez que el usuario captura un personaje o gana un combate, consigue un número determinado de puntos de experiencia.

Además, en el juego existen campañas que agrupan retos. Cuando el usuario completa los retos que pertenecen a una campaña, a parte del premio extra, recibe su medalla correspondiente.

### *2.2.3. Retos diarios*


*Ilustración 2 - Retos Pokémon GO*

En el juego Pokémon GO las tareas o acciones que tiene que realizar un usuario para conseguir puntos de experiencia se denominan retos. El usuario debe completar dichos retos en un determinado tiempo para conseguir una recompensa. Estos retos pueden contener una o varias acciones a realizar para completar el reto.

### *2.2.4. Super cofre*

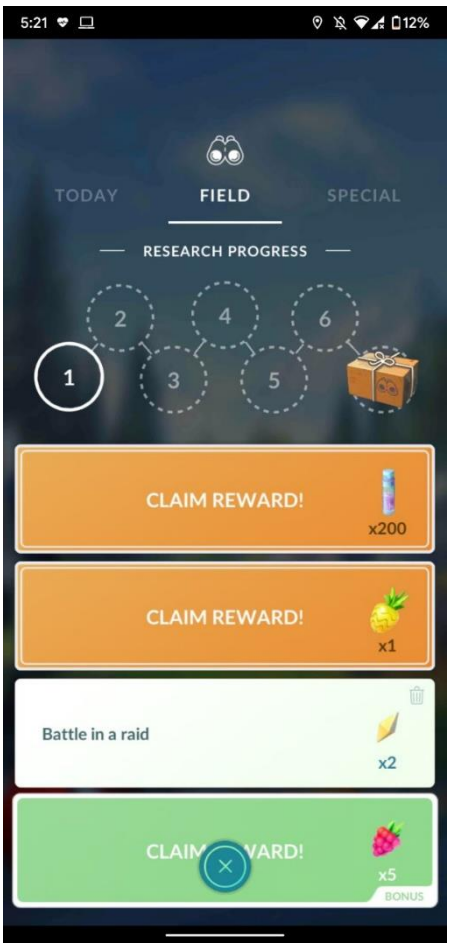



*Ilustración 3 - Super cofre Pokémon GO*

En el juego, tras completar un cierto número de retos, independientemente de la campaña a la que pertenezcan, el usuario puede abrir un "super cofre". Este super cofre contiene premios similares a los que se pueden encontrar en los retos, y se consiguen como premio por haber completado un determinado número de retos.

El usuario puede ver en todo momento cuántos retos lleva completados hasta desbloquear el super cofre. Una vez se desbloquea y el usuario canjea los premios, se le asigna un nuevo super cofre y se reinicia su progreso.

## 2.3. Contexto tecnológico

Este apartado viene a explicar el por qué se ha decidido desarrollar la aplicación de forma nativa, explicando las causas y limitaciones encontradas en el desarrollo híbrido. Además, se realiza un breve repaso a las tecnologías utilizadas durante el desarrollo del proyecto.

### *2.3.1. Aplicación Nativa vs. Aplicación Hibrida*

Una **aplicación nativa** está desarrollada en el lenguaje soportado por el sistema operativo a la que está destinada. Esto quiere decir que están desarrolladas para un equipo o plataforma determinada, en este caso Android.

La principal ventaja de este tipo de aplicaciones es que se adaptan 100% al dispositivo, aprovechando todos sus recursos y permitiendo utilizar todas sus funciones (cámara, GPS, giroscopio…). Esto las vuelve rápidas, ligeras y amigables para el usuario.

Pero si el cliente desea una aplicación multiplataforma, es decir, una aplicación soportada por diferentes sistemas operativos, es necesario desarrollar una aplicación nativa para cada sistema. Algo que puede evitarse con las aplicaciones hibridas.

Una **aplicación hibrida** está desarrollada sobre tecnología web, por lo que son soportadas por diferentes sistemas operativos. Esto reduce el coste de desarrollo, puesto que sólo es necesario desarrollar una aplicación para todas las plataformas. Pero, al no desarrollarse en un lenguaje soportado por el sistema operativo, no se pueden usar todos sus recursos y funciones. Son, por tanto, más lentas y menos amigables para el usuario, y su funcionalidad puede quedar limitada por las funciones del sistema operativo a las que se puede acceder.

Este proyecto comprende una aplicación que trabaja con el GPS del dispositivo, el modem bluetooth, la cámara y los archivos internos del teléfono. Además, la aplicación debe ser capaz de realizar tareas en segundo plano, algo muy difícil de controlar en una aplicación hibrida.

Por estas razones, y dada la necesidad de crear una aplicación para todos los dispositivos móviles, se decide desarrollar dos aplicaciones nativas: Una aplicación para dispositivos iOS, y otra para dispositivos Android, de la cual trata este proyecto.

### *2.3.2. Tecnologías utilizadas*

Durante el desarrollo de este proyecto se ha hecho uso de varias herramientas y tecnologías, las cuales se explican a continuación.

#### *2.3.2.1. Android Studio*

Android Studio [3] es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android, basado en IntelliJ IDEA. Además de un editor de código, Android Studio ofrece un sistema de compilación basado en *Gradle*, un emulador rápido y multifuncional, un entorno unificado que incluye todas las plataformas Android, integración con clientes GIT y variedad de marcos de trabajo y herramientas de prueba.

#### *2.3.2.2. Kotlin*

Kotlin [4] es un lenguaje de programación fuertemente tipado desarrollado por JetBrains (los creadores de IntelliJ IDEA). El salto respecto al lenguaje Java (su lenguaje antecesor) es enorme y la productividad mejora considerablemente. Se le conoce comúnmente como un lenguaje que soluciona los problemas de su lenguaje antecesor.

#### *2.3.2.3. rxKotlin (programación reactiva)*

La programación reactiva [5] es un paradigma enfocado en el trabajo con flujos de datos de manera asíncrona. Esto permite que los datos se propaguen generando cambios en la aplicación. Existen librerías que trasladan la programación reactiva casi a cualquier lenguaje de programación. rxKotlin parte de la librería rxJava con una API diseñada pensando en Kotlin.

#### *2.3.2.4. Gradle*

Gradle [6] es una herramienta de automatización de construcción de código de software. dispone de una gran flexibilidad que le permite trabajar con numerosos lenguajes, incluyendo Kotlin. Cuenta además con un sistema de gestión de dependencias muy estable, y es el sistema de compilación oficial para Android.

#### *2.3.2.5. Trello*

Trello [7] es una herramienta de gestión de proyectos online que permite organizar tareas priorizando el trabajo en grupo, mediante el uso de tableros virtuales divididos en listas de tareas. Se basa en el método Kanban donde se utilizan los famosos To Do (por hacer), Doing (en proceso) y Done (finalizado).

#### *2.3.2.6. Redmine*

Redmine [8] es una herramienta para la gestión de proyectos, que con sus diversas funcionalidades permite a los usuarios de diferentes proyectos realizar el seguimiento y organización de los mismos. Incluye herramientas como calendario de actividades, diagramas de Gantt para la representación visual de la línea del tiempo de los proyectos, control de flujo de trabajo basado en roles, integración con correo electrónico...

#### *2.3.2.7. Git*

Git [9] es un software de control de versiones en el que destaca la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

#### *2.3.2.8. BitBucket*

BitBucket [10] es un servicio de alojamiento Git para equipos profesionales. Además de funciones como gestión de proyectos privados, revisión de código, gestión de permisos o compilación automática, BitBucket ofrece integración con servicios externos como Trello o Jira. Existen diferentes planes, desde un plan gratuito con un número limitado de funciones, hasta un plan premium con todas las funciones habilitadas.

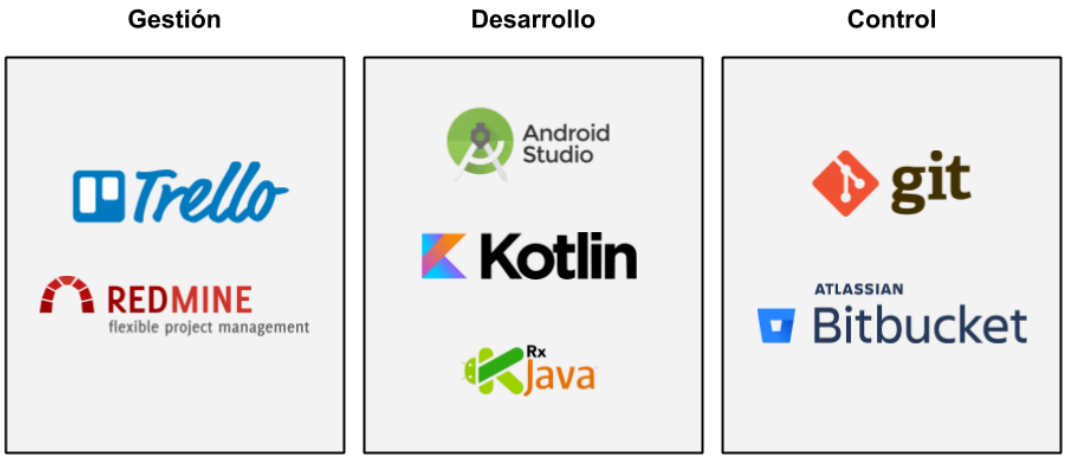



*Ilustración 4 - Estructura de marco tecnológico*

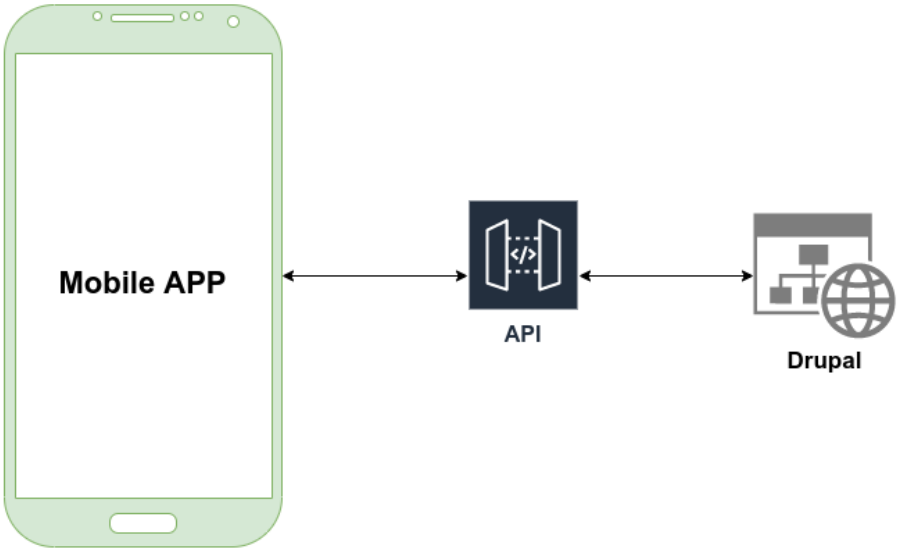



*Ilustración 5 - Conexiones de la app*

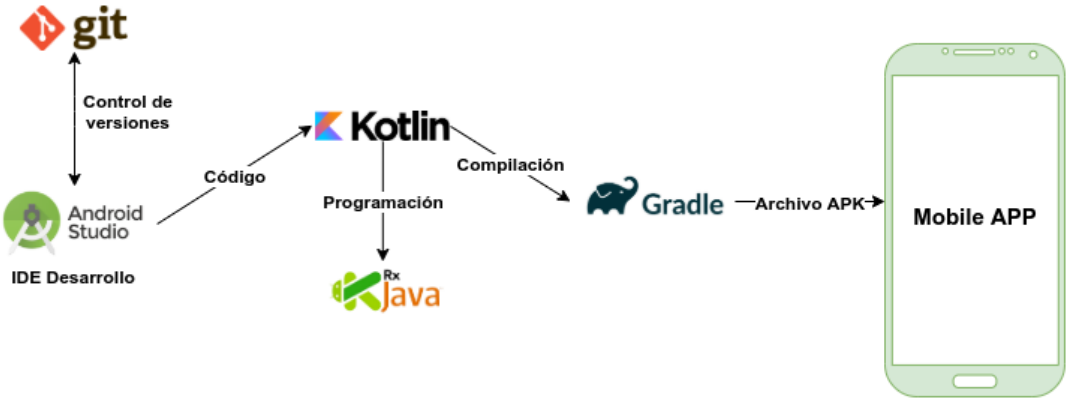



*Ilustración 6 - Estructura del código*

# 3. Objetivos

El objetivo de este proyecto de fin de grado es elaborar el Front-End de una aplicación nativa para dispositivos móviles Android para la fidelización de clientes en el centro comercial Puerto Venecia. En esta aplicación, los usuarios pueden registrarse, escanear tickets, completar retos, participar en campañas, consultar información acerca del centro comercial, e incluso invitar a amigos y familiares con su código de referido para obtener más recompensas.

La forma de llegar al usuario es un punto clave de la aplicación, por lo que el uso de una interfaz amigable y fácil de usar han estado presentes durante la elaboración de todo el proyecto. Esto, junto a la robustez de la propia aplicación, definen la calidad del producto final.

## 3.1. Objetivos del proyecto

Para conseguir el objetivo global se definieron los siguientes objetivos específicos:

1. **Análisis** de los requisitos de la aplicación, con el objetivo de aclarar exactamente lo que se va a realizar. Las principales características a tener en cuenta son:
    - Desarrollo de una aplicación nativa en lenguaje Kotlin.
    - Uso de gamificación con características similares al juego Pokémon GO.
    - Conexión con una API de gestión de contenido.
    - Empleo de programación reactiva para trabajar con los flujos de datos de la aplicación.
2. **Diseño** de la aplicación. Esto incluye el diseño de su arquitectura, pantallas e interfaz.
3. **Desarrollo** de la parte Front-End en lenguaje Kotlin. Se tienen en cuenta las diferentes resoluciones que los dispositivos puedan tener, así como el tipo de dispositivo (móvil o tablet).
4. **Integración** con la parte Back-End mediante una API diseñada específicamente para la aplicación.

Tras las primeras reuniones con el equipo fue necesario añadir un nuevo objetivo específico a realizar: un **estudio** sobre ciertas tecnologías adicionales para la aplicación. Estas tecnologías no estaban del todo claras en un principio, por lo que fue necesario un estudio para definirlas.

Este estudio, del cual se hablará más adelante, se realizó durante la fase de análisis del proyecto.

# 4. Metodología

Durante la realización de este proyecto se ha utilizado la metodología ágil SCRUM, presente en la mayoría de proyectos de la empresa Hiberus Tecnologías Diferenciales. No obstante, en este proyecto esta metodología ha sido adaptada a los tiempos y necesidades del cliente, el cual estableció el ritmo de trabajo de este proyecto.

## 4.1. SCRUM

### *4.1.1. Adaptación de la metodología*

Si bien es cierto que la metodología SCRUM ha sido la única metodología ágil utilizada en el proyecto, ésta ha sufrido algunas modificaciones adaptándose a las necesidades del cliente:

- Esta metodología está basada en ciclos o sprints, con una duración de entre 2 y 4 semanas y una única entrega final. Sin embargo, en este proyecto se decide realizar una entrega por cada tarea realizada. Esto se traduce en sprints con varias entregas cada uno, por lo que la validación es continua durante todo el desarrollo. Al final del sprint, y tras la última entrega, el cliente realiza una validación más profunda de todas sus tareas y su integración.
- Los roles es otra parte fundamental en esta metodología. En este proyecto existen tres equipos de trabajo diferentes, cada uno encargado de una parte del proyecto en concreto, como se puede ver en la ilustración 2 "Conexiones de la app". Se decide implantar un modelo de gestión integral, en la que se define al *scrum master* y al *product owner* como director del proyecto. La estructura queda distribuida de la siguiente manera:

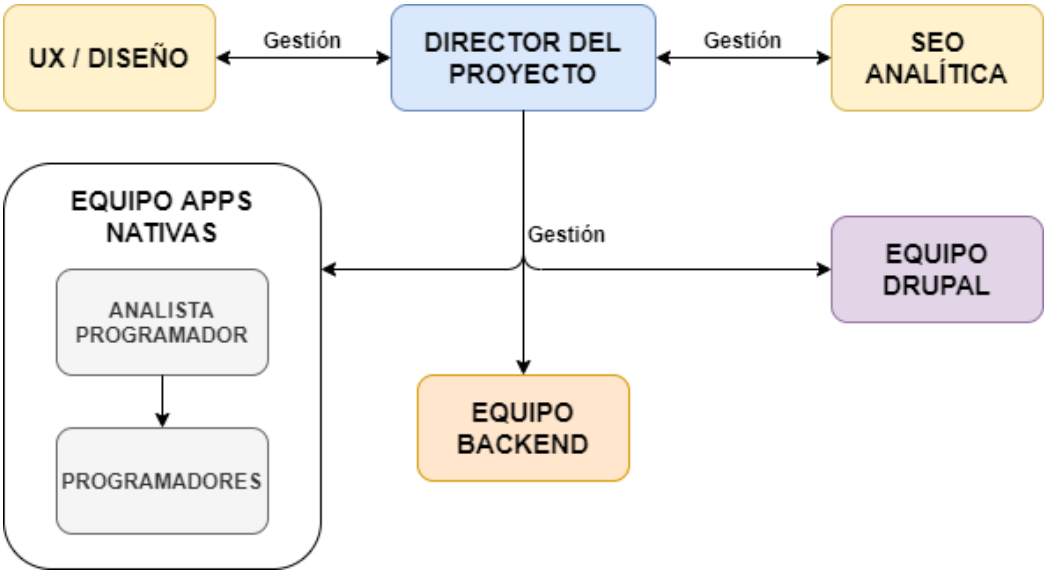



*Ilustración 7 - Estructura de equipos*

Como se puede observar en la ilustración anterior, el equipo encargado del desarrollo de la aplicación en nativo se ha dimensionado de la siguiente manera:

- Director del proyecto: Se encarga de gestionar el proyecto y establecer las relaciones con el cliente. Actúa como *scrum master* y *product owner* del proyecto. Es la única persona encargada de gestionar el proyecto.

- Analista programador: Se encarga de supervisar a los programadores y asegurar un código coherente y de calidad. Es el encargado de tomar las decisiones críticas en cuanto al desarrollo del código se refiere.

- Programadores: Bajo la supervisión del analista, trabajan en el desarrollo de la aplicación tanto para iOS como para Android. El equipo está compuesto por dos programadores, cada uno destinado al desarrollo en una plataforma.

- Las reuniones juegan un importante papel en la metodología. SCRUM define reuniones diarias o "daily meetings" para comentar con el resto del equipo el estado actual de las tareas y solucionar errores si los hubiera. Durante el desarrollo de este proyecto, se decidió realizar una reunión cada primer día de la semana, además de reuniones internas cada dos o tres días entre analista programador y programadores. Las tareas definidas eran algo extensas, y era inviable tener reuniones diarias para informar de su estado.

### *4.1.2. Ventajas y desventajas*

A pesar de tener bien definidos los requerimientos desde un principio, estos sufrieron algunos cambios a lo largo del desarrollo del proyecto. SCRUM ofrece flexibilidad y adaptabilidad en los proyectos, por lo que los cambios no son un problema.

Además, al dividir el proyecto en sprints, su desarrollo se vuelve más sencillo y rápido. Todos los miembros del equipo saben las tareas que se están realizando, por lo que se realiza un control y seguimiento constante de las mismas. Durante las reuniones el director del proyecto puede evaluar el nivel de implicación de cada uno de los miembros del equipo.

Pero el uso de esta metodología también ha traído algunos problemas. Durante el desarrollo del proyecto fue necesario sustituir a un miembro del equipo, lo cual causó retraso durante el desarrollo de algunas tareas. Asimismo, limitar el tiempo de las tareas ocasionó cierta presión en el equipo.

### *4.1.3. Herramientas utilizadas*

Como ya se ha mencionado en el apartado 3.3.2 "Tecnologias utilizadas", las herramientas encargadas de la gestión y seguimiento del proyecto son Trello y Redmine.

#### *4.1.3.1. Trello*

A través de esta plataforma online se ha gestionado el proyecto en su totalidad. Se han creado diferentes tableros, uno por plataforma:

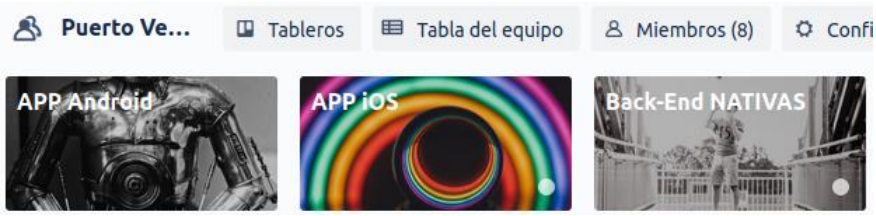



*Ilustración 8 - Tableros en Trello*

En el tablero "APP Android", destinado a este proyecto, se han categorizado las tareas de la siguiente manera:


*Ilustración 9 - Categorías en Trello APP Android*

Cada categoría hace referencia a un apartado de la aplicación, lo que facilita la segregación de tareas. Además, las tareas se dividen por estados, como se puede ver en el Anexo IV "Distribución Trello". Trello agrupa estos estados por columnas. Se decide usar 5 estados diferentes, además de una columna extra con la documentación necesaria:

- Estado *TO DO*: Tareas pendientes de realizar.
- Estado *BUGS*: Errores pendientes de resolver.
- Estado *IN PROGRESS*: Tareas que se están llevando a cabo.
- Estado *CHECK*: Tareas enviadas a validación.
- Estado *VALIDATED*: Tareas completadas.

#### *4.1.3.2. Redmine*

Esta herramienta online es gestionada por la empresa Hiberus Tecnologías Diferenciales, y es usada para imputar las horas dedicadas a un proyecto. Se ha creado un proyecto por cada tablero de Trello. Dentro de cada proyecto se encuentran el mismo número de tareas definidas en Trello.

Cada tarea tiene un estado, que va cambiando según su desarrollo. Además, se les asigna prioridad, una o varias personas encargadas de llevarla a cabo, un responsable del equipo y una categoría específica, la cual coincide con las etiquetas creadas en Trello.

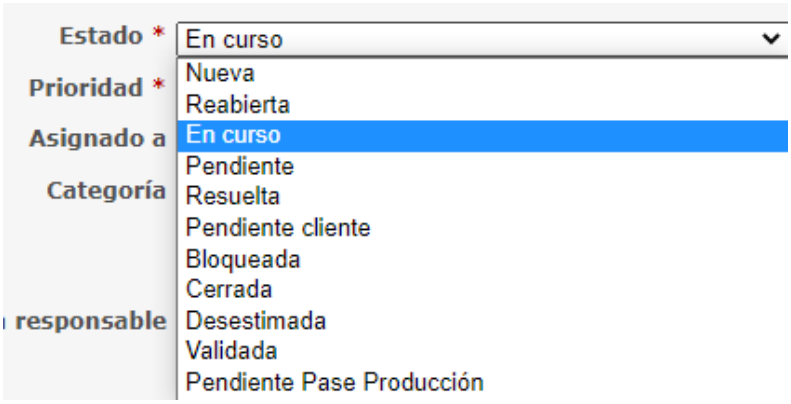



*Ilustración 10 - Estado de tareas en Redmine*

Debido a la importancia de esta herramienta para la empresa, ésta es gestionada por el director del proyecto. El resto de miembros del equipo gestionan las tareas en Trello, y usan únicamente Redmine para dejar constancia de las horas dedicadas.

## 4.2. Planificación

Conociendo la fecha de inicio del proyecto, y tras realizar algunas reuniones con el cliente para aclarar algunos puntos, se define la planificación para este proyecto de fin de grado. Esta planificación otorga una duración de 16 semanas a este proyecto, las cuales se encuentran distribuidas como se muestra a continuación.

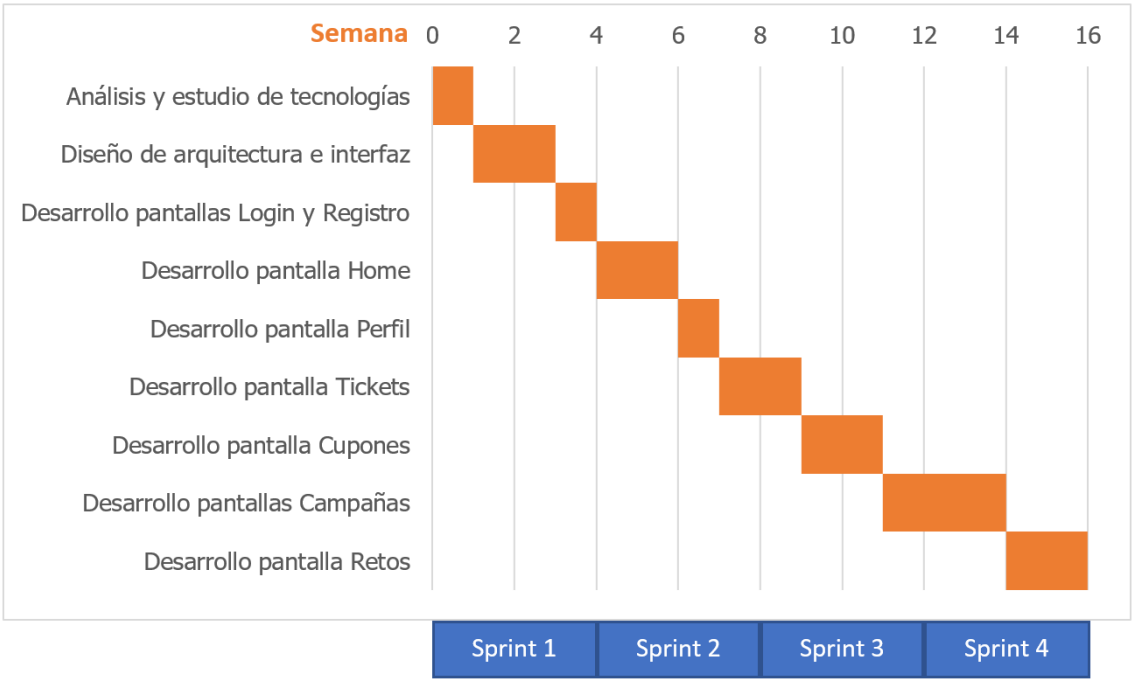



*Ilustración 11 - Diagrama de Gantt planificación de tareas*

Como se puede observar, la planificación se compone de cuatro sprints, con una duración de cuatro semanas cada uno. Al final de cada tarea, se realiza una entrega con los últimos cambios de la aplicación al cliente, para su posterior validación.

La distribución de tareas en cada sprint es la siguiente:

- **Sprint 1:** Se inicia en la semana 1, e incluye los trabajos de análisis y estudio de tecnologías a utilizar, el diseño de la arquitectura y la interfaz de la aplicación, y las pantallas de Login y Registro.
- **Sprint 2:** Se inicia en la semana 5, y comprende el desarrollo e integración con la parte Back-End de las pantallas Home, Perfil y parte de la pantalla de Tickets.
- **Sprint 3:** Se inicia en la semana 9, y comprende el desarrollo e integración con la parte Back-End de las pantallas de Tickets, Cupones y parte de las pantallas de Campañas.
- **Sprint 4:** Se inicia en la semana 13, y comprende el desarrollo e integración con la parte Back-End del resto de pantallas de Campañas y la pantalla de Retos.

# 5. Análisis

En este apartado se detalla la arquitectura tecnológica para este proyecto, así como un análisis de los requisitos desde el punto de vista Front-End de la aplicación.

## 5.1. Arquitectura tecnológica

Los diferentes elementos tecnológicos que componen el sistema se disponen de la siguiente manera:

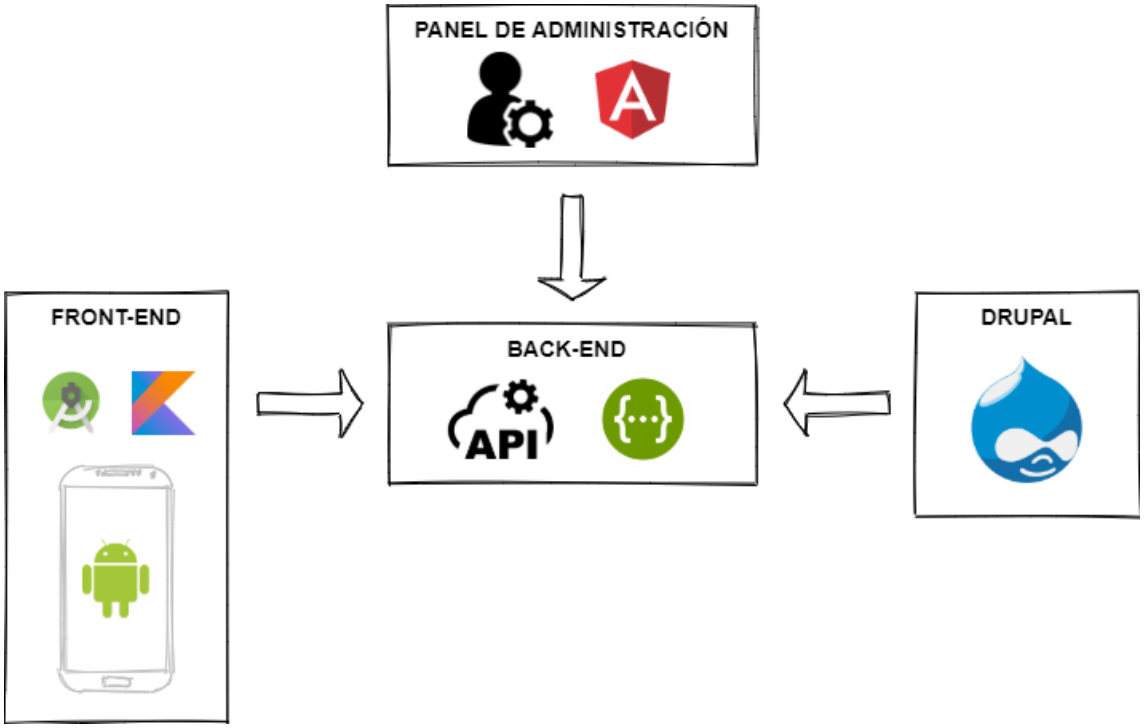



*Ilustración 12 - Esquema tecnológico*

### *5.1.1. Drupal*

Drupal [11] es el gestor de contenidos utilizado por Puerto Venecia. Esta herramienta se encarga de gestionar a todos los usuarios registrados desde la página web o desde las aplicaciones móviles del centro comercial. Además, es la encargada de generar la información necesaria para las campañas, cupones y retos de la aplicación. Actualmente se está haciendo uso de la versión 8 de esta herramienta.

### *5.1.2. Back-End*

La parte Back-End de la aplicación está desarrollada mediante las tecnologías NodeJS [12] y Swagger [13]. Su principal función es actuar de intermediario entre Drupal y la parte Front-End

de la aplicación. Mediante el uso de estas tecnologías se consigue una integración mucho más rápida y ágil entre ambas partes, liberándolas de procesos y recursos innecesarios.

La integración de esta parte con Drupal fue algo compleja debido a las limitaciones de dicha herramienta. Aunque Drupal facilita una API para acceder a algunas de sus funciones, fue necesario realizar varias adaptaciones en Back-End para soportar toda su funcionalidad.

### *5.1.3. Front-End*

Esta parte comprende las aplicaciones para Android e iOS. La aplicación para Android, de la cual trata este proyecto, está desarrollada en lenguaje Kotlin [4], el cual es oficial para esta plataforma desde 2017. Además, se hace uso de la tecnología rxKotlin [5], así como diferentes servicios de Google para gestionar notificaciones, analítica de usuario y errores.

Ambas aplicaciones se comunican exclusivamente con la parte Back-End, la cual les suministra el contenido para todas sus pantallas. En la aplicación de Android, toda esta comunicación se realiza mediante el uso de la herramienta Retrofit [14], siguiendo el patrón Model-View-ViewModel [15].

### *5.1.4. Panel de administración*

Esta parte ha sido desarrollada para gestionar el contenido personalizado de la aplicación. Desde este panel, las personas encargadas pueden gestionar los permisos de los usuarios, validar tickets o gestionar cupones y campañas. Ha sido desarrollado mediante la tecnología Angular [16].

## 5.2. Descripción funcional

En esta sección se explica la funcionalidad de la aplicación para dispositivos Android. Se muestra un flujo de todas sus pantallas para posteriormente explicarlas al detalle.

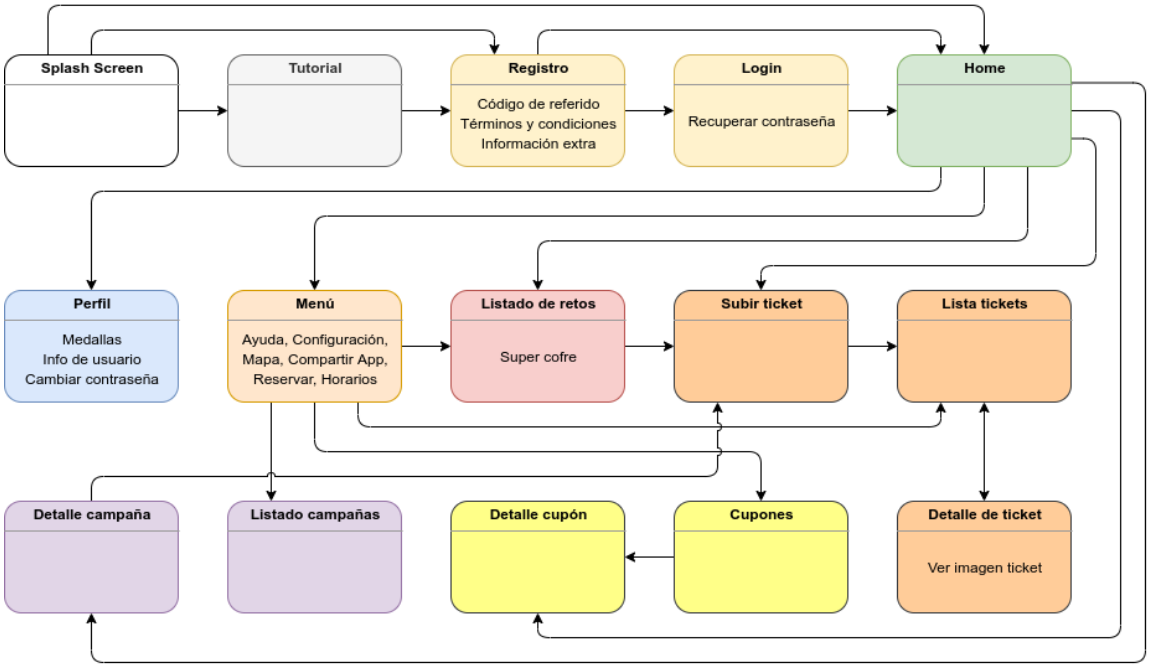



*Ilustración 13 - Flujo de pantallas de la aplicación*

### *5.2.1. Splash screen*


*Ilustración 14 - Splash screen*

Esta pantalla aparece cada vez que el usuario abre la aplicación en su dispositivo. Su principal función es mostrar una interfaz ligera mientras carga la pantalla principal de la aplicación, la cual necesita de más tiempo y recursos para mostrarse. Así pues, esta pantalla hace de punto intermedio entre el escritorio y nuestra aplicación.

### *5.2.2. Tutorial*


*Ilustración 15 - Pantalla de Tutorial*

Cuando el usuario abre la aplicación por primera vez, se le muestra un tutorial genérico haciéndole un breve resumen del contenido de la aplicación. En esta pantalla debe aceptar los permisos del dispositivo que la aplicación necesita para un correcto funcionamiento, como son la ubicación, el acceso a cámara y archivos o el acceso bluetooth.

Esta pantalla se divide en varias secciones, cada una con una instrucción diferente. Cuando el usuario completa el tutorial, este se cierra de manera permanente y da paso a la siguiente pantalla de la aplicación.

### *5.2.3. Registro*

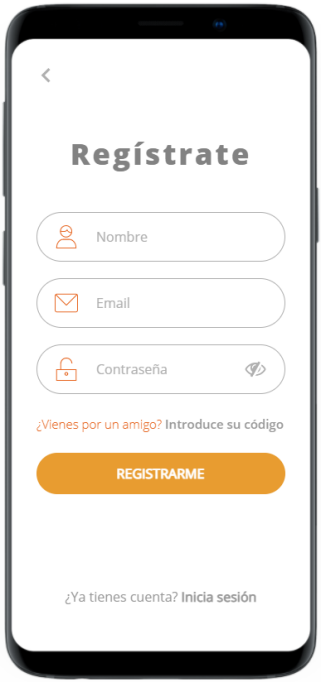



*Ilustración 16 - Pantalla de Registro*

Si un usuario no tiene una sesión iniciada todavía, una vez ha pasado por las pantallas anteriores se le muestra un formulario de registro, mediante el cual puede crearse una cuenta para la aplicación. La información requerida es bastante simple, aunque veremos que puede ser ampliada posteriormente.

El usuario puede ver u ocultar la contraseña introducida pulsando en el icono que hay en el lado derecho de ese campo.

Si el usuario ya tiene una cuenta, se le ofrece la opción de iniciar sesión con dicha cuenta (ver apartado 5.2.4). Además, está aplicación cuenta con un sistema de referidos, por el que la persona que invita recibe una recompensa. Ahora veremos con más detalle esta última pantalla.

Una vez el usuario pulsa en "Registrarme", y tras realizar una validación de los campos del formulario, la aplicación muestra las pantallas de información extra de registro (ver apartado 5.2.3.2).

#### *5.2.3.1. Código de referido*


*Ilustración 17 - Pantalla código de referido*

Si un usuario ha sido invitado por otro, puede introducir su código de referido durante su registro. Este código de referido es generado automáticamente cuando un usuario se registra en la aplicación. Cada vez que un usuario se registra con tu código de referido, recibes ciertos puntos de experiencia como premio, además de completar los retos en los que se requiera esta acción.

Tras pulsar en "Enviar", la aplicación regresa a la pantalla de registro.

#### *5.2.3.2. Información extra*

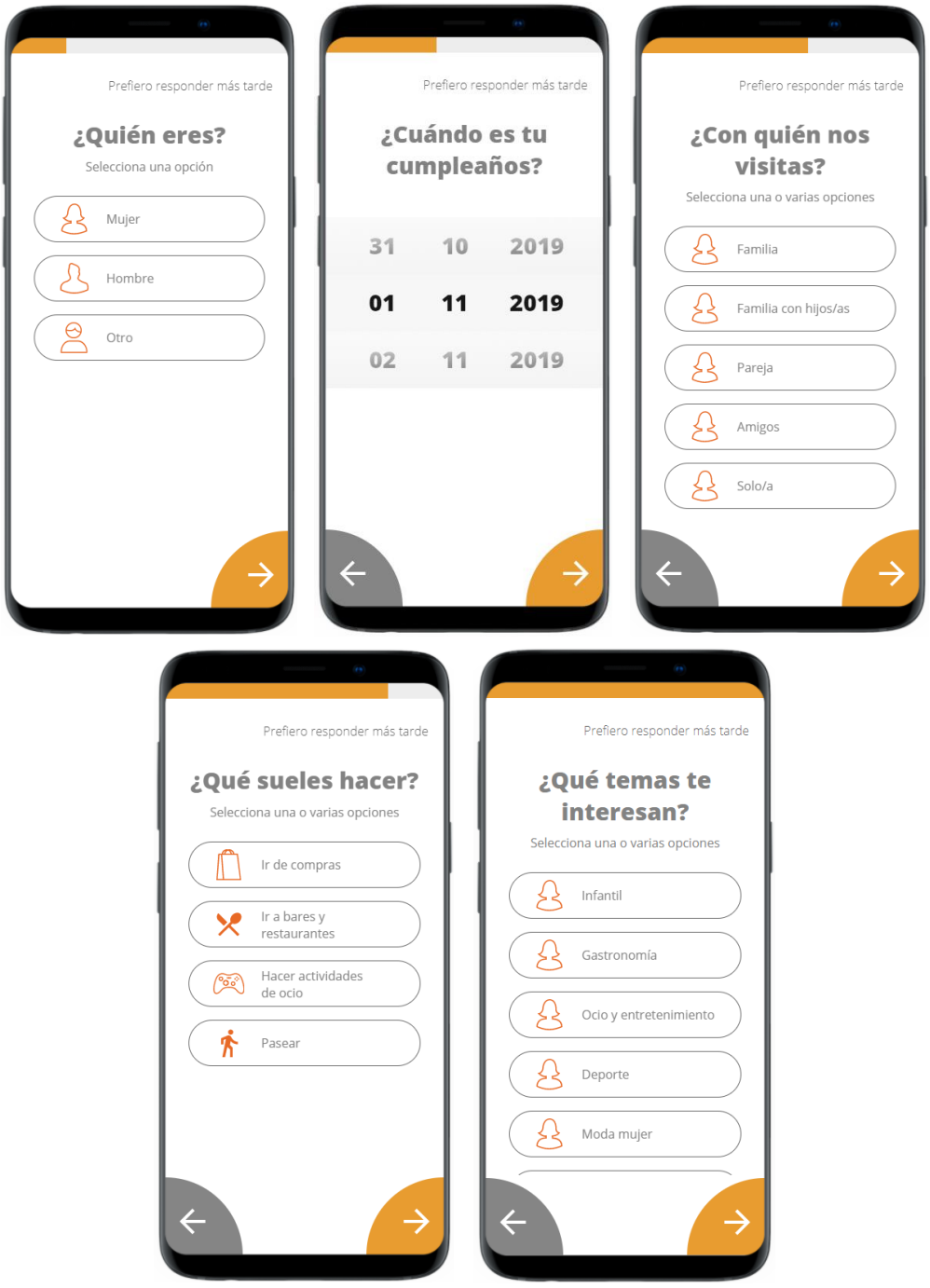



*Ilustración 18 - Pantallas información extra en registro*

Estas pantallas cumplimentan la información del usuario en su registro. Cuando el usuario completa dicha información, se le recompensa con un cierto número de puntos de experiencia. Una vez completa estos formularios o pulsa en "Prefiero responder más tarde", se inicia su sesión.

### *5.2.4. Login*


*Ilustración 19 - Pantalla de Iniciar Sesión*

Si el usuario no tiene una sesión iniciada en la aplicación, y ya cuenta con un perfil registrado, puede iniciar sesión mediante esta pantalla. Tras realizar una validación de los campos del formulario, se procede a iniciar su sesión.

El usuario puede ver u ocultar la contraseña introducida pulsando en el icono que hay en el lado derecho de ese campo.

Si el usuario ha olvidado su contraseña de inicio de sesión, puede restablecerla pulsando en "¿Has olvidado tu contraseña?". Se le enviará un email con las instrucciones necesarias para restablecer su contraseña vía web.

### *5.2.5. Home*


*Ilustración 20 - Pantalla de Inicio*

La pantalla de inicio o "Home" está diseñada para resumir la mayoría del contenido de la aplicación. El usuario con sesión iniciada puede ver información de su perfil como nombre, foto, numero de medallas, numero de perfil y puntos de experiencia.

Justo debajo de la información de perfil, el usuario puede ver el número de retos que tiene asignados actualmente mediante una caja de información. Pulsando sobre dicha caja la aplicación navega hasta la sección de retos (ver apartado 5.2.9).

Por último, se muestra el listado de campañas activas y cupones conseguidos. Este contenido se muestra además en pantallas independientes (ver apartado 5.2.10 para campañas y 5.2.11 para cupones). Al pulsar sobre un elemento, la aplicación navega hasta la vista de detalles correspondiente.

Al final de esta pantalla se encuentra una barra de navegación, con la que el usuario puede acceder al perfil, al menú de la aplicación, o a la pantalla de escaneo de ticket.

### *5.2.6. Menú*


*Ilustración 21 - Menú de la aplicación*

El menú de la aplicación se divide en dos partes, como se puede ver en el siguiente diagrama:

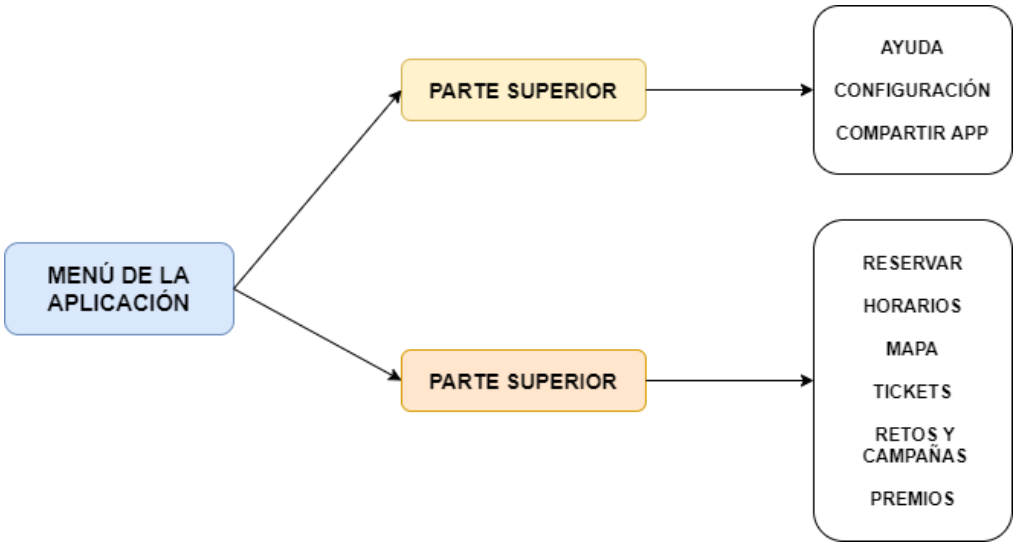



*Ilustración 22 - Diagrama componentes del menú de la aplicación*

- **Parte superior**: Contiene información de interés, la cual es tratada fuera de la aplicación.
    - **Ayuda**: Abre un enlace en el navegador del dispositivo con cierta información de ayuda y contacto.
    - **Configuración**: Abre la configuración de la aplicación, dentro de los ajustes del dispositivo. Aquí se encuentra configuración acerca de los permisos o el almacenamiento de la aplicación.
    - **Compartir App**: Permite compartir un texto con el enlace a la aplicación y el código de referido del usuario.
- **Parte inferior**: Contiene información interna de la aplicación.
    - **Reservar**: Abre una página web internamente donde el usuario puede reservar mesa en la mayoría de los restaurantes del centro comercial.
    - **Horarios**: Abre una página web internamente con información referente a las fechas y horas de apertura del centro comercial.
    - **Mapa**: Abre una página web internamente que muestra el aforo actual en el centro comercial dividido en secciones
    - **Tickets**: Muestra la pantalla de listado de tickets (ver apartado 5.2.8).
    - **Retos y campañas**: Muestra una pantalla con el listado de retos activos y campañas disponibles (ver apartados 5.2.9 y 5.2.10).
    - **Premios**: Muestra la pantalla de listado de cupones (ver apartado 5.2.11).

### 5.2.7. *Perfil*

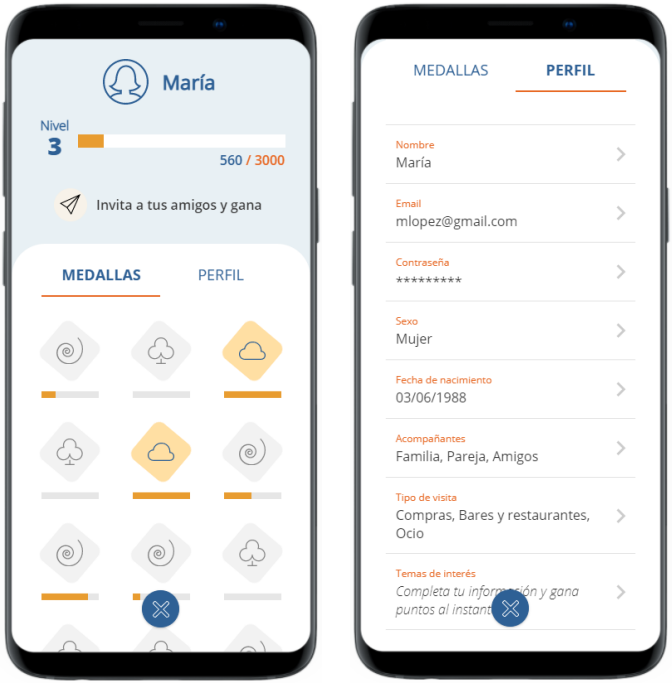



*Ilustración 23 - Pantallas de Perfil*

La pantalla de perfil se divide en dos partes, tal y como se muestra en la ilustración anterior. La parte de medallas contiene un listado con las medallas que el usuario ha conseguido o puede conseguir. Al pulsar sobre una medalla, se muestra un diálogo con cierta información sobre cómo conseguirla.

La parte de perfil como tal muestra información relevante sobre el usuario. Todos los campos son editables. Se requiere un sistema de auto guardado, por el cual cada campo es guardado automáticamente una vez editado.

Se decide implementar una lógica similar a la utilizada en el juego de referencia Pokémon GO. El usuario de la aplicación recibe un determinado número de puntos de experiencia cada vez que completa una acción. Estas acciones suelen estar agrupadas en campañas, y cada vez que el usuario completa una campaña, recibe su medalla correspondiente. Dependiendo de la cantidad de puntos de experiencia que vaya consiguiendo, se le asignará un número de nivel determinado.

### *5.2.8. Tickets*

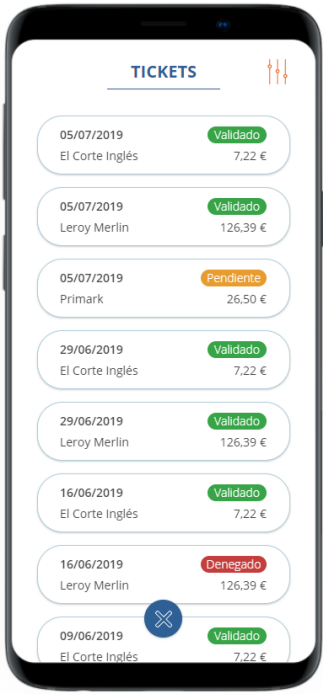



*Ilustración 24 - Pantalla de listado de Tickets*

Esta pantalla muestra un listado de tickets enviados por el usuario. Los tickets pueden tener tres estados:

- **Pendiente**: El ticket ha sido enviado con éxito y está pendiente de ser validado por el personal de Puerto Venecia.
- **Validado**: El ticket ha sido validado correctamente por el personal de Puerto Venecia y, por ende, ha puntuado correctamente en los retos y campañas necesarios.
- **Denegado**: El ticket no ha podido ser validado por el personal de Puerto Venecia debido a un error o información no visible.

Al pulsar en el icono de la parte superior derecha este listado puede ser filtrado por nombre, estado, importe o establecimiento.

Al pulsar en un elemento del listado se abre la pantalla de detalles del ticket, siempre y cuando éste haya sido validado o denegado.

#### *5.2.8.1. Detalles de ticket*

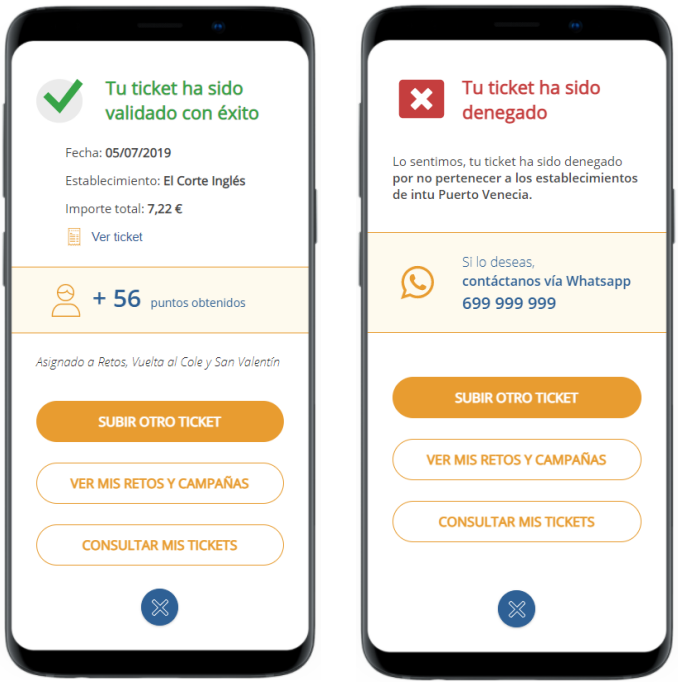



*Ilustración 25 - Pantallas de estado de Ticket*

La pantalla de detalles tiene algunas diferencias dependiendo del estado actual del ticket.

Si el ticket ha sido validado, muestra información referente al ticket, como la fecha de compra, el importe, el establecimiento o una foto del ticket (accesible al pulsar en la opción "Ver ticket"). Además, muestra dónde ha puntuado ese ticket (si así fuera).

En cambio, si el ticket ha sido denegado, muestra información de contacto por si el cliente desea hablar con el personal de Puerto Venecia.

Hay tres botones que se comparten en ambos estados del ticket:

- **Subir otro ticket** abre la pantalla de subida de ticket (ver apartado siguiente).
- **Ver mis retos y campañas** abre la pantalla con el listado de retos y campañas (ver apartados 5.2.9 y 5.2.10 respectivamente).
- **Consultar mis tickets** abre de nuevo la pantalla de listado de tickets.

#### *5.2.8.2. Subida de ticket*

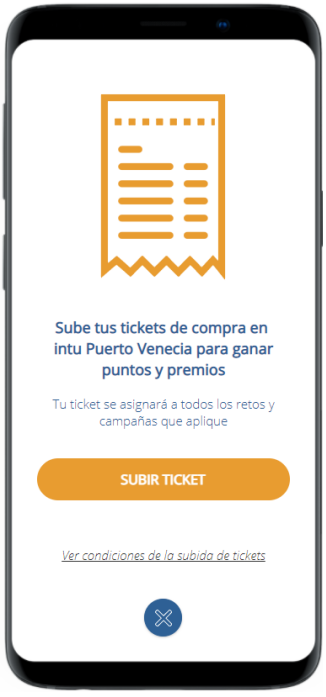



*Ilustración 26 - Pantalla de subida de Ticket*

En esta pantalla el usuario puede subir una foto de un ticket de alguno de los establecimientos del centro comercial. Si hay alguna campaña o reto que requiera tickets para completarse, este será puntuable siempre y cuando sea válido y cumpla las condiciones del reto o campaña.

Al pulsar en el botón "Subir ticket" se abrirá el selector de archivos del dispositivo, en el cual se dará opción a seleccionar una imagen usando la cámara o la galería del dispositivo. Una vez se seleccione la imagen del ticket, se subirá automáticamente y se le mostrará un mensaje al usuario indicando el resultado de la subida (subida correcta o incorrecta).

Al pulsar en "Ver condiciones de la subida de tickets" se abrirá una página web internamente en la aplicación donde el usuario podrá leer las condiciones de la subida de sus tickets.

### *5.2.9. Retos*


*Ilustración 27 - Pantalla de Retos*

Las tareas o acciones que tiene que realizar un usuario para conseguir puntos de experiencia se denominan retos. Al igual que en la aplicación de referencia Pokémon GO, el usuario debe completar dichos retos en un determinado tiempo para conseguir una recompensa (cupón).

En esta pantalla se muestran todos los retos activos que tiene el usuario. Cada reto muestra una descripción de la tarea y una barra de progreso con el número de acciones de dicha tarea completadas. Cuando el usuario completa un reto se muestra un botón sobre él por el cual puede reclamar los premios correspondientes.

En la parte superior se muestra el progreso del usuario en el super cofre. Hay retos que forman parte de un super cofre. Cada reto de este tipo completado suma un punto en el progreso. Cuando el usuario llega a 5 puntos, el super cofre se desbloquea y puede reclamar premios extra. Una vez se reclaman los premios, se genera un nuevo super cofre para completar.

En la parte inferior derecha de la pantalla aparece un botón el cual abre la pantalla de subida de ticket, botón que estará presente en varias pantallas de la aplicación.

### *5.2.10. Campañas*


*Ilustración 28 - Pantalla de listado de Campañas*

Esta pantalla muestra un listado con todas las campañas activas en la aplicación. En cada ítem muestra información relevante sobre la campaña, como el periodo de duración, los desafíos disponibles o el progreso actual.

Al pulsar sobre una campaña, se abre una nueva pantalla con sus detalles. Como vamos a ver ahora, dependiendo del tipo de campaña se mostrará una pantalla de detalles diferente, ya que no todas las campañas funcionan igual ni requieren las mismas acciones.

#### *5.2.10.1. Campaña general*


*Ilustración 29 - Pantalla de Campaña General*

Este tipo de campaña es el más sencillo. Está compuesta por varios retos, y una vez el usuario los completa todos, puede cobrar los premios pertenecientes a la campaña.

Aparece en primer lugar una descripción bastante similar a la vista en el listado de campañas. Justo después aparece una descripción más extensa de la campaña y un listado con los retos que se deben completar para finalizarla.

Al final del todo aparecen los premios que el usuario conseguirá si completa la campaña, de los cuales siempre se va a asignar una medalla. Vemos que el botón de la parte inferior derecha de la pantalla, el cual lleva a la pantalla de subida de tickets, sigue presente.

#### *5.2.10.2. Campaña tipo Black Week*

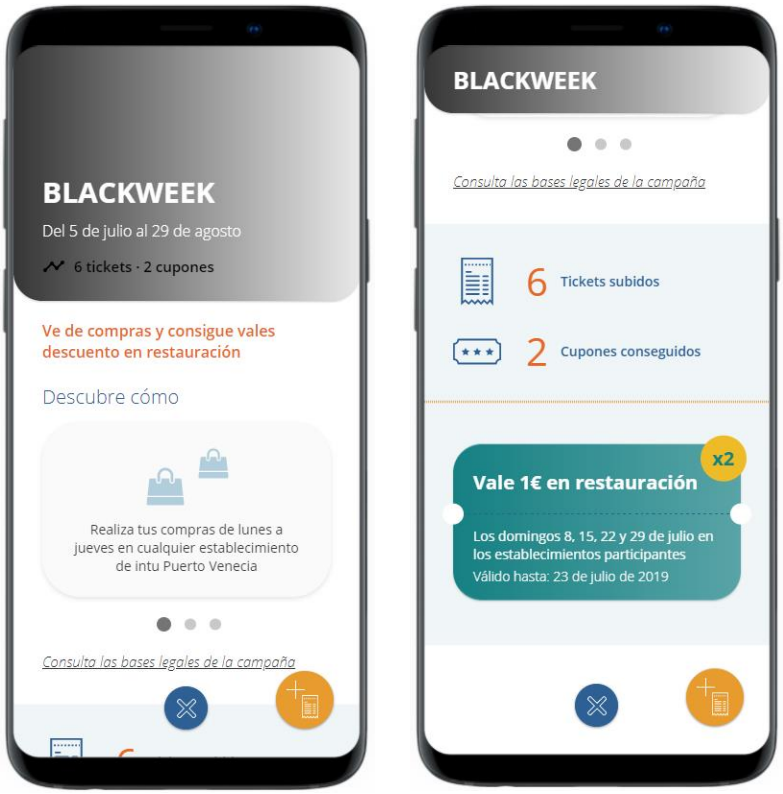



*Ilustración 30 - Pantalla de Campaña tipo Black Week*

Para este tipo de campaña tenemos elementos comunes con la anterior, como son la cabecera informativa o el botón para acceder a la pantalla de subida de tickets. Pero el formato de esta campaña es distinto.

En este tipo de campaña se puntúa simplemente al subir tickets durante su periodo de duración. Las condiciones para puntuar en la campaña se muestran en un pequeño slider situado justo después de la cabecera. Este slider viene acompañado de un botón para consultar las bases legales de la campaña, las cuales se abren como página web de manera interna en la aplicación.

En la parte inferior muestra el número de tickets subidos y los correspondientes cupones que se han conseguido con esos tickets. El usuario debe ser rápido, ya que hay un número limitado de cupones y la campaña deja de estar disponible cuando se consiguen todos.

#### *5.2.10.3. Campaña tipo Tournament*


*Ilustración 31 - Pantallas de Campaña tipo Tournament*

Este tipo de campaña se basa en la cooperación de usuarios. A este tipo de campañas se asignan una serie de equipos, normalmente pertenecientes a colegios o universidades. Para participar en esta campaña el usuario debe apuntarse en uno de los equipos participantes. Para ello, debe pulsar en el botón "Seleccionar colegio" y seleccionar el equipo correspondiente.

Una vez seleccionado el equipo, se muestra el número de puntos que tiene actualmente y el puesto en el que se encuentra, como se muestra en la segunda pantalla. Justo después, se muestra la aportación que el usuario ha tenido para ese equipo.

La forma de puntuar en este tipo de campañas, al igual que en el tipo Black Week, es mediante subida de tickets. Contra más tickets suba un usuario, más puntos generará para el equipo al que pertenezca. El usuario puede cambiar de equipo siempre que quiera, manteniendo sus puntos aportados.

Al igual que en el resto de las campañas, se muestra una cabecera informativa y un botón para acceder a la pantalla de subida de tickets.

### *5.2.11. Cupones*


*Ilustración 32 - Pantalla de listado de Cupones*

Esta pantalla muestra el listado de premios o cupones que el usuario ha conseguido, ya sea mediante retos, campañas o super cofre. Es posible que el usuario tenga más de un cupón del mismo tipo, en cuyo caso se indica con un contador en la parte superior derecha del ítem correspondiente.

Al pulsar sobre un cupón se abre su pantalla de detalles, siempre y cuando esté disponible para canjear. Si un cupón ha caducado o no se puede usar en ese momento, se muestra un candado que evita que el usuario pueda abrir los detalles.

#### *5.2.11.1. Detalles de cupón*

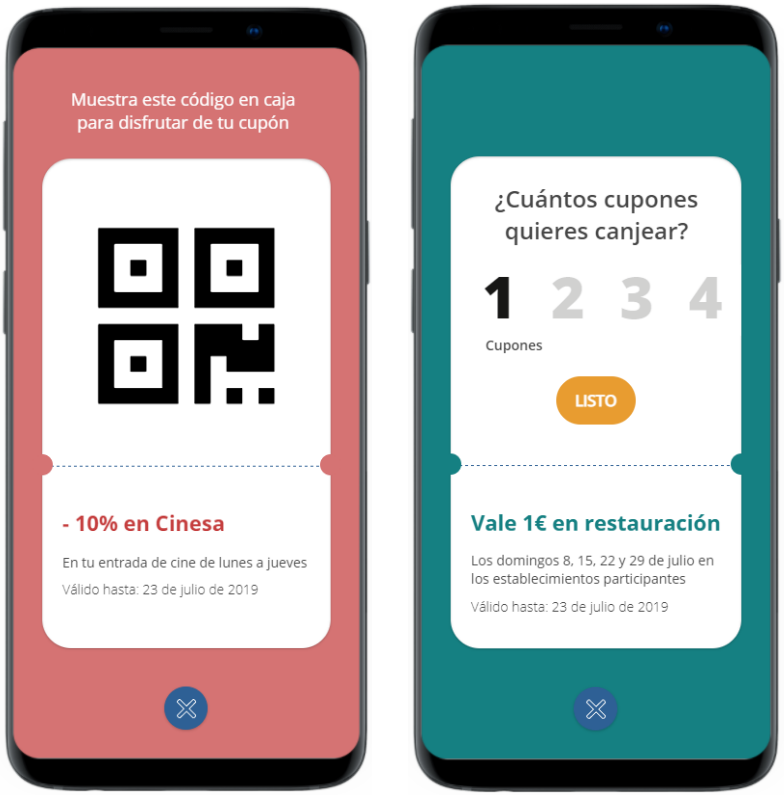



*Ilustración 33 - Pantalla de detalles de Cupón*

La vista de detalles de un cupón depende del número de cupones de ese tipo del que se disponga.

Si el usuario tiene un único cupón de ese tipo, se muestra una descripción del cupón y un código QR que deberá escanear el personal del establecimiento donde se desee canjear. Una vez canjeado, se mostrará un mensaje al usuario y el cupón desaparecerá del listado.

Si por el contrario el usuario tiene más de un cupón de ese tipo, primero debe seleccionar el número de cupones de ese tipo que desea canjear, hasta un máximo de 10. Una vez el usuario selecciona el número de cupones que desea canjear, se abre la vista anterior con un código QR que deberá ser escaneado por el personal del establecimiento.

# 6. Desarrollo

Este apartado pretende introducir algunos conceptos trabajados en este proyecto, para posteriormente describir las características y toma de decisiones llevadas a cabo durante su desarrollo.

Esta última parte estará dividida en sprints, siguiendo la estructura de planificación recogida en el apartado 4.2 "Planificación".

## 6.1. Componentes y arquitectura de la aplicación

Antes de explicar la arquitectura de la aplicación, es necesario conocer todos sus componentes. En Android se pueden clasificar de dos maneras: componentes visibles y componentes internos, como se puede ver a continuación.

### *6.1.1. Vista (View)*

Las vistas [16] son los componentes básicos con los que se construye la interfaz gráfica de la aplicación. Android pone a disposición de los desarrolladores una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

Las vistas pueden ser definidas en código, aunque la forma habitual de definirlos es utilizando código XML, algo que se asemeja bastante al lenguaje HTML en una página web.

### *6.1.2. Layout*

Un Layout [17] es un contenedor de una o más vistas y controla su comportamiento y posición. Por norma general, una vista tiene que estar contenida siempre dentro de un layout. Existen diferentes tipos, dependiendo de la posición que se le quiera dar a las vistas de su interior (lineal, cuadricular, superpuestas, adaptadas a la pantalla del dispositivo...). Además, un layout puede contener más layouts en su interior.

### *6.1.3. Actividad (Activity)*

Una actividad [18] es el componente principal de la interfaz gráfica de una aplicación en Android. A cada actividad se le asigna una ventana en la cual se dibuja la interfaz de usuario, con la que el usuario podrá interaccionar para realizar las diversas acciones que hayamos contemplado en la aplicación.

Una aplicación para Android puede contener una o más actividades, es decir, una o más pantallas. La aplicación para Android comienza mostrando la actividad principal, y a partir de ahí la aplicación puede hacer posible abrir actividades adicionales.

El ciclo de vida de una actividad es el siguiente:

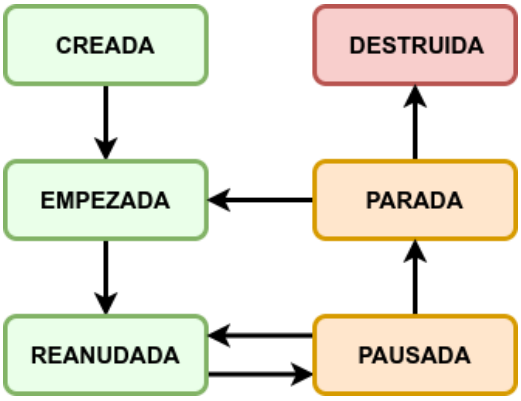



*Ilustración 34 - Ciclo de vida de una actividad en Android*

Cuando se inicia una aplicación para Android, se crea la actividad principal. La actividad pasa entonces por 3 estados antes de estar lista para servir al usuario: Creada, iniciada y reanudada. Si la actividad principal puede abrir cualquier otra actividad (pantallas) estas actividades pasarán por los mismos 3 estados cuando se abran.

Si una actividad A abre otra actividad B, entonces la actividad A se detendrá. Esto significa que la actividad A entra en el estado de pausa. Cuando el usuario hace clic en el botón de regreso y regresa a la actividad A, la actividad A regresa al estado de reanudada.

Si el usuario vuelve a la pantalla de inicio del dispositivo Android, todas las actividades se detendrán y luego se pararán. Si el usuario regresa a la aplicación, las actividades pasarán por los estados de inicio y reanudación. Si el dispositivo Android necesita la memoria que la aplicación ocupa en él, éste puede destruir completamente la aplicación, lo que significa que las actividades de la aplicación pasan al estado de destruida.

### *6.1.4. Fragmento (Fragment)*

Un fragmento [19] está formado por la unión de varias vistas para crear un bloque funcional de la interfaz de usuario. Un fragmento viene a ser una pieza de una actividad que permite un diseño de actividad más modular. De hecho, se suele decir que un fragmento es una especie de sub-actividad.

Hay algunas observaciones de los fragmentos que hay que tener en cuenta:

- Un fragmento tiene sus propias vistas.

- Se puede añadir o quitar fragmentos de una actividad mientras la actividad está en ejecución.
- Se pueden combinar múltiples fragmentos en una sola actividad.
- Un fragmento puede utilizarse en múltiples actividades.

Al igual que la actividad, un fragmento también tiene un ciclo de vida. Pero, al estar contenido dentro de una actividad, su ciclo de vida depende de ésta.

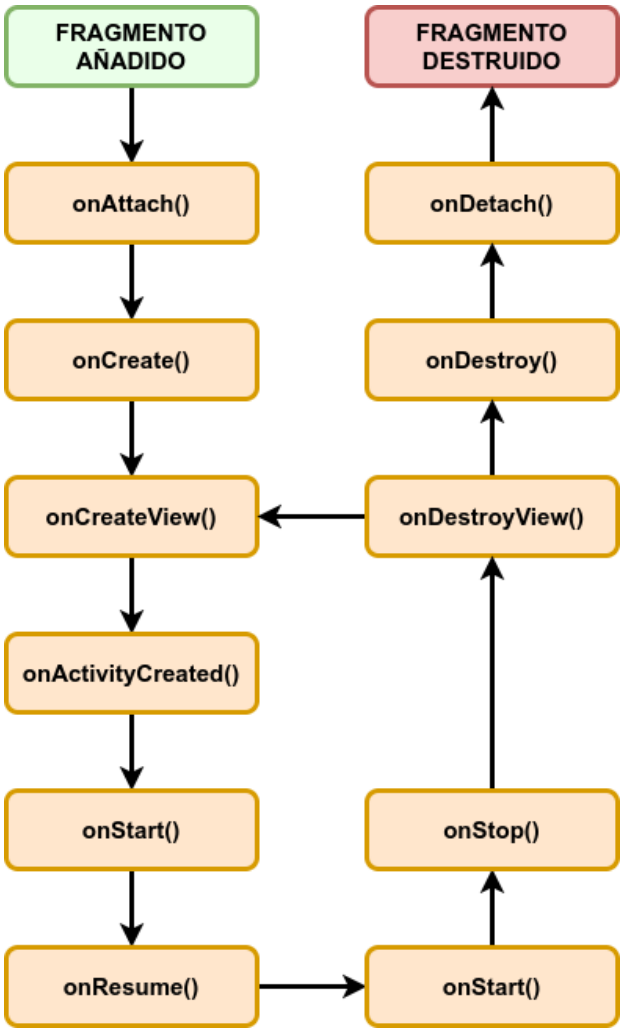



*Ilustración 35 - Ciclo de vida de un fragmento en Android*

Primero se añade un fragmento a una actividad. Esto inicia el ciclo de vida del fragmento.

Una vez añadido, se llaman los métodos *onAttach()*, *onCreate()*, *onCreateView()*, *onActivityCreated()*, *onStart()* y *onResume().* El métido *onActivityCreated()* se llama cuando la actividad contenedora está completamente creada.

Si el fragmento se elimina de su actividad contenedora, o dicha actividad se mueve al fondo de la aplicación (porque otra actividad se mueve al primer plano), entonces se llaman los métodos *onPause()*, *onStop()* y *onDestroyView()*. Si el fragmento vuelve a ser visible, puede pasar de *onDestroyView()* a *onCreateView()* y volverse visible de nuevo.

### *6.1.5. Servicio (Service)*

Un servicio [20] en Android se utiliza para realizar operaciones en segundo plano, como reproducir música, manejar operaciones de red, interactuar con proveedores de contenido, etc. Al contrario que las actividades o fragmentos, no tiene ningún tipo de interfaz de usuario.

Existen dos tipos de servicio:

- **Servicios foreground**: Se ejecutan en segundo plano mientras la aplicación esté ejecutándose. Una vez se cierra la aplicación, el servicio es destruido.
- **Servicios background**: Se ejecutan en segundo plano incluso cuando la aplicación ha sido finalizada. Es el propio sistema operativo el que se encarga de gestionar el tiempo de ejecución del servicio, el cual depende, entre otros aspectos, del consumo de batería o el uso de memoria.

Los servicios, al igual que las actividades o los fragmentos, también tienen un ciclo de vida.

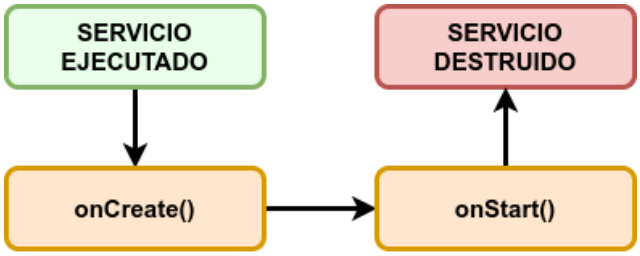



*Ilustración 36 - Ciclo de vida de un servicio en Android*

El ciclo de vida de un servicio es muy simple. Cuando se ejecuta un servicio, primero se llama al método *onCreate()* y seguidamente al método *onStart()*. Cuando una actividad o el propio sistema operativo paran el servicio, se llama al método *onDestroy()* y se destruye el servicio.

### *6.1.6. Intent*

Un Intent [21] en Android es un mensaje que se pasa entre los componentes de la aplicación, como las actividades, los proveedores de contenido, los receptores de emisión, los servicios, etc. En otras palabras, es una "intención" de realizar una acción.

Dependiendo de la acción, las aplicaciones o el sistema operativo podrían estar escuchando y reaccionarán en consecuencia.

### *6.1.7. Receptor de emisión (Broadcast Receiver)*

Un receptor de emisión [22] es un componente que permite al sistema enviar eventos a la aplicación, como por ejemplo enviar un mensaje de batería baja. Las aplicaciones también pueden iniciar transmisiones para que otras aplicaciones sepan que los datos necesarios están disponibles en un dispositivo para utilizarlo.

Por lo general, se usan Intents para entregar eventos de emisión a otras aplicaciones.

### *6.1.8. Proveedores de contenido (Content Provider)*

Los proveedores de contenido [23] son útiles para intercambiar datos entre aplicaciones en función de las solicitudes. Los proveedores de contenido pueden compartir los datos de la aplicación que se almacenan en el sistema de archivos, en una base de datos SQLite, en una web o en cualquier otra ubicación de almacenamiento a la que nuestra aplicación pueda acceder.

Mediante el uso de los proveedores de contenido, otras aplicaciones pueden consultar o modificar los datos de nuestra aplicación en función de los permisos proporcionados por el proveedor de contenido. Por ejemplo, android proporciona un proveedor de contenido (ContactsContract.Data) para administrar la información de los contactos. Mediante el uso de los permisos adecuados cualquier aplicación puede consultar al proveedor de contenido para realizar operaciones de lectura y escritura en la información de los contactos.

### *6.1.9. Archivo Manifest*

El archivo *AndroidManifest.xml* [24] es un archivo de configuración para la aplicación y contiene la información sobre Actividades, Intents, Proveedores de Contenido, Servicios, Receptores de Emisión o permisos que tiene la aplicación. Además, incluye información acerca del nombre, número de versión o el icono de inicio.

Al usar Android Studio para crear una aplicación, se genera automáticamente el archivo de manifiesto, y la mayoría de los elementos esenciales de este se van agregando a medida que se compila la aplicación.

### *6.1.10. Recursos (resources)*

Los recursos son todos los elementos externos que requiere la aplicación, como por ejemplo imágenes, audio, video, textos, ficheros de texto plano... Vienen a ser todos los elementos de la aplicación que no contienen código.

### *6.1.11. Librerías*

Las librerías son piezas de código que se pueden reutilizar entre aplicaciones. Imagina que quieres desarrollar una funcionalidad en la aplicación, por ejemplo, un gráfico estadístico. Es muy posible que exista una librería que pueda hacer esto por ti.

Existen librerías oficiales, las cuales son proporcionadas por el propio Google, y librerías de terceros, que son atendidas por usuarios. Android es un sistema operativo de código abierto, por lo que tiene una gran comunidad de desarrolladores encargados de elaborar librerías.

Android Studio [3] incluye un gestor de dependencias llamado Gradle [6] mediante el cual podemos añadir librerías a la aplicación, simplemente añadiendo una línea de código con su nombre de paquete y la versión que se desea utilizar.

### *6.1.12. Patrón MVVM (Model-View-ViewModel)*

Tener una aplicación con un código limpio es crucial para un buen funcionamiento de la misma. Además, seguir un patrón de diseño garantiza una buena transmisión de conocimientos a nuevos desarrolladores.

Los patrones de diseño ofrecen una arquitectura y un conjunto de reglas definidos previamente, los cuales deben de ser cumplidos durante el desarrollo de nuestra aplicación.

El principal objetivo del patrón MVVM [15] es tratar de desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación. Vamos a ver cada una de sus partes:

#### *6.1.12.1. El modelo (Model)*

El modelo se suele definir como el objeto del dominio. El modelo representa los datos y/o la información real con la que estamos trabajando. Un ejemplo de un modelo podría ser un contacto, el cual contiene nombre, número de teléfono, dirección, etc.

El modelo contiene la información, pero no los comportamientos o servicios que manipulan la información. No es responsable de formatear el texto para que se vea bonito en la pantalla, o de obtener una lista de elementos de un servidor remoto. Sólo contiene información.

#### *6.1.12.2. La vista (View)*

La vista es lo único con lo que el usuario final interactúa realmente. Es la presentación de los datos. La vista se toma ciertas libertades para hacer estos datos más presentables. Por ejemplo, una fecha puede ser almacenada en el modelo como milisegundos, y presentarse en la vista como día, mes y año.

Una vista también puede tener comportamientos asociados a ella, como aceptar entrada de texto del usuario. La vista gestiona la entrada de texto (pulsaciones de teclas, movimientos del ratón, gestos de tacto) lo que en última instancia manipula las propiedades del modelo.

Lo interesante de la vista es que no es responsable de mantener su estado, ni la información que permanece en ella. Esto lo sincronizará con el modelo de la vista.

#### *6.1.12.3. El modelo de la vista (ViewModel)*

El modelo de vista (controlador) es una pieza clave, ya que se encarga de mantener la vista separada del modelo. En lugar de hacer que el modelo conozca la visión que el usuario tiene de una fecha (día, mes y año), el modelo simplemente mantiene los datos en milisegundos, la vista mantiene la fecha formateada, y el controlador actúa como el enlace entre ambos (es quien la formatea). El controlador puede tomar los valores de entrada del usuario de la vista y colocarlos en el modelo, o puede interactuar con un servicio para recuperar el modelo, y luego traducir las propiedades y colocarlo en la vista.

El modelo de vista también expone métodos, comandos y otros puntos que ayudan a mantener el estado de la vista, manipular el modelo como resultado de acciones en la vista, y desencadenar eventos en la propia vista.

En la siguiente imagen, cortesía de Alberto Moral, podemos ver un esquema del funcionamiento de este patrón.

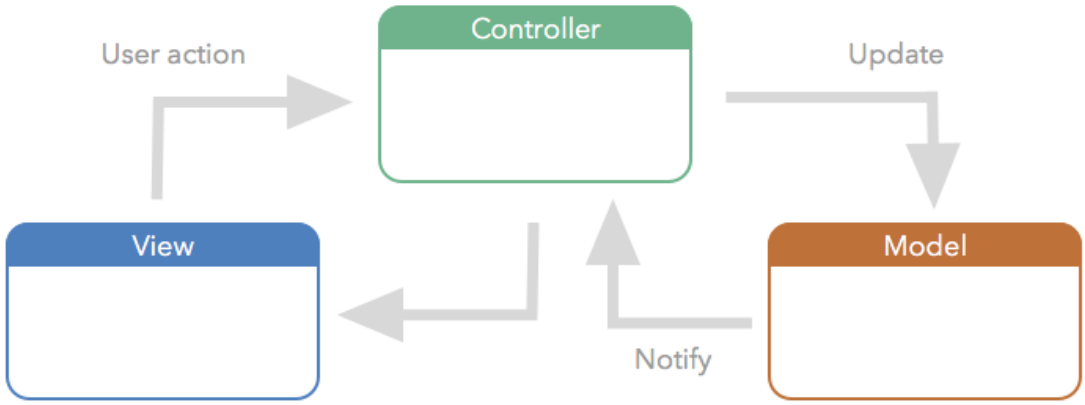



*Ilustración 37 - Architectural Pattern: Apple MVC's variation [25]*

## 6.2. Sprint 1

Este sprint comprende todas las tareas llevadas a cabo como inicio del proyecto. Esto incluye un estudio de la arquitectura, metodología y diseño que se utilizará durante todo su desarrollo.

Inicialmente se planteó este sprint en 4 semanas de duración. No obstante, las tareas de definición y diseño de la arquitectura ocuparon menos tiempo del previsto. Como consecuencia, pudimos adelantar el desarrollo del siguiente sprint.

### *6.2.1. Decisiones previas al desarrollo*

Antes de comenzar con el desarrollo del proyecto se decide marcar unas pautas para ello. Estas pautas definirán la estructura de archivos, clases, variables, uso de librerías externas y entorno de Git.

#### *6.2.1.1. Estructura de archivos*

Se decide junto al resto del equipo de desarrollo una estructura de archivos en común, que incluye la ubicación de éstos, su nombre, y el nombre de sus variables.

Se deciden crear cinco módulos (o capas) para la aplicación: *app*, *data*, *datasources*, *domain* y *model*.

- La **capa de aplicación** (o módulo de presentación) contiene todo lo relacionado con las partes de nuestra aplicación a las que se enfrenta el usuario, como Actividades, Fragmentos...
- La **capa de datos** es el punto de acceso a las capas de datos externos y se utiliza para obtener datos de múltiples fuentes (como la caché y la red).
- La **capa de fuente de datos** contiene las implementaciones de obtención de datos para cada una de las fuentes utilizadas.
- La **capa de dominio** es el núcleo de la aplicación. Este módulo contiene los casos de uso que interactúan con la aplicación, como por ejemplo obtener la información de usuario, actualizarla, obtener la lista de cupones...
- La **capa de modelo** contiene todos los objetos de datos que serán abstraídos de la capa de dominio para ser utilizados en la capa de presentación.

Dentro de cada módulo se realiza una distinción por clases y secciones de la aplicación. Por ejemplo, en el módulo de fuente de datos se realiza una distinción entre la fuente de red y la fuente de caché. En los módulos de aplicación y modelo, se realiza una distinción por sección de la aplicación (login, registro, home, perfil, campañas...).

Respecto al nombre de los archivos y sus variables, se sigue la convención CamelCase:

- Los nombres de carpetas permanecen siempre en minúsculas.
- Los nombres de archivos y clases siguen la forma UpperCamelCase. Por ejemplo: *SplashActivity*, *HomeFragment*, *NotificationPushService*.
- Los nombres de variables siguen la forma lowerCamelCase. Por ejemplo: *ticketList*, *name*, *userPhoneNumber*.

Con esta serie de reglas se asegura una estructura de código común entre todos los desarrolladores.

#### *6.2.1.2. Uso de librerías externas*

Se decide aceptar el uso de librerías externas que cumplan con los siguientes requisitos:

- Deben ser librerías con cierta reputación y de código abierto. Como todas las librerías de código abierto suelen estar en un repositorio Git, podemos conocer su reputación en base al nivel de actividad que tiene el repositorio. Si la librería lleva tiempo sin actualizarse, o maneja un nivel de usuarios muy bajo, es posible que dicha librería quede archivada o se abandone en un corto periodo de tiempo, por lo que tiene que descartarse su uso.
- Deben contener las últimas tecnologías disponibles, sin exceso de métodos obsoletos o funciones restringidas para ciertas versiones del sistema operativo.
- En el caso de necesitar una librería que no cumpla estas condiciones, ésta deberá ser mantenida por nosotros, lo que puede atrasar el desarrollo del proyecto. Es necesario mantener siempre la autoría original de la librería.

#### *6.2.1.3. Estructura en Git*

Por defecto en el repositorio de Git se crean dos ramas: *master* y *develop*. La rama *master* contiene todo el trabajo terminado y validado, mientras que la rama *develop* contiene los trabajos en desarrollo de tareas.

A la hora de trabajar en una nueva funcionalidad se realizará siempre sobre una rama específica para ello (heredada de la rama *develop*), cuyo nombre seguirá la estructura *feature/nombre-de-tarea*. Cuando esta funcionalidad sea crítica para el desarrollo, el nombre de la rama seguirá la estructura *bugfix/nombre-de-error*.

Al finalizar una rama, antes de enviarla a la rama padre *develop*, será necesario realizar una validación o pull request. Esta validación la realizará el analista programador, quien dará un feedback final y aceptará la fusión con la rama padre.

Al final de cada Sprint es necesario generar una nueva versión o release de la aplicación. Para llevar un control sobre el contenido de cada versión, antes de actualizar el número en la aplicación es necesario crear una nueva rama con la estructura *release/vX.X.X* la cual contenta el número de versión. Tras actualizar el número de versión, esta rama se fusiona tanto en *develop* como en *master* y se genera un nuevo archivo de aplicación compilado para el cliente.

### *6.2.2. Desarrollo de tareas*

En el departamento de desarrollo nativo para dispositivos móviles de la empresa Hiberus Tecnologías Diferenciales se trabaja internamente en el desarrollo de una plantilla genérica para cada plataforma (Android e iOS). Estas plantillas contienen elementos comunes que es muy probable que sean utilizados en todos los proyectos de la empresa.

Todos los proyectos se crean a partir de estas plantillas las cuales contienen, entre otras cosas, una primera definición de la arquitectura anteriormente mencionada, así como las pantallas de Splash Screen y tutorial. Por lo tanto, con unos simples cambios en los textos e imágenes de estas pantallas, se consigue finalizar su desarrollo.

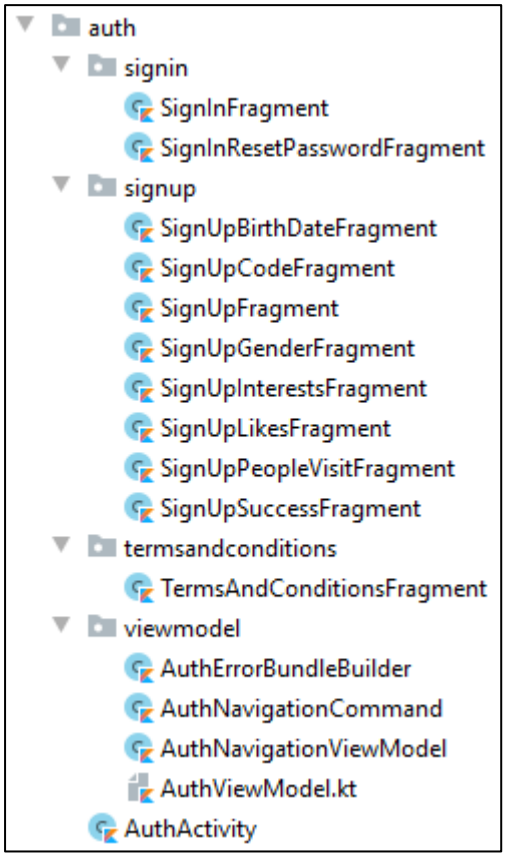
En este primer sprint se comienza con el desarrollo de la funcionalidad de la aplicación, concretamente con la funcionalidad de login y registro.


*Ilustración 38 - Tareas para Sprint 1*

### *6.2.3. Pantalla de Login y registro*

En todas las capas de la aplicación, ambas secciones de la app se han unificado dentro de una carpeta llamada *auth*.

Como veremos a lo largo de toda la aplicación, y siguiendo con el patrón MVVM, todas las pantallas contienen un elemento ViewModel que se encarga de gestionar los datos de cada pantalla. El ViewModel viene siempre acompañado de una clase encargada de gestionar los posibles errores, denominada *ErrorBundleBuilder*. Estos elementos se encuentran en la capa de presentación, como podemos ver en la imagen de su estructura.



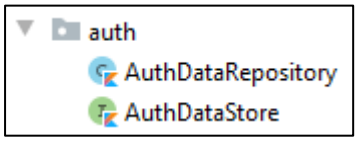

La pantalla de inicio de sesión se compone de dos fragmentos. El fragmento principal *SignInFragment* contiene el formulario de inicio de sesión, donde el usuario tendrá que introducir su email y contraseña.

El fragmento *SignInResetPasswordFragment* contiene un formulario para recuperar la contraseña de usuario, siendo necesario introducir el email de la cuenta.

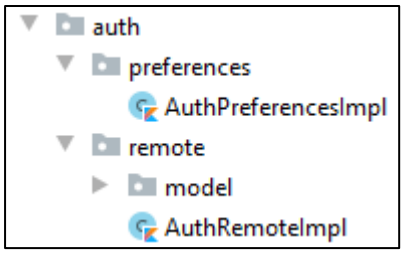
La pantalla de registro es más compleja. El fragmento principal *SignUpFragment* contiene un formulario básico de registro de sesión. Inicialmente se decidió utilizar únicamente cuatro campos: nombre, email, código de referido y contraseña. Durante el desarrollo de esta pantalla se decide añadir dos nuevos campos: número de teléfono y código postal.

Si el usuario tiene un código de referido para el registro se muestra el fragmento *SignUpCodeFragment*. En este fragmento el usuario debe introducir el código para posteriormente volver a la pantalla anterior.

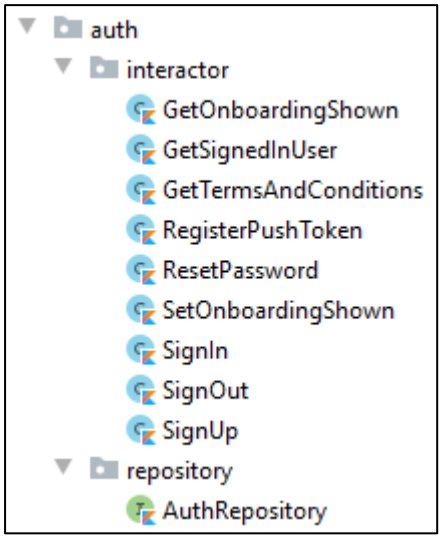
Una vez el usuario se registra con éxito, entra en juego el fragmento *SignUpSuccessFragment*. Este fragmento, además de dar la bienvenida al usuario, le permite completar su registro con ciertos campos opcionales. Para cada uno de estos campos se utiliza un fragmento independiente, los cuales se van mostrando en forma de secuencia permitiendo al usuario salir de ella en cualquier momento. Si el usuario decide completar alguno de estos campos opcionales más tarde, o los ha completado todos, se inicia sesión automáticamente y se inicia la pantalla de Home.



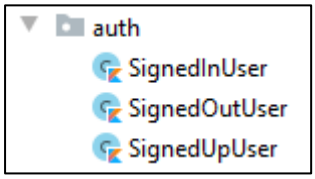

Esta es la estructura de la capa de datos. Contiene la gestión de las fuentes de datos a utilizar.




Esta es la estructura de la capa de fuentes de datos. En este caso se requiere la fuente en remoto y una fuente interna de preferencias para guardar los tokens de acceso. Estos tokens permiten acceder a la aplicación sin necesidad de iniciar sesión de nuevo.




Esta es la estructura de la capa de dominio. Contiene todas las acciones a realizar en ambas pantallas además de la interfaz de repositorio que será implementada en la capa anterior.




Esta es la estructura de la capa de modelo. Para estas dos pantallas, sólo es necesario utilizar estos tres modelos.

En la última semana de este sprint se comenzó con el desarrollo de la siguiente pantalla, aunque su desarrollo completo se detalla en el siguiente sprint.

## 6.3. Sprint 2

Este sprint comienza con el desarrollo de la pantalla de inicio o Home. Esta pantalla es algo más compleja, ya que el cliente manifestó la necesidad de implementar un resumen de todas las secciones de la aplicación en esta pantalla.

Esta es la lista completa de tareas para este sprint:

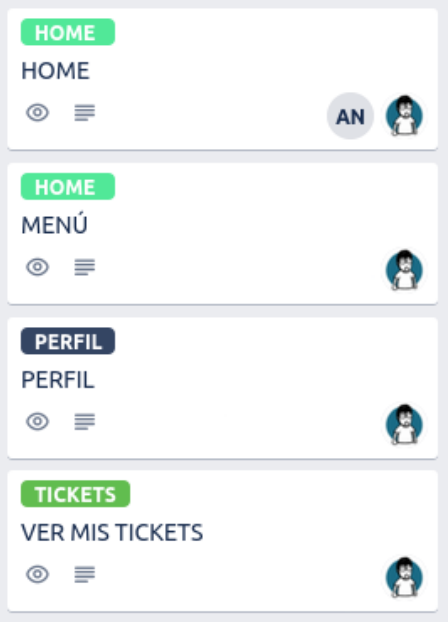
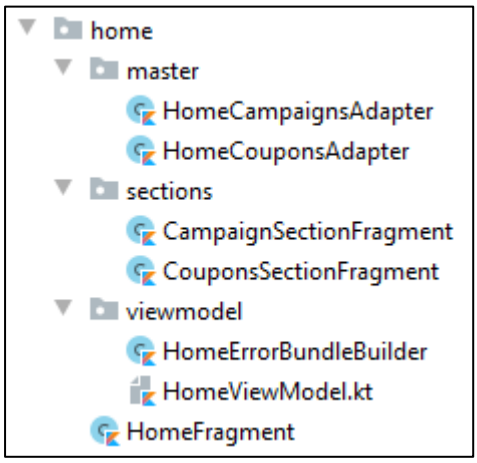



*Ilustración 39 - Tareas para Sprint 2*

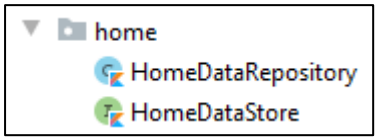
### *6.3.1. Pantalla de inicio (Home)*

Esta pantalla comparte contenido con las pantallas de perfil, campañas, cupones y retos. Como estas pantallas no se han desarrollado todavía, se realiza una implementación básica a la espera de tener el resto de las pantallas desarrolladas.

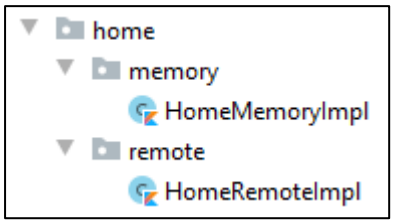



La capa de aplicación de esta pantalla, además de incluir el fragmento principal y el ViewModel, incluye dos fragmentos internos que contienen una lista horizontal con los cupones y campañas disponibles.

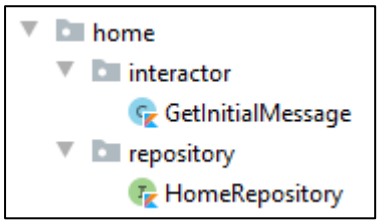
En Android, todas las listas deben estar gestionadas por un Adaptador, el cual se encarga de añadir los datos a cada uno de los elementos de la lista.




La capa de datos sigue la misma estructura que la pantalla anterior.




La capa de fuentes de datos cambia con respecto a la pantalla anterior. Además de una fuente en remoto, se implementa una fuente en memoria (caché) para garantizar la permanencia de datos en toda la aplicación. Esto evita realizar una llamada a remoto cada vez que necesitamos el contenido.

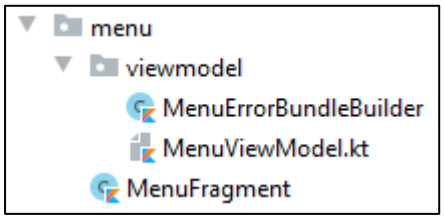



En la capa de dominio se han incluido las acciones específicas de esta pantalla. Al compartir contenido con las pantallas de cupones, perfil, campañas y retos, se usan las acciones específicas de esas pantallas.


Con la capa de modelo pasa lo mismo, sólo se incluyen los modelos específicos de esta pantalla.

#### *6.3.1.1. Menú*

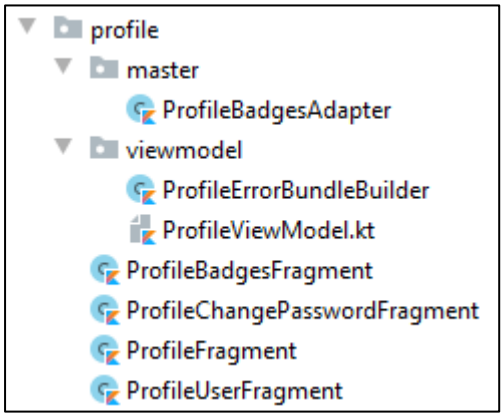
Durante el desarrollo de la pantalla de Home se trabaja también en la pantalla de menú. Estas pantallas son complementarias, aunque se tratan como pantallas diferentes.




La estructura de esta pantalla es muy simple, ya que toda la lógica está concentrada en la capa de aplicación. Se compone de un fragmento y un ViewModel encargado de gestionar todos los enlaces del menú.

### *6.3.2. Pantalla de Perfil*

El desarrollo se inicia al comenzar la tercera semana del sprint.



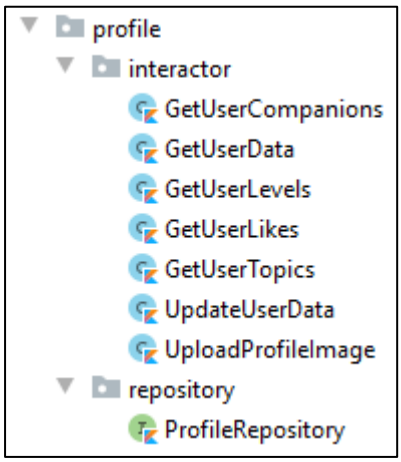

Esta pantalla se divide en dos apartados diferentes, siendo el fragmento *ProfileFragment* el encargado de gestionarlos.

El fragmento *ProfileBadgesFragment* contiene una lista con las diferentes medallas que el usuario puede conseguir completando campañas. Como todas las listas, tiene un adaptador el cual se encuentra en la carpeta "master".

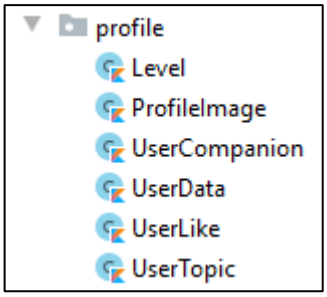
El fragmento *ProfileUserFragment* contiene toda la información de usuario, la cual puede ser actualizada fácilmente. Como se explicó en el ciclo de vida de un fragmento (apartado 6.1.4), es posible conocer cuando éste va a ser destruido. Es entonces cuando procedemos a guardar la información actualizada del usuario, una vez el usuario sale de esa pantalla.

Uno de los campos que puede ser actualizado por el usuario es su contraseña. Por cuestiones de seguridad, la contraseña no se puede actualizar directamente. Es necesario confirmar primero la contraseña antigua antes de cambiarla por una nueva. Esto se gestiona mediante un fragmento llamado *ProfileChangePasswordFragment*, el cual contiene tres campos: contraseña antigua, contraseña nueva y confirmar contraseña nueva. Una vez se realiza el cambio, se vuelve al fragmento anterior.

Las capas de datos y fuente de datos siguen la misma estructura vista en la pantalla de inicio.




En la capa de dominio nos encontramos con todas las acciones necesarias para obtener la información de usuario y actualizarla.




Y en la capa de modelo nos encontramos con los siguientes.

### *6.3.3. Pantalla de tickets (listado)*

La pantalla de tickets se comparte tanto en este sprint como en el siguiente. Durante este sprint se lleva a cabo el desarrollo de la parte del listado de tickets, y el siguiente sprint comenzará con la parte de detalles y subida de un ticket. Como la estructura de ambas partes es compartida, se va a comentar en el siguiente.

## 6.4. Sprint 3

Como se acaba de mencionar, este sprint comienza con el desarrollo de la vista de detalles y subida de tickets. Para desarrollar la vista de detalles era necesario tener implementada la lista de tickets, lo cual se realizó la semana anterior.

Esta es la lista completa de tareas para este sprint:

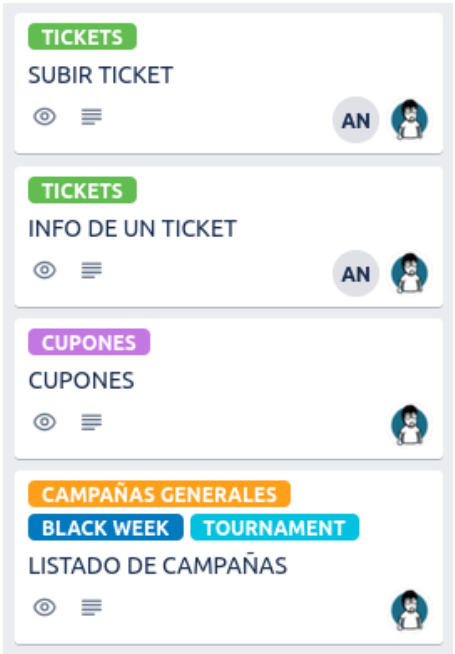
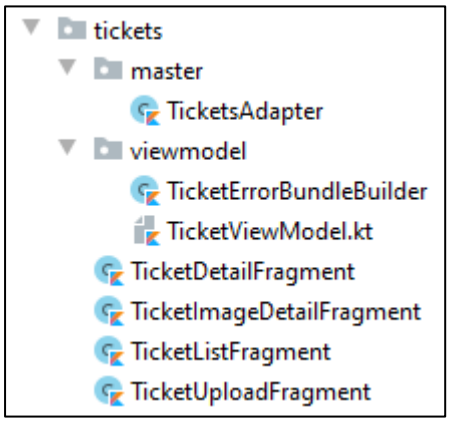



*Ilustración 40 - Tareas para Sprint 3*

### *6.4.1. Pantalla de tickets (detalles y subida)*



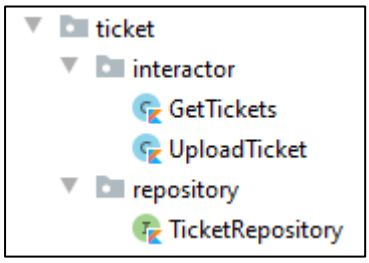

La pantalla de tickets está compuesta por cuatro fragmentos. El primero de ellos, *TicketListFragment*, contiene el listado de tickets.

El fragmento *TicketDetailFragment* contiene los detalles de un ticket como son su imagen, estado, importe, fecha o establecimiento. Inicialmente se deciden dos estados para esta pantalla: "validado" y "denegado", no siendo posible para el usuario ver los detalles de un ticket mientras está en proceso de validación. Durante el desarrollo de esta pantalla se decide añadir el estado "pendiente" para que el usuario tenga siempre la información del ticket en la aplicación.

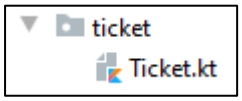
Desde el fragmento de detalles se puede visualizar la imagen del ticket escaneado, para lo cual se crea el fragmento *TicketImageDetailFragment*. Este fragmento contiene únicamente un visor de imágenes, para que el usuario visualice correctamente su ticket.

En último lugar se encuentra el fragmento *TicketUploadFragment*, mediante el cual se pueden subir imágenes de tickets. Este fragmento será accesible desde diferentes pantallas de la aplicación, como los detalles de una campaña o el propio listado de tickets.

Las capas de datos y fuente de datos siguen la misma estructura que las pantallas anteriores.

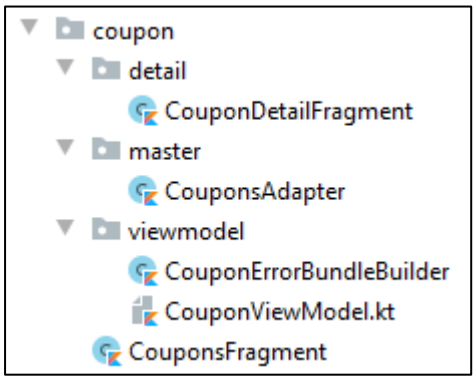



La capa de dominio contiene las siguientes acciones: Obtener listado de tickets y subir ticket.




Y en la capa de modelo nos encontramos con el siguiente.

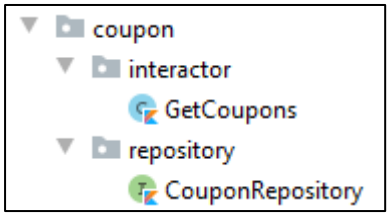
### *6.4.2. Pantalla de Cupones*



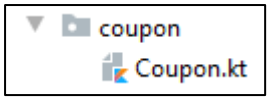

Esta pantalla de divide en dos fragmentos: Fragmento de listado de cupones *CouponsFragment* y fragmento de detalles de cupón *CouponDetailFragment*.

El contenido de ambos fragmentos se gestiona mediante un único ViewModel, ya que se decidió usar uno por pantalla.

Las capas de datos y fuente de datos siguen la misma estructura que las pantallas anteriores.




La capa de dominio incluye únicamente la acción de obtener la lista de cupones. Esto es debido a que todas las vistas de detalles, excepto casos excepcionales, recogen sus datos del propio listado.




En la capa de modelo nos encontramos con el siguiente.

### *6.4.3. Pantalla de Campañas y Retos (listado campañas)*

Las pantallas de campañas y retos se comparten tanto en este sprint como en el siguiente. Durante este sprint se lleva a cabo el desarrollo de la parte del listado de campañas, y el siguiente sprint comenzará con la parte de detalles de campañas. Como la estructura de ambas partes es compartida, se va a comentar en el siguiente.

## 6.5. Sprint 4

Este es el último sprint del desarrollo de este proyecto. Comprende el desarrollo de los detalles para los diferentes tipos de campañas, así como la pantalla de retos. Tras la finalización de este sprint se consigue llegar a los objetivos propuestos en un principio, y con ello dar por finalizado el desarrollo de la parte Front-End de aplicación para Android, de la cual trata este proyecto.

Esta es la lista completa de tareas para este sprint:

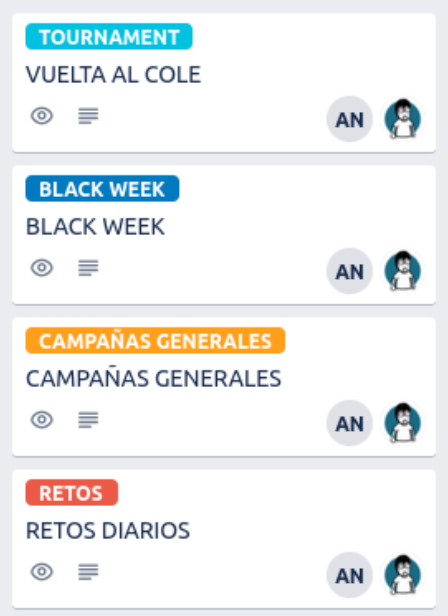
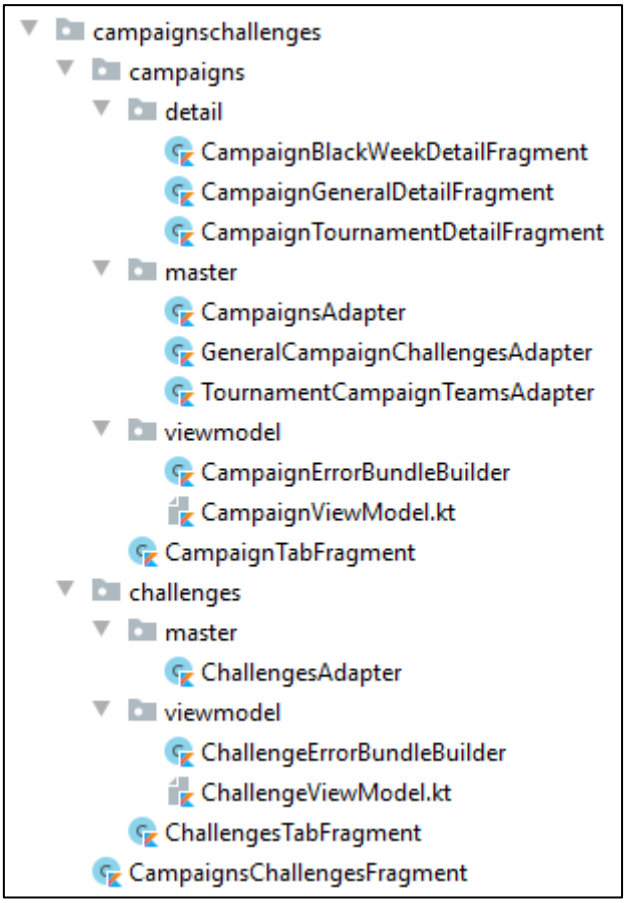



*Ilustración 41 - Tareas para Sprint 4*

### *6.5.1. Pantalla de Campañas y Retos*

Los retos dependen en gran parte de las campañas disponibles, ya que la mayoría de retos provienen de estas campañas. Por ello se decide unificar ambas secciones, aunque sean dos pantallas diferentes en la aplicación.



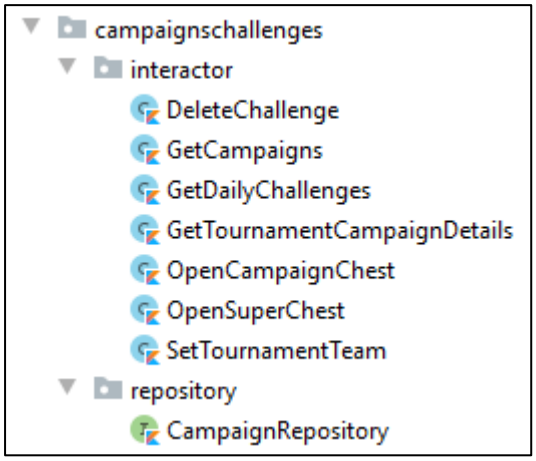

Ambas pantallas se gestionan mediante un fragmento padre *CampaignsChallengesFragment*. Este fragmento permite navegar entre estas dos pantallas.

La pantalla de campañas se divide en dos partes: listado de campañas y detalle de una campaña. *CampaignTabFragment* es el fragmento encargado de mostrar el listado de campañas. Como existen tres tipos de campaña, se ha creado un fragmento de detalles por tipo de campaña, ya que el contenido de cada una es bastante distinto.

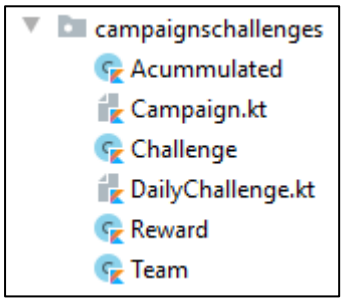
Además del adaptador para la lista de campañas, encontramos dos adaptadores más. El primero forma parte de la lista de retos de una campaña de tipo General, mientras que el segundo forma parte de la lista de equipos de una campaña de tipo Tournament.

Respecto a la pantalla de retos, nos encontramos únicamente con un fragmento que incluye tanto la lista de retos como el super cofre.

Las capas de datos y fuente de datos de ambas pantallas siguen la misma estructura que las pantallas anteriores.




En la capa de dominio nos encontramos todas las acciones necesarias para ambas pantallas. Estas acciones son: obtener lista de campañas, obtener lista de retos (diarios), borrar reto activo, obtener detalles de campaña tipo Tournament, abrir premio de campaña, abrir super cofre y elegir equipo para campaña tipo Tournament.




En la capa de modelo encontramos los siguientes.

### *6.5.2. Beacons, geovallado y notificaciones push*

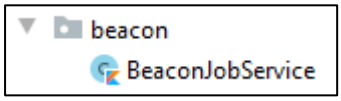
Durante el desarrollo de la pantalla de retos, fue necesario añadir ciertas funcionalidades específicas para su correcto funcionamiento.

Además de los retos de tipo "escanear ticket" existen retos de geolocalización en diferentes puntos del centro comercial. Los retos de geolocalización se pueden clasificar en:

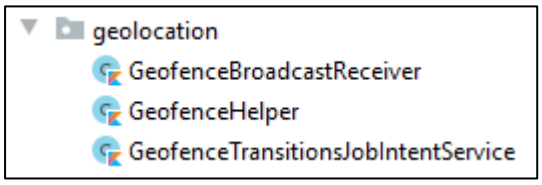
- **Visita a un establecimiento**: En este tipo de retos se pide al usuario que visite un establecimiento del centro comercial. Para detectar su visita se usan unos dispositivos Bluetooth de baja frecuencia llamados Beacons [26]. Estos dispositivos emiten una señal de corto alcance que es detectada por la aplicación. Cuando la aplicación detecta la señal de un Beacon, se confirma que ha visitado el establecimiento.
- **Visitar el centro comercial**: En este tipo de retos se pide al usuario visitar el centro comercial, independientemente del establecimiento. Para detectar su visita se usa Geovallado [27]. Google pone a disposición de los desarrolladores una librería para implementarlo. Se define un área en base a ciertas coordenadas, y cuando el usuario entra dentro de esta área, Google avisa a nuestra aplicación. Se confirma entonces la visita al centro comercial y se completa el reto.

Por otro lado, el cliente manifestó la necesidad de notificar al usuario cada vez que se completa un reto o campaña. Para ello se hace uso de Notificaciones Push [28].

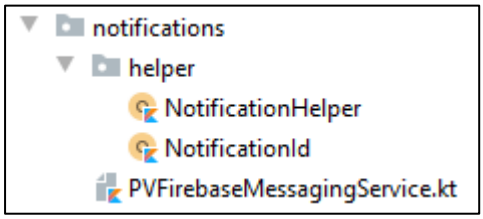
La implementación en código de estas funciones es la siguiente.




Para la implementación de la detención de Beacons se ha creado un servicio en background, el cual se ejecuta cada 15 segundos para garantizar su permanencia. Este servicio se encarga de detectar un Beacon válido y enviarlo a Back-End para ser asignado al reto o retos pertenecientes.




Para la implementación del geovallado se crea un servicio con similares características al usado en detención de Beacons. Se usa además una clase de ayuda para simplificar el código en el servicio y un receptor de emisión mediante el cual Google avisará a la aplicación cuando el usuario entre en el área.




Respecto a la implementación de notificaciones push, se parte de la librería de Google *Firebase Cloud Messaging* [29]. Esta librería exige crear un servicio con ciertas implementaciones para que las notificaciones push funcionen. Se usan dos clases de ayuda para simplificar el código del servicio y generar notificaciones más fácilmente.

# 7. Estudio económico

El estudio económico en este proyecto se va a categorizar según los recursos utilizados: Recursos humanos, recursos materiales y recursos digitales. Finalmente se muestra el coste total de su desarrollo.

Se debe tener en cuenta que los costes recogidos en este apartado corresponden únicamente al diseño y desarrollo de la parte Front-End para Android, de la cual trata este proyecto. Por políticas de privacidad de la empresa Hiberus Tecnologías Diferenciales, se ha realizado una estimación salarial aproximada teniendo en cuenta los datos recogidos en el sitio web StackOverflow, el cual realiza estudios periódicos sobre los salarios en este sector.

## 7.1. Costes en recursos humanos

En la siguiente tabla se recogen los costes de los recursos humanos involucrados en este proyecto de fin de grado. Aunque su autor es principalmente programador, durante el desarrollo ha realizado trabajos como analista y diseñador.

| | Precio por hora | Horas de trabajo | Total |
|---|---|---|---|
| **Analista** | 20 € | 40 h | 800 € |
| **Diseñador** | 14 € | 80 h | 1.120 € |
| **Programador** | 14 € | 520 h | 7.280 € |
| | | **640 h** | **9.200 €** |

*Tabla 1 - Costes en recursos humanos*

Es importante destacar que el coste de recursos humanos para una empresa aumenta en un 30% en concepto de cotización en la seguridad social y otras aportaciones ajenas al trabajador.

## 7.2. Costes en recursos materiales

En la siguiente tabla se recogen los costes de los recursos materiales involucrados en este proyecto de fin de grado. Dichos materiales han sido utilizados durante todo el análisis, diseño y programación de la aplicación Front-End.

| | Precio total | Amortización | Meses de uso | Coste por uso |
|---|---|---|---|---|
| **MacBook Pro 2020** | 2.129 € | 5 años | 4 meses | 142 € |
| **Xiaomi Mi A2** | 249 € | 2 años | 4 meses | 41,50 € |
| **Huawei P40 Lite E** | 179 € | 2 años | 1 mes | 7,50 € |
| | | | | **191 €** |

*Tabla 2 - Costes en recursos materiales*

### 7.3. Costes en recursos digitales

En la siguiente tabla se recogen los costes de los recursos digitales involucrados en este proyecto de fin de grado. Esto incluye los gastos en licencias de software, planes de alojamiento y herramientas en línea.

| | Precio por mes | Meses de uso | Total |
|---|---|---|---|
| **BitBucket** | 6 € | 4 meses | 24 € |
| **Microsoft Office** | 10,50 € | 4 meses | 42 € |
| **Servicio Hosting** | 16 € | 4 meses | 64 € |
| | | | **130 €** |

*Tabla 3 - Costes en recursos digitales*

### 7.4. Costes totales

En la siguiente tabla se recogen los costes de todos los recursos empleados, de forma unificada. Con ello llegamos al presupuesto inicial del proyecto.

| | Total |
|---|---|
| **Costes en recursos humanos** | 9.200 € |
| **Costes en recursos materiales** | 191 € |
| **Costes en recursos digitales** | 130 € |
| | **9.521 €** |

*Tabla 4 - Presupuesto inicial*

# 8. Resultados

El resultado final de este proyecto de fin de grado es la parte Front-End de una aplicación nativa para dispositivos Android con la que el cliente, Puerto Venecia, desea fidelizar a sus clientes.

A grandes rasgos se debe destacar el correcto cumplimiento de todos los objetivos del proyecto, lo que ha causado una gran satisfacción personal y del cliente, el cual ha visto cubiertas todas sus necesidades. Además, la toma inicial de decisiones en cuanto a tecnología, diseño y arquitectura de la aplicación ha sido correcta, y apenas ha sufrido modificaciones.

No obstante, durante el desarrollo del proyecto hubo algunos reajustes en su planificación, surgidos tras la necesidad de subsanar algunos problemas. Dichos reajustes se detallan a continuación.

## 8.1. Problemas encontrados

A pesar de cumplir con todos los objetivos del proyecto, se dieron ciertas complicaciones que obligaron a cambiar la planificación durante su desarrollo.

Las tecnologías han sido elegidas teniendo en cuenta los conocimientos previos del autor, pero la arquitectura de la aplicación y sobre todo el uso permanente del patrón MVVM [15] ha dificultado el desarrollo de algunas tareas, debido en gran parte a su desconocimiento. Las ganas de aprender siempre han estado presentes durante el desarrollo del proyecto, lo que sumado a la ayuda de los compañeros en la empresa consiguió subsanar este problema de manera rápida.

Otro de los problemas a los que hubo que enfrentarse es a la diversidad de dispositivos Android que existen actualmente, los cuales ofrecen diferentes adaptaciones del sistema operativo, diferentes resoluciones de pantalla, diferentes escalas… Esto estuvo presente durante el desarrollo de todas las pantallas de la aplicación, y fue necesario el uso de *dimensiones dinámicas* [30] para conseguir un resultado similar independientemente del tamaño y escala de la pantalla del dispositivo en el que se esté ejecutando.

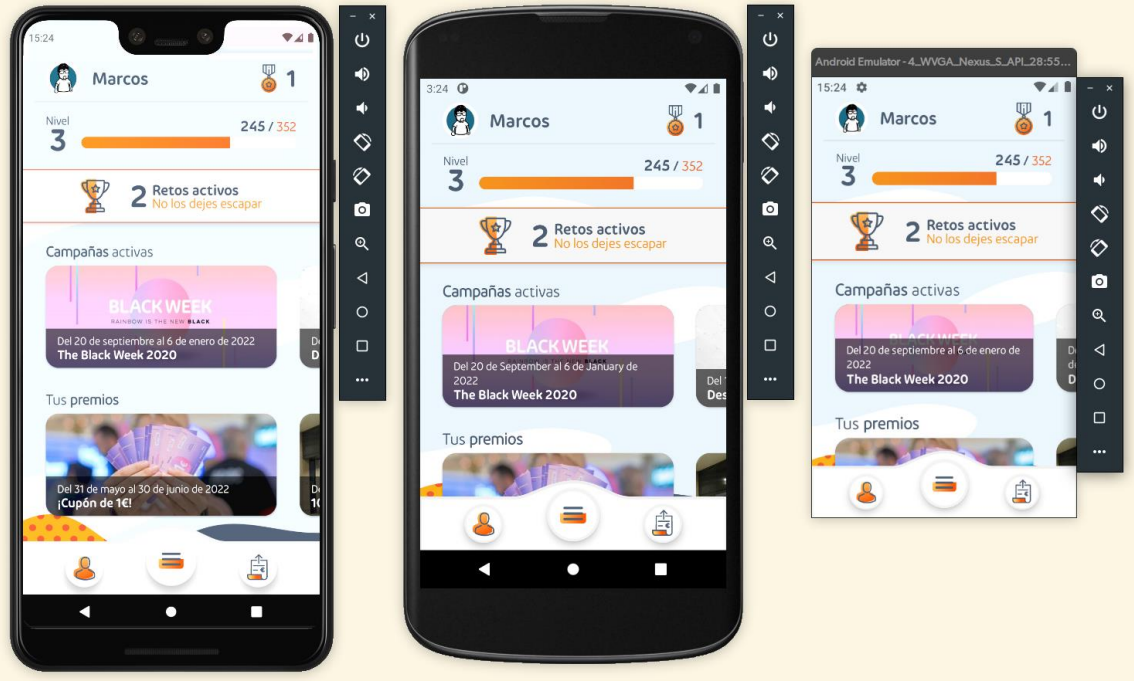



*Ilustración 42 - Aplicación funcionando en diferentes dispositivos*

Pero sin duda, el mayor problema en el desarrollo de este proyecto ha sido la integración con la API de Back-End. Esta API fue desarrollada con cierto desconocimiento sobre los requisitos que las aplicaciones debían cumplir, ya que para aquel entonces todavía no estaban del todo claros ni se había empezado con su desarrollo. Además, la arquitectura en esta parte tampoco estaba bien definida, por lo que fue necesario una remodelación completa mientras se estaba trabajando ya en la aplicación en Front-End.

Esto retrasó algunas tareas de desarrollo, y fueron necesarias varias reuniones con todo el equipo para cambiar su planificación.

## 8.2. Planificación final

Debido a los problemas anteriormente mencionados, y en especial a la integración con Back-End, la planificación de este proyecto tuvo importantes cambios con respecto a su inicio. Estos cambios fueron decisión del director del proyecto y de los propios desarrolladores.

Aunque la duración del proyecto no se excedió, la duración de algunas tareas sufrió ciertos cambios, y su desarrollo por tanto se acabó superponiendo. Fue necesario además realizar horas extra, consensuadas previamente con el autor, para poder completar a tiempo todas las tareas.

La planificación final llevada a cabo durante todo el desarrollo del proyecto se detalla a continuación.

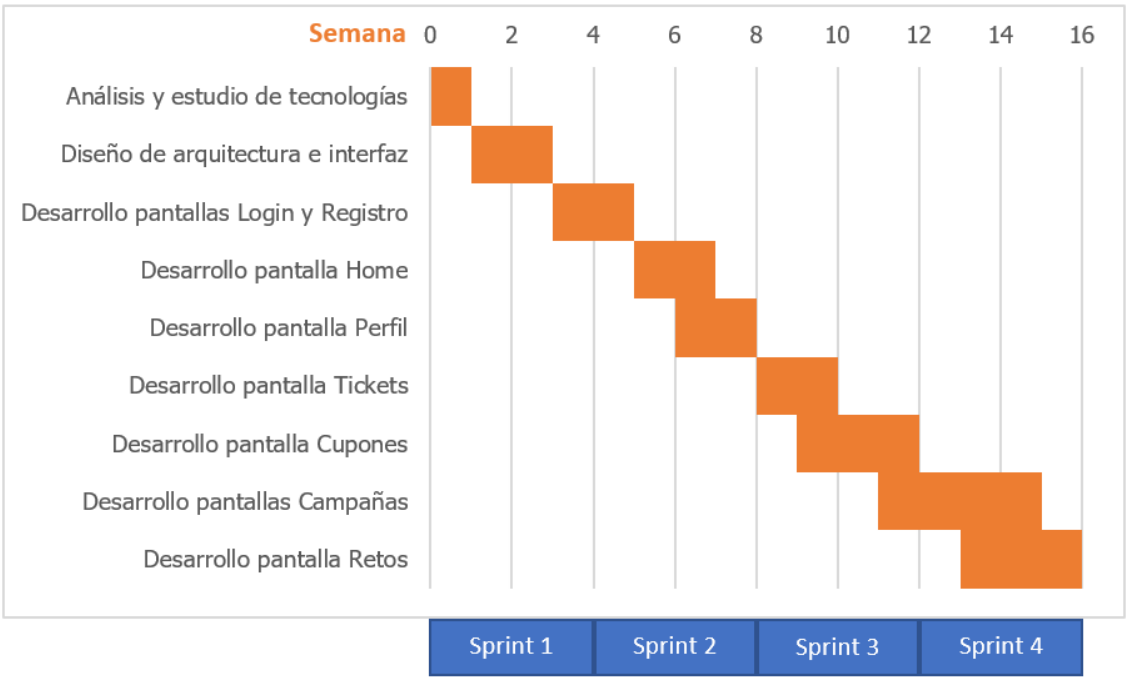



*Ilustración 43 - Diagrama de Gantt planificación final de tareas*

La distribución de tareas en cada sprint es la siguiente:

- **Sprint 1:** Se inicia en la semana 1, e incluye los trabajos de análisis y estudio de tecnologías a utilizar, el diseño de la arquitectura y la interfaz de la aplicación, y parte de las pantallas de Login y Registro.
- **Sprint 2:** Se inicia en la semana 5, y comprende el desarrollo e integración con la parte Back-End del resto de las pantallas de Login y Registro, así como las pantallas de Home y Perfil.
- **Sprint 3:** Se inicia en la semana 9, y comprende el desarrollo e integración con la parte Back-End de las pantallas de Tickets, Cupones y parte de las pantallas de Campañas.
- **Sprint 4:** Se inicia en la semana 13, y comprende el desarrollo e integración con la parte Back-End del resto de pantallas de Campañas y la pantalla de Retos.

Estos cambios durante el desarrollo del proyecto no han seguido una distribución lineal, sino que cada tarea se iba desarrollando conforme los cambios en la API de Back-End iban estando disponibles. Esto explica en gran parte el cambio en la distribución de tareas con respecto a la planificación inicial.

## 8.3. Presupuesto final

El presupuesto final es algo diferente al presupuestado inicialmente, debido a un aumento en las horas de desarrollo del proyecto, tal y como hemos visto anteriormente. Se necesitan 24 horas extra de desarrollo para completar todas las tareas a tiempo.

Los costes finales en recursos humanos se reflejan en la siguiente tabla.

| | Precio por hora | Horas de trabajo | Total |
|---|---|---|---|
| **Analista** | 20 € | 40 h | 800 € |
| **Diseñador** | 14 € | 80 h | 1.120 € |
| **Programador** | 14 € | 544 h | 7.616 € |
| | | **664 h** | **9.536 €** |

*Tabla 5 - Costes finales en recursos humanos*

En la siguiente tabla se recogen los costes de todos los recursos empleados, de forma unificada. Con ello llegamos al presupuesto final del proyecto.

| | Total |
|---|---|
| **Costes en recursos humanos** | 9.536 € |
| **Costes en recursos materiales** | 191 € |
| **Costes en recursos digitales** | 130 € |
| | **9.857 €** |

*Tabla 6 - Presupuesto final*

## 8.4. Producto final

El producto final es bastante similar al visto en el análisis del proyecto. No obstante, algunas pantallas han sufrido pequeñas variaciones debido a peticiones por parte del cliente. A continuación se van a explicar dichas variaciones, acompañadas de una captura de pantalla del resultado final. Las capturas de pantalla de la aplicación completa se encuentran en el Anexo V "Capturas de pantalla".

### *8.4.1. Pantalla de registro*

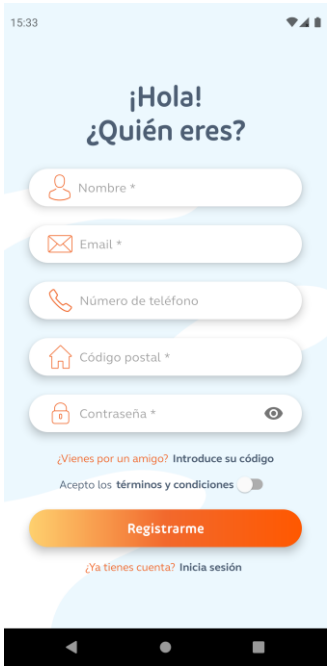



*Ilustración 44 - Pantalla final de Registro*

Tras la revisión por parte del cliente del desarrollo realizado en el sprint 1, el cual incluye esta pantalla, éste decide incluir dos nuevos campos en el registro: Número de teléfono y código postal.

Además, el cliente nos transmite la necesidad de indicar los campos del registro que es necesario rellenar, ya que el nuevo campo de código postal no es necesario para el registro.

### *8.4.2. Pantalla de perfil de usuario*


*Ilustración 45 – Pantalla final de Perfil de Usuario*

Tras la modificación de la pantalla de registro, se decide añadir los dos nuevos campos a la pantalla de perfil de usuario. Ambos campos son editables, manteniendo la no obligatoriedad del campo de código postal.

Durante el desarrollo de esta pantalla se decide además, bajo petición del cliente, deshabilitar la edición del campo de correo electrónico, siendo éste el único campo que no puede ser modificado.

### 8.4.3. Pantalla de detalles de ticket pendiente

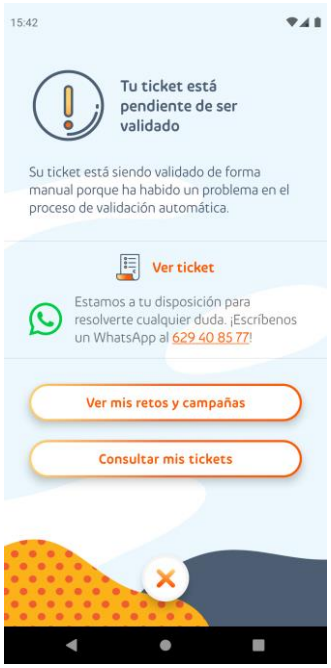



*Ilustración 46 - Pantalla final de detalles de Ticket Pendiente*

Tras la revisión del trabajo realizado en el sprint 3, donde se realiza la pantalla de detalles para un ticket subido, el cliente expresa la necesidad de incluir los tickets en estado "pendiente" en esta pantalla.

En un principio, cuando un ticket permanecía en estado "pendiente" no era posible visualizar sus detalles ya que sus datos, al no estar validados, podían ser incorrectos. El cliente decide mostrar una pantalla de detalles sencilla sin dar ningún dato relevante al ticket, más allá de la foto previamente adjuntada por el usuario.

### 8.4.4. Pantalla de listado de campañas


*Ilustración 47 – Pantalla final de listado de Campañas*

Durante el desarrollo de esta pantalla, y tras ciertas complicaciones en la parte Back-End de la aplicación, se decide cambiar el contenido mostrado en cada elemento del listado.

En un principio, cada elemento del listado mostraba el progreso que el usuario llevaba en esa campaña. Finalmente se decide mostrar únicamente el nombre de la campaña y el periodo en el que está activa para el usuario.

# 9. Conclusiones

Tras la elaboración de este proyecto de fin de grado puedo decir que he cumplido gratamente los objetivos que me había propuesto desde un principio. Además de haber extendido mis conocimientos sobre programación, he aumentado mi habilidad para analizar y tomar decisiones, siendo ésta la primera vez que me enfrento a un proyecto de tal envergadura.

A pesar de haber estudiado con anterioridad el manejo de la metodología ágil Scrum, nunca antes la había utilizado en un proyecto. Durante el desarrollo me he ido dando cuenta de lo importante que resulta apoyarse en una metodología ágil para organizar todas sus tareas y entregas, priorizando las más importantes y conociendo en todo momento la carga de trabajo restante.

Un punto clave en mi aprendizaje ha sido el uso de un patrón de arquitectura como es MVVM. Si bien es cierto que al principio me costaba entenderlo y aplicarlo correctamente, una vez conseguí captar toda su lógica me resultó muy cómodo de utilizar. Y es que el uso de un patrón de arquitectura acaba simplificando bastante la entrada de nuevas funcionalidades al proyecto, y facilita el traspaso de conocimiento al resto del equipo.

Pero sin duda lo que más me ha sorprendido durante el desarrollo de este proyecto ha sido la experiencia de trabajo en una empresa del sector como es Hiberus Tecnologías Diferenciales. Saber que todo el trabajo y constancia llevada a cabo durante todo este tiempo en la carrera ha dado sus frutos, demostrar mi conocimiento adquirido, conocer el entorno real de una empresa y, por supuesto, aprender de mis compañeros de trabajo ha sido una de las mejores experiencias de mi vida, y ha sido un punto decisivo en mi futuro.

## 9.1. Puntos de mejora

A pesar de ser consciente de los problemas que podría causar mi inexperiencia en algunos aspectos del desarrollo del proyecto, no contaba con que ese no fuera el principal problema. La falta de coordinación entre todos los miembros del equipo, especialmente en la parte Back-End, cambió bastante la planificación que se tenía en un inicio sobre el desarrollo de tareas en el proyecto.

Creo que este es un importante punto de inflexión para la empresa y más concretamente para los miembros del equipo, ya que una mayor implicación del gestor de proyecto puede ayudar a solventar este problema en futuros desarrollos.

Pero la falta de experiencia en algunas partes del desarrollo también afectó a la planificación inicial de tareas y a la calidad y optimización del código, el cual será necesario refactorizar más adelante.

## 9.2. Tareas futuras

A pesar de dar por finalizado este proyecto de fin de grado, el desarrollo de este proyecto todavía sigue su curso. Actualmente ya se ha acordado con el cliente la inclusión de nuevas funcionalidades a la aplicación, las cuales van a formar la "fase 2" del desarrollo. Algunas de estas funcionalidades son:

- Pantalla de **detalles para retos**: Actualmente, al pulsar en un reto muestra información genérica sobre ellos. Se quiere mostrar información específica para cada reto, así como la acción o acciones pertinentes para completarlo.
- Integración con **Google Analytics** [31]: Se desea conocer el comportamiento que los usuarios tienen en la aplicación, por lo que es necesario implementar esta herramienta de analítica.
- Integración con **Huawei Mobile Services** [32]: Tras la exclusión de los servicios de Google en dispositivos Huawei, la funcionalidad de notificaciones push ha dejado de funcionar para estos dispositivos. Para solventar este problema, y añadir soporte para analítica en dispositivos Huawei, se van a implementar sus propios servicios para ello.
- Pantalla de **contacto**: Actualmente existe una sección en el menú llamada "Ayuda", la cual abre en el navegador la página web de contacto del centro comercial. Se requiere crear una pantalla nativa con información directa de contacto (email, teléfono, dirección, WhatsApp…).
- Pantalla para **mapa 3D**: Se ha decidido cambiar el mapa en versión web que hay implementado por un mapa nativo en 3D. Este mapa se implementará bajo la herramienta "Mappedin".
- **Corrección de errores** de la "fase 1" del desarrollo.

# 10. Bibliografía

[1] Rosa Fernández. *Número de usuarios de smartphones en España entre 2015 y 2022* [en línea]. 4 octubre 2019. Disponible en: https://es.statista.com/estadisticas/493856/pronostico-de-usuarios-de-smartphone-en-espana.

[2] Cannes. *Puerto Venecia gana el Premio MAPIC 2013* [en línea]. Actualizado 18 noviembre 2013. Disponible en: https://www.l35.com/actualidad/noticias-l35/puerto-venecia-gana-el-premio-mapic-2013.html

[3] Android Developers. *Android Studio – Guia del usuario* [en línea]. Actualizado 1 mayo 2020. Disponible en: https://developer.android.com/studio/intro?hl=es-419

[4] Soywiz. *Sobre Kotlin* [en línea]. Actualizado 15 septiembre 2018. Disponible en: https://kotlin.es/sobre-kotlin/

[5] José Antonio Íñigo. *¿Qué es la Programación Reactiva? Una introducción* [en línea]. Actualizado 10 marzo 2017. Disponible en: https://profile.es/blog/que-es-la-programacion-reactiva-una-introduccion/

[6] Cecilio Álvarez Caules. ¿Qué es Gradle? [en línea]. Actualizado 1 abril 2015. Disponible en: https://www.arquitecturajava.com/que-es-gradle/

[7] Daniel Matesa. *Trello. Qué es, para qué sirve y cómo funciona* [en línea]. Actualizado 24 octubre 2020. Disponible en: https://www.expertosnegociosonline.com/que-es-trello-para-que-sirve/

[8] Varios autores. *Redmine* [en línea]. Actualizado 4 septiembre 2019. Disponible en: https://es.wikipedia.org/wiki/Redmine

[9] Varios autores. *Git* [en línea]. Actualizado 2 noviembre 2020. Disponible en: https://es.wikipedia.org/wiki/Git

[10] Atlassian. *BitBucket – Funcionalidades*. Actualizado 9 noviembre 2020. Disponible en: https://www.atlassian.com/es/software/bitbucket/features

[11] Alerta Hosting. *Que es Drupal y para que Sirve* [en línea]. Actualizado 1 abril 2019. Disponible en: https://www.alertahosting.com/que-es-drupal/

[12] Jesús Lucas. *Qué es NodeJS y para qué sirve* [en línea]. Actualizado 4 septiembre 2019. Disponible en: https://openwebinars.net/blog/que-es-nodejs/

[13] Ramón Abad. ¿Qué es Swagger? [en línea]. Actualizado 4 abril 2017. Disponible en: http://ramonabadypuntonet.org/2017/04/04/que-es-swagger/

[14] Retrofit página web oficial [en línea]. Disponible en: https://square.github.io/retrofit/

[15] Leomaris Reyes. *Aplicando el patrón de diseño MVVM* [en línea]. Actualizado 21 diciembre 2018. Disponible en: https://medium.com/@reyes.leomaris/aplicando-el-patr%C3%B3n-de-dise%C3%B1o-mvvm-d4156e51bbe5

[16] Android Developers. *View documentation* [en línea]. Actualizado 30 septiembre 2020. Disponible en: https://developer.android.com/reference/android/view/View

[17] Android Developers. *Declaring layout* [en línea]. Actualizado 13 agosto 2020. Disponible en: https://developer.android.com/guide/topics/ui/declaring-layout

[18] Android Developers. *Activity documentation* [en línea]. Actualizado 30 septiembre 2020. Disponible en: https://developer.android.com/reference/android/app/Activity

[19] Android Developers. *Fragment documentation* [en línea]. Actualizado 30 septiembre 2020. Disponible en: https://developer.android.com/reference/android/app/Fragment

[20] Android Developers. *Service documentation* [en línea]. Actualizado 30 septiembre 2020. Disponible en: https://developer.android.com/reference/android/app/Service

[21] Android Developers. *Intent documentation* [en línea]. Actualizado 30 septiembre 2020. Disponible en: https://developer.android.com/reference/android/content/Intent

[22] Android Developers. *Descripción general de las transmisiones* [en línea] Actualizado 27 diciembre 2019. Disponible en: https://developer.android.com/guide/components/broadcasts

[23] Android Developers. *Proveedores de contenido* [en línea]. Actualizado 27 diciembre 2019. Disponible en: https://developer.android.com/guide/topics/providers/content-providers

[24] Android Developers. *Descripción general del manifiesto de una app* [en línea]. Actualizado 28 octubre 2020. Disponible en: https://developer.android.com/guide/topics/manifest/manifest-intro

[25] Alberto Moral. *Architectural Pattern: Apple MVC's variation* [en línea]. Actualizado 1 abril 2016. Disponible en: http://moralalberto.github.io/2016/01/10/model-view-controller.html

[26] Raquel Ramo. *Beacons, ¿Qué son y cómo funcionan?* [en línea]. Actualizado 8 mayo 2017. Disponible en: https://trends.inycom.es/beacons-que-son-y-como-funcionan/

[27] Android Developers. *Crea y supervisa geovallas* [en línea]. Actualizado 12 agosto 2020. Disponible en: https://developer.android.com/training/location/geofencing

[28] Qode. *¿Qué son las Notificaciones Push?* [en línea]. Actualizado 4 febrero 2015 Disponible en: https://www.qode.pro/blog/que-son-las-notificaciones-push/

[29] Firebase. *Firebase Cloud Messaging – Documentación y guías* [en línea]. Disponible en: https://firebase.google.com/docs/cloud-messaging

[30] Android Developers. *Más tipos de recursos* [en línea]. Actualizado 27 diciembre 2019. Disponible en: https://developer.android.com/guide/topics/resources/more-resources?hl=es-419#Dimension

[31] Analytics

[32] Huawei

# Anexo I – Propuesta del proyecto

| | |
|---|---|
| **Nombre alumno:** | Marcos Angel Calvo García |
| **Titulación:** | Ingeniería Informática |
| **Curso académico:** | 2020-2021 |

## 1. TÍTULO DEL PROYECTO

Aplicación Android de fidelización para centro comercial Puerto Venecia.

## 2. DIRECTOR DEL PROYECTO

Este es un proyecto real de empresa, desarrollado en Hiberus Tecnología. Es por ello que añado un contacto dentro de la empresa, por si fuera necesario.

**Profesora:**

Violeta Monasterio Bazán

vmonasterio@usj.es

**Responsable en la empresa:**

Antonio Rentero Cano

arentero@hiberus.com

## 3. DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

Elaboración de aplicación móvil para dispositivos Android enfocada en la fidelización de clientes para el centro comercial Puerto Venecia. Se plantea hacer uso de la gamificación, con la que se pretende convertir cada visita al centro comercial en un juego, por el que el usuario puede conseguir multitud de recompensas.

Esta aplicación pretende además establecer una línea de contacto directo con el cliente, a través de la cual se le pueden hacer llegar distintas recompensas y notificaciones interactivas.

## 4. OBJETIVOS DEL PROYECTO

Los objetivos del proyecto son:

- Análisis de aplicación nativa para dispositivos móviles Android.
- Diseño de la aplicación.
- Desarrollo de la parte Front-End.
- Integración con la parte Back-End.

## 5. METODOLOGÍA

El desarrollo del proyecto estará marcado por el uso de la metodología Scrum[1], metodología ágil usada con bastante frecuencia en la empresa.

La fecha y duración de sprints[2] será fijada conforme se desarrolle el proyecto.

## 6. PLANIFICACIÓN DE TAREAS

La planificación de tareas será fijada en las fases iniciales del proyecto.

## 7. OBSERVACIONES ADICIONALES

N/A.

1 https://proyectosagiles.org/que-es-scrum/
2 https://openwebinars.net/blog/que-es-un-sprint-scrum/

# Anexo II – Reuniones

*Reunión 1 – Revisión de la propuesta*

| | |
|---|---|
| **Fecha** | 29/09/2020 |
| **Hora** | 17:30 a 17:45 |
| **Participantes** | Violeta Monasterio y Marcos Calvo |
| **Descripción** | Revisión de la propuesta de proyecto de fin de grado. |

*Reunión 2 – Revisión inicial de la memoria*

| | |
|---|---|
| **Fecha** | 23/11/2020 |
| **Hora** | 17:30 a 18:15 |
| **Participantes** | Violeta Monasterio y Marcos Calvo |
| **Descripción** | Revisión inicial de la memoria y planificación. |

*Reunión 3 – Revisión de la memoria*

| | |
|---|---|
| **Fecha** | 9/12/2020 |
| **Hora** | 17:30 a 18:00 |
| **Participantes** | Violeta Monasterio y Marcos Calvo |
| **Descripción** | Revisión de los tres primeros apartados. |

*Reunión 4 – Revisión de la memoria*

| | |
|---|---|
| **Fecha** | 21/12/2020 |
| **Hora** | 17:30 a 18:15 |
| **Participantes** | Violeta Monasterio y Marcos Calvo |
| **Descripción** | Revisión de los tres siguientes apartados. |

*Reunión 5 – Revisión final de la memoria*

| | |
|---|---|
| **Fecha** | 07/01/2021 |
| **Hora** | 17:30 a 18:15 |
| **Participantes** | Violeta Monasterio y Marcos Calvo |
| **Descripción** | Revisión de los últimos apartados y aclaraciones finales. |

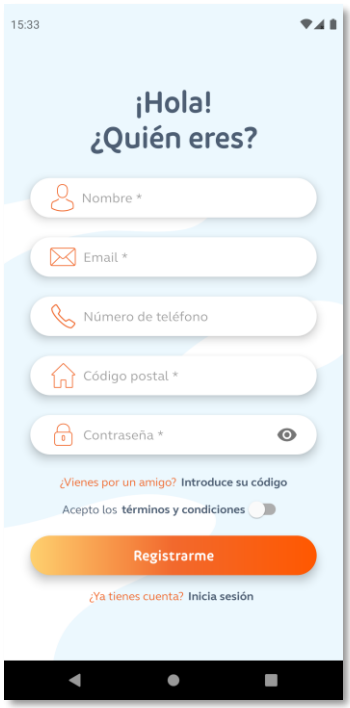
# Anexo III – Material adicional

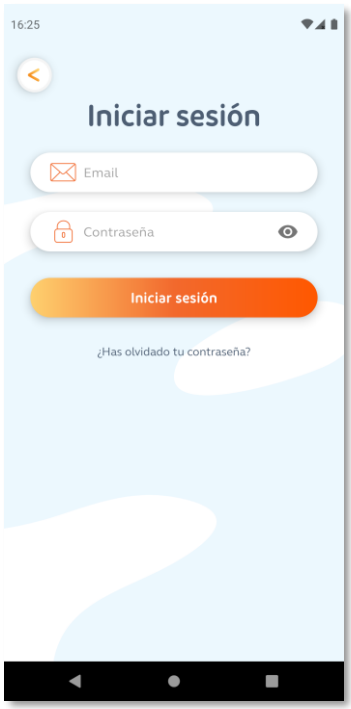
Como anexo a esta memoria se adjunta el código correspondiente a este proyecto. Se ha incluido una carpeta que incluye dos directorios. En el directorio "Versiones compiladas" se encuentran versiones compiladas de la aplicación en su versión de pruebas y en su versión en producción. Por otro lado, en el directorio "Código fuente" se encuentra el código Front-End de la aplicación a fecha 11/01/2021.

Se ha añadido un comentario en cada fichero de código explicando la implicación del autor en esa parte y añadiendo fuentes externas si las hubiera. Aunque el autor ha trabajado en todas las pantallas de la parte Front-End de la aplicación, en algunas lo ha hecho de manera indirecta, es decir, pantallas que ya tenían una base realizada pero que ha sido necesario modificar o completar. En el comentario de los ficheros de código, esta distinción se realiza mediante la frase "Creado por" o "Modificado por" respectivamente.

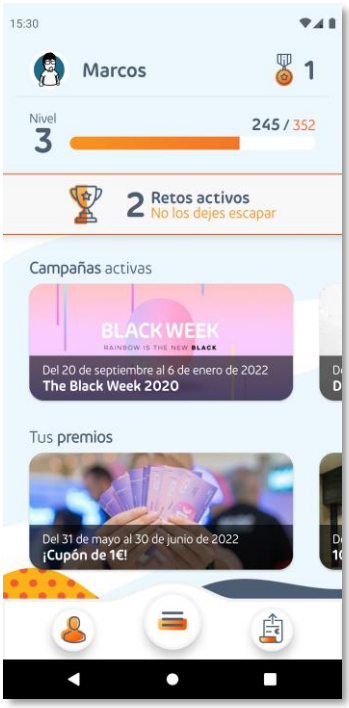
# Anexo IV – Distribución Trello

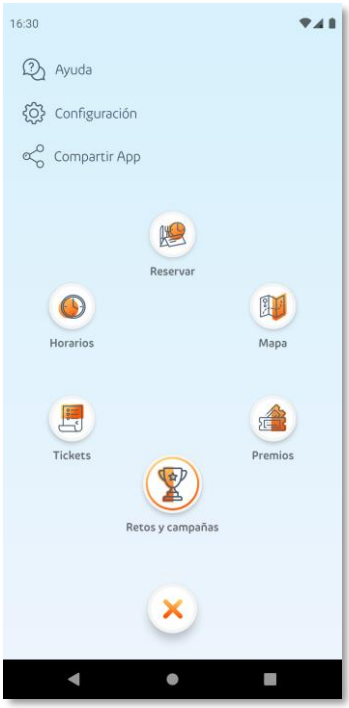
# Anexo V – Capturas de pantalla

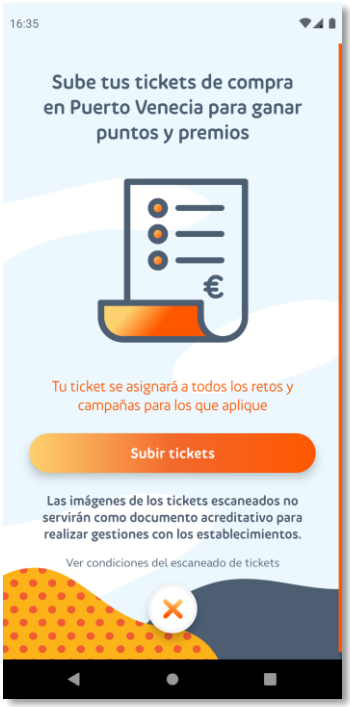


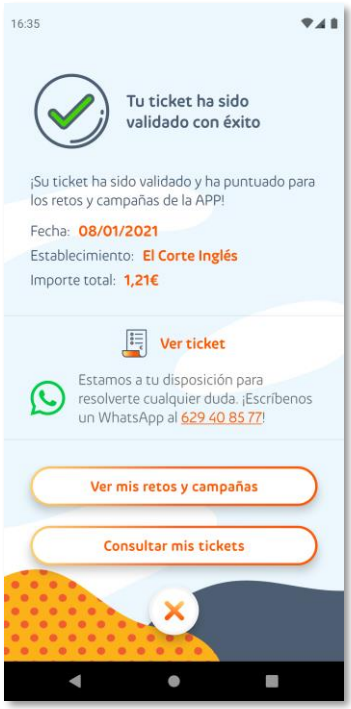




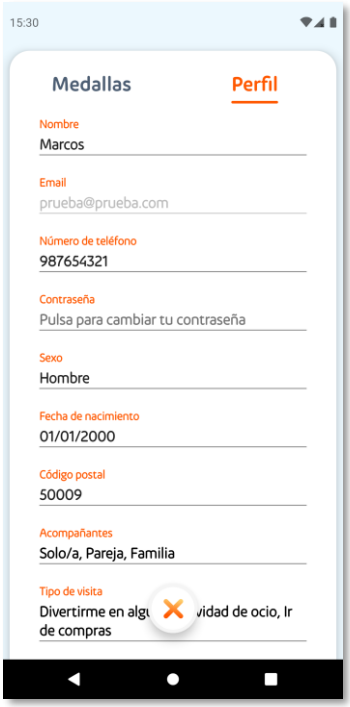



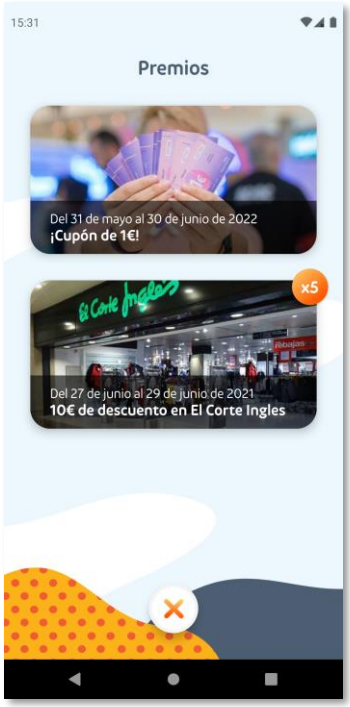


16:35
Sube tus tickets de compra en Puerto Venecia para ganar puntos y premios
Tu ticket se asignará a todos los retos y campañas para los que aplique
Subir tickets
Las imágenes de los tickets escaneados no servirán como documento acreditativo para realizar gestiones con los establecimientos.
Ver condiciones del escaneado de tickets


15:42
Tickets
06/11/2020 Pendiente
N/A 42,44€
27/10/2020 Denegado
UDON 18,63€
22/10/2020 Validado
PEPE JEANS 229,90€
22/10/2020 Validado
PEPE JEANS 229,90€
22/10/2020 Validado
PEPE JEANS 229,90€
22/10/2020 Pendiente
PEPE JEANS 229,90€
22/10/2020 Validado
PEPE JEANS
22/10/2020 Validado


16:35
Tu ticket ha sido validado con éxito
¡Su ticket ha sido validado y ha puntuado para los retos y campañas de la APP!
Fecha: 08/01/2021
Establecimiento: El Corte Inglés
Importe total: 1,21€
Ver ticket
Estamos a tu disposición para resolverte cualquier duda. ¡Escríbenos un WhatsApp al 629 40 85 77!
Ver mis retos y campañas
Consultar mis tickets


16:24
Marcos
Nivel 9
1846 / 1874
Invita a tus amigos
Medallas
Perfil
Medalla Desayuno
Medalla Cole


15:30
Medallas
Perfil
Nombre
Marcos
Email
prueba@prueba.com
Número de teléfono
987654321
Contraseña
Pulsa para cambiar tu contraseña
Sexo
Hombre
Fecha de nacimiento
01/01/2000
Código postal
50009
Acompañantes
Solo/a, Pareja, Familia
Tipo de visita
Divertirme en alg vidad de ocio, Ir de compras


15:31
Premios
Del 31 de mayo al 30 de junio de 2022
¡Cupón de 1€!
x5
Del 27 de junio al 29 de junio de 2021
10€ de descuento en El Corte Ingles


15:31
Retos
Campañas
BLACK WEEK
RAINBOW IS THE NEW BLACK
Del 20 de septiembre al 6 de enero de 2022
The Black Week 2020
Del 16 de septiembre al 20 de mayo de 2021
Desayunos y Shopping
Del 17 de agosto al 15 de septiembre de 2021


15:33
Del 16 de septiembre al 20 de mayo de 2021
Desayunos y Shopping
1 reto de 1 completado
Cuando vengas a desayunar a Puerto Venecia... ¡No olvides hacerte una foto y subirla con el hastag! #DesayunandoenPV Directamente entrarás en el sorteo para que ¡todas tus compras te salgan gratis!
Ven a Puerto Venecia
1/1


15:33
Ven a Puerto Venecia
1/1
Premios
XP
+250 puntos
Medalla Desayuno
20€ de Descuento
Tarjeta regalo de 10€
10% de Descuento
Reclamar premios


16:45
Retos
Campañas
¡Enhorabuena! ¡Estos son sus premios!
XP
+100 puntos
10€ de descuento en El Corte Ingles
Cupón de prueba


16:42
10€ de descuento en El Corte Ingles
Del 27 de junio al 29 de junio de 2021
10€ de descuento en El Corte Ingles


16:46
Saltar
Participa en campañas
Descubre los retos y las campañas
Participa en nuestros divertidos retos y campañas. ¡Podrás encontrar increíbles sorpresas y regalos!
Siguiente