

## MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Multiplataforma

# GameHub – Comunidad de Videojuegos

**Autor:** Mario Alfonso Medina Macías

**Tutor:** Mario Gago

**Fecha de entrega:** 10/02/2023

**Convocatoria:** 1S – 2223

**Documentos del proyecto:**

[https://drive.google.com/drive/folders/1AHe6ztG7WQXSViRVNzWxsbKc\\_ihs1\\_U5?usp=share\\_link](https://drive.google.com/drive/folders/1AHe6ztG7WQXSViRVNzWxsbKc_ihs1_U5?usp=share_link)



## Índice de contenidos

<b>1. INTRODUCCIÓN .....</b>	<b>4</b>
1.1. Motivación .....	4
1.2. Abstract.....	5
1.3. Objetivos propuestos (generales y específicos) .....	5
1.4. Contexto laboral.....	6
<b>2. METODOLOGÍA USADA .....</b>	<b>7</b>
2.1.1 ¿En qué consiste Kanban? .....	7
2.1.2 Ventajas y Desventajas de la Metodología Kanban .....	8
2.2 Ciclo de vida del proyecto.....	9
2.3 Fases del diseño del proyecto.....	10
2.3.1 ¿En qué consiste el Modelo Iterativo Incremental? .....	10
<b>3. TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS EN EL PROYECTO .....</b>	<b>12</b>
<b>4. ESTIMACIÓN DE RECURSOS Y PLANIFICACIÓN .....</b>	<b>15</b>
4.1 Componentes de un diagrama de Gantt.....	15
<b>5. ANÁLISIS DEL PROYECTO .....</b>	<b>20</b>
5.1 Requisitos funcionales y no funcionales .....	20
5.2 Diagramas Entidad – Relación, Casos de uso y Clases .....	22
5.2.1 Diagrama Entidad – Relación .....	22
5.2.2 Diagrama de Casos de uso .....	23
5.2.3 Diagrama de Clases .....	27
<b>6. DISEÑO DEL PROYECTO .....</b>	<b>29</b>

<b>6.1 Mockup's y Diseños de vistas .....</b>	<b>29</b>
<b>6.2 Realización del proyecto, arquitectura y desarrollo de código .....</b>	<b>30</b>
6.2.1 Estructura de carpetas.....	31
6.2.2 SGBD, TablasNoSQL y Administración .....	33
6.2.3 Activity's, Fragments, Modelos y Layouts .....	34
<b>7. DESPLIEGUE Y PRUEBAS .....</b>	<b>46</b>
<b>8. CONCLUSIONES .....</b>	<b>49</b>
<b>9. VÍAS FUTURAS.....</b>	<b>51</b>
<b>10. BIBLIOGRAFÍA/WEBGRAFÍA.....</b>	<b>53</b>
<b>11. ANEXOS .....</b>	<b>55</b>
<b>11.1 Manual de Usuario .....</b>	<b>55</b>
0. Consideraciones Previas .....	55
1. Inicio de sesión .....	55
2. Registro de usuario.....	57
3. Gaming .....	58
4. Mensajes.....	59
5. Perfil .....	59
6. Informar de un error .....	60
<b>11.2 Glosario.....</b>	<b>62</b>

***En la normativa de proyectos vigente encontrarás una breve descripción de cada uno de estos apartados para saber qué información debes incluir en ellos***

# 1. Introducción

Game Hub – Comunidad de videojugadores es una aplicación móvil para el sistema operativo Android que ayuda a los usuarios relacionados con el mundo del “gaming” a ponerse en contacto entre ellos mismos para poder compartir su pasión por el mundo de los videojuegos en tiempo real, así como buscar compañero para jugar juegos multijugador, discutir sobre algún próximo lanzamiento, organizar torneos de algún título en específico, encontrar un compañero sentimental con gustos similares, o simplemente socializar y forjar una comunidad sana y abierta a poder introducir nuevas personas a esta afición como son los videojuegos.

Con dicha propuesta dictada, podemos intuir las diferentes tecnologías de las que tendremos que hacer uso durante el desarrollo de nuestro proyecto, entre todas ellas podríamos mencionar el lenguaje de marcado XML como principal responsable del aspecto visual y de diseño de nuestra APP Android, siguiendo con JSON para la estructura de nuestra base de datos en la nube a tiempo real con Firebase – Realtime Database. En cuanto al lenguaje de programación utilizado será Kotlin muy aconsejado por Google y su filosofía *Kotlin first* y su respectivo IDE de la mano de IntelliJ “Android Studio”, también para ayudarnos con algunos archivos específicos, usaremos el editor de código Visual Studio Code una potente herramienta, a veces incluso pareciéndose a un IDE, que nos permitirá ver código claramente con las funcionalidades justas que necesitamos en ese momento.

## 1.1. Motivación

La principal motivación que he tenido a la hora de emprender este proyecto es primero de todo generar una aplicación Android, ya que a día de hoy un gran porcentaje de la población mundial usa dispositivos móviles antes que un sistema informático de escritorio como pudieran ser los ordenadores personales, y dentro de estos, el sistema operativo en telefonía móvil Android reina sobre un 80% aproximadamente de usuarios frente a otros sistemas operativos como iOS de la compañía Apple o Windows Phone en desuso de Microsoft.

Además de todo ello, el poder hacer una aplicación que se relacione con una de mis mayores aficiones como son los videojuegos, unido a la programación con Kotlin y Android Studio para crear un punto de encuentro entre jugadores de videojuegos y comunicarse entre sí es una idea que realmente me fascina y me genera una oportunidad de poder recibir ingresos por el desarrollo de mi aplicación en la tienda de aplicaciones de Google como es la Play Store.

## 1.2. Abstract

Introducing GameHub - Find your ideal gaming partner.

Our application provides a database of active players of different video games with whom you can contact at any time in real time.

We have a card system where you will see the face of each of the players registered in GameHub so that you are the one who decides to send that first message or to be able to see the contact information of that person that you think would suit you in order to organize a game. multiplayer, talking about a specific video game title, finding your ideal partner with similar tastes to yours in the world of video games on any platform, even getting to know a romantic partner with whom you can enjoy your favorite hobby.

GameHub is still in development so we provide a system to contact the development team in case you experience any bugs in the app so we can fix it as soon as possible.

In addition, GameHub is a totally secure platform against cyberattacks since it uses a database signed by Google such as Firebase. Also, GameHub - Find your ideal video game partner, is under constant surveillance by our IT team to avoid harassment among users registered on the platform, thus avoiding any bad experience that the average user may experience on the internet.

\*Currently GameHub - Find your ideal video game partner is only in the Spanish language, but we are working tirelessly so that English language speakers can enjoy our application as soon as possible without the limitation that the language barrier may entail. If this is your case, please stay tuned for future updates.

With nothing more to say, we hope you have a pleasant experience at GameHub - Find your ideal gaming partner and welcome you to our revolutionary app for gamers.

## 1.3. Objetivos propuestos (generales y específicos)

El objetivo fundamental de GameHub es poder mostrar una serie de usuarios registrados en la aplicación con sus datos de contacto para que el usuario que esté usando el aplicativo en ese momento tenga información con la que poder referirse al otro usuario y valorar si es la persona que más se acomoda a este decidiendo así ponerse en contacto a través de la aplicación o en su defecto mandando un correo electrónico a esa persona indicada.

Para lograr esto el usuario deberá de poder comunicarse con la aplicación móvil de distinta manera según quien haya iniciado sesión, lo que supone tener que pasar por un sistema de

seguridad OTP (código de seguridad recibido mediante mensaje de texto en el teléfono móvil) que conlleva inherentemente una función para comprobación y autenticación de aquellos de usuarios registrados en GameHub. Proporcionando así una aplicación totalmente dinámica e interactiva a tiempo real, en la que la interactividad que tendrán los usuarios sea fluida y genere los vínculos que podrán forjarse para cada una de estas distintas personas entusiasmadas por el ámbito de los videojuegos.

Ampliando los objetivos generales indagaremos más en algunos aspectos específicos que deberá tener nuestra aplicación:

- Listado en forma de tarjetas de cada usuario en cuestión que esté registrado en nuestra aplicación.
- Formulario de contacto para poder informar de bugs, crasheos u otros problemas técnicos que pudiera experimentar el usuario con GameHub.
- Sistema de mensajería en tiempo real que permita a los usuarios intercambiar mensajes a modo de chat estándar para poder conocerse y concretar actividades más específicas.
- Modelo de perfil de usuario que nos permita editar la información de residencia o datos de contacto facilitando así la información a distintos usuarios y que la información de cada uno de ellos sea siempre integral y actualizada.

#### 1.4. Contexto laboral

La elaboración y salida de GameHub al mercado de aplicaciones Android es una solución perfecta para todas aquellas personas que no conocen o no tienen a disposición otra persona con la que compartir sus intereses en el mundo del gaming actual, esto lleva a que muchas personas tengan que registrarse en sitios web o en aplicaciones poco específicas a sus necesidades para encontrar personas acordes a este tipo de perfil. Esto hace que GameHub tenga la oportunidad de diferenciarse en el mercado de la Play Store de Google. A partir de esto GameHub puede llegar a proporcionar beneficios a corto y largo plazo con la inclusión de suscripciones para funciones premium y la inclusión de banners de anuncios monetizados por número de visualizaciones que influirá en la posición del ranking de aplicaciones y que sea más visible a ojos de posibles usuarios y posibles descargas de GameHub (ASO). Esto además promocionará al desarrollador de la aplicación haciendo posible que las nuevas aplicaciones que se desarrollen en el futuro tengan más repercusión debido al interés que generó GameHub, mientras dentro de un plano pesimista en el caso de que la aplicación no

generase beneficios económicos es una muy buena demostración para un portfolio de proyectos de programación bajo mi marca que podría atraer inversores o incluso facilitar un contrato en una compañía del sector IT.

## 2. Metodología usada

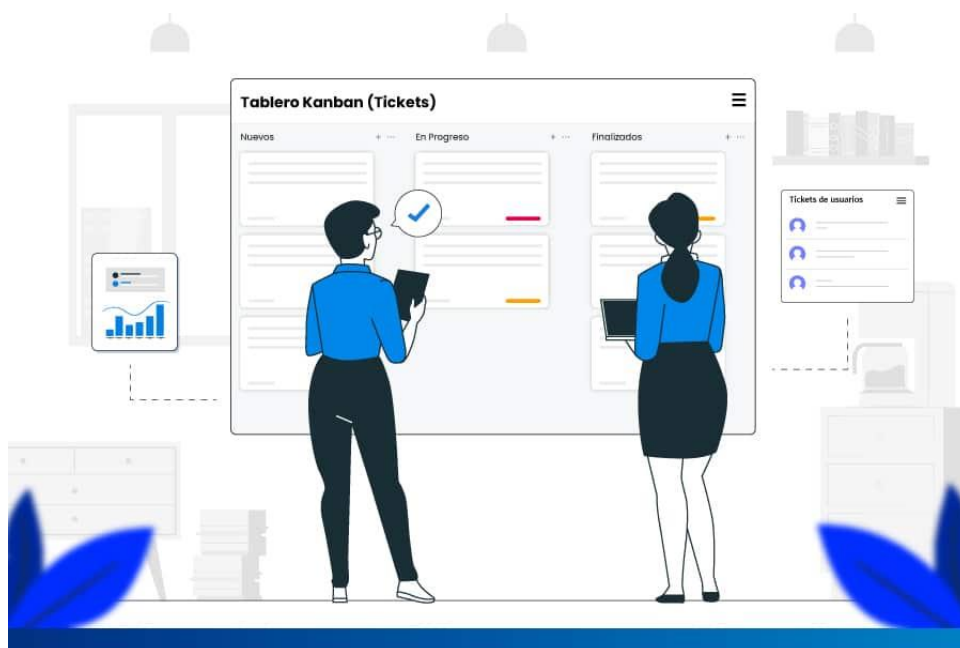
Para poder organizar un desarrollo cumpliendo tareas y fechas de entrega, he adaptado una metodología japonesa nacida en Toyota que ha sido clave los últimos años para las grandes consultoras de software y empresas relacionadas con el cumplimiento de objetivos y tareas en el sector de la Tecnología, pudiendo así organizar el flujo de trabajo entre diferentes hitos y repartiéndose el trabajo de unos desarrolladores a otros y aunque yo sea el único responsable de la elaboración del código del proyecto me ha sido muy útil, como he comentado antes, para poder tener rápidamente un resumen visual de cómo está evolucionando el proyecto y de aquellos aspectos que quedan por pulir, terminar o actualizar; dicha metodología se le conoce como Kanban.

### 2.1.1 ¿En qué consiste Kanban?

La metodología Kanban es un método de trabajo enfocado a grandes desarrollos que consiste en un tablero dividido en tres columnas: Tareas a realizar, Tareas en Proceso y Tareas Finalizadas. Dentro de cada una de estas columnas y a modo de celdas se les añade los hitos en su respectiva categoría a forma de tarjetas, así el o los encargados de dicha tarea irán pasando su asignación de tareas entre las diferentes columnas del panel Kanban.

Esta metodología fue ideada por un ingeniero japonés, en la fábrica de automoción de Toyota en el año 1940. Su método de trabajo fue tan revolucionario y eficiente que en los años venideros fue adaptada por la industria del software y elaboración de soluciones informáticas en medianas y grandes empresas mundiales.

## 2.1.2 Ventajas y Desventajas de la Metodología Kanban



*Ilustración 1: Representación Kanban. Fuente: Google Images.*

La metodología Kanban conlleva muchas ventajas que la llevaron a convertirse en un estándar en la industria, pero también algunos inconvenientes, por ello vamos a enumerar algunos de estos aspectos:

### **Ventajas:**

- **Análisis del rendimiento del equipo de trabajo:** la metodología Kanban nos permita analizar la eficiencia de los trabajadores y de las tareas, así como identificar problemas que estén afectando al rendimiento de los hitos a cumplir.
- **Establecer y cumplir las *deadlines* (fechas estipuladas) de cada tarea:** con Kanban podemos tener un control de la planificación establecida en un principio por el jefe de proyecto o de aquella persona que sea responsable de la dirección del desarrollo en cuestión.
- **Dosificación y síntesis del trabajo a realizar por los distintos trabajadores:** este apartado es el más valioso de la forma de planificar con Kanban ya que se detalla de una forma simple y concisa lo que cada trabajador debe de realizar en las diferentes fases del proyecto, evitando así sobrecarga de trabajo y síntomas del burnout de los trabajadores.



- **Incremento en la calidad del trabajo realizado:** gracias a todas estas ventajas que nos proporciona Kanban, el resultado del producto es generalmente superior en estándares de calidad y seguridad.

#### Desventajas:

- **Incremento del coste de la producción:** si se trata de un proyecto voluminoso y con mucho flujo de tareas resultará muy caótico para los desarrolladores además conllevará un aumento de coste para todo el almacenamiento de tablas Kanban.
- **No admite incrementar en gran medida el número de tareas:** en trabajos inmensos no es aconsejable abusar de la metodología Kanban ya que el número de tareas es difícilmente modificable en estos casos y habría trabajadores que se mantienen realizando una tarea grande y sin embargo otros quedarían desocupados de trabajo.
- **Dificultad de adaptación a cambios de gestión inesperados:** Kanban como ya hemos visto no admite fácilmente modificaciones en los cambios de planificación de proyecto ya que si fuera así podríamos causar un desbordamiento del volumen de trabajo.

## 2.2 Ciclo de vida del proyecto

A la hora de haber aceptado la realización de un proyecto, las empresas tienden a seguir un estándar de fases diferentes dependiendo de en cuál de ellas se encuentre el ciclo de vida del proyecto en cuestión. Se realizan tareas distintas en cada una de estas fases y las podemos dividir en:

- **Inicio:** el inicio de un proyecto de software es la primera fase del desarrollo de este, es ahí donde se valora la viabilidad y el valor del proyecto a realizar, se genera un documento de caso de negocio y se hace un estudio de viabilidad analizando los costos y haciendo un cronograma equilibrando los requisitos para pasar a la fase de planificación.
- **Planificación:** una vez el proyecto haya sido aprobado en la fase de inicio es momento de crear un plan sólido que agrupe el tiempo que se le planea dedicar al desarrollo y los costes de este, en resumen, crear las estrategias necesarias que establezcan un camino a seguir en cuanto al coste, los riesgos y beneficios del proyecto.
- **Ejecución:** una vez se haya realizado toda la planificación pasamos al propio desarrollo del proyecto y a la producción de código. Es importante que en esta fase exista una buena comunicación por parte del todo el equipo para poder gestionar el

riesgo de implementación, cambios, actualizaciones del desarrollo. Esta puesta en marcha es indispensable realizarla con una planificación sólida y estudiada.

- **Supervisión y Control:** en esta fase del ciclo de vida del proyecto se enfoca en la monitorización de todos aquellos procesos que se van implementando en la fase de ejecución, es así inherente a la fase de ejecución, no se conciben de forma separada y es necesario que ambos equipos trabajen de forma simultánea para evitar riesgos o identificar cambios necesarios antes de entregar el producto al cliente, cuidando así la calidad del trabajo realizado.
- **Cierre:** fase de finalización del proyecto donde se reúnen ambas partes interesadas en el proyecto, consultora y cliente donde se acuerdan y verifican que se han cumplido los objetivos. Si no se ha llegado a un acuerdo, se negocian nuevos requisitos y se vuelve a la fase de planificación del proyecto. Una vez el acuerdo queda cerrado se termina documentando el proyecto y se comunica al cliente que el proyecto ha concluido.

## 2.3 Fases del diseño del proyecto

Una vez tenemos una metodología (Kanban) y un ciclo de proyecto bien implementado es necesario ejecutarlo desde unas fases de desarrollo específicas que se adecue al volumen del trabajo, los objetivos de este y al equipo de trabajo. Existen diferentes modelos y formas de trabajar para un proyecto en ejecución de desarrollo, debido a las características de mi proyecto y tomándome en cuenta a mí mismo como única persona frente a la dirección y producción del código del proyecto he concluido que el modelo iterativo incremental es el que más se ajusta al proceso de implementación para GameHub.

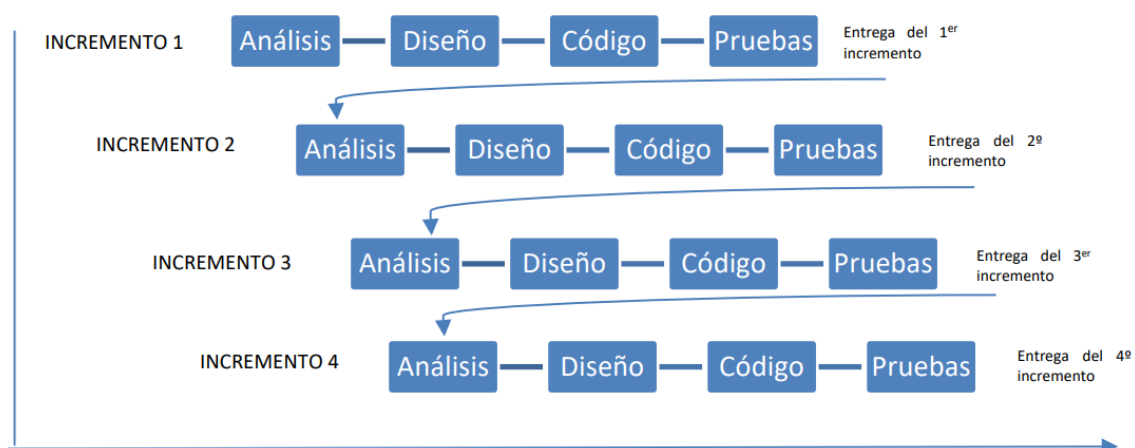
### 2.3.1 ¿En qué consiste el Modelo Iterativo Incremental?

El modelo iterativo incremental es un modelo de trabajo que está basado en varios ciclos del modelo en cascada retroalimentados y aplicados reiteradamente y consecutivamente hasta que finalice la última funcionalidad a desarrollar.

Empezamos con un pequeño análisis de aquella funcionalidad que queramos implementar, una vez hecho esto hacemos un breve diseño de lo que se necesita desarrollar para tener una referencia teórica. Una vez pasadas estas dos fases empezamos a desarrollar la aplicación escribiendo el código hasta que tengamos esta funcionalidad objetiva terminada. Por último, terminamos el ciclo con unas pruebas del código en producción. Si este paso es

satisfactorio deberemos agregar un incremento al bucle de funcionalidades empezando otra vez desde la fase del análisis de la siguiente funcionalidad, en caso contrario, deberemos retroceder y volver a desarrollar o corregir el código para de nuevo pasar unas pruebas e incrementar el ciclo de modelo iterativo incremental.

Este modelo de trabajo es perfecto para mis necesidades ya que es fácil realizar modificaciones cuando sea necesario, simplemente tendremos que deshacer un ciclo de la cascada y además resulta barato ya que no se realizan grandes funcionalidades a la vez con lo cual no ponemos en riesgo el desarrollo del proyecto si algo no sale bien, además no es necesario saber desde el principio exactamente todas las funciones que tiene que tener el aplicativo y el cliente puede maniobrar más y tener una visión de si el producto está siendo desarrollado como él esperaba y en el caso de que no sea así no se habrá perdido tanto trabajo como en otros modelos de trabajo, aunque esto conlleva que sea difícil estimar el trabajo y el coste del producto final, además como otro punto negativo se corre el riesgo de no acabar nunca el desarrollo queriendo aplicar cada vez de cada iteración funciones nuevas y no llegar a las fechas acordadas de entrega.



*Ilustración 2: Modelo Iterativo Incremental. Fuente: Material Didáctico de Entornos de Desarrollo.*

### 3. Tecnologías y herramientas utilizadas en el proyecto

Todas las tecnologías y herramientas que han sido usadas para la realización de este proyecto de fin de ciclo (PFC) de Desarrollo de Aplicaciones Multiplataforma (DAM) están relacionadas o han sido estudiadas durante todos aquellos módulos de los que el ciclo formativo de grado superior se compone, las cuales enumeraremos a continuación:

❖ Lenguaje de marcado y de estilo:

- **XML:** XML (eXtensible Markup Language) Lenguaje de Marcado Extensible es un metalenguaje que permite desarrollar variantes como otros lenguajes de marcas implementado y desarrollado por la World Wide Web Consortium (W3C) viene basado en el lenguaje SGML y permite almacenar datos a diferencia de otros lenguajes de marcas, también permite dar estilos gracias al IDE Android Studio en aplicaciones Android.
- **JSON:** JSON (JavaScript Object Notation) es un formato de texto sencillo para la interacción y el intercambio de datos de una aplicación. Se compone de un subconjunto de la notación de objetos de JavaScript y se estandarizó como alternativa a XML en el almacenamiento de datos de objetos, es utilizado en API's y BDDNoSQL (Bases de datos no relacionales).

❖ Lenguaje de programación:

- **Kotlin:** Kotlin es un lenguaje de programación de tipado estático que hace uso de la JVM (Máquina Virtual de Java) y que puede ser transpilado a código fuente de JavaScript. Fue desarrollado principalmente por JetBrains (empresa desarrolladora del IDE para Java ubicada en San Petesburgo, Rusia). Su nombre hace referencia a la isla Kotlin situada próxima a San Petesburgo. Está considerada por Google como el lenguaje de programación oficial de Android siguiendo su filosofía Kotlin First (Kotlin primero) ya que aún es posible desarrollar Apps en Java, sin embargo, Kotlin es la que primero recibe actualizaciones y soporte por parte de la compañía de Mountain View.

❖ Entorno de desarrollo integrado (IDE) y Editor de código:

- **Android Studio:** Android Studio es el IDE (Entorno de Desarrollo Integrado) oficial de Google y desarrollado también en colaboración con la compañía Rusa IntelliJ y basado en su IDE IntelliJ IDEA.
- **Visual Studio Code:** Visual Studio Code se trata de un editor de código fuente desarrollado por la compañía americana Microsoft, esta herramienta es de

código libre, lo que permite a desarrolladores optimizarla y desarrollar plugins aunque sean ajenos a Microsoft. Está escrita en Electron framework basado en Node.js y tecnologías web.

❖ Sistema Gestor de Bases de Datos No Relacional:

- **Firebase – Realtime Database:** Firebase es una plataforma enfocada en bases de datos no relacionales en la nube acompañada con Google Cloud Platform que sincroniza, actualiza y genera análisis en tiempo real bases de datos de tanto aplicaciones web como de aplicaciones móviles.

❖ Sistema Operativo:

- **Windows 10/11:** Sistemas operativos privativos para ordenadores personales (PC) y tablets con procesadores x64. Escrito en C/C++/C# y ensamblador ofrece herramientas a nivel usuario y a nivel de desarrollador. Es el sistema operativo predominante a nivel doméstico.

❖ Navegador Web:

- **Google Chrome:** es un navegador web de código cerrado creado y desarrollado por Google en base al proyecto de código abierto Chromium y de uso gratuito.

❖ Servicios de Google:

- **Google Apps:** ahora llamado Google Workspace, ha pasado por diferentes nombres como G Suite o Google Apps for Work, es un servicio de suite ofimática en la nube que proporciona aspectos ofimáticos de la mano de Google, así como Gmail como gestor web de correo electrónico, Google Drive sistema de almacenamiento en la nube, Pages como hojas de cálculo, etc...
- **Firebase Console:** portal y plataforma donde se organizan y se administran los distintos proyectos enlazados aplicando diferentes directivas y servicios según las necesidades del producto relacionado con las Bases de datos no relacionales en la nube.
- **Motor de búsqueda de Google.**

❖ Herramientas de control de versiones:

- **Git:** es un software multiplataforma (Windows, Linux, MacOS) dedicado al control de versiones de proyectos software es reconocido como estándar en control de versiones en un gran porcentaje de empresas a nivel mundial.
- **GitHub:** sitio web enfocado a la industria del software que sirve para alojar millones de proyectos Git en repositorios remotos, la empresa fue comprada hace varios años por Microsoft, debido a ello existen planes de pago con

funciones restringidas para usuarios que no aporten suscripción. Modelo de negocio freemium

❖ Librerías:

- **Glide:** glide es una librería y herramienta rápida y eficiente que permite abrir elementos multimedia y cargar imágenes en aplicaciones cacheándolas en memoria y mostrándolas en una interfaz sencilla y amigable para el usuario medio.
- **Font Awesome:** framework enfocado a iconos sólidos para el usuario de imágenes vectoriales como diseños XML.

\*Ha sido necesario el uso de estas dos herramientas con la justificación del problema de cargar imágenes en Android sin tener que añadirle más peso al aplicativo Android en su código fuente, cuidando así su estilo intuitivo y descarga liviana para la comodidad del usuario final.

## 4. Estimación de recursos y planificación

La realización de proyectos, en especial proyectos relacionados con el desarrollo de software, es de vital importancia para el control de estas, cumplimiento de fechas asignadas y evitar la sobrecarga de trabajo tanto de los directivos como los desarrolladores de la aplicación que se disponga de una planificación a seguir durante el transcurso del producto y así conseguir un ambiente y condiciones de trabajos sanas y respetables con dicha organización del equipo en cuestión.

El diagrama de Gantt es una herramienta muy útil para planificar este tipo de proyectos y que proporciona una vista general y una serie de tareas que estarán programadas en este y las fechas límites.

### 4.1 Componentes de un diagrama de Gantt

Fechas estipuladas: son aquellos días que podemos estimar que nos llevará terminar las funcionalidades o las fases de la aplicación en consonancia con la entrega al cliente o en este caso al tribunal para la defensa del proyecto.

Nombre de la tarea a realizar: aquí haremos referencia a sobre lo que estamos trabajando, pueden ser fases del ciclo del proyecto o aquella funcionalidad importante que sea necesaria para avanzar en el desarrollo.

Plazos de tiempo previstos y reales: este es uno de los puntos más importantes del diagrama de Gantt ya que nos proporciona la información y nos sirve como guía para saber dónde deberíamos estar en el proyecto, si llevamos atrasos o por si el contrario la estimación de las tareas a realizar se completaron antes de tiempo.

Progreso en cada una de las tareas: este apartado representa cómo estamos avanzando en cada uno de los hitos a cumplir para dar por finalizado el desarrollo de nuestro proyecto.

Comprendiendo lo que es un diagrama de Gantt veamos algunas ventajas y desventajas del uso de esta herramienta tan útil:

#### Ventajas:

- **Gestión más o menos realista del tiempo del desarrollo:** este punto es imprescindible en un diagrama de Gantt, este nos permite analizar el tiempo y las

fechas de todo el proceso del proyecto haciéndonos una idea de cómo deberíamos de estar avanzando durante este.

- **Vista sencilla del progreso del desarrollo:** nos clarifica de una forma muy resumida toda la información que contiene esta herramienta. Es más fácil ver las tareas en forma de diagrama que con un documento de texto con detalle y demasiado texto que leer.
- **Flexibilidad sobre las fechas:** si no son cambios importantes el diagrama de Gantt es fácilmente manipulable para acordar nuevas fechas para las distintas fases o para ver reflejado la estimación real que nos está llevando todas las partes de nuestro proyecto.
- **Comunicación del progreso con el equipo de trabajo:** al estar a la vista de todos los empleados es fácil discutir sobre las tareas asignadas a cada uno y si es realista conforme a las fechas acordadas en el diagrama dando así más refuerzo a la virtud de poder modificar el diagrama a conveniencia de la situación.

#### Desventajas:

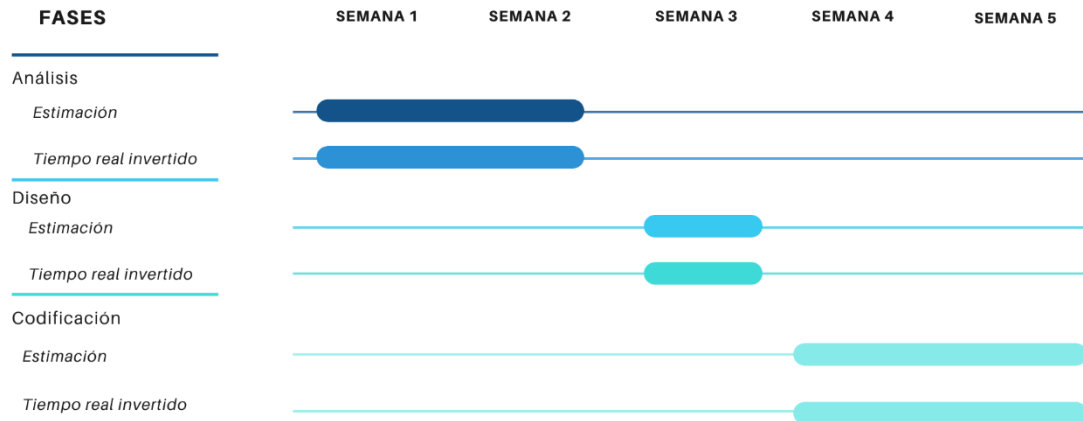
- **Diagramas complejos debido al volumen de trabajo:** un diagrama de Gantt en proyectos de una gran envergadura puede provocar caos entre los trabajadores ya que no estaría cumpliendo su función principal la cual es poder ver de un vistazo y estar informado de las fechas límites y esperadas en el desarrollo.
- **Linealidad del proyecto:** es muy importante definir todos aquellos requisitos de nuestro trabajo para que no debamos emplear más trabajo del deseado a modificar el diagrama o incluso que quede obsoleto puesto a que se han realizado cambios significativos a los previstos.
- **Trabajo en actualizar el diagrama según cambios y avances del proyecto:** como explicábamos en el anterior apartado en proyectos que varíen mucho de requisitos y sean voluminosos puede ser un problema para el responsable de la gestión del diagrama ya que se estarían empleando más recursos de los necesarios en mantener el diagrama de una forma coherente y legible.

Por esto he decidido incluir en mi Proyecto de Fin de Ciclo (PFC) esta forma de planificación.



## Diagrama de Gantt - Programación PFC

Mes de Diciembre



## Diagrama de Gantt - Programación PFC

Mes de Enero

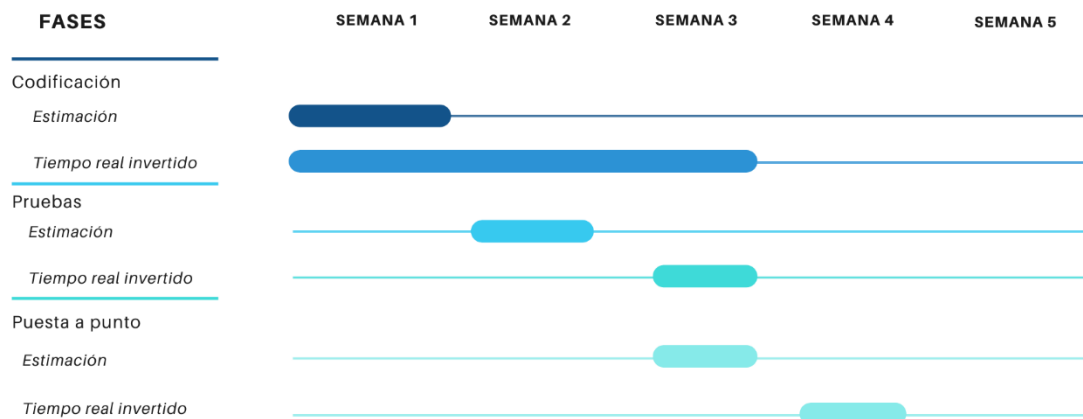


Ilustración 3, 4: Representación diagrama de Gantt. Fuente: elaboración propia en

“<https://www.canva.com/>”

Como hemos podido ver en el diagrama de Gantt que he realizado vamos a repasar y especificar más cómo han sido las fases de desarrollo para GameHub:

### ❖ Análisis:

- Estimación: Semana 1 de Diciembre – Semana 2 de Diciembre (dos semanas)

- Tiempo real invertido: Semana 1 de Diciembre – Semana 2 de Diciembre (dos semanas)

Dado a que decidí hacer la entrega de mi proyecto de fin de ciclo en desarrollo de aplicaciones multiplataforma en la convocatoria extraordinaria tuve tiempo de plantearme un buen análisis realista, aunque de grandes proporciones para GameHub durante un par de semanas, podemos decir entonces que se cumplieron los plazos estipulados para esta primera fase del proyecto.

#### ❖ Diseño:

- Estimación: Semana 3 de Diciembre (una semana)
- Tiempo real invertido: Semana 3 de Diciembre (una semana)

Considero que uno de mis puntos fuertes a la hora de desarrollar una aplicación móvil es precisamente el diseño así que pude cumplir bien también en esta fase con las fechas marcadas, generando un diseño minimalista, sobrio, intuitivo y agradable a ojos del usuario final.

#### ❖ Codificación:

- Estimación: Semana 4 de Diciembre – Semana 1 de Enero (tres semanas)
- Tiempo real invertido: Semana 4 de Diciembre – Semana 3 de Enero (cinco semanas)

En la fase de codificación, en mi opinión una de las más difíciles de ejecutar (si no la que más) empecé a tener problemas con las previsiones de tiempo. Tenía que empezar a dedicar tiempo a otros estudios como la universidad y la alta complejidad de las funcionalidades de mi aplicación tampoco me lo pusieron fácil. Si bien no cumplí con las fechas estipuladas, se puede decir que conseguí superar la fase de codificación satisfactoriamente, aunque con un retraso de dos semanas.

#### ❖ Pruebas:

- Estimación: Semana 2 de Enero (una semana)
- Tiempo real invertido: Semana 3 de enero (una semana)

La fase de pruebas tuvo un inicio retrasado debido a la fase de codificación, pero sí se cumplió la estimación de la duración de esta (una semana). No me llevó mucho tiempo debido a los conocimientos adquiridos en la exhaustiva fase de codificación.

#### ❖ Puesta a punto:

- Estimación: Semana 3 de Enero (una semana)

➤ Tiempo real invertido: Semana 4 de Enero (una semana)

Como en la fase de pruebas fue posible hacerla únicamente en una semana, pero de nuevo llevábamos un desfase de tiempo de una semana debido a la ya mencionada fase de codificación. Tanto la puesta a punto como las pruebas de mi aplicación iniciaron su recorrido simultáneamente con la fase de codificación para que la aplicación estuviera lista para la fecha de entrega ya que aún me quedaría por realizar la memoria escrita del proyecto, el PowerPoint de presentación y la videodefensa.

## 5. Análisis del proyecto

### 5.1 Requisitos funcionales y no funcionales

Para poder hacer un correcto análisis de la aplicación así como definir diagramas de clases, casos de uso o de entidad relación, es necesario tener en cuenta de una manera clara y exhaustiva tanto los requisitos funcionales (funcionalidades de la aplicación que se ajuste a las necesidades del usuario) como los requisitos no funcionales (aquellos aspectos no relacionados directamente con la funcionalidad que proporciona la aplicación pero que sí son necesarios para la integridad de la aplicación y ejecución de esta de manera segura).

Por ello, empezaremos listando todos los requisitos necesarios en GameHub:

#### ❖ **Requisitos funcionales:**

- **Registro de usuario:** todo usuario de GameHub deberá pasar por un proceso de registro donde se almacenen sus datos en una BBDD no relacional, para poder luego estar identificado y poder interactuar con la aplicación de manera personalizada.
- **Inicio de sesión de usuario:** el usuario podrá autenticarse a través de un formulario de inicio de sesión donde se comprobarán dichas credenciales en la BBDD para tener acceso al aplicativo y todas sus funcionalidades.
- **Navegación entre las vistas a través del menú de navegación:** los usuarios una vez ingresados en la aplicación podrán hacer uso del menú de navegación e interactuar con la interfaz gráfica que proporciona la aplicación Android.
- **Uso del menú contextual lateral de opciones:** el menú de desplazamiento lateral mostrará opciones adicionales al usuario que haya sido autenticado dando la oportunidad a este de acceder a aún más funcionalidades del aplicativo.
- **Visualización y desplazamiento de tarjetas de usuarios:** los usuarios podrán conocer a distintos tipos de personas a través de un sistema de tarjetas donde se mostrarán correctamente otros perfiles de usuarios mostrando su imagen de perfil, nombre y correo electrónico de contacto que previamente rellenaron en el registro o en la edición de dichos perfiles.
- **Formulario de Informe de errores:** funcionalidad adicional que permite a todo usuario de la App ponerse en contacto con el equipo de desarrollo para el reporte de errores dentro de la aplicación, informando del error y adjuntando imágenes declaratorias del problema en cuestión.

- **Función de comunicación entre usuarios (chat):** función principal de la aplicación donde los usuarios a través del sistema de tarjetas pueden ponerse en contactos unos con otros dentro de la aplicación, mandando mensajes en cualquier momento y estando conectados en tiempo real gracias a GameHub.
- **Edición de parámetros y datos del perfil registrado:** cada usuario registrado tendrá la opción en cualquier momento de editar sus datos de contacto, su ciudad o incluso su imagen de perfil que mostrar a los demás usuarios de la aplicación.
- **Cierre de sesión:** cada usuario podrá poner fin a su sesión iniciada y mostrarse inactivo a ojos de los datos de la aplicación. Volviendo a tener que iniciar sesión si es que necesita acceder de nuevo a GameHub.

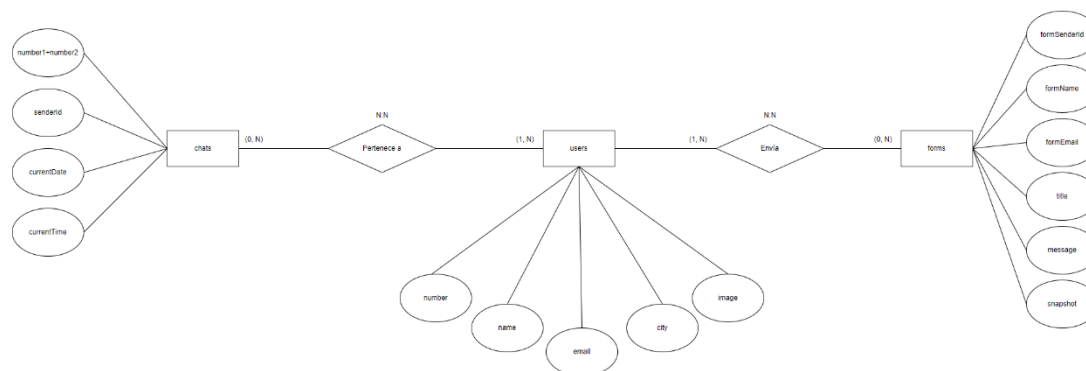
#### ❖ **Requisitos no funcionales:**

- **Visualización responsive:** la aplicación deberá responder de manera proporcional en términos de interfaz en diferentes dispositivos que difieran en tamaños de pantallas y resolución de estas.
- **Usabilidad:** el uso por parte de aquellas personas registradas en GameHub debe ser intuitivo y de fácil uso, proporcionando así una buena ergonomía y comunicación con el software de manera satisfactoria por parte del usuario.
- **Mantenibilidad del código de la aplicación:** la estructuración de archivos y carpetas, así como las clases y el código dentro de Android Studio ha de tener coherencia y una buena disposición a cambios y refactorizaciones de código que brinde al desarrollador facilidad al lanzar y desarrollar actualizaciones y parches destinado a la mejora de la aplicación.
- **Compatibilidad entre versiones Android:** la versión utilizada de la API de Android deberá ser segura a la vez que estable y proporcionar servicio a la mayoría de dispositivos del mercado aún con dispositivos con versiones de Android iguales o superiores a la versión Lollipop 5.0/5.1.1.
- **Integridad y cifrado de datos respetando políticas de privacidad:** todos los datos introducidos en GameHub por parte de los usuarios registrados estarán respaldados por una seguridad informática que imposibilite a posibles delincuentes informáticos de interceptar y/o robar dichos datos en los protocolos de intercomunicación entre la BBDD y la aplicación GameHub.

## 5.2 Diagramas Entidad – Relación, Casos de uso y Clases

Una vez ya hemos definido claramente y específicamente cada uno de los requisitos funcionales y no funcionales de nuestra aplicación, deberemos de elaborar una serie de diagramas para que el equipo de desarrollo pueda entender aún mejor las necesidades de la aplicación y de cómo desarrollar esta junto al modelo de almacenamiento en la base de datos, las relaciones entre estas y los casos de uso que nos llevarán todas estas funciones integradas dentro de GameHub.

### 5.2.1 Diagrama Entidad – Relación

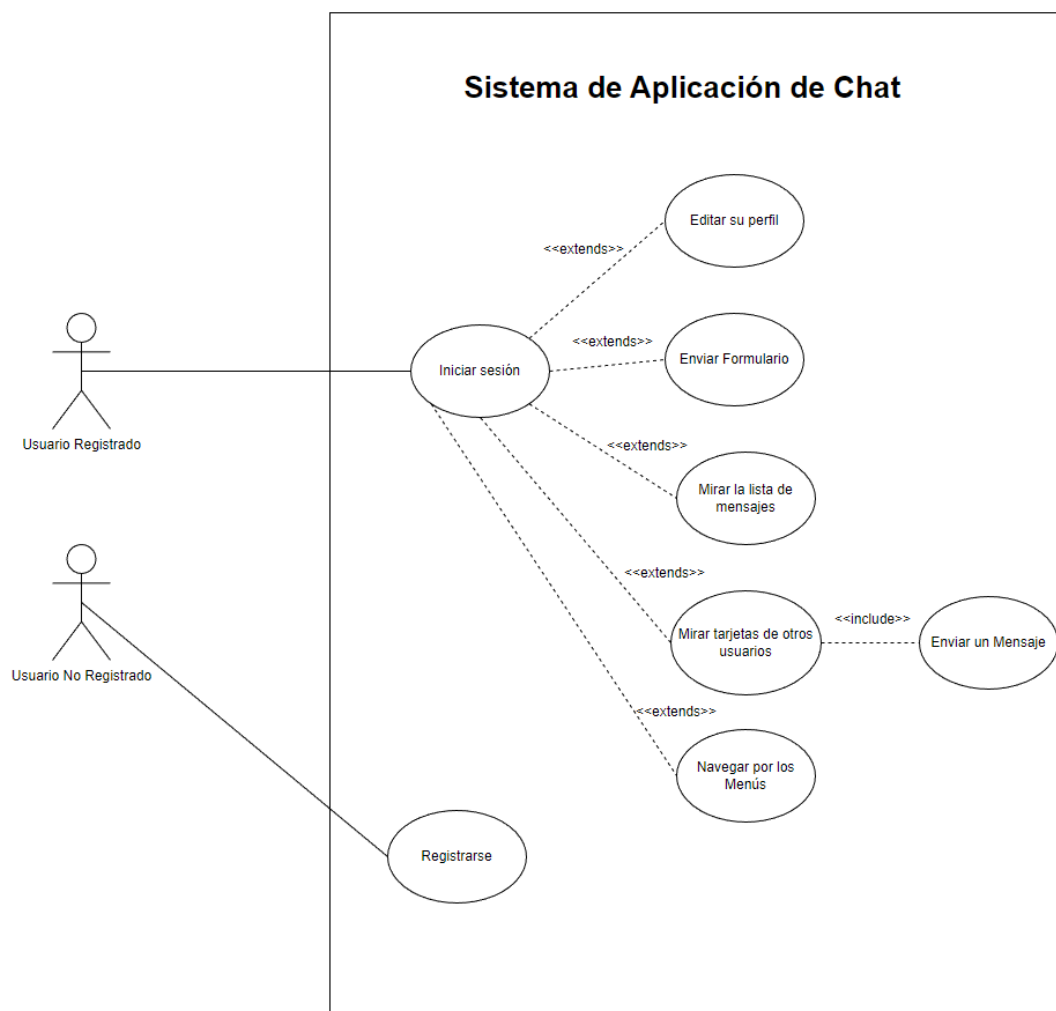


*Ilustración 5: Representación diagrama de Entidad - Relación. Fuente: elaboración propia en “<https://app.diagrams.net/>”*

Basándonos en el diagrama Entidad – Relación proporcionado, podemos ver el resultado de tres entidades diferentes relacionadas a la vez las unas con las otras.

La entidad “users” que alberga todos los atributos necesarios para poder identificarse, está relacionada a su vez con la entidad “chats” y con la entidad “forms”. Con chats establece una relación pública ya que una conversación entre el usuario activo y otro usuario es posible reiterarse con otro usuario distinto en la plataforma, por eso se denomina como relación de muchos a muchos. A su vez, como mencionamos anteriormente, la entidad users mantiene una relación del mismo tipo con la entidad forms, ya que de nuevo, un usuario pueda mandar tantos mensajes quiera a los desarrolladores informando de un error o poniéndose en contacto con ellos con los atributos que la entidad form contiene: formSenderId, formName, formEmail, etc...

## 5.2.2 Diagrama de Casos de uso



*Ilustración 6: Representación diagrama de Casos de uso. Fuente: elaboración propia en "https://app.diagrams.net"*

Para apoyar el Diagrama de Casos de uso facilitado nos ayudaremos de las siguientes tablas de casos de uso describiendo todos los escenarios posibles de uso de la aplicación GameHub:

<b>Identificador de caso de uso</b>	1
<b>Nombre del caso de uso</b>	Registrarse

<b>Actores Implicados</b>	Usuario No Registrado
<b>Precondición</b>	El usuario es la primera vez que inicia la app, por lo que deberá registrarse antes de usarla.
<b>Curso normal</b>	El usuario no registrado cumplimenta todos los campos del formulario de registro.
<b>Poscondiciones</b>	El usuario es registrado con éxito y carga la vista de menús de la aplicación
<b>Alternativas</b>	El usuario no rellena correctamente los campos de registro y la aplicación no permite realizar el registro. Se muestra un mensaje para que dicho usuario cumplimente correctamente el formulario de registro.

<b>Identificador de caso de uso</b>	2
<b>Nombre del caso de uso</b>	Iniciar sesión
<b>Actores Implicados</b>	Usuario Registrado
<b>Precondición</b>	El usuario ya se ha registrado correctamente en la aplicación y se dispone a iniciar sesión en el aplicativo.
<b>Curso normal</b>	El usuario introduce su número de teléfono y proporciona la clave OTP facilitada por SMS.
<b>Poscondiciones</b>	El usuario inicia sesión con éxito y carga la vista de menús de la aplicación
<b>Alternativas</b>	El usuario no rellena correctamente los campos de inicio de sesión y la aplicación no permite realizar el inicio de esta. Se muestra un mensaje para que dicho usuario cumplimente correctamente su número de teléfono y una clave OTP válida.

<b>Identificador de caso de uso</b>	3
<b>Nombre del caso de uso</b>	Editar su perfil
<b>Actores Implicados</b>	Usuario Registrado
<b>Precondición</b>	El usuario ha debido iniciar sesión y navegado hasta el menú de edición de perfil, este pulsa en el botón de "Editar Perfil".



<b>Curso normal</b>	El usuario rellena correctamente todos los campos, actualizando su información mostrada en el perfil y presiona en "Enviar"
<b>Poscondiciones</b>	El usuario es redirigido a la pantalla de Editar Perfil mostrando los nuevos datos actualizados tanto en la vista de la aplicación como en la BBDD.
<b>Alternativas</b>	El usuario no rellena correctamente los campos de edición de perfil y la aplicación no permite realizar la actualización. Se aborta la actualización en la BBDD y se muestra un mensaje para que dicho usuario cumplimente correctamente el formulario de edición de perfil.

<b>Identificador de caso de uso</b>	4
<b>Nombre del caso de uso</b>	Enviar Formulario
<b>Actores Implicados</b>	Usuario Registrado
<b>Precondición</b>	El usuario inicia sesión con éxito y carga la vista de menú desplegable de la aplicación, entra en la opción de "Informar sobre un error".
<b>Curso normal</b>	El usuario completa debidamente los campos necesarios para enviar el mensaje de contacto a los desarrolladores.
<b>Poscondiciones</b>	El mensaje es enviado y recibido en la base de datos en la nube y se le redirige automáticamente a la última vista de menú donde este se encontrase.
<b>Alternativas</b>	El usuario no rellena correctamente los campos de contacto y la aplicación no permite enviar la información. Se muestra un mensaje para que dicho usuario cumplimente correctamente el formulario de contacto.

<b>Identificador de caso de uso</b>	5
<b>Nombre del caso de uso</b>	Mirar la lista de mensajes
<b>Actores Implicados</b>	Usuario Registrado
<b>Precondición</b>	El usuario ha iniciado sesión en el aplicativo.
<b>Curso normal</b>	El usuario navega hasta la ventana de "Mensajes"

<b>Poscondiciones</b>	El usuario mira o no la lista de personas con las que mantiene una conversación si es que previamente había mandado un mensaje desde la pestaña de “Gaming”
<b>Alternativas</b>	No hay ninguna posible alternativa a este caso de uso

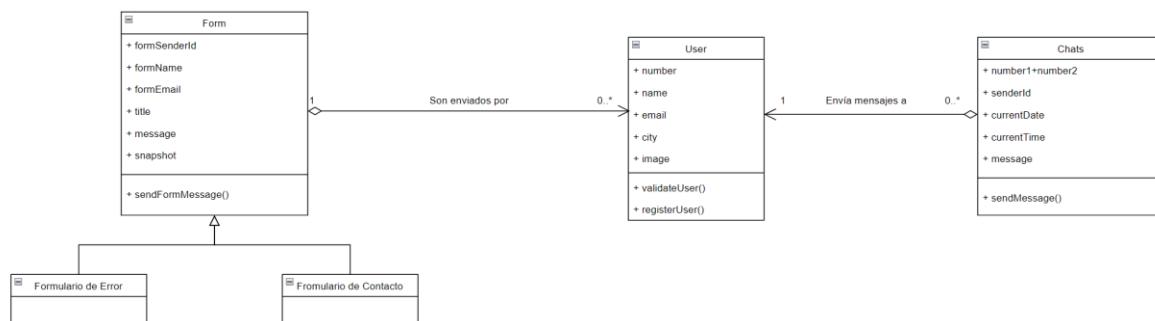
<b>Identificador de caso de uso</b>	6
<b>Nombre del caso de uso</b>	Mirar tarjetas de otros usuarios
<b>Actores Implicados</b>	Usuario Registrado
<b>Precondición</b>	Inicio de sesión del usuario activo.
<b>Curso normal</b>	El usuario navega hasta la ventana de “Gaming”
<b>Poscondiciones</b>	El usuario puede deslizar entre las múltiples tarjetas de perfil donde se muestran todos los usuarios registrados en la aplicación.
<b>Alternativas</b>	La conexión a la base de datos ha fallado y mostrará un mensaje diciendo: “Algo ha salido mal”

<b>Identificador de caso de uso</b>	7
<b>Nombre del caso de uso</b>	Enviar un Mensaje
<b>Actores Implicados</b>	Usuario Registrado
<b>Precondición</b>	El usuario navega hasta la ventana de “Gaming” y accede al botón de enviar un mensaje en la tarjeta de perfil de otro usuario.
<b>Curso normal</b>	El usuario activo envía un mensaje al usuario destinatario y es mostrado automáticamente en la vista en la que se encuentra.
<b>Poscondiciones</b>	El usuario destinatario del mensaje lo recibe en su pestaña del menú “Mensajes”
<b>Alternativas</b>	La conexión a la base de datos ha fallado y mostrará un mensaje diciendo: “Algo ha salido mal”

<b>Identificador de caso de uso</b>	8
<b>Nombre del caso de uso</b>	Navegar por los Menús

<b>Actores Implicados</b>	Usuario Registrado
<b>Precondición</b>	El usuario ha iniciado sesión
<b>Curso normal</b>	El usuario inicia sesión correctamente y tras la pantalla de carga, se muestran los menús de navegación
<b>Poscondiciones</b>	El usuario puede navegar indistintamente entre las tres pestañas del menú de navegación inferior.
<b>Alternativas</b>	No hay ninguna posible alternativa a este caso de uso.

### 5.2.3 Diagrama de Clases



*Ilustración 7: Representación diagrama de Clases. Fuente: elaboración propia en “<https://app.diagrams.net/>”*

Haremos en este último diagrama una explicación un poco más detallada de lo que representa.

El diagrama consta de tres superclases: Form, User y Chats, la utilidad de este diagrama reside en ver la disposición de los datos de una manera funcional de todos aquellos métodos, atributos y clases prediseñadas para la ejecución y estructuración del código de la aplicación.

1. **Superclase Form:** ésta es la encargada de almacenar todos los métodos y clases para poder redactar un formulario de contacto, a su vez crea dos subclases hijas que son “Formulario de Error” y “Formulario de Contacto” que son distintas variantes

dependiendo de para qué vayamos a utilizar esta funcionalidad. Además, la superclase Form está relacionada con la superclase User que nos permite hacer una relación de tipo “agregación” (composición débil) que hace que la funcionalidad de Formulario sea totalmente independiente a la superclase User, ya que no dependen estrictamente una de la otra para subsistir la aplicación.

2. **Superclase User:** esta superclase está relacionada simultáneamente con las superclases Form y Chats, también contiene todos los atributos y métodos necesarios para autenticar y registrar cuantos usuarios se requieran dentro de la aplicación. Comparte relación de tipo agregación con Form y Chats, de nuevo esto es debido a que son tres superclases independientes unas de otras, aun así, un usuario puede pertenecer a muchos chats y también puede enviar diversos mensajes a los desarrolladores informando de errores o poniéndose en contacto con ellos.
3. **Superclase Chats:** por último, la superclase Chats, alberga toda la información de los mensajes que son intercambiados entre dos usuarios distintos, así como la información relacionada con el momento del mensaje (currentDate y currentTime) y los usuarios implicados en dicha conversación (senderId y number1+number2). También posee el método que le permite a esta superclase mandar mensajes a otros usuarios.

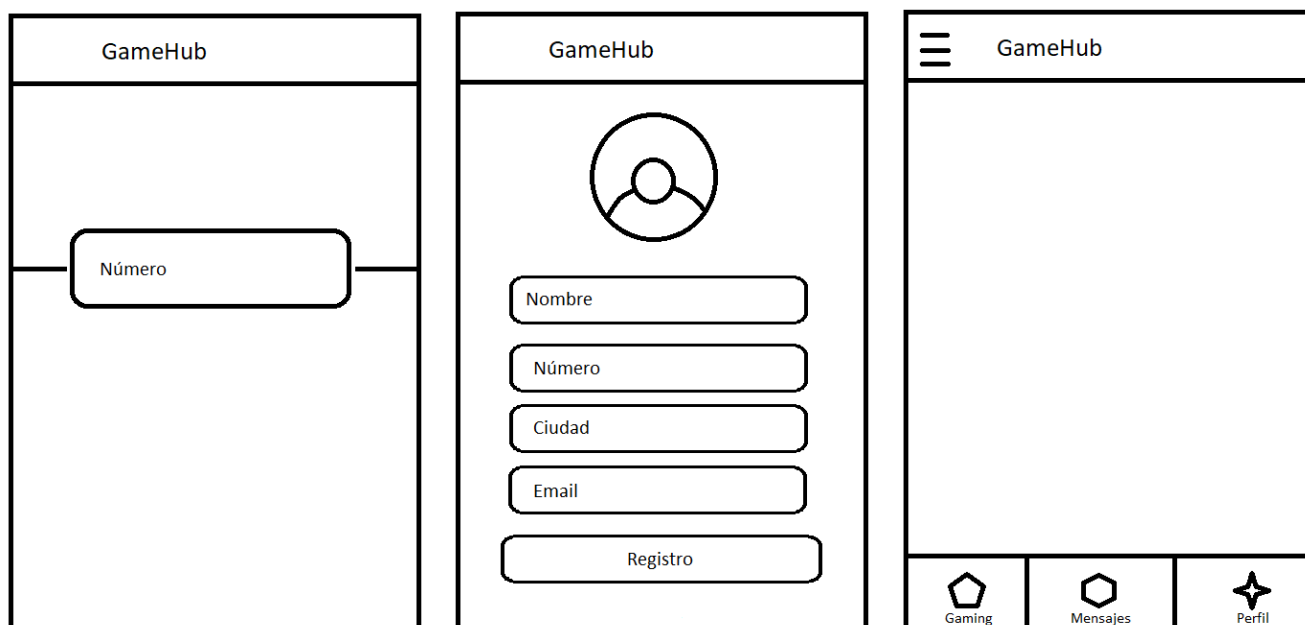
## 6. Diseño del proyecto

Como breve introducción a este punto de nuestra memoria nos enfocaremos en los diferentes aspectos de análisis del diseño y las vistas de la aplicación GameHub, después de esto tomaremos unas pautas para explicar funciones, arquitectura y funciones que hemos tenido que aplicar en el código de la aplicación.

### 6.1 Mockup's y Diseños de vistas

Antes de empezar a desarrollar el estilo de nuestra aplicación es necesario hacer algunos bocetos de las diferentes vistas que puede tener el usuario con la aplicación para así tener una referencia en la que basarnos a la hora de empezar a hacer el código “FrontEnd” de dichas interfaces.

Veamos así algunos de estos Mockup's que he estado realizando junto a las paletas de colores escogidas:



*Ilustración 8,9,10: Representación Mockups. Fuente: elaboración propia con la aplicación de “Paint”*

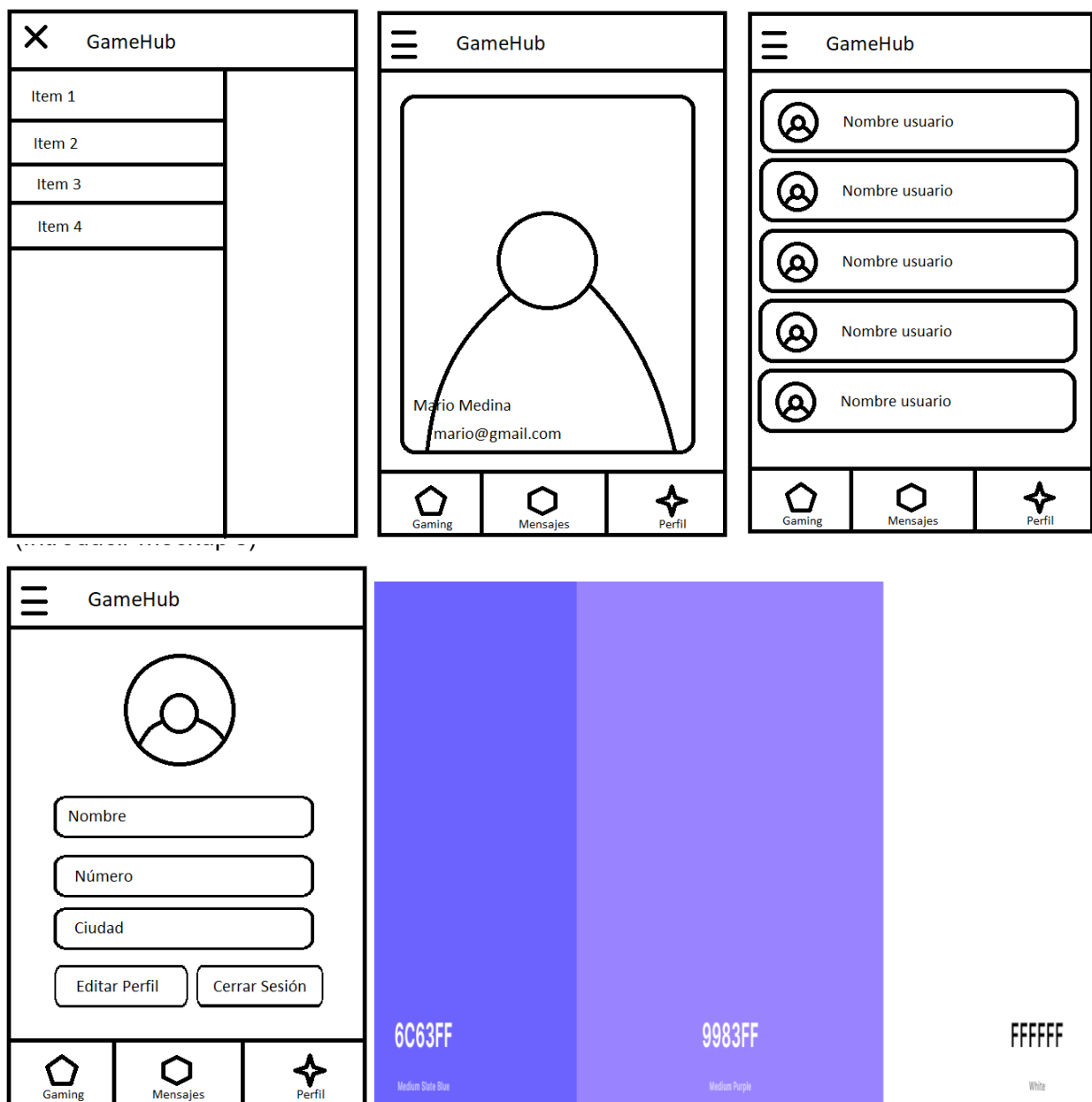


Ilustración 11,12,13 y 14: Representación Mockups. Fuente: elaboración propia con la aplicación de “Paint”

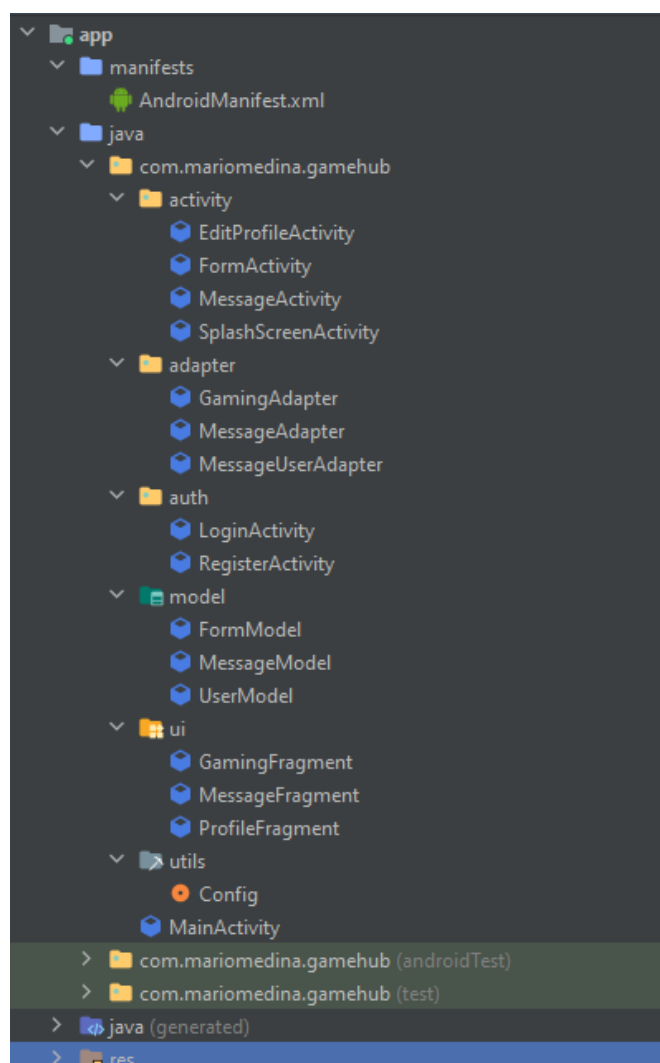
Ilustración 15: Paleta de colores de la Aplicación. Fuente: elaboración propia en: “<https://coolers.co/>”

## 6.2 Realización del proyecto, arquitectura y desarrollo de código

A continuación, haremos una explicación detallada de toda la fase de codificación dividiéndolas en diferentes apartados para una mejor comprensión y disposición de fases.

## 6.2.1 Estructura de carpetas

Primero, haremos un breve repaso de la organización de carpetas, funciones y archivos a nivel general, dicha organización ha sido necesaria para mantener una buena arquitectura, buenas prácticas y mantenibilidad de la aplicación.



*Ilustración 16: Estructura de carpetas. Fuente: elaboración propia.*

Como se aprecia en la captura de pantalla, la arquitectura de nuestro aplicativo se divide en los siguientes directorios y subdirectorios:

- **Manifests:** carpeta autogenerada contenedora del archivo AndroidManifest.xml también autogenerado el cuál, contiene todas las directrices de la aplicación Android (versión de la API, referencias al icono de la aplicación, habilitar backup's de la

aplicación, etc...) así como las llamadas y declaraciones de archivos “.activity” y la .activity por defecto al abrir la aplicación

- Java: carpeta donde se organizan los diferentes Fragments, Activity's y Modelos de la base de datos de nuestra aplicación, esta, contiene las siguientes subcarpetas:
  - Activity: aquí se encuentran todas las funcionalidades divididas por ficheros “.kt” (ficheros Kotlin) donde cada uno tiene el código necesario para que las funciones de la aplicación funcionen.
  - Adapter: este directorio almacena todas las funciones que son necesarias importar en el código de los ficheros Activity y permiten conectar y hacer llamadas a la base de datos de Google Firebase.
  - Auth: muy parecida a la función que tiene la carpeta Activity, pero con la diferenciación de que esta tiene un propósito muy específico: autenticación y registro del usuario. También alberga ficheros Activity.kt
  - Model: carpeta destinada a la estructura de bases de datos no relacionales como puede ser firebase. Se muestran los diferentes archivos que más tarde se convertirán en tablas “.json”.
  - UI: su nombre hace referencia a “User Interface” y se encarga de almacenar los archivos Fragment que componen la vista de usuario y su correcto funcionamiento entre ellas.
  - Utils: carpeta de utilidades donde encontraremos un archivo de configuración de variables privadas para ser importadas en según qué situación en el que el código nos lo exija
- Res: carpeta autogenerada la cual almacena en diferentes subdirectorios todos los archivos estéticos de la aplicación, diseños, vistas y código de lenguaje de marcas tales como el XML.
  - Drawable: en esta carpeta se encuentran todos los archivos .xml, .jpg, .png o svg que necesitemos llamar para el montaje de vistas del usuario tales como iconos o ideogramas.
  - Layout: archivos.xml asociados a su determinado Activity, es decir toda la vista que tiene el usuario en el momento en el que está realizando X funcionalidad.
  - Menu: realmente similar a la carpeta Layout, sin embargo esta está enfocada únicamente a las barras de navegación y menús desplegables, están asociados a los archivos .Fragment.
  - Navigation: archivo xml que ordena y establece orden relacionados con los menús Fragment de navegación.

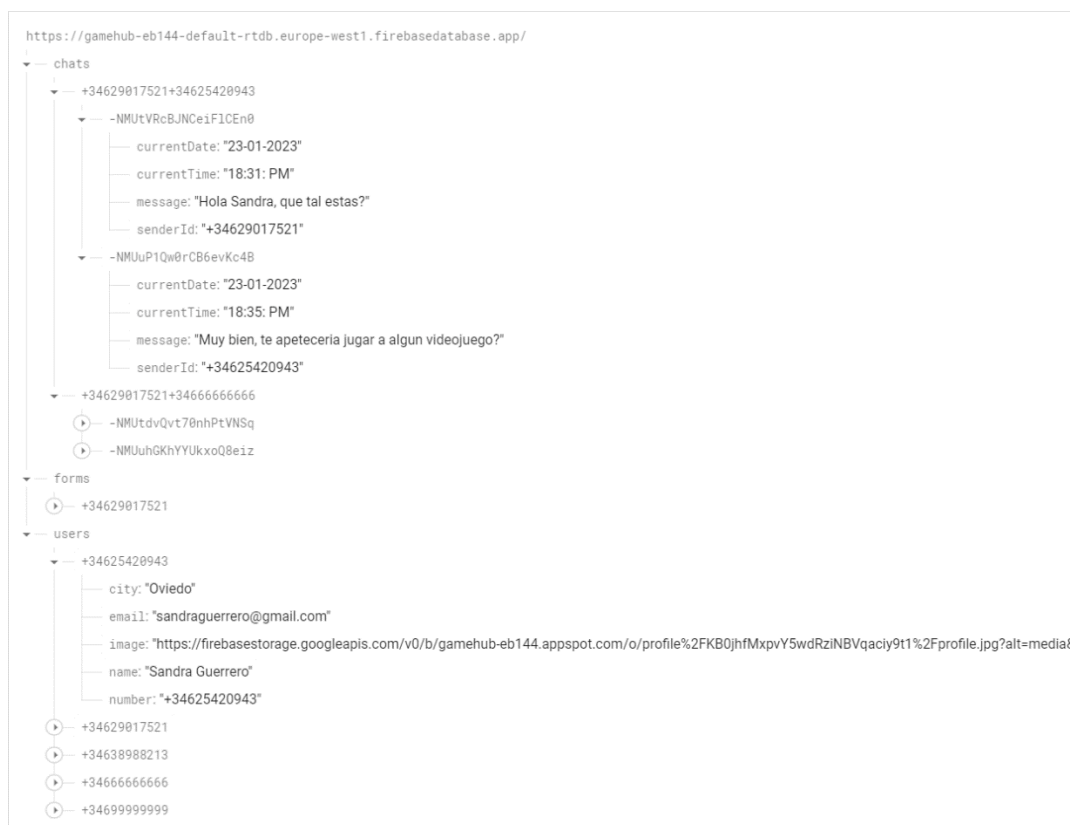


- **Values:** como su nombre indica valores que son almacenados para agilizar al desarrollador llamar a variables de estilo como son los colores en lugar de tener que codificarlo siempre en sistema hexagesimal.

### 6.2.2 SGBD, TablasNoSQL y Administración

Como ya hemos visto en el apartado anterior, nuestra aplicación goza de una buena estructuración de directorios la cual nos permite localizar fácilmente los archivos y códigos del programa, antes de pasar a ver cada carpeta más en detalle, es importante hacer mención al Sistema Gestor de Bases de Datos que hemos utilizado para las peticiones y almacenamiento de estos datos en la aplicación.

Como Sistema Gestor de Bases de Datos que hemos utilizado para GameHub nos encontramos a Firebase – Realtime Database. Una base de datos no relacional que nos permite hacer peticiones a la BBDD de una manera rápida a tiempo real de la ejecución del programa.



*Ilustración 17: Realtime Database – Firebase BBDDNoSQL. Fuente: elaboración propia.*

Como muestra la imagen proporcionada, GameHub actualmente cuenta con tres tablas en un sistema JSON de BBDDNoSQL aunque relacionadas estrechamente entre sí:

- Users:
  - (PK) “+34\_Number”:
    - ❖ City
    - ❖ Email
    - ❖ Image
    - ❖ Name
    - ❖ Number
- Chats:
  - (PK) “+34\_Number1+34\_Number2”:
    - ❖ CurrentDate
    - ❖ CurrentTime
    - ❖ Messsage
    - ❖ (FK) SenderId
- Forms:
  - (PK) “+34\_Number”:
    - ❖ FormEmail
    - ❖ FormName
    - ❖ (FK) FormSenderId
    - ❖ Message
    - ❖ Snapshot
    - ❖ Title

Cabe mencionar que el SGBDNoSQL usado: Firebase – Realtime Databse, utiliza un panel de control llamado Firebase console desde el que me ha sido de mucha ayuda poder navegar entre todas las soluciones que permite esta fantástica solución de google como sistema de bases de datos no relacional en la nube.

### 6.2.3 Activity’s, Fragments, Modelos y Layouts

Una vez hemos hecho un vistazo general a todos los directorios de nuestros archivos y hemos visto la disposición de la base de datos es el momento para explicar detalladamente cada uno de estas funciones, modelos y vistas al usuario de la aplicación.

### 6.2.3.1 Directorio Java y subdirectorios

En este subapartado veremos todas las carpetas más relevantes con su correspondiente código de programación relacionadas con el correcto funcionamiento de GameHub.

#### 6.2.3.1.1 Carpeta Activity

Como mencionamos antes esta carpeta alberga todos aquellos archivos activity relacionados con las funcionalidades del usuario que puede este ejecutar, vamos así cada uno de estos archivos:

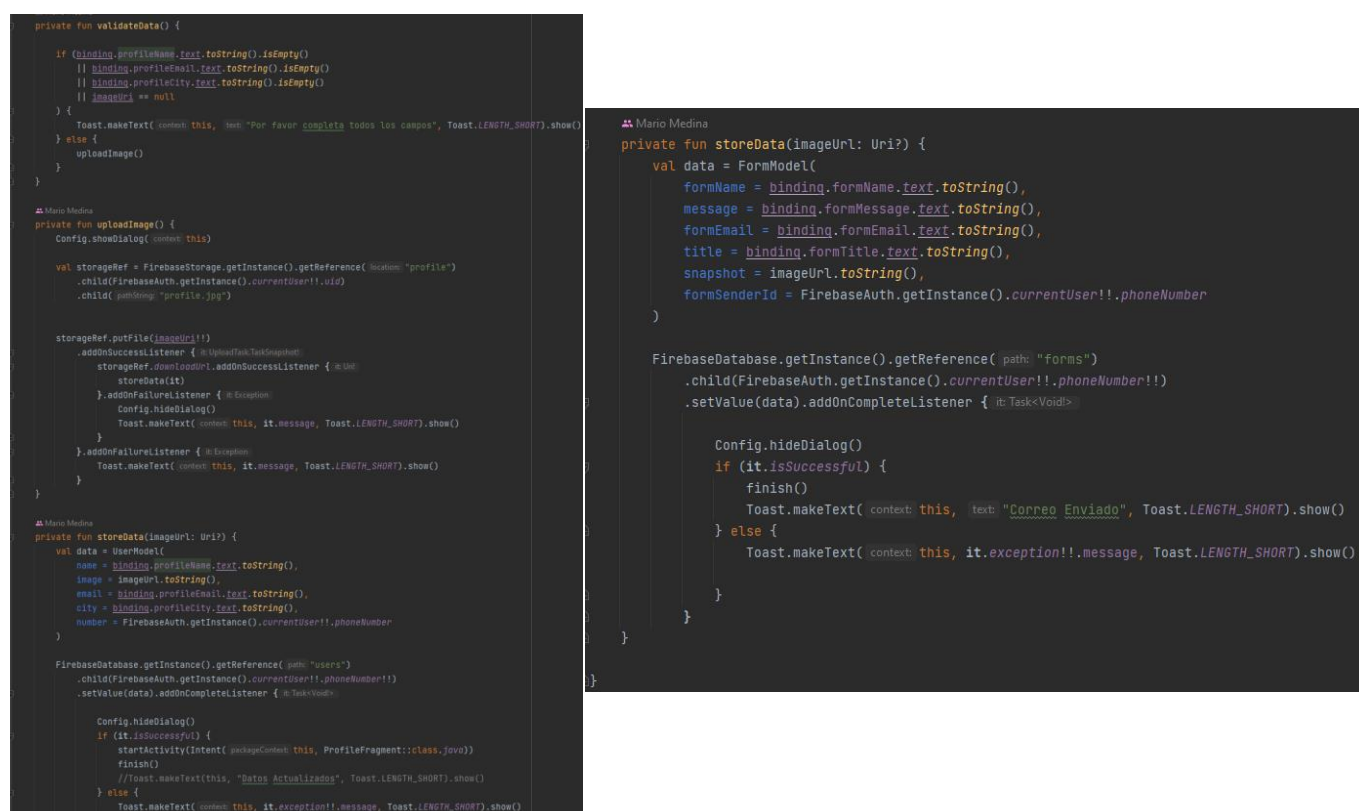


Ilustración 18 y 19: Código Activity. Fuente: elaboración propia.

- EditProfileActivity: fichero Kotlin que responde a las funciones de editar el perfil de usuario, se compone de las siguientes funciones:
  - onCreate: esta función hace uso de la variable Config para ayudarse a cargar la vista correspondiente al perfil de usuario, también se asegura, comprueba y transforma los datos introducidos a variables String relacionados con la base de datos.
  - validateData: se encarga de validar que los campos previos a la edición del perfil estén vacíos, si el usuario ingresa las directrices pasa a la función

storeData, en el caso contrario muestra un mensaje diciendo que por favor se completen todos los datos del formulario.

- uploadImage: una vez los datos son cumplimentados y se ha seleccionado una imagen de perfil, hace una llamada a la base de datos para subir dicha imagen.
- storeData: se encarga básicamente de almacenar todos los datos que se han rellenado previamente en los formularios en la base de datos no relacional de firebase.
- FormActivity: fichero Kotlin que responde a la funcionalidad de enviar mensajes de bugs o crasheos de la aplicación al desarrollador, hace uso de las mismas funciones que hemos visto en EditProfileActivity a diferencia de que los datos son almacenados en la tabla “forms” de la base de datos.
- MessageActivity: fichero Kotlin que se utiliza para la actualización y mantenimiento de la tabla “chats” de GameHub, hace uso de funciones storeData junto a las siguientes:
  - getData: hace una llamada a la base de datos para comprobar que efectivamente hay mensajes disponibles para mostrar de la tabla chats, también hace uso del “currentUser” y el usuario al que le hayan llegado o enviado mensajes.
  - verifyChatId: verifica ambas partes de la conversación, usuario activo (sender) y el usuario que recibe el mensaje (receiver).
  - onCreate: como en todas las funciones, ejecuta secuencialmente el código, si es correcto los mensajes son enviados a la base de datos y si es incorrecto le muestra un mensaje al usuario diciendo: “Algo salió mal”
- SplashScreenActivity: es el encargado de inicializar la animación de carga una vez las credenciales de inicio de sesión han sido verificadas.

### 6.2.3.1.2 Carpeta Adapter

Archivos que apoyan a las funciones Activity y los archivos del directorio UI para que ambos queden conectados.

```

Mario Medina
override fun onBindViewHolder(holder: GamingViewHolder, position: Int) {

    holder.binding.textView3.text = list[position].name
    holder.binding.textView2.text = list[position].email

    Glide.with(context).load(list[position].image).into(holder.binding.userImage)

    holder.binding.chat.setOnClickListener { it: View!
        val inte = Intent(context, MessageActivity::class.java)
        inte.putExtra(name: "userId", list[position].number)
        context.startActivity(inte)
    }

}

Mario Medina
override fun getItemCount(): Int {
    return list.size
}

```

Ilustración 20: Código Adapter. Fuente: elaboración propia.

- GamingAdapter: hace referencia al adaptador del menú “Gaming” y se compone de funciones onCreateViewHolder y onBindViewHolder las cuales permiten cargar y mostrar correspondientemente la información de los usuarios en la base de datos.
  - getItemCount: inicializa la función para contar el número de tarjetas cargadas en la vista de usuario en la pestaña “Gaming”
- MessageAdapter: vuelve a hacer uso de las funciones onCreateViewHolder, onBindViewHolder y getItemCount, pero en esta ocasión para poder cargar la lista de mensajes entre usuarios.
  - getItemViewType: función exclusiva de MessageAdapter donde la aplicación ordena los mensajes en función del sender o el receiver a la izquierda o a la derecha respectivamente.
- MessageUserAdapter: de nuevo usa las funciones ya anteriormente mencionadas para hacer un listado de usuarios con los cuales el usuario activo ha tenido una conversación y así poder mostrarse en dicho menú de la barra de navegación inferior.

### 6.2.3.1.3 Carpeta Auth

```

1  Mario Medina +1
2  private fun sendOTP(number: String) {
3      //binding.sendOTP.showLoadingButton()
4
5      dialog.show()
6
7      val callbacks = object : PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
8
9          override fun onVerificationCompleted(credential: PhoneAuthCredential) {
10
11              //binding.sendOTP.showNormalButton()
12
13              signInWithPhoneAuthCredential(credential)
14          }
15
16          override fun onVerificationFailed(e: FirebaseException) {
17
18          }
19
20          override fun onCodeSent(
21              verificationId: String,
22              token: PhoneAuthProvider.ForceResendingToken
23          ) {
24              this@LoginActivity.verificationId = verificationId
25
26              dialog.dismiss()
27              //binding.sendOTP.showNormalButton()
28
29              binding.numberLayout.visibility = GONE
30              binding.OTPLayout.visibility = VISIBLE
31          }
32      }
33
34      val options = PhoneAuthOptions.newBuilder(auth)
35          .setPhoneNumber("+34$number") // Phone number to verify
36          .setTimeout(60L, TimeUnit.SECONDS) // Timeout and unit
37          .setActivity(this) // Activity (for callback binding)
38          .setCallbacks(callbacks) // OnVerificationStateChangedCallbacks
39          .build()
40      PhoneAuthProvider.verifyPhoneNumber(options)
41  }

```

Ilustración 21 Código Auth. Fuente: elaboración propia.

Funciones Activity relacionadas con la autenticación y registro de usuarios.

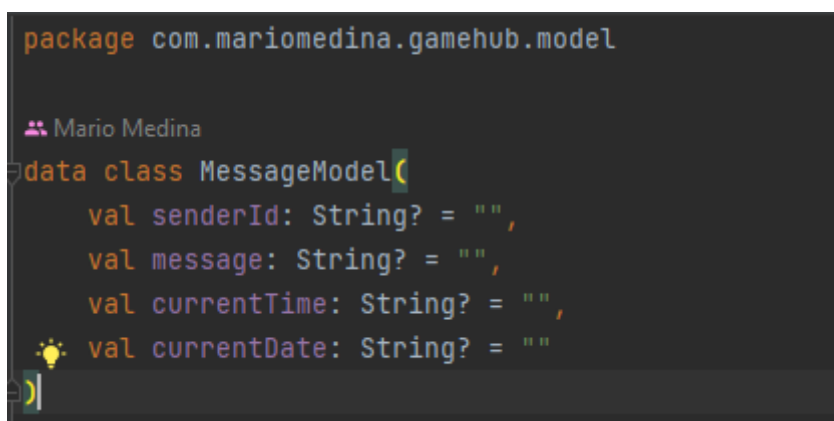
- LoginActivity: Archivo Kotlin que contiene las funciones necesarias para el inicio de sesión del usuario.
  - verifyOTP: comprueba que el mensaje de texto previamente enviado y recibido coincide con el código de seguridad para iniciar sesión.
  - sendOTP: envía un mensaje de texto al smartphone del usuario como código de seguridad.
  - signInWithPhoneAuthCredential: hace que el campo de ingresar el número de teléfono por parte del usuario cargue las vistas y envíe la información a la función sendOTP.
  - checkUserExist: comprueba que el número de teléfono ingresado en el respectivo campo sea de un usuario previamente registrado o no. En el caso de que no lo sea carga la vista de registro y si este ya estaba previamente registrado simplemente da acceso a la aplicación.

- RegisterActivity: Archivo Kotlin que contiene las funciones necesarias para el registro del usuario. Vuelve a hacer uso de las funciones validateData, uploadImage, storeData pero en este caso es utilizado para hacer el registro de usuario en la tabla “users” de la base de datos.

#### 6.2.3.1.4 Carpeta Model

Contiene los modelos y filas de la estructura de las tres tablas de la aplicación con sus correspondientes campos y datos para apoyarse en las funciones Activity y Adapter de la aplicación. Como ya vimos en el apartado 6.2.2 representan el tipo de datos que vamos a necesitar en nuestra BBDD en la nube

- FormModel: modelo de la tabla “forms”.
- MessageModel: modelo de la tabla “messages”.
- UserModel: modelo de la tabla “users”.



```
package com.mariomedina.gamehub.model

data class MessageModel(
    val senderId: String? = "",
    val message: String? = "",
    val currentTime: String? = "",
    val currentDate: String? = ""
)
```

Ilustración 22: Código Model. Fuente: elaboración propia.

### 6.2.3.1.5 Carpeta UI

Contiene toda la programación necesaria para poder mostrar y navegar por la barra inferior de navegación cargado el respectivo archivo “Fragment” que sea necesario en cada momento.

```

Mario Medina
class GamingFragment : Fragment() {

    private lateinit var binding: FragmentGamingBinding
    private lateinit var manager: CardStackLayoutManager

    Mario Medina
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        binding = FragmentGamingBinding.inflate(inflater)

        getData()
        return binding.root
    }
}

```

*Ilustración 23: Código UI. Fuente: elaboración propia.*

- GamingFragment: haciendo uso de la función onCreateView carga el layout correspondiente apoyándose de funciones anteriormente mencionadas como getData. También avisa al usuario de la última tarjeta de perfil mostrada tras descartarlas todas haciendo swipe.
- MessageFragment: vuelve a apoyarse en funciones onCreateView y getData para proporcionar la lista de mensajes de diferentes usuarios pasando por la barra de navegación.
- ProfileFragment: de nuevo se usa la función onCreateView para poder así navegar hacia el apartado de Perfil y llamar a la Activity correspondiente para mostrar todos los datos del usuario.

### 6.2.3.1.6 Utils

Carpeta contenedora de un único archivo de configuración: Config, donde se declara una variable privada llamada dialog para mostrar diálogos dentro de la aplicación e incluso mensajes de errores.



```

Mario Medina
object Config {

    private var dialog : AlertDialog? = null

    Mario Medina
    fun showDialog(context : Context) {
        dialog = MaterialAlertDialogBuilder(context)
            .setView(R.layout.loading_layout)
            .setCancelable(false)
            .create()

        dialog!!.show()
    }

    Mario Medina
    fun hideDialog() {
        dialog!!.dismiss()
    }
}

```

Ilustración 24: Código Utils. Fuente: elaboración propia.

#### 6.2.3.1.7 MainActivity

```

Mario Medina
override fun onNavigationItemSelected(item: MenuItem): Boolean {
    when(item.itemId){
        R.id.rateus -> {
            Toast.makeText( context: this, text: "Valóranos", Toast.LENGTH_SHORT).show()
        }
        R.id.developer -> {
            startActivity(Intent( packageContext: this, FormActivity::class.java))
        }
        R.id.favourite -> {
            Toast.makeText( context: this, text: "Favorito", Toast.LENGTH_SHORT).show()
        } R.id.share -> {
            Toast.makeText( context: this, text: "Compartir", Toast.LENGTH_SHORT).show()
        }
        R.id.termsCondition -> {
            Toast.makeText( context: this, text: "Términos y Condiciones", Toast.LENGTH_SHORT).show()
        }
        R.id.privacyPolicy -> {
            Toast.makeText( context: this, text: "Privacidad", Toast.LENGTH_SHORT).show()
        }
    }
    return true
}

```

Ilustración 25: Código MainActivity. Fuente: elaboración propia.

Activity principal al lanzar la aplicación, es el fichero Kotlin más importante y se encarga entre otras cosas de cargar todos los menús laterales, los menús de navegación y todas las demás Activity's de la aplicación. Usa funciones como:

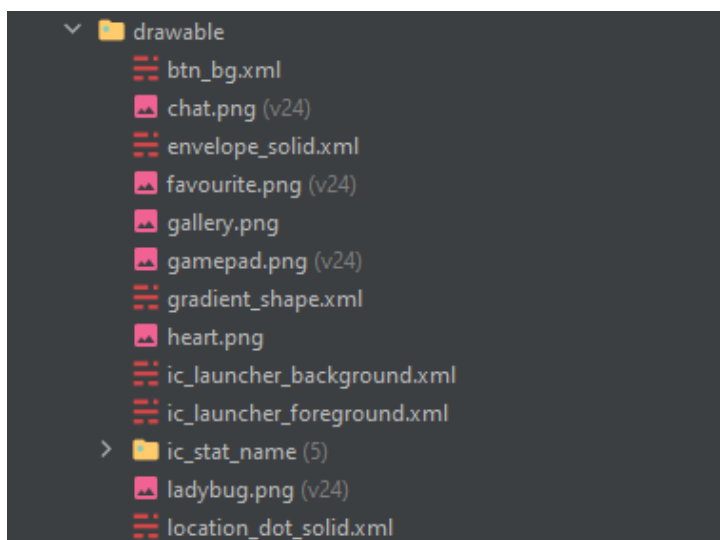
- `onNavigationItemSelectedListener`: se encarga del funcionamiento de todos los apartados del menú lateral.
- `onOptionsItemSelected`: se encarga de desplegar y recoger el menú lateral de la aplicación.

### 6.2.3.2 Directorio Res y subdirectorios

En este subapartado veremos todas las carpetas más relevantes con su correspondiente código de lenguaje de marcado relacionadas con la correcta visualización de GameHub.

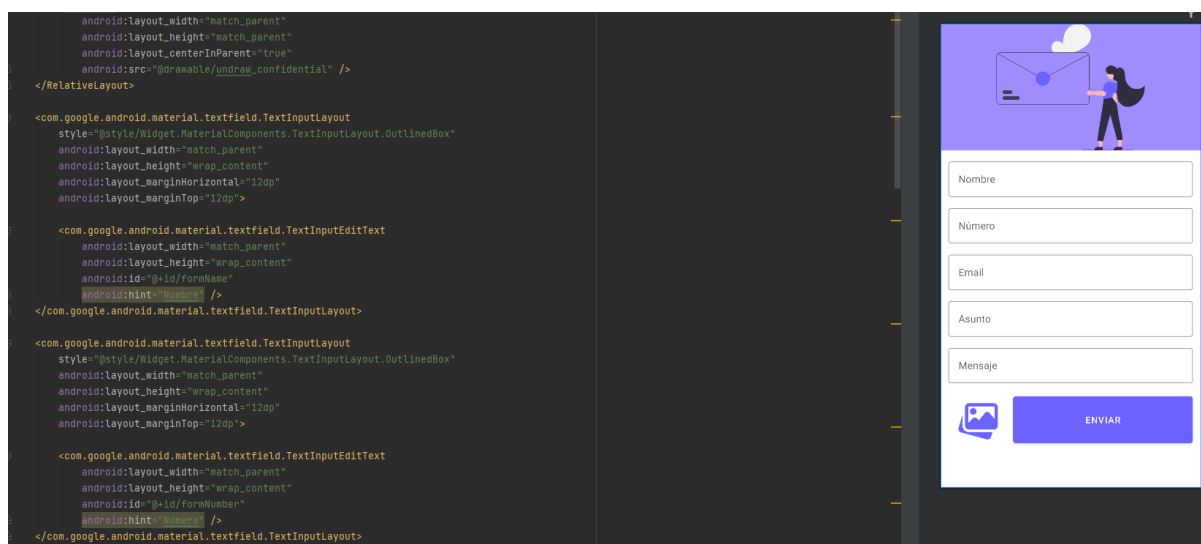
#### 6.2.3.2.1 Carpeta Drawable

Carpeta que almacena todos los iconos, imágenes, vectores y archivos visuales .xml, .jpg, .png o .svg de la aplicación. Estos archivos son necesarios para aplicarle estilo y dinamismo a las vistas de la aplicación, son indispensables para proporcionar una interfaz amigable e intuitiva al usuario.



*Ilustración 26: Carpeta Drawable. Fuente: elaboración propia.*

### 6.2.3.2.2 Carpeta Layout



*Ilustración 27: Código Drawable. Fuente: elaboración propia.*

Aquí encontraremos todas las vistas asociadas a sus Activitis correspondientes, se trata del código XML que se encarga de proporcionar las vistas de la aplicación, así como cargar los archivos de la carpeta Drawable. Sin los Layouts no tendríamos interfaz para comunicarnos de manera persona-máquina.

### 6.2.3.2.3 Carpeta Menu

Dedicada a generar menús contextuales, tales como el menú lateral y el menú inferior de navegación. No proporciona estilos ya que estos se les otorga en los archivos de la carpeta Layout, sin embargo, son necesarios como esqueleto de estas interfaces.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android">
3
4      <item android:id="@id/gamingFragment"
5          android:title="Gaming"
6          android:icon="@drawable/gamepad"/>
7
8      <item android:id="@id/messageFragment"
9          android:title="Mensajes"
10         android:icon="@drawable/chat"/>
11
12     <item android:id="@id/profileFragment"
13         android:title="Perfil"
14         android:icon="@drawable/profile"/>
15
16 </menu>

```

Ilustración 28: Código Menu. Fuente: elaboración propia.

#### 6.2.3.2.4 Carpeta Navigation

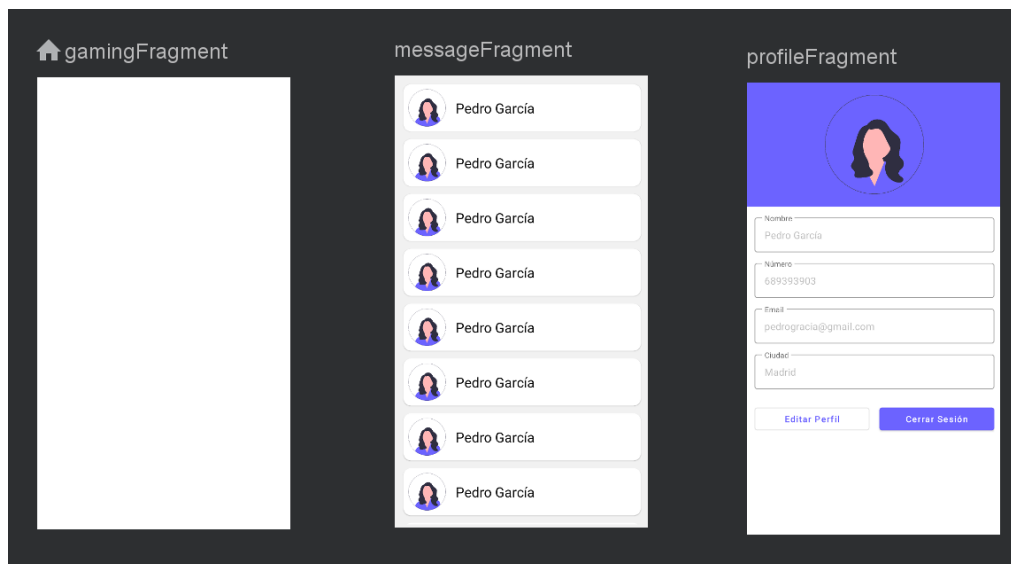



Ilustración 29: Carpeta Navigation. Fuente: elaboración propia.

Contiene un único archivo .xml dedicado a establecer el orden de navegación en los archivos Fragment, aquí las vistas son ordenadas en forme al menú de navegación inferior de la aplicación.

### 6.2.3.2.5 Carpeta Values

Como su nombre indica hace referencia a todos los valores que hemos tenido que usar a la hora de codificar las vistas, en el archivo colors.xml se almacenan variables de colores en formato hexadecimal seguidas de una declaración en forma de nombre más intuitivo para poder codificar en otros archivos de Layout de manera mucho más cómoda. También encontramos el archivo strings.xml donde encontraremos los textos de los menús de navegación. Actualmente la aplicación sólo está en español, pero con este archivo podemos declarar otros “strings” para que al cambiar el smartphone de idioma la aplicación cambie de idioma también simultáneamente.



```

1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="purple_200">#FFBB86FC</color>
4      <color name="purple_500">#FF6200EE</color>
5      <color name="purple_700">#FF3700B3</color>
6      <color name="teal_200">#FF03DAC5</color>
7      <color name="teal_700">#FF018786</color>
8      <color name="black">#FF000000</color>
9      <color name="white">#FFFFFFFF</color>
10     <color name="light_grey">#F1F1F1</color>
11     <color name="colorPrimary">#6C63FF</color>
12     <color name="colorSecondary">#9983FF</color>
13     <color name="colorForm">#A18CFD</color>
14     <color name="bg_light_grey">#ECECEC</color>
15 </resources>

```

Ilustración 30: Código Values. Fuente: elaboración propia.

## 7. Despliegue y pruebas

Una vez hemos desplegado nuestra aplicación y hemos terminado las fases de Diseño y Análisis vistos en el punto 5 y 6 de esta memoria de proyecto de desarrollo de aplicaciones multiplataforma procederemos a probar todas las funcionalidades y vistas de la aplicación para asegurar así la estabilidad e integridad del aplicativo, proporciono así una tabla de pruebas con estructura de lista de todos los elementos probados:

Índice numeral	ESPECIFICACIONES DE LA PRUEBA
1	<ul style="list-style-type: none"> <li>Objetivo probado: registro de nuevo usuario</li> <li>Requisitos de la prueba: registro de usuario satisfactorio almacenando credenciales y datos de este en firebase.</li> <li>Pruebas realizadas: crear un nuevo usuario a través de la aplicación</li> </ul>
2	<ul style="list-style-type: none"> <li>Objetivo probado: registro de nuevo usuario con credenciales erróneas</li> <li>Requisitos de la prueba: registro de usuario con credenciales erróneas evitando que pase a la próxima vista y anulando el almacenamiento en firebase.</li> <li>Pruebas realizadas: crear un nuevo usuario a través de la aplicación con datos erróneos</li> </ul>
3	<ul style="list-style-type: none"> <li>Objetivo probado: inicio de sesión de usuario existente</li> <li>Requisitos de la prueba: poder iniciar sesión de nuevo con un usuario previamente almacenados en la base de datos.</li> <li>Pruebas realizadas: ingresar los datos de verificación correctamente en el formulario de validación OTP</li> </ul>
4	<ul style="list-style-type: none"> <li>Objetivo probado: inicio de sesión de usuario existente con credenciales erróneas.</li> <li>Requisitos de la prueba: evitar iniciar sesión de nuevo con un usuario previamente almacenados en la base de datos incluyendo credenciales erróneas.</li> <li>Pruebas realizadas: ingresar los datos de verificación incorrectamente en el formulario de validación OTP</li> </ul>
5	<ul style="list-style-type: none"> <li>Objetivo probado: acceso al menú de navegación y a sus vistas</li> <li>Requisitos de la prueba: navegar por el menú inferior entre todas sus vistas correctamente.</li> <li>Pruebas realizadas: usar el menú de navegación repetidas veces haciendo saltos erráticos no secuenciales para confirmar el funcionamiento.</li> </ul>
6	<ul style="list-style-type: none"> <li>Objetivo probado: despliegue del menú lateral de funciones secundarias</li> <li>Requisitos de la prueba: visualización de la transición y efectos hasta mostrar correctamente el menú lateral.</li> <li>Pruebas realizadas: pulsar repetidas veces en el botón de despliegue y en cada una de sus funcionalidades.</li> </ul>
7	<ul style="list-style-type: none"> <li>Objetivo probado: Envío de mensajes de informe de errores</li> </ul>

	<ul style="list-style-type: none"> <li>• Requisitos de la prueba: envío directo y almacenamiento en firebase de los campos cumplimentados y memoria en los campos específicos de la aplicación para futuros mensajes de errores</li> <li>• Pruebas realizadas: rellenar completamente un formulario de errores con la información necesaria.</li> </ul>
8	<ul style="list-style-type: none"> <li>• Objetivo probado: Vista íntegra de todos los usuarios en forma de tarjetas en la pestaña de Gaming</li> <li>• Requisitos de la prueba: el funcionamiento del sistema de tarjetas donde se muestra la información de los usuarios es satisfactoria.</li> <li>• Pruebas realizadas: deslizar todas las tarjetas de los usuarios registrados hasta llegar a la última de estas.</li> </ul>
9	<ul style="list-style-type: none"> <li>• Objetivo probado: Uso de la pestaña de perfil, muestra la información contenida en la base de datos</li> <li>• Requisitos de la prueba: mostrar todos los datos del usuario que ha iniciado sesión con llamadas a firebase.</li> <li>• Pruebas realizadas: entrar reiteradamente a la pestaña de Perfil.</li> </ul>
10	<ul style="list-style-type: none"> <li>• Objetivo probado: Uso de la pestaña de perfil, actualiza la información del usuario.</li> <li>• Requisitos de la prueba: uso del botón "Editar Perfil" rellenar el formulario con los campos habilitados para ello y actualizar la información dentro de la base de datos de firebase.</li> <li>• Pruebas realizadas: editar múltiples perfiles de usuarios previamente registrados.</li> </ul>
11	<ul style="list-style-type: none"> <li>• Objetivo probado: Uso de la pestaña de perfil, se visualiza la información actualizada del perfil editado.</li> <li>• Requisitos de la prueba: después de haber hecho las respectivas consultas de la base de datos, la pestaña Perfil muestra de nuevo la información previamente actualizada.</li> <li>• Pruebas realizadas: editar múltiples perfiles de usuarios previamente registrados y volver a la visualización del perfil.</li> </ul>
12	<ul style="list-style-type: none"> <li>• Objetivo probado: En la pestaña de Gaming, botón y vista de enviar un mensaje es satisfactorio.</li> <li>• Requisitos de la prueba: enviar correctamente mensajes a un usuario y que este se almacene en la base de datos con la información correspondiente.</li> <li>• Pruebas realizadas: enviar un primer mensaje a un usuario.</li> </ul>
13	<ul style="list-style-type: none"> <li>• Objetivo probado: En la pestaña de Mensajes se muestran la lista de todos los usuarios a los que hemos enviado un mensaje y los que hemos recibido por otros usuarios.</li> <li>• Requisitos de la prueba: la pestaña de Mensajes visualiza todas las conversaciones del usuario en cuestión</li> <li>• Pruebas realizadas: entrar reiteradamente a la pestaña de Mensajes.</li> </ul>
14	<ul style="list-style-type: none"> <li>• Objetivo probado: Se muestra la información contenida por la base de datos de todos los mensajes dentro de la vista de chat entre dos usuarios.</li> </ul>

	<ul style="list-style-type: none"> <li>Requisitos de la prueba: se debe de poder tener en todo momento acceso a las conversaciones guardadas en firebase de mensajes previos entre dos usuarios</li> <li>Pruebas realizadas: entrar reiteradamente en diferentes conversaciones entre los usuarios.</li> </ul>
15	<ul style="list-style-type: none"> <li>Objetivo probado: Se actualiza la información contenida en la base de datos de nuevos mensajes enviados a otros usuarios en la vista del chat.</li> <li>Requisitos de la prueba: se visualiza correctamente la información de los nuevos mensajes en tiempo real que se están enviando y mostrándose en la vista correspondiente.</li> <li>Pruebas realizadas: enviar nuevos mensajes a diferentes usuarios dentro de la vista de Mensajes.</li> </ul>
16	<ul style="list-style-type: none"> <li>Objetivo probado: Cierre de sesión correcto del usuario activo</li> <li>Requisitos de la prueba: terminar la sesión de un usuario activo y la aplicación muestre de nuevo la vista de inicio de sesión.</li> <li>Pruebas realizadas: cerrar sesión con varios usuarios.</li> </ul>
17	<ul style="list-style-type: none"> <li>Objetivo probado: Nuevo inicio de sesión con otro usuario diferente después de haber cerrado otra sesión.</li> <li>Requisitos de la prueba: después de haber cerrado sesión con un usuario, iniciar de nuevo de manera exitosa otro diferente cargando todos sus datos almacenados en firebase.</li> <li>Pruebas realizadas: cierre de sesión con varios usuarios e inicio de sesión con otros diferentes.</li> </ul>
18	<ul style="list-style-type: none"> <li>Objetivo probado: Guardar los datos de la sesión iniciada aún habiendo finalizando la ejecución de la App en segundo plano.</li> <li>Requisitos de la prueba: tras cerrar la aplicación en la multitarea e iniciar de nuevo, los datos de inicio de sesión están guardados y la sesión sigue activa.</li> <li>Pruebas realizadas: cierre repetido de la App e inicialización de nuevo de ella.</li> </ul>
19	<ul style="list-style-type: none"> <li>Objetivo probado: Recibir correctamente el código de seguridad OTP vía mensaje de texto para el inicio de sesión.</li> <li>Requisitos de la prueba: hacer un inicio o registro de usuario para que el mensaje de texto sea válido al introducirlo dentro del aplicativo.</li> <li>Pruebas realizadas: iniciar sesión repetidas veces y comprobar si el mensaje de texto recibido es correcto.</li> </ul>
20	<ul style="list-style-type: none"> <li>Objetivo probado: Eliminar la memoria de la aplicación para forzar el cierre de sesión actual y datos en caché.</li> <li>Requisitos de la prueba: acceder a la información de la app dentro del sistema operativo Android y eliminar todos los datos de la aplicación y que después de esto la aplicación efectivamente no recuerde la última sesión activa.</li> <li>Pruebas realizadas: eliminar todos los datos de la aplicación tanto de almacenamiento como de caché y forzar la detención de la App.</li> </ul>



## 8. Conclusiones

Llegando al fin de esta documentación de la síntesis de todos los aspectos más importantes en el desarrollo de mi aplicación haremos un breve repaso por los objetivos más específicos que se han alcanzado en este proyecto de fin de ciclo en Desarrollo de Aplicaciones Multiplataforma.

### Objetivos cumplidos:

- Funcionalidad de registro de diferentes usuarios dentro de la aplicación.
- Inicio de sesión controlado y recordado por la caché y datos de la aplicación.
- Acceso denegado al usuario en el inicio de sesión si se introduce una clave OTP no correspondiente a la facilitada por mensaje de texto móvil (SMS).
- Todos los usuarios registrados en el aplicativo pueden navegar entre las pestañas de “Perfil”, “Gaming” y “Mensajes”.
- Todos los usuarios registrados en el aplicativo pueden desplegar el menú contextual y acceder a los diferentes apartados.
- Todos los usuarios pueden rellenar correctamente el formulario de Informe de errores, los datos son almacenados íntegramente en la base de datos y los datos relevantes de contacto son recordados en peticiones de firebase a través de la aplicación.
- Todos los usuarios pueden navegar entre las tarjetas de otros usuarios registrados descartando los necesarios y llegando al fin de la última tarjeta.
- Todos los usuarios pueden comenzar una conversación desde la vista de tarjetas y continuarlas dentro de la pestaña de mensajes de la aplicación. Guardándose todos los mensajes con la fecha y la hora dentro de la base de datos en tiempo real de firebase.
- Todos los usuarios pueden editar correctamente su foto de perfil, su ciudad de residencia y su correo electrónico de contacto.
- El usuario identificado puede cerrar su sesión de forma segura protegiendo así sus datos.

### Reflexión y conclusiones de fin del PFC:

Como conclusiones finales me gustaría hablar sobre el desarrollo de aplicaciones mobile en la industria tecnológica y las consultoras y es que con la elaboración del código por parte de

una única persona como en mi caso para este proyecto de fin de ciclo, realizar una aplicación robusta y con altas funcionalidades así como consultas y validaciones a bases de datos junto a conocimientos de protocolos seguros como puede ser el TCP/IP es un desafío de un nivel considerable teniendo en cuenta fechas y entregas a supuestos clientes, sin embargo, también pienso que la oportunidad que brinda este ciclo formativo de grado superior son abundantes, ya sea como desarrollador independiente teniendo buenas ideas para el desarrollo de aplicaciones Android y monetizarlas de una manera sencilla gracias a la PlayStore de Google o incluso en el sector empresarial ganando experiencia y pudiendo obtener cargos de responsabilidad como “tech lead” gracias a conocimientos profundos en diversos lenguajes de programación y algoritmos eficientes.

También me gustaría darle la importancia que se merece al trabajo realizado tanto de código como de documentación y presentación ya que tareas así suelen intimidar a muchos recién titulados y perfiles junior de programación, además, teniendo presentes las fechas estipuladas ha habido momentos en los que realmente pensaba que no llegaría a las fechas de entrega y eso ha generado muchos días de ansiedad al ver que el código no pasaba las pruebas y tenía que buscar una manera de solucionarlo lo antes posible para la disposición del tribunal y la defensa de este trabajo, sin embargo, la autorrealización y la satisfacción que nacía en mí cada vez que veía un pequeño avance en este desafiante reto hacía que mereciese la pena todos esos momentos de crisis que finalmente se acabaron traduciendo en aprendizaje y habilidad a la hora de enfrentarme a problemas en un entorno laboral simulado.

Este trabajo también ha conseguido que amplié aún más mis conocimientos de arquitectura de software y programación ya que es el segundo ciclo de grado superior que supero satisfactoriamente después de DAW y eso a su vez consigue añadir más contenido a mi portfolio como desarrollador para generar posibles entrevistas y mejores puestos en compañías de TI a los que aplicar y desenvolverme con mayor soltura, y es que aunque me quede aún por realizar un largo recorrido y conocer a la par que estudiar nuevas tecnologías y metodologías de trabajo voy empezando a sentirme más seguro de mis habilidades olvidando así un poco el síndrome del impostor que tenemos todos a la hora de aventurarnos los primeros años en la carrera profesional que hayamos escogido.

Como una breve síntesis de todo, considero que la elaboración, codificación y defensa de este trabajo ha reforzado mis capacidades como desarrollador de software *mobile* profesional sin despreciar las infravaloradas *softskills* que todo buen profesional debería de conocer.

## 9. Vías futuras

Aunque se han cumplido mayoritariamente la gran mayoría de objetivos propuestos para la aplicación, podemos enriquecer y adoptar una serie de medidas y funcionalidades nuevas que brindaría a nuestra aplicación una mejor disposición a ser descargada y mantenida en los dispositivos de los usuarios por el mayor tiempo posible, actualizándose así y creando buenas referencias tanto a la compañía desarrolladora (mi perfil de empresa dentro de la PlayStore) como para mí como desarrollador de aplicaciones móviles.

Además, todas estas características que serán implementadas ayudarán a tener una experiencia mucho más satisfactoria por parte del usuario.

Habiendo comentado esto, he realizado un listado de funcionalidades a aplicar a corto y largo plazo para que GameHub no pierda valor dentro del mercado de apps:

1. **Lanzamiento de GameHub a la PlayStore de Google**, teniendo así que abonar las tasas de desarrollo de la PlayStore y creando mi perfil como desarrollador en esta.
2. **Hacer un análisis ASO** promoviendo visitas y descargas mayoritarias en la tienda de aplicaciones gracias a un buen márketing y posicionamiento.
3. **Agregar una función de videollamada** entre usuarios para poder tener un contacto más cercano y poder conocerse mejor sin tener que utilizar apps de terceros.
4. **Añadir un sistema de rating** o de reseñas para que el usuario pueda evaluar y compartir cómo ha sido su experiencia con su compañero de juego en GameHub y así otros usuarios puedan tener una referencia de este.
5. **Filtrar usuarios** según los intereses del consumidor, poder escoger ver solo jugadores que jueguen a un título específico o que la app solo muestre usuarios con reseñas escritas por parte de otros jugadores.
6. **Reescribir todo el código en Dart**, lenguaje de programación oficial de Flutter y poder así generar simultáneamente la aplicación para usuarios de iOS y que puedan disfrutar de las funciones de GameHub.
7. **Aplicar un algoritmo de seguimiento**, es decir que aquellas personas con las que te ha gustado interactuar puedas seguirla a modo de red social y ver sus últimas publicaciones.
8. **Implementar un sistema de notificaciones** push a través de API REST para que los usuarios puedan saber sin estar dentro de la aplicación si han recibido un mensaje o alguien le ha escrito una reseña.

9. **Integrar conexiones entre plataformas de videojuegos** poder mostrar en los perfiles de manera oficial de otras plataformas de videojuegos como pueden ser Xbox, PlayStation, Nintendo, Battle.net y que permita la autenticación también por estas vías.

## 10. Bibliografía/Webgrafía

- colaboradores de Wikipedia. (2022a, julio 26). *Firestore*. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Firebase>
- colaboradores de Wikipedia. (2022b, agosto 2). *Kotlin (lenguaje de programación)*. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Kotlin\\_\(lenguaje\\_de\\_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Kotlin_(lenguaje_de_programaci%C3%B3n))
- colaboradores de Wikipedia. (2022c, octubre 4). *Extensible Markup Language*. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Extensible\\_Markup\\_Language](https://es.wikipedia.org/wiki/Extensible_Markup_Language)
- colaboradores de Wikipedia. (2022d, diciembre 17). *Google Workspace*. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Google\\_Workspace](https://es.wikipedia.org/wiki/Google_Workspace)
- colaboradores de Wikipedia. (2022e, diciembre 20). *Android Studio*. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Android\\_Studio](https://es.wikipedia.org/wiki/Android_Studio)
- colaboradores de Wikipedia. (2023a, enero 18). *Windows 10*. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Windows\\_10](https://es.wikipedia.org/wiki/Windows_10)
- colaboradores de Wikipedia. (2023b, enero 18). *Windows 11*. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Windows\\_11](https://es.wikipedia.org/wiki/Windows_11)
- colaboradores de Wikipedia. (2023c, enero 19). *JSON*. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/JSON>
- colaboradores de Wikipedia. (2023d, enero 23). *Google Chrome*. Wikipedia, la enciclopedia libre. [https://es.wikipedia.org/wiki/Google\\_Chrome](https://es.wikipedia.org/wiki/Google_Chrome)
- *GitHub - bumptech/glide: An image loading and caching library for Android focused on smooth scrolling*. (s. f.). GitHub. <https://github.com/bumptech/glide>
- *Las 5 fases del ciclo de vida de un proyecto*. (2021, 21 junio). Platzi. [https://platzi.com/blog/fases-ciclo-proyecto/?utm\\_source=google](https://platzi.com/blog/fases-ciclo-proyecto/?utm_source=google)

- Mas, D. (2019, 5 junio). *Metodología Kanban: Pros y contras en la gestión de proyectos*. FHIOS Consultoría Estratégica. <https://www.fhios.es/metodologia-kanban-pros-y-contras/>
- Waelput, B. (2022, 7 septiembre). *¿Qué es y para qué sirve un diagrama de Gantt?* España. <https://www.teamleader.es/blog/diagrama-de-gantt>

## 11. Anexos

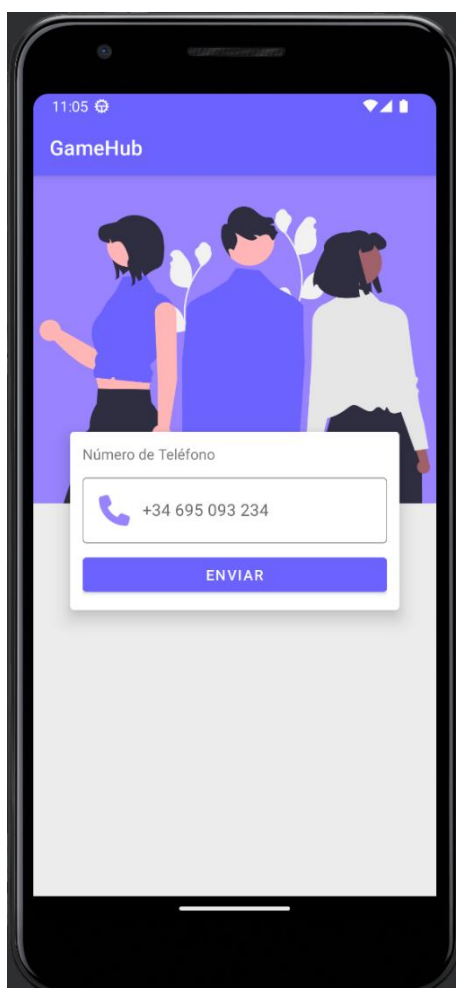
### 11.1 Manual de Usuario

#### 0. Consideraciones Previas

Para poder hacer uso de GameHub deberemos de tener una versión de **Android 5.0 Lollipop o superior**. Puede consultar la versión de su dispositivo en los **ajustes del teléfono**.

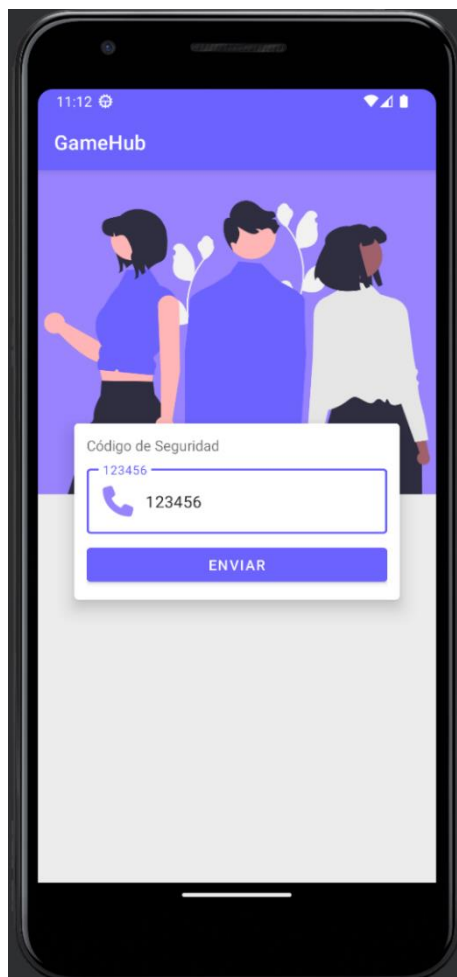
#### 1. Inicio de sesión

La primera vez que iniciemos GameHub deberemos pasar por el proceso de autenticación, para ello simplemente ingresaremos un número de teléfono en el campo correspondiente y presionaremos en el botón “Enviar”.



*Ilustración 1 del Manual de Usuario, Inicio de sesión. Fuente: elaboración propia.*

Acto seguido seremos redireccionados a la comprobación de si somos un robot, si dicha comprobación ha sido satisfactoria, seremos de nuevo redirigidos a la pantalla para ingresar nuestro código de seguridad que recibiremos por SMS o mensaje de texto en nuestro smartphone



*Ilustración 2 del Manual de Usuario, confirmación SMS. Fuente: elaboración propia.*

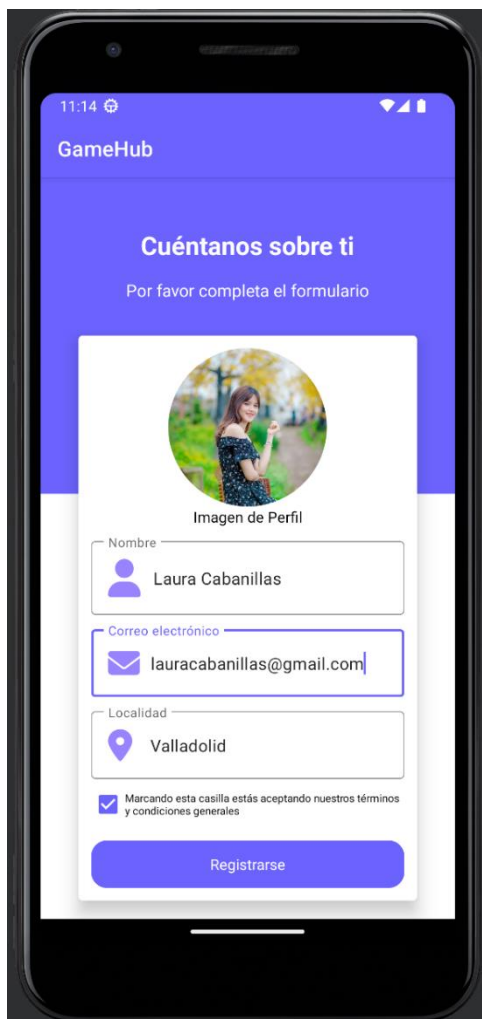


## 2. Registro de usuario

Cuando hayamos introducido nuestra clave OTP y no tengamos una cuenta registrada en GameHub será momento de crearla. Para ello proporcionamos todos los datos que se requieren cumplimentar:

- Nombre
- Correo electrónico
- Localidad
- Términos y condiciones generales de uso

Esta última casilla deberemos de aceptarla si es que aceptamos los términos de uso y de privacidad para el uso de GameHub.



11:14

GameHub

**Cuéntanos sobre ti**

Por favor completa el formulario

Imagen de Perfil

Nombre

Laura Cabanillas

Correo electrónico

lauracabanillas@gmail.com

Localidad

Valladolid

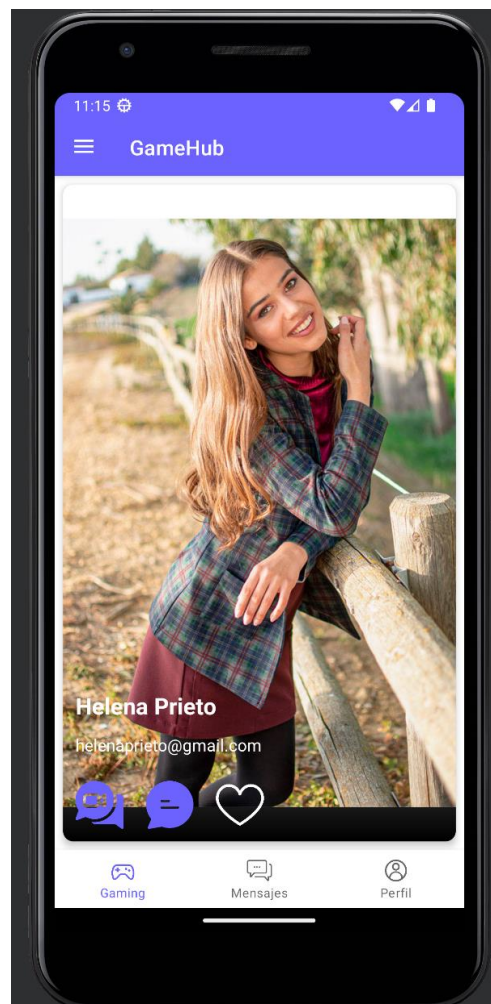
☒ Marcando esta casilla estás aceptando nuestros términos y condiciones generales

Registrarse

*Ilustración 3 del Manual de Usuario, Registro de usuario. Fuente: elaboración propia.*

### 3. Gaming

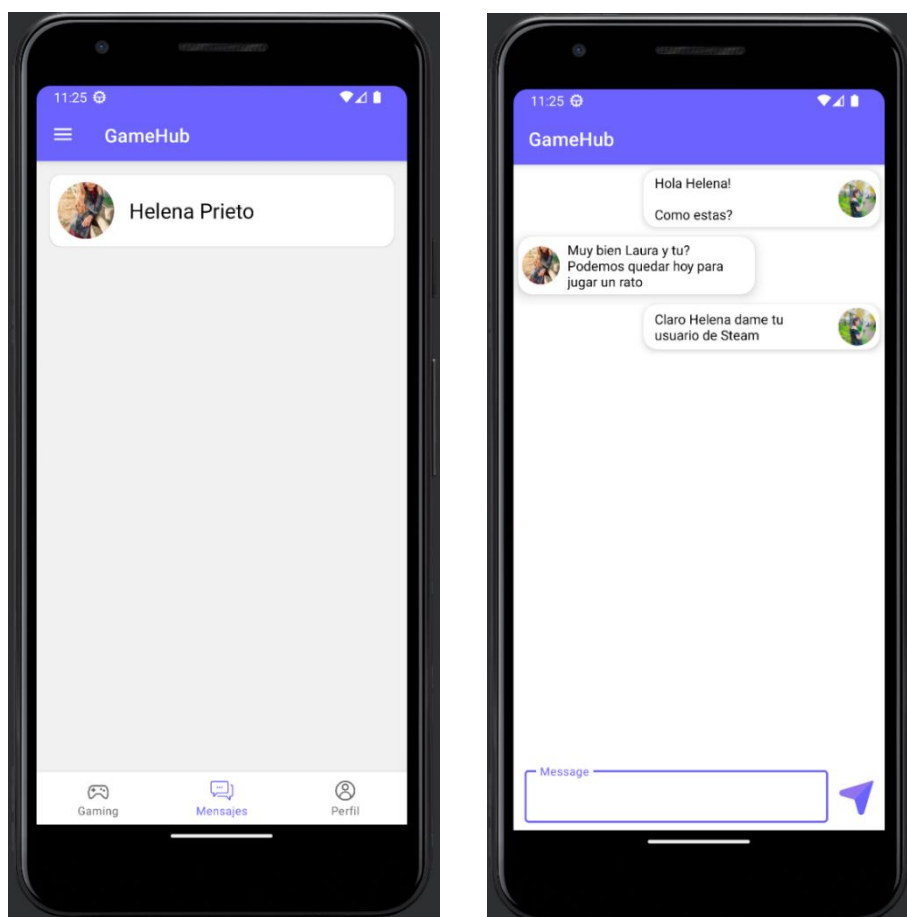
Una vez hayamos iniciado sesión nos cargará la aplicación en la ventana de Gaming donde veremos una lista de usuarios registrados en GameHub con los que conectar y poder enviar mensajes. Podemos navegar por las tarjetas de usuario deslizando a izquierda o derecha de la foto de perfil de cada usuario. Si queremos enviar un mensaje a alguno de estos, simplemente tocaremos en el icono de mensaje de la tarjeta de dicho usuario.



*Ilustración 4 del Manual de Usuario, Pantalla “Gaming”. Fuente: elaboración propia.*

## 4. Mensajes

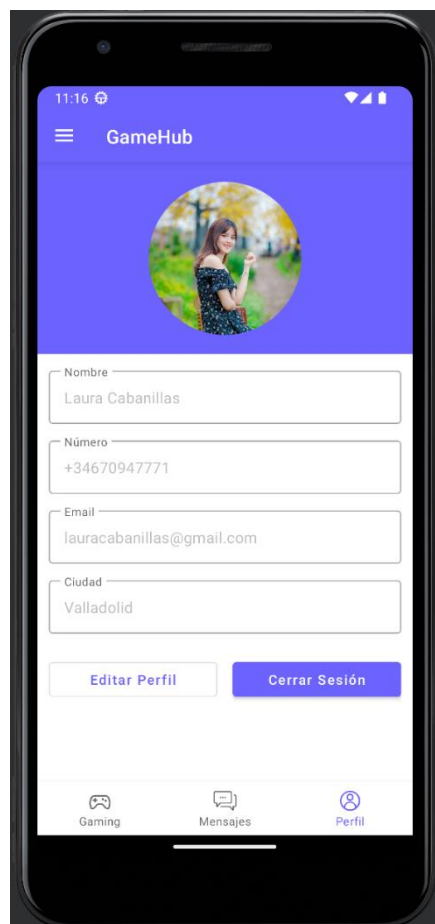
En el menú de navegación inferior, si pulsamos en el icono central iremos a la pantalla de mensajes, donde se nos mostrarán todos los mensajes que hayamos recibido o enviado a usuarios de GameHub. En el listado nos saldrán las diferentes personas con las que hayamos interactuado. Pulsando en cada uno de estos nombres podremos acceder a la sala de chat donde veremos todos los mensajes que hemos intercambiado.



*Ilustración 5 y 6 del Manual de Usuario, Pantalla “Mensajes”. Fuente: elaboración propia.*

## 5. Perfil

En el último icono de la barra de navegación podremos encontrar el apartado de Perfil. Donde podremos revisar nuestros datos de GameHub y modificarlos si es necesario pulsando el botón de “Editar Perfil”, una vez hayamos editados nuestros datos pulsaremos en el botón de “Guardar Perfil” y GameHub nos redireccionará al apartado de Perfil con nuestros datos actualizados.



*Ilustración 7 del Manual de Usuario, Pantalla “Perfil”. Fuente: elaboración propia.*

## 6. Informar de un error

La aplicación aún se encuentra en fase Beta, entonces es posible que experimentemos bugs o crasheos de la aplicación mientras está en ejecución. Desde el equipo de GameHub agradecemos cualquier ayuda para poder solucionar dichos problemas con la mayor brevedad posible. Si usted experimenta algunos de estos errores en la aplicación puede enviarnos un informe de errores en el menú lateral contextual donde deberá pulsar en la opción de “Informar de un error”, allí se le abrirá un formulario que deberá rellenar junto a una captura de pantalla que represente el problema que ha sufrido con GameHub. Esto nos servirá de mucha ayuda para el mantenimiento de la aplicación, parches de seguridad y próximas actualizaciones.

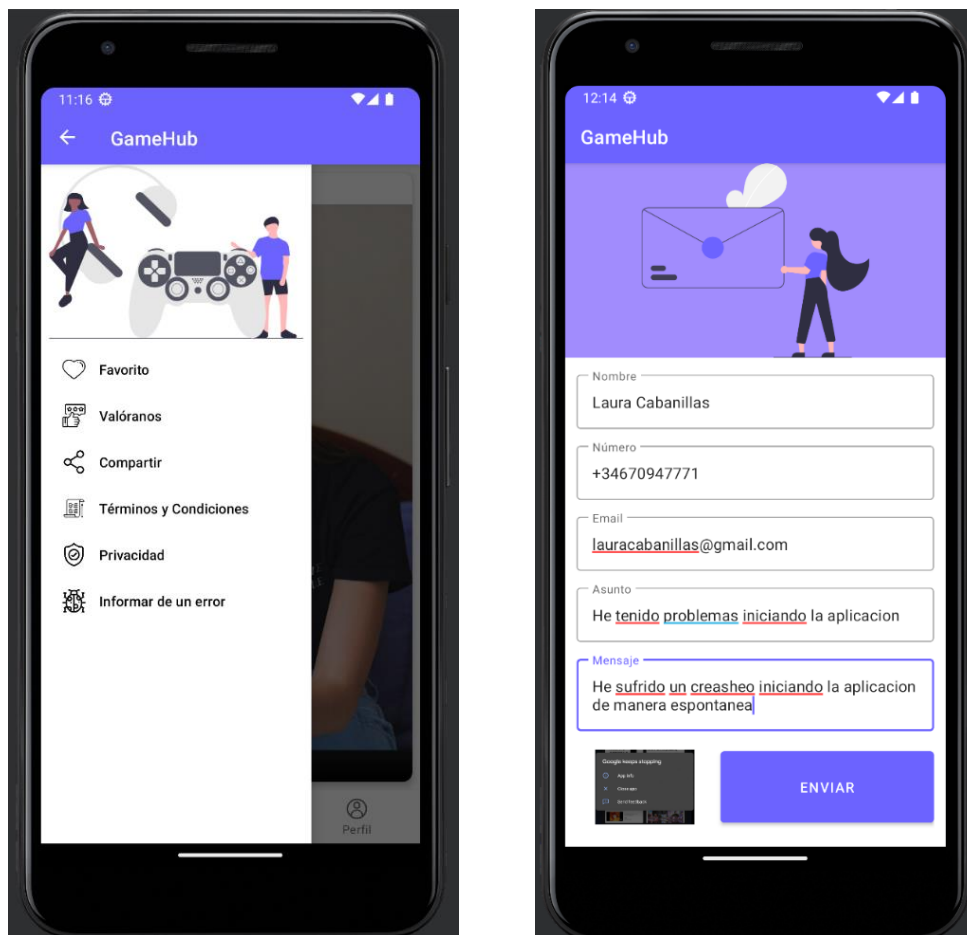


Ilustración 8 y 9 del Manual de Usuario, Menú lateral (izquierda) y Formulario de Informe de errores (derecha). Fuente: elaboración propia.

## 11.2 Glosario

- Bug: término anglosajón que refiere a errores de distinto tipo dentro de un software, fue acuñado este término en el siglo XIX cuando las partes móviles de la maquinaria compleja se quedaban atascadas entre ellas algún bicho.
- Crash: término inglés que hace referencia al error de ejecución en un software, generalmente acompaña con consecuencias desde tener bug's visuales hasta quedar inutilizable el software hasta que este no reinicie correctamente.
- API: de las siglas en inglés Application Programing Interfaces son un conjunto de funciones que hacen uso de bibliotecas de datos necesarias para la ejecución de ciertos softwares.
- Deadlines: término anglosajón referido al tiempo límite y muy usado en las fases de proyecto, representan las fechas límites de entrega de aspectos de proyectos en producción.
- Tech lead: Líder tecnológico, palabra del inglés usada generalmente para describir programadores senior o trabajadores muy cualificados en el área de las tecnologías de la información.
- Flutter: framework junto a SDK de código abierto creado por Google y es usado para hacer un desarrollo simultáneo de aplicaciones Android y iOS.
- Clave OTP: contraseña de uso único es usado generalmente en sistemas de verificación en dos factores.
- Receiver: término inglés con el significado de “persona que recibe” en español.
- Análisis ASO: App Store Optimization, en este tipo de análisis se estudian aquellos aspectos más relevantes para generar el mayor beneficio posible y posicionamiento en la tienda de aplicaciones en la que se haya lanzado dicha aplicación (PlayStore, AppStore, Microsoft Store, etc...).
- Softskills: en español, “habilidades blandas” hacen referencia a todas las habilidades relacionadas con el trabajo en equipo, empatía, resolución de problemas entre compañeros de equipo, etc...