Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
(Mención en Ingeniería de Software)

Route66App

Aplicación móvil de gamificación empresarial para conseguir fidelización de cliente (interfaz de cliente)

Autor:
D. Álvaro Carreras Regorigo
Tutora:
Dña. Margarita Gonzalo Tasis

Resumen

Cada vez es más común encontrarse la ludificación aplicada a los procesos empresariales como una forma de incrementar la productividad sobre un determinado aspecto. Una de las aplicaciones más comunes es la fidelización de clientes, en donde, por medio de una serie de juegos y actividades, se incita a aumentar el consumo de los usuarios sobre la marca y esta es justa la idea fundamental sobre la que se sustenta este Trabajo de Fin de Grado.

Route66App es una aplicación nativa para dispositivos *Android*, orientada a aplicar técnicas de ludificación empresarial de fidelización de clientes para una cadena de restaurantes de comida rápida. Se ha realizado con las últimas tecnologías propuestas por *Google* para el desarrollo de aplicaciones móviles: *Firebase* y *Kotlin*.

Abstract

It is more and more common to find the gamification applied to the different business procedures as a way of increasing the productivity over a certain business aspect. One of the most common applications is the earning of customers loyalty where, thanks to different games and activities, customers are encouraged to increase the expense in the products of the brand and that is exactly the main idea of this Bachelor's Thesis.

Route66App is a native Android application oriented to apply the business gamification techniques of earning the customers loyalty for a quick-meal restaurant chain. It has been made using the latest technologies proposed by Google for the mobile app development: Firebase and Kotlin.

Agradecimientos

Quisiera agradecer, en primer lugar, a mi tutora del Trabajo de Fin de Grado, Dña. Margarita Gonzalo Tasis, por su dedicación e ilusión puesta en este proyecto.

En segundo lugar, a mi familia y amigos, por el apoyo que me han dado en estos cuatro años de carrera y en especial a mis padres por haberme ayudado en este trabajo con sus ideas y opiniones.

Por último, a mis compañeros de equipo de Telefónica Investigación y Desarrollo, que me han dado la oportunidad de aprender tecnologías muy utilizadas y relacionadas con el desarrollo de aplicaciones Android.

Gracias a todos.

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y contexto

1.1. Introducción y objetivos

1.1.1. Introducción

La ludificación consiste, según IEBSchool [29] en *"el uso de mecánicas de juego en un contexto de no juego para conducir el comportamiento de los participantes (mediante la participación, la interacción, la adicción o, incluso, la competición) hacia la consecución de un determinado objetivo de negocio"*.

Aunque habitualmente se ha enfocado la ludificación al área comercial o de ventas de cara a los clientes, (como por ejemplo el [35] *Monopoly Ganador de McDonalds*), existen también ejemplos en la educación, en las relaciones con los proveedores e, incluso, para incrementar el número de ventas en un centro comercial, (ludificación orientada a los empleados), tal como se recoge en el Trabajo de Fin de Grado de la alumna del Grado en Ingeniería Industrial, [56] Dña. Ana Ruiz Caballero.

Hoy en día, miles de compañías utilizan la ludificación en sus procesos empresariales desde sus relaciones con los empleados hasta con los clientes, pasando por sus comerciales. La ludificación realmente funciona y, por increíble que parezca, es muy efectiva. De hecho, no son pocos los ejemplos de una aplicación más que satisfactoria de la misma. Así, en 2011, [1] Volkswagen decidió inventar en China, su mercado más importante, una nueva versión de su *"people's car"* y para ello contó con la ayuda de sus clientes, a quienes ofreció una herramienta de diseño y un sistema de puntuaciones. El resultado de este proyecto fue la obtención de más de 50.000 propuestas diferentes. Otro caso que merece la pena reseñar es el de Correos [30], que en 2012 rediseñó su *web* y se planteó o bien contratar a una empresa por miles de euros, bien, plantear un sistema de ludificación en el que los empleados propusieran nuevos diseños a cambio de pequeños regalos. Esta segunda opción fue la elegida (se presentaron más de 50.000 ideas en un tiempo récord), y esto supuso un ahorro de un 70 % frente a la primera alternativa.

El caso más destacable en nuestro país es el de [38] *BBVA Game*, una plataforma de ludificación asociada a la banca virtual. En ella, los usuarios consiguen puntos al superar distintos retos como consultar movimientos, domiciliar la nómina, realizar transferencias, etcétera. Igualmente, también se pueden conseguir incentivos por compartir un mensaje en redes sociales, invitar a amigos o contratar productos, que es posible canjear por premios directos o por participaciones en los sorteos que realizan. El resultado de este proyecto fue un éxito rotundo, ya que consiguieron 100.000 nuevos clientes en los nueve primeros meses y aumentaron el uso de la plataforma por parte de sus clientes ya existentes.

1.1.2. Objetivos

El propósito de este Trabajo Fin de Grado es el de realizar una aplicación *Android* basada en un sistema de ludificación para un restaurante de comida rápida, partiendo de la idea desarrollada en el [55] Trabajo de Fin de Grado de Dña. Cristina Martínez Martínez. Los objetivos concretos de la aplicación vendrán detallados en el documento de análisis.

Este Trabajo Fin de Grado detallará el desarrollo, análisis, diseño e implementación de la parte cliente del sistema informático en cuestión y se ha diseñado para que esta cambie las diferentes misiones y desafíos de acuerdo con los parámetros que el administrador del sistema establezca en el *backend* (parte de gestión).

1.1.3. Motivación

La motivación es doble, por un lado, analizar, diseñar e implementar un sistema de ludificación, diseñado por una antigua alumna de esta Universidad para completar así una de las líneas futuras propuestas por la misma en su Trabajo de Fin de Grado, y por otro, se busca desarrollar un sistema de ludificación que permita a los alumnos del Grado en Ingeniería de Organización Industrial de nuestra Universidad poner en marcha y probar un sistema de este tipo, que no tiene alternativas de licencia de prueba para estudiantes en el mercado.

1.2. Análisis de aplicaciones similares

Se han analizado diversas aplicaciones de restaurantes de comida rápida presentes en España, con el fin de conocer cómo son y si han implementado técnicas de ludificación a las mismas. Como se verá más adelante, los sistemas de ludificación aplicados en estos ejemplos son bastante básicos y no permiten extraer todo el potencial que podrían tener.

1.2.1. Análisis aplicaciones de restaurantes de comida rápida.

1.2.1.1. *McDonald's*

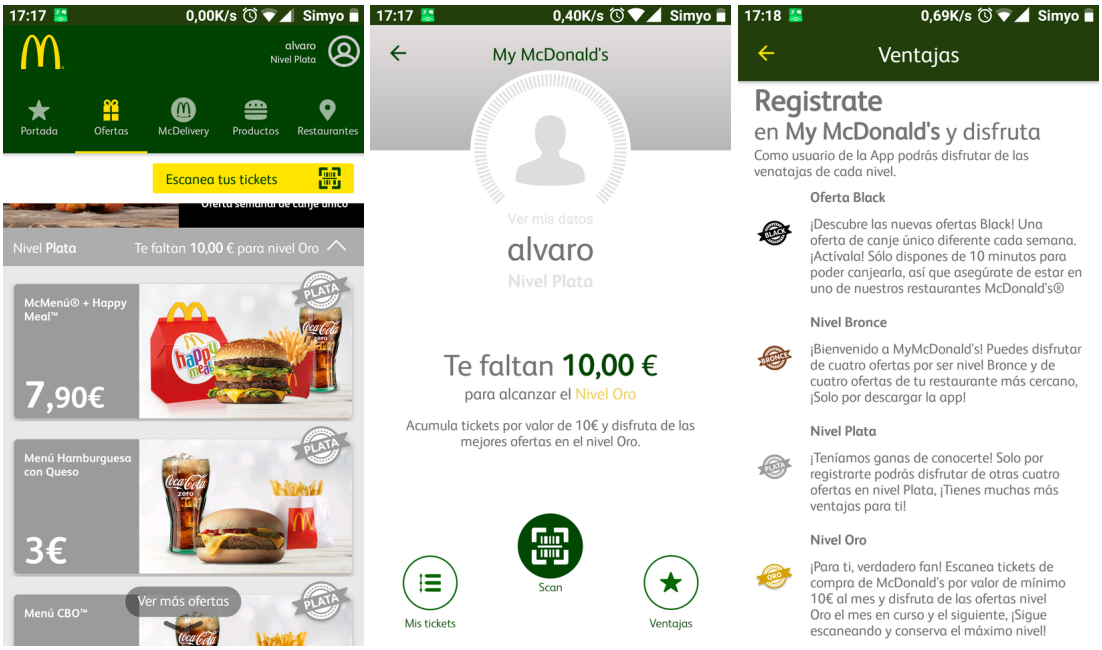


Figura 1.1: Capturas de pantalla de la aplicación *"McDonald's España - Ofertas"*. [34]

- **Aplicación estudiada:** [34] *McDonald's España - Ofertas.*
- **Análisis:** La aplicación móvil *McDonalds* ofrece un gran número de funcionalidades entre las que se encuentran un servicio de ofertas, canjeo de cupones, listado de la carta de productos o envío de pedidos a domicilio.

  El sistema de ludificación implementado es bastante básico ya que su estrategia se basa en utilizar un sistema por niveles (oro, plata y bronce), en el que cuanto mayor sea el nivel del usuario, mejores ofertas se pueden encontrar. Para poder subir de nivel, será necesario escanear los *tickets* de compra y llegar a un mínimo de gasto. Durante el tiempo que la he tenido instalada, me ha pedido dar la opinión de la *app* por correo electrónico a cambio de obtener un obsequio.
- **Puntos fuertes:**
  - Promueve el consumo de los clientes en los restaurantes de la cadena, recomendándoles hacer un determinado gasto para poder mantenerse en el mismo nivel.
  - Ofertas *"Black"*: Son ofertas que se activan solo durante diez minutos y hay que estar presente en el local en ese momento para canjearlas.

- **Puntos débiles:**
  - Se limita al consumo y no a promocionar la marca por redes sociales ayudándose de los clientes.
  - Las promociones solo se pueden hacer efectivas en los establecimientos físicos, no en pedidos por Internet.
  - No existe ningún historial de pedidos.

1.2.1.2. Burger King

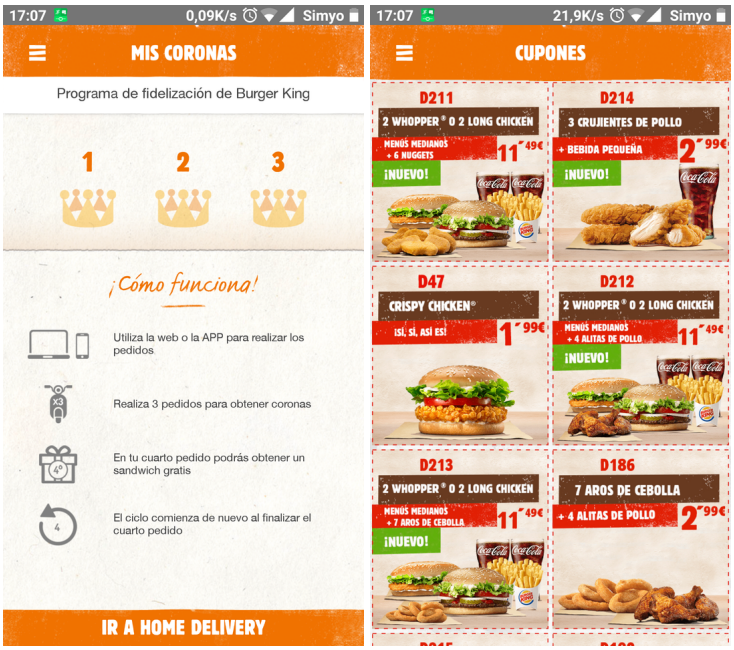


Figura 1.2: Capturas de pantalla de la aplicación *"BURGER KING® España"*. [5]

- **Aplicación estudiada:** [5] *BURGER KING® España.*
- **Análisis:** Al igual que McDonalds, ofrece un sistema de fidelización llamado *"Mis Coronas"*, consistente en que cada tres pedidos realizados por medio de la *app* móvil o el sitio *web* de *Burger King*, se acumulan coronas y cada cuatro se consigue una hamburguesa gratis. Además, invitan a registrarse en la *app* y con el historial de pedidos efectúan ofertas personalizadas al cliente.

  Las opciones de esta *app* están bastante limitadas: conseguir un listado de restaurantes, realizar pedidos a domicilio, ver la carta de productos o visualizar un listado de cupones o promociones.
- **Puntos fuertes:**
  - Avisa mediante notificaciones *PUSH* de nuevas ofertas.
  - Solicita seguir a la marca por redes sociales (*Facebook*).
  - Tiene un sistema de fidelización para sus pedidos *online*.
  - Permite visualizar el historial de pedidos por Internet.

- **Puntos débiles:**
  - La parte de fidelización está solo dedicada a la realización de pedidos por Internet.
  - Las promociones son muy similares, independientemente del perfil del cliente.
  - No está orientada a que el cliente realice un determinado consumo.
  - No se pueden aplicar los cupones de la aplicación a pedidos a domicilio.

1.2.1.3. KFC

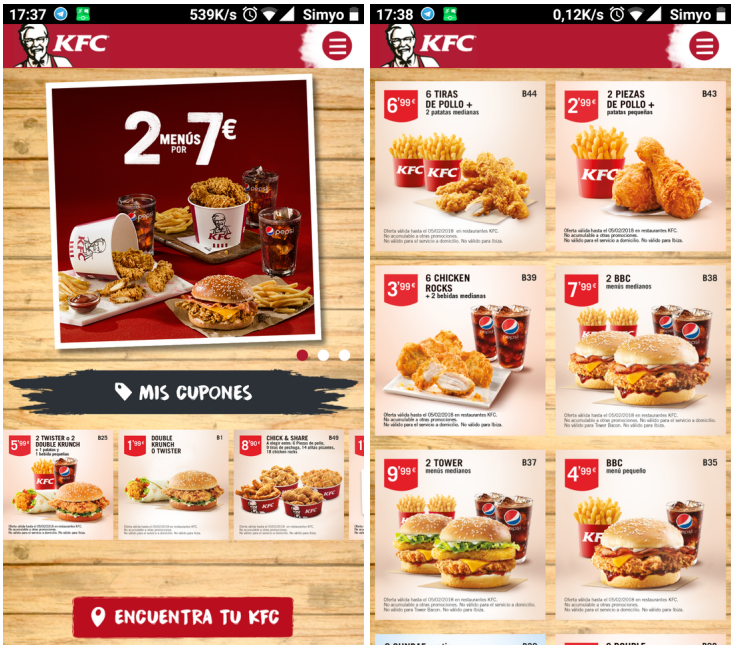


Figura 1.3: Capturas de pantalla de la aplicación *"KFC España–ofertas cerca de ti"*. [26]

- **Aplicación estudiada:** [26] *KFC España–ofertas cerca de ti.*
- **Análisis:**

  Es una de las aplicaciones más simples de las estudiadas pues se limita a poner una carta de productos, un listado de restaurantes y una opción para obtener ofertas.
- **Puntos fuertes:**
  - Notificaciones *PUSH* para alertar al cliente de nuevas ofertas y promociones.
- **Puntos débiles:**
  - No tiene un sistema de fidelización.

1.2.1.4. VIPS

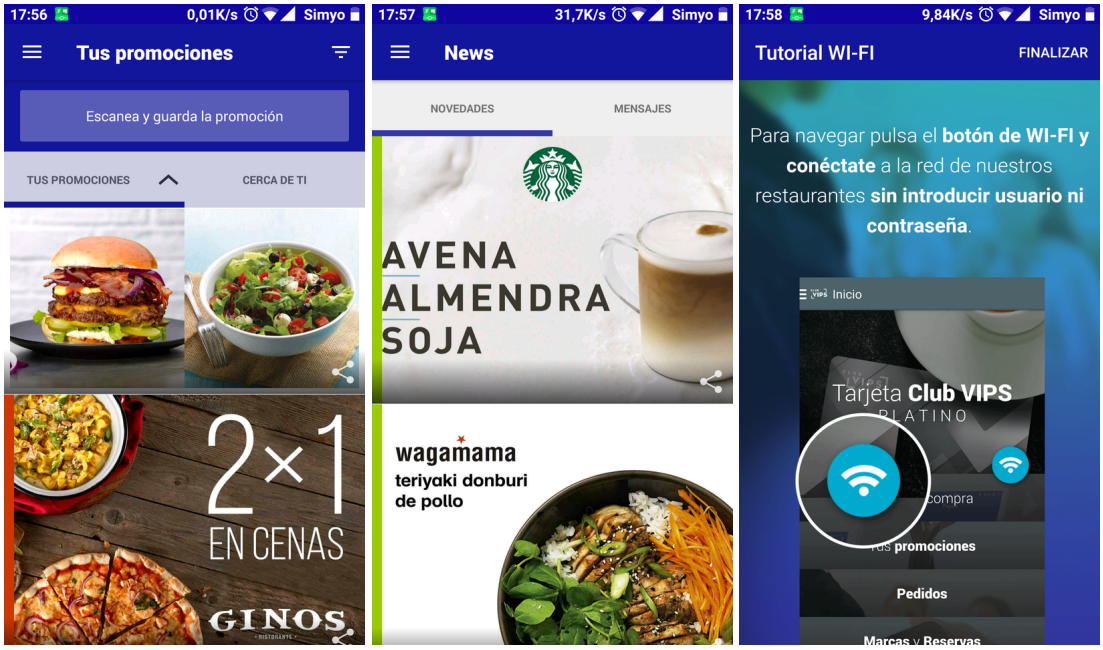


Figura 1.4: Capturas de pantalla de la aplicación *"Club VIPS: Promociones y pedidos Take Away"*. [46]

- **Aplicación estudiada:** [46] *Club VIPS: Promociones y pedidos Take Away.*
- **Análisis:** La aplicación de la cadena *VIPS* es, sin lugar a dudas, la más completa de las analizadas.

  Para fidelizar clientes utiliza el servicio *ClubEuroVIPS*, en el que ofrece un *cashback* de un 3 %, pudiendo duplicarse bajo ciertas condiciones como aumentar el consumo o hacerlo en determinadas franjas horarias. Además ofrece un sistema de clasificación de usuarios por niveles de gasto (niveles clásico, oro y platino) a partir de los cuales podremos obtener *WiFi Premium*, bebidas extras o puntos extras.

  Paralelamente, la aplicación invita al usuario a sus lanzamientos o estrenos de nuevos productos. Por ejemplo "*te invitamos a nuestras hamburguesas del Chef o a nuestras nuevas pizzas Chicago Style*" o "*te invitamos a probar nuestras pizzas*" y se pueden encontrar funcionalidades propias de un restaurante que apuesta por el crecimiento de su app móvil, como son la posibilidad de pagar desde la app, la funcionalidad *Shake-It*, que consiste en realizar su pedido favorito con solo agitar su teléfono móvil, o la opción de guardar promociones en la aplicación. Además de permitir realizar pedidos online del tipo *take-away*.
- **Puntos fuertes:**
  - La aplicación integra la tarjeta *Club VIPS*, que permite sumar puntos gracias a las compras hechas en restaurantes de la marca.
  - Se puede duplicar el valor de los puntos del *Club VIPS* bajo condiciones específicas.
  - Notificaciones *PUSH* para informar al cliente de nuevas ofertas y promociones.
  - Historial de pedidos.
  - Integra un sistema para conectarse automáticamente con las redes *WiFi* de los restaurantes de la marca. Posibilidad de tener *WiFi* Premium en caso de alcanzar un cierto consumo.
  - Permite capturar nuevas promociones escaneando el código *QR* de la publicidad.

- Invitaciones a eventos por el hecho de ser usuario de un determinado nivel.
- Clasificación del usuario por niveles basándose en el consumo que se hace en los restaurantes.
- Se realiza *cashback* en los pedidos realizados siendo socio del *Club VIPS*.
- Se puede conseguir hasta un 25 % de descuento al invitar a usuarios a hacerse socios del *Club VIPS*.

- **Puntos débiles:**
  - No existe ningún juego ni interacción con redes sociales.

1.2.1.5. *Pans & Company*

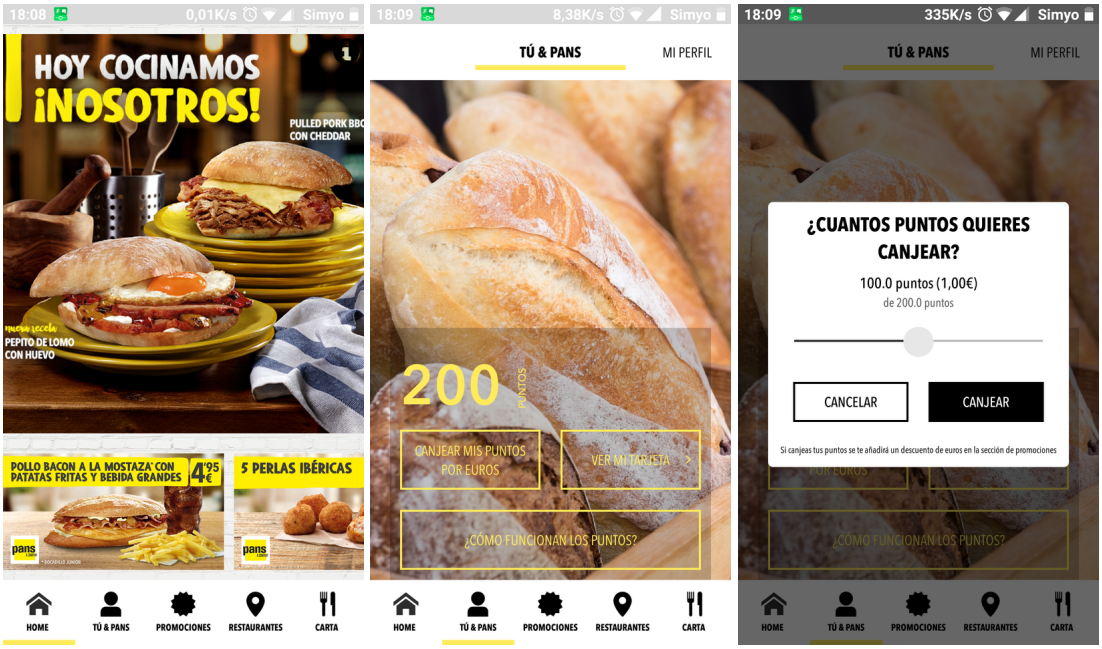


Figura 1.5: Capturas de pantalla de la aplicación *"Pans & Company"*.
[39]

- **Aplicación estudiada:** [39] *Pans & Company.*
- **Análisis:** *Pans & Company* ofrece únicamente una tarjeta de fidelidad integrada en la misma *app*. Esta tarjeta básicamente consiste en un código *QR* que el usuario puede mostrar en caja para acumular puntos que, posteriormente, se podrán convertir en dinero. Además, se realizarán ofertas personalizadas.

  Sus funcionalidades son las típicas de una aplicación de esta categoría: listado de restaurantes, ofertas y listado de la carta.

- **Puntos fuertes:**
  - Notificaciones *PUSH* para alertar al cliente de nuevas ofertas y promociones.
  - La aplicación integra una tarjeta de fidelización en el que se acumulan puntos a la hora de realizar compras. Los puntos se podrán canjear posteriormente por descuentos en compras en los restaurantes.

- **Puntos débiles:**
  - No cuenta con interacción entre usuarios ni redes sociales, se limita a la tarjeta de fidelización.

1.2.1.6. *Foster's Hollywood*

Figura 1.6: Capturas de pantalla de la aplicación *"Foster's Hollywood"*. [21]

- **Aplicación estudiada:** [21] *Foster's Hollywood.*
- **Análisis:** El sistema de fidelización de clientes de *Foster's Hollywood* se basa en un sistema de niveles (*blue*, *silver* y *gold*), en el que cuanto mejor sea el nivel, mayores ventajas tendrá el usuario: postre de regalo, descuentos de ocio (en empresas asociadas), o incluso, promoción por el cumpleaños. Para poder aumentar de nivel será necesario ir un número de veces a los establecimientos de la cadena.

  Se nos identificará como clientes mediante la tarjeta *Foster's Hollywood*, un código *QR* localizado en la aplicación y que el cliente simplemente debe enseñar al realizar el pago, con el fin de que la compra quede asociada a su usuario. Por otra parte, también es posible visualizar la carta del restaurante o ver el listado de restaurantes, así como realizar el pedido *online*.
- **Puntos fuertes:**
  - Notificaciones *PUSH* para informar al cliente de nuevas ofertas y promociones.
  - Sistema de fidelización basado en niveles e integrado en la aplicación.
- **Puntos débiles:**
  - No cuenta con interacción entre usuarios ni redes sociales, se limita al sistema de fidelización.

1.2.2. Conclusiones extraídas

Del análisis de las anteriores aplicaciones, pueden extraerse las siguientes conclusiones:

- Ninguna emplea un sistema de ludificación avanzado tal y como se propone en este trabajo.
- Los sistemas de fidelización de clientes son bastante limitados siendo, en muchos casos, una simple tarjeta virtual donde acumular puntos.

- Todas ofrecen un sistema para obtener ofertas.
- Las aplicaciones se limitan a la interacción Usuario-Empresa, ninguna emplea las redes sociales para aumentar su base de usuarios.
- La mayoría de las aplicaciones (todas exceptuando Burger King y KFC) promueven que el usuario vuelva al local, mediante la realización de ofertas específicas para ello. Irían dirigidas a un perfil de jugador de tipo *triunfador*, de acuerdo con la teoría de Bartle [9].

1.3. Estructura de la memoria

Este trabajo está divido en cuatro partes y, a su vez, en secciones y subsecciones, abarcando así los distintos documentos recogidos en esta memoria:

1. Capítulo I. Introducción y Objetivos: incluye una breve descripción de lo que es la ludificación, así como un resumen de la propuesta realizada por la diseñadora del sistema de ludificación sobre la que está basado el proyecto. Finalmente, también contiene un análisis de aplicaciones de restaurantes de comida rápida que operan en España, con la finalidad de estudiar si estos incluyen o no un sistema de ludificación.
2. Capítulo II. Herramientas utilizadas: justificación del empleo de alguna de ellas.
3. Capítulo III. Route66App: es la parte más importante del proyecto porque incluye la gran mayoría de los documentos, como el Plan de Desarrollo de Software, el documento de análisis, el documento de diseño y todos aquellos detalles de la implementación que merecen la pena reseñar.
4. Capítulo IV. Conclusiones y líneas futuras.
5. Webgrafía y bibliografía.
6. Anexos.

1.4. Propuesta

Este Trabajo de Fin de Grado (T.F.G) estará centrado en el T.F.G. [55] "*Estudio de ludificación en una empresa para mejorar la fidelización de los clientes*", elaborado por Dª Cristina Martínez Martínez, graduada durante el curso 2016-2017 en Ingeniería de Organización Industrial por la Universidad de Valladolid.

El trabajo de esta alumna, centrado en el diseño de un sistema de ludificación aplicado a un posible caso real,(restaurante de comida americana), tenía el problema de la imposibilidad de llevarlo a cabo con herramientas profesionales, dado su coste y a la falta de disponibilidad de las mismas. Por este motivo, el T.F.G. se tuvo que defender contando solo con una serie de prototipos basados en un sistema web.

Como la ludificación es un área en investigacion que en experiencias anteriores ha obtenido muy buenos resultados, la tutora del T.F.G., Dña. Margarita Gonzalo Tasis, propuso la realización de este proyecto. Por otro lado, el resultado del mismo puede servir a alumnos de otras facultades de nuestra Universidad para poder utilizar una herramienta de ludificación real.

1.4.1. Resumen de “Estudio de ludificación en una empresa para mejorar la fidelización de los clientes”

Se propone la realización de un sistema de ludificación para una cadena de restaurantes de comida americana, que cuenta con un conjunto de franquicias repartidas por todo el territorio español. Es una empresa con muy buenos resultados económicos y que está respaldada por una amplia y sólida base de clientes para quienes decide elaborar una aplicación que mejore la opinión de los mismos sobre la marca, ya que han llegado críticas sobre el sistema vigente de cupones y descuentos.

Finalmente, el equipo directivo decide encargar una aplicación basada en la ludificación cuyos objetivos son los siguientes:

1. Aumentar la fidelización de los clientes.
2. Incentivar las ventas.
3. Mejorar la imagen de la marca en redes sociales.
4. Conseguir nuevos clientes.
5. Mejorar la confianza y satisfacción del cliente.

Se medirá el cumplimiento de los objetivos anteriores mediante el cálculo de una serie de ratios.

Público objetivo

El público objetivo estará formado por hombres y mujeres con edades comprendidas entre los doce y los cincuenta y cinco años.

Los conocimientos tecnológicos esperados en este grupo de edad no tienen por qué ser altos, considerándose suficientes, conocimientos básicos en uso de *smartphones* y de aplicaciones móviles. En el caso en el que nos encontramos, la aplicación que se va a desarrollar estará enfocada a ser ejecutada sobre terminales de tipo Android.

Los potenciales usuarios de la aplicación la utilizarán para conseguir recompensas y descuentos derivados del uso de la misma.

Por último, con el fin de fidelizar a un mayor número de clientes, y de acuerdo con la [9] teoría de Bartle [9], se realizarán actividades que permitan satisfacer las necesidades de los cuatro tipos de jugador.

Roles

Se proponen dos tipos de roles: Administrador y Cliente.

Este trabajo se centrará en el rol cliente. El administrador tendrá su protagonismo en la parte de gestión, que no está incluida en este proyecto.

Descripción del juego

El juego se escenifica en la Ruta 66 americana que alude así la temática del restaurante. Se dividirá el camino en once etapas o niveles correspondientes a las principales ciudades por las que pasa esta vía y, para avanzar, el cliente deberá completar una serie de actividades, que además, otorgarán insignias, logros y premios. Cabe destacar que la dificultad de los niveles irá aumentando gradualmente, con el objetivo de no perder la motivación.

1. Chicago.	4. Tulsa.	7. Santa Fe.	10. Williams.
2. Springfield.	5. Oklahoma City.	8. Albuquerque.	
3. St. Louis.	6. Amarillo.	9. Flagstaff.	11. Los Ángeles.

Cuadro 1.1: Ciudades o niveles de la aplicación.

Todos los niveles excepto el de Chicago tendrán un planteamiento similar:

- Nivel 1. Chicago: Compuesto por las siguientes actividades:
  - Configuración del perfil del jugador.
  - Crear un equipo.
  - Seguir en redes sociales la página del restaurante.
  - Puntuar la aplicación en la tienda de aplicaciones.
  - Comentar la aplicación en la tienda de aplicaciones.
- Resto de niveles: Consistirán en diez misiones:
  - Misiones de nivel social: consistirán en tareas como compartir el progreso, subir una foto hecha en el restaurante a las redes sociales, enviar invitaciones, etcétera.
  - Misiones de minijuegos: serán dos casillas y consistirá en jugar a un minijuego de un perfil de jugador.
  - Misiones de consumo: el jugador deberá realizar consumiciones en los restaurantes de la cadena. Avanzará más o menos casillas en función del gasto que realice.
  - Misiones de retos especiales: en días señalados, los jugadores podrán participar en retos creados específicamente para ese día.

Recompensas

Hay tres tipos de recompensa:

- Puntos: el usuario obtendrá diez puntos por cada casilla que avance. Y la suma total se agrupará en tres categorías: consumo, social y competitivo.
- Insignias: cuando el jugador haya completado un nivel obtendrá una insignia (matrícula de la ciudad asociada). Habrá otras insignias, como la familiar o las de grupo, que podrá imponer el administrador del sistema.
- Premios: los premios serán regalos. Se desbloquearán cuando el usuario haya completado un nivel o haya realizado algún reto especial.

Perfiles de jugador [9]

Se ha aplicado la teoría de Bartle [9] para clasificar los jugadores en cuatro categorías:

- Triunfador: tendrá como objetivo llevar a cabo las misiones y obtener los premios o recompensas.
- Explorador: son jugadores que les gusta descubrir o aprender cosas nuevas.
- Socializadores: más que interés en conseguir los logros, buscan aprovecharlos para entablar relaciones sociales.
- Killers: buscan ser los primeros en el juego. Quieren destacar sobre otros jugadores.

Es importante destacar que lo que hace que un jugador actúe de una forma u otra es su personalidad (en este contexto hablaremos de perfiles). Normalmente, las aplicaciones de juegos están orientadas a un único tipo de jugador, consiguiendo que el resto las abandonen al poco tiempo, al no coincidir los objetivos del juego con su personalidad. Sin embargo, paradójicamente, los objetivos de estos sistemas informáticos son siempre fidelizar a la mayor cantidad de usuarios durante la mayor cantidad de tiempo.

Lo novedoso del sistema de ludificación es que no está orientado a un único perfil de jugador sino a los cuatro a la vez. Para ello, se cuenta con actividades que tengan como función satisfacer a estos tipos de usuario.

Grupos

Los jugadores podrán crear el número de equipos que deseen y participarán con los mismos a la hora de alcanzar las recompensas.

Capítulo 2

Herramientas utilizadas

2.1. Herramientas utilizadas

Para la elaboración de este Trabajo Fin de Grado se han utilizado las siguientes herramientas:

- Como editor LaTeX, *Texmaker* v. 4.4.1. [1].
- Como sistema de control de versiones, *GitHub* [2].
- Para la realización de copias de seguridad, *Dropbox* [3].
- Para tomar notas e ideas, *Microsoft OneNote* [4].
- Para realizar los diagramas *UML*, *Astah Professional* 7.1.0 [5].
- Para la elaboración de los prototipos iniciales, se ha utilizado *Ionic* 3 [6].
- Para la elaboración de la aplicación *Android* nativa, se ha utilizando *Android Studio* 3.1.
- Para la gestión de historias de usuario, se ha utilizado *Trello*. [7].
- Para el control y seguimiento del proyecto se ha utilizado la herramienta *Asana* [8].
- Para la realización de los diagramas de *Gantt* se ha empleado *Wrike* en una versión de prueba gratuita [9].

[1] http://www.xm1math.net/texmaker/
[2] https://www.github.com
[3] https://www.dropbox.com/
[4] https://www.onenote.com/
[5] http://astah.net/
[6] https://ionicframework.com/
[7] https://trello.com/#
[8] https://app.asana.com
[9] https://www.wrike.com/main/

2.2. Motivación de las tecnologías utilizadas

En esta sección se indican las diferentes tecnologías señaladas por la tutora para elaborar este trabajo: *Firebase* y *Kotlin*.

2.2.1. *Kotlin*

Figura 2.1: Logotipo de *Kotlin*.
[28]

Kotlin [49] es un lenguaje de programación diseñado por la empresa *Jetbrains* (creadora de *Android Studio* e *IntelliJ IDEA*), orientado a objetos y de tipado estático, que se ejecuta, en el caso de Android, sobre la máquina virtual de *Java (JVM)*. Fue señalado por *Google* en el *Google IO* de 2017 [10] como uno de los dos lenguajes de programación oficiales de Android y se espera que en un futuro sustituya a Java en esta tarea. Actualmente es muy conocido en el mundo del desarrollo de aplicaciones móviles, pero se espera que próximamente se empiece a utilizar tanto en el back-end de sistemas informáticos como en el front-end (páginas *Web*).

Point.kt

```kotlin
data class Point (var x: Float, var y:Float)
```

Point.java

```java
public class Point{

    private float x;
    private float y;

    public Point (float x, float y){
        this.x = x;
        this.y = y;
    }

    public float getX(){
        return x;
    }
    public float getY(){
        return y;
    }
}
```

Figura 2.2: *Snippet* ejemplo de un *POJO* Punto
A la izquierda, la versión en *Kotlin*, a la derecha, la de *Java*. Elaboración propia.

Kotlin tiene una serie de ventajas que merecen la pena señalar [28]:

1. Permite una interoperabilidad completa con Java: al utilizar ambos la misma máquina virtual (y compilar en *bytecode*), es posible la ejecución de métodos de *Java* en *Kotlin* y viceversa. De esta forma, permite integrar componentes desarrollados en un lenguaje desde el otro.

2. Es conciso: Kotlin elimina gran cantidad de *"boilerplate"* en el código fuente. Por ejemplo, en el caso de una clase *POJO*, los setters y los getters son generados en tiempo de compilación en el caso de que empleemos una clase de datos (*data class)*.

[10] https://events.google.com/io2017/recap/

3. Es *"null-safe"*: Según el blog de programación *Tapiki* [47], en un estudio realizado sobre 1.000 sistemas informáticos, un 70 % de ellos contenían errores de puntero nulo, que fue el motivo de excepción más frecuente. *Kotlin* intenta solucionar este problema forzando al programador a utilizar objetos que no pueden ser nulos, o bien, haciendo que tenga que indicar expresamente que un puntero puede ser nulo.

4. Es actualmente un estándar en programación de aplicaciones *Android*: El apoyo que *Google* está dando al lenguaje de programación es tal que resulta más que recomendable empezar a utilizarlo y acostumbrarse al mismo.

Por otro lado, también resulta de interés considerar los inconvenientes que suponen utilizar esta tecnología [43]:

1. Curva de aprendizaje: *Kotlin* tiene características que no tiene Java y, por tanto, existe una cierta curva de aprendizaje que es necesario solventar.

2. Menor cantidad de documentación frente a *Java*: al ser un lenguaje tan nuevo, apenas se cuenta con una comunidad de desarrolladores, por lo que encontrar documentación concreta puede llegar a ser un problema.

2.2.2. *Firebase*

Figura 2.3: Logotipo de *Firebase*.
[20]

Firebase es un producto de *Google* enmarcado dentro de su división *Google Cloud*, plataforma del Gigante de Internet que ofrece recursos informáticos bajo demanda. Es utilizada para facilitar el desarrollo de aplicaciones *Web* y móviles, que posibilita la reducción del tiempo que se dedica a desarrollar áreas no específicas de la aplicación haciendo que el desarrollador pueda centrarse en aquello que la hará distinta. Fue creada en el año 2011 aunque en 2014 fue adquirida por *Google*.

Firebase ofrece un gran número de componentes ya desarrollados y listos para integrar, como las sesiones, las bases de datos, el alojamiento de archivos, las funciones, las estadísticas, etcétera.

Los motivos por los que se ha escogido *Firebase* son los siguientes [36]:

1. Facilita el desarrollo: los componentes que conforman el ecosistema *Firebase* ya se encuentran desarrollados y están listos para ser integrados en la aplicación. Estos son probados y utilizados por millones de usuarios todos los días y aplica los últimos estándares de seguridad informática.

2. *Serverless* : en caso de no utilizar *Firebase*, el equipo de desarrollo tendría que preocuparse por los servidores y conexiones de red, intentando tener el máximo tiempo de *uptime*, instalar actualizaciones, escalar los recursos informáticos, asegurar la redundancia, etcétera.

3. Facilita la sincronización de los datos: *Firebase* está construido para que el usuario pueda trabajar sin conexión sobre la aplicación y actualizar los datos en cuanto tenga acceso a la red.

Firebase también tiene una serie de desventajas:

1. Dependencia de una plataforma específica: Una vez se lanza una aplicación elaborada con *Firebase* es muy complicado cambiar de tecnología, puesto que deberemos añadir los costes de desarrollo de todos aquellos componentes que utilicen la plataforma.
2. Interfaces: Las interfaces de acceso a datos de *Firebase* no son muy flexibles y en algunos casos no podremos consultar directamente la información a la base de datos, sino que será necesario descargarla por completo para realizar las consultas en otra plataforma.

En cuanto a las bases de datos de *Firebase*, se ha de destacar que ofrece en su paquete dos de tipo no relacional, *Firebase Firestore* (basada en documentos) y *Firebase Realtime* (basada en pares clave-valor). Para este trabajo se ha elegido la primera por varios motivos:

1. Mayor complejidad en las consultas: *Firestore* ofrece operaciones básicas de tipo *WHERE*. Igualmente, ofrece los operadores *ORDER BY* y *LIMIT*.
2. Mayor escalabilidad: En *Realtime* cuando se accede a una parte del árbol (clave-valor), el usuario debe descargarse todos los hijos. En *Firestore* únicamente aquellos documentos que se deseen.
3. Mayor seguridad: *Firestore* ofrece mayores medidas de seguridad al tener reglas de lectura y escritura más complejas y flexibles.

2.3. Comunicación con el servidor

La comunicación con el servidor es fundamental para el buen funcionamiento de la aplicación. En concreto, se deberá comunicar por los siguientes motivos:

- Realización de estadísticas: inicios y finalizaciones de una misión (para comprobar el éxito de las mismas), uso del usuario de la aplicación, etcétera.
- Realización de misiones: las misiones de consumo y de reto social requieren una conexión a Internet para asegurarse de que el usuario haya completado el reto.
- Descarga de nuevas misiones y niveles: uno de los requisitos del sistema informático en su conjunto es el de que el administrador pueda gestionar desde la parte Web los niveles y misiones, añadiéndolas y modificándolas bajo demanda.
- Manteninimiento de un ranking global.
- Conexión multidispositivo: se busca que el usuario pueda acceder a la aplicación desde distintos dispositivos, utilizando en todos ellos el sistema operativo *Android*.

Encontraremos dos tipos de conexiones:

- Conexiones realizadas a *Firebase*: datos estadísticos, de sesiones, invitaciones, base de datos, etcétera.
- Conexiones realizadas para lanzar funciones de *Firebase* utilizando la autenticación de usuarios.

Capítulo 3

Route66App

3.1. Plan de Desarrollo del *Software*

3.1.1. Descripción del proyecto

La ludificación comenzó a popularizarse en 2010 [56], y desde entonces, tal y como he explicado en puntos anteriores de esta memoria, se ha aplicado en variados ámbitos (educación, ventas, proveedores, etcétera). En el caso que nos ocupa, un restaurante de comida americana, no podemos encontrar en España ningún ejemplo con una aplicación similar.

Route66App es una *app* para dispositivos móviles basado las ideas expuestas en el Trabajo Fin de Grado de una alumna del Grado en Ingeniería en Organización Industrial [55]. Esta aplicación emplea la Teoría *Bartle* [9] para identificar los diferentes tipos de jugadores y buscará fidelizar y comprometer al usuario con los objetivos del negocio por medio del juego.

Es importante destacar que este trabajo abarca solo una parte del proyecto completo: la del cliente. Para su pleno funcionamiento se requerirá el gestor web, en el que se permitirán añadir y modificar las diferentes misiones que conforman el juego.

3.1.1.1. Propósito, objetivos y alcance

El objetivo fundamental de este trabajo es el de la implementación de un sistema de ludificación, aplicado a un restaurante de comida americana. En concreto se ocupará de la parte cliente, ejecutada desde terminales móviles con el Sistema Operativo *Android*.

La aplicación deberá contar con las siguientes funciones:

- Acciones propias del usuario.
  - Realizar misiones y juegos, con el fin de conseguir puntos y aumentar el nivel o posición en un ranking.
  - Modificación del perfil del usuario, permitiendo actualizar el avatar del usuario.
  - Consulta del historial de logros del usuario.
  - Consulta de las insignias del usuario.
  - Consulta del código *QR* que identifica al usuario de forma única.
  - Consulta de la posición global del usuario (ranking).

- Acciones a realizar en grupo.
  - Cosulta de los equipos a los que pertenece el usuario.
  - Creación de nuevos equipos.

Los objetivos serán:

- Elaborar un sistema de usuarios mediante el uso de sesiones.
- Gestionar un sistema de insignias, premios y logros (asignables tanto a equipos como a usuarios individuales).
- Aumentar el compromiso del usuario con la aplicación, mediante el uso de juegos o misiones. Se busca que el usuario realice acciones correspondientes a los cuatro roles mencionados anteriormente.
- Hacer las funciones de una tarjeta de fidelización virtual, que permita al usuario identificarse a la hora de pagar.

3.1.2. Metodología de Desarrollo del proyecto

El proyecto aplicará el Proceso Unificado (*Unified Process*), en concreto el proceso UPedu [44], cuyas características básicas [53] son:

- Dirigido por los casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

La iteratividad supone ciertas ventajas, como la de poder mitigar los riesgos más críticos en las fases más tempranas o permitir obtener *feedback* mucho antes, además de reaprovechar el aprendizaje obtenido en etapas posteriores.

3.1.3. Restricciones

Se imponen las siguientes restricciones, extraídas tras una serie de entrevistas con la tutora del trabajo:

1. **Tiempo:** Este proyecto se realiza para cumplir con los objetivos de la asignatura Trabajo de Fin de Grado. Mención Ingeniería del *Software*, correspondiente a 12 créditos ECTS, unas 300 horas.
2. **Plazo mínimo de entrega:** Este proyecto se podrá entregar cuando haya terminado el resto de asignaturas de la Mención. Por tanto, hasta que no termine y tenga calificada la única asignatura que tendré en el segundo cuatrimestre (Prácticas en Empresa), no podrá ser entregado. Las prácticas curriculares en la empresa *Telefónica Investigación y Desarrollo* concluyen el día 10 de Abril de 2018.
3. **Requisitos de la plataforma:** La aplicación será instalable en terminales *Android* cuya versión de *API* sea igual o superior a la 19 (*Android* 4.4, *KitKat*). Según el sitio *Web* oficial de desarrolladores de *Android* [1], esto supone cubrir la ingente mayoría de los dispositivos (un 92.5 %, a fecha 25 de Noviembre de 2017) [2].
4. **Requisitos de software**: Se impone que la aplicación se deba desarrollar en el lenguaje de programación *Kotlin*. Así mismo, se ha recomendado el uso de *Firebase* como plataforma para alojar la base de datos y gestionar las sesiones.
5. **Datos en el dispositivo**: la aplicación deberá contar con una caché de los datos en el dispositivo del cliente con el fin de reducir el número de peticiones que se hagan contra *Firebase*, así como para acelerar el funcionamiento de la aplicación en lugares con una baja conectividad.

[1] `https://developer.android.com/`

3.1.4. Estructura organizativa

3.1.4.1. Estructura organizativa interna

La estructura organizativa del proyecto estará formada por una única persona, que deberá ejercer cada uno de los roles en el momento en el que sea necesario. [44]

Rol	Encargado
Desarrollador	Álvaro Carreras Regorigo
Analista	Álvaro Carreras Regorigo
Gestor de proyecto	Álvaro Carreras Regorigo
Diseñador	Álvaro Carreras Regorigo

Cuadro 3.1: Roles.

Desarrollador

Este rol se encarga de implementar el código y de realizar las pruebas para los entregables *software* generados, con el fin de que sean integrados.

Analista

Organiza y coordina tanto la elicitación de requisitos como el modelado de casos de uso, indicando y limitando la funcionalidad del sistema y el alcance del mismo.

Gestor de Proyecto

Realiza el control y mando del proyecto, asigna recursos, gestiona los riesgos, distribuye responsabilidades, gestiona la interacción con los clientes, etcétera.

Diseñador

Como indica su nombre, está encargado de diseñar el sistema, definiendo las responsabilidades, operaciones y relaciones de cada uno de los componentes del sistema.

3.1.4.2. Estructura organizativa externa

Solo se identifica un rol en la estructura organizativa externa: el del usuario de la aplicación.

3.1.5. Gestión del proyecto

3.1.5.1. Planificación del proyecto

En este proyecto se utilizará la metodología UPedu [44], compuesta por fases y estas por iteraciones.

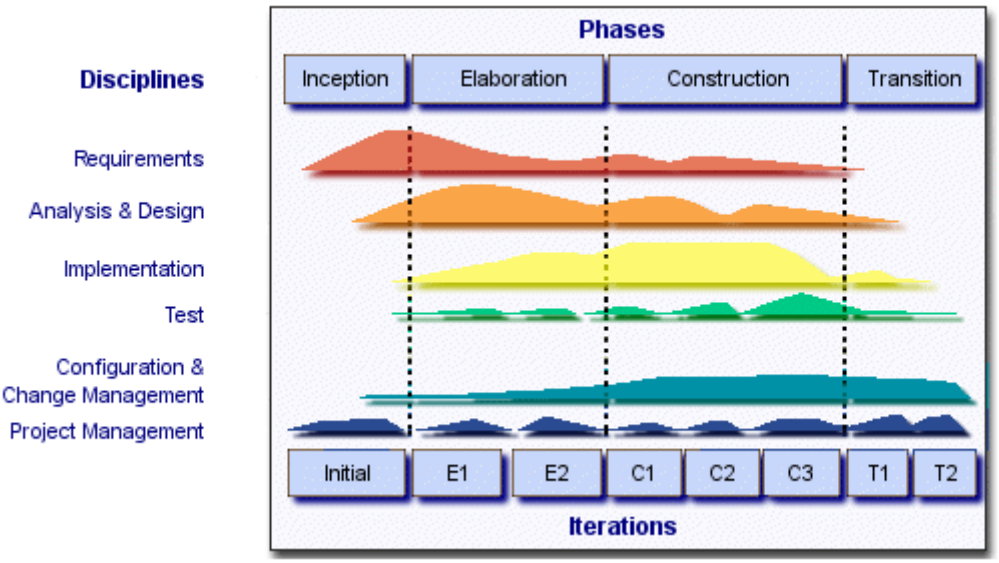


Figura 3.1: Fases de la metodología UPedu.
[44]

En la figura 3.1, se puede observar el esfuerzo necesario de cada disciplina en cada fase del proyecto.

Fase de inicio

En esta fase no se realiza ningún entregable de tipo *software*, pues estos se deben tratar antes de comenzar con la implementación. Los objetivos fundamentales de esta fase son la definición del alcance del proyecto, la determinación de los casos de uso críticos, la estimación del coste y la duración del proyecto, y la elaboración de una primera gestión de riesgos.

Fase de elaboración

El objetivo fundamental de esta fase [53] es el de la elaboración del diseño y arquitectura del sistema informático. Al mismo tiempo, se realizará un análisis del dominio del sistema y se eliminarán los elementos de mayor riesgo para el desarrollo del proyecto.

Después de esta fase, se obtendrán una arquitectura, requisitos y planes de desarrollo estables y se habrán reducido los riesgos más destacables.

Fase de construcción

Es sin ninguna duda la parte más larga del proyecto y la primera que tendrá como *output* entregables software utilizables. Se irá iterando y mejorando la calidad del producto software en cada ciclo.

La salida de esta fase [53] es software en estado beta.

Fase de transición

Se deberá conseguir la aceptación por parte del usuario, indicando que cumple con la visión inicial del proyecto. De igual modo se distribuirá el *software*.

3.1.5.2. Calendario del proyecto

Como se ha indicado en los apartados anteriores, este proyecto se ha dividido en las fases que el Proceso Unificado, en su variación UPedu [44] establece. A continuación, se lista cada fase, el número de iteraciones y las horas de trabajo y duración estimadas. Estos cálculos son orientativos, las variaciones sobre los mismos están expuestas en el apartado correspondiente.

Nombre de la fase	Núm.. Iteraciones	Horas de trabajo	Duración estimada
Inicio	1	10 horas/semana	61 horas
Elaboración	2	20 horas/semana	80 horas
Construcción	2	20 horas/semana	133 horas
Transición	1	20 horas/semana	42 horas

Cuadro 3.2: Fases del proceso UPedu.

La duración estimada total es de 316 horas, a las que habría que añadir las necesarias para acudir a las tutorías.

Fase de inicio

Comienza el día 6 de Noviembre de 2017 y finaliza el 10 de Diciembre de 2017. Consta de una única iteración.

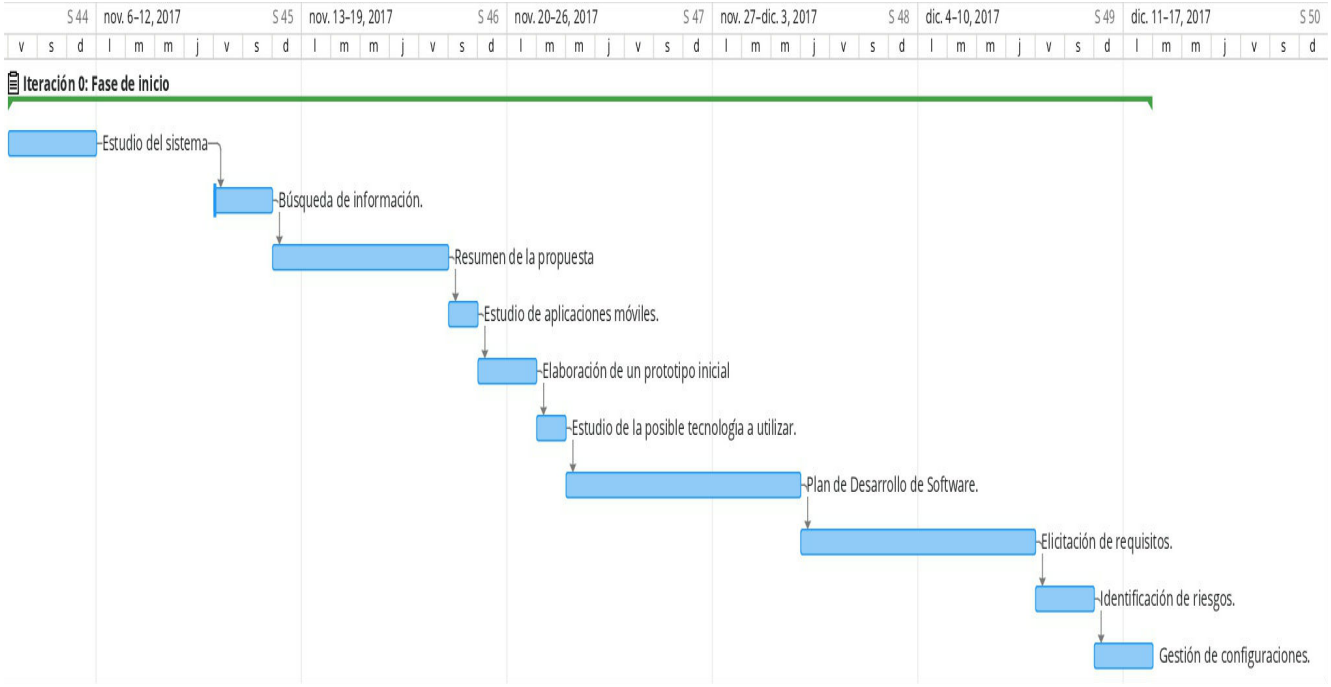


Figura 3.2: Diagrama de Gantt para la iteración 0 (fase de inicio).

Nombre de la tarea	Fecha de comienzo	Fecha de fin	Duración
Estudio del sistema.	03/11/2017	05/11/2017	10 horas
Búsqueda de información.	10/11/2017	11/11/2017	4 horas
Resumen de la propuesta	11/11/2017	17/11/2017	6 horas
Estudio de aplicaciones móviles.	17/11/2017	17/11/2017	2 horas
Elaboración de un prototipo inicial	18/11/2017	19/11/2017	7 horas
Estudio de la posible tecnología a utilizar.	19/11/2017	19/11/2017	3 horas
Plan de Desarrollo de Software.	24/11/2017	01/12/2017	13 horas
Elicitación de requisitos.	01/12/2017	08/12/2017	12 horas
Identificación de riesgos.	08/12/2017	09/12/2017	3 horas
Gestión de configuraciones.	09/12/2017	10/12/2017	1 horas

Cuadro 3.3: Iteración 0. Fase de Inicio.

Reuniones realizadas con la tutora del T.F.G.:

- 30 de Octubre de 2017.
- 6 de Noviembre de 2017.
- 20 de Noviembre de 2017.
- 4 de Diciembre de 2017.

Fase de elaboración

Se ha decidido dejar un lapso temporal de 41 días (10 de Diciembre - 20 Enero) por la gran acumulación de trabajos y exámenes.

Esta fase comienza el día 20 de Enero de 2017 y finaliza el 1 de Marzo de 2018. Consta de dos iteraciones.

- Iteración 1: Con una duración de 3 semanas.
- Iteración 2: Con una duración de 1 semana y media.

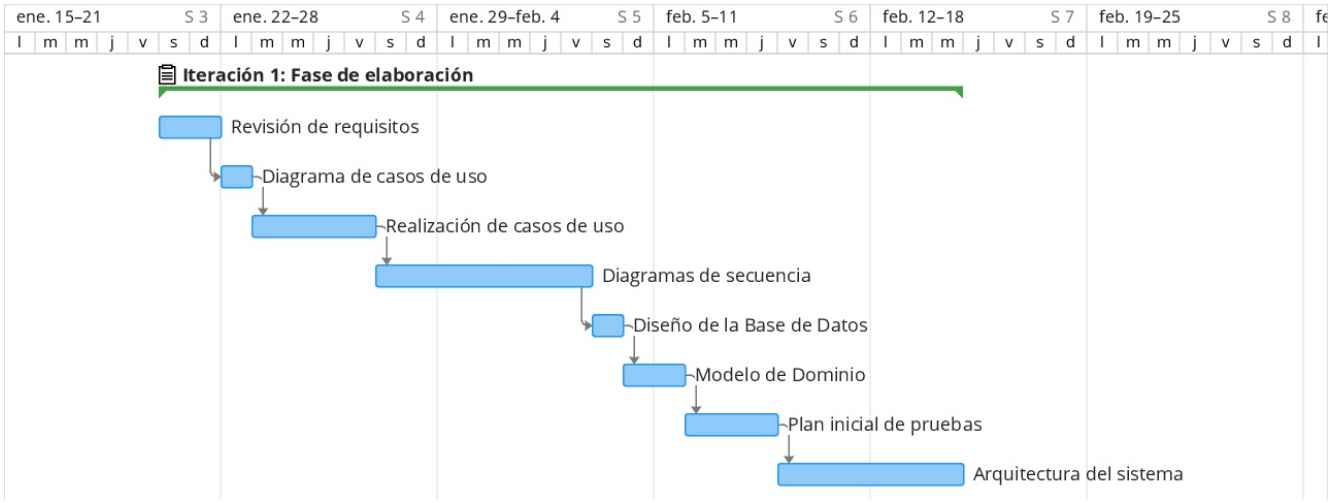


Figura 3.3: Diagrama de Gantt para la iteración 1 (fase de elaboración).

Nombre de la tarea	Fecha de comienzo	Fecha de fin	Duración
Revisión de requisitos	20/01/2018	21/01/2018	5 horas
Diagrama de casos de uso	22/01/2018	22/01/2018	1 hora
Realización de casos de uso	22/01/2018	25/01/2018	10 horas
Diagramas de secuencia	26/01/2018	01/02/2018	20 horas
Modelo de Dominio	02/02/2018	03/02/2018	5 horas
Diseño de la Base de Datos	02/02/2018	02/02/2018	3 horas
Plan inicial de pruebas	03/02/2018	05/02/2018	5 horas
Arquitectura del sistema	05/02/2018	10/02/2018	13 horas

Cuadro 3.4: Iteración 1. Fase de elaboración.

Reuniones realizadas con la tutora del T.F.G.:

- 6 de Febrero de 2018.

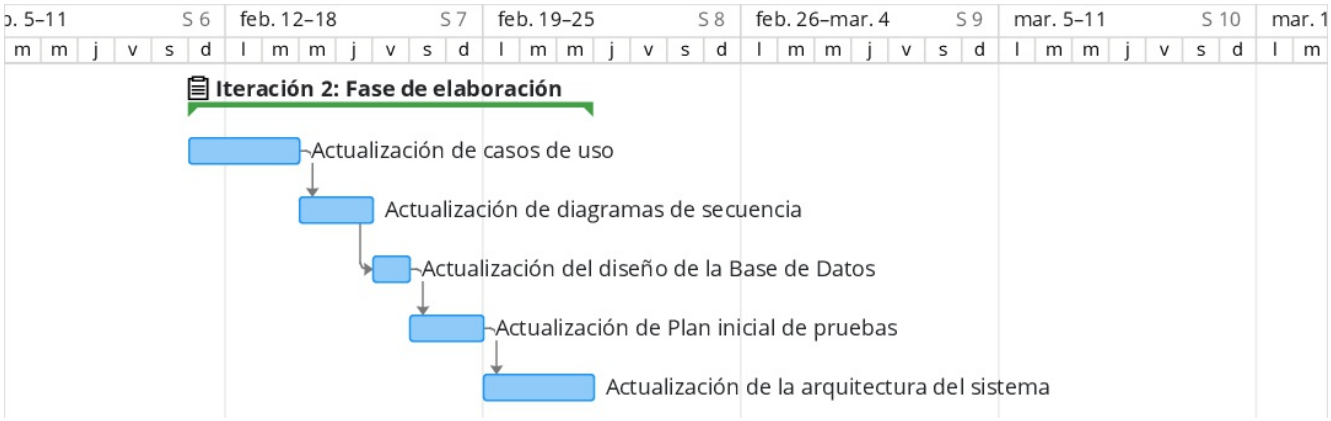


Figura 3.4: Diagrama de Gantt para la iteración 2 (fase de elaboración).

Nombre de la tarea	Fecha de comienzo	Fecha de fin	Duración
Actualización de casos de uso	11/02/2018	13/02/2018	5 horas
Actualización de diagramas de secuencia	14/02/2018	15/02/2018	4 horas
Actualización del diseño de la Base de Datos	16/02/2018	16/02/2018	2 horas
Actualización de Plan inicial de pruebas	17/02/2018	18/02/2018	2 horas
Actualización de la arquitectura del sistema	18/02/2018	20/02/2018	5 horas

Cuadro 3.5: Iteración 2. Fase de elaboración.

El día 20 de Febrero se hace entrega a la tutora de todos los entregables realizados hasta la fecha, con el fin de que puedan ser revisados y validados en profundidad.

Fase de construcción

Comienza el día 1 de Marzo de 2018 y finaliza el 10 de Abril de 2018.

Figura 3.5: Diagrama de Gantt para la iteración 3 (fase de construcción).

Nombre de la tarea	Fecha de comienzo	Fecha de fin	Duración
Implementación del sistema Realización de casos de prueba	20/02/2018	01/04/2018	116 horas

Cuadro 3.6: Iteración 3. Fase de construcción.

En la iteración 3 (Fase de construcción) se ha estipulado la misma duración y mismas fechas a ambas tareas porque se planea implementar e inmediatamente probar.

Reuniones realizadas con la tutora del T.F.G.:

- 6 de Marzo de 2018.
- 20 de Marzo de 2018.

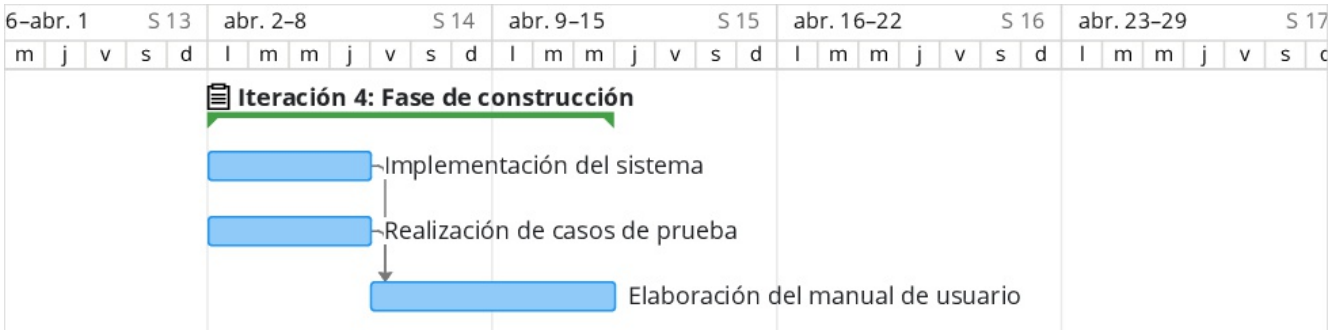


Figura 3.6: Diagrama de Gantt para la iteración 4 (fase de construcción).

Nombre de la tarea	Fecha de comienzo	Fecha de fin	Duración
Implementación del sistema Realización de casos de prueba	02/04/2018	05/04/2018	7 horas
Elaboración del manual de usuario	05/04/2018	10/04/2018	10 horas

Cuadro 3.7: Iteración 4. Fase de construcción.

Reuniones realizadas con la tutora del T.F.G.:

- 10 de Abril de 2018.

Fase de transición

Comienza el día 10 de Abril de 2018 y finaliza el 30 de Abril de 2018.

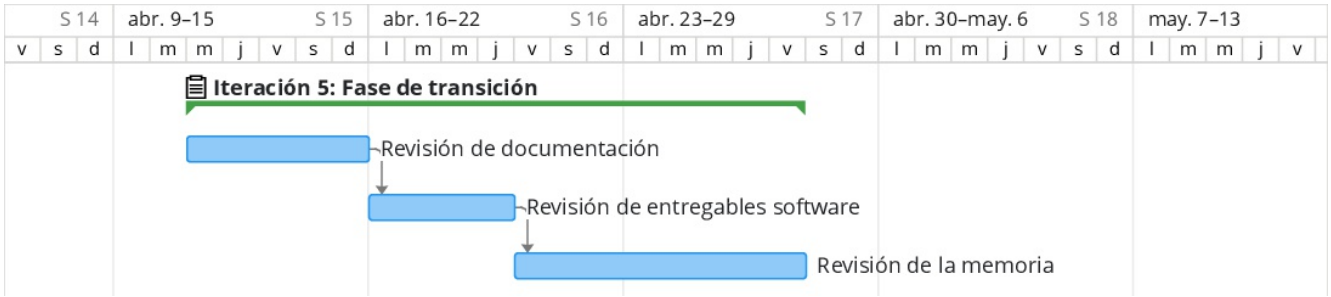


Figura 3.7: Diagrama de Gantt para la iteración 5 (fase de transición).

Nombre de la tarea	Fecha de comienzo	Fecha de fin	Duración
Revisión de documentación	11/04/2018	15/04/2018	12 horas
Revisión de entregables software	15/04/2018	18/04/2018	9 horas
Revisión de la memoria	18/04/2018	25/04/2018	21 horas

Cuadro 3.8: Iteración 5. Fase de transición.

3.1.6. Relación de entregables no software

Los distintos entregables, que se considerarán como definitivos en su última versión, son los siguientes:

Fase de inicio

- Plan de Desarrollo de *Software*.
- Versión inicial del Plan de Gestión de Riesgos.

Fase de elaboración

- Documento de requisitos *software*.
- Versión inicial de los Casos de Uso.
- Diagramas de secuencia.
- Arquitectura del sistema.
- Diseño de la Base de Datos.
- Elaboración de prototipos.

Fase de construcción

- Versión inicial del manual de usuario.
- Documento de casos de prueba.
- Documento de resultados de las pruebas.

Fase de transición

- Versión final del manual de instalación.
- Versión final del manual de usuario.

3.1.7. Control y seguimiento del proyecto

Se hará un control y seguimiento del proyecto en todo momento, utilizando la herramienta *Asana* [2], tal y como se ha expuesto en el apartado “Herramientas utilizadas“.

El control y seguimiento del proyecto resulta ser una tarea fundamental para asegurar el buen progreso del proyecto y asegurar el cumplimiento de los objetivos marcados, en especial en cuanto a la realización de los hitos establecidos, ya que el gestor del proyecto podrá observar las distintas variaciones de tiempo programadas versus reales, para así poder reaccionar y tomar las medidas que sean precisas para ajustarse al cronograma del proyecto.

[2] `https://app.asana.com`

3.1.8. Gestión de riesgos

Se identifican los siguientes riesgos:

R0	Falta de experiencia del alumno
Descripción	El alumno ha realizado prácticas en *Android*, pero puede encontrarse en momentos en los que requiera experiencia adicional.
Consecuencia	Ralentización del trabajo.
Probabilidad	Media.
Impacto	Bajo.
Estrategia	Reservar el riesgo.
Plan de acción	Buscar información y ayuda, ya sea preguntando a la tutora del trabajo o utilizando otros medios.
Plan de contingencia	Revisión de conocimientos existentes.

R1	Retraso en la planificación
Descripción	Debido a errores en la estimación, la planificación no se ajusta a la realidad y se producen demoras en las fechas de entrega.
Consecuencia	Retraso en la entrega de los entregables.
Probabilidad	Alta.
Impacto	Crítico.
Estrategia	Reservar el riesgo.
Plan de acción	Realizar una nueva planificación más realista.
Plan de contingencia	Realizar revisiones periódicas del calendario de planificación.

R2	Adelantos en la planificación
Descripción	Debido a errores en la estimación, la planificación no se ajusta a la realidad y se producen adelantos en las fechas de entrega.
Consecuencia	Modificación de toda la planificación.
Probabilidad	Alta.
Impacto	Crítico.
Estrategia	Reservar el riesgo.
Plan de acción	Realizar una nueva planificación más realista.
Plan de contingencia	Realizar revisiones periódicas del calendario de planificación.

R3	Falta de experiencia en grandes proyectos
Descripción	El equipo no cuenta con experiencia propia de proyectos de gran extensión que le permita calcular mejor las estimaciones, así como realizar el proyecto con mayor celeridad.
Consecuencia	Peor organización y estimaciones. Grandes picos de trabajo.
Probabilidad	Alta
Impacto	Alto.
Estrategia	Reducción del riesgo.
Plan de acción	Realizar una nueva planificación.
Plan de contingencia	Realizar revisiones periódicas del calendario de planificación.

R4	Falta de disponibilidad
Descripción	El alumno no puede continuar con el trabajo por falta de disponibilidad.
Consecuencia	Dependerá del tiempo en el que esté no disponible y las holguras en la planificación. Si es muy largo, puede suponer la no entrega del trabajo a tiempo.
Probabilidad	Baja.
Impacto	Crítico.
Estrategia	Reservar del riesgo.
Plan de acción	Realizar una nueva planificación basándose en las nuevas fechas de disponibilidad.
Plan de contingencia	Monitorizar la disponibilidad del alumno.

R5	Pérdida de datos
Descripción	Por cualquier motivo, se pierde total o parcialmente cualquiera de los entregables ya realizados o en proceso de realización.
Consecuencia	Ralentización muy alta del trabajo. Repetición de los entregables perdidos.
Probabilidad	Bajo.
Impacto	Crítico.
Estrategia	Evitación del riesgo.
Plan de acción	No se aplica.
Plan de contingencia	Realización de copias de seguridad y uso de un sistema de control de versiones (ver apartado gestión de configuraciones).

R6	Falta de medios software de desarrollo
Descripción	Las herramientas designadas para desarrollar el proyecto fallan, desaparecen o dejan de estar disponibles.
Consecuencia	Ralentización muy alta del trabajo. Repetición de los entregables perdidos.
Probabilidad	Bajo.
Impacto	Crítico.
Estrategia	Evitación del riesgo.
Plan de acción	No se aplica.
Plan de contingencia	Utilización de herramientas profesionales, aplicación de las últimas actualizaciones.

R7	Falta de medios hardware de desarrollo
Descripción	Las herramientas designadas para desarrollar el proyecto fallan o dejan de estar disponibles.
Consecuencia	Adquisición de nuevos materiales.
Probabilidad	Bajo.
Impacto	Alto.
Estrategia	Evitación del riesgo.
Plan de acción	No se aplica.
Plan de contingencia	Realización de copias de seguridad de los entregables, para evitar su pérdida en caso de ocurrir este riesgo.

R8	Diseño pobre o incorrecto
Descripción	El diseño realizado del sistema es pobre o incorrecto.
Consecuencia	Ralentización del trabajo.
Probabilidad	Medio.
Impacto	Crítico.
Estrategia	Reducción del riesgo.
Plan de acción	Rediseñar la arquitectura del sistema.
Plan de contingencia	Realizar revisiones con el cliente y realizar pruebas sobre el modelo diseñado.

3.1.9. Gestión de configuraciones

Hoy en día la mayoría de los servicios públicos que ofrecen alojamiento de repositorios *Git*, permiten realizar una buena gestión de configuraciones. Esto es debido a que cada cambio que se hace sobre el código fuente (*commit*) viene asociado a informaciones sobre qué se ha cambiado, quién lo ha cambiado, cuándo lo ha cambiado, etcétera. Además, las ramas o *branches*, aquellos lugares donde van dirigidos los cambios, permiten imponer bloqueos sobre escritura a ciertos usuarios, con el fin de proteger los entregables que estuvieran en la misma.

Por otro lado, estos servicios también pueden ser utilizados para la gestión de *releases*. Esto se puede llevar a cabo mediante la creación de una rama específica para ese propósito o, directamente, y si se permite, mediante el apartado correspondiente desde el sitio web.

En el caso de este proyecto se han utilizado dos repositorios privados en *GitHub*[3], en uno de ellos se tratará el desarrollo de esta memoria y en otro, el del código del sistema informático que se desea construir.

3.1.10. Estimación de los costes

Para una correcta estimación de los costes, se debe procurar partir de los datos proporcionados por proyectos reales, que nos permitan realizarla partiendo de una experiencia. Desafortunadamente, como estos no se hacen públicos (pues son parte de la estrategia de la empresa), no es posible contar con ellos. Por este motivo, los cálculos se han realizado basándose en lo estudiado en la asignatura Planificación y Gestión de Proyectos [52] y en información encontrada por Internet.

El resultado de la estimación total de costes es de **10.094,96€**, que se detallarán a continuación.

3.1.10.1. Costes *hardware* y *software*

Costes *hardware*

Dispositivo	Coste de compra	Horas dispositivo	Horas utilizadas	Coste total
Teléfono *Android*	120 €	17.520 horas (2 años, 24 horas diarias)	133	0,92 €
Portátil	399 €	11.680 horas (4 años, 8 horas diarias)	300 horas	10,25 €

Cuadro 3.9: Costes de los recursos *hardware*.

Coste total del *hardware* es: **11,16 €**.

Los costes de software son, en realidad, 0 €, ya que se ha priorizado utilizar software gratuito (*freeware*) y, en los casos en los que se ha decidido usar uno privativo, se han empleado licencias gratuitas para estudiantes. Se ha de tener presente que estos son cálculos que sí se aplicarían en un proyecto real, que tenga como fin un beneficio económico.

[3] `https://github.com/`

Servicio	Coste mensual	Coste total
GitHub Developer	5,90 €	47,20 €
Firebase	21 €	168€
Astah Professional	7,50 €	60 €

Cuadro 3.10: Costes de los recursos *software*.

Coste total de software: **240,80 €**.

Los datos anteriores se han calculado basándose en los datos públicos de las herramientas, convertiendo los importes a euros según la tasa de cambio del 28 de Noviembre de 2017, para un período de 7 meses (Noviembre-Mayo).

El resto de *software* utilizado (sistema operativo, *IDE*, etcétera) es de licencia gratuita. No se ha empleado software cuya licencia solo fuera gratuita en casos de uso no comercial del mismo.

Coste total de este apartado: **251,96 €**.

3.1.10.2. Costes de dietas, viajes y aprendizajes

Los costes de dietas y de aprendizaje no se aplican en este apartado porque, en primer lugar, no se hará ninguna dieta externa y, en segundo lugar, se aplicarán los conocimientos adquiridos a lo largo del Grado, así como otros externos que se han adquirido fuera del tiempo de asignación del proyecto.

En cuanto a los viajes, aquellos que serían precisos realizar para poder reunirse con la persona que ha encargado el proyecto (la tutora del mismo) van desde mi residencia habitual que está en Laguna de Duero hasta el Campus Miguel Delibes. Utilizado la herramienta de mapas *Vía Michelín*[4] y un coche urbano de diésel, los costes de viajes y transportes son:

- **Kilómetros recorridos por viaje:** 33 (16,5 por trayecto).
- **Coste por viaje:** 3 € (1,50 € por trayecto).
- **Viajes realizados:** 15.
- **Coste total viajes:** 45 €.

Por tanto, el coste total de este apartado es: **45€**.

3.1.10.3. Coste del esfuerzo (recursos humanos y seguros sociales)

Para el caso de los primeros, los costes de los recursos humanos, debemos ver el listado de roles que se van a practicar a lo largo del proyecto:

Rol	Salario medio	€/Hora	Horas estimadas	Coste total
Analista	42.987 €	22 €	300 horas	6.600 €
Desarrollador	42.987 €	22 €		
Gestor de proyecto	42.987 €	22 €		
Diseñador	42.987 €	22 €		
Director de proyecto	42.987 €	22 €		

Cuadro 3.11: Costes de los recursos humanos.

[4] `https://www.viamichelin.es`

Los datos de la tabla anterior han sido calculados basándose en los datos del Informe Anual *Infojobs* 2016 España [25] (salario promedio bruto anual de un arquitecto de proyectos informático). Se ha dividido la cantidad anterior en doce pagas y veinte días de trabajo por mes.

Igualmente, habría que considerar el coste de los seguros sociales. Como la estimación del proyecto se ha realizado teniendo un único empleado en régimen de autónomo, solo se aplicaría el coste de las cuotas que establece el Régimen Especial de Autónomos, que en 2018 es de 50 € al mes para nuevos autónomos menores de 30 años [24].

- Cuota mensual autónomos.
  - **Importe mensual:** 50 €.
  - **Número de cuotas (meses):** 7.
  - **Importe total:** 350 €.

  El coste total de este apartado es de **6.950 €**.

3.1.10.4. Costes indirectos aplicados al personal del proyecto

Los costes indirectos son aquellos derivados del uso de calefacción, electricidad, comunicaciones, etcétera.

- Alquiler: se utilizará el servicio de vivero de empresas que propone el Parque Científico de la Universidad de Valladolid [45] para una sala de 23 m^2. Este tipo de alquileres ya incluye los gastos de calefacción y de electricidad.
  - **Coste estimado (mensual):** 276 €.
  - **Número de meses:** 7.
  - **Coste total:** 1.932€.
- Conexión a Internet y llamadas *(Movistar Fibra óptica 300 MB)*.
  - **Coste estimado (mensual):** 45 € al mes + 19.99 € al mes por la cuota de línea. Total: 65 € al mes.
  - **Número de meses:** 7.
  - **Coste total:** 455 €.
- Telefonía móvil *(Simyo 4GB + llamadas ilimitadas)*.
  - **Coste estimado (mensual):** 18 € al mes.
  - **Número de meses:** 7.
  - **Coste total:** 126 €.

  El coste total de este apartado es de **2.513 €**.

3.1.11. Desviaciones sobre el calendario planificado

Se adjunta a continuación la tabla de estimaciones de planificación, así como los retrasos que ha habido en realidad.

Iteración	Etapa	Fecha de inicio real	Fecha inicio estimada	Fecha de fin real	Fecha fin estimada	Diferencia finalización
0	Inicio	03/11/2017	06/11/2017	12/12/2017	10/12/2017	2 días
1	Elaboración	20/01/2018	20/01/2018	15/02/2018	10/02/2018	5 días
2	Elaboración	16/02/2018	11/02/2018	23/02/2018	20/02/2018	3 días
3	Construcción	24/02/2018	20/02/2018	03/04/2018	01/04/2018	2 días
4	Construcción	04/04/2018	02/04/2018	12/04/2018	10/04/2018	2 días
5	Transición	13/04/2018	11/04/2018	24/04/2018	25/04/2018	1 día

Cuadro 3.12: Desviaciones sobre el calendario planificado.

Tras un análisis de los retrasos sobre la planificación establecida se extraen las siguientes conclusiones:

- La planificación ha estado bastante ajustada a las estimaciones realizadas, encontrándose una mayor desviación en la iteración 1, correspondiente a la fase de elaboración. El motivo fue que se tuvo que compaginar el comienzo de las prácticas (y formación adicional) junto con el T.F.G., por lo que durante esos días no se pudo contar con el número de horas especificadas.
- En la segunda fase de elaboración, el retraso de tres días se debe a unos rediseños en el modelo de dominio y de la base de datos para ajustarse más al problema concreto.
- En la primera fase de construcción, hubo dos días en los que no se pudo trabajar porque se decidió ampliar la memoria del ordenador y sustituir el disco duro por uno de estado sólido (SSD) con el fin de aumentar la productividad en esta fase. En esos días me dediqué a instalar esos dos componentes, así como los elementos *software* requeridos para la continuación del trabajo.
- En la segunda fase de construcción, el retraso de dos días se debe al tiempo que se dedicó a redactar la memoria de las Prácticas en Empresa. Fue un retraso que no se contó en la planificación inicial.

3.2. Análisis

3.2.1. Requisitos

3.2.1.1. Requisitos funcionales

Relacionados con los usuarios

RF0	Inicio de sesión
Descripción	El sistema permitirá iniciar sesión.

RF1	Cierre de sesión
Descripción	El sistema permitirá cerrar sesión.

RF2	Registro e inicio de sesión rápido
Descripción	El sistema permitirá iniciar sesión y registrarse mediante el uso de proveedores de acceso.

RF3	Registro de usuarios
Descripción	El sistema permitirá registrar usuarios.

RF4	Configuración del usuario
Descripción	El sistema permitirá editar la configuración del usuario.

RF5	Visualización de la configuración del usuario
Descripción	El sistema permitirá visualizar la configuración actual del usuario.

RF6	Cambio de contraseña
Descripción	El sistema permitirá cambiar la contraseña actual del usuario.

RF7	Ranking de usuarios
Descripción	El sistema permitirá ver el ranking global de usuarios.

RF8	Invitación de usuarios a la aplicación
Descripción	El sistema permitirá invitar a usuarios a la aplicación.

RF9	Avatares
Descripción	El sistema permitirá añadir una fotografía de perfil (avatar) al usuario.

RF10	Avatares II
Descripción	El sistema permitirá visualizar la fotografía de perfil de otros usuarios de la plataforma.

RF11	Logros
Descripción	El sistema permitirá consultar los logros de un usuario.

RF12	Insignias
Descripción	El sistema permitirá consultar las insignias que tiene un usuario.

RF13	Premios
Descripción	El sistema permitirá consultar los premios que tiene un usuario.

RF14	Código *QR*
Descripción	El sistema permitirá mostrar el código *QR* asociado a una misión de tipo consumo.

RF15	Progreso del usuario
Descripción	El sistema permitirá mostrar los círculos de progreso relativos a cada usuario.

RF16	Identificación de usuarios
Descripción	El sistema permitirá identificar usuarios por correo electrónico.

RF17	Concesión de premios
Descripción	El sistema permitirá conceder premios a los usuarios.

RF18	Concesión de insignias
Descripción	El sistema permitirá conceder insignias a los usuarios.

Relacionados con los grupos

RF19	Creación de un grupo
Descripción	El sistema permitirá crear un grupo.

RF20	Eliminación de un grupo
Descripción	El sistema permitirá eliminar un grupo.

RF21	Unirse a un grupo
Descripción	El sistema permitirá unirse a un grupo.

RF22	Salirse de un grupo
Descripción	El sistema permitirá salirse de un grupo.

RF23	Expulsión de un grupo
Descripción	El sistema permitirá expulsar a un usuario de un grupo.

RF24	Invitación de usuarios a un grupo
Descripción	El sistema permitirá invitar a un usuario a un grupo.

RF25	Detalles de los grupos
Descripción	El sistema permitirá ver los detalles de los grupos a los que pertenece el usuario.

RF26	Avatares en los grupos
Descripción	El sistema permitirá añadir una fotografía de perfil (avatar) al grupo

RF27	Listado de los miembros del equipo
Descripción	El sistema permitirá mostrar el listado de usuarios miembros del grupo.

RF28	Ascenso en un grupo
Descripción	El sistema permitirá ascender a un usuario al rol administrador.

RF29	Logros de equipo
Descripción	El sistema permitirá asignar logros a un equipo.

RF30	Premios de equipo
Descripción	El sistema permitirá conceder premios a un equipo.

Relacionados con el ranking

RF31	Puntuación del usuario a nivel global
Descripción	El sistema mostrará la puntuación del usuario en el juego.

RF32	Ranking global
Descripción	El sistema permitirá mostrar un ranking global de jugadores.

Relacionados con las misiones

RF33	Misiones
Descripción	El sistema mostrará al usuario qué acción debe realizar para considerar la misión como finalizada.

RF34	Misiones de tipo juego
Descripción	El sistema permitirá mostrar un juego para que el usuario pueda completar una misión de tipo juego (competitiva).

RF35	Misiones de tipo social
Descripción	El sistema permitirá mostrar la información necesaria para completar una misión de tipo social.

RF36	Misiones de tipo consumo
Descripción	El sistema comprobará el gasto que haga el usuario en los restaurantes de la cadena, con el fin de comprobar si una misión de tipo gasto ha sido completada.

RF37	Consumo restante en misiones de tipo consumo
Descripción	El sistema permitirá mostrar el consumo que ya ha realizado el usuario en una determinada misión de tipo consumo.

Otros aspectos

RF38	Notificaciones
Descripción	El sistema emitirá notificaciones.

RF39	Misiones
Descripción	El sistema permitirá cumplir o realizar misiones.

RF40	Tratamiento de los errores de la aplicación
Descripción	El sistema monitorizará los errores de aplicación de los usuarios.

RF41	Estadísticas
Descripción	El sistema registrará estadísticas de uso de la aplicación.

Requisitos funcionales de información

RF42	Usuario
Descripción	El sistema almacenará, para cada jugador, la siguiente información: ■ Nombre público. ■ Correo electrónico. ■ Sexo. ■ Contraseña. ■ Avatar. ■ Misiones cumplidas. ■ Logros del jugador. ■ Insignias recibidas. ■ Premios obtenidos. ■ Consumos realizados. ■ Puntuaciones: puntuación global, puntuación social, puntuación competitiva y puntuación de consumo.

RF43	Insignia
Descripción	El sistema almacenará, para cada insignia, la siguiente información: ■ Tipo de insignia.

RF44	Logro de usuario
Descripción	El sistema almacenará, para cada logro de usuario, la siguiente información: ■ Tipo de logro. ■ Fecha y hora del logro. ■ Consumo realizado (en caso de estar relacionado con una misión de tipo consumo). ■ Misión (en caso de estar relacionado con una misión). ■ Puntos.

RF45	Premio
Descripción	El sistema almacenará, para cada premio (tanto de usuario como de equipo), la siguiente información: ■ Tipo de premio (o premio en concreto asignado). ■ Fecha y hora de comienzo de la validez del premio. ■ Fecha y hora de finalización de la validez del premio.

RF46	Grupo/Equipo
Descripción	El sistema almacenará, para cada grupo, la siguiente información. ▪ Nombre del grupo. ▪ Administrador del grupo ▪ Avatar del grupo. ▪ Premios obtenidos. ▪ Logros del grupo conseguidos.

RF47	Logro de equipo
Descripción	El sistema almacenará, para cada logro de grupo/equipo, la siguiente información. ▪ Tipo de logro. ▪ Fecha y hora del logro.

3.2.1.2. Requisitos no funcionales

Relacionados con los usuarios

RNF0	Unicidad del correo electrónico de registro
Descripción	El correo electrónico registrado y asociado a una cuenta de usuario deberá ser único.

RNF1	Proveedores de acceso
Descripción	Se ofrecerá un login/registro rápido mediante *Google*.

RNF2	Identificación de usuarios
Descripción	Se identificará a los usuarios internamente por el identificador único de usuario generado por *Firebase*.

Relacionados con los grupos/equipos

RNF3	Unicidad del administrador del grupo
Descripción	Únicamente se permitirá que haya un único administrador por grupo.

RNF4	Edición de parámetros del grupo
Descripción	El avatar del grupo, así como el resto de parámetros del equipo solo podrá ser modificado por el administrador del grupo.

RNF5	Expulsión de usuarios del grupo
Descripción	El administrador del grupo será el único miembro que podrá expulsar a un usuario del mismo.

RNF6	Nombramiento del administrador del grupo
Descripción	El administrador del grupo será aquel que creó el grupo, o bien, aquel que fue designado por un anterior administrador del grupo.

RNF7	Número mínimo y máximo de usuarios en un grupo
Descripción	Un equipo podrá estar formado, como mínimo por un usuario.

Software a utilizar para el despliegue de la aplicación.

RNF8	Datos estadísticos de la aplicación
Descripción	El sistema utilizará *Firebase Analytics* para la gestión de ciertas estadísiticas (número de usuarios de la aplicación, usuarios activos, datos de bloqueos, efectividad de las notificaciones, etcétera.).

RNF9	Almacenamiento de avatares
Descripción	El sistema utilizará *Firebase Storage* para el almacenamiento de los avatares de perfil tanto de los usuarios como de los grupos.

RNF10	Biblioteca de gestión de solicitudes *HTTP*
Descripción	El sistema utilizará *OkHTTP* como biblioteca de gestión de solicitudes *HTTP*.

RNF11	Gestión de sesiones
Descripción	El sistema utilizará *Firebase Authentication* para la gestión de sesiones.

RNF12	Invitación de usuarios a un grupo
Descripción	El sistema utilizará el nombre visible de los usuarios para la búsqueda a la hora de invitar a un usuario a un grupo.

RNF13	Invitación de usuarios a la aplicación
Descripción	El sistema utilizará *Firebase Invites* para invitar a un usuario a la aplicación.

RNF14	Notificaciones a los usuarios
Descripción	El sistema utilizará *Firebase Notifications* para el envío y recepción de notificaciones.

RNF15	Biblioteca de generación de códigos *QR*
Descripción	El sistema utilizará la biblioteca *ZXing* para la generación de códigos *QR*.

RNF16	Grupos de notificaciones
Descripción	Se utilizará la estrategia de envío de notificaciones por tema.

Otros aspectos

RNF17	*SDK* mínimo
Descripción	La aplicación se compilará con un *SDK* que soporte, como mínimo, la *API* 19 de *Android* (*Android* 4.4, *KitKat*) [2].

RNF18	Biblioteca de persistencia *"Room"*
Descripción	La aplicación utilizará la biblioteca de persistencia *Room* [3] como gestor de la base de datos local no gestionada por *Firebase*.

RNF19	Avatares y tamaño máximo
Descripción	Los avatares serán en formato imagen y de un tamaño máximo de 2MiB.

Dinámica del juego

RNF20	Niveles
Descripción	Un nivel solo podrá ser completado cuando todas las misiones del mismo hayan sido completadas.

Juegos

RNF21	Juegos
Descripción	Los juegos (misiones de tipo juego) estarán programados con tecnología Web.

RNF22	Reproducción de juegos
Descripción	Se utilizará un *Webview* integrado en la aplicación para la reproducción de los juegos.

RNF23	Recepción de la información
Descripción	Se utilizará un *bridge* Javascript con el fin de transferir los datos resultado del juego a la aplicación.

Logros

RNF24	Tipos de logros
Descripción	Existirán los siguientes tipos de logros: ▪ Logros asociados a una invitación (invitación del usuario a la aplicación). ▪ Logros asociados al hecho de completar una misión.

RNF25	Tipos de premios
Descripción	Existirán los siguientes tipos de premios: ▪ Premios de tipo descuento. ▪ Premios de tipo producto gratis. ▪ Premio de un viaje a Estados Unidos.

3.2.2. Casos de uso

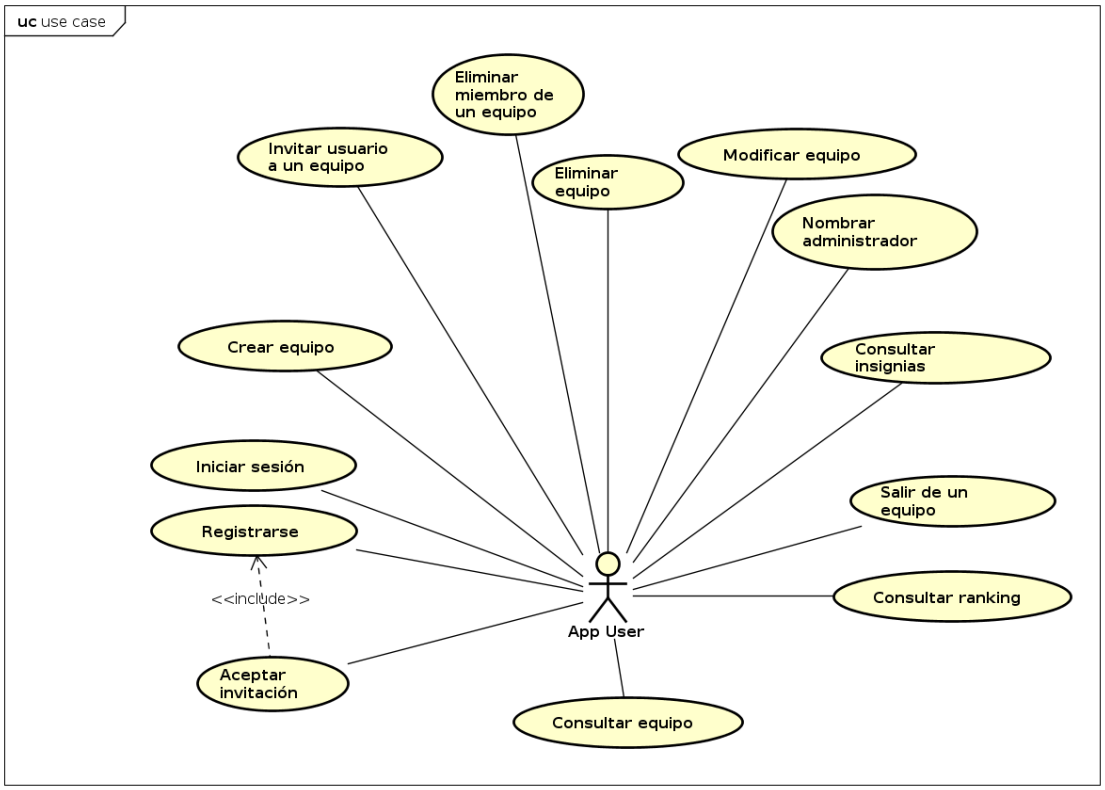


Figura 3.8: Diagrama de casos de uso. Parte I.

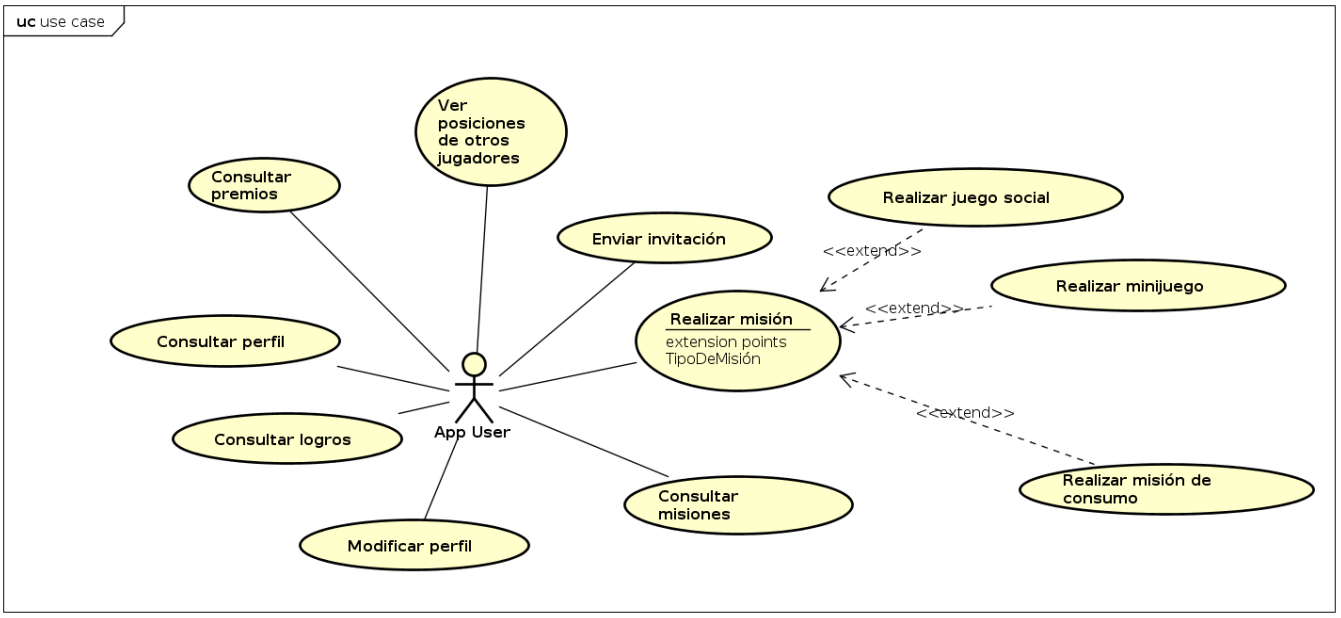


Figura 3.9: Diagrama de casos de uso. Parte II.

3.2.2.1. Actores

Se identifica un único actor:

- **App user:** Es el actor base de Route66App, en su parte de aplicación móvil. Realiza todas las funciones: crear grupos, gestionar aquellos en los que actúa como administrador, cumplir misiones, realizar consumos, etcétera.

3.2.2.2. Casos de uso

CU1	Modificar equipo
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	El equipo ha sido actualizado con la nueva configuración.
Flujo principal	1. El usuario selecciona “Gestionar un equipo“. 2. El sistema muestra al usuario el listado de equipos. 3. El usuario selecciona un equipo de la lista. 4. El sistema muestra el conjunto de parámetros de configuración del grupo, así como el valor de los mismos. 5. El usuario elige un parámetro y lo modifica. 6. El sistema actualiza la configuración del grupo.
Flujos alternativos	▪ 1a 3a 5a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 2a El sistema no encuentra ningún equipo administrado por el usuario. En ese caso, se muestra un mensaje y el caso de uso queda sin efecto. ▪ 6a El parámetro introducido por el usuario no es válido (porque se ha introducido en un formato incorrecto). En ese caso, se notifica al usuario y se vuelve al paso 5.

CU2	Invitar usuario al equipo
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	El usuario ha sido añadido al equipo.
Flujo principal	1. El usuario elige “Invitar usuario al equipo“. 2. El sistema muestra al usuario el listado de equipos que administra. 3. El usuario elige un equipo. 4. El sistema pregunta al usuario el nombre del usuario que desea añadir. 5. El usuario introduce el nombre. 6. El sistema comprueba el dato y añade al usuario al equipo.
Flujos alternativos	▪ 1a 3a 5a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 2a El usuario no administra ningún equipo. En ese caso, se notifica al usuario y el caso de uso queda sin efecto. ▪ 6a El nombre de usuario no se corresponde con un usuario de la aplicación. En ese caso se notifica al usuario y se vuelve al paso 5, indicando en el formulario el nombre que hubiera introducido el usuario con anterioridad.

CU3	Nombrar administrador de un equipo
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	El usuario seleccionado ha sido ascendido a administrador.
Flujo principal	1. El usuario elige “Nombrar administrador“. 2. El sistema muestra al usuario el listado de equipos que administra. 3. El usuario elige un equipo. 4. El sistema muestra al usuario el conjunto de miembros del equipo 5. El usuario elige un miembro. 6. El sistema actualiza la configuración de ese equipo.
Flujos alternativos	▪ 1a 3a 5a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 2a El usuario no administra ningún equipo. En ese caso, se notifica al usuario y el caso de uso queda sin efecto.

CU4	Eliminar miembro de un equipo
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	El usuario seleccionado ha sido expulsado del equipo.
Flujo principal	1. El usuario elige “Eliminar miembro del equipo“. 2. El sistema muestra al usuario el listado de equipos que administra. 3. El usuario elige un equipo. 4. El sistema muestra al usuario el conjunto de miembros del equipo 5. El usuario elige un miembro. 6. El sistema actualiza la configuración de ese equipo.
Flujos alternativos	▪ 1a 3a 5a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 2a El usuario no administra ningún equipo. En ese caso, se notifica al usuario y el caso de uso queda sin efecto.

CU5	Eliminar equipo
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	El equipo ha sido eliminado del sistema.
Flujo principal	1. El usuario elige “Eliminar equipo“. 2. El sistema muestra al usuario el listado de equipos que administra. 3. El usuario elige un equipo. 4. El sistema elimina el equipo
Flujos alternativos	▪ 1a 3a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 2a El usuario no administra ningún equipo. En ese caso, se notifica al usuario y el caso de uso queda sin efecto.

CU6	Consultar misiones
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El usuario elige “Consultar misiones“. 2. El sistema muestra al usuario el listado de niveles. 3. El usuario selecciona un nivel. 4. El sistema muestra al usuario las distintas misiones que conforman el nivel.
Flujos alternativos	▪ 3a. El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 3b. El usuario ha seleccionado un nivel distinto a aquel en el que se encuentra actualmente. En este caso, se muestra un mensaje al usuario y se vuelve al paso 2.

CU7	Realizar misión
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Se ha creado un logro asociado a la misión.
Flujo principal	1. El usuario elige “Realizar misión“. 2. El sistema muestra al usuario el conjunto de niveles con los que cuenta la aplicación. 3. El usuario escoge un nivel. 4. El sistema muestra al usuario las misiones de ese nivel. 5. El usuario escoge la misión que desea realizar 6. El sistema busca en la base de datos si hay registros de anteriores intentos de resolver la misión y muestra al usuario la misión (*punto de extensión TipoDeMisión*). 7. El usuario completa la misión y genera el logro asociado a la misma. 8. El sistema actualiza la información, guardando que el usuario ha completado la misión.
Flujos alternativos	▪ 1a 3a 5a 7a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 3b El usuario no selecciona el nivel que le corresponde completar (el nivel más bajo al que le falte por completar misiones). En ese caso, el sistema muestra un mensaje y vuelve al paso 2. ▪ 6a El usuario ha escogido una misión (*punto de extensión TipoDeMisión*) de tipo *“juego social“*. En este caso, se acude al caso de uso “Realizar juego social“, pasando a este caso de uso el identificador de la misión que se va a realizar. ▪ 6b El usuario ha escogido una misión (*punto de extensión TipoDeMisión*) de tipo *“minijuego“*. En este caso, se acude al caso de uso “Realizar minijuego“, pasando a este caso de uso el identificador de la misión que se va a realizar. ▪ 6c El usuario ha escogido una misión (*punto de extensión TipoDeMisión*) de tipo *“consumo“*. En este caso, se acude al caso de uso “Realizar misión de consumo“, pasando a este caso de uso el identificador de la misión que se va a realizar. ▪ 7b El usuario no completa la misión. En ese caso, se pregunta al usuario si desea volver a intentar

CU8	Realizar misión de consumo mínimo
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El sistema obtiene los detalles de la misión, a partir del identificador de misión que se ha proporcionado. 2. El sistema muestra al usuario el consumo que debe realizar para marcar la misión como completada. Igualmente, también muestra el código QR asociado a esta misión de consumo mínimo. 3. El sistema comprueba si existe algún registro de consumo asociado al usuario que tenga un importe, como mínimo, el que exige la misión para ser completada. Este registro deberá haberse registrado con fecha posterior a la hora de entrada por primera vez a la actividad.
Flujos alternativos	▪ 3a No existe ningún registro de consumo que cumpla tales caracterísiticas, en ese caso, se informa al usuario y el caso de uso quedaría sin efecto.

CU9	Realizar minijuego.
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El sistema obtiene los detalles de la misión, a partir del identificador de misión que se ha proporcionado. 2. El sistema muestra al usuario el minijuego asociado a la misión. 3. El sistema introduce un escuchador detrás del navegador donde se muestra el juego para obtener los resultados del mismo. 4. El sistema da el control del mismo al juego. 5. El sistema recibe los resultados del juego por medio del escuchador. 6. El sistema comprueba que los resultados del juego estén dentro del intervalo de lo pedido para la misión.
Flujos alternativos	▪ 2a 4a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 6a Los parámetros recibidos no se corresponden con los valores que se exigen para completar la misión. En ese caso, se muestra un mensaje al usuario, se envía un mensaje de reporte al servidor y se pregunta al usuario si desea volver a intentar la misión. En caso afirmativo se volvería al paso 2. En caso negativo, el caso de uso quedaría sin efecto.

CU10	Realizar juego social.
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El sistema obtiene los detalles de la misión, a partir del identificador de misión que se ha proporcionado. 2. El sistema muestra al usuario el reto social que debe realizar para marcar la misión como completada. 3. El sistema comprueba si existe algún registro de esta misión social asociado al usuario. Este registro deberá haberse registrado con fecha posterior a la hora de entrada, por primera vez a la actividad.
Flujos alternativos	▪ 2a 3a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 3a No existe ningún registro de consumo que cumpla tales caracterísiticas, en ese caso, se informa al usuario.

CU11	Modificar perfil
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	El perfil ha sido actualizado.
Flujo principal	1. El usuario elige “Modificar perfil“. 2. El sistema muestra al usuario el conjunto de parámetros que puede modificar y los valores que tiene asociados. 3. El usuario elige un parámetro y lo modifica. 4. El sistema actualiza el nuevo valor.
Flujos alternativos	▪ 1a 3a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 4a El parámetro introducido por el usuario no es válido (porque se ha introducido en un formato incorrecto). En ese caso, se notifica al usuario y se vuelve al paso 2.

CU12	Consultar perfil
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El usuario elige “Consultar perfil“. 2. El sistema muestra al usuario su perfil, incluidos aquellos detalles como sus “porcentajes de compromiso“ con la aplicación.
Flujos alternativos	Ninguno.

CU13	Consultar logros
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El usuario elige “Consultar mis logros“. 2. El sistema muestra al usuario los logros realizados, ordenados por puntuación obtenida.
Flujos alternativos	Ninguno.

CU14	Consultar insignias
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El usuario elige “Consultar mis insignias“. 2. El sistema muestra al usuario las insignias obtenidas, ordenadas por fecha.
Flujos alternativos	Ninguno.

CU15	Consultar ranking
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El usuario elige “Consultar ranking“. 2. El sistema muestra al usuario el ranking global de usuarios, ordenados por la puntuación global obtenida.
Flujos alternativos	Ninguno.

CU16	Consultar equipo
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El usuario elige “Consultar equipo“. 2. El sistema muestra al usuario el listado de equipos en los que es miembro. 3. El usuario selecciona un equipo. 4. El sistema muestra al usuario los detalles de ese equipo (miembros, nombre, avatar, logros y premios).
Flujos alternativos	▪ 1a 3a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 2a El sistema no encuentra ningún equipo en el que se encuentre el miembro. En ese caso, se muestra un mensaje al usuario y el caso de uso queda sin efecto.

CU17	Crear equipo
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Se ha creado un nuevo equipo. El usuario es asociado como administrador del mismo.
Flujo principal	1. El usuario elige “Crear equipo“. 2. El sistema pregunta al usuario el nombre del equipo. 3. El usuario introduce el dato solicitado. 4. El sistema crea el equipo.
Flujos alternativos	▪ 1a 3a 5a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto.

CU18	Registrarse
Actor	App user.
Precondiciones	Ninguna.
Postcondiciones	Se ha creado un nuevo usuario en el sistema.
Flujo principal	1. El usuario elige “Registrarse“. 2. El sistema pregunta al usuario cuál es su correo electrónico y su contraseña. 3. El usuario introduce la información. 4. El sistema comprueba la información y crea el usuario.
Flujos alternativos	▪ 1a 3a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 4a El sistema identifica un usuario ya registrado con el correo electrónico proporcionado. En ese caso, se le notifica y el caso de uso queda sin efecto.

CU19	Aceptar invitación.
Actor	App user.
Precondiciones	Existe una conexión estable a Internet.
Postcondiciones	▪ Se ha creado un usuario nuevo. ▪ Se ha generado un logro de tipo invitación.
Nota	Este caso de uso depende del componente *Firebase Invites*.
Flujo principal	1. El sistema recibe una llamada de *Firebase Invites* que le proporciona el identificador del usuario que ha enviado la invitación. 2. Se pasa al caso de uso “Registrarse“, junto con el dato del usuario que ha enviado la invitación. 3. El sistema envía una solicitud *HTTP* contra un *endpoint* que habilita la función *lambda* que añade los puntos y notifica al usuario que ha generado la invitación.
Flujos alternativos	▪ 1a 2a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto.

CU20	Enviar invitación
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El usuario elige “Enviar invitación“. 2. El sistema pregunta al usuario qué mensaje desea añadir a la invitación. 3. El usuario introduce el mensaje. 4. El sistema comprueba el mensaje y muestra al usuario el cuadro de diálogo del sistema para enviar la invitación. 5. El usuario comparte la invitación por correo electrónico.
Flujos alternativos	▪ 1a 2a 3a 4a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto.

CU21	Iniciar sesión
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Se ha generado una nueva sesión en el sistema.
Flujo principal	1. El usuario elige “Iniciar sesión“. 2. El sistema pregunta al usuario el correo electrónico, así como su contraseña. 3. El usuario introduce la información solicitada. 4. El sistema identifica al usuario.
Flujos alternativos	▪ 1a 3a El usuario cancela la operación. En este caso, el caso de uso termina y queda sin efecto. ▪ 4a El usuario ha introducido un correo electrónico de un usuario no perteneciente al sistema. En este caso, se notifica al usuario y se vuelve al paso 2. ▪ 4b El usuario ha introducido una contraseña distinta a la que tiene el usuario. En este caso, se notifica al usuario y se vuelve al paso 2. ▪ 4c El usuario ha introducido un correo electrónico que no se ajusta al formato. En ese caso se notifica al usuario y se vuelve al paso 2.

CU22	Ver posiciones de otros jugadores
Actor	App user
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El usuario elige un nivel. 2. El sistema muestra al usuario las distintas posiciones de los miembros que se encuentren en ese nivel.
Flujos alternativos	Ninguno.

CU23	Consultar premios
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	Ninguna.
Flujo principal	1. El usuario elige “Consultar mis premios“. 2. El sistema muestra al usuario los premios obtenidos que sigan estando vigentes, ordenados por fecha de caducidad.
Flujos alternativos	Ninguno.

CU24	Salir de un equipo
Actor	App user.
Precondiciones	El usuario ha iniciado sesión en el sistema.
Postcondiciones	El usuario ha salido del equipo.
Flujo principal	1. El usuario elige “Salir de un equipo“. 2. El sistema muestra al usuario el conjunto de grupos a los que pertenece. 3. El usuario escoge un grupo. 4. El sistema comprueba la posibilidad de que el usuario abandone el equipo. 5. El sistema elimina al usuario del equipo.
Flujos alternativos	▪ 1a 3a El usuario cancela la operación. En este caso, el caso de uso se cancela y queda sin efecto. ▪ 4a El usuario es administrador del grupo escogido. En ese caso se notifica y se vuelve al paso 2.

3.2.3. Modelo de dominio

El modelo de dominio se ha decidido dividirlo en dos partes: aquella que interacciona con la base de datos de *Firebase Firestore* y aquella que solo lo hace con la local (*SQLite*).

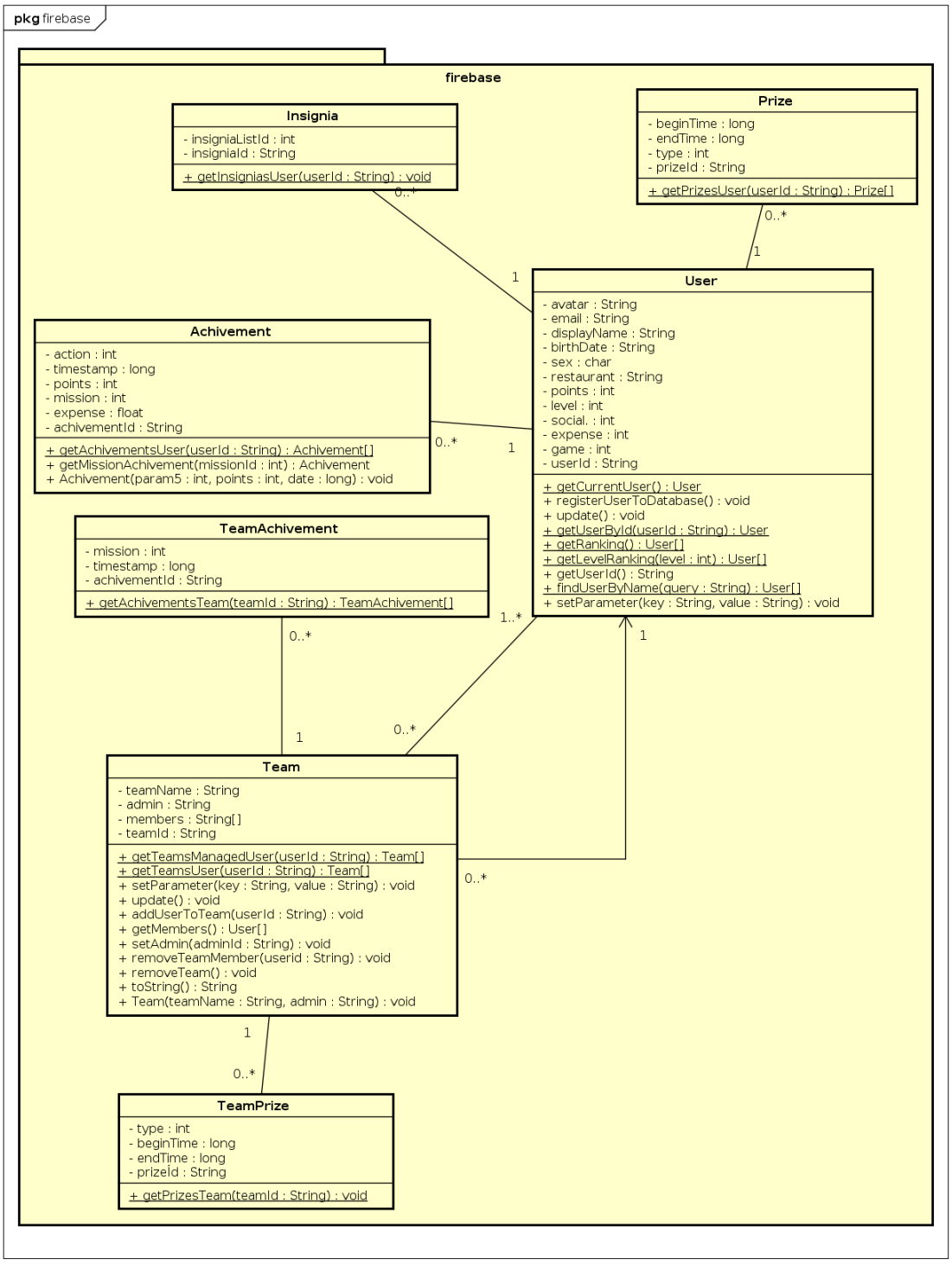


Figura 3.10: Modelo de dominio. Parte que interacciona con *Firebase*.

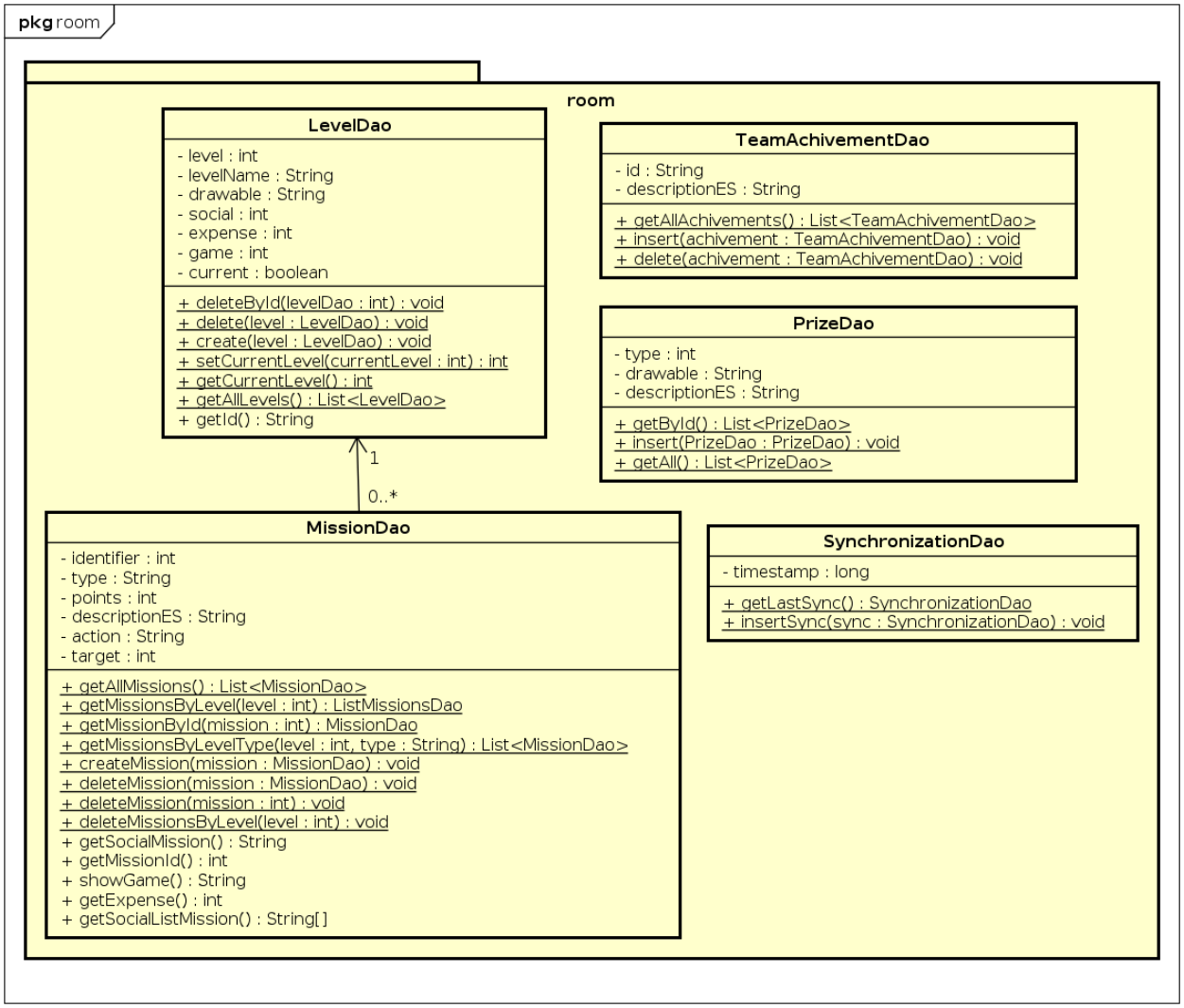


Figura 3.11: Modelo de dominio. Parte que interacciona con la base de datos local.

3.2.4. Diagramas de secuencia

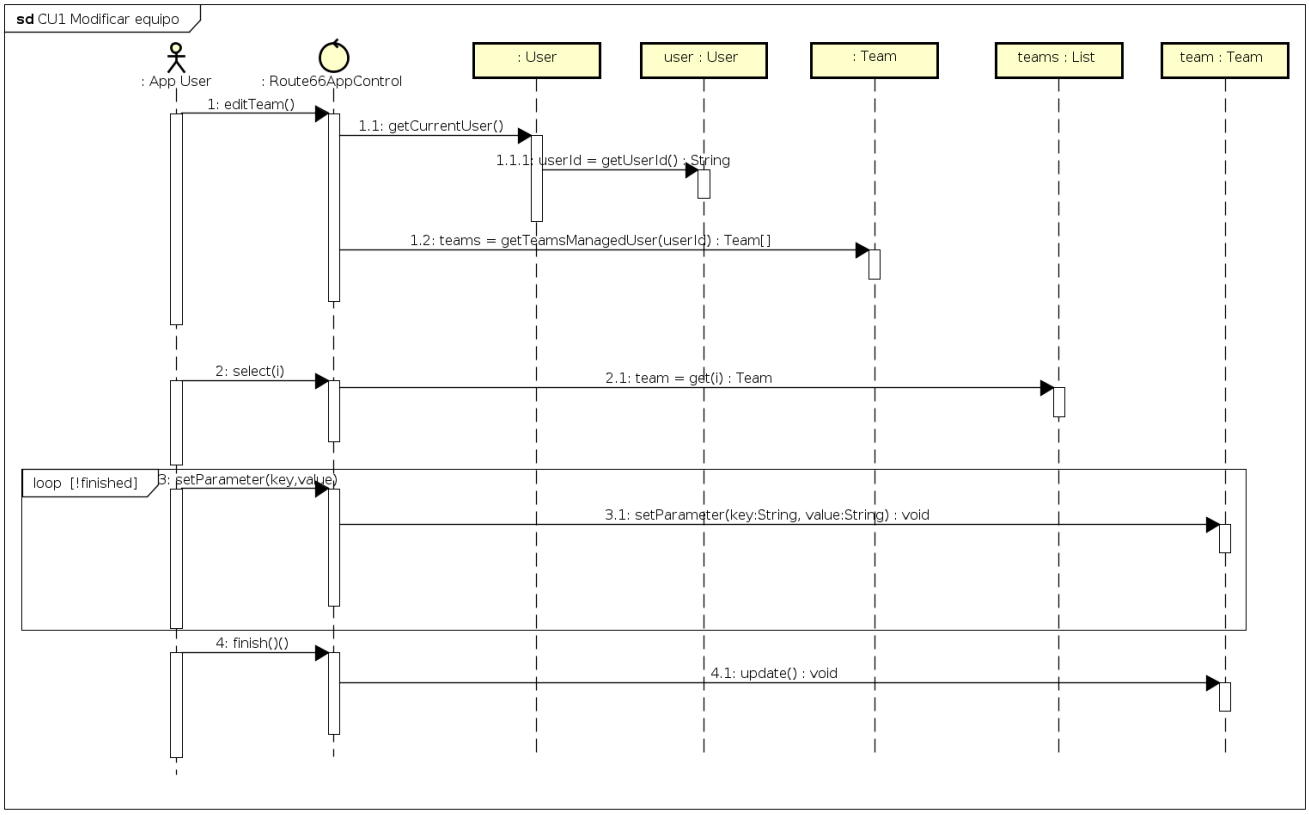


Figura 3.12: Diagrama de secuencia 1 “Modificar equipo“.

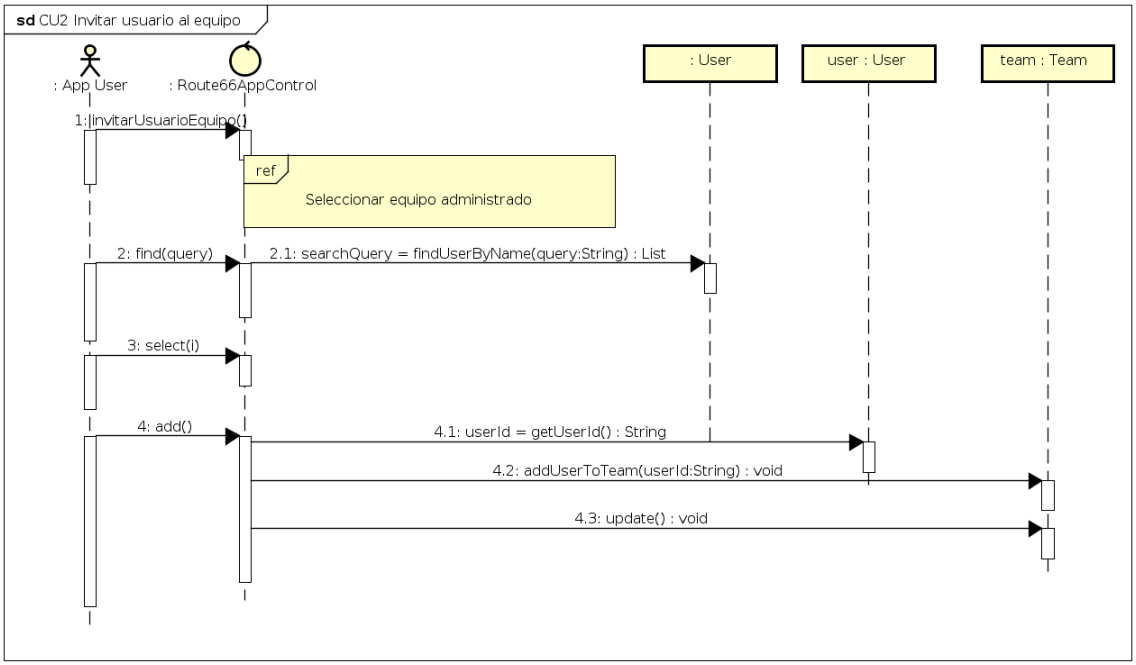


Figura 3.13: Diagrama de secuencia del caso de uso 2 “Invitar usuario al equipo“.

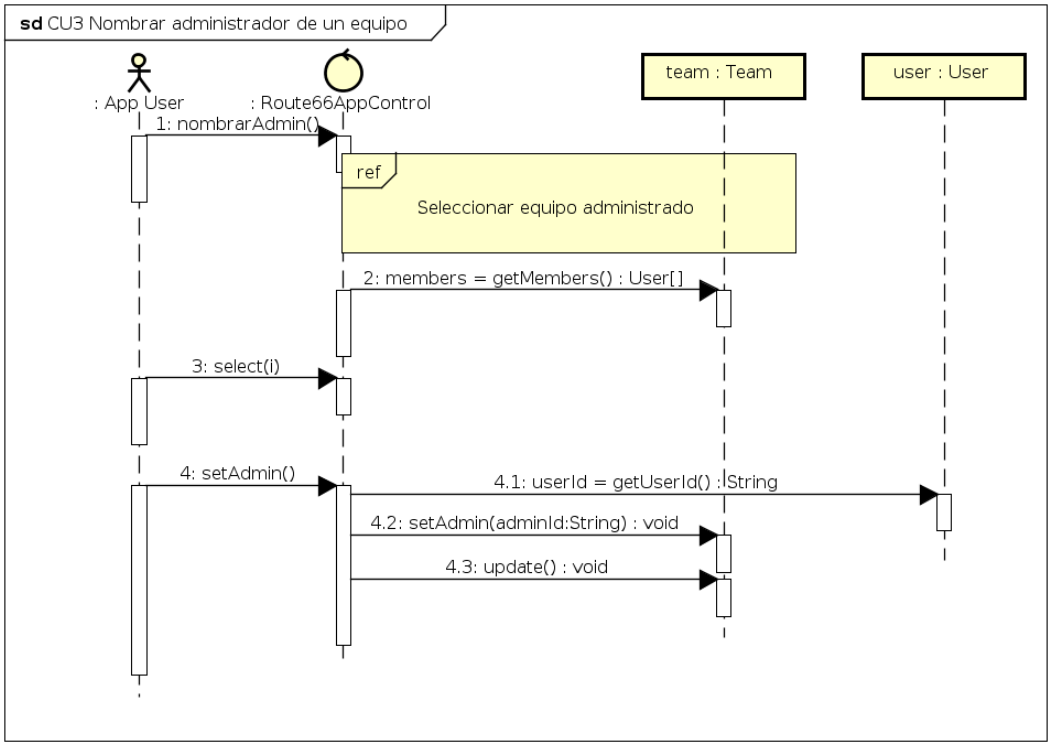


Figura 3.14: Diagrama de secuencia del caso de uso 3 “Nombrar un administrador“.

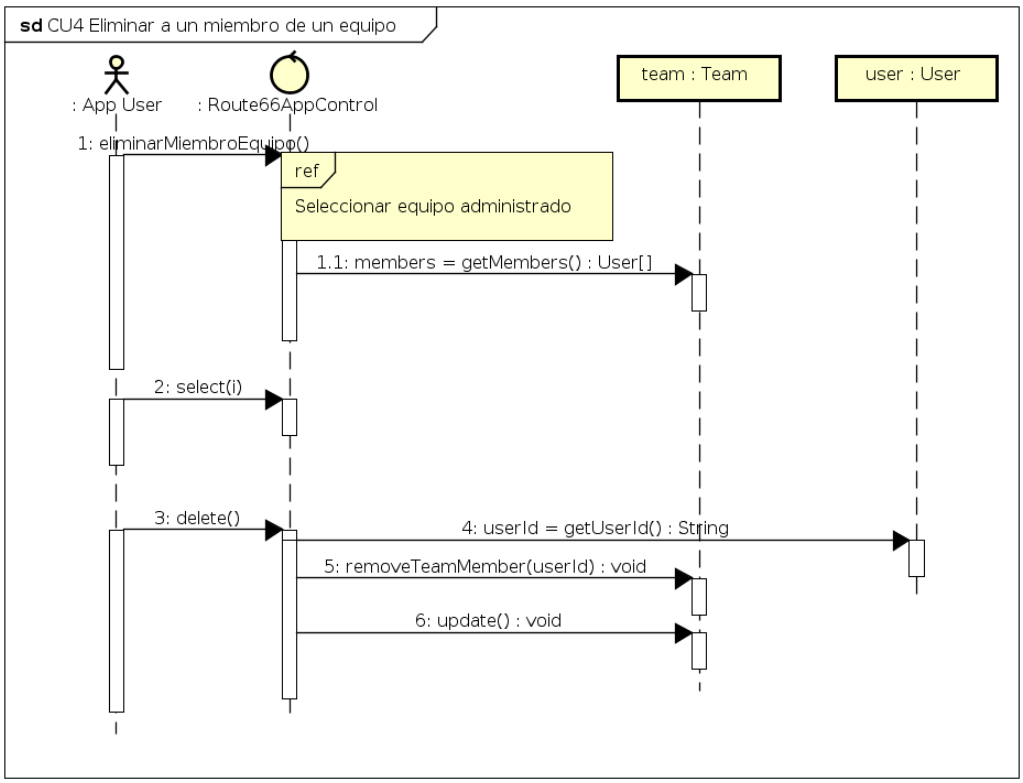


Figura 3.15: Diagrama de secuencia del caso de uso 4 “Eliminar a un miembro de un equipo“.

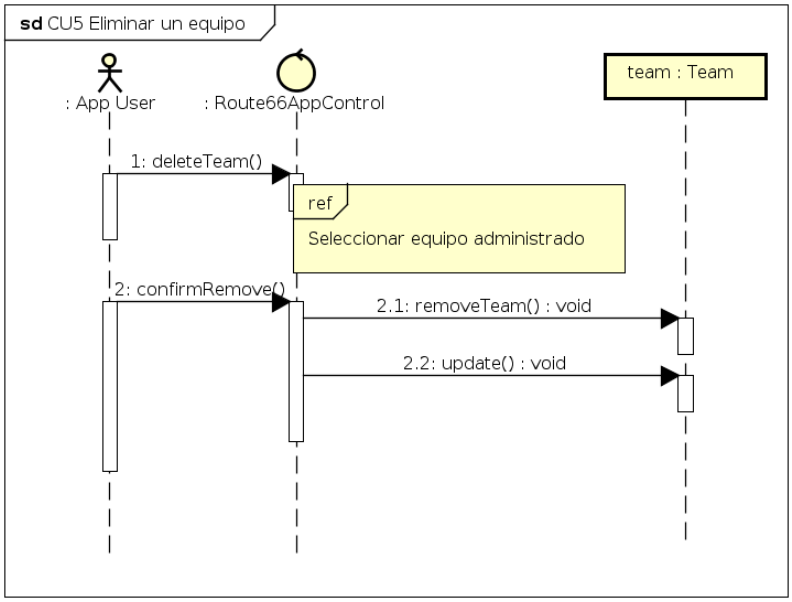


Figura 3.16: Diagrama de secuencia del caso de uso 5 “Eliminar un equipo“.

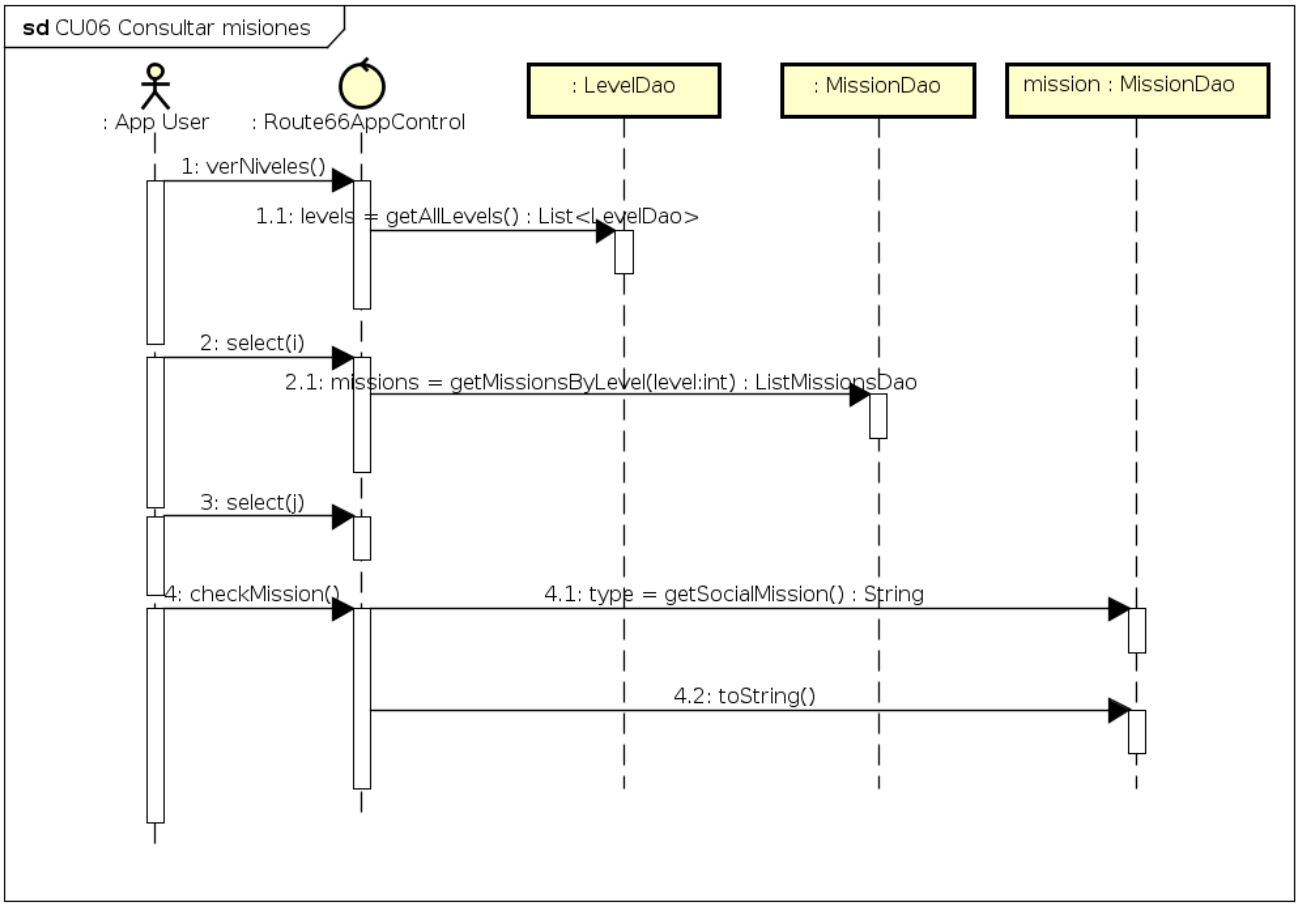


Figura 3.17: Diagrama de secuencia del caso de uso 6 “Consultar misiones“.

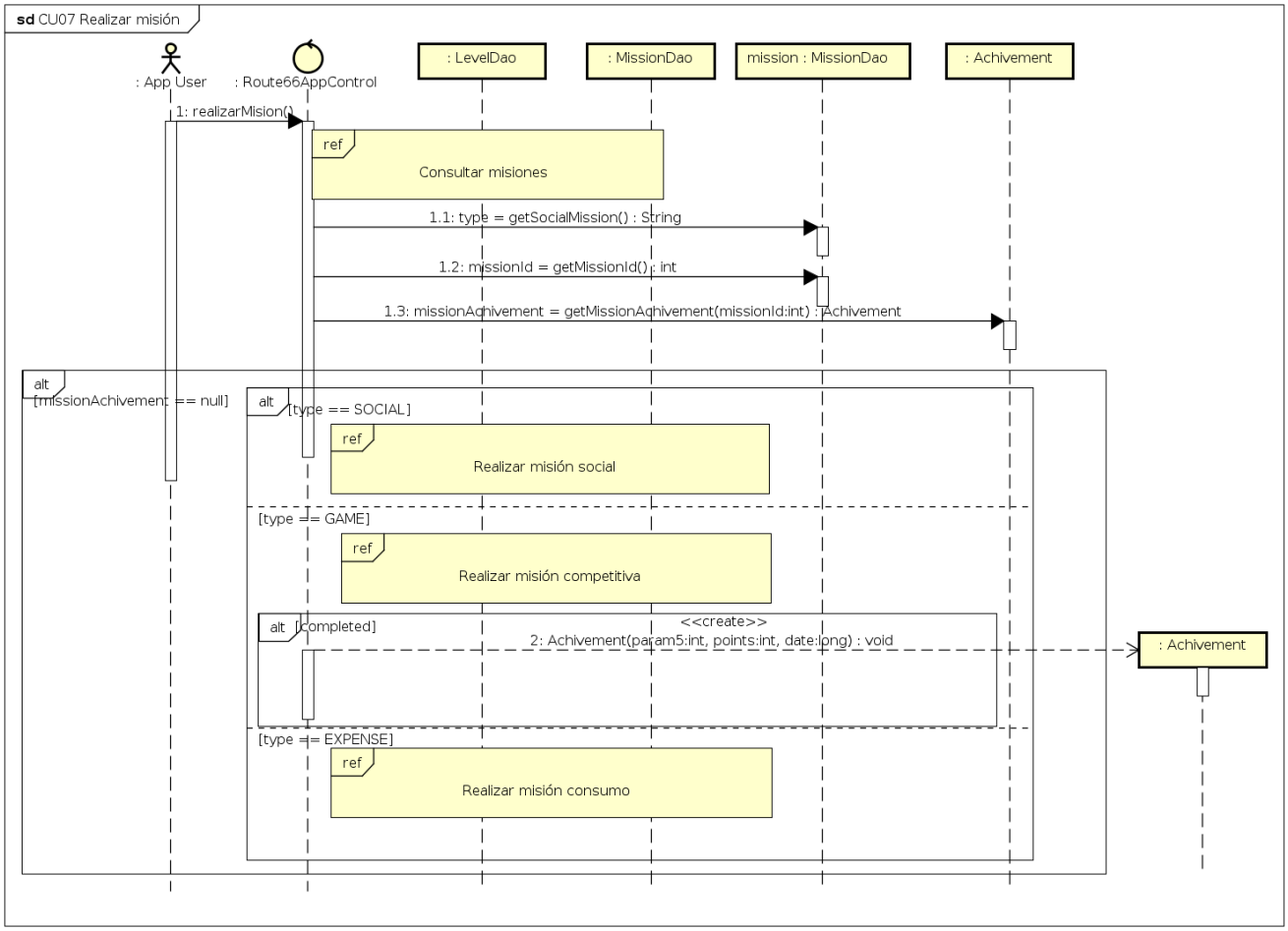


Figura 3.18: Diagrama de secuencia del caso de uso 7 “Realizar misión“.

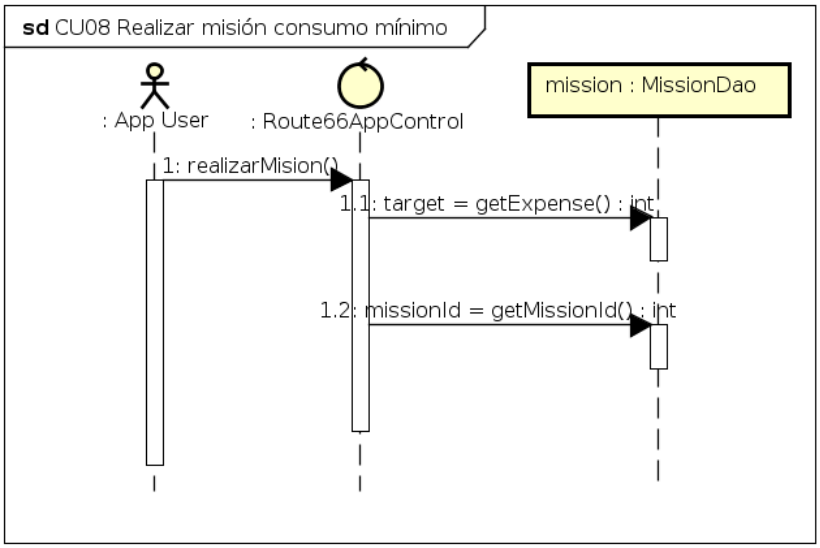


Figura 3.19: Diagrama de secuencia del caso de uso 8 “Realizar misión de consumo mínimo“.

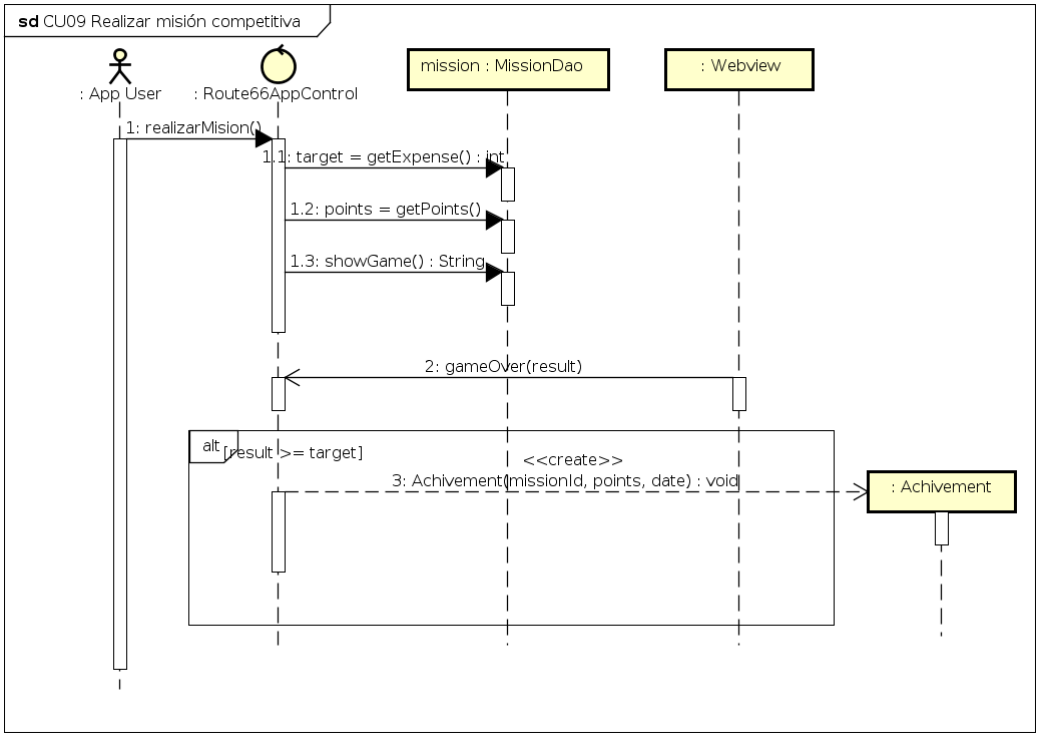


Figura 3.20: Diagrama de secuencia del caso de uso 9 “Realizar misión competitiva“.

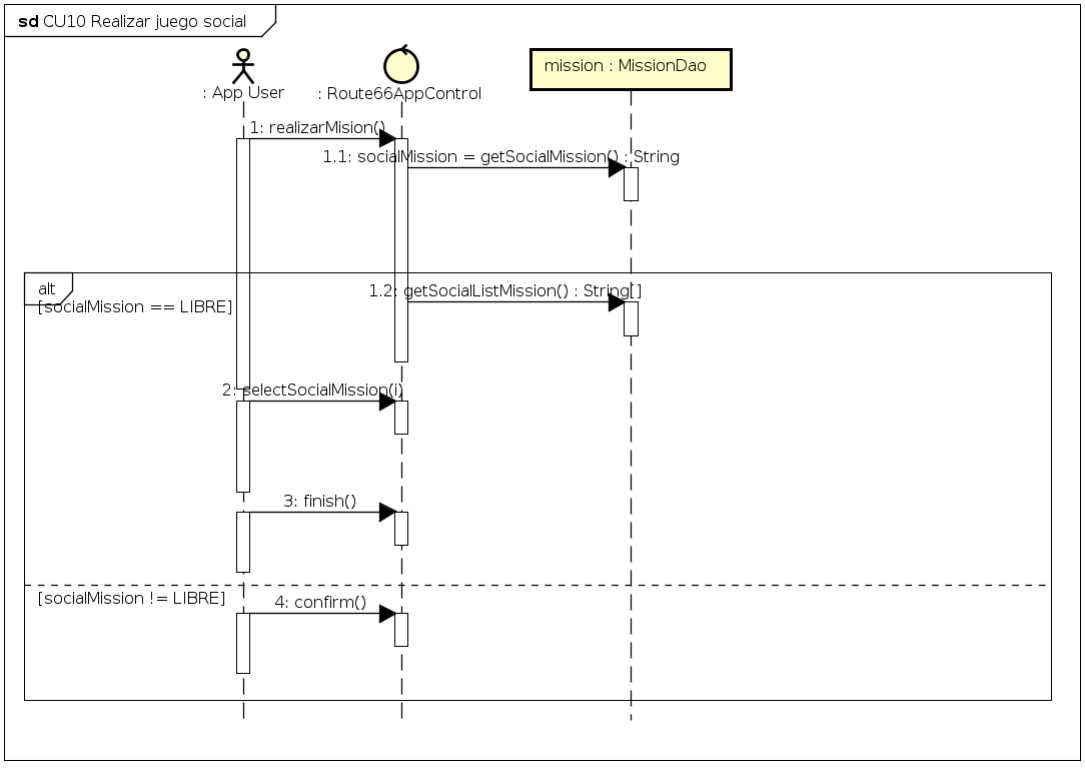


Figura 3.21: Diagrama de secuencia del caso de uso 10 “Realizar juego social“.

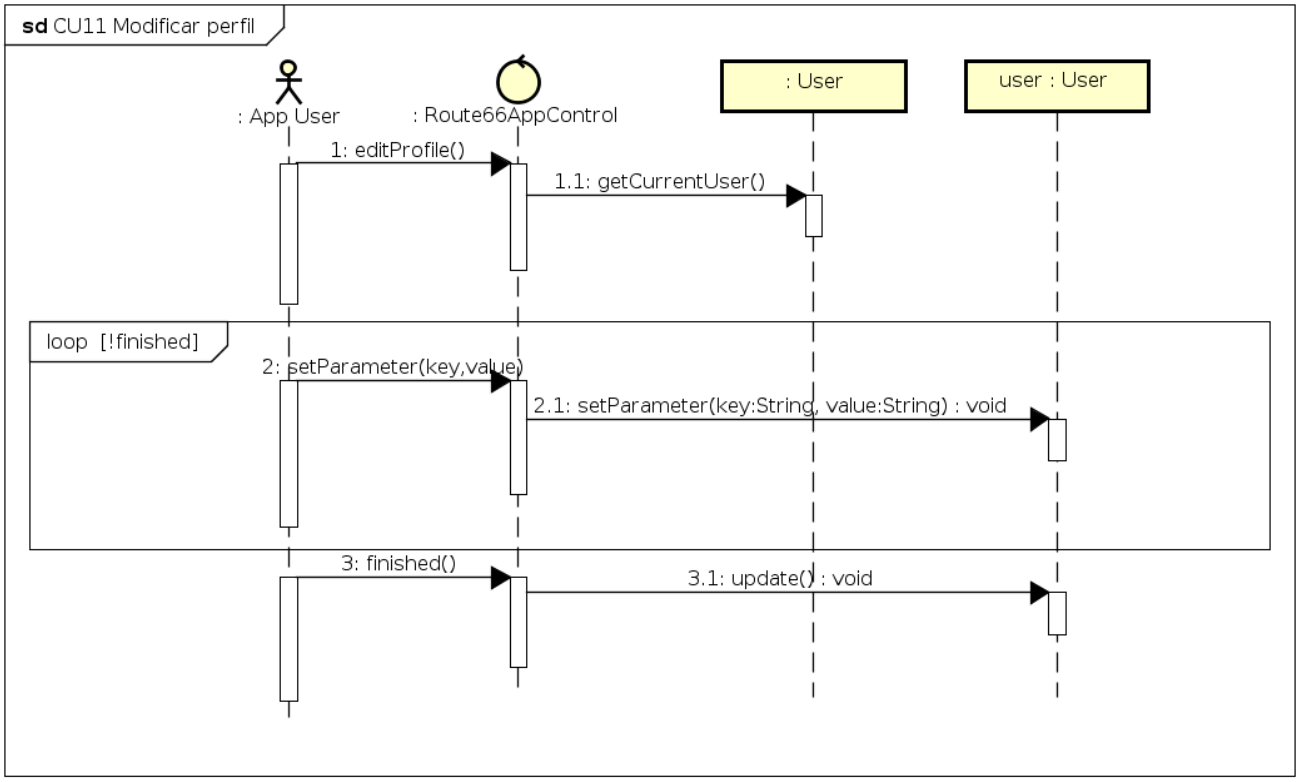


Figura 3.22: Diagrama de secuencia del caso de uso 11 “Modificar perfil“.

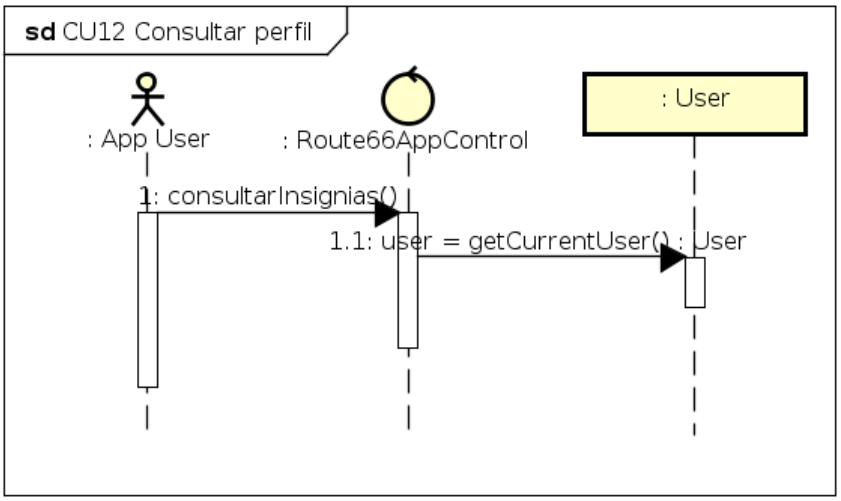


Figura 3.23: Diagrama de secuencia del caso de uso 12 “Consultar perfil“.

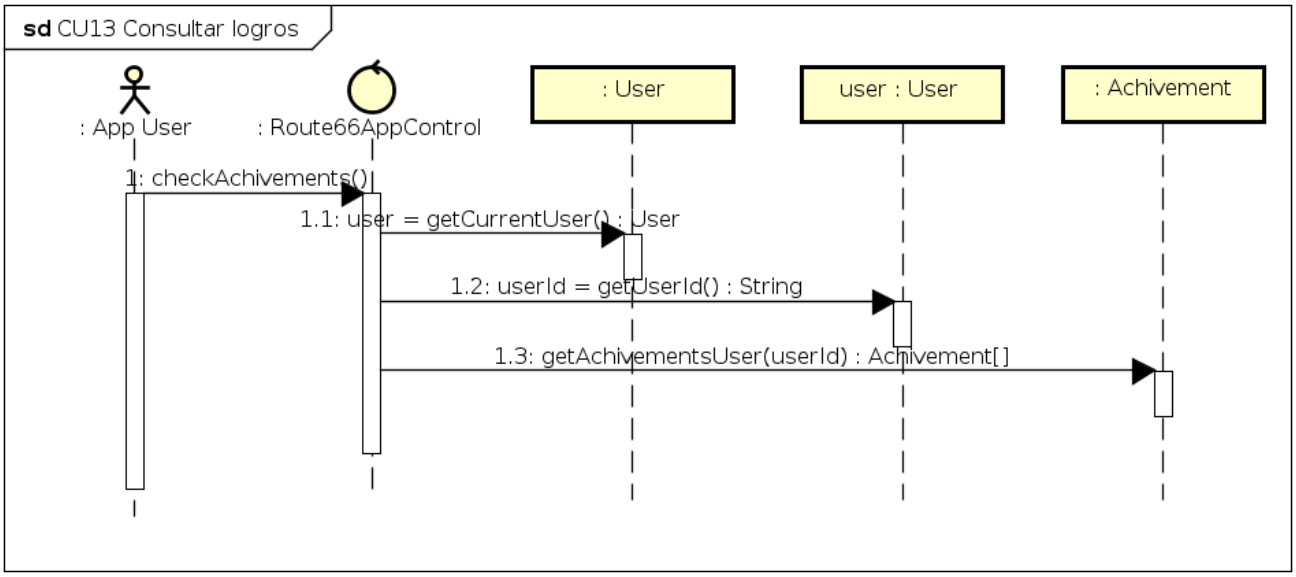


Figura 3.24: Diagrama de secuencia del caso de uso 13 “Consultar logros“.

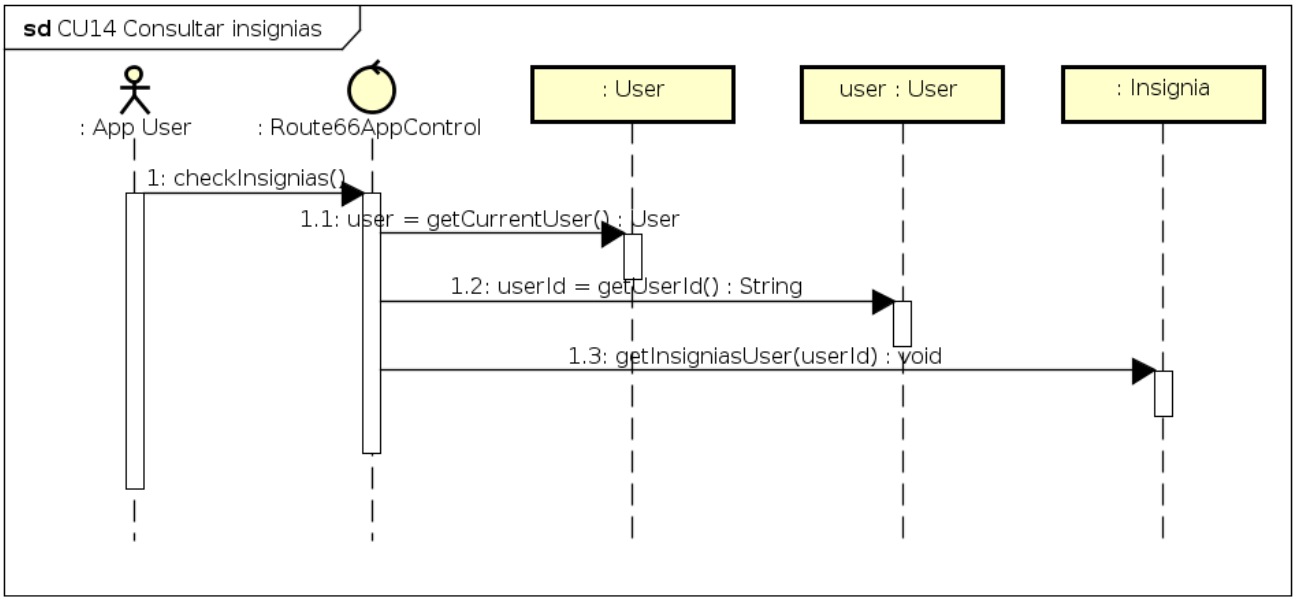


Figura 3.25: Diagrama de secuencia del caso de uso 14 “Consultar insignias“.

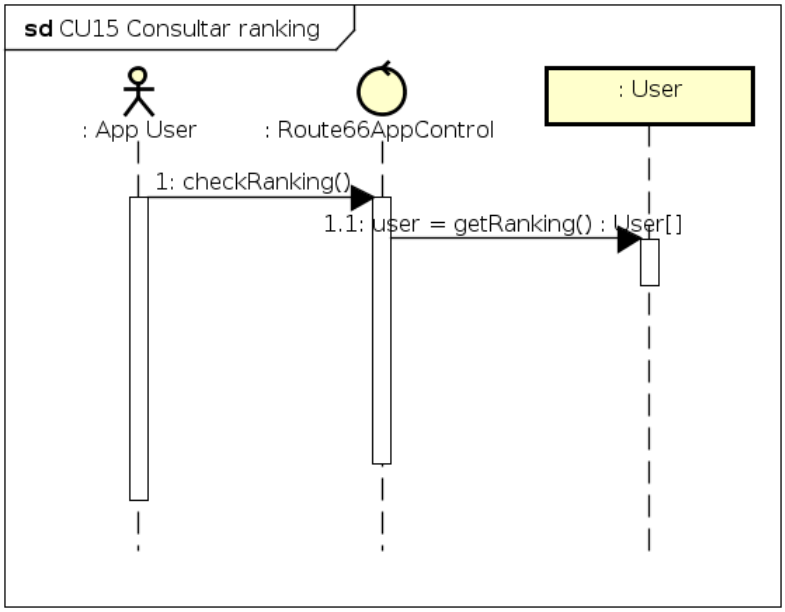


Figura 3.26: Diagrama de secuencia del caso de uso 15 “Consultar ranking“.

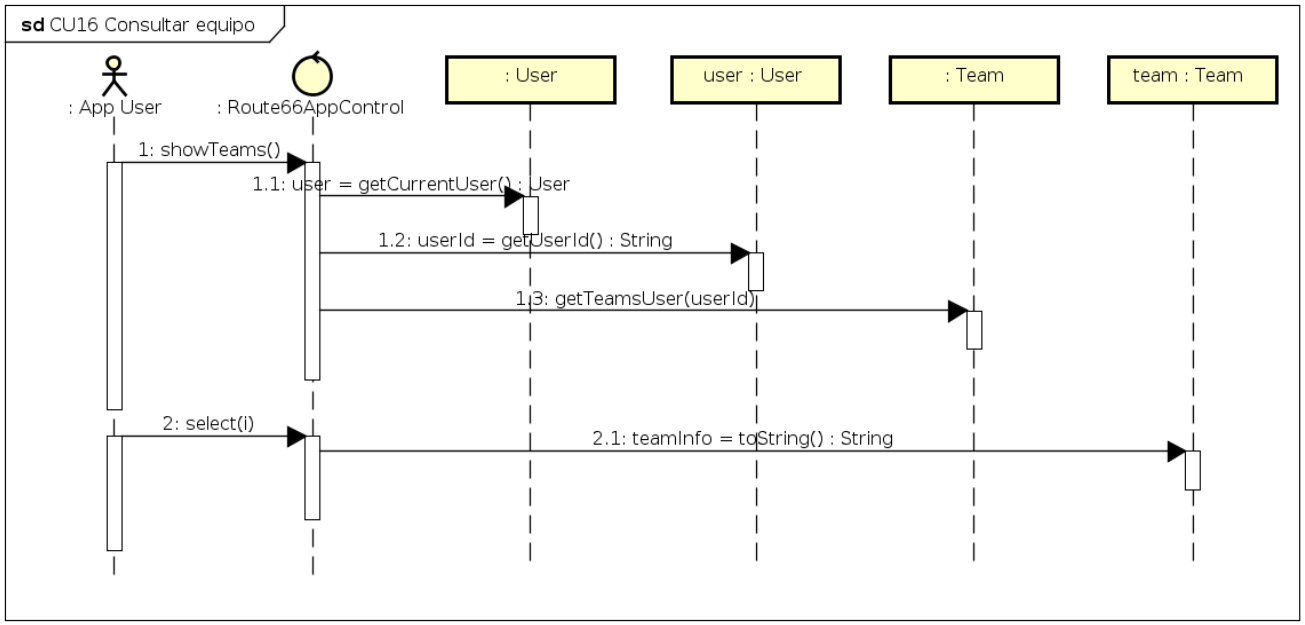


Figura 3.27: Diagrama de secuencia del caso de uso 16 “Consultar equipo“.

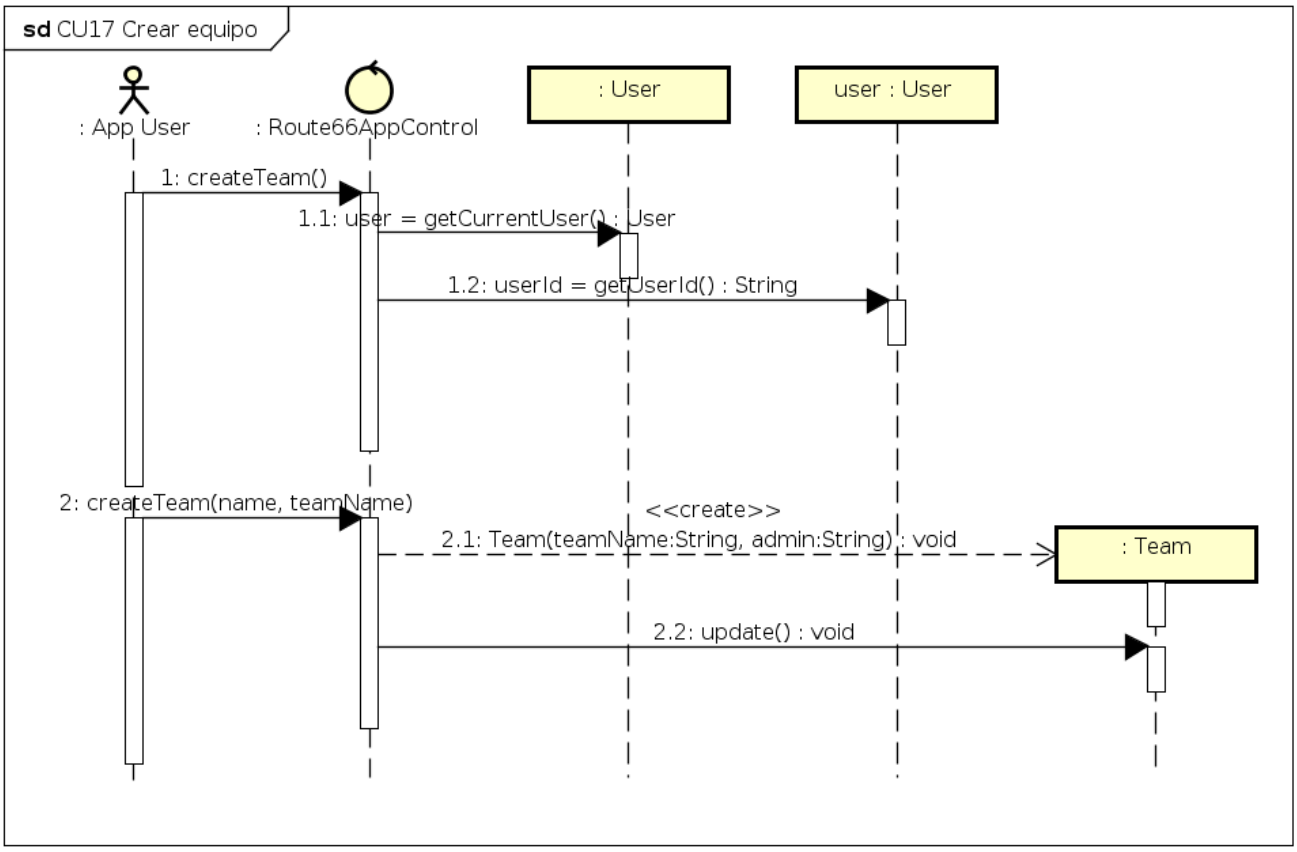


Figura 3.28: Diagrama de secuencia del caso de uso 17 “Crear equipo“.

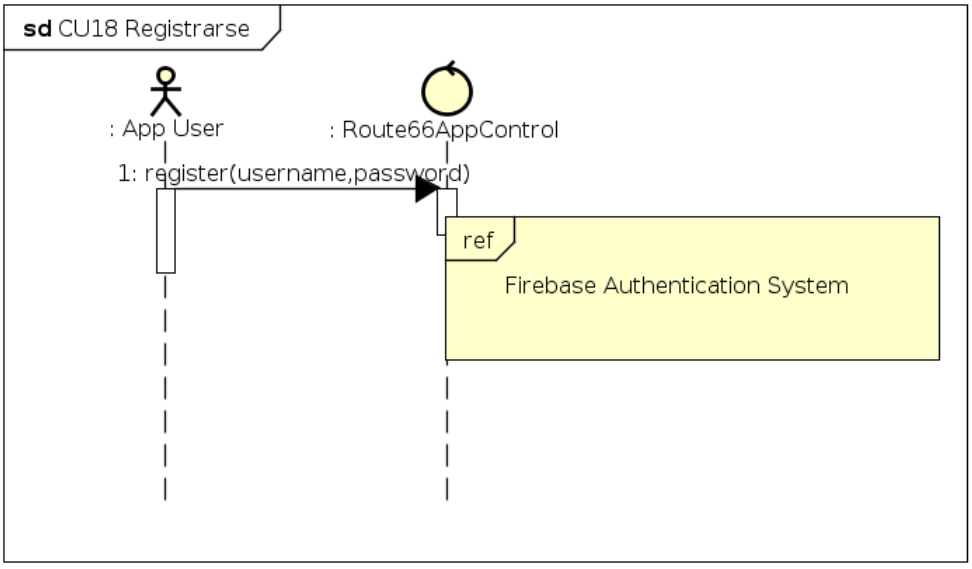


Figura 3.29: Diagrama de secuencia del caso de uso 18 “Registrarse“.

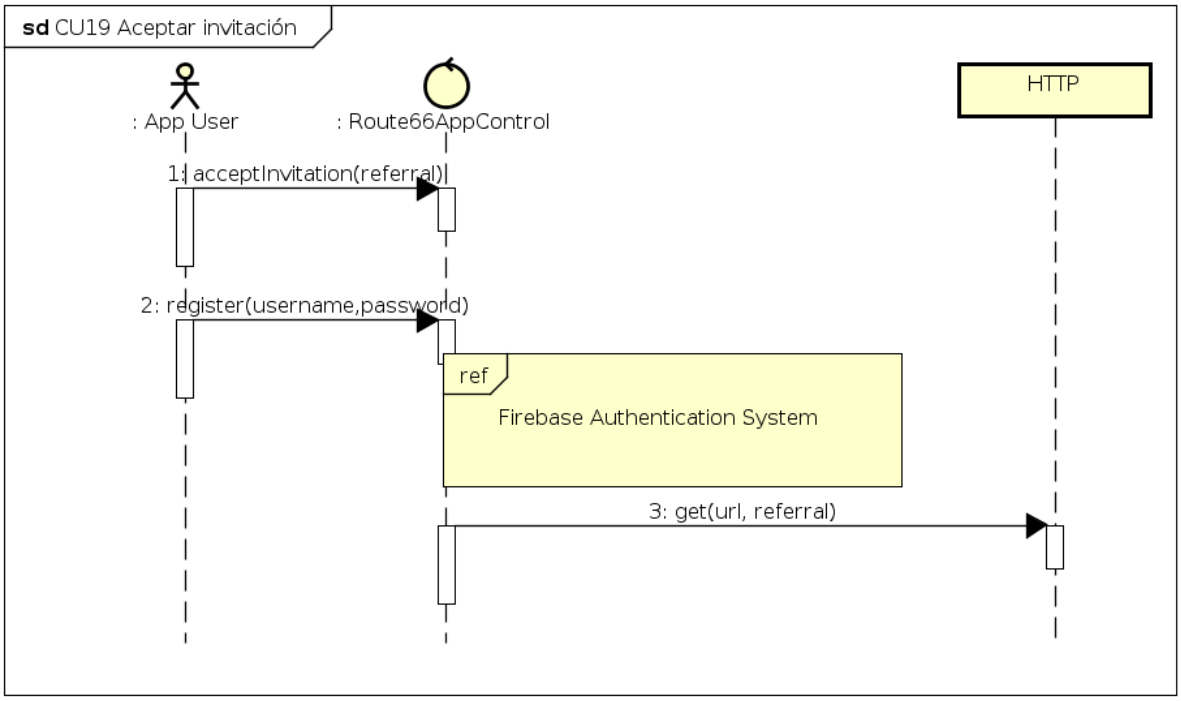


Figura 3.30: Diagrama de secuencia del caso de uso 19 “Aceptar invitación“.

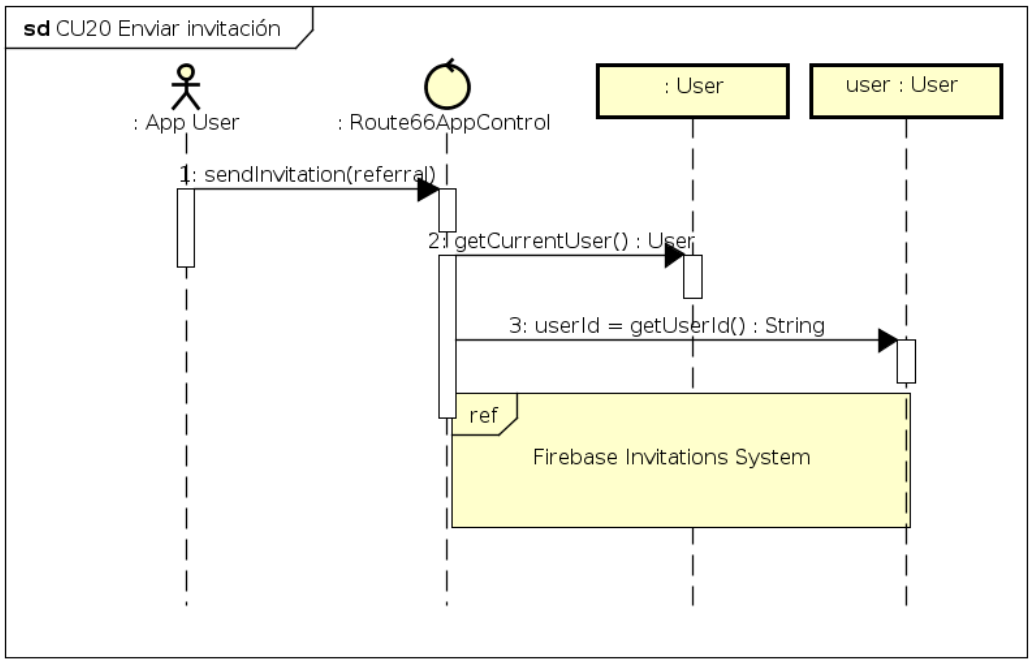


Figura 3.31: Diagrama de secuencia del caso de uso 20 “Enviar invitación“.

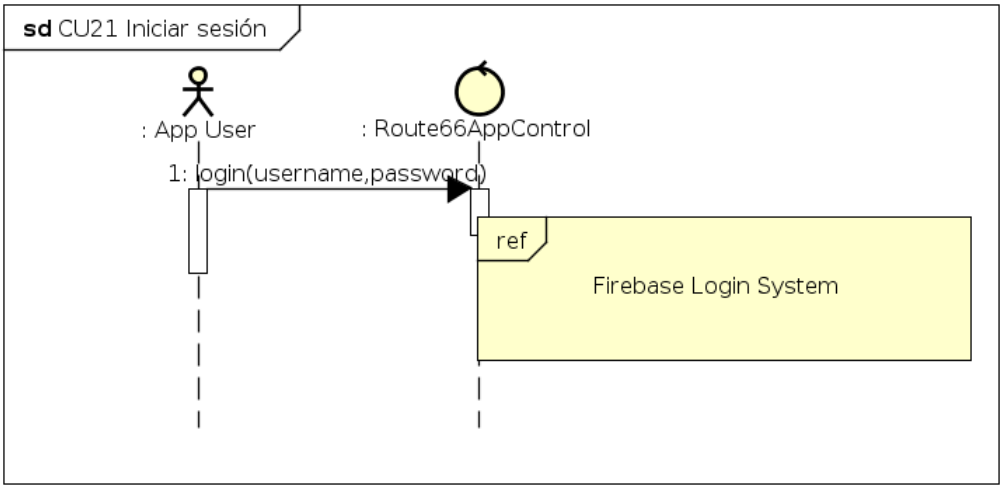


Figura 3.32: Diagrama de secuencia del caso de uso 21 “Iniciar sesión“.

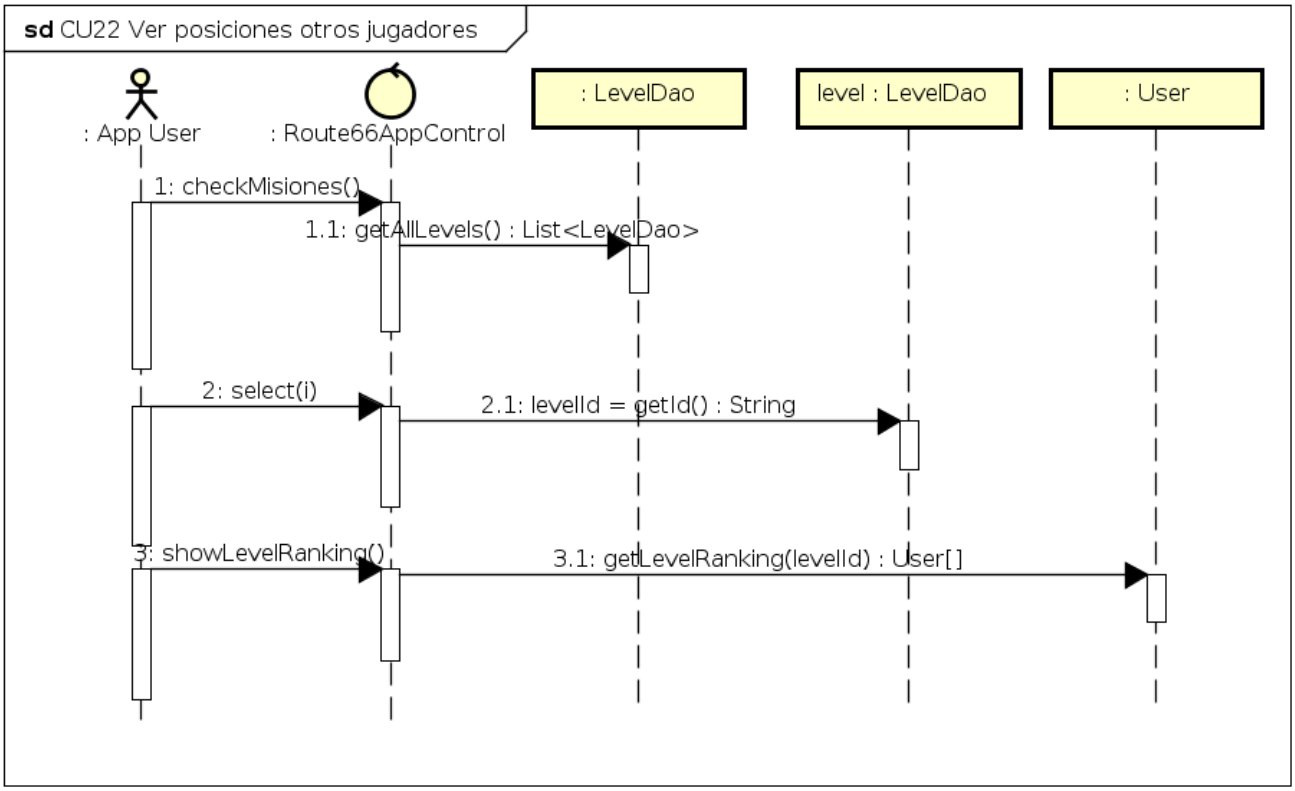


Figura 3.33: Diagrama de secuencia del caso de uso 22 “Ver posiciones de otros jugadores“.

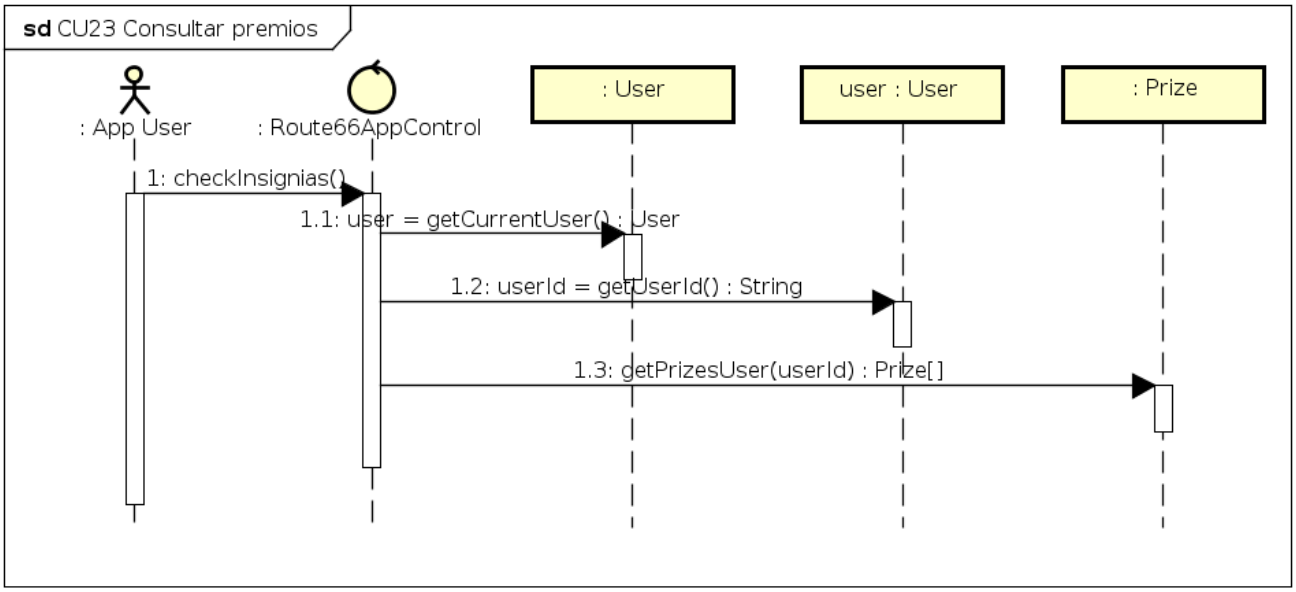


Figura 3.34: Diagrama de secuencia del caso de uso 23 “Consultar premios“.

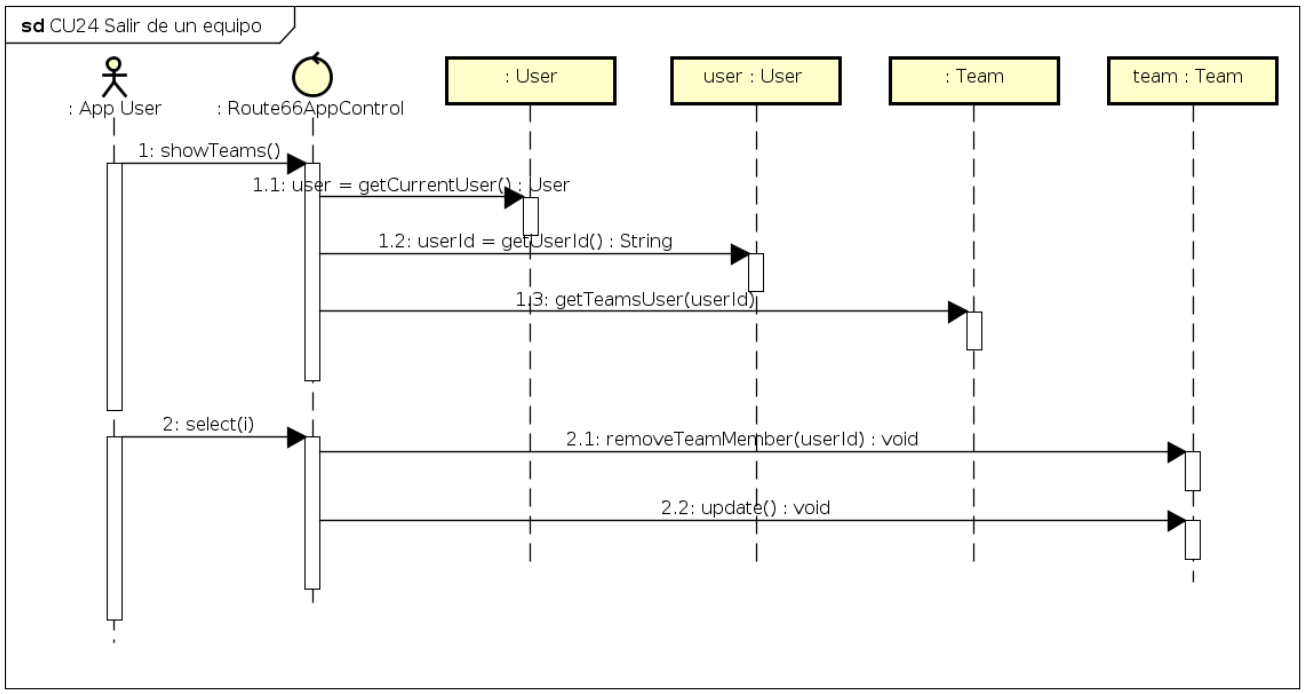


Figura 3.35: Diagrama de secuencia del caso de uso 24 “Salir de un equipo“.

3.2.5. Usabilidad

Para el diseño de la interfaz gráfica de Route66App (iconos, *widgets* utilizados y disposiciones de elementos) se ha tenido siempre encuenta el estándar de diseño de aplicaciones para la plataforma *Android*, *Material Design* [31] [32]. El motivo de todo esto ha sido que los usuarios están más acostumbrados a interactuar con interfaces que aplican este estándar, por lo que la curva de aprendizaje será siempre mucho menor. Se han aplicado los siguientes **atributos de usabilidad**:

- Facilidad de recuerdo: ayudar al usuario a recordar cómo funciona el sistema informático.
- Facilidad de aprendizaje: el usuario puede deducir cómo se usa el sistema explorando la interfaz y probando las diferentes acciones.
- Gestión de errores: evitar los posibles errores al introducir datos en la interfaz, o bien, se informa de ello.

Por otro lado, también se ha basado el diseño en determinados **patrones visuales**:

1. ***Navigation Drawer***: es un patrón muy típico en *Android*. Consiste en un menú desplegable que se encuentra en la parte izquierda de la aplicación.
2. **Notificaciones**: avisos sonoros y visuales que se muestran de forma asíncrona en la bandeja de notificaciones del sistema.
3. **Preferencias**: recoger las preferencias en lugares habilitados para las mismas.
4. ***Cards***: visualización de información muy concreta y muy relacionada en un lugar determinado y delimitado de la pantalla.
5. **Secciones de navegación**: etiquetas muy concretas que indican en qué sección se encuentra el usuario y la información que puede encontrar en la misma.

3.3. Diseño

3.3.1. Arquitectura

La aplicación se estructura, a alto nivel en los siguientes paquetes:

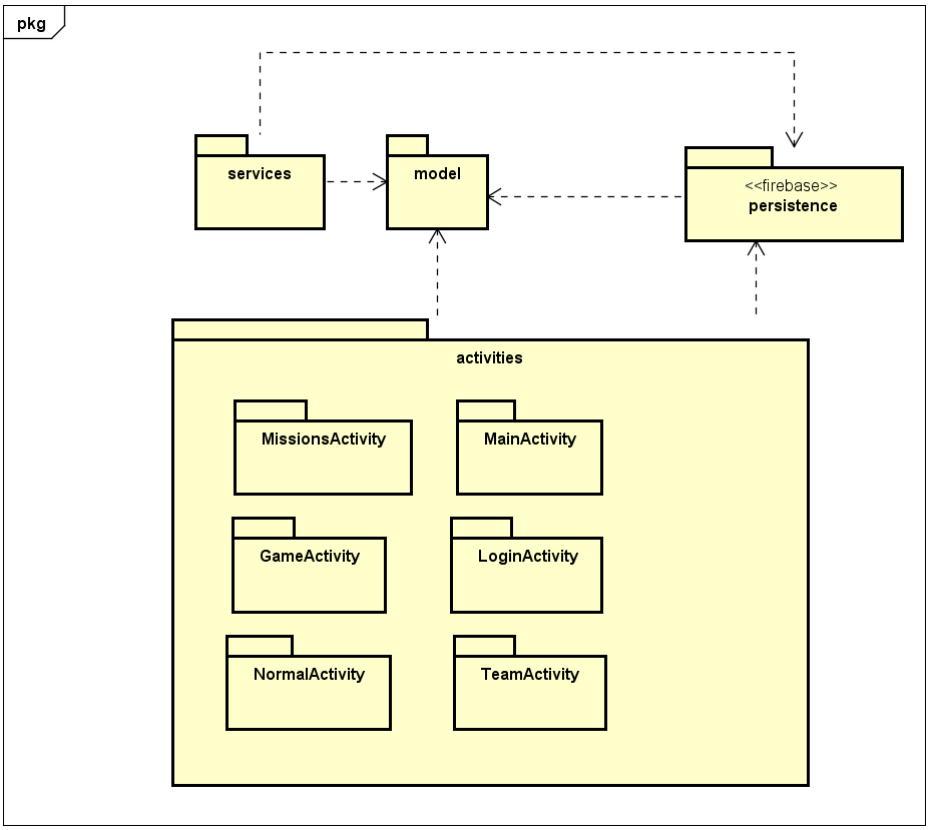


Figura 3.36: Diagrama de paquetes de alto nivel de la aplicación.

La estructura interna de los paquetes es:

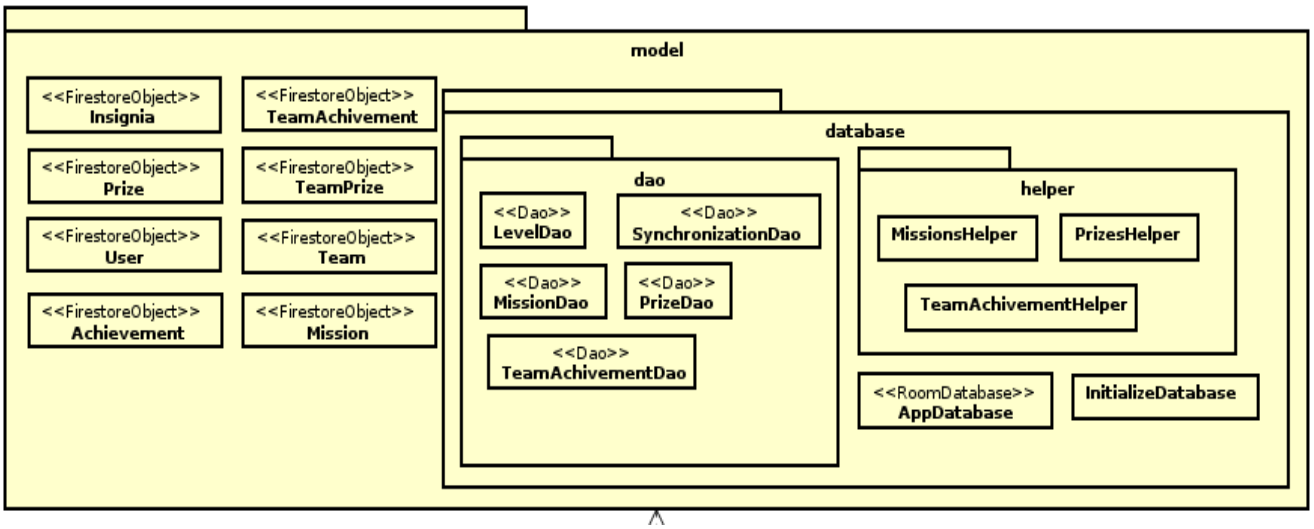


Figura 3.37: Diagrama de clases *Model*.

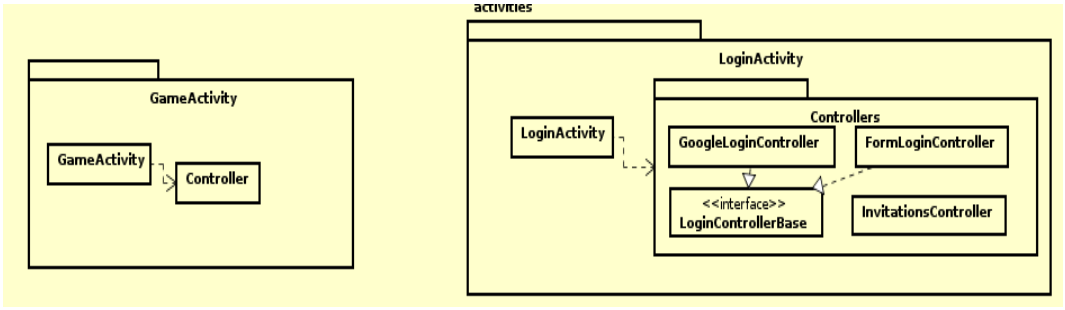


Figura 3.38: Diagrama de clases de los paquetes *LoginActivity* y *GameActivity*.

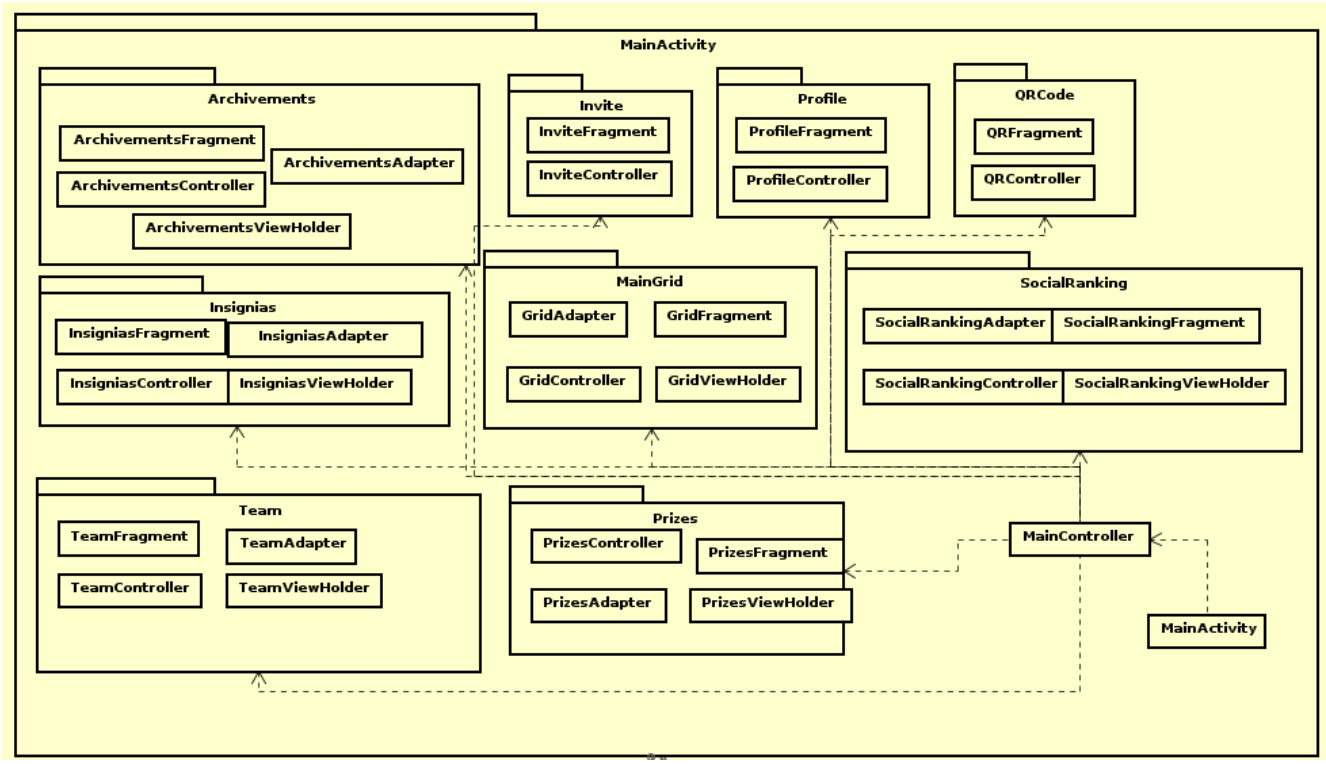


Figura 3.39: Diagrama de clases del paquete *MainActivity*.

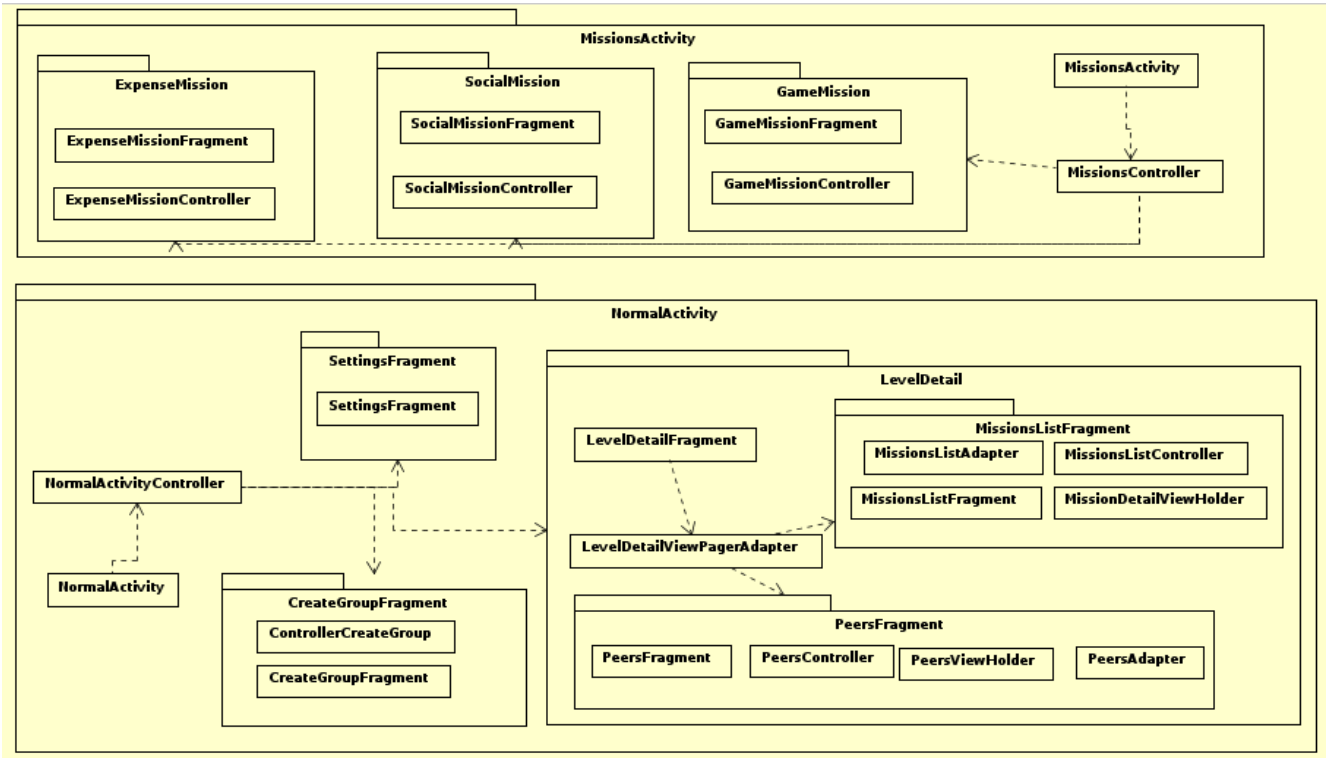


Figura 3.40: Diagrama de clases de los paquetes *MissionsActivity* y *NormalActivity*.

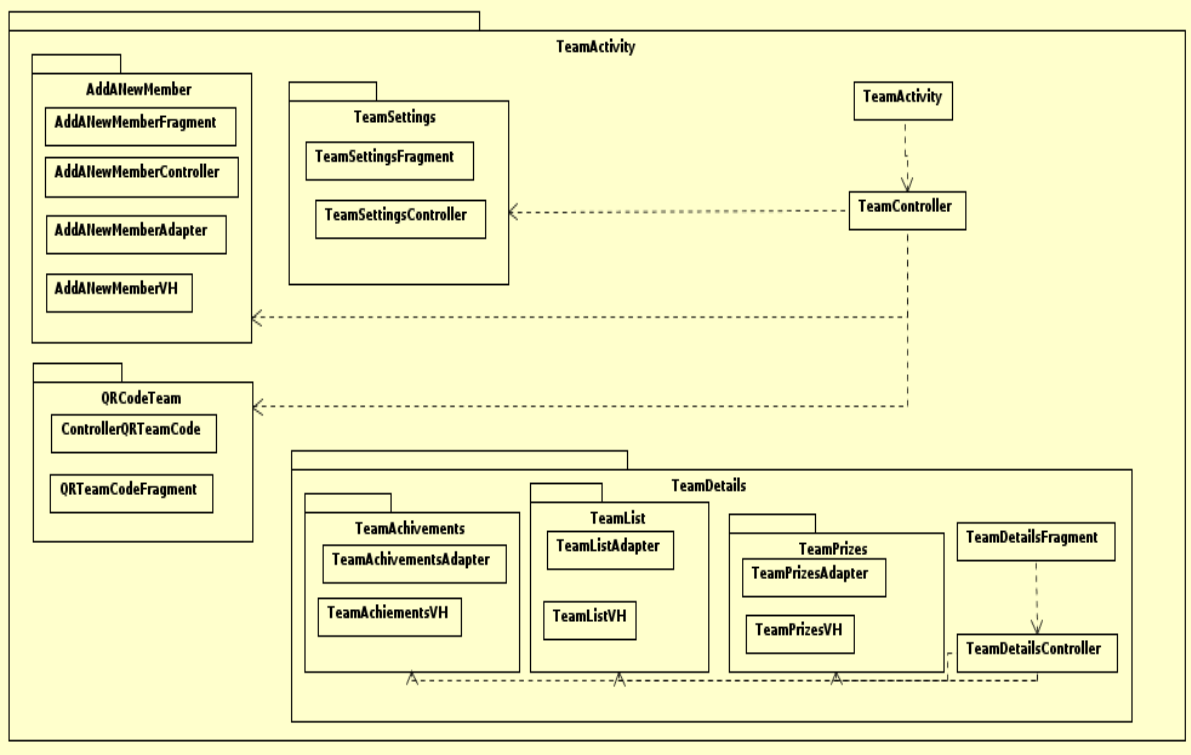


Figura 3.41: Diagrama de clases del paquete *TeamActivity*.

3.3.2. Patrones utilizados

Se ha decidido emplear los siguientes patrones en el diseño e implementación de la aplicación.

3.3.2.1. M.V.C. (Modelo Vista Controlador)

El M.V.C. es un patrón de arquitectura software utilizado para la implementación de interfaces de usuario (*GUI*). El fundamento de este patrón es el de separar la lógica de la aplicación en tres componentes completamente diferenciados [51]:

- **Modelo:** el modelo define la información que contiene la aplicación. Además, deberá tener acceso a un sistema de persistencia que permita acceder a la misma en futuras ocasiones y, en el caso de que el modelo fuera activo, notificará a las vistas de los cambios que ocurrieran sobre el estado, con el fin de que estas refrescaran la información visible al usuario.
- **Vista:** define cómo se deberá mostrar la información al usuario.
- **Controlador:** controla la lógica de negocio necesaria para leer la información introducida por el usuario en la vista y actualizar el controlador. Además, transformará la información proveniente del modelo para que pueda ser mostrada al usuario por la vista.

La siguiente imagen corresponde a un diagrama de clases de una parte de la arquitectura de la aplicación, en la que se aprecia este patrón:

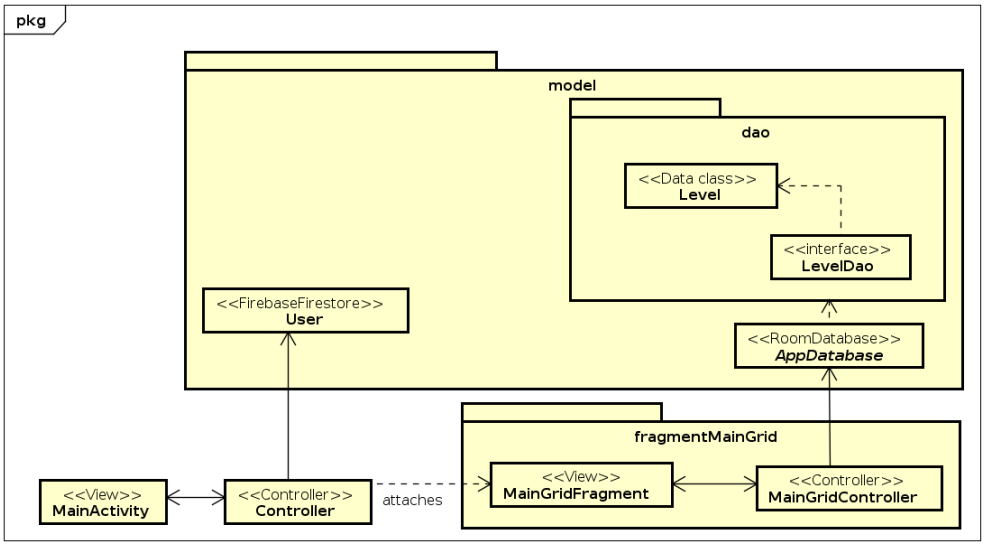


Figura 3.42: Patrón M.V.C.

3.3.2.2. D.A.O. (*Data Access Object*)

D.A.O. realiza una capa de abstracción entre los servicios y llamadas de bajo nivel para acceder a los datos del resto de código, evitando así mezclar código de diferentes capas.

Este patrón de acceso a datos solo se ha podido emplear en las operaciones *CRUD* (*Create, Read, Update, Delete*) realizadas contra la base de datos que es administrada por *Room* [3] y no con la que gestiona *Firebase Firestore*.

El siguiente diagrama de clases muestra el patrón D.A.O. aplicado en la base de datos gestionada por *Room*.

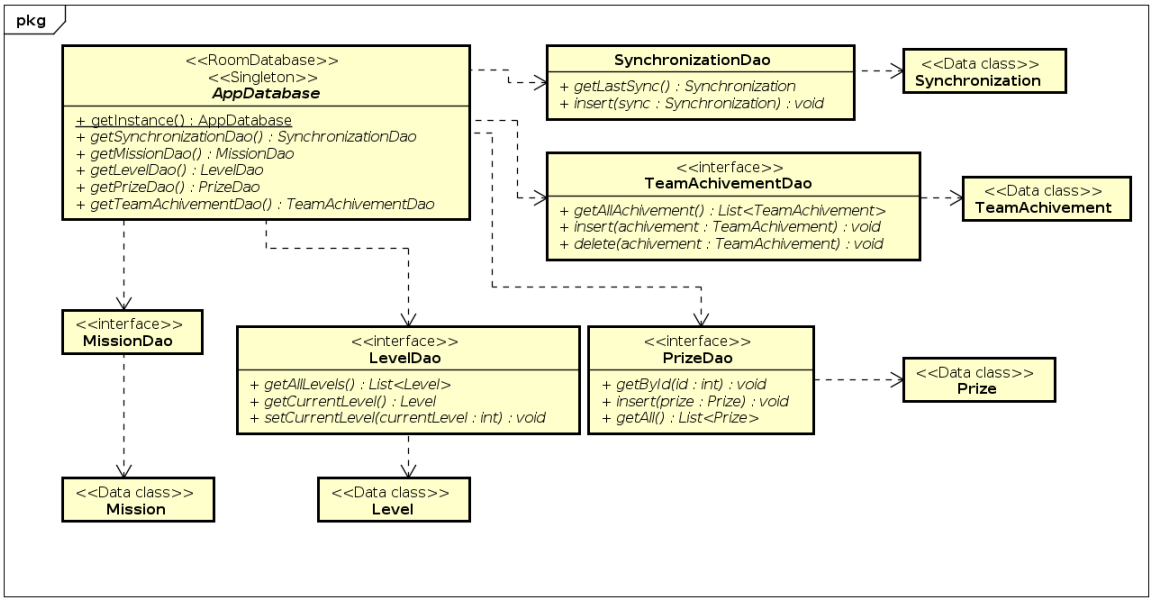


Figura 3.43: Patrón D.A.O.

3.3.3. Esquema de la base de datos

Como se ha indicado en secciones anteriores, se ha decidido utilizar *Firebase* como motor de bases de datos. En concreto, de las dos que ofrece la plataforma: *Firebase Realtime Database* y *Firebase Firestore*, se ha elegido la segunda por ser mucho más compleja y permitir una mayor personalización. Esta base de datos seleccionada es de tipo no relacional, basada en documentos.

Durante el Grado, se ha estudiado la asignatura Diseño de Bases de Datos, impartida por D. Manuel Barrio Solorzano. Sin embargo, esta asignatura se limitaba a enseñar el diseño de bases de datos de tipo relacional y no de tipo no relacional y, al no contar con formación específica en la materia y tras consultar [54], se han realizado los siguientes diseños basándose en el libro citado y en diagramas de *UML*. La documentación que *Google* proporciona [19] también ha servido de mucha ayuda.

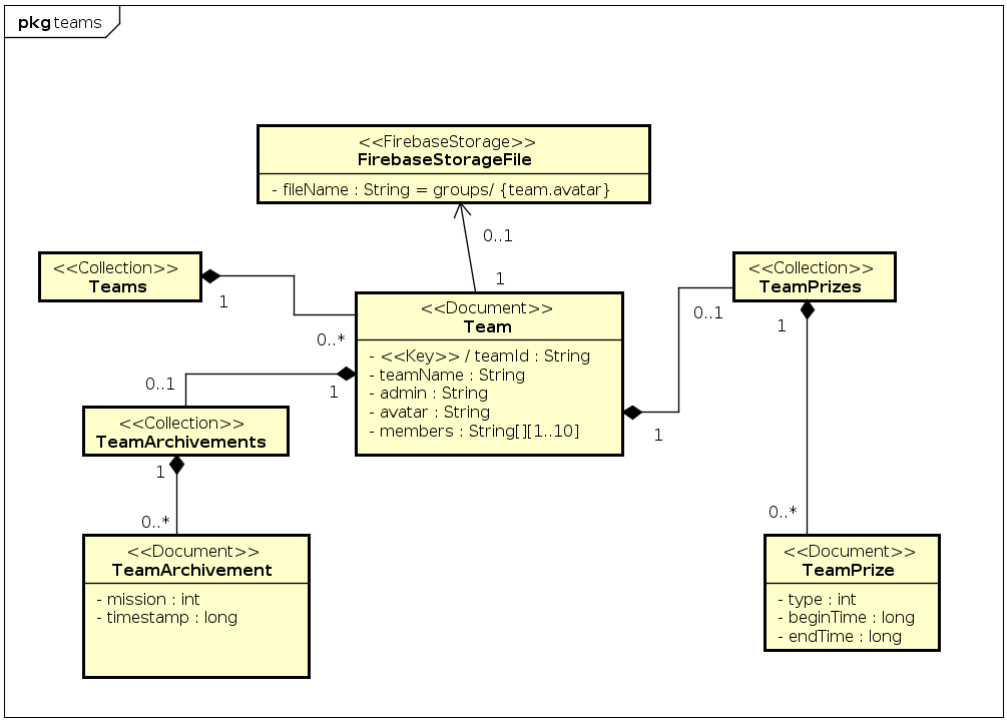


Figura 3.44: Esquema no relacional basado en documentos y colecciones para las entidades relacionadas con equipos.

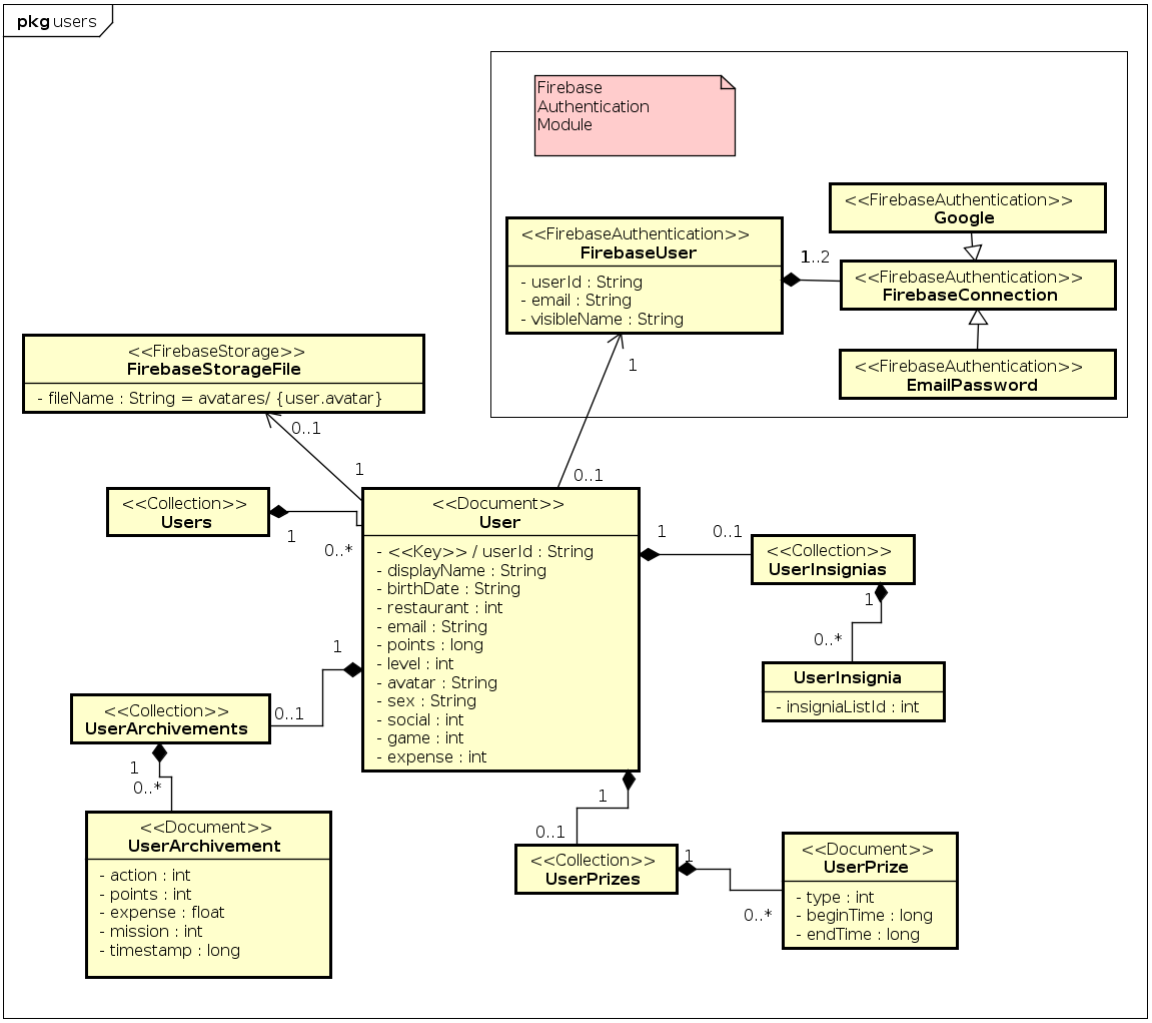


Figura 3.45: Esquema no relacional basado en documentos y colecciones para las entidades relacionadas con usuarios.

De las figuras 3.44 y 3.45 es preciso aclarar que los elementos etiquetados con los estereotipos *Firebase Authentication* y *Firebase Storage* no pertenecen a la base de datos *Firebase Firestore*, sino que se ubican en sus respectivos servicios. Se han querido señalar en este esquema por su gran relación con el resto de colecciones y documentos de la base de datos.

Con el fin de cumplir con los requisitos de persistencia y permitir una actualización de los niveles, también se ha utilizado una base de datos de tipo relacional, gestionada por la biblioteca de persistencia *Room* [3]. Esta base de datos tendrá como objetivo mantener un listado de niveles, misiones y las sincronizaciones que se hayan hecho para descargarse las nuevas misiones, logros y premios.

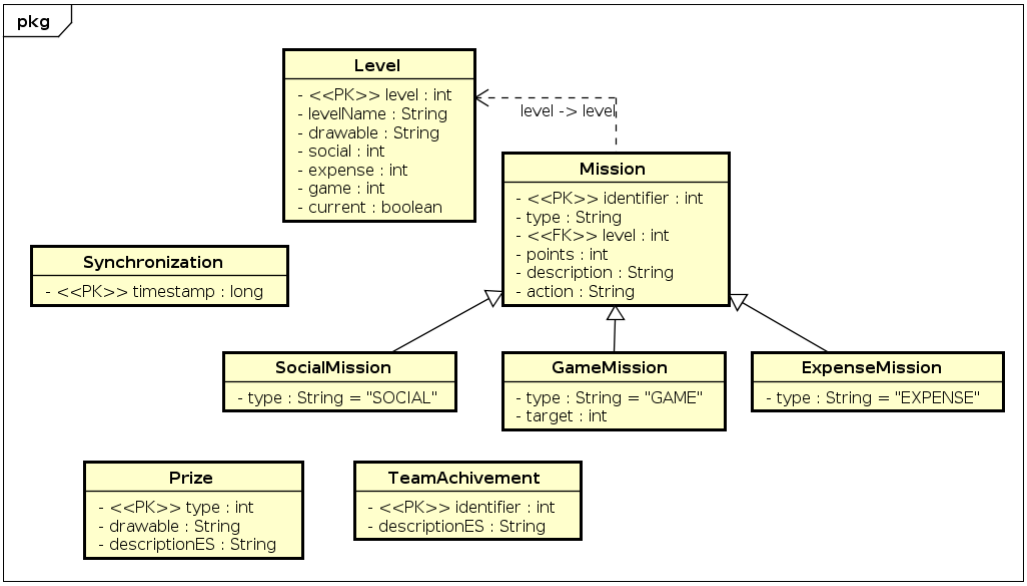


Figura 3.46: Esquema de la base de datos relacional gestionada por la biblioteca de persistencia *Room* [3].

3.3.4. Diagrama de despliegue

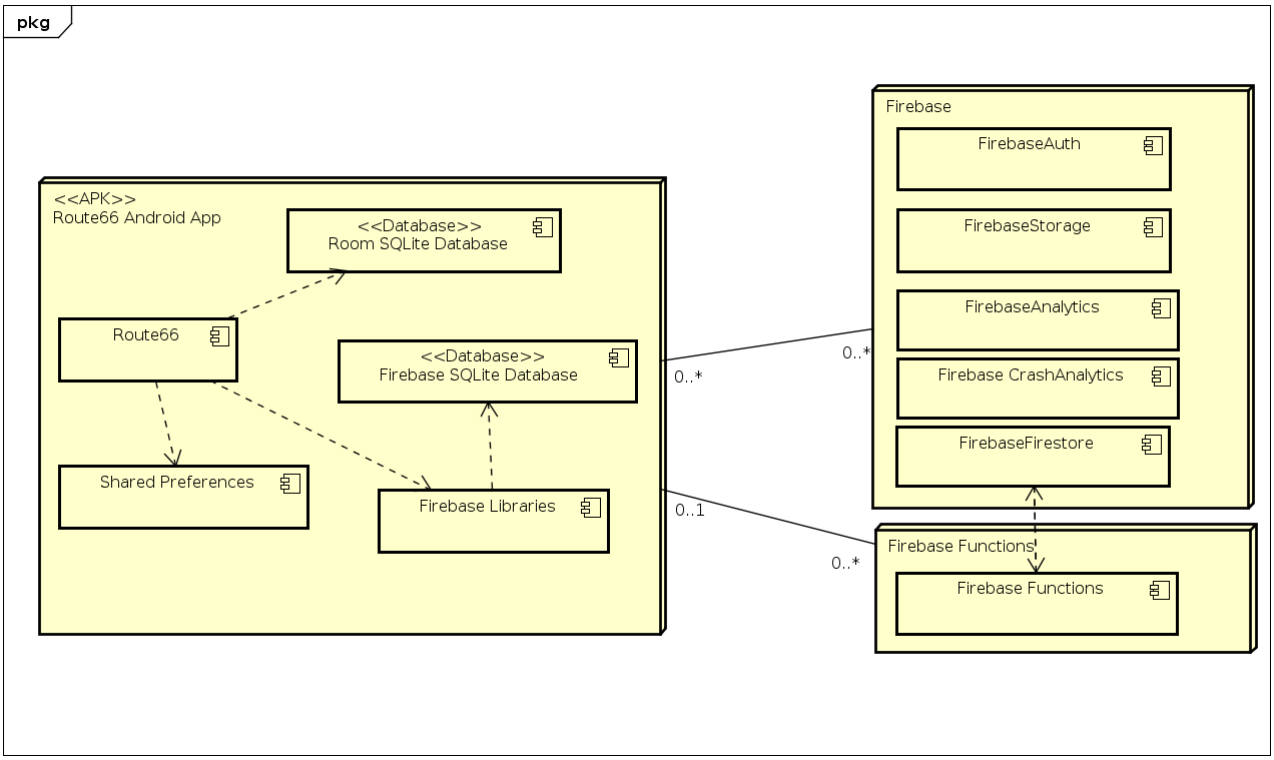


Figura 3.47: Diagrama de despliegue de la aplicación *Android*.

3.4. Implementación

3.4.1. Niveles, insignias y premios

3.4.1.1. De usuarios

Los usuarios de la aplicación contarán con un conjunto de niveles, misiones, premios e insignias que deberán ir obteniendo o completando para avanzar en la aplicación.

- Insignias: Hay una por cada nivel y se obtienen cuando el jugador concluye todas las pruebas del mismo.
- Premios: Se obtienen dos por cada nivel:
  - Nivel 1 (Chicago): “Bebida gratis comprando un plato principal“ y “Dos acompañamientos (ensalada y patatas) gratis comprando un plato principal“.
  - Nivel 2 (Springfield): “Descuento del 9 %“ y “Postre gratis con compras superiores a 30 €“.
  - Nivel 3 (St. Louis): “Descuento del 11 %“ y “Patatas gratis con compras superiores a 30 €“.
  - Nivel 4 (Tulsa): “Descuento del 13 %“ y “Plato principal gratis con compras superiores a 30€“.
  - Nivel 5 (Oklahoma City): “Descuento del 15 %“ y “Producto nuevo gratis con compras superiores a 30€“.
  - Nivel 6 (Amarillo): “Descuento del 17 %“ y “Dos bebidas gratis con compras superiores a 30€“.
  - Nivel 7 (Santa Fe): “Descuento del 19 %“ y “Dos bebidas gratis con compras superiores a 30€“.
  - Nivel 8 (Alburquerque): “Descuento del 21 %“ y “Dos postres gratis con compras superiores a 30€“.
  - Nivel 9 (Flagstaff): “Descuento del 23 %“ y “Dos de patatas gratis con compras superiores a 30€“.
  - Nivel 10 (Williams): “Descuento del 25 %“ y “Dos platos principales gratis con compras superiores a 30€“.
  - Nivel 11 (Los Ángeles): “Descuento del 27 %“ y “Dos productos nuevos gratis con compras superiores a 30€“.

  Finalmente, los diez primeros usuarios que completen todo el juego obtendrán un viaje a Estados Unidos, representado por el premio “Viaje a Estados Unidos gratis“.

Por otro lado, se ha decidido rediseñar parte de la estrategia del juego, centrándose siempre en la idea desarrollada en el T.F.G. sobre el que está basado este proyecto [55], al no contar con el suficiente detalle que se ha requerido en la implementación. De esta forma, los niveles y los detalles concretos de los mismos son los siguientes:

Nivel	Nombre	Puntos sociales	Puntos competitivos	Puntos de consumo	Consumo mínimo	Puntuación mínima juego
1	Chicago	500	0	0	0 €	0
2	Springfield	300	300	300	5 €	300
3	San Louis	400	400	400	10 €	400
4	Tulsa	500	500	500	15 €	500
5	Oklahoma	600	600	600	20 €	600
6	Amarillo	700	700	700	25 €	700
7	Santa Fe	800	800	800	30 €	800
8	Alburquerque	900	900	900	35 €	900
9	Flagstaff	1000	1000	1000	40 €	1000
10	Williams	1100	1100	1100	45 €	1100
11	Los Ángeles	1200	1200	1200	50 €	1200

Cuadro 3.13: Niveles del juego junto con sus detalles correspondientes.

En cada nivel habrá cinco casillas:

- **Casillas de tipo social**: Dependerá del tipo de nivel:
  - Nivel 1 (Chicago): Habrá cinco casillas de tipo social (todas). Cada una vendrá dedicada a un tipo de misión social.
  - Resto de niveles: Por cada ciudad habrá dos casillas sociales y se considerarán de tipo libre, es decir, el usuario puede elegir qué acción desea realizar para completarla.
- **Casillas de tipo competitivo**: exceptuando el primer nivel, habrá dos casillas, cada una será de un juego concreto y su nivel de dificultad irá aumentando gradualmente (la puntuación que se deberá alcanzar será mayor).
- **Casillas de tipo consumo**: habrá una casilla por ciudad (exceptuando en el nivel uno, que no habrá ninguna). El consumo irá aumentando en cada nivel en cinco euros.

No se impone que el usuario deba realizar todo el consumo correspondiente en una sola ocasión, permitiendo dividirlo en varias compras (siempre y cuando enseñe el código *QR* correspondiente a la misión concreta). Por otra parte, todo gasto que exceda del indicado (por ejemplo si registra un gasto de 10€ en el nivel *Springfield*), no será acumulado para la siguiente ciudad.

3.4.1.2. De grupos/equipos

En los equipos solo contaremos con logros y premios:

- **Logros**:
  - Añadido el primer miembro.
  - Añadido el quinto miembro.
  - Añadido el décimo miembro.
  - Celebrados dos cumpleaños en un mismo año.
  - Celebrados cinco cumpleaños en un mismo año.
- **Premios**: serán exactamente los mismos que en el caso de los usuarios, aunque de aplicación a todo el grupo.

3.4.2. Dependencias

La aplicación Route66App cuenta con las siguientes dependencias externas:

1. ***FirebaseUI Firestore*** [15]: ofrece un conjunto de clases utilizadas para mostrar consultas de la base de datos de *Firestore* en un *RecyclerView*.
2. ***Mikepenz MaterialDrawer*** [40]: sirve para generar un *DrawerLayout* siguiendo las guías del lenguaje visual *Material Design*.
3. ***Firebase Crashanalytics*** [13]: utilizado para generar informes de excepciones y de errores en tiempo de ejecución.
4. ***Room*** [41]: biblioteca de gestión de bases de datos *SQLite* en *Android*.
5. ***Glide*** [22]: biblioteca de visualización de imágenes. Ofrece la posibilidad de realizar una caché de las imágenes, con el fin de reducir el número de descargas.

6. ***Firebase Auth*** [11]: servicio de *Firebase* para ofrecer autenticación y mantener un sistema de sesiones.
7. ***Google Play Services: Authentication*** [23]: biblioteca para gestionar la autenticación de los usuarios por medio de *Google Accounts*.
8. ***Zxing Core*** [48]: biblioteca para generar códigos *QR*.
9. ***Firebase Invites*** [16]: servicio de *Firebase* para enviar y recibir invitaciones de usuarios.
10. ***Firebase Firestore*** [14]: base de datos no relacional basada en documentos de la suite *Firebase*.
11. ***Firebase Storage*** [18]: almacenamiento de datos de tipo fichero.
12. ***Firebase Remote Config*** [17]: servicio de *Firebase* para almacenar y actualizar una serie de parámetros de configuración cambiables sin tener que realizar una modificación del *APK*.
13. ***Firebase Cloud Messaging*** [12]: servicio de *Firebase* utilizado para enviar y recibir mensajes que pueden ser posteriormente mostrados como una notificación en el dispositivo.
14. ***Firebase Core*** [10]: bibliotecas centrales de *Firebase*. Ofrecen soporte a *Firebase CrashAnalytics* y *Firebase Analytics*.
15. ***OKHttp*** [37]: Biblioteca de gestión de solicitudes HTTP.
16. ***CircleImageView*** [8]: *Widget* que muestra imágenes (avatares) en una vista redondeada.
17. ***Material Spinner*** [33]: *Widget* que muestra un *Spinner* con un texto anexado en la parte superior, siguiendo así la guía de *Material Design*.
18. ***FAB (Floating Action Button)*** [42]: *Widget* que permite personalizar un *Floating Action Button*, así como añadir opciones adicionales.
19. ***Image Compressor*** [7]: Biblioteca de compresión de imágenes.

3.4.3. Juegos

Se han escogido dos juegos que serán los utilizados y modificados para servir en las misiones de tipo competitivo. Deben cumplir los siguientes criterios:

- Ser fácilmente modificables: deben tener un código fácil de entender y que permita realizar una serie de modificaciones que se utilizarán para notificar a la aplicación *Android* el resultado de la partida.
- Deben tener una licencia que permita su uso y modificación.
- Deben estar escritos en JavaScript y ser enteramente ejecutables por un navegador web móvil.

Los juegos escogidos son:

- **Distance Flyer Blueprint** [27]: el motor de juegos KiwiJS ofrece en su página oficial, en la sección *Blueprints*, el código ejemplo de un clon del famoso juego *Flappy Bird*.
- **2048** [6]: clon en *JavaScript* del juego 2048.

3.4.4. Explicación de la arquitectura escogida

Una aplicación *Android* está basada en cuatro tipos de componentes [4] [50]. Todos deben ser declarados explícitamente en el archivo *AndroidManifest.xml*:

1. **Actividades**: son los elementos más importantes y destacables en una aplicación *Android* y cumplen el papel de interfaz entre la aplicación y la vista.
   Route66 cuenta con las siguientes actividades:

   - ***LoginActivity***: sirve para identificar y registrar a los usuarios en la aplicación (incluyendo aquellos que tienen una invitación). Es la actividad que siempre se lanzará cuando la aplicación sea abierta. Si el usuario ha iniciado sesión le redirigirá a *MainActivity*.
   - ***MainActivity***: es la única actividad que contará con un *DrawerLayout* (el menú principal de la aplicación). De la misma dependen un conjunto de fragmentos (vistas que se acoplan y se desacoplan bajo demanda):
     a) *Archievements*: es la vista que muestra al usuario el conjunto de logros que haya obtenido.
     b) *Insignias*: es el fragmento encargado de mostrar las insignias que haya obtenido el usuario.
     c) *Invite*: su finalidad es servir de interfaz entre la aplicación y el sistema de *Firebase Invitations*, con el fin de enviar invitaciones.
     d) *MainGrid*: muestra el conjunto de niveles con los que cuenta la aplicación y servirá como puerta de enlace a realizar misiones y consultar los usuarios que se encuentran en un determinado nivel.
     e) *Prizes*: lista el conjunto de premios que hubiera conseguido el usuario, así como detalles específicos de los mismos.
     f) *Profile*: permite al usuario editar las propiedades de su perfil (avatar, nombre, restaurante favorito, etcétera).
     g) *SocialRanking*: es el fragmento encargado de mostrar el ranking global de usuarios.
     h) *Team*: es el fragmento que lista el conjunto de equipos a los que pertenece el usuario, así como el de permitir crear uno nuevo.
   - ***MissionsActivity***: es la actividad que servirá como interfaz para realizar misiones de tipo consumo, social y competitivo.
     a) *ExpenseMission*: es el fragmento encargado de mostrar al usuario los detalles de una misión de tipo consumo, incluyendo el código *QR*.
     b) *SocialMission*: es el fragmento encargado de mostrar al usuario los detalles de una misión de tipo social.
     c) *GameMission*: : ofrece al usuario los detalles de una misión de tipo competitivo.
   - ***GameActivity***: es la actividad que servirá como interfaz para realizar misiones de tipo juego. Por las características de este tipo de misiones (la vista es distinta, así como la posible rotación de la pantalla), se ha decidido separarlas de la actividad *MissionsActivity*.
     a) *GameMission*: es el fragmento encargado de mostrar al usuario los detalles de una misión de tipo juego y, si fuera el caso, marcarla como completada.

- ***NormalActivity***: el objetivo de esta actividad es el de servir como auxiliar a la actividad principal (MainActivity). Es decir, será aquella actividad que será llamada por MainActivity.
  a) *CreateGroup*: es el fragmento encargado de mostrar el formulario para crear un nuevo grupo.
  b) *LevelDetail*: es el fragmento encargado de mostrar los detalles de un nivel. Su vista está basada en un *TabLayout* mostrando las siguientes vistas:
    1) *MissionsListFragment*: es el fragmento encargado de listar las distintas misiones de un nivel.
    2) *PeersFragment*: es el fragmento encargado de mostrar el conjunto de usuarios que se encuentran en el nivel.
  c) *SettingsFragment*: panel de configuración básica para ajustar el tono de notificación que se deberá utilizar para notificar al usuario.
- ***TeamActivity***: es la actividad encargada de mostrar las distintas opciones que cuenta un equipo.
  a) *AddANewMember*: fragmento utilizado para buscar a los nuevos miembros que serán añadidos al grupo.
  b) *QRCodeTeam*: vista que muestra el código *QR* asociado a un equipo.
  c) *TeamDetails*: muestra los detalles del grupo, entre los que se encuentran el nombre, los premios, miembros y logros.
  d) *TeamSettings*: panel de administración de los grupos. Permite modificar la imagen de perfil, el nombre e incluso eliminar el equipo.

2. **Servicios**: son componentes que se ejecutan en segundo plano (*background*), tanto de forma automática (por ejemplo, al iniciar el dispositivo), como bajo demanda (mediante una llamada). Se utilizan fundamentalmente para procesar tareas pesadas o que puedan necesitar mucho tiempo. A diferencia de las actividades, no cuentan con interfaz de usuario y están pensados para serle totalmente transparentes.
   Route66App cuenta con dos servicios:

   a) ***UpdaterService***: es utilizado para actualizar la versión en caché de la aplicación de las misiones, niveles, insignias y premios y almacenar estos parámetros en la base de datos *SQLite*. Por otro lado, los ficheros asociados (en su gran mayoría imágenes), los guarda en una carpeta específica de la aplicación.
   b) ***Route66MessagingService***: es el servicio encargado de recibir los mensajes de notificación que llegan de *Firebase Cloud Messaging* y de realizar la notificación adecuada.

3. **Proveedores de contenidos** (*"Content providers"*): sirven de interfaz para administrar el acceso a un conjunto compartido de datos de la aplicación y proporcionar los mecanismos de seguridad necesarios para ello. Una de sus principales aplicaciones es el de permitir compartir grandes cantidades de información entre procesos, aunque también pueden ser utilizados únicamente por la misma aplicación en la que están contenidos.
   Route66App no cuenta con ningún proveedor de contenidos.

4. **Receptores de mensajes** (*"Broadcast receivers"*): este tipo de componente es utilizado para responder ante eventos que ocurran en el sistema (por ejemplo, una falta de disponibilidad de conexión a la red, el arranque del sistema *Android*, etcétera). El objetivo fundamental de un receptor de mensajes es el de alertar y no el de realizar tareas complejas.
   Route66App no cuenta con receptores de mensajes.

3.4.5. Almacenamiento de datos

La aplicación utiliza varios tipos de almacenamiento:

1. En *Firebase Firestore*: es la base de datos principal de la aplicación. Contiene la información necesaria para saber las misiones, los logros o los premios obtenidos y para conocer a qué grupos pertenece el usuario y los parámetros de los mismos. Hay una versión almacenada localmente en el dispositivo (*caché)*, que se sincronizará con la parte remota cuando el usuario disponga de una conexión a Internet.
2. En *Firebase Authentication*: gestionará todos los flujos de identificación de usuarios y gestionará el sistema de sesiones.
3. En *Firebase Analytics*: almacenará los diferentes datos recogidos por la aplicación cuyo único interés sea el de realizar estadísticas.
4. En *Firebase Cloud Messaging*: guardará y enviará las diferentes notificaciones que se mostrarán en los dispositivos de los usuarios.
5. En *Firebase Storage*: sincroniza los avatares de los usuarios y de los grupos.
6. En una base de datos local *SQLite*: sirve para almacenar los niveles, insignias, premios y misiones de la aplicación, con sus correspondientes parámetros. De igual forma, también guardará el historial de sincronizaciones de los mismos. Está gestionada por [3] la biblioteca de persistencia Room.
7. En ficheros: se utilizan ficheros para mostrar las imágenes de los niveles, los avatares de los usuarios, etcétera (cuando estos se encuentren en la caché). Igualmente, también se requieren ficheros para obtener los niveles y misiones en cuanto el usuario instale la aplicación y no se tenga en ese momento conexión a Internet.
8. En *Shared Preferences*: empleada para el almacenamiento persistente de la configuración de determinados parámetros de la aplicación, como por ejemplo, el tono de notificación.
9. En *Firebase Remote Config*: almacena toda la información necesaria para la sincronzación de las misiones, niveles, logros e insignias.

3.4.6. Implementación del modelo

Tal y como se ha visto en el punto anterior se utilizan varias clases de almacenamiento dependiendo del tipo de datos que queremos tratar. Se indicarán a continuación los detalles de cada soporte de almacenamiento.

3.4.6.1. *Firebase Firestore*

Firebase Firestore es la base de datos principal del servicio y en donde se almacena información crucial para el correcto funcionamiento del sistema informático como:

- Datos sobre las misiones completadas por los usuarios y equipos.
- Parámetros del perfil del usuario. item Premios obtenidos.
- Insignias recibidas.

De igual manera, el mismo componente ofrece una capa de seguridad por medio de una serie de reglas de seguridad, ya que *Firestore* internamente realiza cuantas comprobaciones de permisos sean precisas para asegurar tanto el acceso de lectura, como el de escritura. De esta forma, añadimos al servicio una buena capa de seguridad y evitamos tener que recurrir a un *backend* que no deja de ser otro punto en el que puede haber vulnerabilidades. Estas reglas de permisos aplicadas se adjuntan en el anexo.

3.4.6.2. *Authentication*

Firebase Authentication es utilizado en el sistema informático para almacenar las sesiones y los diferentes métodos de autenticación del usuario. Además, es utilizado en primera instancia cuando el usuario se acaba de registrar para recuperar de los proveedores de acceso datos iniciales así como su nombre o su correo electrónico.

3.4.6.3. Analytics

Esta parte está enfocada directamente a realizar las estadísticas que pueden ser empleados por la parte de gestión de la aplicación. Para ello se envían datos anonimizados y relacionados con:

- Intentos de misión.
- Tiempo de uso de la aplicación.
- Veces que abre la aplicación.
- Secciones que más visita.

La implementación se hace directamente llamando a los métodos adecuados desde el mismo controlador. El mismo *Firebase Analytics* por medio de la dirección IP y los datos de sesión (así como la cuenta de *Google* asociada al terminal *Android*), es capaz de determinar la ubicación, versión de la aplicación y versión de *Android*.

3.4.6.4. *Cloud Messaging*

Cloud Messaging es el servicio de notificaciones que *Google Cloud* ofrece dentro del paquete de *Firebase*.

Las notificaciones pueden ser enviadas a un dispositivo individual o a un grupo (máximo 20 dispositivos). En el caso de Route66App utilizaremos siempre el envío a un grupo en concreto (todos los grupos de un mismo usuario) por varios motivos:

1. El usuario puede estar utilizado cualquier dispositivo vinculado a su cuenta. Si la notificación se enviara a uno distinto del que esté usando en ese momento, no la recibirá.
2. Un usuario puede haber cedido su dispositivo a otra persona, o incluso, haber dejado de usarlo. Si no se actualiza en la lista de dispositivos de un usuario, nunca recibirá la notificación.

Un usuario entrará en su grupo de dispositivos en el momento en el que inicie sesión o se registre en Route66App.

3.4.6.5. *Storage*

Firebase Storage es la parte encargada de almacenar los avatares de usuario y de grupo. Es un almacenamiento de objetos construido sobre *Google Cloud Storage*.

Se ha utilizado la siguiente estructura de archivos:

- */avatares*: donde se almacenarán los avatares (imágenes de perfil) de los diferentes usuarios.
- */groups*: similar al punto anterior, pero para las imágenes de perfil de los equipos.

Por motivos de escalabilidad y de eficiencia se han establecido los siguientes requisitos:

- Formatos de imagen: *JPG*, *PNG* o *GIF*.
- Tamaño de referencia: 0.2 MiB.

Todas las imágenes subidas al servicio serán previamente redimensionadas para ajustar al tamaño máximo indicado anteriormente. De esta forma se consigue que:

- Se decremente la cantidad de datos que se deben transmitir por la red.
- Se evite tener que tener una función (con el gasto que eso conlleva de tiempo) que realice esta modificación.
- Se ahorre espacio en *Firebase Storage*.

Con el fin de reducir el uso de datos y el número de descargas, se realizará una caché de los datos gracias a *Glide*, un plugin de visualización de imágenes en *Android*.

3.4.6.6. Base de datos local *SQLite*

Como se ha indicado anteriormente, se ha decidido utilizar la biblioteca de persistencia *Room*, ofrecida por *Android* y muy recurrida en el desarrollo de aplicaciones móviles.

Para su utilización se ha empleado:

- Un archivo *AppDatabase*: representa a toda la base de datos *SQLite*. Es el punto de entrada hacia la misma (sigue el patrón *Facade*). Con el fin de asegurar que solo haya una única instancia se ha aplicado el patrón *Singleton*.
- *D.A.O.*: Un *D.A.O.* en *Room* es simplemente una interfaz que incluye la cabecera de cuantos métodos se desee incluir. Tiene una serie de métodos ya implementados y con los que basta añadir una anotación y otros (como consultas *SELECT* complejas), en los que ya es preciso escribir la consulta completa.
- *Entities*: Representan una tabla concreta de la base de datos e incluye como atributos cada uno de los campos que la conforman, (incluyendo su tipo, ciertas propiedades características de una base de datos relacional y el uso de claves primarias o autoincrementos).

Esta base de datos permite mantener una sincronización de las diferentes misiones, niveles, insignias y logros. Se debe recordar que existe un requisito en el que se indicaba que estos debían ser actualizables en cualquier momento por parte del administrador desde el *backend*.

3.4.6.7. Ficheros

Los ficheros resultan de especial importancia para inicializar la base de datos, en caso de que la primera vez que el usuario abra la aplicación no tenga conexión o esta sea de mala calidad. Se cuentan con los siguientes ficheros:

- *missions.json*: Incluye el conjunto de niveles y misiones de la aplicación.
- *insignias.json*: Incluye las insignias con las que cuenta la aplicación.
- *prizes.json*: Incluye los diferentes premios existentes en el sistema, la dirección a la imagen que los representa y una breve descripción en castellano.

3.4.6.8. *Shared preferences*

Las preferencias compartidas son un método de persistencia utilizado en *Android* para el almacenamiento de pequeñas cantidades de datos. Se utilizan en formato clave-valor, por lo que son un mecanismo ideal para el almacenamiento de parámetros de configuración.

En Route66App se han utilizado para guardar únicamente el tono de notificación del usuario, que será el que suene cuando le llegue una notificación entrante proveniente de una confirmación de misión realizada o consumo realizado.

3.4.6.9. *Firebase Remote Config*

Este producto de *Firebase* resulta de especial utilidad en aplicaciones en las que se desea que un mismo parámetro de configuración de la aplicación, cambie a la vez en todas las instalaciones, o bien, en ciertas específicas que cumplan una determinada condición (por ejemplo, idioma del dispositivo, región o versión de *Android* instalada). De esta forma, es posible aplicar un determinado ajuste a una franja de usuarios para comprobar así el efecto que tiene y, en caso de resultar satisfactorio, ir ampliando gradualmente el número de instalaciones con ese parámetro. Cabe destacar que en ningún momento se exige que el usuario se vuelva a instalar o actualizar la aplicación.

En Route66App se ha utilizado este producto para ajustar la URL de los archivos de misiones, premios e insignias, así como la hora y fecha de última actualización de esos archivos.

3.4.7. Funciones

Otro de los aspectos fundamentales de Route66App es el uso de las funciones, muy conocidas en el mundo del *Cloud Computing*. Son pequeños fragmentos de código que se ejecutan en el caso de que ocurra una determinada condición. En Route66App, se ha decidido utilizar *Firebase Cloud Functions* escuchando a los siguientes eventos:

1. Las escrituras de *Firebase Firestore*: Es preciso escuchar todos aquellos eventos de escritura y actualización sobre los objetos almacenados en las colecciones de *Achivements* (logros). Esto es necesario para:
   a) Poder actualizar la base de datos con la suma de las puntuaciones del usuario.
   b) Poder notificar al usuario en caso de que se le añada un nuevo logro de tipo consumo.
   c) Avisar de que un logro de tipo social ha sido comprobado.
2. Las llamadas *HTTP* contra el *endpoint*. (*/invitations/userId*): Este método será ejecutado cuando llegue una llamada *GET* entrante y anotará en la cuenta del usuario que ha invitado a un usuario.

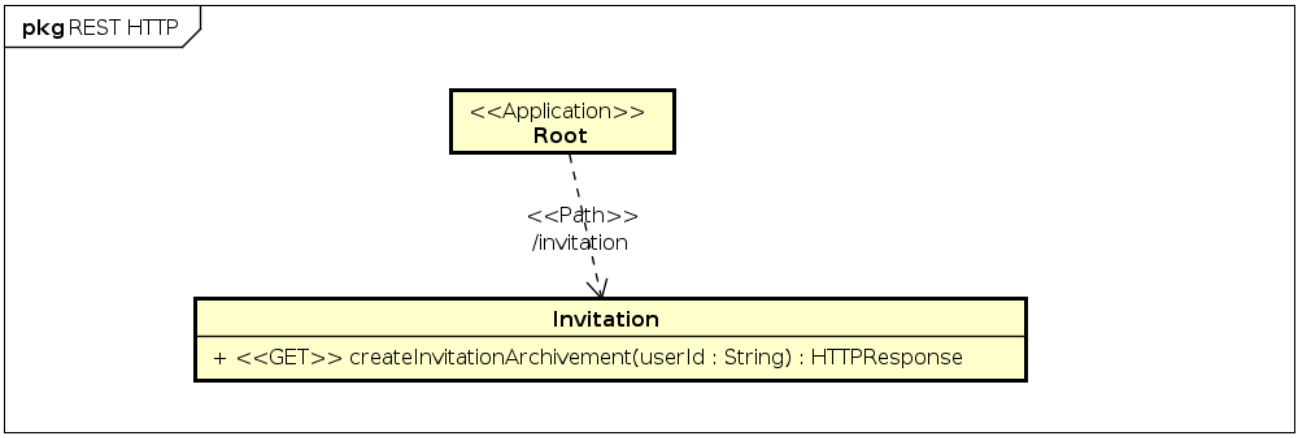


Figura 3.48: Diagrama en el que se muestra la operación *GET*.

3.5. Pruebas

En el desarrollo de *software* resultan de especial relevancia las pruebas de código, que pueden ser tanto de caja negra (sin visualizar el código) como de caja blanca. Una de las formas más efectivas de comprobar que aquel código que se modifica no "rompa" el sistema (es decir, deje de funcionar tal y como está previsto), es mediante el uso de pruebas unitarias que, en *Android*, se suelen realizar mediante el *framework JUnit* [5].

Por otro lado, al hablar de una aplicación con interfaz gráfica también conviene señalar la importancia de las pruebas sobre la interfaz de usuario, que no suelen ser muy conocidas, pues muchos desarrolladores las realizan manualmente, probando la aplicación ellos mismos. En *Android* contamos con los siguientes *frameworks* diseñados por *Google*:

- *Espresso* [6] se utilizará cuando tengamos acceso al código fuente de la aplicación (pues se realizan sobre el mismo proyecto de la aplicación). Útil cuando vamos a hacer pruebas sobre nuestra propia aplicación.
- *UIAutomator* [7] cuando no tengamos acceso al código fuente (por ejemplo, cuando queremos probar una aplicación de terceros).

Sin embargo, en esta aplicación no se realizarán pruebas automatizadas, sino que serán puramente manuales y de caja negra, de acuerdo con los requisitos señalados por la tutora del T.F.G. Estas pruebas son las que describiré con detalle a continuación.

TEST1	El usuario se registra mediante correo electrónico.
Resultado esperado	El usuario se registra. Se le muestra la pantalla principal de la aplicación, con los distintos niveles.
Resultado obtenido	Se ha obtenido el resultado esperado.

[5] https://junit.org/junit5/es
[6] https://developer.android.com/training/testing/espresso/index.html
[7] https://developer.android.com/training/testing/ui-automator.html

TEST2	El usuario se registra/inicia sesión utilizando *Google*.
Resultado esperado	Se le muestra al usuario un listado con las cuentas de *Google* asociadas con el dispositivo y se le permite escoger una con la que iniciar sesión.
Resultado obtenido	Error a la hora de iniciar sesión (mensaje en un Toast). Se revisa las trazas de Logcat y se llega a la conclusión de que es porque el hash *SHA1* del certificado que firma el *APK* instalado no se corresponde con aquel registrado en *Firebase*.

TEST3	El usuario se registra/inicia sesión utilizando *Google*.
Resultado esperado	Se le muestra al usuario un listado con las cuentas de *Google* asociadas con el dispositivo y se le permite escoger una con la que iniciar sesión.
Resultado obtenido	Se obtiene el resultado esperado.

TEST4	El usuario inicia sesión mediante correo electrónico.
Resultado esperado	El usuario inicia sesión. Se le muestra la pantalla principal de la aplicación, con los distintos niveles.
Resultado obtenido	Se ha obtenido el resultado esperado.

TEST5	El usuario modifica su configuración de perfil.
Resultado esperado	Se actualiza la configuración de su perfil y aparece reflejado como tal en el Drawer de la aplicación.
Resultado obtenido	Se ha obtenido el resultado esperado.

TEST6	El usuario sube una fotografía de perfil (avatar).
Resultado esperado	Se sube la fotografía de perfil al servidor y se actualiza en todos aquellos lugares en los que debiera aparecer.
Resultado obtenido	Se sube al servidor, pero esta no se descarga de nuevo (no aparece actualizada en el *Drawer* y en el resto de opciones que utilizan el avatar). Finalmente se decide que para obligar a *Glide* a refrescar de nuevo la fotografía es necesario dar un identificador único a cada imagen que se sube al servidor.

TEST7	El usuario sube una fotografía de perfil (avatar).
Resultado esperado	Se sube la fotografía de perfil al servidor y se actualiza en todos aquellos lugares en los que debiera aparecer.
Resultado obtenido	Se obtiene el resultado esperado.

TEST8	El usuario realiza una misión de tipo social.
Resultado esperado	La misión aparece en estado de comprobación. Cuando ha sido revisada por el equipo de Route66App, es marcada como completada en caso de ser validada.
Resultado obtenido	Se ha obtenido el resultado esperado.

TEST9	El usuario realiza un consumo en una misión de tal tipo.
Resultado esperado	El consumo es reflejado en la pantalla. Si al introducir el consumo el importe es igual o superior al exigido por la misión, esta es marcada como completada.
Resultado obtenido	Se produce un error fatal de la aplicación. Se revisan las trazas y es porque se ha registrado el consumo como *String* en vez de como número en coma flotante.

TEST10	El usuario realiza un consumo en una misión de tal tipo.
Resultado esperado	El consumo es reflejado en la pantalla. Si al introducir el consumo el importe es igual o superior al exigido por la misión, esta es marcada como completada.
Resultado obtenido	Se obtiene el resultado esperado.

TEST11	El usuario realiza una misión de tipo competitiva.
Resultado esperado	Aparece una descripción del juego correspondiente, así como la puntuación que debe alcanzar.
Resultado obtenido	Se obtiene el resultado esperado.

TEST12	El usuario visualiza los logros recibidos.
Resultado esperado	Se visualizan los logros del usuario.
Resultado obtenido	Efectivamente se visualizan los logros del usuario, pero se muestran también aquellos con una puntuación nula (correspondiente a misiones aún no completadas o validadas). Se decide ocultar las mismas modificando el *adapter* asociado.

TEST13	El usuario visualiza los logros recibidos.
Resultado esperado	Se visualizan los logros del usuario.
Resultado obtenido	Se obtiene el resultado esperado.

TEST14	El usuario visualiza las insignias recibidas.
Resultado esperado	Se muestran las insignias recibidas.
Resultado obtenido	Se obtiene el resultado esperado.

TEST15	El usuario visualiza los premios que tiene asignados.
Resultado esperado	Se muestran los premios que tiene asignados.
Resultado obtenido	Se obtiene el resultado esperado.

TEST16	El usuario envía una invitación.
Resultado esperado	Se envía la invitación.
Resultado obtenido	Se obtiene el resultado esperado.

TEST17	El usuario acepta la invitación y se registra.
Resultado esperado	El usuario se registra y el que le ha invitado es notificado.
Resultado obtenido	Se obtiene el resultado esperado.

TEST18	El usuario visualiza el ranking global.
Resultado esperado	Se muestra el ranking global de los diez usuarios con más puntuación en la aplicación.
Resultado obtenido	Se obtiene el resultado esperado.

TEST19	El usuario visualiza los equipos que tiene asignados.
Resultado esperado	Se muestra el listado de los equipos que tiene asignado el usuario, junto con la imagen de avatar de cada uno de ellos.
Resultado obtenido	Se obtiene el resultado esperado.

TEST20	El usuario visualiza los logros de un equipo.
Resultado esperado	Se visualizan los logros del equipo.
Resultado obtenido	Se obtiene el resultado esperado.

TEST21	El usuario crea un nuevo equipo.
Resultado esperado	Se muestra un formulario preguntando el nombre del nuevo equipo, se introduce el dato y se crea un nuevo equipo.
Resultado obtenido	Se obtiene el resultado esperado.

TEST22	El usuario, con rol administrador, añade un usuario al equipo.
Resultado esperado	El usuario es buscado en una lista y añadido al equipo.
Resultado obtenido	No se muestra el usuario que se ha buscado. Debido a la implementación de *Firebase Firestore*, es preciso utilizar operadores comparadores ($\leqslant$ y $\geqslant$) para poder encontrar un *String* que comience por una determinada cadena. Se procede a la modificación del algoritmo utilizado para la búsqueda.

TEST23	El usuario, con rol administrador, añade un usuario al equipo.
Resultado esperado	El usuario es buscado en una lista y añadido al equipo.
Resultado obtenido	Se obtiene el resultado esperado.

TEST24	El usuario, con rol administrador, elimina un equipo.
Resultado esperado	El equipo es eliminado.
Resultado obtenido	Se obtiene el resultado esperado.

TEST25	El usuario, con rol usuario normal, sale de un equipo.
Resultado esperado	El usuario sale del equipo.
Resultado obtenido	Se obtiene el resultado esperado.

TEST26	Cierre de sesión
Resultado esperado	Se cierra sesión en la cuenta Route66App.
Resultado obtenido	Se cierra sesión, pero se produce una excepción sobre la actividad *MainActivity*. El motivo de este error es que no se está eliminando el escuchador sobre el estado del documento de *Firestore* asociado al usuario. Se corrige el *bug*.

TEST27	Cierre de sesión
Resultado esperado	Se cierra sesión en la cuenta Route66App.
Resultado obtenido	Se obtiene el resultado esperado.

3.6. Manual de instalación

Los pasos que hay que seguir para la instalación de Route66App son los mismos de cualquier aplicación *Android* nativa que no se encuentra alojada en una tienda oficial de aplicaciones (como *Google Play Store*), aunque habrá que prestar atención que el dispositivo sobre el que se ejecuta la aplicación cumple los siguientes requisitos mínimos:

- **Versión mínima de *Android***: *Android* 4.4 (más conocido como *Android KitKat*).
- **Software con el que debe contar el dispositivo**: *Google Play Services* actualizado, *WebView* del sistema *Android*.
- **Conexión a Internet**: conexión a Internet estable.
- **Espacio de almacenamiento**: se requerirán, al menos, 50 MiB libres de almacenamiento en el dispositivo.

3.7. Guía de usuario

3.7.1. Usuarios

La aplicación sigue totalmente los estándares de *Material Design*, con el fin de conseguir que el usuario medio de *Android* le resulte una interfaz gráfica familiar. En el apartado “Usabilidad“ (página 68) se han explicado los diferentes principios de Interacción Persona-Computadora aplicados en la interfaz gráfica de la aplicación.

Una vez instalada la aplicación nos encontramos con la siguiente vista que permite iniciar sesión o registrarse.

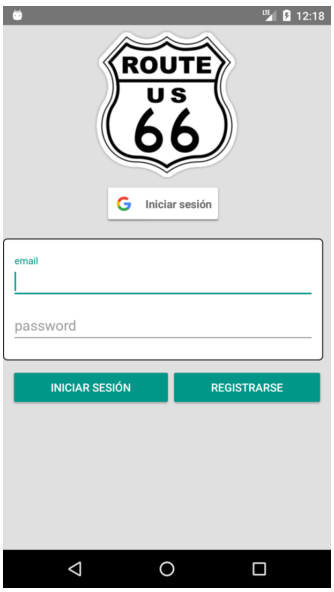
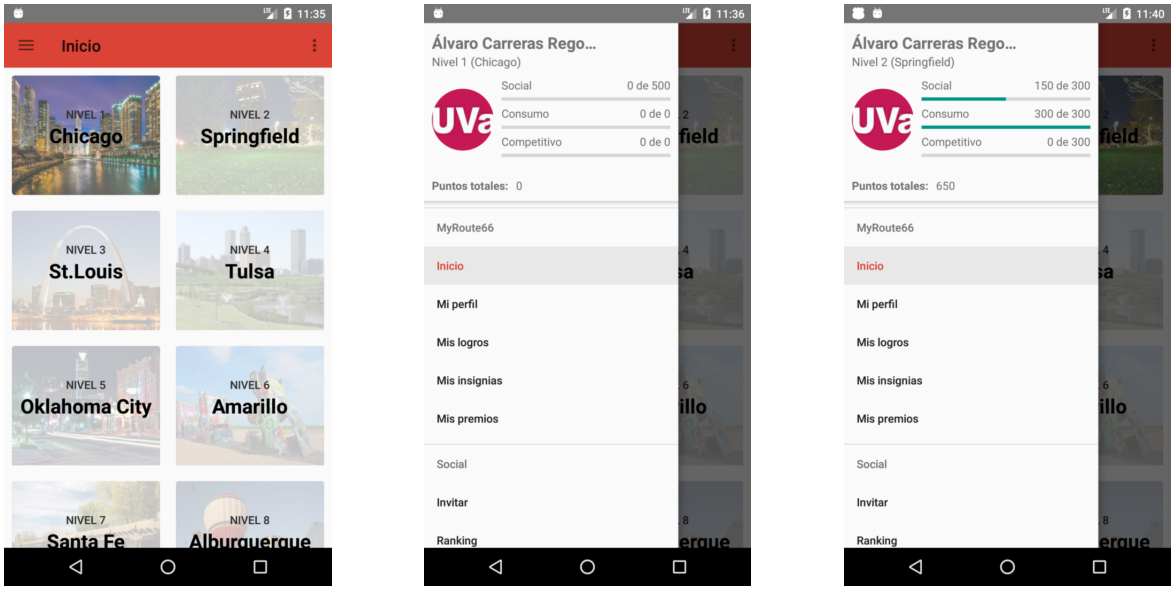


Figura 3.49: Captura de pantalla de la vista de inicio de sesión y registro.

Esta vista permite:

- Iniciar sesión o registrarse utilizando una de las cuentas de *Google* registradas en el dispositivo.
- Iniciar sesión con una cuenta previamente registrada en el sistema utilizando el método de registro con correo electrónico y contraseña.
- Registrarse en el sistema por medio de correo electrónico y contraseña.

Una vez se haya iniciado sesión (registrarse implica iniciar sesión automáticamente), se verá la actividad principal de la aplicación.



(a) Vista del selector de niveles.

(b) Menú principal sin puntuaciones asociadas.

(c) Menú principal con puntuaciones asociadas.

Figura 3.50: Capturas de pantalla de la vista principal de la aplicación.

En el menú podemos observar las diversas opciones con las que contamos:

1. **Inicio**: para volver a la vista principal de la aplicación.
2. **Mi perfil**: permite editar el perfil de usuario, con opciones como subir una nueva fotografía (avatar), cambiar la contraseña o modificar el nombre, restaurante favorito, sexo o la fecha de nacimiento.
3. **Mis logros**: lista el conjunto de logros de un usuario. Un logro es una invitación que se haya completado o una misión terminada. Cabe destacar que solo los logros conllevan un incremento de puntuación, por lo que esta misma vista puede servir como historial de la puntuación del usuario.
4. **Mis insignias**: lista las insignias que tiene asociadas el usuario. Las insignias indican los niveles terminados por parte del usuario.
5. **Mis premios**: muestra los premios que tiene asignados el usuario, destacando las fechas de comienzo y finalización de los mismos.
6. **Invitar**: permite enviar una invitación a otro usuario por correo electrónico. Cuando una misión se completa se añaden los puntos correspondientes en el perfil del usuario que ha enviado la invitación.

7. **Ranking**: consiste en un listado de todos los usuarios de la aplicación, ordenados por la puntuación que han obtenido.
8. **Equipo**: acceso a las opciones correspondientes de los equipos. Se detallarán más adelante.

A continuación se muestran las vistas de las opciones indicadas con anterioridad:

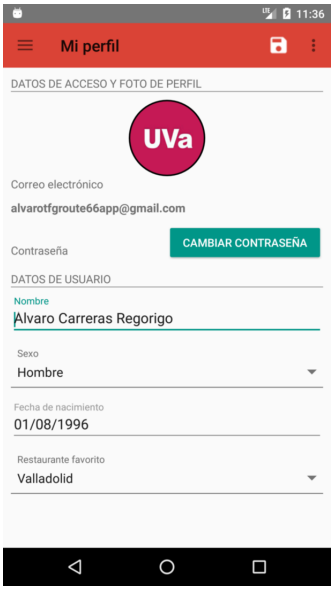
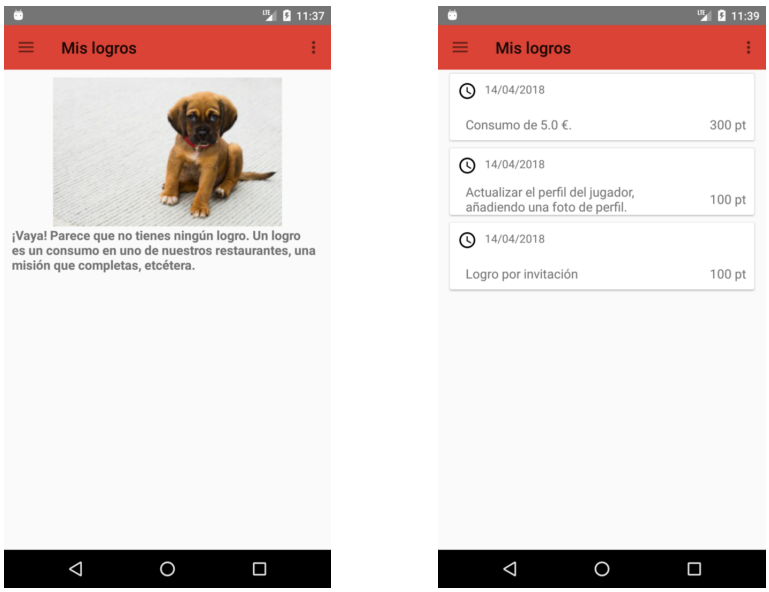
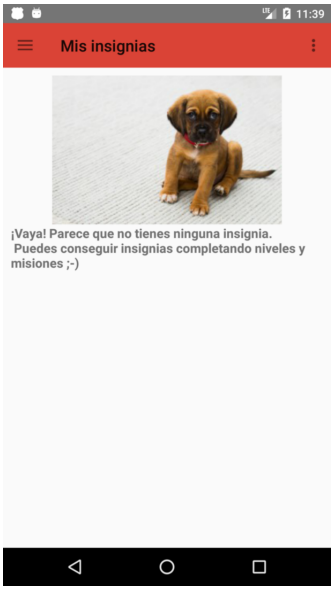


Figura 3.51: Captura de pantalla de “Mi perfil“.

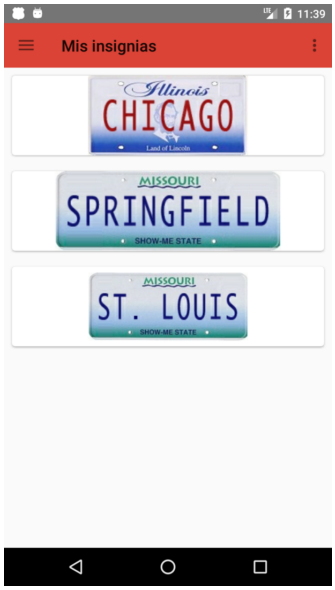


(a) “Mis logros“ sin logros obtenidos. (b) “Mis logros“ con logros obtenidos.

Figura 3.52: Capturas de pantalla de “Mis logros“.

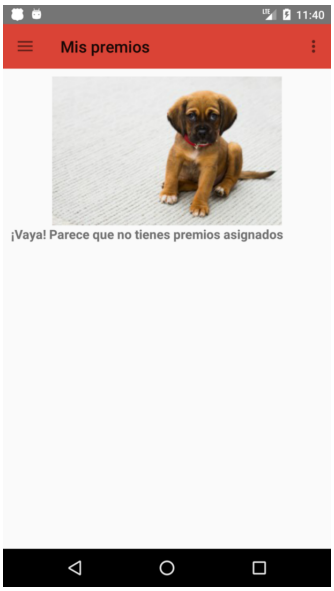


(a) “Mis insignias“ sin insignias conseguidas.

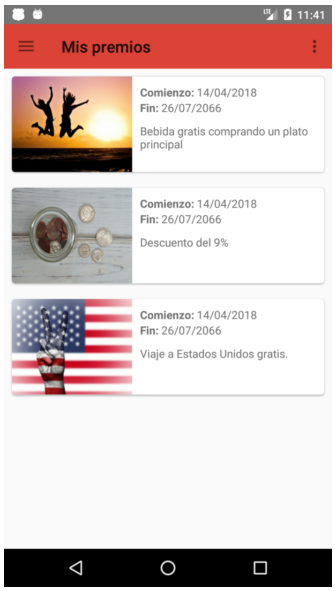


(b) “Mis insignias“ con insignias conseguidas.

Figura 3.53: Capturas de pantalla de “Mis insignias“.



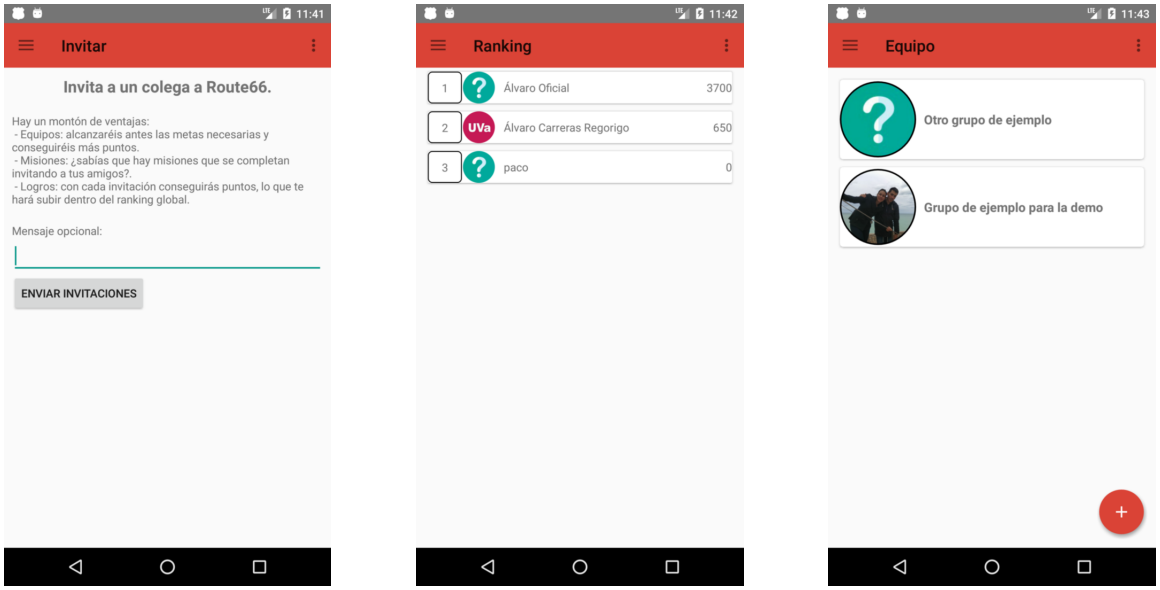
(a) “Mis premios“ sin premios conseguidos.



(b) “Mis premios“ con premios.

Figura 3.54: Capturas de pantalla de “Mis premios“.

La vista invitar permite incluir un mensaje personalizado que será adjuntado al correo electrónico que se envíe.

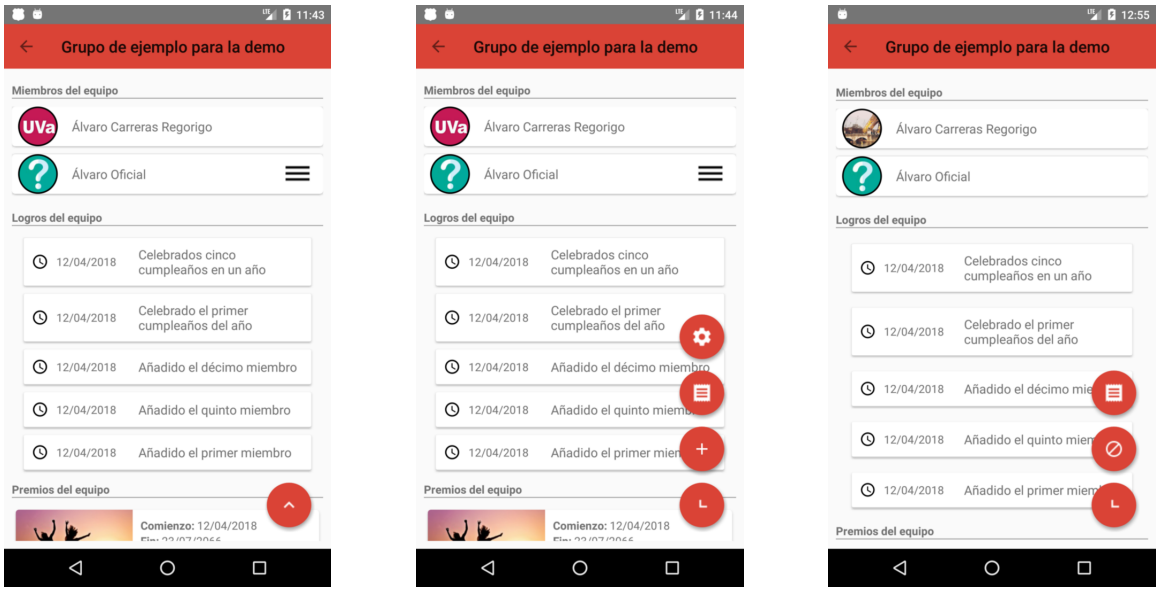


(a) Vista “Invitar“. (b) Vista “Ranking global“. (c) Vista “Equipos“.

Figura 3.55: Capturas de pantalla de “Invitar“, “Ranking global“ y “Equipos“.

3.7.2. Equipos

En cuanto a los equipos, la vista principal de uno concreto es la siguiente:



(a) Vista principal de un equipo (rol administrador). (b) Vista principal de un equipo con el menú de opciones deplegado (rol administrador). (c) Vista principal de un equipo (rol usuario normal).

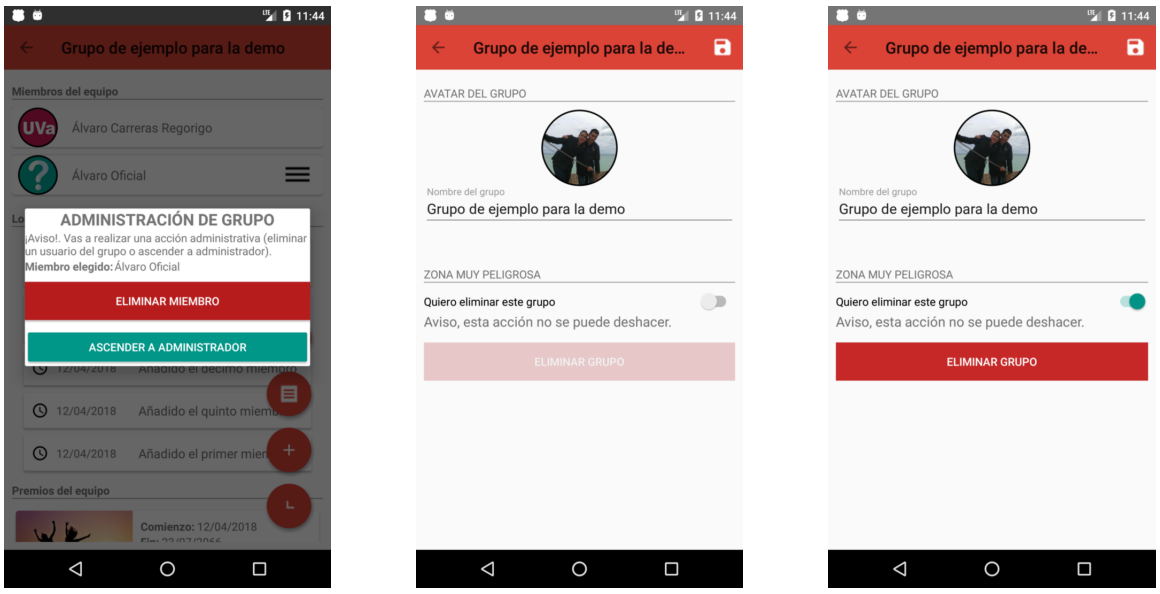
Figura 3.56: Capturas de pantalla de la vista que muestra los detalles de un equipo.

En el menú desplegado tenemos las siguientes opciones:

1. En caso de ser administrador:
   a) Configuración del equipo: permite editar la imagen de perfil (avatar) del equipo, así como cambiar el nombre del mismo e incluso, eliminarlo.
   b) Código *QR*: muestra el código *QR* del equipo.
   c) Añadir miembro: permite buscar a un miembro de la plataforma y añadirlo al equipo.
2. En caso de ser usuario normal:
   a) Código *QR*: muestra el código *QR* del equipo.
   b) Salir del equipo.

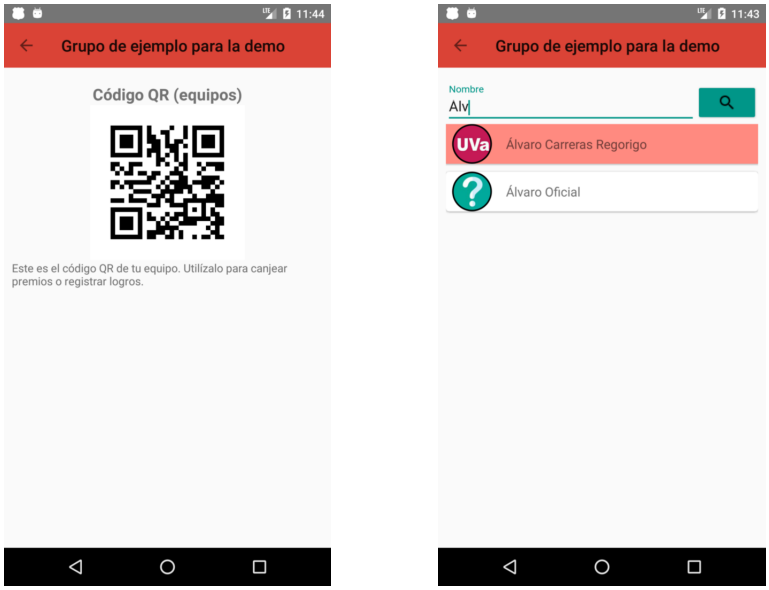
Por otro lado, siendo administrador, se da la posibilidad de ascender a administrador a otro miembro del equipo y de expulsar.

A continuación se muestran las capturas de pantalla de cada una de estas opciones:



(a) Vista “Gestionar usuario“. (b) Vista “Configuración del equipo“. (c) Vista “Configuración del equipo“.

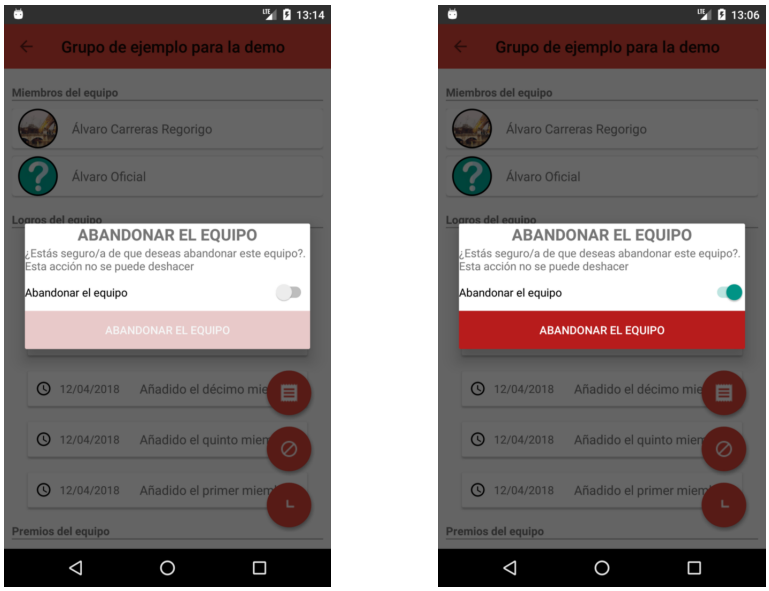
Figura 3.57: Capturas de pantalla de “Gestionar usuario“ y “Configuración del equipo“



(a) Vista “Código *QR* del equipo“. (b) Vista “Añadir miembro al equipo“.

Figura 3.58: Capturas de pantalla de “Código *QR* del equipo“ y “Añadir miembro al equipo“.

Por último, como rol de usuario normal, además de permitir mostrar el código *QR* (la vista es igual que en el caso de administrador), se permite salir de un grupo.

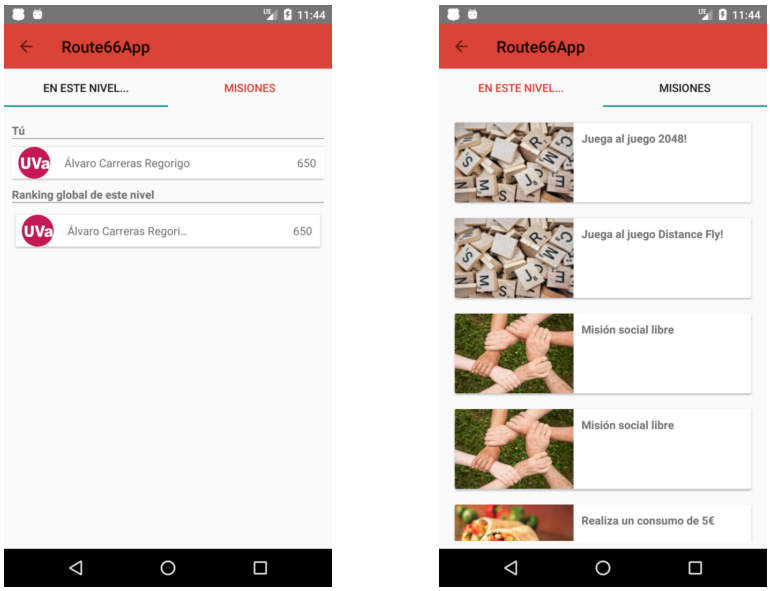


(a) Vista “Salir del equipo“. (b) Vista “Salir del equipo“.

Figura 3.59: Capturas de pantalla de “Salir del equipo“.

3.7.3. Niveles y misiones

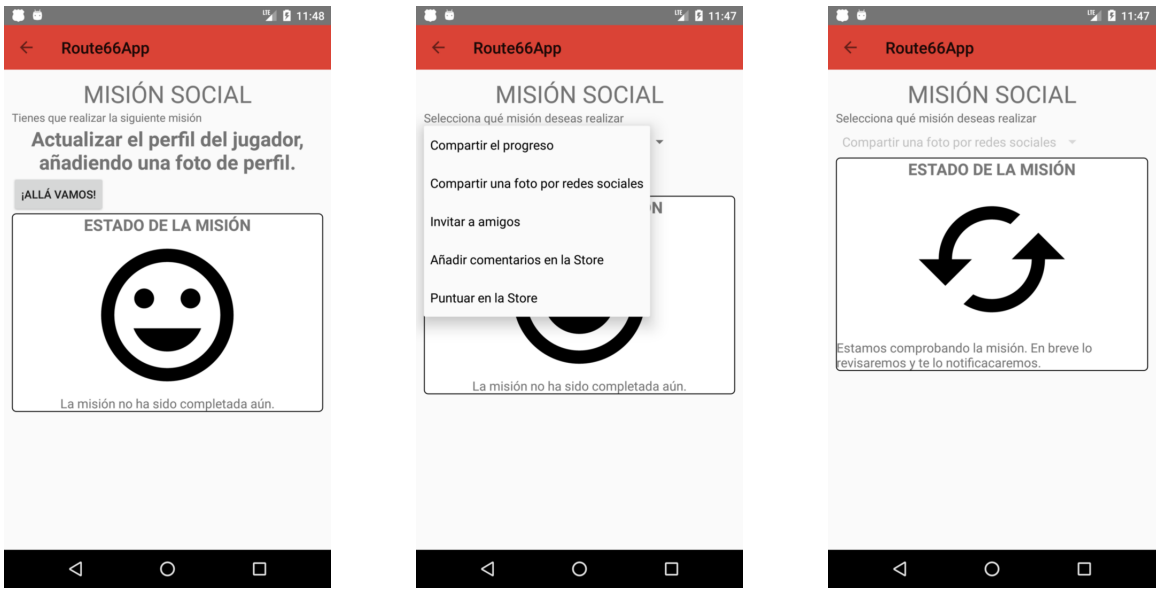
Cuando entramos en un nivel desde la vista principal de usuarios veremos las siguientes vistas:



(a) Vista “Miembros de un nivel“. (b) Vista “Misiones de un nivel“.

Figura 3.60: Capturas de pantalla de “Miembros de un nivel“ y “Misiones de un nivel“.

Finalmente, el apartado de misiones distingue entre su tipo: social, competitivo y consumo. Se adjuntan las imágenes de las mismas.

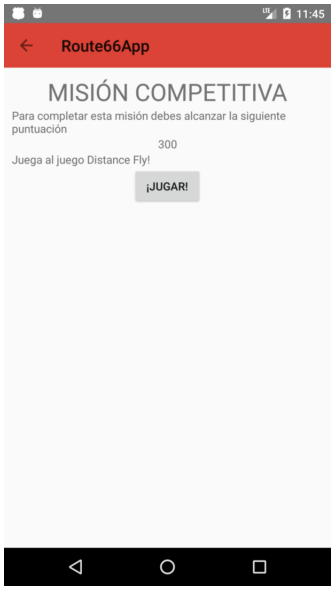


(a) Vista “Misión social“ restringida. Nivel *Chicago*.

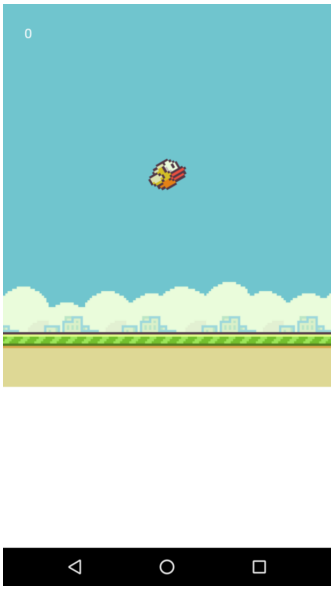
(b) Vista “Misión social“ libre.

(c) Vista “Misión social“ libre en estado de comprobación.

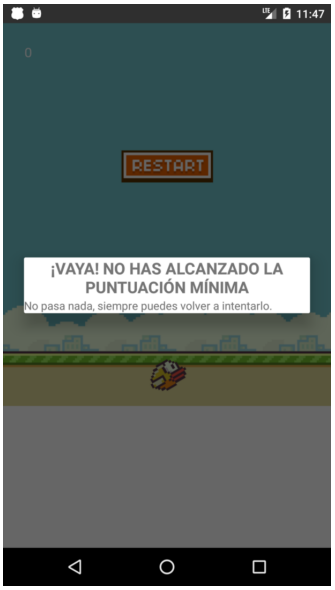
Figura 3.61: Capturas de pantalla de una misión de tipo social.



(a) Puntuación que se debe alcanzar en esta misión.



(b) Misión en progreso.



(c) Mensaje de no haber completado la misión.

Figura 3.62: Capturas de pantalla de una misión de tipo competitiva.

Figura 3.63: Captura de pantalla de una misión de tipo consumo.

Capítulo 4

Conclusiones y trabajo futuro

4.1. Conclusiones

Conclusiones de este Trabajo Fin de Grado:

1. Se ha elaborado un sistema informático basado en la ludificación a partir de una idea desarrollada en el T.F.G. de Dña. Cristina Martínez Martínez, alumna de nuestra Universidad.
2. Se ha implementado una aplicación *Android* que permite fidelizar clientes, por medio de la realización de una serie de misiones y la obtención de insignias y premios, con el objetivo de satisfacer los criterios de cada uno de los tipos de usuario de la teoría de Bartle [9].
3. Se ha diseñado y desarrollado un sistema de gestión de grupos o equipos, con el fin de añadir una capa aún más social a la aplicación.
4. Se han utilizado dos tecnologías totalmente nuevas y propuestas por la tutora del T.F.G. (*Kotlin* y *Firebase*), que han demostrado ser lo suficientemente maduras como para poder ser empleadas en el desarrollo de nuevas aplicaciones móviles.

4.2. Trabajo futuro

Se proponen las siguientes ideas para extender las funcionalidades de la aplicación:

1. Integrar el sistema de ludificación desarrollado en una aplicación que será utilizada en un ambiente real. Un ejemplo podría ser utilizar la cafetería de la Escuela de Ingeniería Informática para ello.
2. Al igual que se ha visto en el estudio de alternativas en el caso de *VIPS*, permitir pagar la cuenta directamente con saldo recargable. Con cada recarga, el usuario obtendría una serie de puntos por completar logros de recargas.
3. Poder realizar pedidos directamente desde la aplicación.
4. Realizar la aplicación para la plataforma iOS.
5. Traducir la aplicación a idiomas como el inglés, catalán, euskera, gallego, valenciano, etcétera.
6. Incluir más métodos de autenticación.

Capítulo 5

Webgrafía y Bibliografía

5.1. Webgrafía

[1] ACCENTURE *Why gamification is serious business*, Fecha de última visita: 6 de Noviembre de 2017 https://www.accenture.com/us-en/insight-outlook-why-gamification-is-serious-business.

[2] ANDROID DEVELOPERS OFFICIAL WEBSITE,*Dashboards. Platform Versions.* Fecha de última visita: 02 de Diciembre de 2017 https://developer.android.com/about/dashboards/index.html.

[3] ANDROID DEVELOPERS OFFICIAL WEBSITE. *Room Persistence Library*. Fecha de última visita: 23/01/2018 https://developer.android.com/topic/libraries/architecture/room.html.

[4] ANDROID REFERENCE GUIDE Aspectos fundamentales de la aplicación. Fecha de última visita 24/02/2018. https://developer.android.com/guide/components/fundamentals.html.

[5] BURGER KING,*Aplicación móvil Burger King para Android* https://play.google.com/store/apps/details?id=es.burgerking.android.

[6] CIRULLI, GABRIEL Repositiroio en *GitHub* del juego en *JavaScript* 2048. Fecha de última visita 05/03/2018. https://github.com/gabrielecirulli/2048.

[7] COMPRESSOR Repositorio en *GitHub* de *Compressor*. Fecha de última visita 01/03/2018. https://github.com/zetbaitsu/Compressor.

[8] DODENHOF, HENNING Repositorio en *GitHub* de *CircleImageView*. Fecha de última visita 01/03/2018. https://github.com/hdodenhof/CircleImageView.

[9] FERRAN ALTARRIBA BERTRAN. IEBS SCHOOL *Tipos de jugadores en Gamification: teoría Bartle*. Fecha de última visita: 6 de Noviembre de 2017, http://www.iebschool.com/blog/tipos-jugadores-gamification-2-innovacion/.

[10] FIREBASE ANALYTICS Documentación de *Firebase Analytics*. Fecha de última visita 01/03/2018. https://firebase.google.com/docs/analytics/?hl=es-419.

[11] FIREBASE AUTHENTICATION Documentación de *Firebase Authentication*. Fecha de última visita 01/03/2018. https://firebase.google.com/docs/auth/?hl=es-419.

[12] FIREBASE CLOUD MESSAGING Documentación de *Firebase Cloud Messaging*. Fecha de última visita 01/03/2018. https://firebase.google.com/docs/cloud-messaging/?hl=es-419.

[13] FIREBASE CRASHANALYTICS *Firebase Crashlytics. Introduction*. Fecha de última visita 01/03/2018. https://firebase.google.com/docs/crashlytics/?hl=es-419.

[14] FIREBASE FIRESTORE Documentación de *Firebase Firestore*. Fecha de última visita 01/03/2018. https://firebase.google.com/docs/firestore/?hl=es-419.

[15] FIREBASE FIRESTORE *FirebaseUI for Cloud Firestore*. Fecha de última visita 01/02/2018. https://github.com/firebase/FirebaseUI-Android/blob/master/firestore/README.md.

[16] FIREBASE INVITES Documentación de *Firebase Invites*. Fecha de última visita 01/03/2018. https://firebase.google.com/docs/invites/?hl=es-419.

[17] FIREBASE REMOTE CONFIG Documentación de *Firebase Remote Config*. Fecha de última visita 01/03/2018. https://firebase.google.com/docs/remote-config/?hl=es-419.

[18] FIREBASE STORAGE Documentación de *Firebase Storage*. Fecha de última visita 01/03/2018. https://firebase.google.com/docs/storage/?hl=es-419.

[19] FIREBASE Trabajar con arreglos, listas y conjuntos. Fecha de última visita 24/02/2018. https://firebase.google.com/docs/firestore/solutions/arrays?hl=es-419.

[20] FIREBASE. Página Web de *Firebase*. Fecha de última visita: 08/02/2018.

[21] FOSTER'S HOLLYWOOD,*Aplicación móvil Foster's Hollywood para Android* https://play.google.com/store/apps/details?id=com.zena.Fosters.

[22] GLIDE Repositorio de *Glide* en *GitHub*. Fecha de última visita 01/03/2018. https://github.com/bumptech/glide.

[23] GOOGLE PLAY SERVICES: AUTHENTICATION Documentación de *Google Play Services Authentication*. Fecha de última visita 01/03/2018. https://developers.google.com/identity/sign-in/android/start-integrating.

[24] INFOAUTÓNOMOS. EL ECONOMISTA. Tarifa plana de 50 € para autónomos, jóvenes y mayores de 30. Fecha de última visita: 22/01/2018 https://infoautonomos.eleconomista.es/seguridad-social/tarifa-plana-autonomos-50-euros-mayores-30-jovenes/.

[25] INFOJOBS. Estado del mercado laboral en España. Año 2016. pág 31. Fecha de última visita: 20/01/2018 http://tueligesinfojobs.net/informe-anual-infojobs-2016.pdf.

[26] KFC,*Aplicación móvil KFC para Android* https://play.google.com/store/apps/details?id=es.kfc.spain.

[27] KIWIJS *Blueprints* del motor de juegos *JavaScript KiwiJS*. Fecha de última visita 05/03/2018. http://www.kiwijs.org/blueprints/.

[28] KOTLIN PROGRAMMING LANGUAGE. INTELLIJ. Página Web del lenguaje de programación *Kotlin*. Fecha de última visita: 08/02/2018 https://kotlinlang.org/.

[29] MARC RODRÍGUEZ. IEBS COMUNIDAD *La ludificación como fenómeno social y herramienta global*, Fecha de última visita: 6 de Noviembre de 2017 http://comunidad.iebschool.com/pedrolopezugarte/2015/11/20/la-gamificacion-como-fenomeno-social-y-herramienta-global/.

[30] MARÍA BENITO *"Gamificación o cómo lograr que los empleados hagan un trabajo extra gratis"*, Fecha de última visita: 6 de Noviembre de 2017 https://www.elconfidencial.com/empresas/2014-04-27/gamificacion-o-como-lograr-que-los-empleados-hagan-un-trabajo-extra-gratis_121168/.

[31] MATERIAL DESIGN Sitio web informativo de *Material Design*. Fecha de última visita 1/02/2018. https://material.io/.

[32] MATERIALDOC Sitio web informativo de *Material Design*. Fecha de última visita 1/02/2018. https://materialdoc.com/.

[33] MATERIALSPINNER Repositorio en *GitHub* de *CircleImageView*. Fecha de última visita 01/03/2018. https://github.com/ganfra/MaterialSpinner.

[34] MCDONALDS,*Aplicación móvil McDonalds para Android* https://play.google.com/store/apps/details?id=com.mcdonalds.android.

[35] MCDONALDS. *Monopoly Ganador de McDonalds*. Fecha de última visita: 20/01/2018 https://www.mcdonalds.es/sites/default/files/mcdonalds_lanza_su_nuevo_monopoly_ganador.pdf.

[36] NATHAN, LEE. FREECODECAMP *Firebase Pros and Cons and How to Deal With the Cons*. Fecha de última visita 08/02/2018 https://medium.com/@leetheguy/firebase-pros-and-cons-ce37c766190a.

[37] OKHTTP Documentación de *OkHTTP*. Fecha de última visita 01/03/2018. http://square.github.io/okhttp/.

[38] OMNIUM GAMES. *BBVA Game: El mayor caso de éxito de la ludificación en España.*, Fecha de última visita: 21 de Noviembre de 2017 http://omniumgames.com/bbva-game-el-mayor-caso-de-exito-de-la-gamificacion-en-espana/.

[39] PANS & COMPANY,*Aplicación móvil Pans & Company para Android* https://play.google.com/store/apps/details?id=es.eatout.panscompany.

[40] PENZ, MIKE *MaterialDrawer*. Fecha de última visita 01/02/2018. https://github.com/mikepenz/MaterialDrawer.

[41] ROOM Documentación de *Room*. Introducción. Fecha de última visita 01/03/2018. https://developer.android.com/topic/libraries/architecture/room.html.

[42] TARIANYK, DMYTRO Repositorio en *GitHub* de *Floating Action Button*. Fecha de última visita 01/03/2018. https://github.com/Clans/FloatingActionButton.

[43] TIMOKHINA, VALERIA. DZONE. *The Pros and Cons of Kotlin for Android Development*. Fecha de última visita 08/02/2018. https://dzone.com/articles/the-pros-and-cons-of-kotlin-for-android-developmen.

[44] UNIFIED PROCESS FOR EDUCATION. UNIVERSIDAD POLITÉCNICA DE MONTREAL. Fecha de última visita: 25/11/2017 http://www.upedu.org/process/workers/wk_implm.htm.

[45] UNIVERSIDAD DE VALLADOLID. PARQUE CIENTÍFICO. Oficinas del Parque Científico de la Universidad de Valladolid. Fecha de última visita: 22/01/2018 http://www.parquecientificouva.es/Upload/ESPACIOS/CTTA/CTTA_FichaOficinas_Jun2013.pdf.

[46] VIPS,*Aplicación móvil VIPS para Android* https://play.google.com/store/apps/details?id=com.clubvips.app.

[47] ZHITNITSKY, ALEX. *The Top 10 Exception Types in Production Java Applications - Based on 1B Events*. Fecha de última visita: 08/02/2018 https://blog.takipi.com/the-top-10-exceptions-types-in-production-java-applications-based-on-1b-events.

[48] ZXING CORE: ANDROID Repositorio *GitHub* de *ZXing Core*. Fecha de última visita 01/03/2018. https://github.com/zxing/zxing.

5.2. Bibliografía

[49] ADIEGO RODRÍGUEZ, JOAQUÍN. Departamento de Informática de la UVa. Apuntes de la asignatura Sistemas Móviles. Curso 2017-2018. Kotlin.

[50] BOYER, RICK & MEW, KYLE. *Android Application Development Cookbook - Second Edition. Over 100 recipes to help you solve the most common problems faced by Android Developers today*. Editorial *PacktPub*.

[51] CRESPO CARVAJAL, YANIA; GONZALO TASIS, MARGARITA; HERNÁNDEZ DÍAZ, CARMEN; MARTÍNEZ MONÉS, ALEJANDRA Modelo vista controlador (MVC). Prácticas. Apuntes de la asignatura Interacción Persona-Computador. Curso 2015-2016. Departamento de Informática de la UVa. Universidad de Valladolid.

[52] DE LA FUENTE REDONDO, PABLO LUCIO. Departamento de Informática de la UVa. Apuntes de la asignatura Planificación y Gestión de Proyectos. Curso 2017-2018. Tema 2: Planificación de Proyectos.

[53] DE LA FUENTE REDONDO, PABLO LUCIO. Departamento de Informática de la UVa. Apuntes de la asignatura Planificación y Gestión de Proyectos. Curso 2017-2018. Tema 3: Proceso Unificado.

[54] HILLS, TED. *NoSQL and SQL Data Modelling. Bringing together data, semantics and software*. Editorial *Technics Publications*.

[55] MARTÍNEZ MARTÍNEZ, CRISTINA *T.F.G. "Estudio de ludificación en una empresa para mejorar la fidelización de los clientes"*. Curso 2016-2017. Grado en Ingeniería en Organización Industrial. Universidad de Valladolid.

[56] RUIZ CABALLERO, ANA *T.F.G. "Estudio de la ludificación de una empresa para incentivar la motivación. "*. Curso 2015-2016. Grado en Ingeniería en Organización Industrial. Universidad de Valladolid.

Parte I

Anexos

Anexo I: Reglas de seguridad en *Firebase*

Firestore

Se han establecido las siguientes reglas de seguridad que se explican detalladamente a continuación:

```
1  service cloud.firestore {
2    match /databases/{database}/documents {
3      match /{document=**} {
4        allow read, write: if request.auth.uid != null;
5      }
6      match /users/{userId}{
7       allow write: if request.auth.uid == userId;
8        match /achivements {
9        allow read, write: if request.auth.uid == userId;
10       }
11       match /insignias {
12       allow read, write: if request.auth.uid == userId;
13       }
14       match /prizes {
15       allow read, write: if request.auth.uid == userId;
16       }
17     }
18   }
19 }
```

Por defecto, *Firebase Firestore* deniega la lectura y escritura de todos los documentos. Solo si se cumple la regla de seguridad que corresponda se podrá proceder a la lectura o a la escritura.

Las reglas de seguridad anteriores:

- Impiden que un usuario no registrado pueda leer o escribir sobre la base de datos (línea 4).
- Impiden que un usuario pueda editar el perfil de otro usuario (línea 7).
- Impiden que un usuario pueda ver las insignias, premios, logros o misiones que haya completado otro usuario (líneas 9, 12 y 15).

Firebase Storage

Por otra parte, las reglas de seguridad del almacenamiento son las siguientes:

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null
        && request.resource.size < 2 * 1024 * 1024
          && request.resource.contentType.matches('image/.*');
    }
  }
}
```

Las reglas anteriores aseguran tres cosas sobre todo aquel archivo que se suba al servicio:

- Solo pueden subir imágenes usuarios autenticados.
- El tamaño máximo de un archivo es de 2 *MiB*.
- Todos los ficheros que se suban son de tipo imagen.