UNIVERSITAT POLITÈCNICA DE VALÈNCIA

**ESCOLA POLITÈCNICA SUPERIOR DE GANDIA**

**Grado en Tecnologías Interactivas**

---



# “Desarrollo de un diario de salud electrónico para la autogestión y empoderamiento de los pacientes.”

***TRABAJO FINAL DE GRADO***

Autor/a:

**Diego Alejandro Aguirre Ramírez**

Tutor/a:

**Jaime Lloret Mauri**

Cotutor/a:

**Jesús Tomás Gironés**

***GANDIA, 2022***

# Agradecimientos

Me gustaría poder usar este apartado para agradecer a todos los profesores y compañeros de mi carrera GTI y mi familia por todos los buenos momentos, las enseñanzas , la paciencia y el compañerismo han hecho de mi la persona que soy ahora.

# Resumen:

En este proyecto se presenta el desarrollo de una aplicación Android dirigida al cuidado de la salud mental, buscando crear una comunidad virtual que funcione como red de apoyo. La aplicación cuenta con herramientas que le permitirán al usuario interactuar con otros mediante publicaciones y un chat, al mismo tiempo que podrá llevar una bitácora que le permitirá llevar un registro de sus emociones diarias.

A lo largo del documento se presentará la aplicación, las referencias tomadas de aplicaciones que ya operan en este mercado de la salud mental, las herramientas utilizadas para el desarrollo del proyecto, una descripción de los requisitos necesarios para desarrollar la herramienta y un análisis de todas las fases del proyecto. Que van desde el planteamiento del diseño hasta las pruebas realizadas tras la implementación. Por último, contaremos con un manual de usuario para entender el funcionamiento de la aplicación.

**Palabras clave:** Firebase, AndroidStudio, Aplicación Móvil, Chat, Base de Datos, Google, NoSQL

# ABSTRACT:

This project presents the development of an Android application aimed at mental health care, seeking to meet a personal need at a psychological and emotional level. The application has tools that will allow the user to interact with others through publications and a chat, while he will be able to keep a log that will allow him to keep a record of his emotions.

Throughout the document, the application will be presented, the references taken from applications that already operate in this market, the tools used for the development of the project, a description of the requirements necessary to develop the tool and an analysis of all the phases of the project. that go from the approach of the design to the tests carried out after the implementation. Finally, we will have a user manual to understand how the application works.

**Keywords:** Firebase, AndroidStudio, Mobile app, Chat, Database, Google, NoSQL

# Resum:

En este projecte es presenta el desenrotllament d'una aplicació Android dirigida a cura de la salut mental, buscant suplir una necessitat personal a nivell psicològic i emocional. L'aplicació compta amb ferramentes que li permetran a l'usuari interactuar amb altres per mitjà de publicacions i un xat, alhora que podrà portar una bitàcola que li permetrà portar un registre de les seues emocions.

Al llarg del document es presentarà l'aplicació, les referències preses d'aplicacions que ja operen en este mercat de la salut mental, les ferramentes utilitzades per al desenrotllament del projecte, una descripció dels requisits necessaris per a desenrotllar la ferramenta i una anàlisi de totes les fases del projecte. Que van des del plantejament del disseny fins a les proves realitzades després de la implementació. Finalment, comptarem amb un manual d'usuari per a entendre el funcionament de l'aplicacio.

**Paraules clau:** Firebase, AndroidStudio, Aplicació Mòvil, Xat, Base de Dades, Google, NoSQL

# Tabla de Contenido

# Tabla de Figuras

# 1 Introducción

Nadie esperaba que nuestras vidas cambiasen tanto en tan poco tiempo. Porque, la llegada de la pandemia del COVID-19 (SARS-CoV-2) nos tomó por sorpresa y cambio nuestro modelo de vida completamente, y, aunque es verdad que ya hemos avanzado mucho en la lucha contra el coronavirus siguen quedando una incertidumbre y miedo por nuevas variantes de la enfermedad y la desconfianza de que algún día se recupera la cotidianidad a la que estábamos acostumbrados hace un par de años.

Entre las secuelas de la pandemia, hay millones de fallecidos, crisis económica a lo largo del globo y un inquietante aumento en los casos de ansiedad y depresión. Según el jefe de Psiquiatría del hospital 12 de octubre de Madrid, Gabriel Rubio, se han duplicado las demandas a los servicios de Salud por casos de depresión en los adolescentes. Y, aunque la tendencia de la enfermedad tiende a la baja, la gente lo sigue pasando mal.

Si hablamos de depresión, uno de los problemas en España es que no tiene un registro que sigue los casos de depresión como otros países como Canadá. Y si ahondamos mucho más, es que realmente la gente que tiene estos padecimientos tiende a no buscar ayuda, debido a que sigue existiendo un estigma social que nos lleva a pensar que la depresión y ansiedad solo son una forma dramática de decir que estás pasando un mal momento.  Es decir, podría decirse que ni la gente ni la sanidad están gestionando bien este problema.

Es ahí que cuando existe un problema de confianza, es la tecnología la que debe abrirse camino para encontrar una solución. Y, mi contribución es una app con la que aportar mi granito de arena dónde, usando los conocimientos adquiridos durante mi carrera busco crear una comunidad virtual en la que las personas que estén pasando por un mal momento puedan compartir sus emociones en un espacio seguro para no sentirse solas y aisladas.

En los últimos años, se ha demostrado la versatilidad de los dispositivos móviles para poder suplir una enorme cantidad de exigencias, pues desde un dispositivo móvil se puede pagar la cena, comprar un electrodoméstico, chatear entre amigos y guardar archivos de lo más preciados en la nube. Sin duda, si se busca crear un nuevo producto software en esta era, ese debe ser en una aplicación móvil.

Claramente existen varios inconvenientes como la necesidad de conexión constante a internet y las limitaciones (cada vez menores) que puede presentar un dispositivo móvil. Pero, es la gran base de usuarios las constantes mejoras en la tecnología y la gran cantidad de funcionalidades

que se pueden desarrollar las que nos llevan a la indudable decisión de que una aplicación móvil tiene mucho potencial para desarrollar una herramienta como la que se busca desarrollar.

Por último, el avance de las tecnologías, cada vez más cercanas a las personas pueden hacer que lo que hoy es una aplicación móvil en el futuro acabe siendo una herramienta muy completa para realizar diferentes actividades como el control de una enfermedad crónica hasta ser una ayuda a nivel emocional en el día a día de las personas.

## 1.1 Objetivos

El desarrollo de una aplicación que en algún punto permita conectar profesionales de la salud mental, con el aumento de casos de personas con malestares tales como la depresión y la ansiedad requiere de una plataforma que sea conocida y también sea usada con la mayor frecuencia posible, y, aunque tenemos plataformas como IOS, es indudable que la demanda de Android supera todas las anteriores por bastante.

Otra de las ventajas de desarrollas una app, respecto a una aplicación nativa, es el bajo coste de implementación y la facilidad para distribuirse por las plataformas móviles como la Google Store.

Es importante aclarar el porqué de realizar una app para salud mental en Android, es que se ha realizado un estudio previo para ver la viabilidad de este, centrándonos en la cantidad de usuarios en España que utilizan este sistema operativo.

Por tanto, debido a que es una plataforma muy extendida a nivel nacional como internacional y por ser de desarrollo libre, Android es un candidato viable para usarse en este proyecto.

## 1.2 Objetivos del Proyecto

El objetivo principal de este TFG es desarrollar una aplicación tipo red social y/o comunidad virtual con un diario electrónico específico para personas afectadas con episodios de depresión y/o ansiedad causados por la Pandemia del SARS-CoV-2 (Covid19).

Otro de los principales objetivos es que los usuarios tengan las herramientas esenciales para poder utilizar la aplicación, como un chat y la capacidad de contactar con otras personas con las que sentirse apoyadas.

También es importante destacar el reto que supone realizar un proyecto software desde cero y la planificación de todas sus etapas, desde el estudio y análisis, hasta la codificación y el testeo. Siendo necesario una encontrar una alternativa a la metodología SCRUM para trabajar en solitario y puesta en práctica de los conocimientos adquiridos durante la carrera.

### 1.3 Fases de Realización

En este apartado se da una visión general de todas las fases planteadas y desarrolladas para el desarrollo del proyecto, que son normalmente contempladas en el desarrollo de una aplicación.

- **Estudio:** Es la fase donde se investiga los tipos de tecnologías disponibles para desarrollar el proyecto, también se pueden estudiar aplicaciones relacionadas para extraer una base de la que partir para el desarrollo, finalmente se puede estudiar las aplicaciones y viabilidad de este proyecto en el mercado.
- **Análisis:** En esta fase definiremos los requisitos mínimos que debe tener nuestro proyecto.
- **Diseño:** Se procede a desarrollar una arquitectura para la aplicación, pues ya se tienen claras las fases de estudio y análisis. El diseño se definirá por diagramas de pantallas, la interfaz y la base de datos
- **Implementación:** Programación en Android Studio de la aplicación.
- **Pruebas:** Se comprueba la calidad, viabilidad y efectividad de la aplicación.
- **Documentación:** Es importante desarrollar una documentación que acompañe nuestro software, que, de información sobre el código, la arquitectura y las fases de desarrollo.
- **Mantenimiento y actualización:** Cómo fase final se plantea la orientación que podría tener nuestra aplicación en un futuro, tanto a nivel de versiones que mejoren sus capacidades como ser una idea que pueda ampliarse a futuro.

### 1.4 Estructura del documento

En este documento se desarrollarán las fases ya mencionadas, además de posibles anexos con información relevante sobre el proyecto y por último el manual de uso de la aplicación móvil.

También, se explicará el uso de cada tecnología, herramienta y/o lenguaje que se use durante el desarrollo de la aplicación para así llevar a una mejor comprensión del producto final.

Por otro lado, las siguientes partes del documento contendrá el estudio del estado del arte, explicando, entre otras cosas, los aspectos más importantes del sistema operativo de Android, Firebase y una comparativa con algunas aplicaciones similares a la que se desea desarrollar.

Más adelante, en el capítulo 3, se hablará en detalle del análisis del proyecto, así como la metodología desarrollada, descripción de características y una lista de los requisitos para un correcto desarrollo.

En el capítulo 4 se hablará sobre el diseño, tanto a nivel de la arquitectura como el diseño de las interfases y el diseño de la base de datos.

En el capítulo 5 se explica la implementación de la aplicación, además, de las librerías y tecnologías usadas.

En el capítulo 6 se documentarán las pruebas a las que se someterá la aplicación para comprobar la efectividad de las anteriores fases.

En los últimos dos capítulos se manifiestan las conclusiones del trabajo y se propondrán ideas para mejorar el proyecto una vez ya concluido el Trabajo de Fin de Grado.

# 2 Estado del Arte

## 2.1 Android

Android es un sistema operativo basado en el *kernel* de Linux y otros softwares de código abierto. Inicialmente, fue desarrollado por Android Inc y más tarde, en 2005, comprado por Google. En principio, estaba diseñado para dispositivos móviles, pero a lo largo de su desarrollo se ha extendido a diferentes plataformas como por ejemplo relojes con Android Wear (ahora conocido como Wear OS) y automóviles con Android Auto.

Uno de sus aspectos principales, como ya se ha podido ver, es su orientación multiplataforma y la libertad que da a sus desarrolladores para crear aplicaciones sin tener que invertir mucho dinero, o directamente gratis.

### *2.1.1 Estructura de Android*

En la figura 3, podemos apreciar su estructura en capas.

Cada capa usa servicios ofrecidos por las anteriores y a su vez ofrece servicios a las capas superiores, como se observó en la anterior imagen.

- **Aplicaciones:** Esta capa contiene todas las aplicaciones que tenga por defecto Android y las añadidas posteriormente, ya sean de terceros o de su propio desarrollo.
- **Framework de Aplicaciones:** En esta capa se encuentran las herramientas de desarrollo para cualquier aplicación que se pueda desarrollar, usando el mismo conjunto de API y el framework representado en este nivel.

- **Librerías Nativas:** Todas las librerías usadas por Android se encuentran dentro de esta capa y se encuentran en lenguaje de C/C++ y proporcionan la mayoría de las características a Android.
- **Tiempo de ejecución:** Se encuentra al mismo nivel que las librerías. Está constituido por Core Libreries, que son librerías de clase Java y la máquina visual Dalvik.
- **Núcleo Linux:** Android utiliza el núcleo de Linux 2.6 como capa de abstracción para el hardware de los dispositivos. En esta capa, entonces, se encontrarán los drivers necesarios para que cualquier dispositivo pueda ser utilizado.

### *2.1.2 Versiones*

Las versiones de Android son también una parte esencial del mismo, pues desde su lanzamiento Android ha ido evolucionando continuamente agregando mejoras con cada actualización. Cada versión de Android tiene como nombre el de un postre, siguiendo el orden alfabético para darle nombre a cada una, pero esto solo fue hasta la versión 9, pues desde el 23 de agosto de 2019 Android dejó de usar nombre de postres.

- A: Apple Pie (v1.0): tarta de manzana.
- B: Banana Bread (v1.1): pan de plátano.
- C: Cupcake (v1.5): panqué.
- D: Donut (v1.6): rosquilla.
- E: Éclair (v2.0/v2.1): pastel francés.
- F: Froyo (v2.2) (abreviatura de «frozen yogurt»): yogur helado.
- G: Gingerbread (v2.3): pan de jengibre.
- H: Honeycomb (v3.0/v3.1/v3.2): panal de miel.
- I: Ice Cream Sandwich (v4.0): emparedado de helado.
- J: Jelly Bean (v4.1/v4.2/v4.3): gominola.
- K: KitKat (v4.4): tableta de chocolate con leche.
- Lollipop (v5.0-5.1.1): Piruleta.
- Marshmallow (v6.0-6.0.1): Nube de azúcar.
- Nougat (v7.0-7.1.2): Turrón.
- Oreo (v8.0-8.1)
- Pie(v9.0): Tarta
- Android 10
- Android 11
- Android 12

### *2.1.3 Versión Escogida.*

Otra cosa importante para tener en cuenta, respecto a las versiones, es que muchos de dispositivos Android siguen manteniendo versiones antiguas, lo que hace que existan muchas de ellas simultáneamente. Por eso es importante tener en cuenta la distribución de las versiones a la hora de desarrollar una App, pues desarrollarlo en una versión muy antigua tendrá muchas limitaciones, pero, usarla en la última puede producir que no haya muchos dispositivos que puedan usarla. Google facilitó esa tarea, agregando una API para ver la distribución de las versiones en Android Studio. De esta manera escoger una versión se convierte en una tarea más sencilla, lo cual tiene sentido pues esta información es de mayor utilidad para los desarrolladores de Android. [1].

Si se observa la figura 1, se entiende cual es la mejor versión para escoger, siendo esta Lollipop.

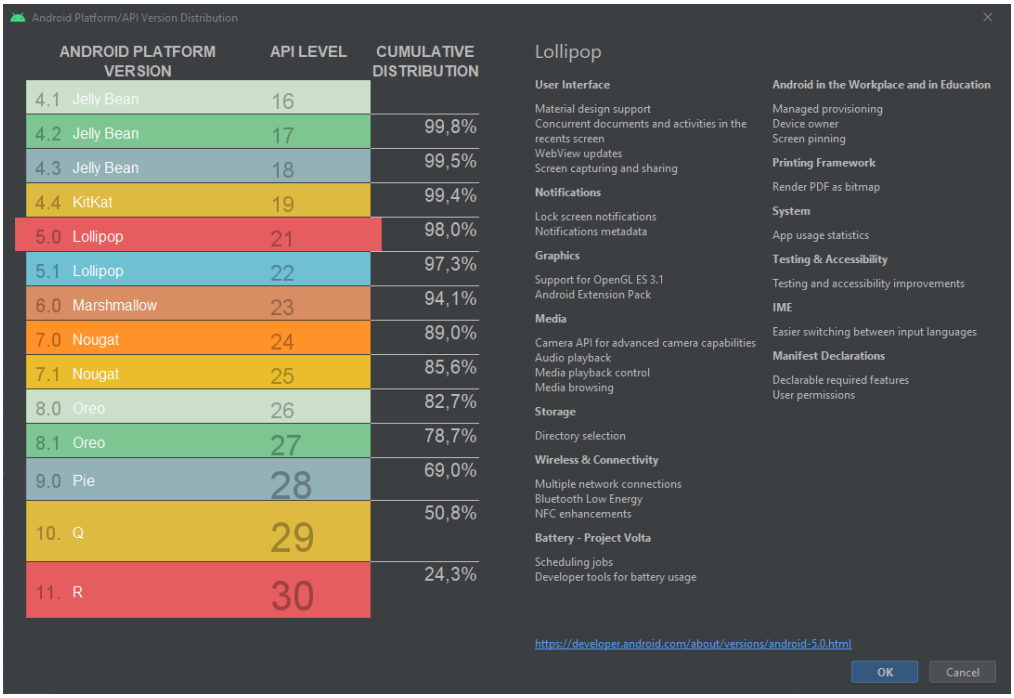


*Figura 1: Distribución de versiones de Android*

## 2.2 Enfoques

En 2021, un estudio de Research Markets incremento, el uso de aplicaciones de salud mental aumentará entre el 2021 y el 2026 pues se ha estipulado que la ansiedad y depresión han aumentado en el número de casos debido a la Pandemia. [2]. Además, muchas de ellas no son aplicaciones que requieran del contacto con un profesional.

Del otro lado, tenemos que la pandemia impidió a la gente salir de sus casas a lo largo de los confinamientos, pero aquellos que debían seguir realizando su trabajo o necesitaban encontrar una forma para mantener sus sesiones con psicólogos pudieron dar con una solución. Pues ya desde 2020 tenemos que los profesionales recurren a plataformas online para atender a un número de pacientes que va en aumento. [3]

Con toda esta información sabemos que existe una demanda creciente por las aplicaciones que ayuden a las personas con su día a día, pues la OMS advierte que luego de la crisis del Covid-19 vendrá la crisis de la gente rota por la angustia y el dolor. [3].

## 2.3 Referentes

Para dar más luz sobre el enfoque que tomará nuestro proyecto, vamos a realizar un breve análisis sobre aplicaciones que siguen los 3 modelos presentados. El modelo de médico / paciente, el modelo que aporta herramientas para que el propio usuario se sienta en un buen estado anímico y el híbrido entre ambos modelos.

### *2.3.1 Therapychat*

Es una app que ofrece un servicio online de psicología muy personalizado siguiendo el modelo médico / paciente. Esta app cuenta con más de 350 profesionales licenciados en España y Reino Unido, por lo que haciendo uso de su sistema de planes te ofrecen al especialista que mejor encaje con tu perfil.

### *2.3.2 Remente*

Una aplicación que permite a cualquier persona tener herramientas para mejorar sus hábitos a través de rutinas y metas, con la intención de así mejorar su bienestar, productividad y motivación. Es así como este modelo de herramientas ofrece retos y ejercicios para el crecimiento personal.

### *2.3.3 Meyo*

Una aplicación que ofrece herramientas desarrolladas con el apoyo de profesionales que buscan mejorar el estado de ánimo de sus usuarios, pero, lo que más destacamos de esta aplicación es que aparte de estar herramientas te permite platicar directamente con especialistas y profesionales, haciendo de este el modelo híbrido ya mencionado.

# 3 Análisis

Este capítulo centrado en la fase de análisis busca definir los requisitos que todo proyecto software necesita valorar durante su creación, además, de también dar una visión global de la arquitectura pensada para el sistema. En el siguiente capítulo se tendrá en cuenta este catálogo de requisitos para el diseño de todos los aspectos de la aplicación. Por ende, es de resaltar la importancia de realizar un buen análisis para todo lo que pueda devenir a nuestro proyecto software.

En el capítulo 3, entonces, buscamos asentar las bases de las que partirán el resto de las fases.

La arquitectura de la aplicación es muy similar a la de una red social, teniendo como características la conectividad, interacción, personalización, mensajería en tiempo real y la capacidad de publicar contenido.

## 3.1 Catálogo de requisitos.

Al igual que se desarrolló durante nuestra carrera de GTI, es importante saber que requisitos mínimos se buscan para el correcto desarrollo del producto. Por otro lado, buscamos que la metodología sea una adaptación de trabajo ágil, pero teniendo en cuenta la limitación de ser solo una persona.

En este catálogo se especifican los requisitos que se consideran prioritarios para el proyecto. Ya teniendo estudiadas las aplicaciones de la salud emocional y sus diferentes modelos podemos empezar a describir los requisitos mínimos para el diseño y desarrollo de la app. Requisitos funcionales

Los requisitos funcionales son aquellos que describen las interacciones que tendrán los usuarios con la aplicación.

#### 3.1.1.1 Gestión de usuarios

**RF1: Registro**

1. El usuario debe poder introducir sus datos en un formulario de registro.
2. El sistema se debe encargar de validar los datos.
3. El sistema mostrará un mensaje de error si alguno de los datos es incorrecto o no cumple las condiciones especificadas del formulario.
4. Si la validación es correcta, el sistema se encargará de guardar los datos en la base de datos.

5. Si el registro es correcto, la aplicación le redirigirá a la pantalla principal con su sesión ya iniciada.

**RF2: Identificación**

1. El usuario deberá identificarse con su nombre de usuario y contraseña para poder iniciar sesión. En el caso de que ya haya una sesión iniciada se entrará a la aplicación automáticamente.
2. El sistema se encargará de validar y permitir o denegar el acceso a la aplicación.
3. El sistema mostrará un mensaje de error en caso de que la validación no sea correcta.
4. Si la identificación es correcta se redirigirá al usuario a la pantalla principal.

**RF3: Cierre de sesión**

1. Cualquier usuario puede cerrar la sesión mediante el menú que indique “Cerrar cesión”.
2. El usuario será redirigido al menú de iniciar sesión/registro, pero en este caso sin estar logeado.

**RF4: Perfil del usuario**

1. Debe existir una pantalla en la que el usuario pueda consultar sus datos y pueda modificarlos.
2. Si modifica sus datos el sistema lo validará.
3. En el caso de que no haya un error, se actualizarán los datos en la base de datos.

#### 3.1.1.2 Aplicación

**RF5:Pantalla de inicio**

1. La pantalla contará con pestañas en la parte inferior donde el usuario podrá navegar en las funcionalidades de la app, las cuales son: Publicaciones, Chat y Cuaderno bitácora.
2. En publicaciones se podrán ver los contenidos subidos por otros usuarios.
3. En el chat se podrá entrar a los chats que se establezcan con los usuarios.
4. Cuaderno bitácora es donde el usuario anotará lo que vea relevante de su día a día.

**RF9:Notificaciones**

1. El sistema debe tener notificaciones que avisen al usuario de eventos importantes, como saber si alguien le ha escrito por el chat.

### *3.1.2 Requisitos no funcionales*

Requisitos complementarios que no juzgan el comportamiento del sistema si no de sus operaciones, como la apariencia, la fluidez o incluso la calidad.

**RNF1: Documentación**

1. Manual de usuario de la aplicación.
2. La codificación del sistema deberá ser clara y documentado en el caso que algún programador desee mejorar, arreglar o modificar alguna funcionalidad en el futuro.

**RNF2: Seguridad**

1. Para usar la aplicación hay que identificarse.
2. Solo será necesario identificarse una vez en el dispositivo.
3. Si el usuario no cierra la sesión, esta se mantendrá abierta para futuros usos de la aplicación.
4. Los datos personales, como la contraseña, serán cifrados.

**RNF3: Interfaz y usabilidad**

1. La interfaz debe ser sencilla e intuitiva.
2. La introducción de datos debe estar estructurada procurando evitar errores.

**RNF4: Rendimiento**

1. Los tiempos de esperan debe ser cortos, en caso de que la aplicación tarde demasiado en reaccionar se mostrará un mensaje de error.
2. Las consultas a la base de datos no deben cargar el dispositivo.

## 3.2 Metodología del desarrollo

Hay que seguir una metodología para el desarrollo de la aplicación, la cual permita estructurar, planificar y controlar el proceso de desarrollo del software. Existen muchos métodos y herramientas, como el Scrum, visto durante la carrera.

En este caso, dado que no trabajamos en equipo se usará una metodología ágil que permita el desarrollo de la aplicación de manera individual, dicho método será el de prototipado dado que se basa en la experiencia de usuario y la agilidad del desarrollo del producto.

Este modelo pertenece al desarrollo evolutivo, donde, partiendo de un mínimo producto viable que satisfaga las necesidades primarias de los usuarios, vaya evolucionando y mejorando mediante la retroalimentación de un cliente que pruebe el prototipo y de esta forma dar un

mejor entendimiento de lo que se debe hacer y permitiendo obtener resultados progresivos en un margen corto de tiempo.

Las etapas de un software construido siguiendo esta metodología son:

- Plan rápido.
- Modelado y diseño rápido.
- Desarrollo
- Entrega
- Retroalimentación
- Comunicación
- Entrega final.

Otro valor útil de este modelo es que permite que el cliente pese a no tener claro los requisitos si pueda dar claro los objetivos generales del software.

### *3.2.1 Contenido de cada prototipo*

El primer prototipo se crea con una base de datos Firebase y una interfaz Android donde el usuario pueda navegar entre las pantallas de la aplicación. Además, de poder permitir al usuario registrarse e iniciar sesión.

El segundo prototipo deberá permitir al usuario crear publicaciones, chatear y el uso de la bitácora, además de permitir la edición de perfil del usuario. También, de ser necesario, se refinará la interfaz para mejorar su aspecto o por si no son del todo intuitivas.

En el tercer y último prototipo corregirá detalles de los anteriores prototipos para mejorar lo mejor posible el flujo de uso de la aplicación. Antes de cada entrega de prototipo se realizarán pruebas para seguir lo indicado en la figura 2.

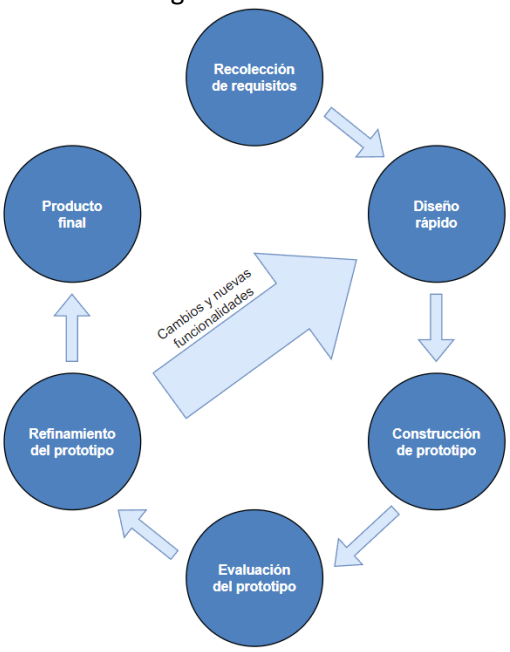


*Figura 2: Diagrama de la metodología por prototipos*

### 3.3 Base de Datos

La base de datos será realizada en Google Firebase lo cual permite un diseño e implementación muy rápidos. Por otro lado, ofrece una gran variedad de herramientas que nos ayudará a desarrollar las funcionalidades como las publicaciones, el chat y el almacenar la información de usuario, dándole una calidad al manejo de datos bastante profesional.

### 3.4 Seguridad

La seguridad de la base de datos será llevada por Google Firebase con sus herramientas de autentificación y base de datos.

# 4 Diseño

En este capítulo se describe acerca del proceso de diseño de la aplicación. Se ha realizado un diseño que abarca todos los requisitos del apartado 3.1 Catálogo de requisitos. El diseño proporciona una idea completa y general de la aplicación desarrollada para este proyecto. Además, se justifican las decisiones tomadas para el desarrollo del software. Es así como dividiremos este capítulo en los siguientes apartados:

- Arquitectura de la Aplicación
- Tecnologías utilizadas
- Diseño de Pantallas
- Diseño de la Interfaz
- Diseño de la Base de Datos
- Diseño de la Seguridad

Como es de suponer la aplicación ha ido sufriendo cambios a lo largo del desarrollo, en consecuencia, en este documento se verán diferentes diagramas de las diferentes versiones y prototipos. Esto se puede deber a factores como:

- Pruebas que no son superadas por dificultad de la implementación o que no superen un umbral de satisfacción de los requisitos.
- Ampliación o adición de algún requisito.
- Adquisición de nuevos conocimientos.

### 4.1 Arquitectura de la Aplicación

Desde el Capítulo 3 – Análisis, tenemos claro que la aplicación se basa en una plataforma estilo red social. En este proyecto lo que buscamos es que las personas puedan interactuar entre ellas a través de un chat estilo WhatsApp aparte de tener otras herramientas como la bitácora o la posibilidad de comunicarse con un profesional de la salud mental.

Para la aplicación lo más importante es tener en cuenta que el usuario que la usará es aquél que quiere compartir sus experiencias y dar consejos a otros a través de las publicaciones, además de permitir hablar a los usuarios entre ellos para darse más apoyo.

Por otro lado, es importante agregar una bitácora donde el usuario pueda ir haciendo un diario sobre sus emociones y así ver tanto su estado anímico a lo largo del tiempo como su evolución, pudiendo luego servir como una herramienta para un profesional que desee tener más información de su paciente.

Entonces, teniendo en cuenta el chat, la bitácora y la parte de las publicaciones nos centraremos en usar los conceptos más generales de las redes sociales como WhatsApp y Facebook para su desarrollo. Ya que tenemos la ventaja de que el usuario entrará en contacto con una aplicación que se hará familiar desde un primer momento.

## 4.2 Tecnologías utilizadas

En este apartado se mencionan las diferentes herramientas y tecnologías utilizadas durante el desarrollo del proyecto.

### *4.2.1 Draw.io*

Es una herramienta gratuita de creación y edición de diagramas. Lo cual nos permitirá diseñar nuestros diagramas de flujo, mapas mentales y representaciones gráficas. Además, permite guardar los archivos en la nube y/o descargarlos para importarlos en otras aplicaciones.

### *4.2.2 GitHub*

Utilizando el sistema de control de versiones Git, permite almacenar proyectos en la nube. En este caso usaremos la versión de escritorio, cuya interfaz agiliza el control de las versiones.

### *4.2.3 Android Studio*

Es el entorno de desarrollo integrado (IDE) oficial para aplicaciones Android y está basado en IntelliJ IDEA. Es un potente editor de código, además de contar con una gran diversidad de herramientas que aumentan la productividad cuando se desarrolla una app, como, por ejemplo:

- Un sistema de compilación flexible basado en Gradle
- Un emulador rápido y cargado de funciones
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Aplicación de cambios para insertar cambios de código y recursos a la app en ejecución sin reiniciarla
- Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra
- Variedad de marcos de trabajo y herramientas de prueba
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros
- Compatibilidad con C++ y NDK
- Compatibilidad integrada con Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine.

### *4.2.4 Firebase*

Es una plataforma de Google que funciona en la nube y que da acceso a herramientas que permite el desarrollo de aplicaciones móviles de mayor calidad sin aumentar la dificultad del desarrollo. En este caso nos centraremos en el uso de la bases de datos que posee la cual es

Cloud Firestore, además de la autentificación de usuarios para facilitar la creación de los métodos de registro e inicio de sesión.

### *4.2.5 Emulador Android Studio*

Es una herramienta de Android Studio que nos permitirá emular un dispositivo móvil donde se instalaran las diferentes versiones de nuestra aplicación y de esta forma agilizar el desarrollo y mejora de esta.

## 4.3 Diseño de Pantallas

Con un primer diagrama de pantallas se pretende mostrar una idea general de la aplicación, donde las funcionalidades estén definidas y los recursos que usaran cada una de ellas. En este apartado todos los diagramas se han realizado en Draw.io.

### *4.3.1 Primer prototipo*

Como se puede ver en la figura 3, en este primer diseño la aplicación ya está bastante definida, teniendo un esquema básico de cómo funciona el registro y el login, haciendo uso de la base de datos de Firebase. También se ven las diferentes pantallas que tendrá la aplicación de manera general. Por otro lado, tenemos el menú principal desde donde el usuario puede navegar a los tabs que posee la aplicación, los cuales son Post, Chat y User.

En Post el usuario es capaz de visualizar, comentar y subir contenido. En el Chat es capaz de ver los diferentes usuarios con los que tiene una conversación. Y, en el User es capaz de ver su perfil y editarlo. Además, se verá como cada uno interactúa con la base de datos de Firestore Database.

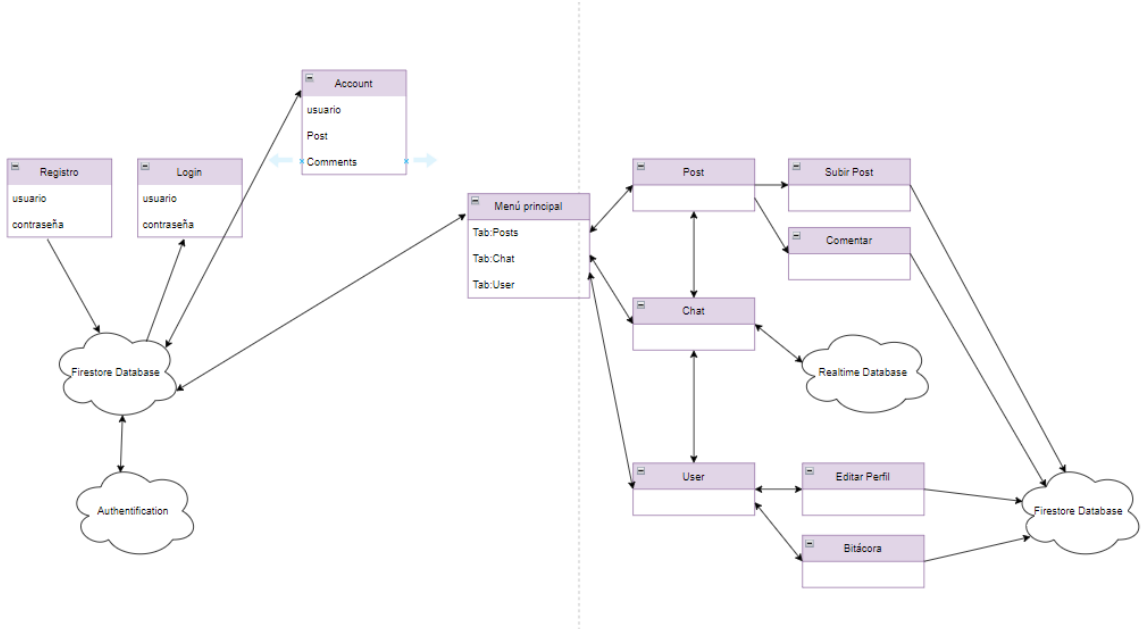


*Figura 3: Primer prototipo del diseño de la arquitectura de la app*

### *4.3.2 Segundo Prototipo*

Siguiendo con la figura 4, en el segundo prototipo se añaden dos tipos de registro el de usuario estándar y el de usuario profesional (especialista) el cual tiene la intención de dar un rol especial para diferenciarlo de un usuario común, de esta manera cuando un especialista de salud de la mente se registre en la app los demás usuarios podrían ser capaces de diferenciarlo. También se añadió una forma de acceder a otros perfiles desde sus publicaciones y poder abrir el chat desde ahí, pues en el anterior prototipo realmente no existía un método real para abrir un chat con un usuario en concreto.

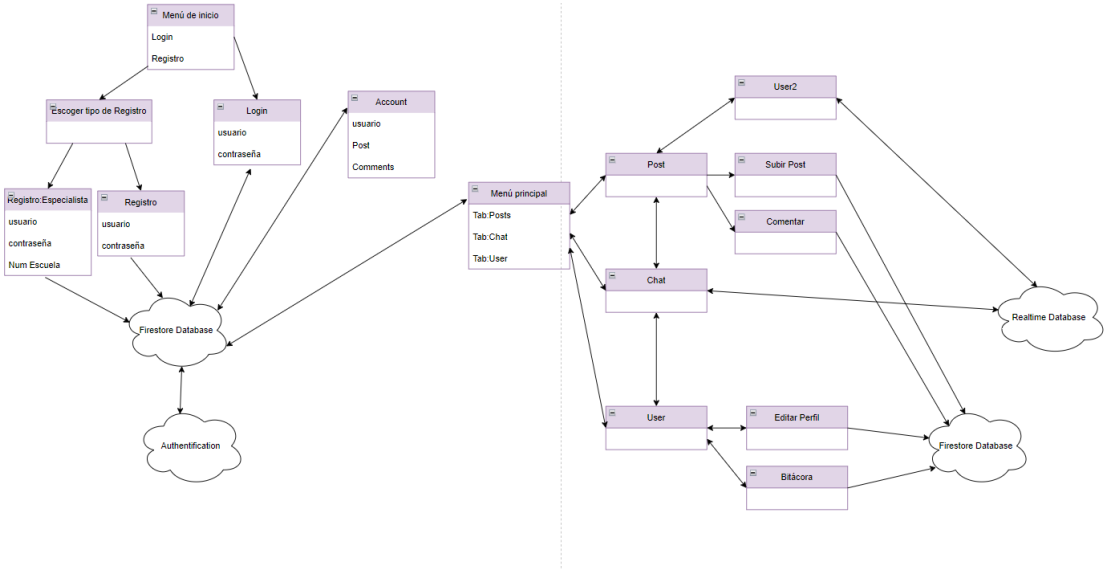


*Figura 4: Segundo prototipo del diseño de la arquitectura de la app*

### *4.3.3 Tercer Prototipo*

En la figura 5, el tercer y último prototipo corrige algunos detalles que no encajaban en un flujo ágil en el uso de la aplicación. Por lo que se decidió añadir la bitácora a un fragmento para tener un acceso más rápido a ella desde el menú deslizante inferior. También, se eliminó la pantalla de *Account* y también el registro para profesionales. También se decidió usar Cloud Firestore para todas las bases de datos, facilitando así el código de la App.

Por otro lado, se eliminó el registro para profesionales para agilizar el desarrollo de la aplicación.

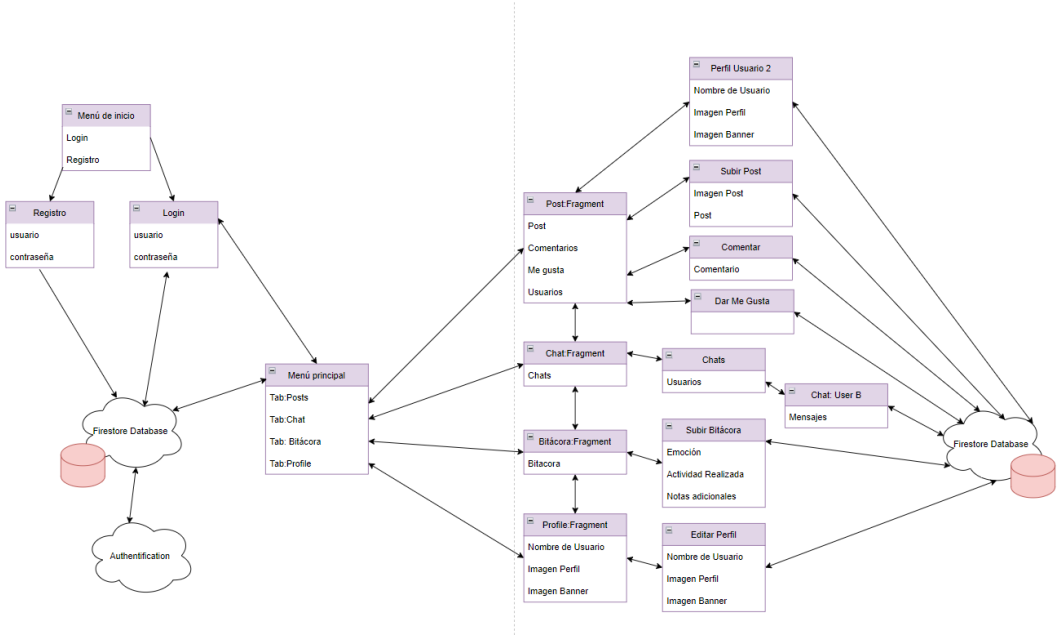


*Figura 5: Tercer y último prototipo del diseño de la arquitectura de la app*

## 4.4 Diseño de la Interfaz

La interfaz se ha diseñado con la intensión de ser familiar e intuitiva, como si se tratara de cualquier red social del mercado. Además de cubrir los requisitos mencionados en el capítulo 3.1 Catálogo de requisitos.

En este apartado se mostrarán todas las maquetas diseñadas en Draw.io de cómo se verá la aplicación. En primer lugar, tenemos el menú de inicio donde el usuario podrá entrar a los formularios de login y registro, además de tener una opción para realizar un inicio de sesión con una cuenta Google. Todas las pantallas son sencillas y bastante familiares pues constan con sus respectivos campos de usuario contraseña y el botón de aceptar. Obsérvese la figura 6.

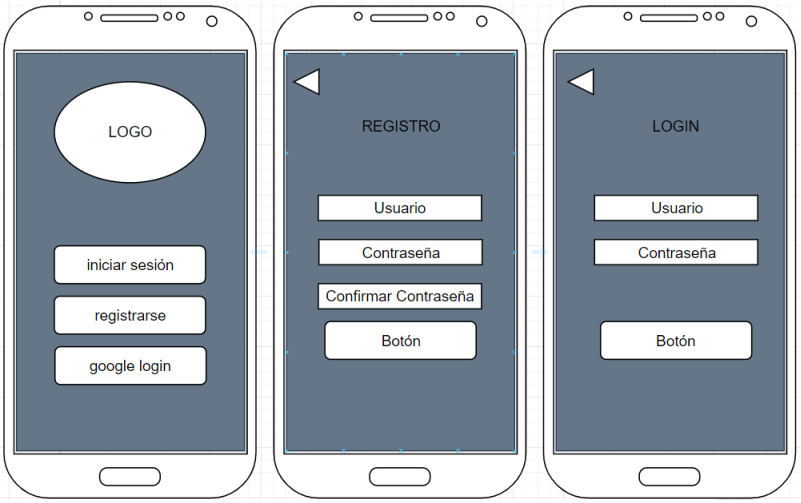


*Figura 6: Maqueta del diseño de la página de inicio, registro y login*

La actividad principal cuenta con 3 pestañas inferiores que te ayudan a ubicarte en las 3 pantallas de relevancia. Las publicaciones, que además será la página de inicio por defecto como suele ser en las redes sociales, el de chat y el de perfil, cada pestaña tiene su icono y se sombreará dependiendo de la pantalla que se muestre.

Por otro lado, en publicación tenemos un botón de acción flotante "+" el cuál sirve para que el usuario pueda crear una nueva publicación.

Todos los detalles mencionados se aprecian en la figura 7.

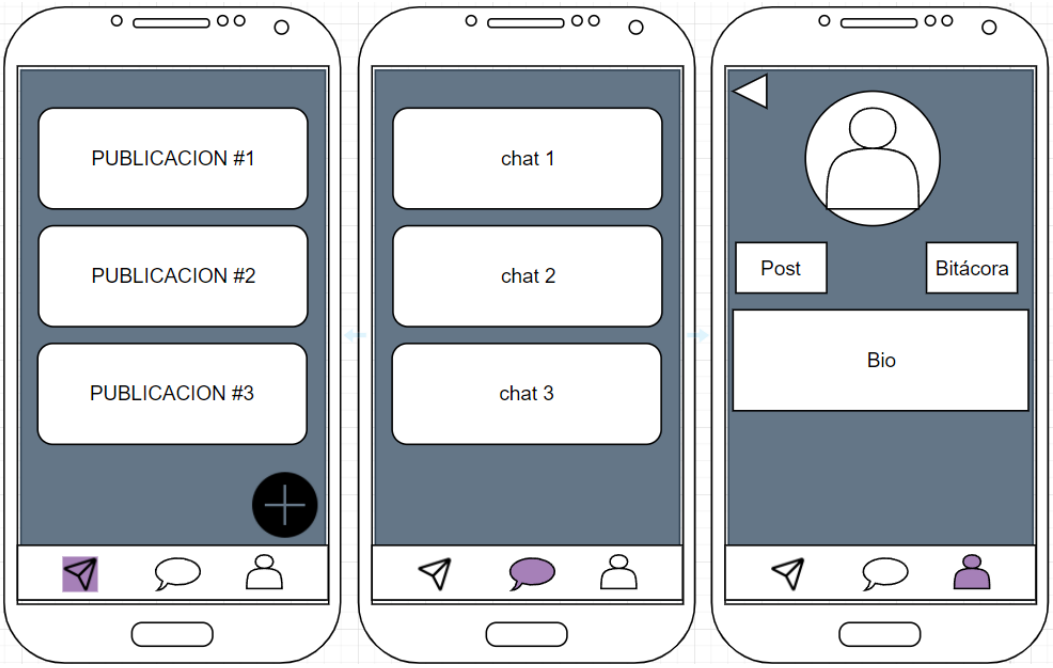


*Figura 7: Primera maqueta del diseño de las pestañas de Publicaciones, Chats y Perfil*

En la pantalla de subir publicación, figura 8, hay dos campos importantes uno donde el usuario puede subir una imagen desde su galería o tomar una foto y el segundo para escribir una cantidad de texto que sirva para explicar la imagen o simplemente redacte un pensamiento puntual como podría ser en una red social habitual.

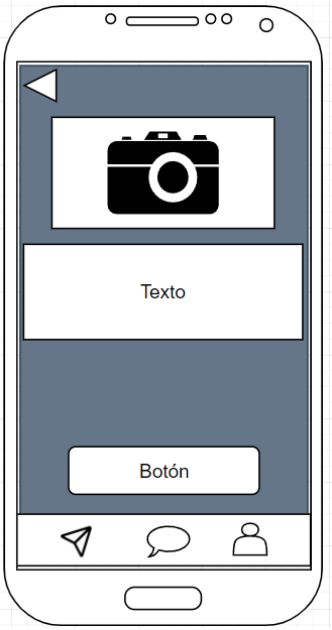


*Figura 8: Maqueta de la actividad de publicar*

Luego tenemos la pantalla del chat, figura 9, que se accede desde la pantalla de “Chats”, donde se podrá escribir con otro usuario igual que en cualquier aplicación de mensajería, por lo tanto, el diseño es el estándar para darle más familiaridad al usuario y pueda usarlo sin ninguna dificultad añadida.


*Figura 9: Maqueta del chat de la aplicación*

Por último, en la figura 10, tenemos la pantalla de perfil desde donde podemos visualizar el perfil, editar el perfil y acceder a la bitácora. En editar perfil tenemos la opción de cambiar la imagen de perfil y el nombre de usuario. Mientras que en bitácora tenemos un calendario que nos permite seleccionar el día de interés y exponer que emoción siente.

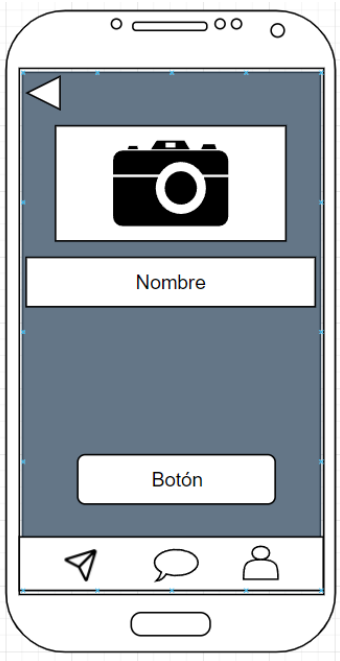


*Figura 10: Maqueta de editar perfil*

En la figura 11, una vez se seleccionada la emoción puede explicar en el campo de texto que le llevo a tener esa emoción o simplemente explicar un poco más detallado el día en cuestión, siendo posteriormente útil para tener un diario de sus emociones.


*Figura 11: Maquetas de la bitacora*

### 4.5 Cambios de la Interfaz (Tercer Prototipo)

Se añade la bitácora al menú inferior y también se cambia la presentación de esta, eliminando el calendario para escoger la fecha, ahora se verán las diferentes páginas de bitácora subidas con la emoción representada con un icono y algunos detalles extra como la fecha de subida de la página y la actividad realizada ese mismo día.

Otros cambios significativos es que ahora el menú inferior no sale en todas las actividades, evitando una navegación confusa.

Todos los cambios se pueden apreciar en la figura 12.


*Figura 12: Maquetas del nuevo diseño de las pestañas de la aplicación*

### 4.6 Escogiendo Base de Datos

A la hora de realizar un proyecto que utilice las bases de datos de Firebase hay que tener en cuenta las dos opciones que se ofrecen Realtime Database y Cloud Firestore.

Por un lado, Realtime Database es la base de datos original y trabaja en tiempo real usando un sistema eficiente y de baja latencia mientras que, por otro lado, tenemos a Cloud Firestore la cual es la base de datos más nueva, funcionando de la misma manera que Realtime Database con un plus de permitir consultas muchas más ricas y rápidas, además de tener un diseño más intuitivo para trabajar.

Realmente ambas opciones son buenas para nuestra aplicación, y, aunque las consultas pueden a llegar a ser sencillas haciendo que nos pudiéramos decantar por Realtime Database, se escogerá Cloud Firestore por su diseño intuitivo y su estructura de datos organizada mediante documentos y colecciones.

### 4.7 Diseño de la Base de Datos

En este apartado se tratan los aspectos de diseño relacionados con la base de datos. Como ya mencionamos en el apartado previo, los diagramas fueron realizados con Draw.io.

El diseño de Cloud Firestore es sencillo, consta de un conjunto de colecciones, figura 14, donde cada colección tiene una serie de documentos y a su vez estos documentos contienen la información almacenada en la base de datos, figura 13.

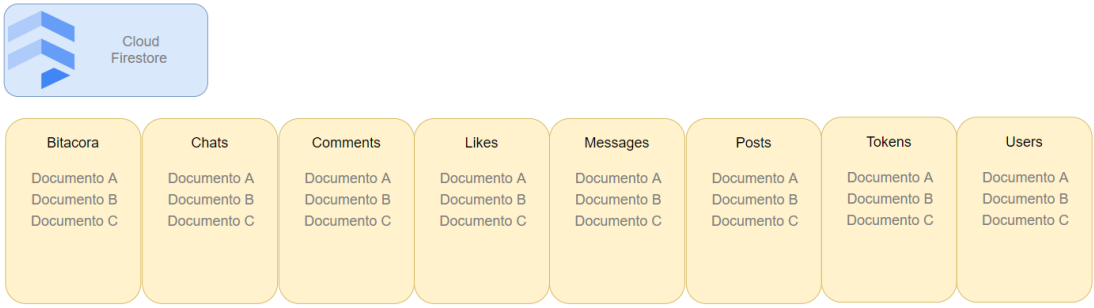


*Figura 14 Diseño de la base de datos Cloud Firestore*

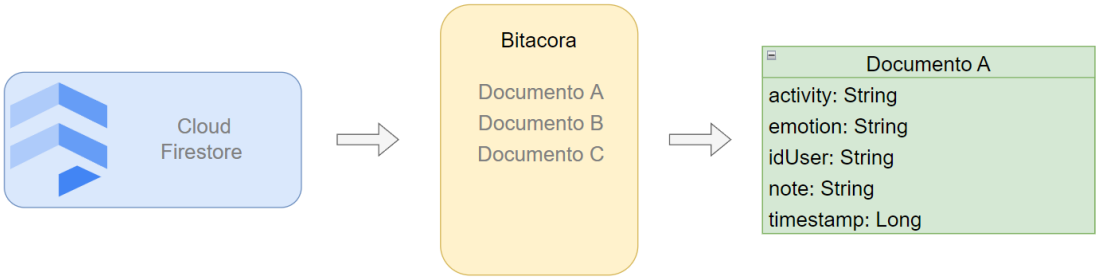


*Figura 13: Diseño de la colección de Bitácora*

### 4.8 Todas las Colecciones de la base de datos

Además del diseño de la Bitácora, ya antes visto tenemos otras 7 colecciones cuyos diseños son los que se pueden observar en las figuras desde la 15 hasta la 21.


*Figura 15: Diseño de colección de Likes*


*Figura 16: Diseño de colección de Comments*

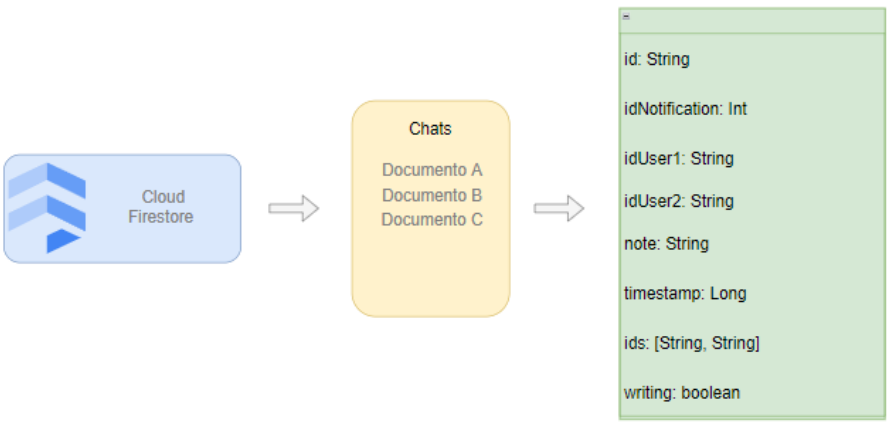


*Figura 17: Diseño de la colección de Chats*

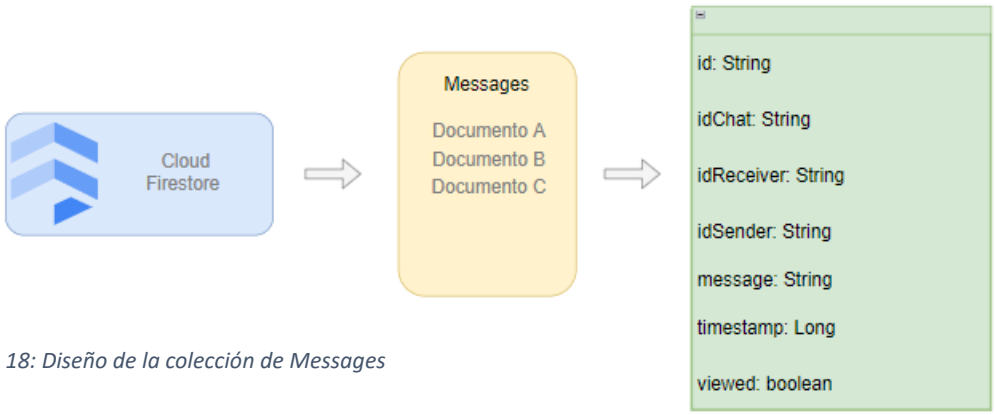


*Figura 18: Diseño de la colección de Messages*

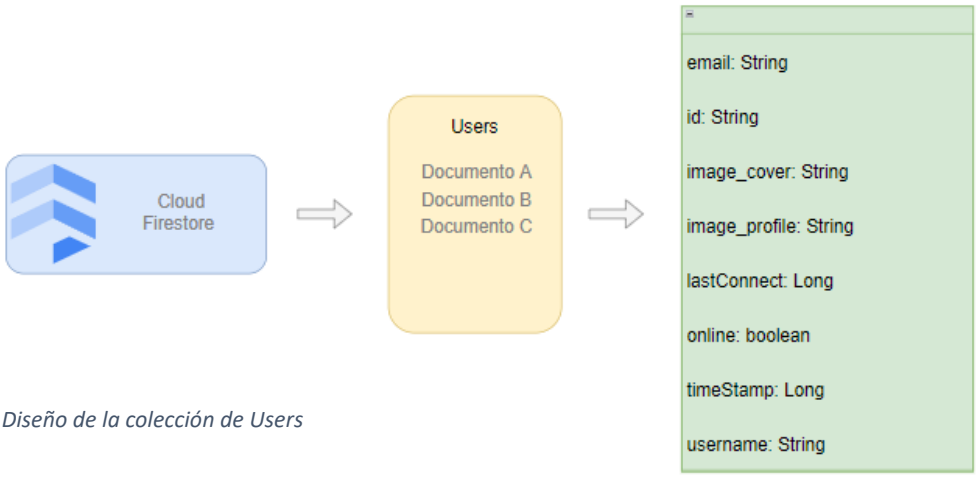


*Figura 19: Diseño de la colección de Users*

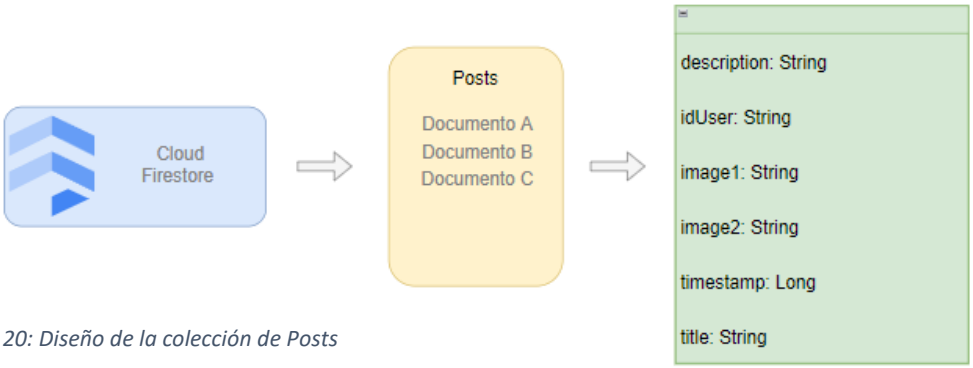


*Figura 20: Diseño de la colección de Posts*


*Figura 21: Diseño de la colección de Tokens*

### 4.9 Diseño de la Seguridad

Toda la seguridad referente a la base de datos está cubierta por Firebase mediante el uso de las reglas. En nuestro caso, solo pueden leer y escribir en la base de datos aquellos usuarios que estén identificados, también se debería evitar que un usuario pueda modificar los datos de otro.

### 4.10 Conclusiones de Diseño

Para resumir y sintetizar todas aquellas decisiones contempladas en el capítulo 4 se realiza este apartado ya que entender el contenido del Diseño será clave para el siguiente capítulo, el cual trata de la implementación. Por lo tanto, se toman en cuenta los siguientes puntos:

- El proyecto se realizará con Google Firebase, usando las herramientas de Firebase Autentificación y Cloud Firestore.
- Se ha realizado un breve estudio de las bases de datos que ofrece Firebase y se ha escogido Cloud Firestore.
- Se ha elaborado el diagrama de pantallas y el flujo de navegación entre ellas.
- Se han diseñado maquetas que intentarán parecerse al producto final.
- Se han diseñado las colecciones y documentos que tendrá Cloud Firestore para el manejo de los datos.

# 5 Implementación

En este capítulo se aborda la fase de la implementación del proyecto. Se expondrán algunos de los detalles más relevantes y algunas dificultades que se encontraron durante el proceso.

Es en esta fase donde se implementan los diseños realizados en el capítulo 4, habiendo seguido a su vez los requisitos en el proceso de análisis del capítulo 3. Como se detalló antes se utilizará Android Studio para la codificación de la aplicación

#### *5.1.1 Elementos básicos*

Los elementos que trataremos primero serán los de la parte lógica de la aplicación, entendiéndose como lógica todos aquellos aspectos no estéticos de la aplicación que le dan la funcionalidad. En este caso, la parte del código funcional se dividió en paquetes para seguir una estructura modular en el lenguaje de Java.

Posteriormente, hablaremos de las librerías utilizadas para mejorar el desempeño y apariencia de la aplicación. Y, finalizaremos con un repaso general de los elementos estéticos de la aplicación.

#### 5.1.1.1 Paquete Activities

Dentro del paquete Activities nos encontraremos con las clases que dan funcionalidad a las herramientas de la aplicación, entre ellas la bitácora, las publicaciones, el perfil de usuario, el chat, el login y el registro.

La clase HomeActivity dónde lo más fundamental es la generación de Token para posteriormente usarse en los Chats y la parte del código que se encarga de darle funcionalidad al menú inferior que se utilizará para navegar dentro de la aplicación, en la siguiente imagen podemos observar cómo funciona el menú inferior, siendo un Listener que va abriendo los diferentes Fragments de la aplicación. (Figura 22).

```
BottomNavigationView.OnNavigationItemSelectedListener navigationItemSelectedListener =
        new BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                if(item.getItemId() == R.id.item_home){
                    openFragment(new HomeFragment());
                } else if(item.getItemId() == R.id.item_chat) {
                    openFragment(new ChatsFragment());
                } else if(item.getItemId() == R.id.item_book) {
                    openFragment(new BinnacleFragment());
                } else if(item.getItemId() == R.id.item_profile) {
                    openFragment(new ProfileFragment());
                }
                return true;
            }
        };
```

*Figura 22: Código del navigation button*

Por otro lado, la clase ***MainActivity*** es la encargada de darle al usuario la posibilidad de hacer el Registro y el Login. También contiene la lógica para realizar el Login del usuario y una vez Logeado la clase se encargará de enviar el usuario al ***HomeActivity***. En las siguientes imágenes se puede ver, en el siguiente orden, la comprobación de saber si el usuario ya está logeado, figura 23 y la tarea de realizar el login, figura 24.

```java
@Override
protected void onStart() {
    super.onStart();
    if (mAuthProvider.getUserSesion() != null) {
        Intent intent = new Intent( packageContext: MainActivity.this, HomeActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }
```

*Figura 24: Código de comprobación para la existencia de una sesión*

```java
private void login(){
    String email = mTextInputEmail.getText().toString();
    String password = mTextInputPassword.getText().toString();
    mDialog.show();
    mAuthProvider.login(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            mDialog.dismiss();
            if(task.isSuccessful()) {
                Intent intent = new Intent( packageContext: MainActivity.this, HomeActivity.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(intent);
            }else{
                Toast.makeText( context: MainActivity.this,  text: "El email o la contraseña no son correctas", Toast.LENGTH_LONG).show();
            }
        }
```

*Figura 23: Código de la función login*

### 5.1.1.2 Paquete Adapters

En este paquete se encuentran las clases que se encargan de extraer la información de la base de datos y representarla gráficamente en la aplicación para que el usuario la entienda, como por ejemplo la bitácora, la lista del chat, los comentarios, los mensajes, las publicaciones y las sliders. En el ejemplo de la imagen se observa como utilizando un *ViewHolder* vamos obteniendo datos de la base de datos e insertándolos en los componentes visuales de la aplicación. (Figura 25).

```java
private void getUserInfo(String idUser, final ViewHolder holder) {
    mUsersProvider.getUser(idUser).addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            if (documentSnapshot.exists()) {
                if (documentSnapshot.contains("username")) {
                    String username = documentSnapshot.getString( field: "username");
                    holder.textViewUsername.setText(username.toUpperCase());
                }
                if (documentSnapshot.contains("image_profile")) {
                    String imageProfile = documentSnapshot.getString( field: "image_profile");
                    if (imageProfile != null) {
                        if (!imageProfile.isEmpty()) {
                            Picasso.with(context).load(imageProfile).into(holder.circleImageComment);
                        }
                    }
                }
            }
        }
    });
}
```

*Figura 25: Código de la función getUserInfo para extraer los datos de la Cloud Firestore e insertarlos en los componentes de Android*

#### 5.1.1.3 Paquete Channel

Este paquete contiene únicamente la clase de ***NotificationHelper***, clase encargada de permitir el uso de las notificaciones en nuestra publicación.

#### 5.1.1.4 Paquete Fragments

Este paquete contiene las clases de los 4 *Fragments* de los cuales se puede navegar con el menú inferior, Bitácora, Chats, *Home* (Publicaciones) y Perfil. En estas clases se encuentra también la lógica que le da funcionalidad a los componentes de los Fragments como lo serían los botones a los cuales les añadimos los métodos *OnClickListener*.

#### 5.1.1.5 Paquete Models

En el paquete de modelos se encuentran todos los objetos y sus respectivos métodos que se utilizaran en el código de la aplicación. Tenemos la Bitácora, el Chat, el Comentario, el *Like*, el Mensaje, la Publicación, el *Slider*, el *Token*, el Usuario y los objetos ***FCMBody*** y ***FCMResponse*** que son usados para las notificaciones. En la figura 26 se encuentra el ejemplo del objeto Like y sus métodos.

```
public class Like {

    private String id;
    private String idPost;
    private String idUser;
    private long timestamp;

    public Like() {

    }

    public Like(String id, String idPost, String idUser, long timestamp) {
        this.id = id;
        this.idPost = idPost;
        this.idUser = idUser;
        this.timestamp = timestamp;
    }

    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public String getIdPost() { return idPost; }

    public void setIdPost(String idPost) { this.idPost = idPost; }

    public String getIdUser() { return idUser; }

    public void setIdUser(String idUser) { this.idUser = idUser; }

    public long getTimestamp() { return timestamp; }

    public void setTimestamp(long timestamp) { this.timestamp = timestamp; }
```

*Figura 26: Objeto Like y sus métodos*

#### 5.1.1.6 Paquete Providers

El paquete *Providers* contiene todas las clases que sirven para realizar las tareas de lectura y escritura en la base de datos de Cloud Firestore. Es gracias a este paquete que el resto de las clases pueden trabajar haciendo uso de la base de datos y utilizar las funciones de Firebase como lo sería la autentificación, registro, login, login con Google, leer o escribir datos de una colección en concreto, crear documentos nuevos en la base de datos, entre otras utilidades.

Como ejemplo, tenemos el ***Token Provider***, el cual se utiliza para generar un Token que se utiliza para asignarle una ID al usuario y así poder generar un chat que contenga las ID de dos usuarios. En la siguiente figura 27 se ve el ***TokenProvider*** y la función ***create*** usada desde el ***Home Activity*** figura 28. Por otro lado, también podemos ver la función ***getUID*** del ***AuthProvider*** usada para obtener la ID del usuario, algo esencial para crear el Token.

```
import ...

public class TokenProvider {

    CollectionReference mCollection;

    public TokenProvider() { mCollection = FirebaseFirestore.getInstance().collection( collectionPath: "Tokens"); }

    public void create(String idUser){
        if (idUser == null) {
            return;
        }
        FirebaseMessaging.getInstance().getToken().addOnSuccessListener(new OnSuccessListener<String>() {
            @Override
            public void onSuccess(String s) {
                Token token = new Token(s);
                mCollection.document(idUser).set(token);
            }
        });
    }

    public Task<DocumentSnapshot> getToken(String idUser) {
        return mCollection.document(idUser).get();
    }

}
```

*Figura 27: Implementación de la clase Token Provider y sus funciones*

```
private void createToken() { mTokenProvider.create(mAuthProvider.getUID()); }
```

*Figura 28: Uso del create Token para generar un Token*

#### 5.1.1.7 Paquete Receivers

Contiene la clase de ***MessageReceiver*** que contiene la lógica para la recepción de los mensajes y las notificaciones que se utilizan en el Chat.

#### 5.1.1.8 Paquete Retrofit

Este paquete contiene la interfaz ***IFCMApi*** y la clase ***RetrofitClient*** necesarias para las peticiones Post que se usaran para generar las notificaciones. (Figuras 29 y 30).

```
public interface IFCMApi {

    @Headers({
            "Content-Type:application/json",
            "Authorization:key=AAAAC82pPQQ:APA91bFjjt-8GpweM3x6u87SqKNWKUu7aAHRcSRnb7Eg0Zgb-RMSN4xwNqNdh4UyLH-w2gM
    })
    @POST("fcm/send")
    Call<FCMResponse> send(@Body FCMBody body);

}
```

*Figura 29: Petición Post de la Interface IFCMApi*

```
public class RetrofitClient {

    public static Retrofit getClient(String url) {
        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(url)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        return retrofit;
    }

}
```

*Figura 30: Clase RetrofitClient*

#### 5.1.1.9 Paquete Service

Contiene la clase ***MyFirebaseMessagingClient*** para mostrar los mensajes de las notificaciones y también es dónde le daremos el aspecto a la notificación y algunas utilidades como poder responder un mensaje de chat a través de la misma publicación, como si fuera WhatsApp.

#### 5.1.1.10 Paquete Utils

Contiene clases muy valiosas para realizar algunas tareas tales como la compresión de imágenes para subirlas a internet más rápido, hacer cálculos con el tiempo y así cambiar su formato de

milisegundos a formato de días y métodos necesarios para comprobar si un mensaje enviado ha sido visto por el otro usuario. Cabe aclarar que estas clases son sacadas de repositorios de código abierto en la nube y son de uso público. En la figura 31 se ve un ejemplo de una función de la clase ***Relative Time***:

```
public static String timeFormatAMPM(long time, Context ctx) {

    SimpleDateFormat formatter = new SimpleDateFormat( pattern: "hh:mm a");


    if (time < 10000000000000L) {
        // if timestamp given in seconds, convert to millis
        time *= 1000;
    }

    long now = System.currentTimeMillis();
    if (time > now || time <= 0) {
        String dateString = formatter.format(new Date(time));
        return dateString;
    }

    // TODO: localize
    final long diff = now - time;
    if (diff < 24 * HOUR_MILLIS) {
        String dateString = formatter.format(new Date(time));
        return dateString;
    } else if (diff < 48 * HOUR_MILLIS) {
        return "Ayer";
    } else {
        return "Hace " + diff / DAY_MILLIS + " dias";
    }
```

*Figura 31: Función de timeFormatAMPM*

### *5.1.2 Librerías*

Para aumentar el rendimiento de la aplicación y que funcione en la mayoría de los dispositivos móviles se han implementado librerías de soporte de Google. Gracias a estas librerías dispositivos de versiones anteriores pueden utilizar bibliotecas más modernas, como es en el caso de ***androidx.legacy:legacy-support-v4:1.0.0***.

También se han usado la librería ***Picasso*** para una mejor manipulación de imágenes. Tenemos, además, las librerías de ***Firebase*** que permiten usar sus herramientas.

Otras librerías de interés son aquellas como ***Florent37:Shapeofview***, ***smarteist:autoimageslider***, ***Hdodenhof:circleImage*** y ***Spots-dialog*** que proporcionan a la interfaz una apariencia más profesional y moderna. Cambiando algunos *Views*, agregando Slider para imágenes o mejorando el diseño general de los Dialog de Android.

Finalmente, se destaca ***Retrofit***, la cual permite realizar las llamadas a la red, necesarias para realizar las notificaciones.

### *5.1.3 Funcionalidades de la App*

Una vez repasadas todos los elementos básicos del código se profundizará en los componentes más importantes de la aplicación, los cuáles son las publicaciones, chat, bitácora y notificaciones. Se explicará de la forma más breve y clara las parte más notables de su código para entender su funcionamiento.

#### 5.1.3.1 Funcionamiento de las publicaciones

Las publicaciones deben realizar las siguientes tareas, presentar los datos de la base de datos de Cloud Firestore de forma que el usuario las entienda y subir la información que publique el usuario a la base de datos, es decir, principalmente demos leer y escribir en la base de datos.

Cómo ya se indicó previamente la lógica para interactuar con la base de datos se encuentra en el paquete de Providers, en este caso tenemos la clase ***PostProvider*** que se encarga específicamente de las publicaciones. Cómo se ve en la siguiente figura 32, tenemos 6 métodos.

```
public class PostProvider {

    CollectionReference mCollection;

    public PostProvider() { mCollection = FirebaseFirestore.getInstance().collection( collectionPath: "Posts"); }

    public Task<Void> save(Post post) { return mCollection.document().set(post); }

    public Query getAll() { return mCollection.orderBy( field: "timestamp", Query.Direction.DESCENDING); }

    public Query getPostbyUser(String id) { return mCollection.whereEqualTo( field: "idUser", id); }

    public Task<DocumentSnapshot> getPostById(String id) { return mCollection.document(id).get(); }

    public Task<Void> delete(String id) { return mCollection.document(id).delete(); }

}
```

*Figura 32: Captura del código de la clase PostProvider*

Antes que nada, como ya se mencionó, al usar la base de datos de Cloud Firestore se debe tener en cuenta que se trabaja con colecciones y documentos. Por ende, los métodos que se han

creado, aparte de ser ofrecidas por la documentación de Google Firebase, sirven para generar colecciones, leer, modificar o borrar documentos.

En este caso, como se observa en la figura 38, al usar ***PostProvider*** se recibirá una instancia de la colección "Posts" que se le asigna al objeto ***CollectionReference*** llamado mCollection y es sobre esta colección que se trabajaran los demás métodos, por ejemplo, en GetAll() extraerá todos los documentos de la colección y los ordenará según el *timestamp* de forma descendente, de esta manera el usuario podrá ver todos los post en la pestaña de Home siguiendo un orden cronológico, donde las publicaciones más nuevas son las que se muestran primero. En la figura 33 se puede revisar su uso en el método onStart.


*Figura 33: Captura del HomeActivity, utilizando el método getAll()*

Luego de revisar el PostProvider, se pasará a explicar el ***PostAdapter***, ya se mencionó que el paquete *Adapter* se encarga de presentar los datos del usuario de una forma que se entienda. La metodología de todos los Adapter siempre consiste en crear un ViewHolder que tenga referenciados todos los componentes gráficos que se muestren por pantalla y luego extraer la información de la base de datos para introducirlos en el *ViewHolder*. En la figura 34 se ve como se genera el *Holder* y en la 35 como se extraen la información de la base de datos para agregarlos al *Holder*.

```java
@NonNull
@Override
public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.cardview_post,
    return new ViewHolder(view);
}

public class ViewHolder extends RecyclerView.ViewHolder {
    TextView textViewTitle;
    TextView textViewDescription;
    TextView textViewUsername;
    TextView textViewLike;
    ImageView imageViewPost;
    ImageView imageViewLike;
    View viewHolder;

    public ViewHolder(View view) {
        super(view);
        textViewTitle = view.findViewById(R.id.textViewTitlePostcard);
        textViewDescription = view.findViewById(R.id.textViewDescriptionPostcard);
        textViewUsername = view.findViewById(R.id.textViewUsernamePostCard);
        textViewLike = view.findViewById(R.id.textViewLike);
        imageViewPost = view.findViewById(R.id.imageViewPostCard);
        imageViewLike = view.findViewById(R.id.imageViewLike);
        viewHolder = view;
    }
```

*Figura 34: Captura del ViewHolder del PostAdapter*

```java
@Override
protected void onBindViewHolder(@NonNull final ViewHolder holder, int position, @NonNull final Post post) {

    DocumentSnapshot document = getSnapshots().getSnapshot(position);
    final String postId = document.getId();

    holder.textViewTitle.setText(post.getTitle().toUpperCase());
    holder.textViewDescription.setText(post.getDescription());
    if (post.getImage1() != null) {
        if (!post.getImage1().isEmpty()) {
            Picasso.with(context).load(post.getImage1()).into(holder.imageViewPost);
        }
    }
    holder.viewHolder.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(context, PostDetailActivity.class);
            intent.putExtra( name: "id", postId);
            context.startActivity(intent);
        }
    });

    holder.imageViewLike.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Like like = new Like();
            like.setIdUser(mAuthProvider.getUID());
            like.setIdPost(postId);
            like.setTimestamp(new Date().getTime());
            like(like, holder);
        }
    });

    getUserInfo(post.getIdUser(), holder);
    getNumberLikesByPost(postId, holder);
    checkIfExistLike(postId, mAuthProvider.getUID(), holder);
}
```

*Figura 35: Captura de pantalla utilizando el onBindViewHolder*

Otro detalle de la figura 35 es el método de dar *Like*, para ello la imageView que tiene el Holder que hace referencia al botón me gusta crea un nuevo *Like*, objeto ya visto en la figura 26, y le introduce los valores necesarios para generar un *Like* en la base de datos.

#### 5.1.3.2 Funcionamiento del chat

De ahora en adelante, ya sabiendo lo que realiza el Adapter y el Provider para cada elemento de la aplicación, se explicaran exclusivamente los algoritmos más importantes para el funcionamiento del Chat y Bitácora.

En el caso del Chat la complejidad reside en entender que se deben usar las ID de dos usuarios para la creación del Id del chat. En la figura 36 se observa cómo se ven los documentos de la base de datos.


*Figura 36: Captura de pantalla del documento de la colección chats*

El detalle más importante, entonces, es evitar que se formen dos chats, uno con el ID del User1+User2 y otro con la ID del User2+User1, para ello en el Chat Provider creamos el método de **getChatByUser1AndUser2** que consiste en usar un *ArrayList* donde se concatenan las dos posibilidades y luego se comprueba en la base de datos, en el caso de ya existir alguna de las dos opciones se evitará que posteriormente se cree otro chat, véase esto en la figura 37.

```
public Query getChatByUser1AndUser2(String idUser1, String idUser2) {
    ArrayList<String> ids = new ArrayList<>();
    ids.add(idUser1 + idUser2);
    ids.add(idUser2 + idUser1);
    return mCollection.whereIn( field: "id", ids);
}
```

*Figura 37: Método para obtener la ID de un chat*

#### 5.1.3.3 Funcionamiento de la bitácora

La bitácora funciona de manera similar a las publicaciones. El detalle más importante es que en la base de datos no se guarda un emoji que representa su emoción si no que directamente se guarda un *String.* Luego ese String debe comprobarse en la aplicación y ser sustituido por un emoji nuevamente, compruébese en la figura 38:

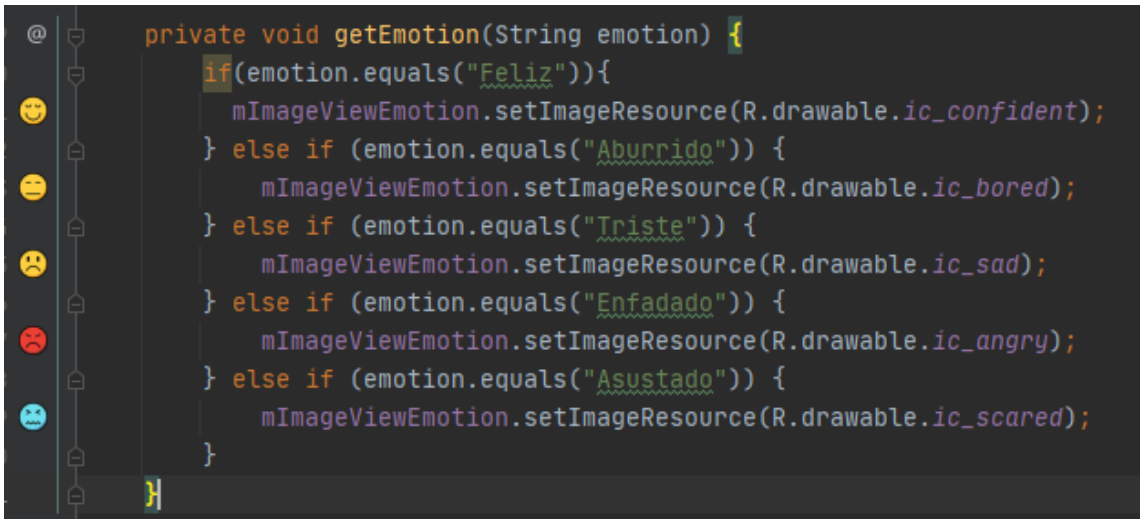


*Figura 38: Función para presentar un emoji dependiendo del String*

#### 5.1.3.4 Funcionamiento de las notificaciones

Para las notificaciones existen muchas soluciones posibles para generarlas, en este caso nosotros usamos el cliente REST para Android y Java, ***Retrofit*** y Firebase Cloud Messaging ( o FCM)**.**

Retrofit permite hacer peticiones del tipo **POST**, necesarias para poder enviar notificaciones, mientras que Firebase Cloud Messaging es un servicio multiplataforma de Google que permite el envío de mensajes.

Luego, se debe crear un token, para ello se crea un objeto *Token*, el cuál se encuentra en el paquete modelos y para subirlo a la base de datos tendremos un token *Adapter*.

Después, para enviar las notificaciones deberemos usar 3 librerías de ***Retrofit*** para realizar las configuraciones necesarias. Obsérvense las librerías en la figura 39:

```
implementation 'com.squareup.retrofit2:retrofit:2.4.0'
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
implementation 'com.squareup.retrofit2:converter-scalars:2.4.0'
```

*Figura 39: Librerías Retrofit*

Ahora, para configurar el FCM, se crea un nuevo objeto llamado FCM Response cuyos campos son la respuesta que mande la petición de Firebase cuando se envíe una notificación. Véase la figura 40.

```
public class FCMResponse {

    private long multicast_id;
    private int success;
    private int failure;
    private int canonical_ids;
    ArrayList<Object> results = new ArrayList<>();

    public FCMResponse(long multicast_id, int success, int failure, int canonical_ids, ArrayList<Object> results) {
        this.multicast_id = multicast_id;
        this.success = success;
        this.failure = failure;
        this.canonical_ids = canonical_ids;
        this.results = results;
    }

    public long getMulticast_id() { return multicast_id; }

    public void setMulticast_id(long multicast_id) { this.multicast_id = multicast_id; }

    public int getSuccess() { return success; }

    public void setSuccess(int success) { this.success = success; }

    public int getFailure() { return failure; }

    public void setFailure(int failure) { this.failure = failure; }

    public int getCanonical_ids() { return canonical_ids; }

    public void setCanonical_ids(int canonical_ids) { this.canonical_ids = canonical_ids; }

    public ArrayList<Object> getResults() { return results; }

    public void setResults(ArrayList<Object> results) { this.results = results; }
}
```

*Figura 40: Objeto FCMResponse*

Además, del objeto de FCM Response debemos crear el objeto FCM Body. En este caso, el objeto representa la forma que tiene la notificación de FCM, figura 41.

```java
public class FCMBody {

    private String to;
    private String priority;
    private String ttl;
    private Map<String, String> data;

    public FCMBody(String to, String priority, String ttl, Map<String, String> data) {
        this.to = to;
        this.priority = priority;
        this.ttl = ttl;
        this.data = data;
    }

    public String getTo() { return to; }

    public void setTo(String to) { this.to = to; }

    public String getPriority() { return priority; }

    public void setPriority(String priority) { this.priority = priority; }

    public String getTtl() { return ttl; }

    public void setTtl(String ttl) { this.ttl = ttl; }

    public Map<String, String> getData() { return data; }

    public void setData(Map<String, String> data) { this.data = data; }
}
```

*Figura 41:Objeto FCM Body*

También será necesario tener un Provider para las notificaciones, siendo este el *Notifcation Provider*, que puede verse en la figura 42. En este caso tenemos la *url* de la API de FCM para el envío de notificaciones y una llamada (método de Retrofit) para poder enviar el cuerpo de la notificación.

```java
public class NotificationProvider {

    private String url = "https://fcm.googleapis.com";

    public NotificationProvider() {

    }

    public Call<FCMResponse> sendNotification(FCMBody body) {
        return RetrofitClient.getClient(url).create(IFCMApi.class).send(body);
    }

}
```

*Figura 42: Captura de Notification Provider*

Con todos estos pasos ya se cumplen los pasos esenciales para crear una notificación. En la figura 43 se mostrará entonces como se puede generar una notificación mediante una función.

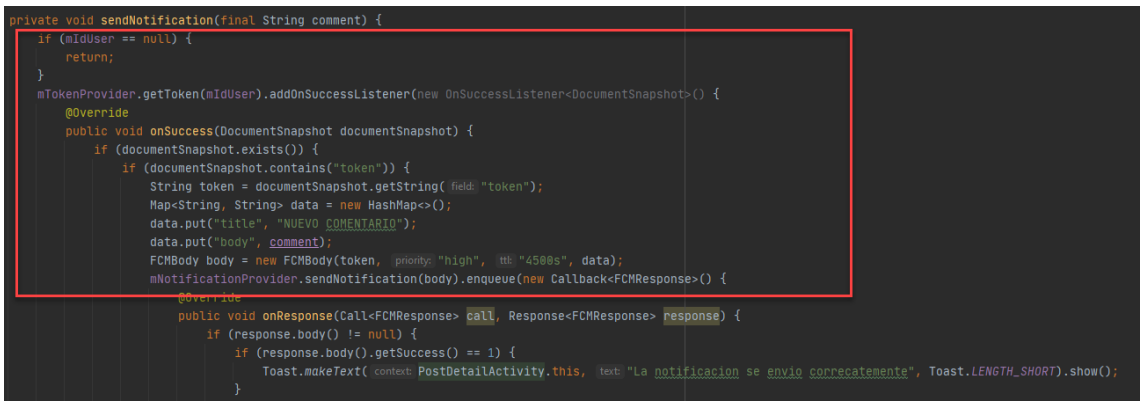


*Figura 43: Función para crear una notificación*

### *5.1.4 Interfaz*

Este apartado enumera los puntos más importantes respecto a la construcción de la interfaz del usuario.

#### 5.1.4.1 Menú

Un punto para destacar de la interfaz es el menú inferior, que permite navegar de forma ágil entre las herramientas más relevantes las cuales son el chat, las publicaciones, la bitácora y el perfil. Este menú utiliza una lista de ítems que contiene el icono y el título de cada herramienta (figura 44) y luego agregamos estas características a un botón de navegación (figura 45) para obtener la apariencia deseada (figura 46).

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/item_home"
        android:icon="@drawable/ic_home"
        android:title="Home" />
    <item
        android:id="@+id/item_chat"
        android:icon="@drawable/ic_chat"
        android:title="Chats" />
    <item
        android:id="@+id/item_book"
        android:icon="@drawable/ic_bitacora"
        android:title="Bitacora" />
    <item
        android:id="@+id/item_profile"
        android:icon="@drawable/ic_profile"
        android:title="Perfil" />
</menu>
```

*Figura 44: Código XML del menú*

```xml
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/container"
    >

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:background="@color/blue_200"
        app:itemIconTint="@color/white"
        app:itemTextColor="@color/white"
        app:menu="@menu/bottom_navigation_menu"
        />
</FrameLayout>
```

*Figura 45: Insertando configuraciones del menú al botón de navegación*

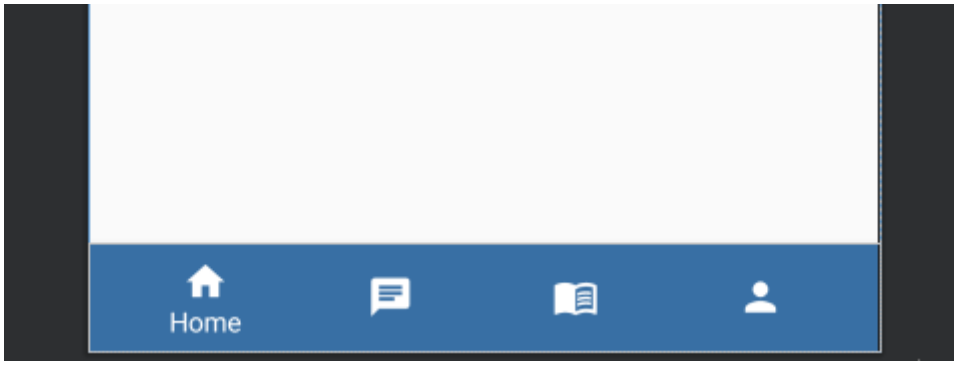


*Figura 46: Diseño del menú*

### *5.1.5 Fragments y Activities*

Posterior a lo mencionado, tenemos el diseño de los 4 Fragments, Home, Chats, Bitacora y por último usuario, donde el usuario podrá de una manera rápida interactuar con sus elementos de una manera intuitiva pues se ha buscado seguir el formato de red social con el cual ya las personas se sienten familiarizadas.

#### 5.1.5.1 Home

En el Home tenemos las publicaciones realizadas por los usuarios de la aplicación, dónde el usuario puede darle un me gusta haciendo uso del icono de corazón que contiene el CardView del post. También puede agregar una nueva publicación haciendo uso del fab con el icono con forma de cruz o entrar dentro de la publicación al pulsar sobre el CardView. Y, por último, un menú desplegable en la esquina superior derecha, de la cual se da la opción de cerrar sesión. (Figura 47).


*Figura 47: Apariencia Home y Publicaciones*

#### 5.1.5.2 Chats

En el Fragment de Chats tendremos la lista de los diferentes usuarios con los que se ha creado uno, mostrándose en el CardView el nombre del otro usuario, su imagen de perfil y el último mensaje que se envió en el chat. Si se presiona el CardView te llevará al Chat dónde podrás leer los bocadillos de mensaje, siendo de un color azul para los mensajes enviados y blancos para los recibidos, por otro lado, tenemos un doble check que indica si el mensaje ha sido leído por el otro usuario. (Figura 48).

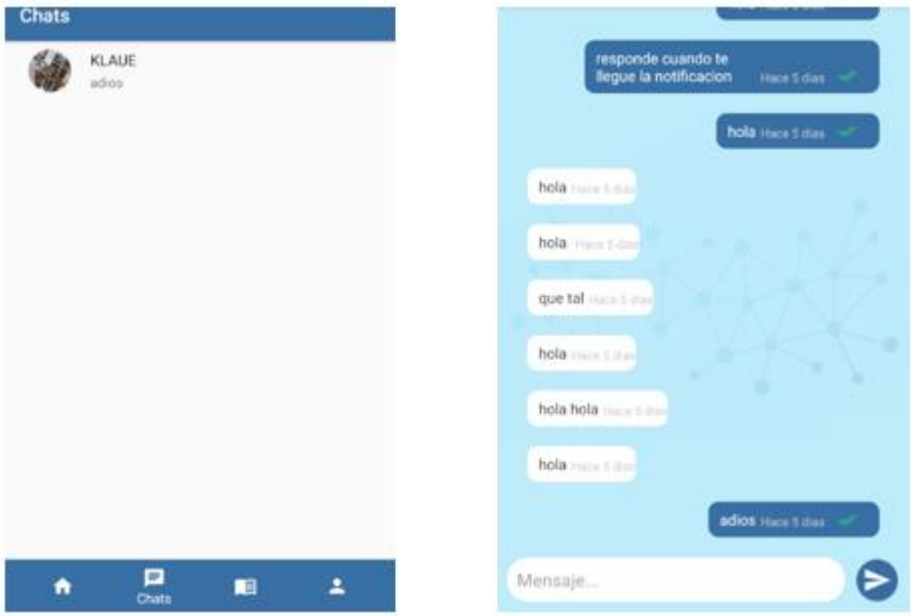


*Figura 48: Apariencia chats*

#### 5.1.5.3 Bitácora

En la bitácora tenemos un diseño muy parecido al de la publicación, con la diferencia que se muestra un emoticono que representa la emoción del título, una actividad que se realizó para generar esa emoción y una fecha en el título para llevar a cabo un seguimiento de las emociones a lo largo de un determinado tiempo. También se puede pulsar dentro de cada *CardView* para ver con detalle la información de la página de Bitácora y por último un *Fav* para agregar una nueva página de bitácora, donde el usuario tendrá un selector con unos emojis que representan una emoción y dos campos adicionales, uno para añadir la actividad que se realizó al momento de sentir esa emoción, y, las notas adicionales que puedan ser de interés para el usuario. (Figura 49).

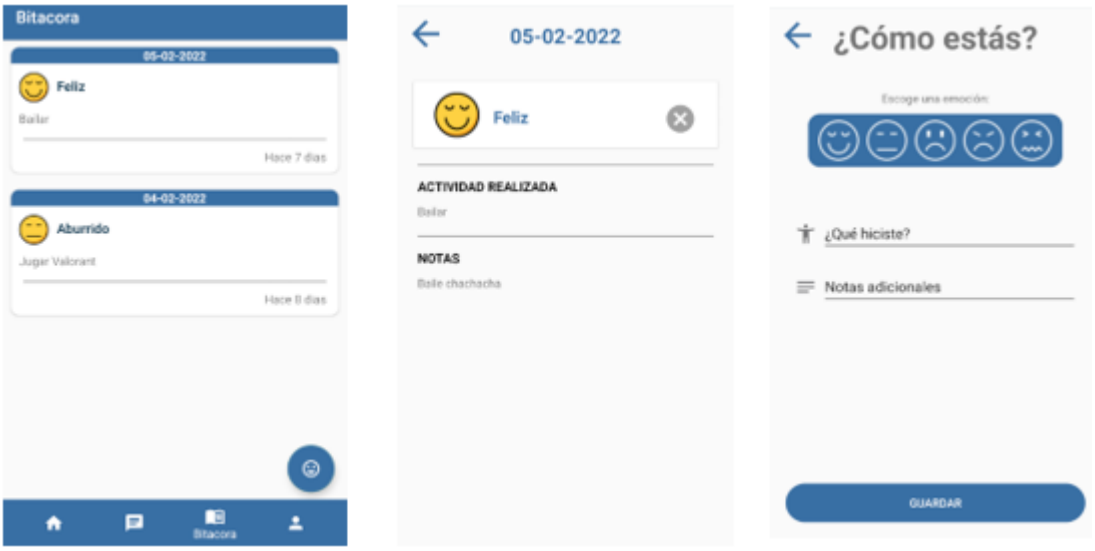


*Figura 49: Apariencia Bitácora*

### 5.1.5.4 Perfil

Por último, en el *Fragment* de Perfil se pueden la cantidad de publicaciones del usuario, su nombre de perfil, su email, la publicación que tiene subida y un botón que permite borrarla, además de una opción de editar perfil que le permitirá cambiar su imagen de perfil, su banner y su nombre de usuario. (Figura 50).

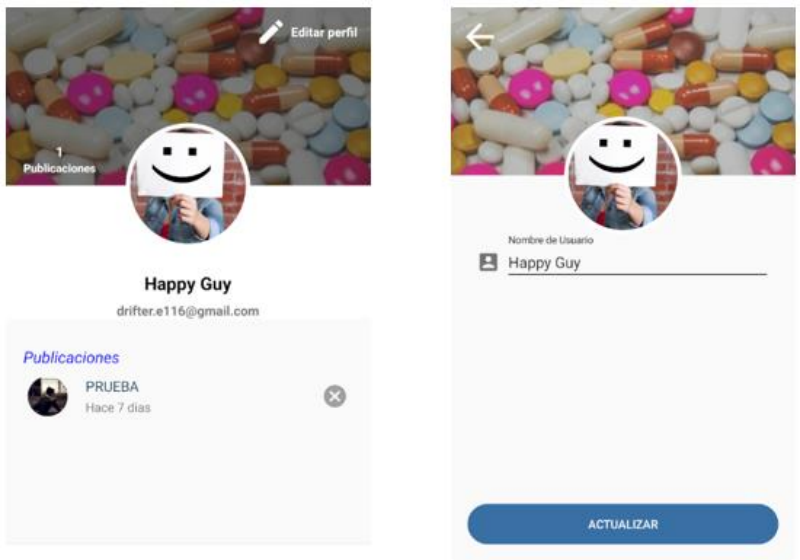


*Figura 50: Apariencia Perfil*

### 5.2 Conclusiones de Implementación

Este apartado servirá como un resumen del capítulo de implementación para dar un vistazo general a todos los aspectos revelados y dar algunas observaciones sobre dificultades encontradas.

Para el desarrollo de la aplicación se usaron paquetes para darle modularidad y así poder agregar a futuro nuevas funcionalidades o actualizar las actuales, también es una forma bastante organizada de llevar a cabo del proyecto, pues al haber una jerarquía se hace más fácil leer el código también con la intención de que futuros programadores puedan interactuar con él sin tener muchas dificultades.

Gracias al uso de Firebase las consultas a la base de datos se hacían mediante funciones predefinidas, lo cual facilitaba la lógica a la hora de realizar consultas.

Por otro lado, dada la simplicidad de la aplicación, parte del desarrollo se invirtió en una apariencia más moderna y profesional, para este fin se usaron diferentes librerías.

También se siguió bastante bien el diseño que se estipuló en el Capítulo 4 y se consiguió sintetizar los requisitos del Capítulo 3. Logrando que el resultado final se acerque bastante al deseado.

Por último, el código se puede encontrar en el siguiente repositorio de GitHub: https://github.com/Dialdroid/App_TFG/tree/main/app/src/main/java/com/example/medicinapp

# 6 Pruebas

Las pruebas fueron fundamentales para la implementación de la aplicación pues es la forma en la que se consiguió detectar todos los posibles errores y corregirlos para obtener un producto bastante más acabado y fiable.

En todas las pruebas se realizaron test de caja negra con las funciones que eran más problemáticas durante su desarrollo siendo estas las siguientes:

- Registro.
- Login.
- Subir un blog.
- Dar me gusta.

- Abrir un Chat.
- Notificaciones.

**Prueba 1:** En la prueba uno se estudió el comportamiento del Registro y Login. Por un lado, era importante que cuando un usuario se registrara todos los campos de la actividad de registro fueran rellenados, además, de que los campos especiales como el de email o el de contraseña tuvieran la información correcta. Y, por otro lado, cuando el usuario hiciera un login se debía comprobar que el usuario existe en la base de datos para evitar un error que causara que la aplicación fallara.

**Prueba 2:** En la segunda prueba se comprobó que ocurría cuando el usuario subía un blog sin imágenes ni texto a la base de datos. Esto ocasionaba que los campos que debían llevarse a la base de datos fueran nulos y al no coincidir con lo esperado en el objeto para crear un Post provocaba que la aplicación fallara. Para arreglarlo, se impide al usuario dejar campos en blanco y se le avisa mediante *Toast* los campos que debe rellenar para completar la tarea. Adicionalmente, al dar me gusta se creaba en la base de datos la información del Me Gusta, pero, no se eliminaba, ocasionando que la base de datos empezara almacenar datos repetidos, por ello se arregló la función para que cuando se pulse el Me Gusta este borre la información relacionada en la base de datos si es que ya existía un Me Gusta.

**Prueba 3:** Esta prueba fue sin duda la más problemática, pues al abrir un chat de un usuario A un usuario B, se generaba en la base de datos el Chat que contenía las ID de ambos usuarios, sin embargo, al abrir el chat del usuario B al usuario A, se creaba otro Chat que contenía las mismas ID solo que intercambiadas y esto provocaba que los mensajes de ambos chats fueran diferentes. Para arreglar este problema se corrigió la función que creaba el Chat haciendo que a la hora de crear un Chat se busque si ya existe uno que contenga las ID de ambos usuarios, sin importar quién fue el que lo abrió, de esta manera se evitaba el conflicto que provocaba que hubiera dos chats.

**Prueba 4:** Para trabajar con las publicaciones lo que se hizo fue utilizar **Postman** para enviar la primera Notificación a un dispositivo físico. Una vez comprada que la publicación se mandaba se pasó a utilizar la librería **Retrofit** para realizar las peticiones **POST** necesarias para realizar la notificación. (Figura 51)

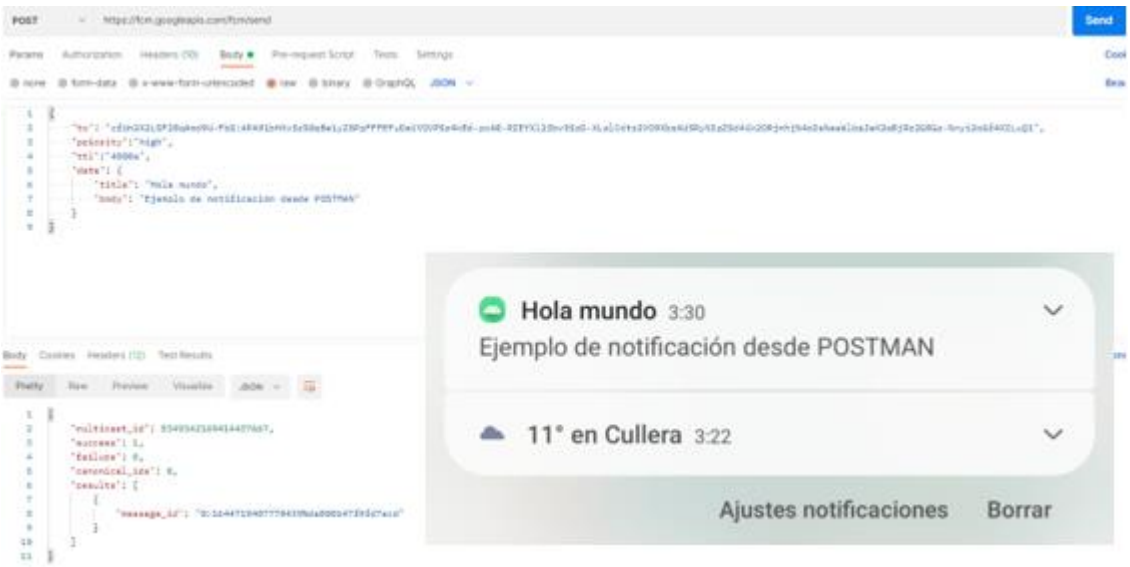


*Figura 51:Prueba de notificación*

# 7 Manual de Usuario

Ahora pasaremos al apartado extra de manual de usuario, en este apartado enseñaremos mediante capturas una demo de la aplicación.

Al arrancar la aplicación se mostrará la pantalla de inicio, dónde el usuario podrá iniciar sesión, registrarse o iniciar sesión con Google. Si selecciona el botón “REGISTRÁTE AQUÍ” pasará a la pantalla de registro donde encontrará el formulario. En el caso de que quiera usar el “Sign In” de Google se le enviará a una pantalla donde puede acabar rellenar un formulario y completar su usuario. (Figura 52)

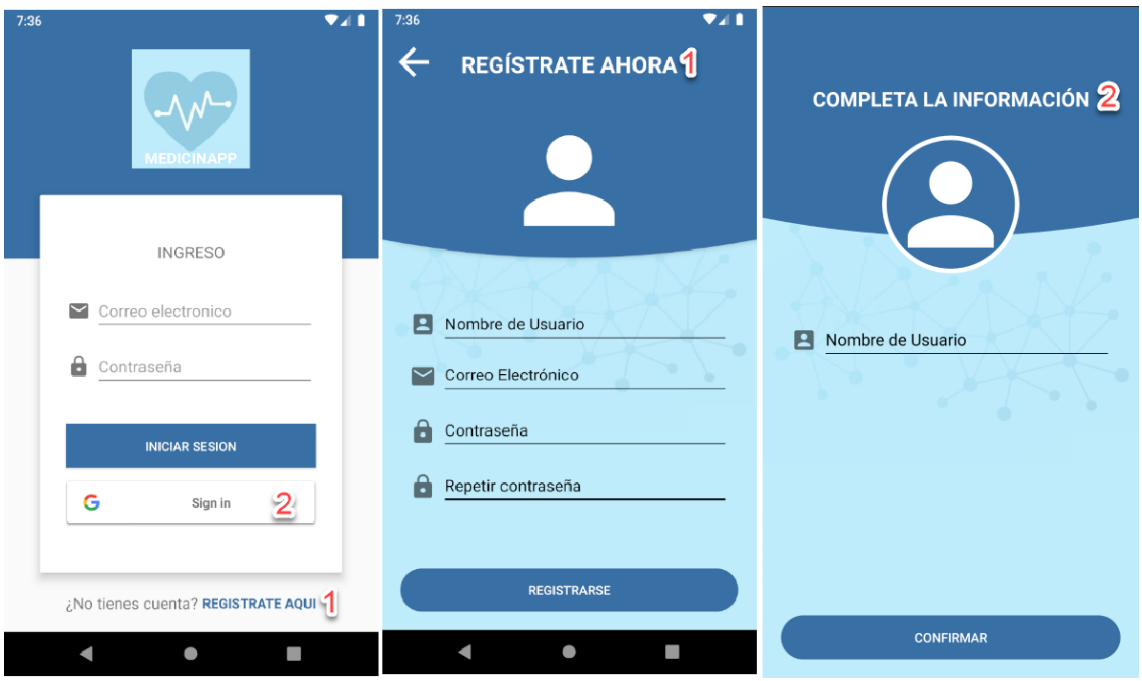


*Figura 52: Manual de usuario Login/Registro*

Una vez el usuario se ha logeado o registrado pasará al menú de Home donde podrá navegar entre las pestañas de home, chats, bitácora y perfil de usuario usando el menú de navegación inferior. (Figura 53)

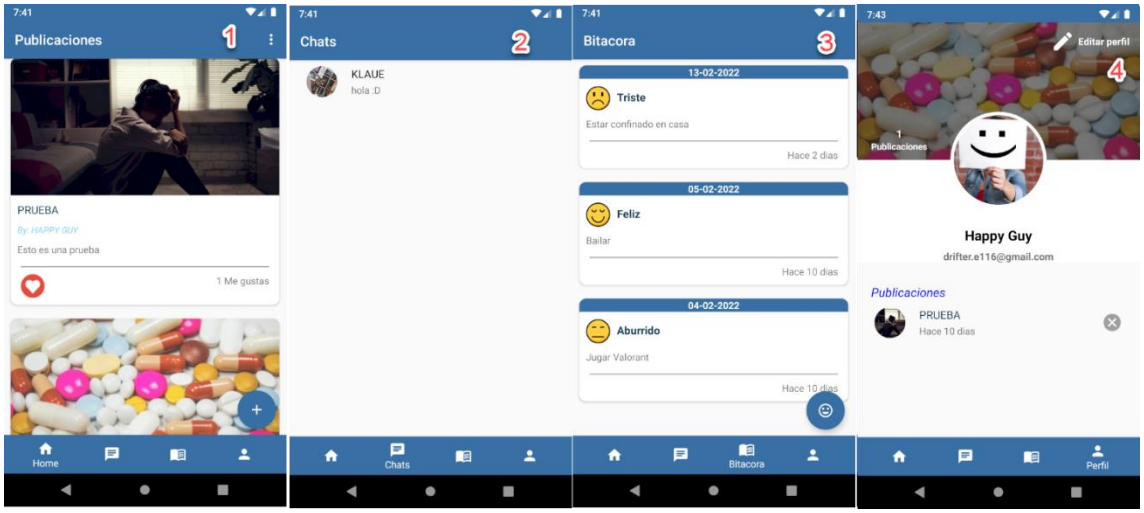


*Figura 53:Manual de usuario navegación de la app*

En publicaciones el usuario es capaz de darle al botón flotante con símbolo + (o también llamado fab) para poder realizar una publicación. Una vez en la actividad de crear publicación el usuario podrá usar los iconos de cámara para poder seleccionar una imagen o subirla directamente de la galería, después podrá usar los campos restantes de texto para ponerle un título y una descripción. (Figura 54).

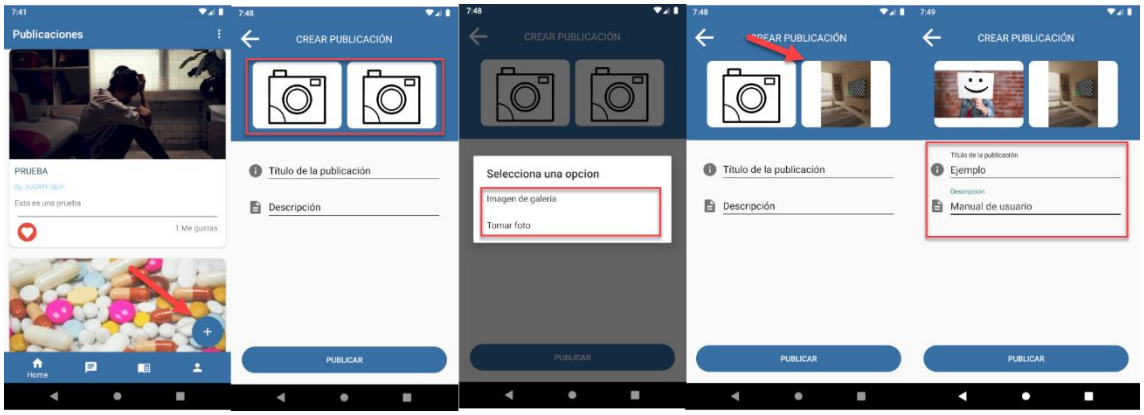


*Figura 54: Manual de usuario, creando una publicación*

Otra actividad que se puede realizar en el Home es poder darle me gusta a la publicación (figura 55) y visitar sus detalles al pulsar sobre la publicación en cuestión, desde ahí podrá comentar la publicación con el usuario, generando además una notificación (figura 56) además de ver el perfil del usuario de la publicación o abrirle un chat (figura 57).

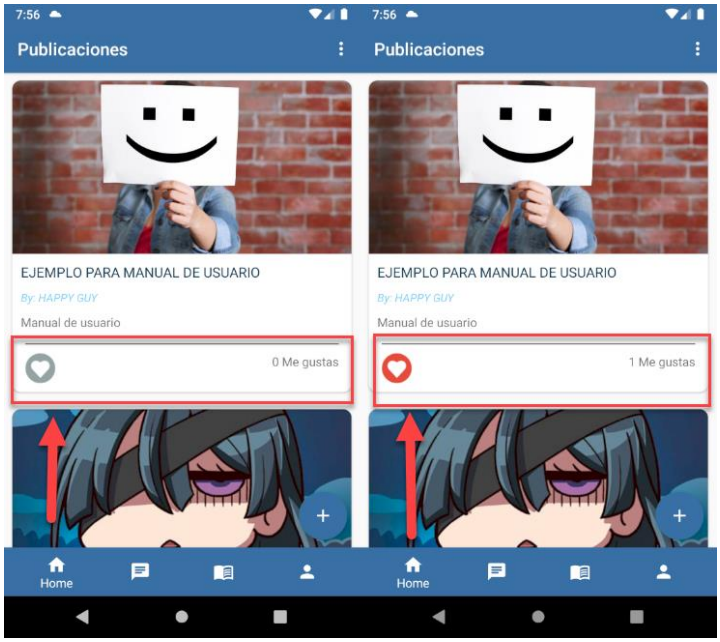


*Figura 55: Manual de usuario, dando me gusta*

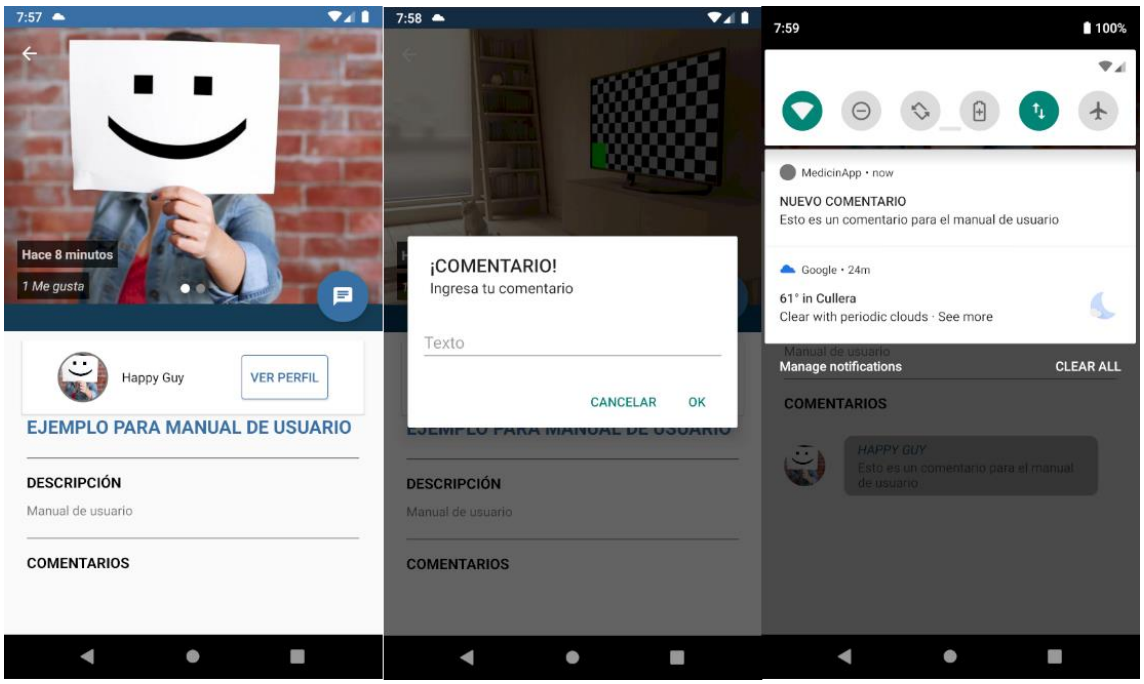


*Figura 56: Manual de usuario, creando comentario*

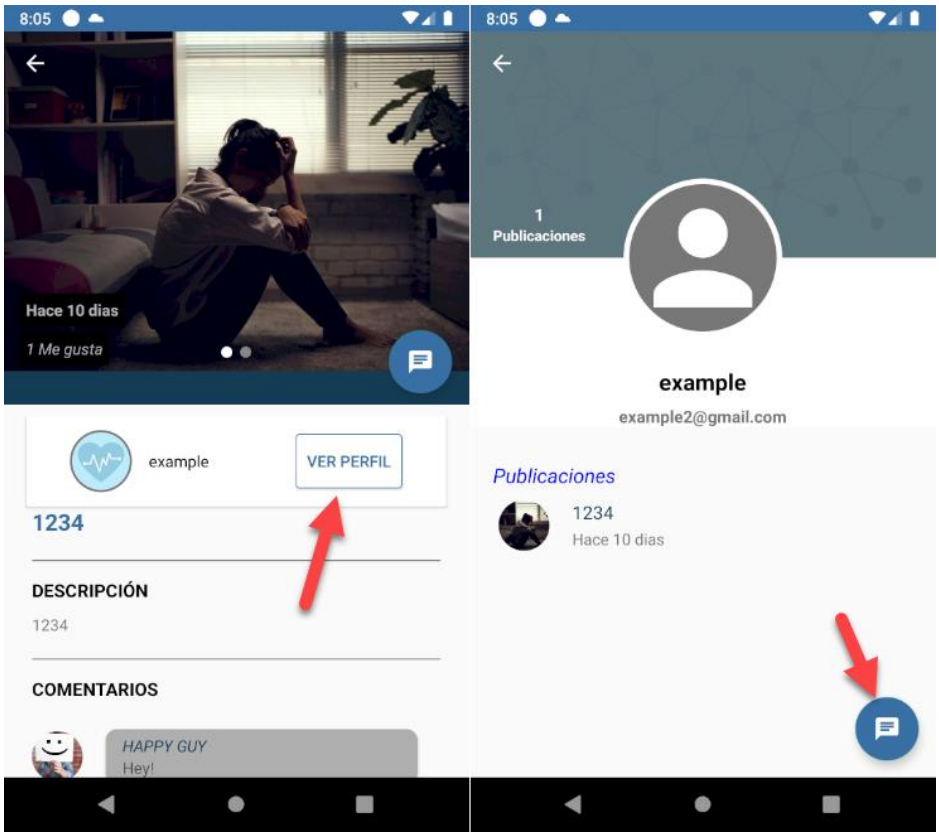


*Figura 57: Manual de usuario, visitando perfil del creador de la publicación*

Ahora al pasar a la pestaña de chats, el usuario podrá ver sus chats, al pulsar sobre ellos podrá ver la conversación. Un detalle importante, es que el usuario puede ver si su compañero está en

línea o si ha leído sus mensajes, esto último se ve cuando el doble check pasas de gris a verde. (Figura 58)

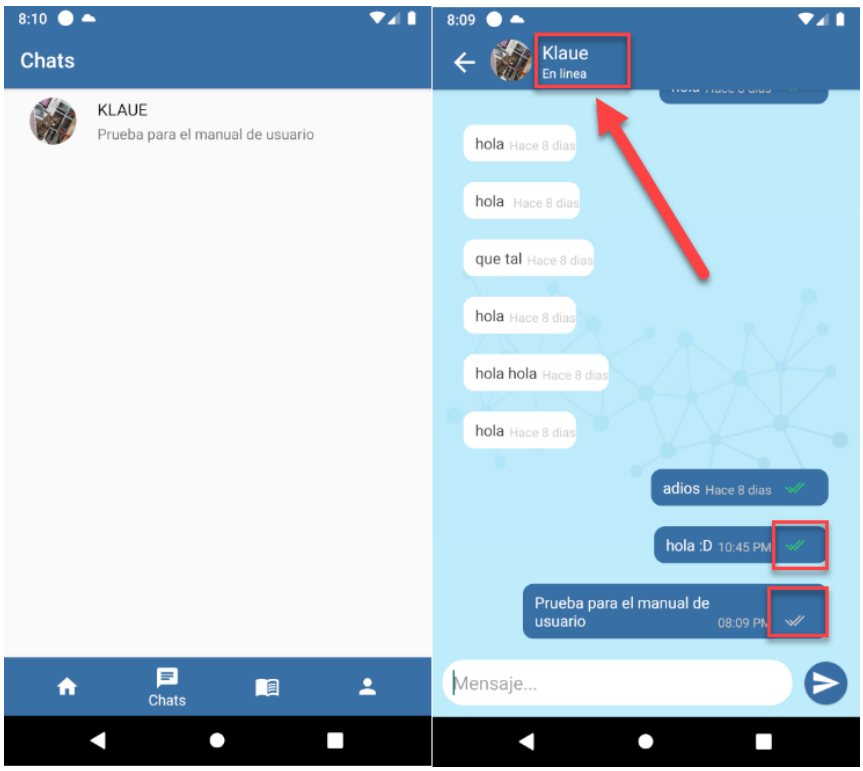


*Figura 58: Manual de usuario chats.*

En bitácora el usuario podrá ver su registro de páginas de bitácora, crear una nueva página de bitácora o ver los detalles. (Figura 59).

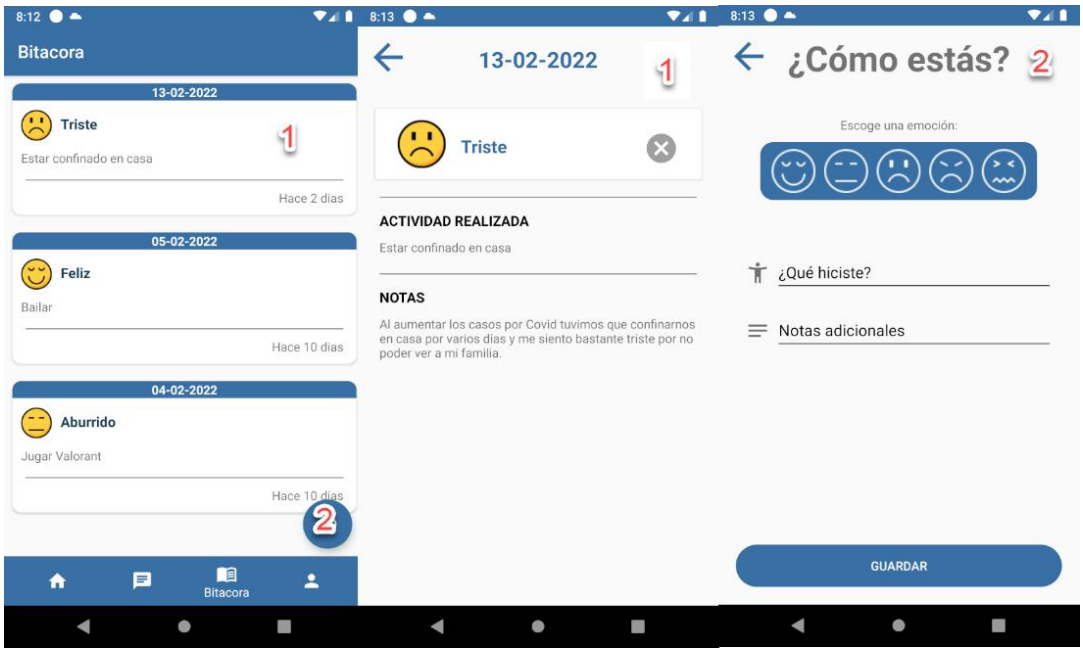


*Figura 59: Manual de usuario, bitácora*

Cuando el usuario se encuentra en los detalles de la bitácora podrá leer las notas y también borrar la página de la bitácora en el caso de querer hacerlo con el botón de la X. (Figura 60).

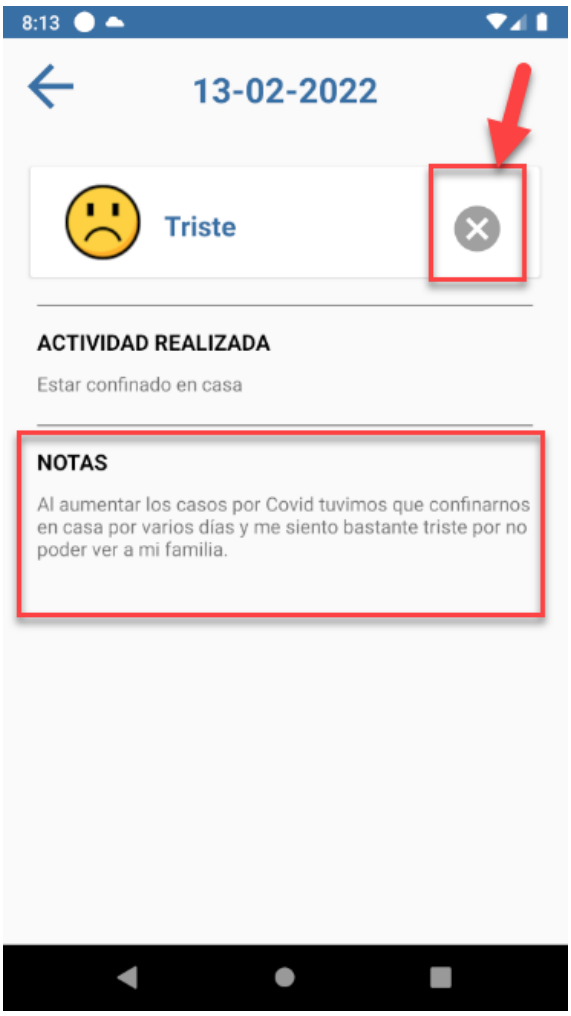


*Figura 60: Manual de usuario, detalles de la bitácora*

Para crear una página de la bitácora el usuario deberá escoger uno de los emoticonos y luego de rellenar los campos. Teniendo 25 caracteres máximos para poner la actividad y 140 para las notas. (Figura 61)

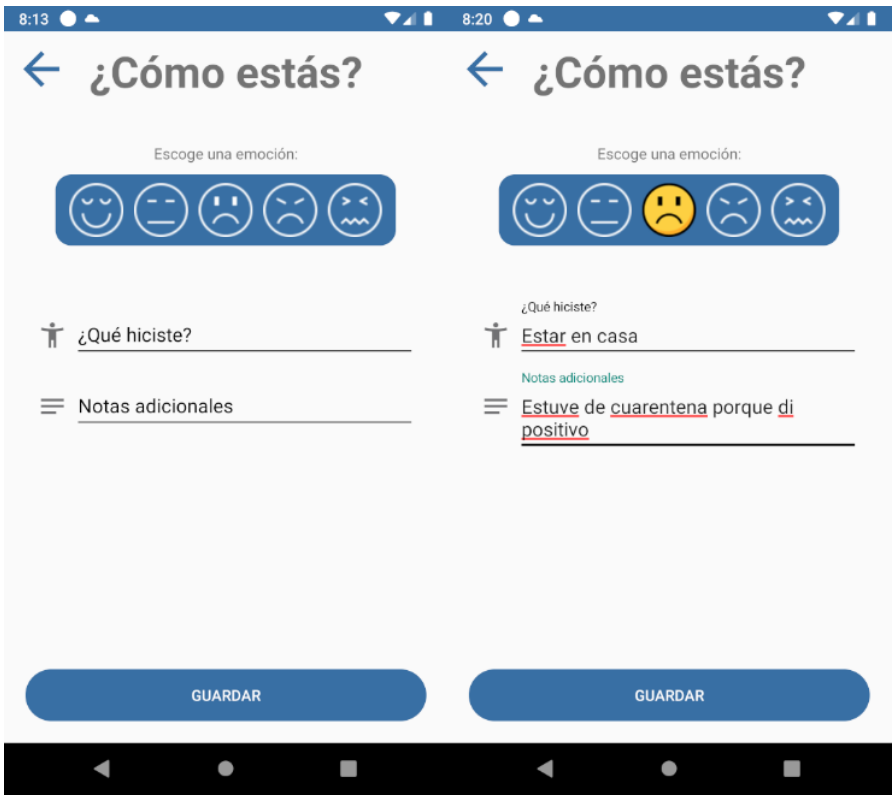


*Figura 61: Manual de usuario, creando página de bitácora*

Finalmente, el usuario podrá visitar su perfil donde podrá ver información como cuántas publicaciones tiene, su nombre de usuario, su email, ver sus publicaciones y poder borrarlas, además, puede editar su perfil desde el botón “Editar perfil”, ahí podrá cambiar su nombre de usuario, su imagen de usuario y su banner de usuario. (Figura 62)

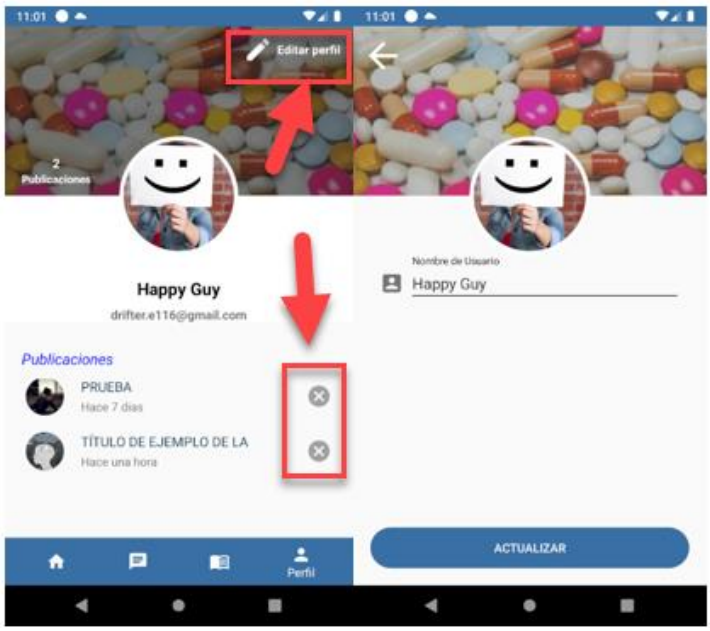


*Figura 62: Manual de usuario, ver/editar perfil*

# 8 Conclusiones

Se han cumplido los objetivos y requisitos que se propusieron en los capítulos. Siendo el principal es crear una aplicación de Android que permitiera chatear entre dos usuarios, subir contenido y tener una bitácora donde guardar datos personales de interés como lo son las emociones que provocaron cierta actividad.

La aplicación que he desarrollado es una base sólida para cualquier tipo de ampliación que tenga que ver con ayudar a un tipo de paciente. En este caso, pese a que fuera un tema relacionado con la depresión, nadie quita que pudiera usarse para pacientes de enfermedades concretas donde apuntar los síntomas que se sienten a lo largo de los días y el poder subir contenido que pueda ir desde ayuda para realizar un tratamiento, hasta información acerca de la enfermedad le sirvan al usuario para sobrellevar mejor su malestar. También, se podría pensar en añadir herramientas con un reloj inteligente que pudiera tomar más información del usuario y usar los datos de la bitácora para generar gráficas y otros datos de interés que permitan un correcto estudio de la enfermedad en cuestión y su evolución.

Por otro lado, he podido repasar algunos fundamentos de programación adquiridos durante la carrera y aprender algunos nuevos, como es el uso de las notificaciones.

Por último, el poder desarrollar un producto de principio a fin por mi cuenta es algo valioso, pues sirve para dar más importancia a lo que es el trabajo en equipo y darme también nociones para poder usar bien mis recursos como lo son el conocimiento y el tiempo.

# 9 Trabajo Futuro

El trabajo que puede realizarse sobre la aplicación puede ser muy variado, desde proponer más herramientas para el seguimiento de una persona con trastornos depresivos hasta poder cambiar la finalidad de esta y usarse para otro tipo de enfermedades.

También se podría empezar a usar bases de datos más potentes que permitan una mayor personalización a la hora de realizar consultar.

Subir más imágenes a las publicaciones y darle más cuerpo a las mismas.

Otro aspecto es el trabajar en la seguridad de la app, como lo sería el añadir filtros para contenido sensible y evitar un mal uso de las publicaciones.

Crear roles para usuarios y especialistas sería algo a contemplar de nuevo, pues, aunque se eliminó esa idea para agilizar el proyecto, a futuro si se introdujese una forma en que el usuario pueda contactar con un especialista confirmado y tener un método de pago para poder realizar sesiones de verdad mediante videollamadas mejoraría bastante la calidad de la aplicación.

Mejorar el chat para poder abrir una conversación sin necesidad de que tengan que usar las publicaciones.

Finalmente resultaría interesante agregar más opciones a la bitácora para poder ser una herramienta de utilidad para un especialista, como el agregar gráficas y otros parámetros que pudiesen servir para tener una historia del paciente.

# 10 Referencias

[1] M. Rahman, «Android Version Distribution statistics will now only be available in Android Studio,» 10 Abril 2020. [En línea]. Available: https://www.xda-developers.com/android-version-distribution-statistics-android-studio/. [Último acceso: 17 Diciembre 2021].

[2] E. Reyes, «Estas apps te ayudan a mejorar tu salud mental,» 8 Octubre 2021. [En línea]. Available: https://expansion.mx/tecnologia/2021/10/08/apps-para-salud-mental-android-apple. [Último acceso: 11 Diciembre 2021].

[3] J. G. Fernández, «El confinamiento dispara la demanda de psicólogos online,» 13 Mayo 2020. [En línea]. Available: https://www.expansion.com/economia-digital/innovacion/2020/05/13/5eb3f24d468aeb246b8b4625.html. [Último acceso: 12 Diciembre 2021].

[4] S. ULLATE, «Salud mental: estas son las mejores apps para cuidarte (desde dentro),» 10 Enero 2022. [En línea]. Available: https://www.harpersbazaar.com/es/cultura/viajes-planes/a38137546/salud-mental-aplicaciones-como-cuidarte-movil/. [Último acceso: 11 Enero 2022].

[5] C. Collado, «Versiones de Android: de la primera a la última versión de Android,» 11 Febrero 2022. [En línea]. Available: https://andro4all.com/guias/android/versiones-android-historia. [Último acceso: 11 Febrero 2022].

[6] J. Lacort, «La salud mental ya es un gran negocio: sus apps disparan la inversión recibida en 2021,» 11 Febrero 2022. [En línea]. Available: https://www.xataka.com/medicina-y-salud/cada-vez-hay-startups-orientadas-a-salud-mental-cada-vez-estan-logrando-inversion. [Último acceso: Febrero 11 2022].

[7] Florent37, «ShapeOfView,» 23 Abril 2021. [En línea]. Available: https://github.com/florent37/ShapeOfView. [Último acceso: 17 Diciembre 2021].

[8] C. Valdovinos, «Mercado de apps de salud mental crecerá un 31% para 2026,» 6 noviembre 2021. [En línea]. Available: https://www.unocero.com/software/apps/salud-mental-apps/. [Último acceso: 14 diciembre 2021].

[9] REDACCIÓN, «La pandemia impulsó la creación de 71.000 apps de salud y fitness en 2020,» 3 Marzo 2021. [En línea]. Available: https://www.cmdsport.com/fitness/actualidad-fitness/la-pandemia-impulso-la-creacion-71-000-apps-salud-fitness-2020/. [Último acceso: 11 Diciembre 2021].

[10] M. M. Roa, «Android e iOS dominan el mercado de los smartphones,» 30 Agosto 2021. [En línea]. Available: https://es.statista.com/grafico/18920/cuota-de-mercado-mundial-de-smartphones-por-sistema-operativo/. [Último acceso: 11 Diciembre 2021].

[11] hdodenhof, «CircleImageView,» 30 Diciembre 2020. [En línea]. Available: https://github.com/hdodenhof/CircleImageView. [Último acceso: 14 Noviembre 2021].

[12] M. M. Roa, «¿En qué países es más popular Android y en cuáles Apple?,» 01 Septiembre 2020. [En línea]. Available: https://es.statista.com/grafico/22758/cuota-de-mercado-de-los-sistemas-operativos-para-moviles/. [Último acceso: 11 Diciembre 2021].

[13] CORREDURÍA INTELIGENTE, «Redes Sociales : definición y características,» 6 Mayo 2019. [En línea]. Available: https://www.mpmsoftware.com/es/blog/redes-sociales-definicion-y-caracteristicas/. [Último acceso: 12 Diciembre 2021].

[14] F. Charte, «Paquetes en Java: qué son, para qué se utilizan, y cómo se usan,» 14 Agosto 2019. [En línea]. Available: https://www.campusmvp.es/recursos/post/paquetes-en-java-que-son-para-que-se-utilizan-y-como-se-usan.aspx. [Último acceso: 11 Diciembre 2021].

[15] dybarsky, «spots-dialog,» 11 Junio 2018. [En línea]. Available: https://github.com/dybarsky/spots-dialog. [Último acceso: 17 Noviembre 2021].

[16] square, «Retrofit,» 2013. [En línea]. Available: https://square.github.io/retrofit/. [Último acceso: 17 Enero 2022].

[17] Wikipedia, «Modelo de prototipos,» [En línea]. Available: https://es.wikipedia.org/wiki/Modelo_de_prototipos. [Último acceso: 18 Diciembre 2021].

[18] Wikipedia, «Java (Lenguaje de programación),» [En línea]. Available: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programación). [Último acceso: 12 Diciembre 2021].

[19] Android Developers, «Documentación Android,» [En línea]. Available: https://developer.android.com/docs. [Último acceso: 12 Diciembre 2021].

[20] Wikipedia, «Android Lollipop,» [En línea]. Available: https://en.wikipedia.org/wiki/Android_Lollipop. [Último acceso: 11 Diciembre 2021].

[21] Wikipedia, «Android,» [En línea]. Available: https://es.wikipedia.org/wiki/Android. [Último acceso: 11 Diciembre 2021].

[22] Google Firebase, [En línea]. Available: https://firebase.google.com/docs?authuser=0. [Último acceso: 19 Octubre 2021].