UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una aplicación de mensajería móvil: Socialme

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Beliaev, Daniil

Tutor/a: Andrés Martínez, David de

CURSO ACADÉMICO: 2022/2023

Resumen

Este Trabajo Final de Grado consiste en diseñar e implementar una aplicación de mensajería móvil para el sistema operativo Android y un *backend* (implementado con Firebase) que se encargará de gestionar las peticiones que reciba de la misma. La aplicación, además de disponer de las características básicas que toda aplicación de mensajería ha de tener (envío y recepción de mensajes de texto, audios, fotos y vídeos), ofrecerá unas nuevas que no se hallan en ninguna otra con el objetivo en mente de facilitar la sociabilidad entre los usuarios.

La aplicación móvil será desarrollada en el entorno de desarrollo Android Studio con el lenguaje de programación Kotlin. En cuanto al *backend*, se utilizarán los diversos servicios de Firebase proporcionados por Google.

Palabras clave: aplicación móvil, Android, mensajería, sociabilidad, Android Studio, Kotlin, Backend, Firebase

Abstract

This Final Degree Project involves designing and implementing a mobile messaging application for the Android operating system and a backend (implemented using Firebase) responsible for handling requests received from it. In addition to featuring the basic functionalities expected from any messaging application (sending and receiving text messages, audio, photos, and videos), this application will introduce new features not found in any other messaging app, aimed at enhancing user sociability.

The mobile application will be developed within the Android Studio development environment using the Kotlin programming language. Regarding the backend, various Firebase services provided by Google will be utilized.

Key words: mobile application, Android, messaging, sociability, Android Studio, Kotlin, Backend, Firebase

Índice general

Índice general v
Índice de figuras vii
Índice de tablas ix

1 Introducción 1
1.1 Motivación . . . 1
1.2 Objetivos . . . 2
1.3 Descripción de los capítulos . . . 3
2 Estado del arte 5
2.1 WhatsApp . . . 5
2.2 Telegram . . . 6
2.3 Signal . . . 8
2.4 Facebook Messenger . . . 9
2.5 Tabla comparativa . . . 10
3 Diseño 13
3.1 Casos de uso . . . 14
3.1.1 Usuario no autenticado . . . 14
3.2 Mockups de la interfaz gráfica . . . 18
3.3 Diagrama de base de datos . . . 27
4 Implementación 31
4.1 Tecnologías empleadas . . . 31
4.1.1 Tecnologías utilizadas en el *frontend* . . . 31
4.1.2 Tecnologías utilizadas en el *backend* . . . 33
4.2 Componentes de una aplicación en Android . . . 34
4.3 Arquitectura para acceso a datos . . . 34
4.4 Estructura del proyecto . . . 35
4.5 Bibliotecas empleadas . . . 36
4.5.1 uCrop . . . 36
4.5.2 CircleImageView . . . 37
4.5.3 Glide . . . 37
4.5.4 Moshi . . . 37
4.5.5 Hilt . . . 37
4.5.6 Retrofit . . . 37
4.5.7 Room . . . 37
4.5.8 Jsoup . . . 38
4.6 Construcción de la base de datos . . . 38
4.7 Configuración de los productos de Firebase . . . 39
4.7.1 Authentication . . . 39
4.7.2 Firestore Database . . . 40
4.7.3 Storage . . . 40
4.7.4 Messaging . . . 41
4.7.5 Functions . . . 41

4.8 Implementación de las funciones 42
4.8.1 Iniciar sesión (LoginActivity) 42
4.8.2 Crear nueva cuenta (CreateAccountActivity) 43
4.8.3 Cambiar contraseña (ChangePasswordGetEmailActivity y ChangePasswordActivity) 44
4.8.4 Cerrar sesión (en MainActivity) 45
4.8.5 Revisar SMS con phishing (SMSFragment) 45
4.8.6 Crear evento (NewEventActivity) 47
5 Resultados **53**
6 Conclusiones **63**
Bibliografía **65**
Anexos **67**
6.1 Descripción detallada de los casos de uso 67
6.1.1 Usuario no autenticado 67
6.1.2 Usuario autenticado 69
6.2 Descripción a detalle de los diagramas de bases de datos 77
6.2.1 Tabla ″Users″ 77
6.2.2 Tabla ″Groups″ 78
6.2.3 Tabla ″GroupsUsers″ 78
6.2.4 Tabla ″Messages″ 78
6.2.5 Tabla ″Events″ 82
6.2.6 Tabla ″EventsUsers″ 82
6.2.7 Tabla ″Congratulations″ 82
6.2.8 Tabla ″SMS″ 82
6.2.9 Tabla ″Users″ del *backend* 83
6.2.10 Tabla ″Groups″ del *backend* 83
6.2.11 Tabla ″Events″ del *backend* 83
6.3 Objetivos de desarrollo sostenible 84

Índice de figuras

2.1 Icono de WhatsApp . . . 5
2.2 Comunidades de WhatsApp . . . 6
2.3 Estados de WhatsApp . . . 6
2.4 Icono de Telegram . . . 7
2.5 Bloqueo al intentar hacer captura de pantalla . . . 7
2.6 Creación de chat secreto con usuario . . . 8
2.7 Icono de Signal . . . 8
2.8 Opción de autodestrucción de los mensajes . . . 9
2.9 Programación de un mensaje . . . 9
2.10 Icono de Facebook Messenger . . . 9
2.11 Opción de traducción de los mensajes . . . 10
2.12 Sugerencia de respuesta . . . 10

3.1 Arquitectura de un producto software . . . 13
3.2 Diagrama de casos de uso - Usuario no autenticado . . . 14
3.3 Diagrama de casos de uso - Usuario autenticado - Chats . . . 15
3.4 Diagrama de casos de uso - Usuario autenticado - Creación de chats . . . 16
3.5 Diagrama de casos de uso - Usuario autenticado - Eventos . . . 17
3.6 Diagrama de casos de uso - Usuario autenticado - Otras funcionalidades . . . 18
3.7 Diseño de la interfaz - Iniciar sesión . . . 19
3.8 Diseño de la interfaz - Crear nueva cuenta . . . 19
3.9 Diseño de la interfaz - Cambiar contraseña . . . 19
3.10 Diseño de la interfaz - Ver lista de últimos chats y Cerrar sesión . . . 20
3.11 Diseño de la interfaz - Ver un chat, Enviar mensaje, Enviar archivo, Traducir mensaje y Programar envío . . . 21
3.12 Diseño de la interfaz - Ver más información sobre el chat y Eliminar participante . . . 22
3.13 Diseño de la interfaz - Crear nuevo chat, Listar usuarios favoritos y Eliminar usuarios de favoritos . . . 22
3.14 Diseño de la interfaz - Buscar usuario y Guardar usuario como favorito . . . 23
3.15 Diseño de la interfaz - Crear nuevo grupo . . . 23
3.16 Diseño de la interfaz - Ver eventos y felicitaciones . . . 24
3.17 Diseño de la interfaz - Crear evento . . . 24
3.18 Diseño de la interfaz - Crear felicitación . . . 25
3.19 Diseño de la interfaz - Ver evento y Eliminar evento . . . 25
3.20 Diseño de la interfaz - Ver felicitación y Eliminar felicitación . . . 26
3.21 Diseño de la interfaz - Revisar SMS con phishing y Ver más detalles de un SMS con phishing . . . 26
3.22 Diseño de la interfaz - Modificar ajustes de la cuenta . . . 27
3.23 Diagrama de base de datos - Aplicación móvil . . . 28
3.24 Diagrama de base de datos - Servidor backend Firebase . . . 29

4.1 Interfaz principal de Android Studio . . . 32
4.2 Diferencias entre Java y Kotlin . . . 32

4.3 Comunicación entre los productos de Firebase y la aplicación 33
4.4 Arquitectura de la aplicación para acceso a datos 34
4.5 Correo electrónico/contraseña establecido como proveedor 39
4.6 Plantilla del correo para cambiar contraseña modificada 40
4.7 Colecciones creadas a partir del diagrama de base de datos 40
4.8 Carpetas creadas en Storage 41
4.9 Activación del servicio Messaging en la consola de Google Cloud 41
4.10 Primera situación al iniciar la aplicación 43
4.11 Segunda situación al iniciar la aplicación 43
4.12 Web predefinida para cambiar la contraseña de la cuenta 44
4.13 Actividad de la aplicación para cambiar la contraseña de la cuenta 45
4.14 Código del diálogo para cerrar sesión 45
4.15 Código del análisis de los SMS 46
4.16 Código para comprobar si un texto incluye un enlace 46
4.17 Código para comprobar si el número de teléfono del remitente es peligroso 47
4.18 Código de la aplicación para crear el evento 48
4.19 Primera parte del código de Cloud Functions para crear el evento 49
4.20 Segunda parte del código de Cloud Functions para crear el evento 50
4.21 Código del servicio para crear el evento 51
4.22 Código de Cloud Functions para eliminar un documento de una colección 51

5.1 Alice inicia la aplicación 54
5.2 Pantalla de crear cuenta con los datos de Alice 54
5.3 Email para verificar el correo electrónico 54
5.4 Alice selecciona abrir el enlace con Socialme 54
5.5 La aplicación informa a Alice del éxito de la verificación del correo 55
5.6 Pantalla principal de Socialme 56
5.7 SMS analizados y detectados como peligrosos 56
5.8 Pantalla de nuevo chat de la aplicación 56
5.9 Alice busca a Bob 56
5.10 Alice añade a Bob a favoritos 57
5.11 Alice le envía un mensaje a Bob 58
5.12 Bob responde a Alice 58
5.13 Alice selecciona la opción de ver más información sobre Bob 58
5.14 Alice selecciona un idioma y escribe un mensaje para que se traduzca . . . 58
5.15 Se traduce el mensaje al idioma seleccionado 59
5.16 Alice accede al apartado "Eventos" 60
5.17 La aplicación insta a Alice a seleccionar usuarios 60
5.18 Alice completa los datos relativos al evento 60
5.19 Aparece listado el evento recién creado 60
5.20 Acceso a los datos del evento recién creado 61
5.21 La aplicación ofrece varias opciones a Alice 62
5.22 Alice solicita cerrar sesión 62

Índice de tablas

2.1 Tabla comparativa de características entre aplicaciones 11

6.1 Descripción del caso de uso - Iniciar sesión 67
6.2 Descripción del caso de uso - Crear nueva cuenta 68
6.3 Descripción del caso de uso - Cambiar contraseña 68
6.4 Descripción del caso de uso - Ver lista de últimos chats 69
6.5 Descripción del caso de uso - Ver un chat 69
6.6 Descripción del caso de uso - Enviar mensaje 69
6.7 Descripción del caso de uso - Enviar archivo 70
6.8 Descripción del caso de uso - Traducir mensaje 70
6.9 Descripción del caso de uso - Programar envío 71
6.10 Descripción del caso de uso - Ver más información sobre el chat 71
6.11 Descripción del caso de uso - Eliminar participante 71
6.12 Descripción del caso de uso - Crear nuevo chat 72
6.13 Descripción del caso de uso - Buscar usuario 72
6.14 Descripción del caso de uso - Guardar usuario como favorito 72
6.15 Descripción del caso de uso - Crear nuevo grupo 73
6.16 Descripción del caso de uso - Listar usuarios favoritos 73
6.17 Descripción del caso de uso - Eliminar usuario de favoritos 73
6.18 Descripción del caso de uso - Ver eventos y felicitaciones 74
6.19 Descripción del caso de uso - Crear evento 74
6.20 Descripción del caso de uso - Crear felicitación 74
6.21 Descripción del caso de uso - Eliminar evento 75
6.22 Descripción del caso de uso - Eliminar felicitación 75
6.23 Descripción del caso de uso - Ver evento 75
6.24 Descripción del caso de uso - Ver felicitación 76
6.25 Descripción del caso de uso - Cerrar sesión 76
6.26 Descripción del caso de uso - Revisar SMS con phishing 76
6.27 Descripción del caso de uso - Ver más detalles de un SMS con phishing . . 77
6.28 Descripción del caso de uso - Modificar ajustes de la cuenta 77

CAPÍTULO 1
Introducción

En las últimas décadas, Internet ha cambiado por completo la forma en que las personas se comunican y conectan en todo el mundo. Desde su aparición como red de comunicación militar a finales de la década de 1960[1] hasta su rápida expansión en la vida cotidiana de las personas, Internet ha revolucionado la forma en que intercambiamos información, compartimos experiencias y mantenemos contacto. Uno de los aspectos más destacados de esta evolución fue el surgimiento de las aplicaciones de mensajería móvil, que redefinieron la comunicación interpersonal al permitir el chat instantáneo a través de dispositivos móviles en cualquier lugar y en todo momento.

Las aplicaciones de mensajería móvil han desempeñado un papel importante en la remodelación de la forma en que las personas se comunican[2]. A medida que los teléfonos móviles se volvieron omnipresentes en la sociedad, estas aplicaciones aprovecharon la conectividad constante que proporcionaban. Desde los primeros servicios de mensajes cortos (SMS, del inglés *Short Message Service*) hasta las plataformas de mensajería más complejas y ricas en funciones como WhatsApp[1], Facebook Messenger[2], Telegram[3] y Signal[4], estas aplicaciones han simplificado y enriquecido la interacción humana a través de la innovación tecnológica.

La evolución de las aplicaciones de mensajería móvil incluye una amplia gama de funciones, desde simples chats de texto hasta la integración de llamadas de voz y vídeo, intercambio de archivos, *stickers* y emoticonos, personalización y creación de grupos y canales para facilitar la comunicación en comunidades más grandes. Estas aplicaciones trascienden las barreras geográficas y del idioma, permitiendo que amigos, familiares y colegas permanezcan conectados incluso cuando están lejos.

Además de su impacto en la comunicación personal, las aplicaciones de mensajería móvil tienen un profundo impacto en las operaciones comerciales y el mundo empresarial. Las empresas han aprovechado estas plataformas como canales para brindar atención al cliente, distribuir información sobre productos y promociones, e incluso facilitar transacciones financieras y comerciales.

1.1 Motivación

La principal razón por la que he decidido crear una aplicación de mensajería es el de cumplir una meta personal ya que desde que estudiaba en 2º de Bachillerato he desea-

[1] https://www.whatsapp.com
[2] https://www.messenger.com/
[3] https://telegram.org/
[4] https://signal.org/es/

do hacerlo y no ha sido sino hasta 4º de carrera que he tenido la posibilidad de cursar una asignatura enfocada en la creación de aplicaciones para dispositivos móviles Android que, junto a lo que he aprendido en la rama de las Tecnologías de la Información, dispongo de los conocimientos necesarios para implementarla.

Otro motivo es, como ya he mencionado anteriormente, poner en práctica lo que he aprendido a lo largo de la carrera: principios de diseño de un producto software, creación y manipulación de bases de datos, integración de programas software y programación de aplicaciones Android.

Otra razón es la inclusión de nuevas características que no se encuentran en las aplicaciones actuales de mensajería.

El último motivo es poder usar la solución final para poder comunicarme con mi familia y amigos en lugar de emplear aplicaciones de terceros de las que no dispongo control y no sé cómo son usados mis datos. Sin embargo, no se busca hacer pública esta aplicación por diversos motivos:

- Dificultad para convencer a las personas de cambiarse a otra aplicación de mensajería instantánea principalmente por el hecho de que todos sus contactos están cómodos en la actual que utilizan.
- No encontrar forma de monetizar la aplicación sin vender datos de usuarios. No se contempla poner anuncios ya que ninguna aplicación en la competencia lo hace.

1.2 Objetivos

El objetivo de este proyecto es diseñar y desarrollar una aplicación de mensajería móvil que, además de incluir las características básicas que proporciona cualquier tipo de aplicación de esta índole, ofrezca otras funcionalidades que no se encuentran disponibles en la competencia.

Para poder cumplir el anterior objetivo será necesario satisfacer las siguientes metas:

- Diseñar e implementar una aplicación Android que se comunique con un servidor *backend* enviando mensajes y procesando los recibidos de éste.
- Diseñar e implementar un servidor *backend* que sirva como intermediario entre los diferentes usuarios que utilicen la aplicación y que gestione las siguientes funciones:
  - Sistema de autentificación y gestión de las sesiones de los usuarios de la aplicación.
  - Base de datos para almacenar los mensajes que han de ser remitidos a los usuarios.
  - Sistema de almacenamiento para guardar los ficheros enviados por los usuarios.
  - Sistema de envío de mensajes a los usuarios.
  - Sistema que procese los mensajes recibidos de la aplicación e interaccione con la base de datos para almacenarlos y el sistema de envío para remitirlos.

Para poder llevar a cabo este proyecto se propone utilizar el entorno de desarrollo Android Studio y la plataforma Google Firebase.

1.3 Descripción de los capítulos

- **Capítulo 1:** En este capítulo se dará una breve introducción al contexto actual en el que se hallan las aplicaciones de mensajería móvil. Además, se explicará la motivación tras la cual se ha decidido realizar este proyecto y los objetivos que se persiguen completar a lo largo del mismo.
- **Capítulo 2:** En este capítulo se analizarán las soluciones actuales ofrecidas por la competencia para establecer un marco común desde el cual empezar a diseñar la aplicación. Una vez realizado el análisis se establecerán las características de las que dispondrá la solución final.
- **Capítulo 3:** En este capítulo se utilizará el análisis anterior para realizar un diseño de las tres capas que tiene cualquier producto software.
- **Capítulo 4:** En este capítulo se explicarán las herramientas y bibliotecas que han sido empleadas a la hora de implementar el producto. Además, se explicará las decisiones tomadas para implementar funciones que se consideran importantes como la comunicación entre la aplicación y el servidor backend.
- **Capítulo 5:** En este capítulo se presentarán las diferentes pantallas y funcionalidades de la solución desarrollada mediante una historia.
- **Capítulo 6:** En este capítulo se darán las conclusiones que se han obtenido al finalizar el trabajo junto a las lecciones aprendidas en éste.

CAPÍTULO 2
Estado del arte

Previamente a la realización del diseño de una aplicación es necesario llevar a cabo un análisis de la competencia por varios motivos:

- Aprendizaje de buenas prácticas: Al estudiar aplicaciones competidoras se pueden observar qué características y funcionalidades son populares entre los usuarios y la forma de diseñar las interfaces gráficas que brindan una óptima experiencia a los mismos.

- Diferenciación: Una vez visto lo que ofrecen las aplicaciones competidoras se pueden idear formas para diferenciar la aplicación que se quiere crear para que destaque. Esto puede incluir la incorporación de nuevas características, incluir características que solamente se hallan en un producto de la competencia o mejorar la interfaz para que sea más intuitiva y brinde una mejor experiencia.

Actualmente existen varias aplicaciones de mensajería móvil disponibles en el mercado que ofrecen diferentes funcionalidades y características. A continuación, se describirán algunas de las más populares y sus principales características:

2.1 WhatsApp

Figura 2.1: Icono de WhatsApp

WhatsApp[1] es una de las aplicaciones de mensajería más populares del mundo, con más de 2 mil millones de usuarios activos mensuales en todo el mundo. Permite a los usuarios enviar mensajes de texto, mensajes de voz, hacer llamadas de voz y video, compartir imágenes, videos y documentos, crear grupos de chat y enviar mensajes de difusión a múltiples destinatarios.

Entre las características más populares de WhatsApp se incluyen la capacidad de crear grupos de chat con hasta 1024 participantes, la función de llamadas de voz y video de alta calidad, la opción de compartir la ubicación en tiempo real, la opción de eliminar mensajes enviados, la posibilidad de agrupar grupos en comunidades (figura 2.2), la pestaña estados en la que los contactos suben actualizaciones sobre sus vidas (figura 2.3), y el cifrado de extremo a extremo para garantizar la privacidad y seguridad de los mensajes.

[1] https://www.whatsapp.com

Además, WhatsApp también ofrece funciones de negocios, como la posibilidad de crear un perfil de empresa, enviar mensajes automatizados y proporcionar atención al cliente a través de la aplicación.

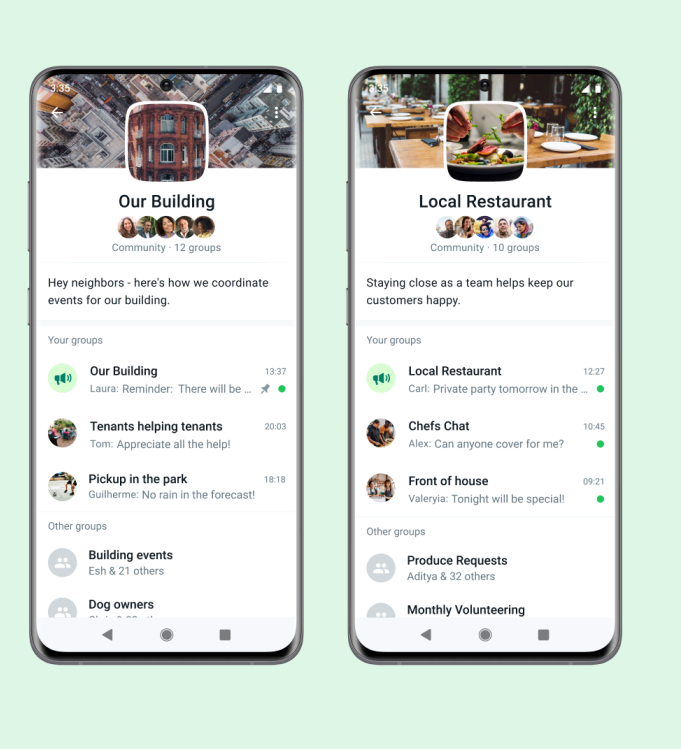

Figura 2.2: Comunidades de WhatsApp

Figura 2.3: Estados de WhatsApp

2.2 Telegram

Figura 2.4: Icono de Telegram

Telegram[2] es otra aplicación de mensajería popular que se centra en la seguridad y privacidad de los usuarios. Al igual que WhatsApp, permite a los usuarios enviar mensajes de texto, mensajes de voz, hacer llamadas de voz y video, compartir imágenes, videos y documentos, crear grupos de chat y enviar mensajes de difusión a múltiples destinatarios.

Entre las características más destacadas de Telegram se incluyen la capacidad de crear grupos de chat con hasta 200.000 participantes, la opción de programar mensajes para que se envíen en una fecha y hora específica, la función de autodestrucción de mensajes, bloqueo de las capturas de pantalla (figura 2.5), y la opción de crear canales públicos para difundir información a un gran número de personas.

Además, Telegram también ofrece funciones de seguridad avanzadas, como el cifrado de extremo a extremo opcional (en los chats secretos, figura 2.6), la autenticación de dos factores y la opción de ocultar el número de teléfono de los contactos.

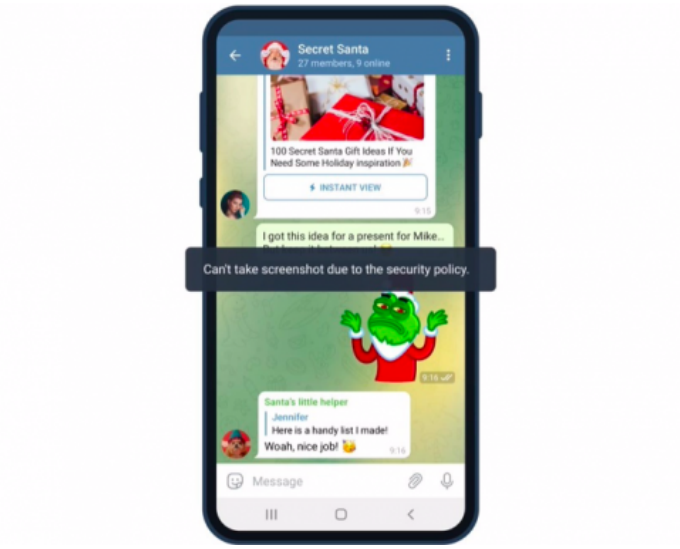

Figura 2.5: Bloqueo al intentar hacer captura de pantalla

[2]https://telegram.org/

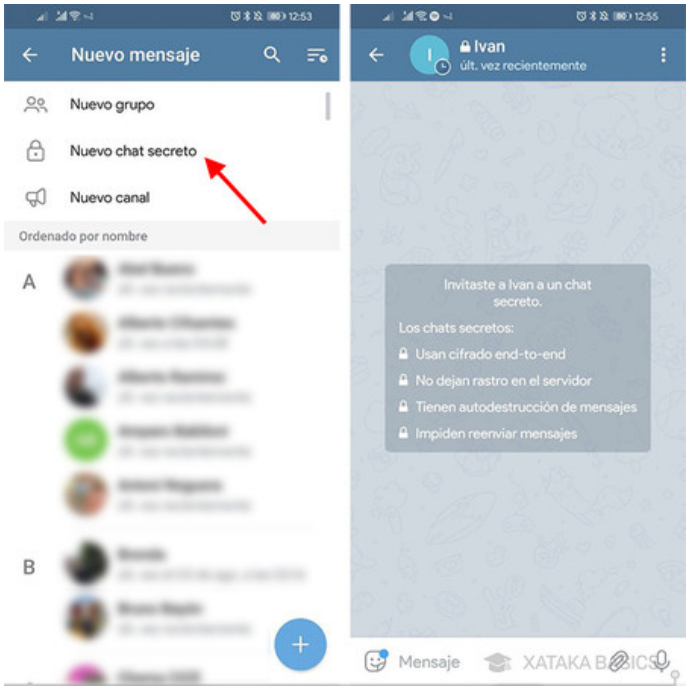

Figura 2.6: Creación de chat secreto con usuario

2.3 Signal

Figura 2.7: Icono de Signal

Signal[3] es una aplicación de mensajería centrada en la privacidad que se ha vuelto cada vez más popular en los últimos años. Al igual que WhatsApp y Telegram, permite a los usuarios enviar mensajes de texto, mensajes de voz, hacer llamadas de voz y video, compartir imágenes, videos y documentos, crear grupos de chat y enviar mensajes de difusión a múltiples destinatarios.

Entre las características más destacadas de Signal se incluyen el cifrado de extremo a extremo para todos los mensajes, la opción de programar mensajes para que se envíen en una fecha y hora específicas (figura 2.9), la función de autodestrucción de mensajes (figura 2.8), y la opción de crear grupos de chat con hasta 1000 participantes.

Además, Signal también ofrece funciones de seguridad avanzadas, como la autenticación de dos factores y la opción de bloquear capturas de pantalla en ciertas conversaciones.

[3]https://signal.org/es/

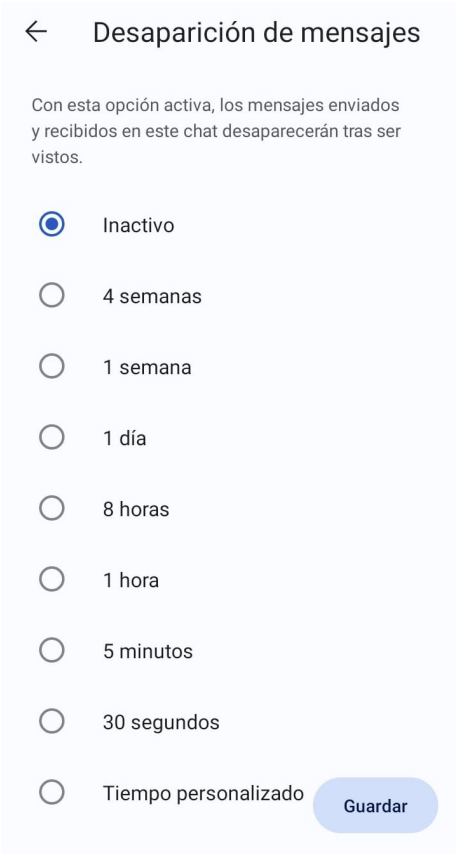

Figura 2.8: Opción de autodestrucción de los mensajes

Figura 2.9: Programación de un mensaje

2.4 Facebook Messenger

Figura 2.10: Icono de Facebook Messenger

Facebook Messenger[4] es la aplicación de mensajería móvil asociada con la red social Facebook, con más de 1.300 millones de usuarios activos mensuales. La aplicación permite a los usuarios enviar mensajes de texto, hacer llamadas de voz y video, compartir archivos, así como enviar mensajes de voz. Además, Messenger también cuenta con una amplia gama de funcionalidades adicionales, como juegos y pagos.

Messenger también permite a los usuarios crear grupos de chat y enviar mensajes a múltiples destinatarios. Además, la aplicación también ofrece funciones de inteligencia artificial, como sugerencias de respuestas (figura 2.12) y traducciones de idiomas en tiempo real (figura 2.11).

[4]https://www.messenger.com/

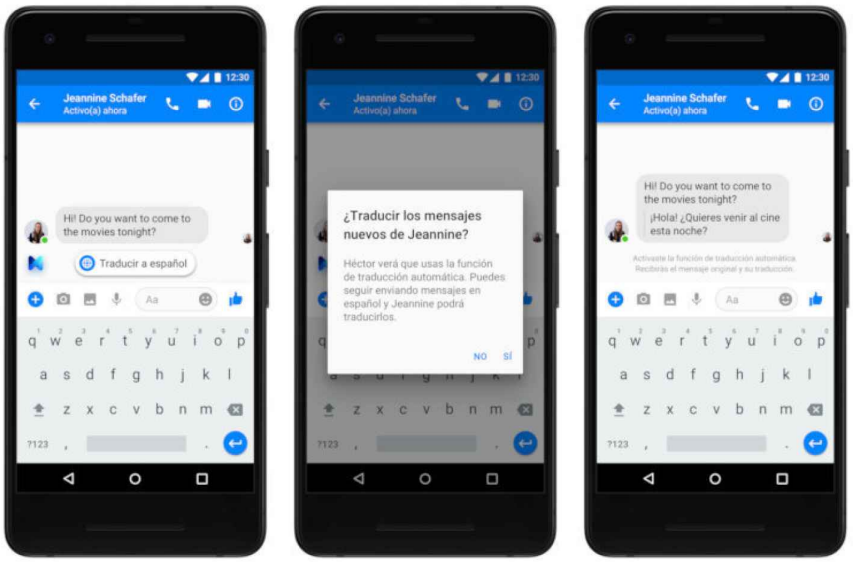

Figura 2.11: Opción de traducción de los mensajes

Figura 2.12: Sugerencia de respuesta

2.5 Tabla comparativa

A continuación, se hará una tabla comparativa con las cuatro aplicaciones anteriormente mencionadas para así poder identificar las similitudes y diferencias que radican entre ellas.

Tabla 2.1: Tabla comparativa de características entre aplicaciones

Funcionalidad	WhatsApp	Telegram	Signal	Facebook Messenger
Enviar mensajes de texto	✓	✓	✓	✓
Enviar mensajes de voz	✓	✓	✓	✓
Enviar archivos: documentos, fotos, vídeos y audios	✓	✓	✓	✓
Programar el envío de los mensajes	✗	✓	✓	✗
Estados de usuarios	✓	✗	✓	✓
Crear grupos de chats	✓	✓	✓	✓
Agrupación de grupos en comunidades	✓	✗	✗	✗
Realizar llamadas de audio y vídeo	✓	✓	✓	✓
Cuenta creada a partir del número de teléfono	✓	✓	✓	✗

De la tabla anterior se puede extraer la siguiente información:

- Las 4 aplicaciones permiten el envío de mensajes de texto, de voz y de archivos, la creación de grupos de chats y la realización de llamadas de audio y vídeo.
- Solo Telegram y Signal permiten programar el envío de los mensajes.
- Salvo en Telegram, en el resto de las aplicaciones se ofrece la pestaña de estados de usuario.
- Solamente en WhatsApp se ofrece la posibilidad de crear comunidades agrupando grupos.
- En WhatsApp, Telegram y Signal la cuenta está enlazada al número de teléfono.

Una vez analizada la competencia, se van a describir las características que se van a añadir a la aplicación que se va a crear:

- Enviar mensajes de texto.
- Enviar archivos: documentos, fotos, vídeos y audios.
- Programar el envío de los mensajes para una fecha y hora concretas.

- Crear grupos de chats.
- Crear cuenta mediante correo electrónico.

Como se puede observar, se han decidido obviar algunas funcionalidades que se hallan en la competencia. Esto es debido a dos motivos: añadirían una mayor complejidad al desarrollo de la solución restándole tiempo a la adición de nuevas características (enviar mensajes de voz y realizar llamadas de audio y vídeo) o que son características poco empleadas por los usuarios (estados de usuario y creación de comunidades).

Por otro lado, se añadirán características nuevas que no se encuentran en la competencia para que la solución tenga un valor añadido:

- Opción de traducir el mensaje que se está redactando a diversos idiomas.
- Detección y aviso de posible phishing mediante la inspección de la bandeja de entrada de los SMS.
- Creación de un apartado llamado ”Eventos” en el que se guardarán la fecha y la hora (y opcionalmente el lugar) de las reuniones acordadas por el usuario.
- Adición de una pestaña llamada ”Felicitaciones” que permitirá al usuario programar mensajes de felicitación para un contacto en específico.

CAPÍTULO 3
Diseño

Tras llevar a cabo el análisis de los productos de la competencia, se ha obtenido una valiosa perspectiva sobre las características y funcionalidades que ofrecen las aplicaciones de mensajería. En este capítulo se va a realizar un proceso de especificación para las tres capas que conforman un producto software (figura 3.1) aprovechando las lecciones aprendidas de las aplicaciones en auge actuales. Cabe mencionar que se ha seleccionado esta arquitectura debido a las numerosas ventajas que ofrece:

- Separación de responsabilidades: La función de cada capa está claramente especificada: la capa de presentación se encarga de la interfaz de usuario, la capa de lógica de negocio se ocupa de la funcionalidad principal y la capa de datos administra el almacenamiento y acceso a datos.

- Reusabilidad de código: Facilidad de reutilizar código, lo que ahorra tiempo y recursos.

- Facilita el desarrollo concurrente: Diferentes equipos de desarrollo pueden trabajar en paralelo sin interferir con los demás.

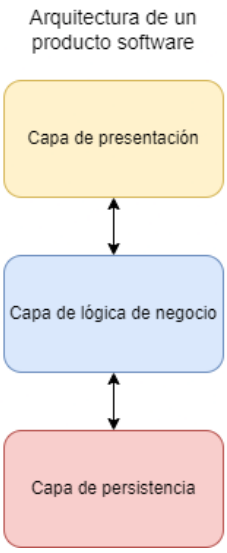

Figura 3.1: Arquitectura de un producto software

3.1 Casos de uso

En lo referente a la lógica de negocio, se va a realizar la especificación de los diferentes casos de uso que serán soportados por la aplicación. Estos casos de uso describen interacciones específicas que un usuario tendrá con la aplicación y cómo la aplicación responderá a esas interacciones. Al comprender en profundidad cómo los usuarios interactuarán con la aplicación, se podrá diseñar una experiencia fluida y eficiente que cumpla con sus expectativas y supere a la competencia.

El diagrama consta de dos actores:

- Usuario no autenticado: Actor que se comunica con el sistema sin credenciales limitando así lo que puede hacer.
- Usuario autenticado: Actor que ha adquirido credenciales y las usa para comunicarse con la mayoría de las funcionalidades del sistema.

A continuación, se presentarán tanto el diagrama de casos de uso dividido para ofrecer una mayor claridad como la descripción de cada caso de uso. Para mayor detalle sobre la descripción de cada caso de uso acudir al Anexo.

3.1.1. Usuario no autenticado

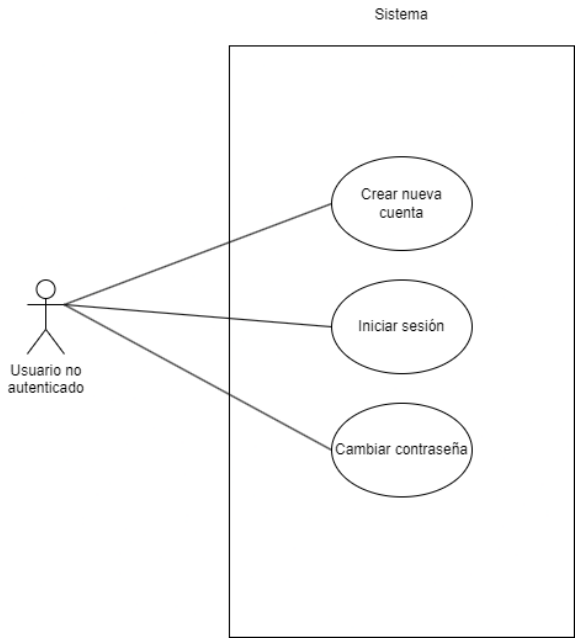

Figura 3.2: Diagrama de casos de uso - Usuario no autenticado

Como se puede apreciar en la figura 3.3, se incluyen los 3 únicos casos de uso que puede llevar a cabo el usuario no autenticado:

- Iniciar sesión (ver Tabla 6.1): El usuario inicia la aplicación e introduce sus credenciales (correo electrónico y contraseña). El sistema comprueba su validez y gestiona la sesión creada. Al acabar, el usuario dispone de una sesión.
- Crear nueva cuenta (ver Tabla 6.2): El usuario crea las credenciales necesarias para poder iniciar sesión. El sistema comprueba si cumplen todos los requisitos y las almacena. Al acabar, el usuario dispone de una cuenta.

- Cambiar contraseña (ver Tabla 6.3): El usuario solicita cambiar la contraseña de una cuenta. Para ello, anteriormente verifica ser poseedor de esta mediante un desafío. El sistema comprueba si cumple todos los requisitos y la almacena. Al acabar, la cuenta del usuario dispone de una nueva contraseña.

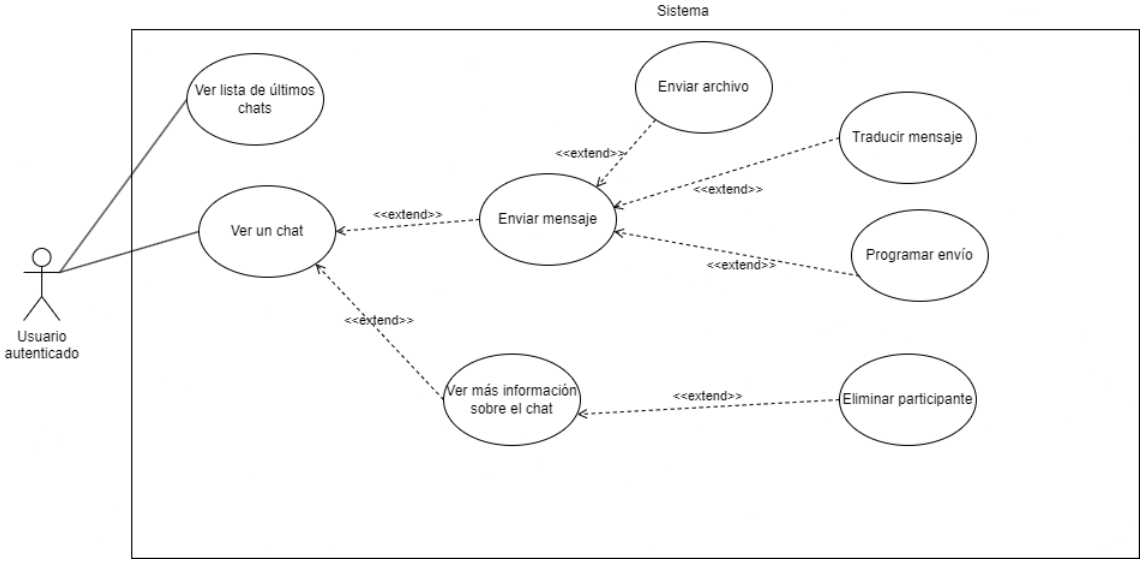

Figura 3.3: Diagrama de casos de uso - Usuario autenticado - Chats

La figura 3.3 representa la parte relacionada con los chats:

- Ver lista de últimos chats (ver Tabla 6.4): El usuario solicita ver las últimas conversaciones que ha tenido. El sistema le muestra una lista con las mismas.

- Ver un chat (ver Tabla 6.5): El usuario solicita ver los mensajes que ha intercambiado con otro usuario o grupo. El sistema le muestra los últimos mensajes enviados por ambas partes.

- Enviar mensaje (ver Tabla 6.6): El usuario redacta un mensaje para otro usuario o grupo. El sistema lo envía al destinatario especificado.

- Enviar archivo (ver Tabla 6.7): El usuario solicita enviar un archivo a otro usuario o grupo. El sistema lo envía al destinatario especificado.

- Traducir mensaje (ver Tabla 6.8): El usuario solicita la traducción del mensaje que ha introducido. El sistema lo traduce.

- Programar envío (ver Tabla 6.9): El usuario solicita programar el envío de un mensaje para una fecha y hora concretas. El sistema lo almacena y lo hará llegar al destinatario en el momento pertinente.

- Ver más información sobre el chat (ver Tabla 6.10): El usuario solicita ver más detalles sobre el usuario o grupo con el que está conversando. El sistema le remite dicha información.

- Eliminar participante (ver Tabla 6.11): El usuario selecciona a un participante para eliminarlo del grupo. El sistema lo elimina y le notifica de su expulsión.

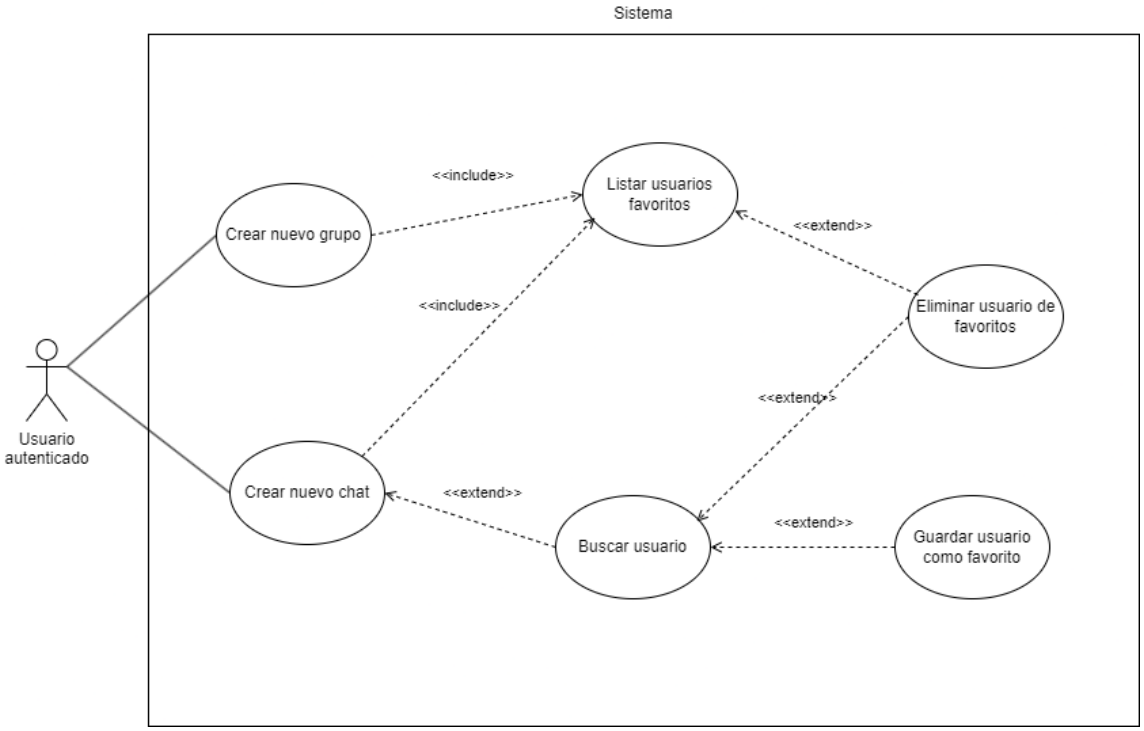

Figura 3.4: Diagrama de casos de uso - Usuario autenticado - Creación de chats

En la figura 3.4 se muestran los casos de uso relacionados con la creación de chats:

- Crear nuevo chat (ver Tabla 6.12): El usuario desea iniciar una nueva conversación. El sistema le muestra todos los contactos que tiene agregados como favoritos. El usuario elije uno de ellos.

- Buscar usuario (ver Tabla 6.13): El usuario desea buscar a otro usuario empleando para ello el nombre de usuario de este último. El sistema le remite las coincidencias.

- Guardar usuario como favorito (ver Tabla 6.14): El usuario solicita guardar a otro usuario como ”favorito”. El sistema lo almacena.

- Crear nuevo grupo (ver Tabla 6.15): El usuario desea crear un grupo. El sistema le muestra todos los contactos que tiene agregados. El usuario elije uno o varios de ellos y le pone un nombre al grupo. El sistema crea el grupo.

- Listar usuarios favoritos (ver Tabla 6.16): El usuario quiere ver los usuarios que tiene guardados como favoritos. El sistema se los devuelve en forma de listado.

- Eliminar usuario de favoritos (ver Tabla 6.17): El usuario solicita eliminar a un usuario de ”favoritos”. El sistema lo elimina.

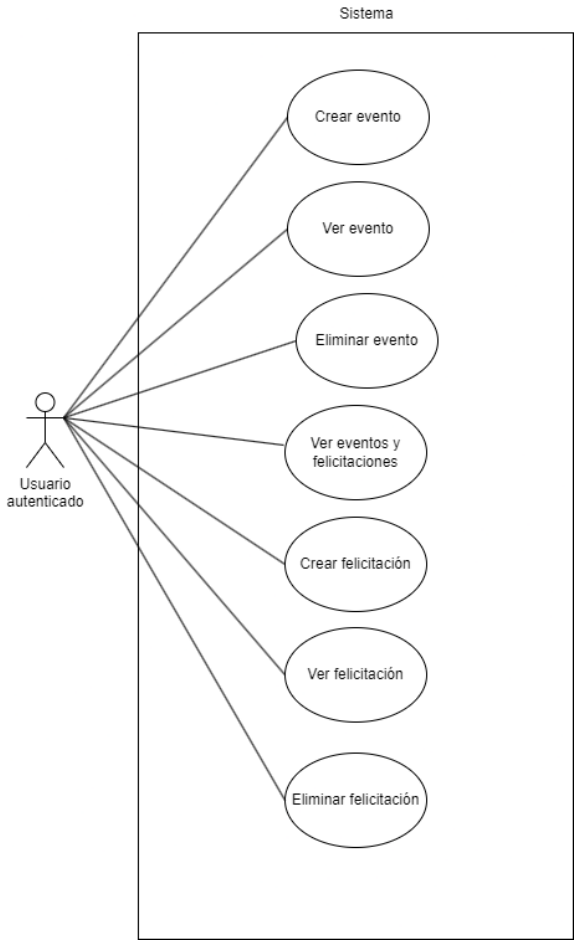

Figura 3.5: Diagrama de casos de uso - Usuario autenticado - Eventos

Como se puede ver en la figura 3.5, se muestran aquellos casos de uso relacionados con los eventos y las felicitaciones:

- Ver eventos y felicitaciones (ver Tabla 6.18): El usuario solicita ver los eventos en los que está incluido y las felicitaciones que ha creado. El sistema se los muestra en forma de listado.
- Crear evento (ver Tabla 6.19): El usuario introduce los datos relativos al evento. El sistema lo almacena y envía a todos los usuarios partícipes.
- Crear felicitación (ver Tabla 6.20): El usuario introduce los datos relativos a la felicitación. El sistema la almacena y la hará llegar al destinatario en el momento pertinente.
- Eliminar evento (ver Tabla 6.21): El usuario solicita eliminar un evento. El sistema lo elimina y notifica a los partícipes de éste.
- Eliminar felicitación (ver Tabla 6.22): El usuario solicita eliminar una felicitación. El sistema la elimina.
- Ver evento (ver Tabla 6.23): El usuario solicita ver más detalles sobre un evento. El sistema se los muestra.
- Ver felicitación (ver Tabla 6.24): El usuario solicita ver más detalles sobre una felicitación. El sistema se los muestra.

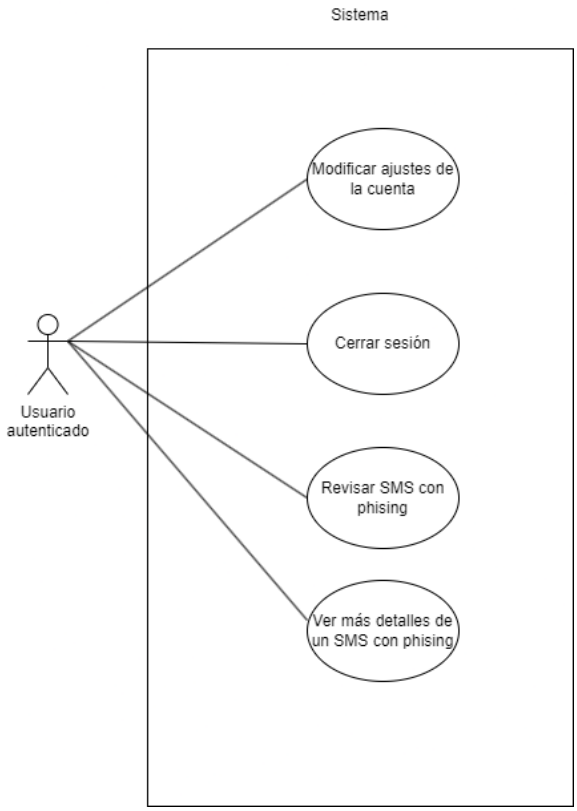

Figura 3.6: Diagrama de casos de uso - Usuario autenticado - Otras funcionalidades

Por último, en la figura 3.6 se visualizan los casos de uso restantes:

- Cerrar sesión (ver Tabla 6.25): El usuario solicita cerrar la sesión. El sistema elimina el token de la sesión.

- Revisar SMS con phishing (ver Tabla 6.26): El usuario solicita ver los SMS con posible phishing que ha recibido. El sistema se los lista.

- Ver más detalles de un SMS con phishing (ver Tabla 6.27): El usuario solicita ver con más detalle un SMS con posible phishing. El sistema le muestra el SMS completo.

- Modificar ajustes de la cuenta (ver Tabla 6.28): El usuario modifica el valor de alguna variable de su cuenta. El sistema almacena los cambios.

3.2 Mockups de la interfaz gráfica

En cuanto a la capa de presentación, se va a tomar como base la especificación de los diferentes casos de uso anteriormente descritos como apoyo para diseñar una interfaz gráfica que soporte todas las funcionalidades. Además, se tendrán en cuenta los diseños de la competencia para elaborar la misma.

Figura 3.7: Diseño de la interfaz - Iniciar sesión

Figura 3.8: Diseño de la interfaz - Crear nueva cuenta

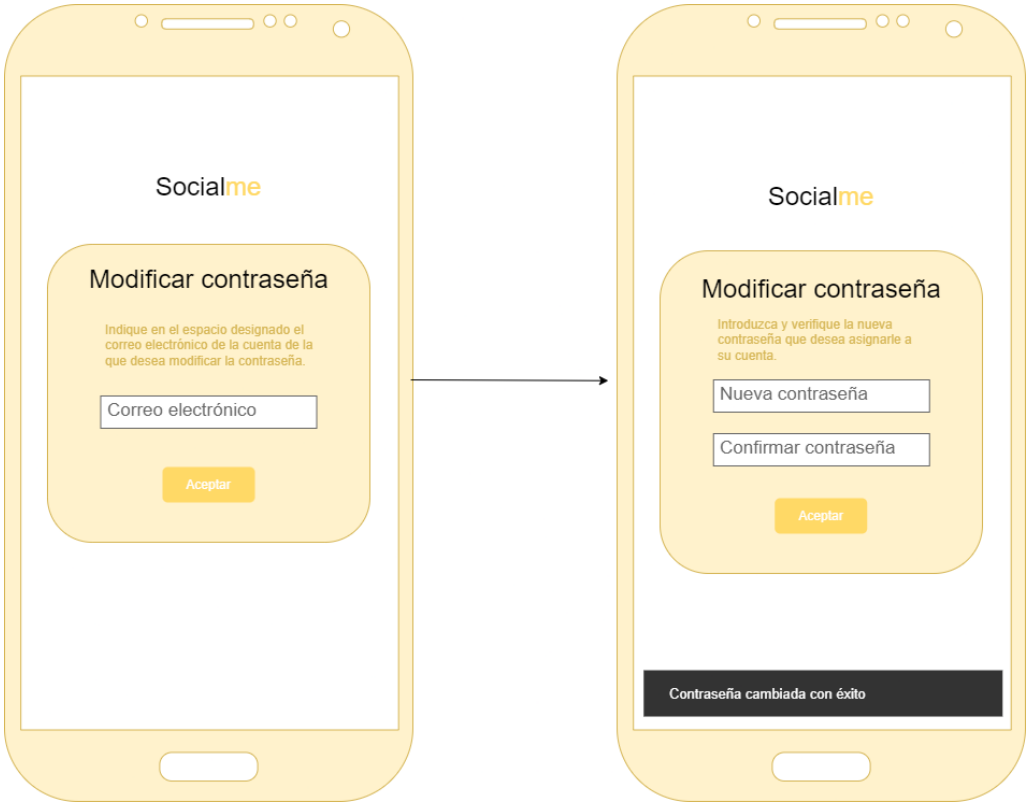

Figura 3.9: Diseño de la interfaz - Cambiar contraseña

Como se puede apreciar, las figuras 3.7, 3.8 y 3.9 muestran el diseño de la interfaz de los casos de uso ”Iniciar sesión”, ”Crear nueva cuenta” y ”Cambiar contraseña”, respectivamente.

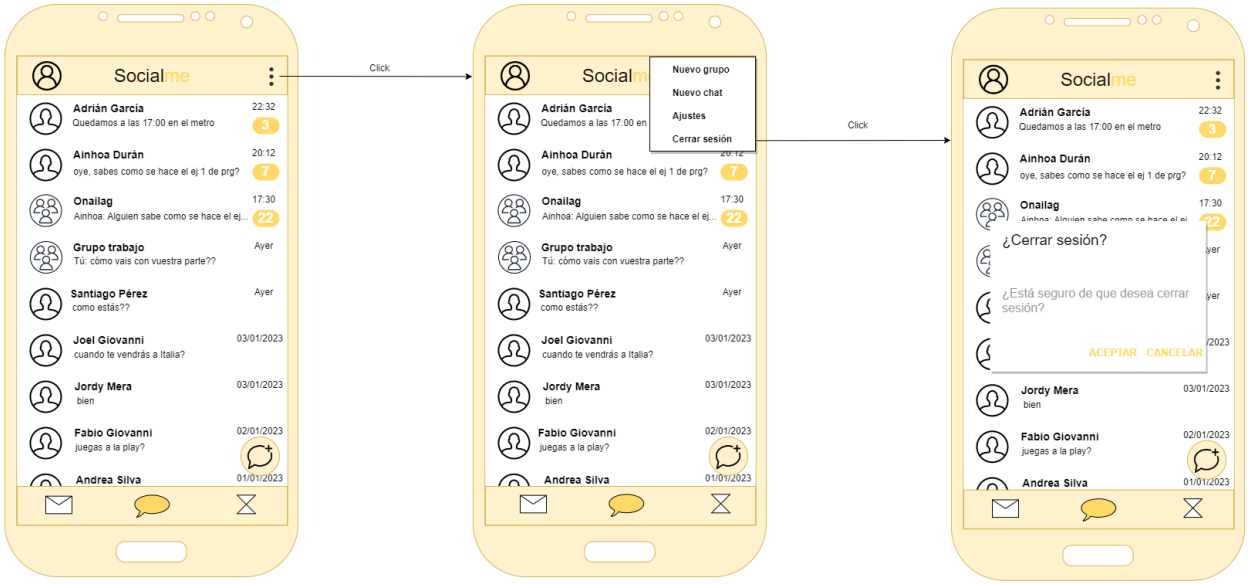

Figura 3.10: Diseño de la interfaz - Ver lista de últimos chats y Cerrar sesión

En la figura 3.10 se muestra la interfaz tanto para el caso de uso ”Ver lista de últimos chats” como para el de ”Cerrar sesión”. Además, la imagen 3.11 representa los casos de uso ”Ver un chat”, ”Enviar mensaje”, ”Enviar archivo”, ”Traducir mensaje” y ”Programar envío”.

Figura 3.11: Diseño de la interfaz - Ver un chat, Enviar mensaje, Enviar archivo, Traducir mensaje y Programar envío

En lo referente a la figura 3.12, ésta ilustra los casos de uso ”Ver más información sobre el chat” y ”Eliminar participante”.

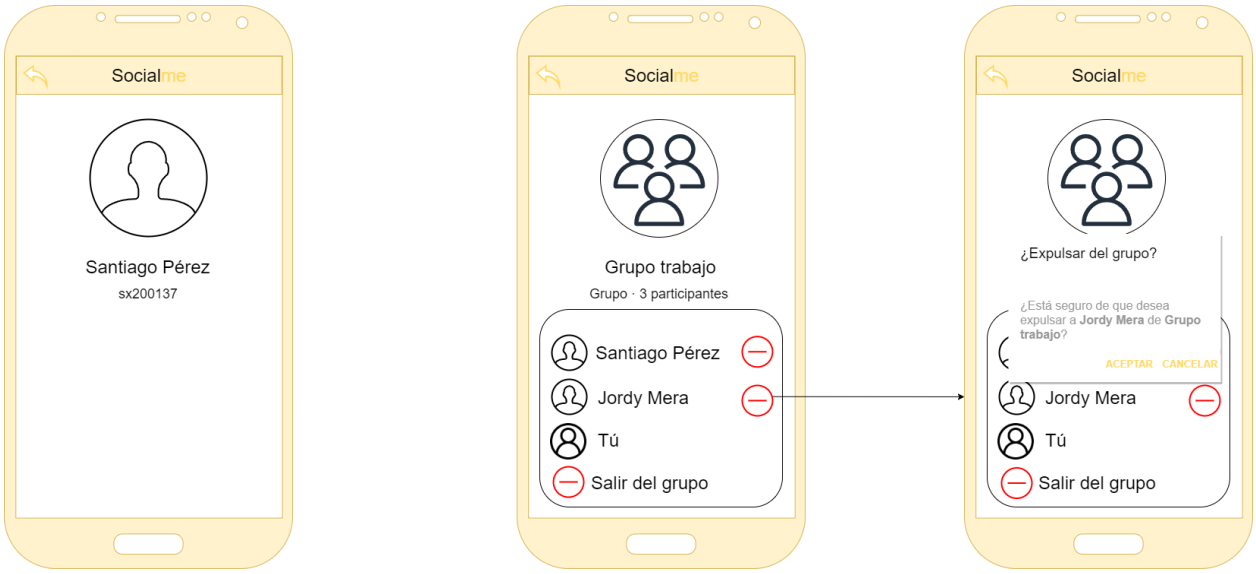

Figura 3.12: Diseño de la interfaz - Ver más información sobre el chat y Eliminar participante

Los casos de uso ”Crear nuevo chat”, ”Listar usuarios favoritos” y ”Eliminar usuarios de favoritos"son representados en la figura 3.13

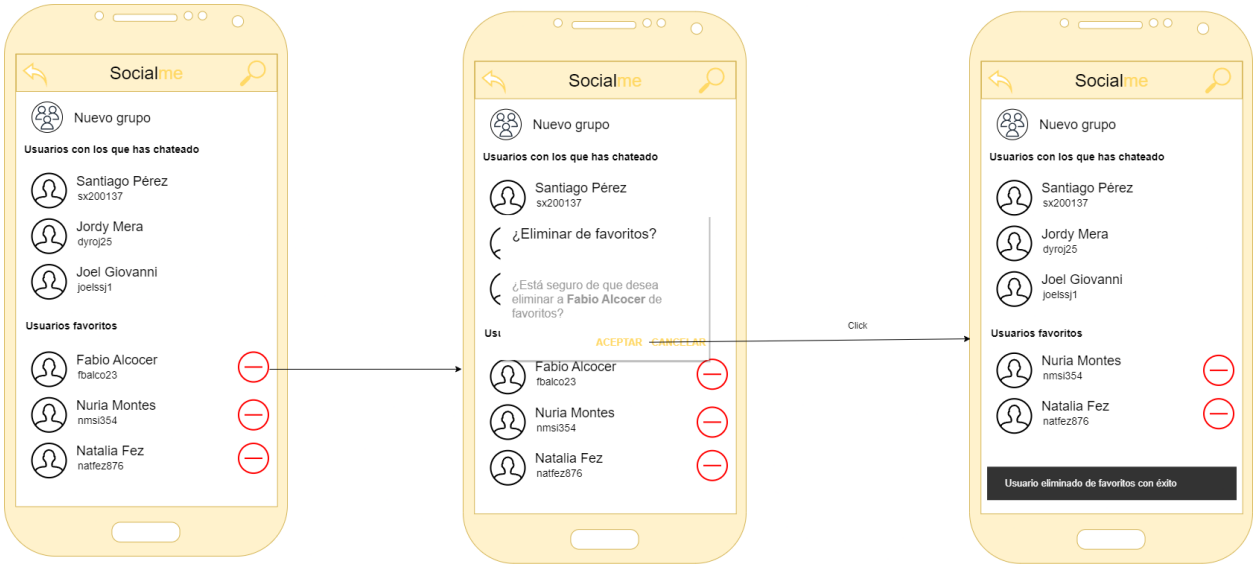

Figura 3.13: Diseño de la interfaz - Crear nuevo chat, Listar usuarios favoritos y Eliminar usuarios de favoritos

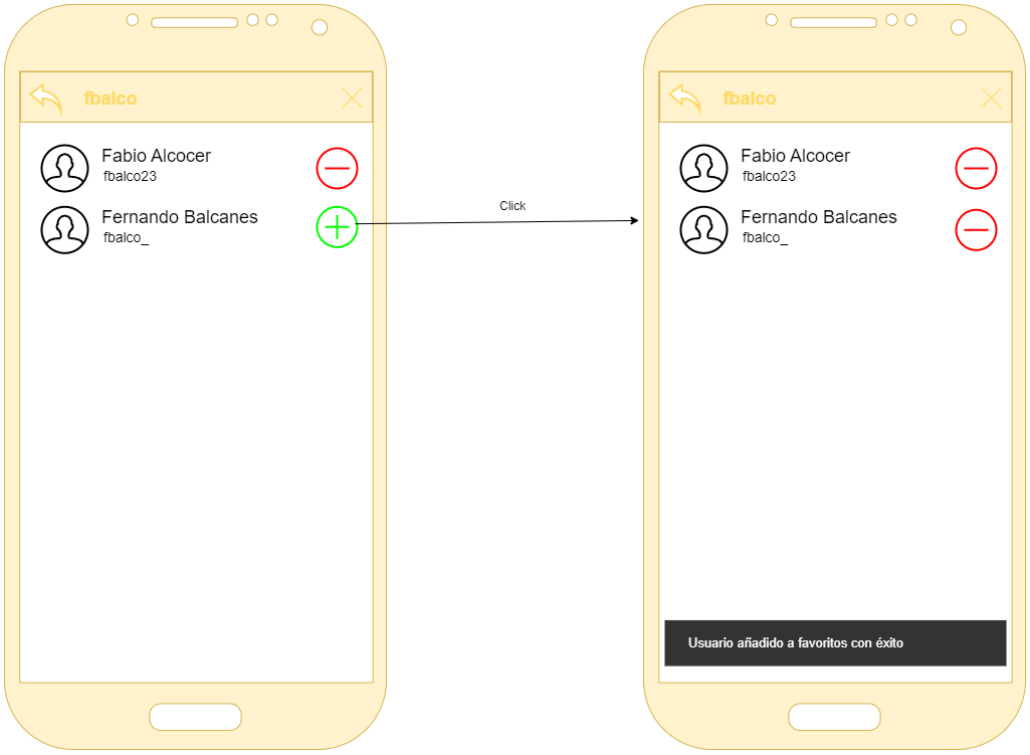

Figura 3.14: Diseño de la interfaz - Buscar usuario y Guardar usuario como favorito

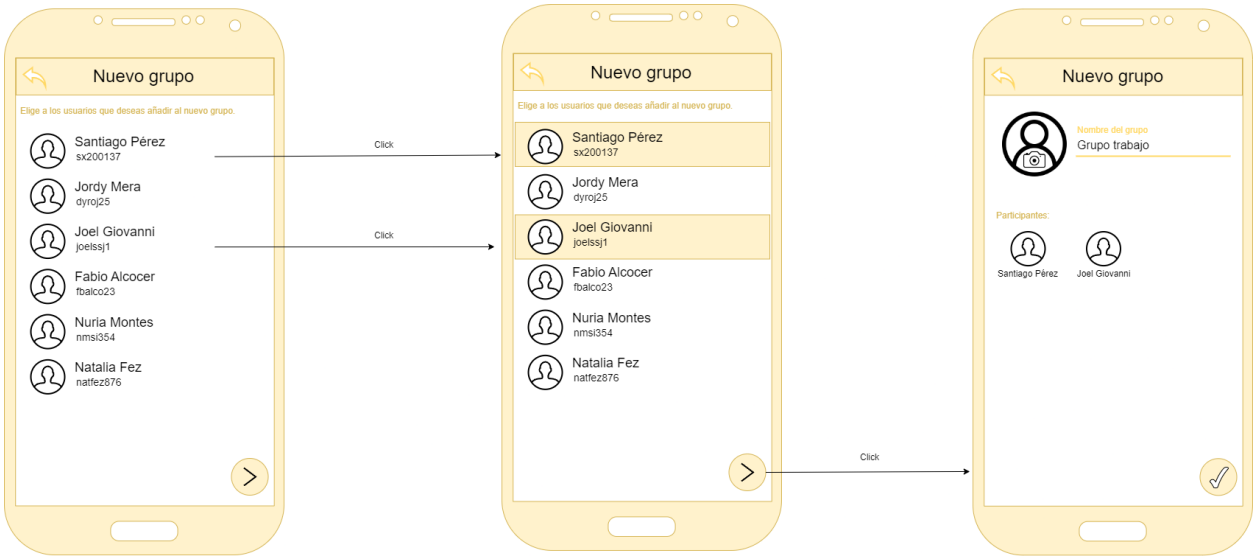

Figura 3.15: Diseño de la interfaz - Crear nuevo grupo

Las figuras 3.14 y 3.15 exhiben los casos de uso "Buscar usuarioz "Guardar usuario como favorito", y Çrear nuevo grupo", respectivamente.

Figura 3.16: Diseño de la interfaz - Ver eventos y felicitaciones

En cuanto a las figuras 3.16, 3.17 y 3.18, manifiestan los casos de uso ”Ver eventos y felicitaciones”, ”Crear eventoz ”Crear felicitación”, respectivamente.

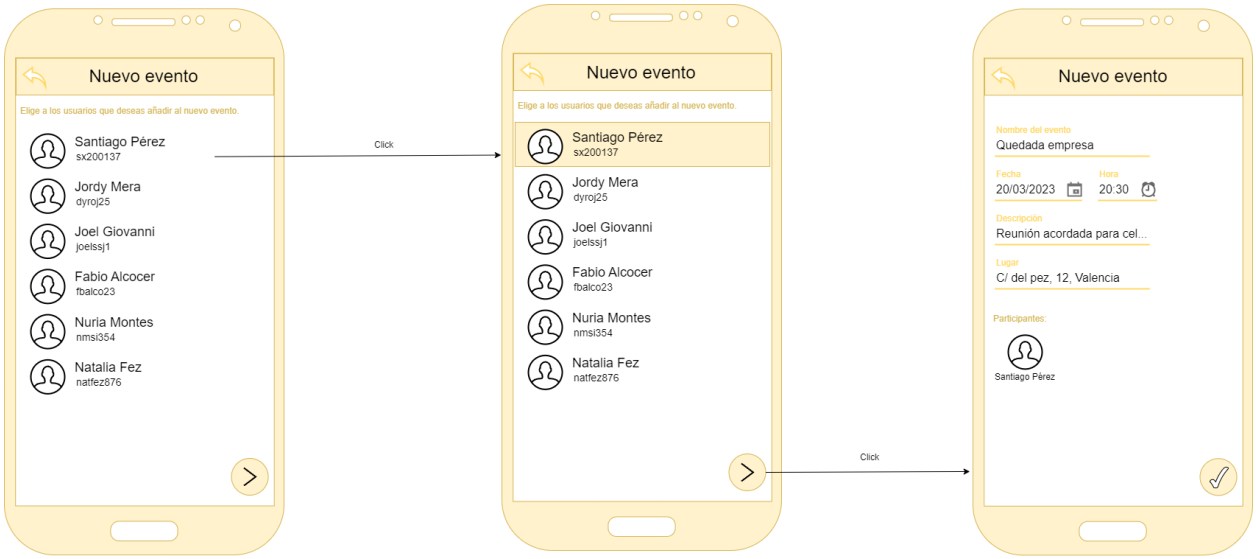

Figura 3.17: Diseño de la interfaz - Crear evento

Figura 3.18: Diseño de la interfaz - Crear felicitación

Los casos de uso ”Ver evento” y ”Eliminar evento” son descritos en la imagen 3.19. De la misma forma que la figura 3.20 que exhibe los casos de uso ”Ver felicitación” y ”Eliminar felicitación”.

Figura 3.19: Diseño de la interfaz - Ver evento y Eliminar evento

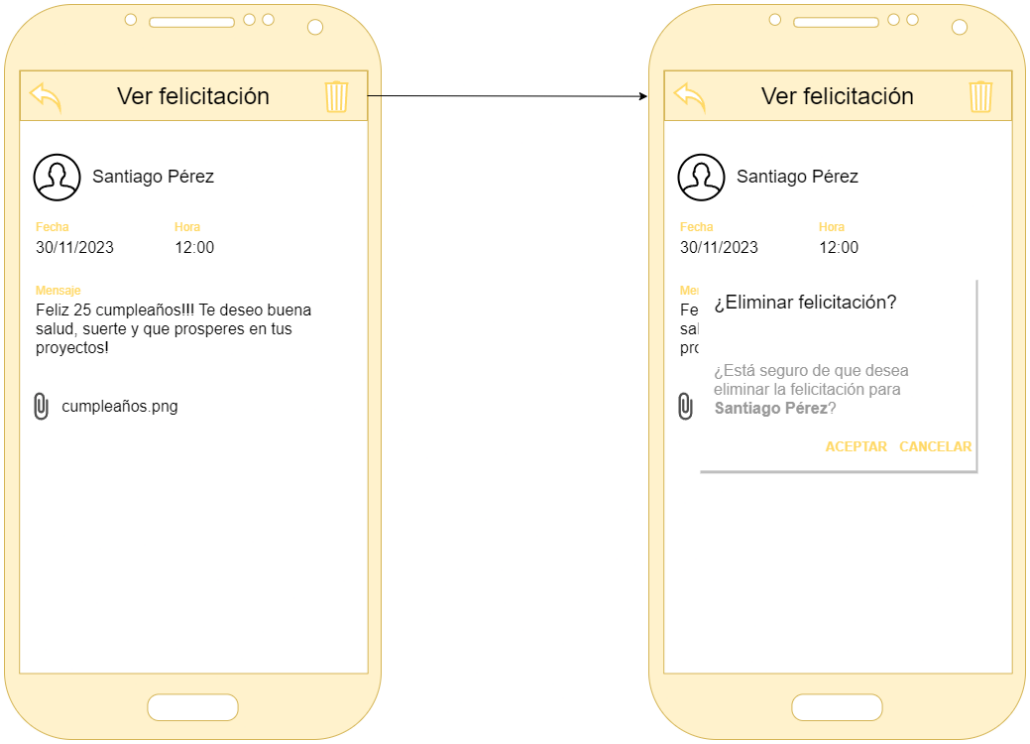

Figura 3.20: Diseño de la interfaz - Ver felicitación y Eliminar felicitación

Finalmente, la figura 3.21 manifiesta los casos de uso ”Revisar SMS con phishing” y ”Ver más detalles de un SMS con phishing”, y la figura 3.22 representa el caso de uso ”Modificar ajustes de la cuenta”.

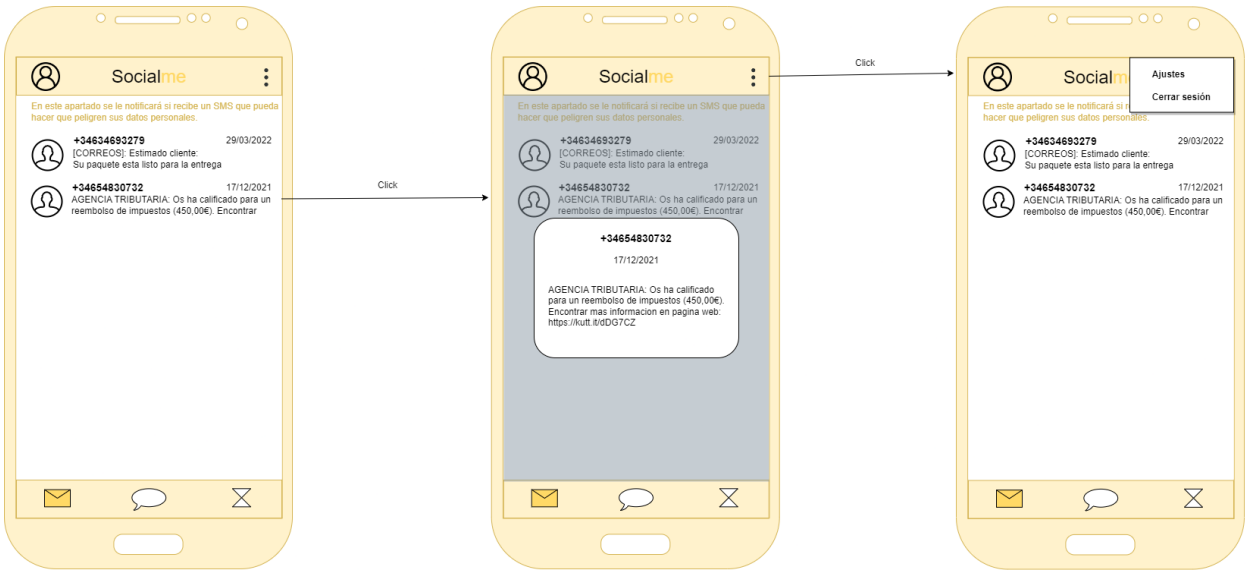

Figura 3.21: Diseño de la interfaz - Revisar SMS con phishing y Ver más detalles de un SMS con phishing

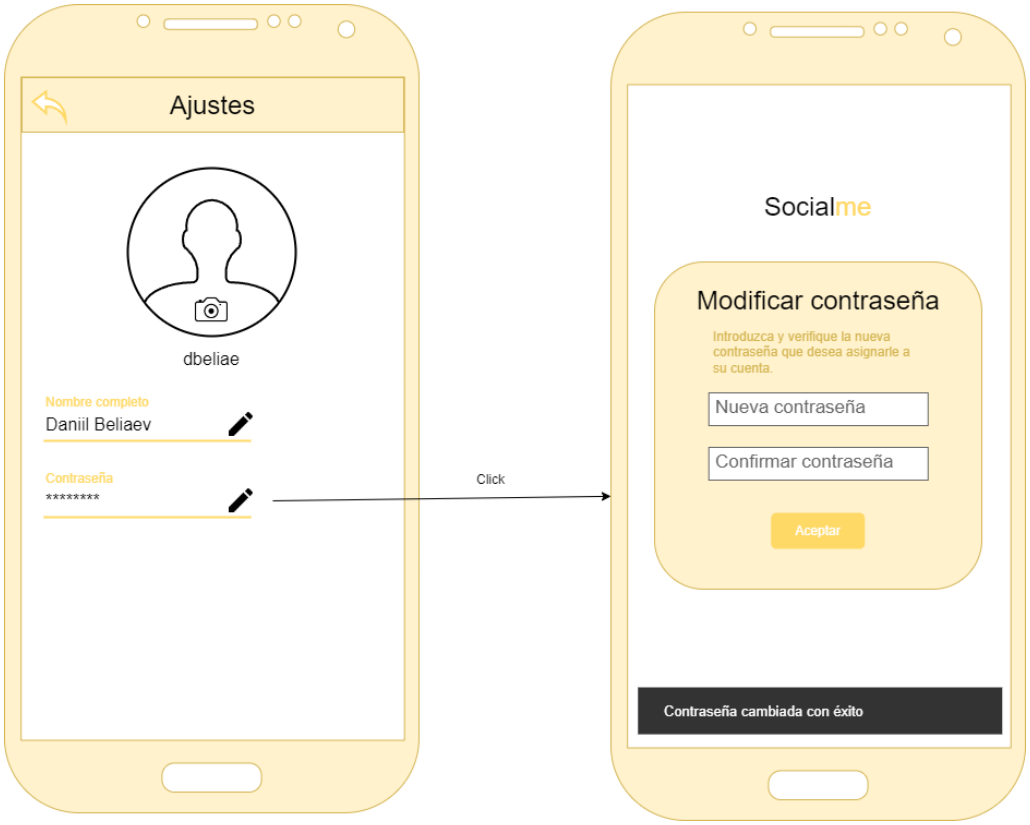

Figura 3.22: Diseño de la interfaz - Modificar ajustes de la cuenta

3.3 Diagrama de base de datos

En esta sección se mostrarán y describirán los diagramas de base de datos tanto para la aplicación móvil como para el servidor backend. Para cada diagrama se explicará el motivo de la existencia de las tablas. Para obtener una descripción detallada de cada atributo (columna), es necesario consultar el anexo.

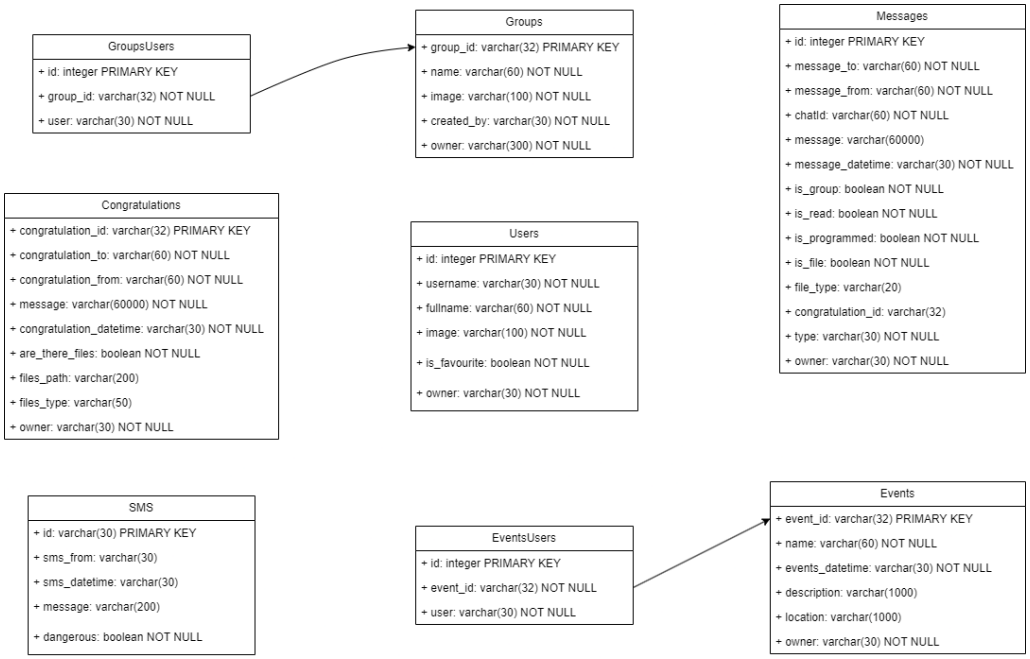

Figura 3.23: Diagrama de base de datos - Aplicación móvil

El diagrama mostrado en la figura 3.23 representa la base de datos que será utilizada en la aplicación móvil.

Tabla ”Users” (ver Sección 6.2.1): Almacena la información de todos los usuarios, tanto del que ha iniciado sesión como de los usuarios con los que ha hablado o ha marcado como favoritos este último.

Tabla ”Groups” (ver Sección 6.2.2): Almacena la información de todos los grupos en los que participa el usuario que ha iniciado sesión.

Tabla ”GroupsUsers” (ver Sección 6.2.3): Guarda los usuarios que pertenecen a un grupo con un id específico.

Tabla ”Messages” (ver Sección 6.2.4): Guarda todos los mensajes enviados y recibidos por el usuario que ha iniciado sesión (tanto los enviados instantáneamente como los programados).

Tabla ”Events” (ver Sección 6.2.5): Guarda todos los eventos de los que es partícipe el usuario que ha iniciado sesión (tanto los creados como en los que ha sido añadido).

Tabla ”EventsUsers” (ver Sección 6.2.6): Guarda los usuarios que pertenecen a un evento con un id específico.

Tabla ”Congratulations” (ver Sección 6.2.7): Guarda todas las felicitaciones creadas por el usuario que ha iniciado sesión.

Tabla ”SMS” (ver Sección 6.2.8): Guarda todos los SMS que han sido analizados.

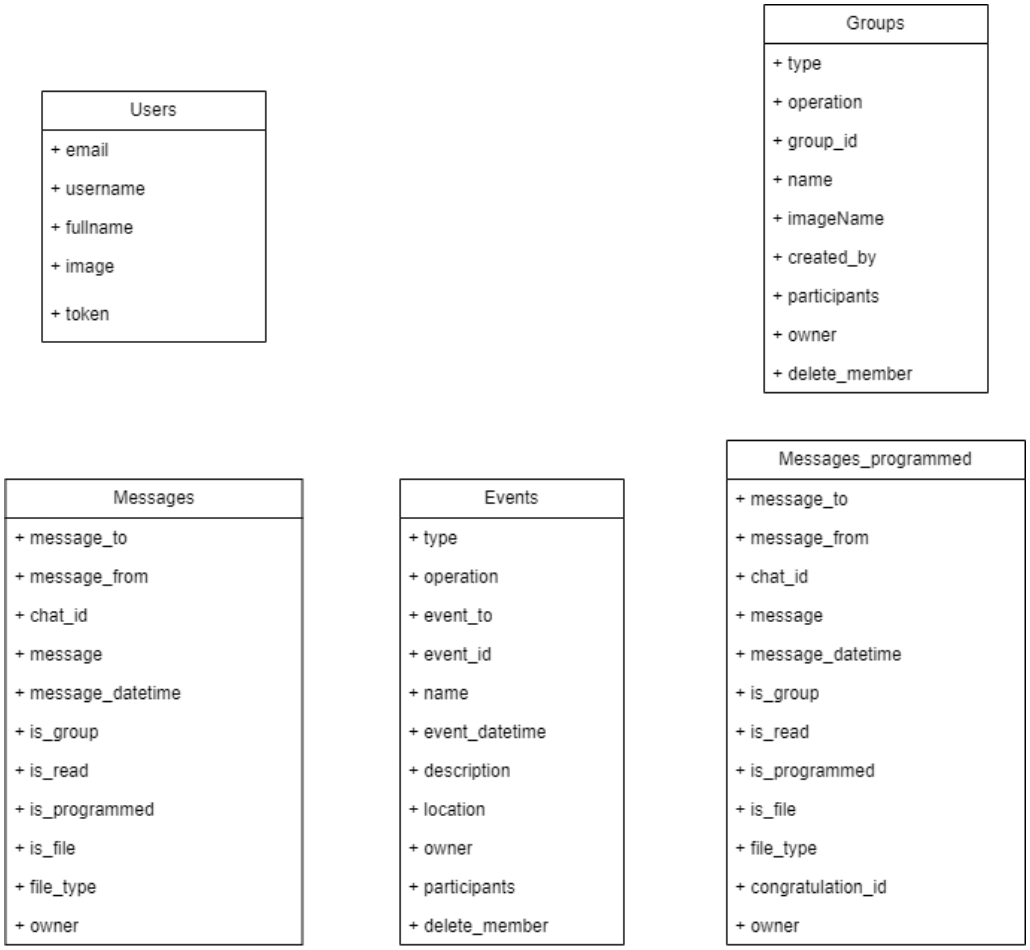

Figura 3.24: Diagrama de base de datos - Servidor backend Firebase

Como se puede apreciar en la figura 3.24, no se han especificado restricciones ni tipo a los atributos de las tablas. Esto es debido a que se pretende utilizar en la implementación una base de datos NoSQL, por lo que se organizan los datos en colecciones (tablas) y documentos (filas) en formato JSON. El significado de las tablas se mantiene igual que el visto en el anterior diagrama (ver Secciones 6.2.9, 6.2.10 y 6.2.11).

CAPÍTULO 4
Implementación

El diseño ha ofrecido un punto de partida a partir del cual comenzar a implementar la aplicación final. En este capítulo se describirán todas las decisiones tomadas partiendo de la generalización ofrecida por el diseño hasta el detalle de la implementación de cada función.

4.1 Tecnologías empleadas

Una de las primeras decisiones a tomar consiste en seleccionar las herramientas y entornos de desarrollo con los cuales se llevará a cabo el proceso de implementación. Esta decisión resulta importante ya que a partir de ésta se construirán los cimientos del producto. Los principales factores a tener en cuenta a la hora de elegir una tecnología u otra es la comodidad y la facilidad para adaptarse a ella y el grado de familiarización que se tiene con ésta.

4.1.1. Tecnologías utilizadas en el *frontend*

Android Studio

Android Studio[1] es el entorno de desarrollo integrado oficial de Android basado en el potente editor de código IntelliJ IDEA[2] y creado con el único fin de acelerar el desarrollo de aplicaciones para los dispositivos Android con el objetivo de que ofrezcan la mayor calidad (figura 4.1). Para ello, ofrece las siguientes características:

- Sistema de compilación flexible y compatible con Gradle.
- Sistema de emulación de diversos dispositivos Android para probar las aplicaciones desarrolladas.
- Entorno unificado en el cual desarrollar para todos los sistemas Android.
- Programación cómoda al ofrecer información sobre los errores en el código y las importaciones que realizar.
- Integración con repositorios Git.
- Ofrecimiento de plantillas de código para las funciones más comunes.

[1] https://developer.android.com/studio
[2] https://www.jetbrains.com/idea/

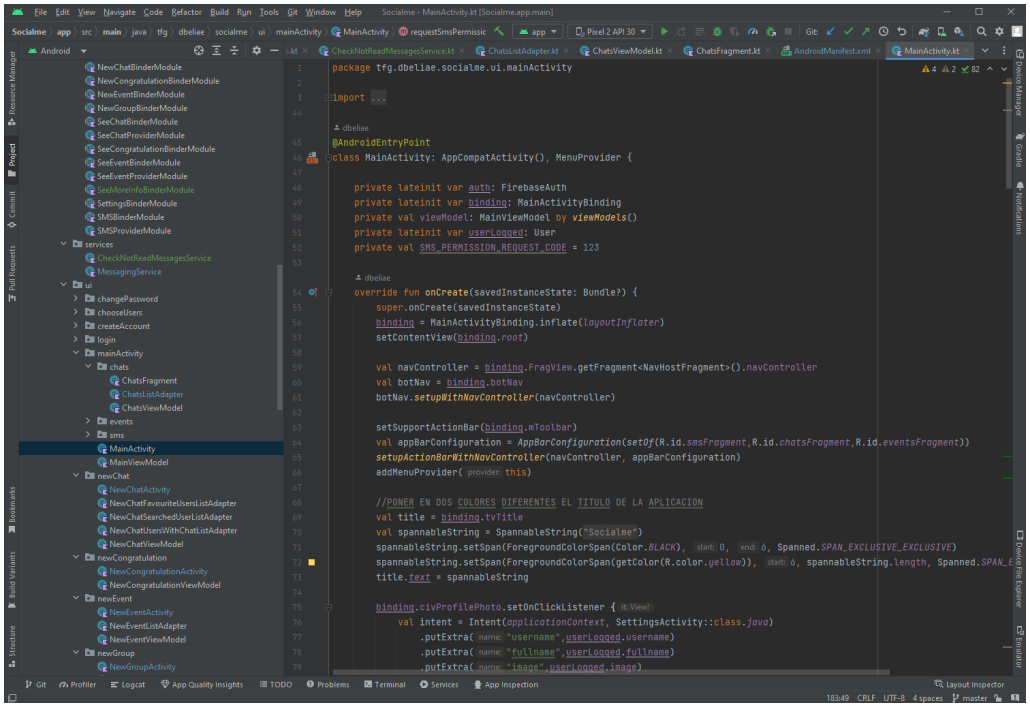

Figura 4.1: Interfaz principal de Android Studio

Kotlin

Kotlin[3] es un lenguaje de programación multiplataforma, estáticamente tipado, de alto nivel y propósito general con inferencia de tipos. Kotlin está diseñado para ser totalmente interoperable con Java[3]. Se ha seleccionado debido a que es un lenguaje con código conciso parecido a Java (ver comparación en la figura 4.2) y es el preferido de Google para el desarrollo de aplicaciones Android.

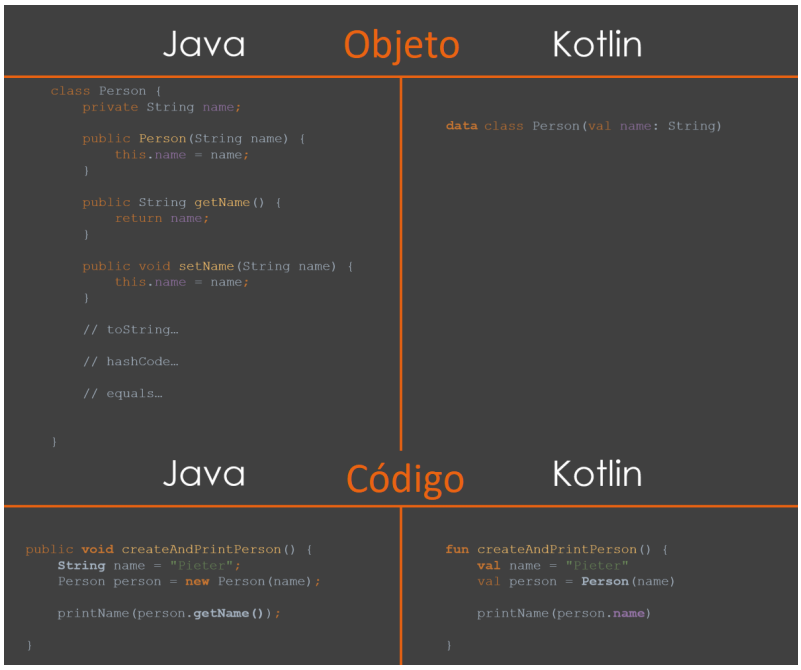

Figura 4.2: Diferencias entre Java y Kotlin

[3]https://kotlinlang.org/

4.1.2. Tecnologías utilizadas en el *backend*

Firebase

Firebase[4] es una plataforma de Google para el desarrollo de aplicaciones web y aplicaciones móviles ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos. Entre sus características se halla la sincronización entre proyectos sin tener que administrar conexiones, la escalabilidad automática de las aplicaciones y la creación de proyectos sin necesidad de un servidor[4].

Firebase ofrece una elevada cantidad de servicios que ofrecen diferentes funcionalidades. Para el desarrollo de la aplicación se han seleccionado los siguientes:

- **Authentication:** Proporciona servicios de *backend* para la autenticación de usuarios de la aplicación. Permite la autenticación mediante contraseña, número de teléfono o proveedores de identidad federada como Google, Facebook y Twitter.
- **Firestore Database:** Base de datos NoSQL flexible y escalable. Los datos se estructuran en documentos que contienen campos que se asignan a valores y estos documentos se almacenan en colecciones, que son contenedores para los mismos.
- **Storage:** Servicio de almacenamiento de imágenes, audio, video y otros tipos de archivos. Además, agrega la seguridad de Google a las operaciones de carga y descarga.
- **Messaging:** Solución de mensajería multiplataforma que permite enviar mensajes de forma segura.
- **Functions:** *Framework* sin servidores que permite ejecutar de forma automática el código de *backend* en respuesta a eventos activados por funciones de Firebase y solicitudes HTTPS.

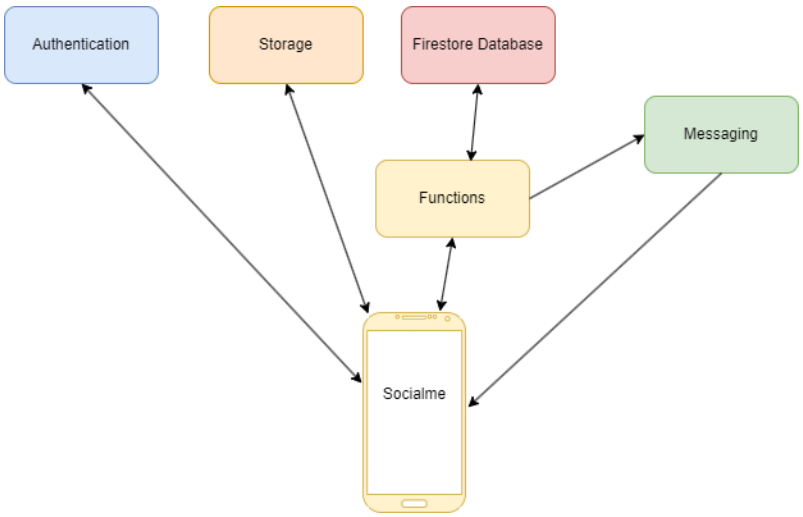

Figura 4.3: Comunicación entre los productos de Firebase y la aplicación

Una vez descritos los servicios de Firebase que se van a emplear, es necesario establecer la arquitectura de comunicaciones entre los servicios y la aplicación Android tal y como se puede apreciar en la figura 4.3.

[4]https://firebase.google.com/?hl=es

4.2 Componentes de una aplicación en Android

Al desarrollar una aplicación para Android es necesario conocer los diferentes componentes que pueden formar parte de ella para poder decidir la mejor alternativa a la hora de implementar una funcionalidad. Una aplicación puede constar de los siguientes componentes[5]:

- **Actividad:** Consta de una pantalla con una interfaz gráfica. Permite interacción con el usuario.
- **Servicio:** Componente que realiza tareas en segundo plano y no dispone de interfaz para interactuar con el usuario.
- **Receptor de emisión:** Ofrece la posibilidad de responder a notificaciones del sistema.
- **Proveedor de contenido:** Permite compartir datos entre aplicaciones.

Para implementar la solución solo ha sido necesario crear actividades y servicios.

4.3 Arquitectura para acceso a datos

Una vez vistos los diferentes componentes que pueden estar disponibles en una aplicación, es necesario establecer una arquitectura mediante la cual esos componentes pueden obtener datos ya sea de una base de datos o Internet. Se ha decidido implantar una arquitectura basada en la separación de dependencias como se puede apreciar en la figura 4.4.

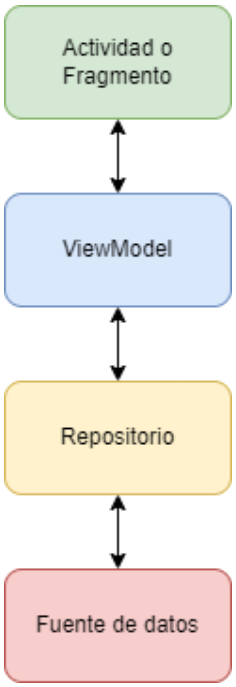

Figura 4.4: Arquitectura de la aplicación para acceso a datos

Cada elemento de esta arquitectura tiene una función:

- Actividad o Fragmento: Elementos de interfaz gráfica que muestran datos en pantalla.
- ViewModel: Mantiene datos, los exponen a la interfaz de usuario y gestiona lógica.
- Repositorio: Su función es abstraer y encapsular la lógica de acceso a datos. Puede proporcionar datos al ViewModel desde diferentes fuentes.
- Fuente de datos: Representa una fuente de datos real, como una base de datos local o una API remota. Su función es proporcionar acceso directo a los datos subyacentes.

4.4 Estructura del proyecto

La estructura de cualquier proyecto de Android Studio se divide en las siguientes tres carpetas[6]:

- **manifests:** Contiene el archivo AndroidManifest.xml
- **java:** Contiene los archivos de código fuente de Kotlin
- **res:** Contiene todos los recursos sin código, como cadenas de IU o la descripción de las interfaces y, además, imágenes de mapa de bits.

A continuación, se darán más detalles sobre cada carpeta de la estructura de la aplicación creada:

- AndroidManifest.xml: En este archivo se declaran todas las actividades, servicios, receptores de emisión y proveedores de contenido creados. Además, se especifican los permisos que la aplicación necesita para su correcto funcionamiento. Para la solución creada ha sido necesario solicitar los siguientes permisos:
  - READ_EXTERNAL_STORAGE: Permite a una aplicación leer el almacenamiento externo.
  - INTERNET: Permite a las aplicaciones abrir *sockets* de red.
  - ACCESS_NETWORK_STATE: Permite a las aplicaciones acceder a información sobre las redes.
  - READ_SMS: Permite a una aplicación leer los mensajes SMS.
- java: En esta carpeta se halla el código fuente de la aplicación distribuido en las siguientes carpetas:
  - data: En esta carpeta se incluyen directorios para cada actividad o servicio que accede a la base de datos o realiza una petición HTTP. En cada de estos directorios se incluye la descripción e implementación de los repositorios y de las fuentes de datos.
  - database: En este directorio se describe la base de datos utilizada por la solución.
  - di: En esta carpeta se declaran las inyecciones de dependencias.
  - services: En este directorio se implementan los servicios.
  - ui: En esta carpeta se implementan las actividades y los ViewModels relacionados a éstas.

- res: Esta carpeta contiene a la vez diversas subcarpetas:
  - drawable: Incluye los iconos utilizados por la aplicación.
  - layout: Incluye todas las descripciones relativas a las interfaces de usuario.
  - values: Conformada por los siguientes archivos:
    - colors.xml: Archivo en el que se almacenan colores.
    - strings.xml: Fichero en el que se almacenan las cadenas de caracteres utilizadas por la aplicación.

4.5 Bibliotecas empleadas

A lo largo del proceso de desarrollo, se han incorporado bibliotecas con el propósito de simplificar dicho proceso y aportar nuevas características.

4.5.1. uCrop

uCrop es una biblioteca que ofrece la funcionalidad de, a partir de una Uri, permitir al usuario recortar una imagen seleccionada por el mismo y almacenarla en la Uri especificada. Se ha incluido en el proyecto tanto para la elección de una foto de usuario al crear una cuenta como para seleccionar una foto al crear un grupo.

Para poder emplearla en la aplicación hay que seguir los siguientes pasos:

1. Añadir UCropActivity en el AndroidManifest.xml:

```
<activity
    android:name="com.yalantis.ucrop.UCropActivity"
    android:screenOrientation="portrait"
    android:theme="@style/Theme.AppCompat.Light.NoActionBar"/>
```

2. Configurar los parámetros e iniciar la actividad:

```
UCrop.of(sourceUri, destinationUri)
    .withAspectRatio(16, 9)
    .withMaxResultSize(maxWidth, maxHeight)
    .start(context)
```

3. Sobrescribir el método onActivityResult y manejar el resultado de la actividad uCrop:

```
override fun onActivityResult(requestCode: Int,
resultCode: Int, data: Intent?) {
    if (resultCode == RESULT_OK && requestCode == UCrop.REQUEST_CROP) {
        val croppedImageUri = UCrop.getOutput(data)
    } else if (resultCode == UCrop.RESULT_ERROR) {
        val cropError = UCrop.getError(data!!)
    }
}
```

4.5.2. CircleImageView

CircleImageView es un ImageView circular que sirve para mostrar las imágenes de perfil de los usuarios y de los grupos.

Para usarlo basta simplemente con introducirlo en el archivo de *layout* XML deseado en lugar del ImageView. Ejemplo de la inserción de un CircleImageView:

```
<de.hdodenhof.circleimageview.CircleImageView
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/profile_image"
    android:layout_width="96dp"
    android:layout_height="96dp"
    android:src="@drawable/profile"
    app:civ_border_width="2dp"
    app:civ_border_color="#FF000000"/>
```

4.5.3. Glide

Glide es una librería que permite cargar imágenes en un elemento de diseño (*widget*) de la interfaz de usuario, como una ImageView o CircleImageView, mediante la especificación de una URL remota o una Uri local.

Se utiliza de la siguiente forma:

```
Glide.with(context)
            .load(UriOrURL)
            .into(binding.circleImageView)
```

4.5.4. Moshi

Moshi es una biblioteca que permite convertir objetos Kotlin en su representación JSON y viceversa. Será usado en la aplicación para implementar la funcionalidad de "Traducir mensaje" al recibir la respuesta JSON del servicio que traduce el mensaje.

4.5.5. Hilt

Hilt es una biblioteca de inyección de dependencias diseñada para Android, la cual simplifica la tarea repetitiva de incorporar dependencias manualmente[7].

4.5.6. Retrofit

Retrofit es un cliente HTTP para Android y Kotlin. Su funcionamiento se basa en describir mediante anotaciones una API HTTP como una interfaz para luego generar la implementación automáticamente.

4.5.7. Room

Room es una librería de persistencia en Android que se usa para simplificar y abstraer la capa de acceso a la base de datos en aplicaciones Android. Room proporciona una capa de abstracción por encima de SQLite, que es el sistema gestor de bases de datos incorporado en Android, para facilitar el almacenamiento y la recuperación de datos de manera eficiente.

4.5.8. Jsoup

Jsoup es una biblioteca Kotlin para trabajar con documentos HTML. Proporciona una API muy conveniente para extraer y manipular datos, utilizando lo mejor de los métodos del DOM de HTML5 y los selectores CSS. Esta librería será usada para analizar el documento HTML obtenido al enviar una petición de análisis del número del remitente de un SMS. Ejemplo de uso en el que se obtiene el valor del atributo "src" de la primera imagen con clase "scoreimage":

```
val document = Jsoup.parse(documentHTML)
val imgElements = document.select("img.scoreimage")
val srcAttributeValue = imgElements[0].attr("src")
```

4.6 Construcción de la base de datos

Como se ha mencionado anteriormente, se ha empleado la biblioteca Room para implementar el diagrama de base de datos mostrado en la figura 3.23. A continuación se va a describir los pasos seguidos para la creación de una base de datos con esta biblioteca[8]:

1. Crear una clase de datos con anotación *@Entity* por cada tabla de la base de datos. Ejemplo para la tabla "Users":

```
@Entity(tableName = "Users")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val username: String,
    var fullname: String,
    val image: String,
    @ColumnInfo(name = "is_favourite") val isFavourite: Boolean,
    val owner: String
)
```

2. Crear una interfaz de objecto de acceso a datos con anotación *@Dao* para cada tabla con el objetivo de proporcionar métodos para operaciones CRUD (*Create, Read, Update, Delete*) sobre la base de datos. Ejemplo para el Dao de "Users":

```
@Dao
interface UserDao {
    @Insert
    suspend fun insertUser(user: User)

    @Query("SELECT * FROM Users WHERE is_favourite = 1 AND owner = :owner")
    fun getAllFavouriteUsers(owner: String): Flow<List<User>>

    ...
    ...
}
```

3. Crear una clase abstracta de la base de datos con anotación *@Database* con el objetivo de proporcionar acceso a las operaciones anteriormente definidas a través de la interfaz *@Dao*. Definición de la base de datos empleada por la aplicación:

```
@Database(entities = [User::class,SMS::class,Event::class,
    EventUser::class,Group::class,GroupUser::class,Message::class,
    Congratulation::class], version = 1)
abstract class SocialmeDatabase: RoomDatabase() {
    abstract fun userDao(): UserDao
    abstract fun smsDao(): SMSDao
    abstract fun eventDao(): EventDao
    abstract fun congratulationDao(): CongratulationDao
    abstract fun eventUserDao(): EventUserDao
    abstract fun messageDao(): MessageDao
    abstract fun groupDao(): GroupDao
    abstract fun groupUserDao(): GroupUserDao
}
```

4.7 Configuración de los productos de Firebase

En esta sección se describirán las configuraciones realizadas a cada producto de Firebase utilizado en la aplicación.

4.7.1. Authentication

Primero se estableció como proveedor de acceso el correo electrónico y la contraseña (figura 4.5).

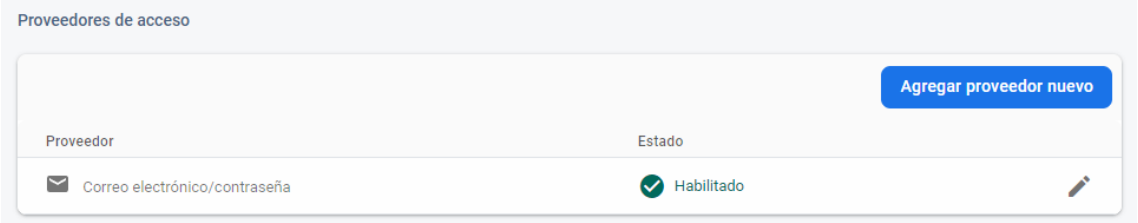

Figura 4.5: Correo electrónico/contraseña establecido como proveedor

Por último, se modificó la plantilla del correo que se envía en caso de que el usuario solicita cambiar la contraseña para introducir instrucciones adicionales (figura 4.6).

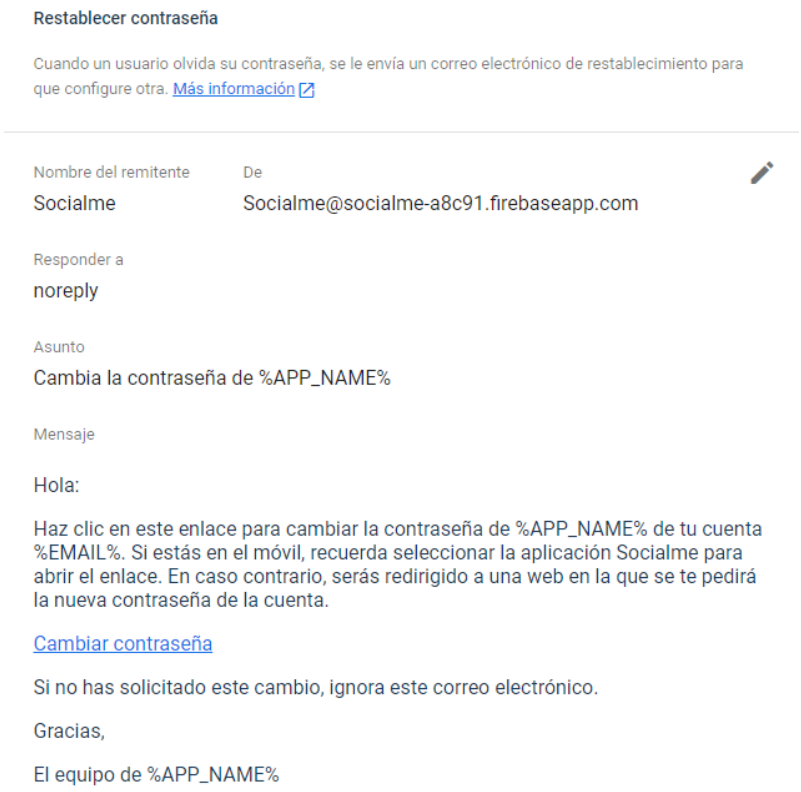

Figura 4.6: Plantilla del correo para cambiar contraseña modificada

4.7.2. Firestore Database

Como se puede apreciar en la figura 4.7, el único procedimiento que se realizó fue el de crear las colecciones pertinentes a partir del diagrama de base de datos de la figura 3.24.

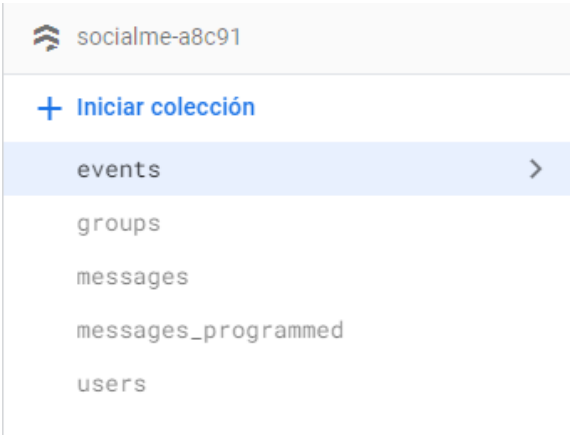

Figura 4.7: Colecciones creadas a partir del diagrama de base de datos

4.7.3. Storage

En este caso se crearon las siguientes carpetas tal y como se puede apreciar en la figura 4.8:

- photo_groups: Carpeta destinada a almacenar las fotos de los grupos creados por los usuarios.
- photo_profiles: La función de esta carpeta es guardar las fotos de perfil de los usuarios registrados.
- users_files: El objetivo de esta carpeta es almacenar los archivos enviados por los usuarios de la aplicación. Para ello, por cada usuario que envía un archivo se crea una carpeta y dentro de ésta se almacena el fichero.

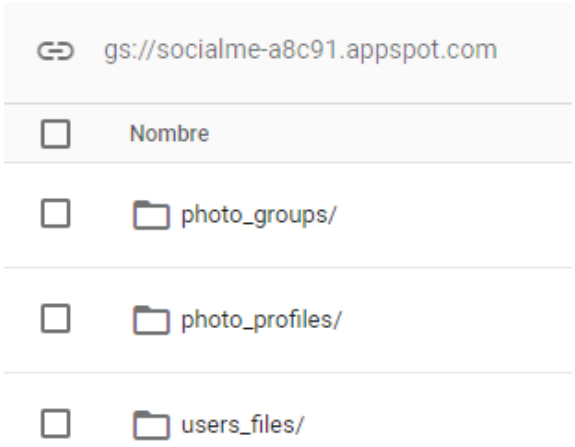

Figura 4.8: Carpetas creadas en Storage

4.7.4. Messaging

En lo referente a este producto, lo único que tuvo que realizarse fue su activación en la consola de Google Cloud tal y como se puede apreciar en la figura 4.9. En caso contrario, siempre se produciría una excepción al tratar de enviar un mensaje mediante este servicio.

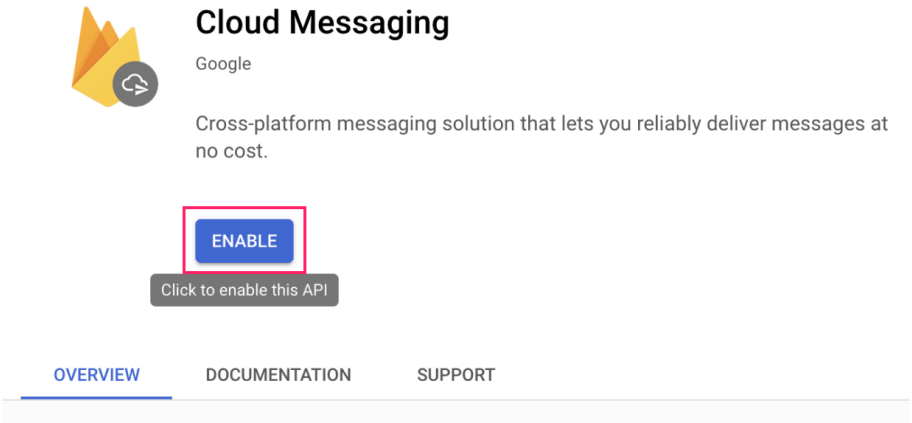

Figura 4.9: Activación del servicio Messaging en la consola de Google Cloud

4.7.5. Functions

La configuración de Functions requiere de los siguientes pasos[9]:

1. Instalar Node.js y npm.
2. Instalar Firebase CLI con el comando: *npm install -g firebase-tools*.
3. Ejecutar el comando: *firebase login* para acceder a través del navegador y autenticar Firebase CLI.
4. Seleccionar una ubicación en la que crear el directorio del proyecto.
5. Ejecutar las instrucciones *firebase init firestore* y *firebase init functions*.
6. Seleccionar JavaScript como lenguaje de programación.
7. Aceptar la instalación de dependencias.

Tras esos pasos se habrá generado la siguiente estructura de directorios en la ubicación seleccionada:

```
myproject
+- .firebaserc    # Archivo oculto que te ayuda a cambiar rápidamente entre
|                 # proyectos con  ‘firebase use‘
|
+- firebase.json  # Describe las propiedades del proyecto
|
+- functions/     # Directorio que contiene todo el código de las funciones
      |
      +- .eslintrc.json  # Archivo opcional que contiene reglas
      |                     # para el linting de JavaScript
      +- package.json  # Archivo de paquete npm que describe
      |                     # el código de Cloud Functions
      +- index.js      # Archivo principal para el código de Cloud Functions
      |
      +- node_modules/ # Directorio donde se instalan las dependencias
                        # declaradas en el archivo package.json
```

En el archivo ”index.js” se implementarán las funciones que atenderán las peticiones de la aplicación Android e interactuarán con Firestore Database, Storage y Messaging.

4.8 Implementación de las funciones

En esta sección, debido a la imposibilidad de una descripción detallada de la implementación, se destacarán los aspectos más relevantes en el desarrollo de las funcionalidades mencionadas en el diseño.

4.8.1. Iniciar sesión (LoginActivity)

En esta funcionalidad pueden darse 2 situaciones al iniciar la aplicación:

1. Que el usuario no haya iniciado sesión previamente o la haya cerrado.
2. Que el usuario ya haya iniciado sesión previamente y no la haya cerrado.

En el primer caso, como se puede apreciar en la figura 4.10, el usuario deberá de introducir las credenciales (correo electrónico y contraseña) y la aplicación se encargará de iniciar sesión y de comprobar que el correo haya sido verificado. Tras confirmar lo anterior, se actualizará el valor del atributo "token" (valor para identificar inequívocamente al dispositivo empleado en el servicio Messaging) del documento del usuario que ha iniciado sesión en Firestore Database ya que éste puede haber cambiado. Finalmente, se iniciará la actividad principal (MainActivity).

```
auth = Firebase.auth
auth.signInWithEmailAndPassword(email, password).addOnCompleteListener { task ->
    if (task.isSuccessful) {
        if (auth.currentUser!!.isEmailVerified) {
            val username = auth.currentUser!!.displayName!!
            //OBTENER EL TOKEN ACTUAL
            FirebaseMessaging.getInstance().token.addOnSuccessListener { token ->
                val documentRef = db.collection( collectionPath: "users").document(username!!)
                val updateToken = hashMapOf<String, Any>("token" to token)
                documentRef.update(updateToken)
                    .addOnSuccessListener { it: Void!
                        // INICIAR ACTIVIDAD PRINCIPAL
                        val intent = Intent(applicationContext,MainActivity::class.java)
                        startActivity(intent)
                    }
```

Figura 4.10: Primera situación al iniciar la aplicación

En el segundo caso, al no haber cerrado sesión, la aplicación obtiene de Authentication el usuario activo y comprueba si éste ha verificado su correo (esto es necesario porque si el usuario cierra la aplicación y la vuelve a abrir después de un intento fallido de inicio de sesión, podría eludir el proceso de verificación de correo). Una vez verificado lo anterior, se iniciará la actividad principal (MainActivity) tal y como se puede ver en la figura 4.11.

```
override fun onStart() {
    super.onStart()
    val currentUser = auth.currentUser
    if (currentUser != null && currentUser.isEmailVerified) {
        val intent = Intent(applicationContext,MainActivity::class.java)
        startActivity(intent)
    }
}
```

Figura 4.11: Segunda situación al iniciar la aplicación

4.8.2. Crear nueva cuenta (CreateAccountActivity)

En esta actividad, tras verificar que ningún campo este vacío y que la contraseña tenga al menos 6 caracteres de longitud, se procede a comprobar si el nombre de usuario está disponible. Si lo anterior se cumple, se crea la cuenta en Authentication con el método *createUserWithEmailAndPassword(email,password)*, se añade un documento con todos los datos del usuario a la colección "Users" de Firestore Database y se envía un *email* de verificación con el método *sendEmailVerification()* de Authentication.

4.8.3. Cambiar contraseña (ChangePasswordGetEmailActivity y ChangePasswordActivity)

Para la realización de esta funcionalidad, primero el usuario ingresa el correo electrónico de la cuenta de la cual desea modificar la contraseña y Authentication le envía un correo tras ejecutar el método *sendPasswordResetEmail(email)* en la actividad ChangePasswordGetEmailActivity.

Una vez realizado lo anterior, el usuario ha de ingresar su correo electrónico y realizar una de las siguientes opciones:

1. Abrir el enlace con el navegador

2. Abrir el enlace con Socialme

En el primer caso, el usuario será redirigido a una página web predefinida de Authentication para ingresar la nueva contraseña deseada (figura 4.12).

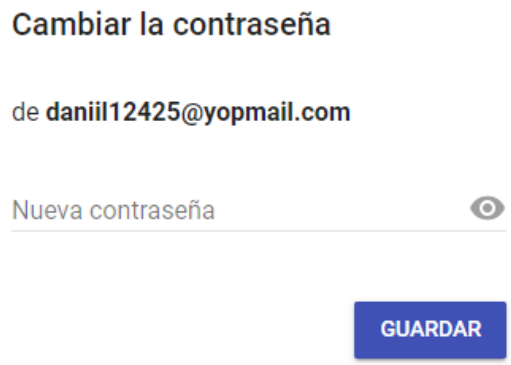

Figura 4.12: Web predefinida para cambiar la contraseña de la cuenta

En el segundo caso, se iniciará la actividad ChangePasswordActivity y el usuario deberá de ingresar y confirmar la nueva contraseña (figura 4.13). Tras ello, se ejecutará el método *confirmPasswordReset(code, newPassword)* de Authentication siendo ”code” el valor del parámetro de consulta ”oobCode” de la URL del correo.

Figura 4.13: Actividad de la aplicación para cambiar la contraseña de la cuenta

4.8.4. Cerrar sesión (en MainActivity)

Cuando el usuario selecciona la opción de ”Cerrar sesión” en el menú, se ejecuta la función de la figura 4.14 y se despliega un AlertDialog:

```
private fun showLogoutDialog() {
    val builder = AlertDialog.Builder( context: this)
    builder.setTitle("Log out")
        .setMessage("Are you sure that you want to log out?")
        .setIcon(AppCompatResources.getDrawable( context: this, R.drawable.ic_logout))
        .setPositiveButton("Submit") {_, _ ->
            auth.signOut()
            finish()
        }
        .setNegativeButton("Cancel") {_, _ ->
            //SE CIERRA EL DIALOGO
        }
        .show()
}
```

Figura 4.14: Código del diálogo para cerrar sesión

4.8.5. Revisar SMS con phishing (SMSFragment)

En esta función, se obtiene la lista de identificadores de los SMS desde la base de datos de la aplicación y se ejecuta la función de la figura 4.15. En ella se obtiene todos los SMS del dispositivo enviando una consulta al ContentResolver y, para cada uno, se comprueba si el identificador del SMS se encuentra en la lista anteriormente obtenida. En

caso de que no se encuentre, primero se ejecuta la función *containsLink(body)* (figura 4.16) la cual comprueba si en el cuerpo del SMS hay un enlace y lo cataloga como peligroso en caso afirmativo. Si no se cumple la condición anterior, se realiza una petición HTTP usando Retrofit para comprobar si el número de teléfono del remitente es peligroso. Para ello, se envía una petición a la web Tellows[5] y se procesa el HTML recibido para obtener el número del nombre de la imagen incluida en el valor del atributo ”src” de la primera imagen con clase ”scoreimage” que indica el grado de peligrosidad del número del remitente (figura 4.17).

```
fun readAndProcessSms(listSms: List<String>) {
    val contentResolver = requireActivity().contentResolver

    val cursor = contentResolver.query(
        Telephony.Sms.CONTENT_URI,
        projection: null,
        selection: null,
        selectionArgs: null,
        sortOrder: "date ASC"
    )

    cursor?.let { it: Cursor

        val addressIndex = it.getColumnIndex(Telephony.Sms.ADDRESS)
        val bodyIndex = it.getColumnIndex(Telephony.Sms.BODY)
        val timeStampMillisIndex = it.getColumnIndex(Telephony.Sms.DATE)
        val smsIdIndex = it.getColumnIndex(Telephony.Sms._ID)

        while (it.moveToNext()) {
            // Procesar los mensajes SMS aquí.
            val address = it.getString(addressIndex)
            val body = it.getString(bodyIndex)
            val timestampMillis = it.getLong(timeStampMillisIndex)
            val smsId = it.getLong(smsIdIndex)

            if (!listSms.contains(smsId.toString())) {
                // Convertir el timestamp a una fecha y hora legible
                val calendar = Calendar.getInstance()
                calendar.timeInMillis = timestampMillis
                val simpleDateFormat = SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss", Locale.getDefault())
                val formattedDateTime = simpleDateFormat.format(calendar.time)

                //COMPROBAR SI HAY UN ENLACE EN EL SMS
                if (containsLink(body)) {
                    //GUARDAR EL SMS EN LA BASE DE DATOS
                    viewModel.insertSMS(SMS(smsId.toString(), address, formattedDateTime, body,  dangerous: true))
                } else {
                    //COMPROBAR SI EL NUMERO QUE HA ENVIADO EL SMS ES PELIGROSO
                    if (address != null) {
                        viewModel.getAddressInfo(address, body, formattedDateTime, timestampMillis.toString(), smsId.toString())
                    }
```

Figura 4.15: Código del análisis de los SMS

```
private fun containsLink(text: String): Boolean {
    val regexPattern = """\b(?:https?://|www\.)\S+\b""".toRegex(RegexOption.IGNORE_CASE)
    return regexPattern.find(text) != null
}
```

Figura 4.16: Código para comprobar si un texto incluye un enlace

[5]https://www.tellows.es/

```
private val retrofitSMSAddressInfoService = retrofit.create(SMSAddressInfoService::class.java)

dbeliae
override suspend fun getAddressInfo(phoneNumber: String): Int {
    val html = retrofitSMSAddressInfoService.getAddressInfo(phoneNumber)
    val document = Jsoup.parse(html)

    val imgElements = document.select( cssQuery: "img.scoreimage")

    if (imgElements.isEmpty()) { return 0 }

    val srcAttributeValue = imgElements[0].attr( attributeKey: "src")
    val fileNameWithoutExtension = srcAttributeValue.substringAfterLast( delimiter: '/').substringBefore( delimiter: '.')
    return fileNameWithoutExtension.substring( startIndex: 1).toInt()
}

dbeliae
interface SMSAddressInfoService {
    dbeliae
    @GET("num/{phoneNumber}")
    suspend fun getAddressInfo(@Path("phoneNumber") phoneNumber: String): String
}
```

Figura 4.17: Código para comprobar si el número de teléfono del remitente es peligroso

4.8.6. Crear evento (NewEventActivity)

En esta sección se va a aprovechar para explicar la forma que tiene de comunicarse la aplicación móvil con el servicio Cloud Functions y la forma de enviar mensajes empleando el producto Messaging desde Cloud Functions.

Una vez que el usuario completa los campos pertinentes se genera un identificador único aleatorio para el evento y se envían todos los datos a la función *createEvent* de Cloud Functions. Si la función tiene éxito se guarda el evento y los participantes de este en la base de datos de la aplicación (figura 4.18).

```
//SE GENERA UNA IDENTIFICADOR ALEATORIO DE 32 BITS PARA EL EVENTO
val randomString = UUID.randomUUID().toString().replace( oldValue: "-",  newValue: "")
val randomString32 = randomString.substring(0, 32)

val desc = if (description == "") { null } else { description }
val plac = if (place == "") { null } else { place }

val event = Event(randomString32, eventName,  eventsDatetime: "$date $time", desc, plac, usernameLogged)

val listSendToParticipantsString = userList.joinToString( separator: ",") { it.username }
val participantsString = "$listSendToParticipantsString,$usernameLogged"


//CREAR MENSAJES DE CREACION DE EVENTO EN FIREBASE Y EN CASO DE EXITO GUARDARLO LOCALMENTE

val functions = FirebaseFunctions.getInstance()

val data = hashMapOf(
    "event_id" to event.eventId,
    "name" to event.name,
    "event_datetime" to event.eventsDatetime,
    "description" to (event.description ?: " "),
    "location" to (event.location ?: " "),
    "participants" to participantsString,
    "list_send_to_participants" to listSendToParticipantsString
)

functions
    .getHttpsCallable( name: "createEvent") HttpsCallableReference
    .call(data) Task<HttpsCallableResult!>
    .addOnSuccessListener { result ->
        val message = result.data as? Map<*, *>
        val successMessage = message?.get("message") as? String
        Log.d( tag: "EVENTO",  msg: successMessage ?: "NO HAY")
        if (successMessage == "Exito") {
            viewModel.insertEvent(event)

            viewModel.insertEventUser(EventUser(eventId = randomString32, user = usernameLogged))
            for (user in userList) {
                val participant = EventUser(eventId = randomString32, user = user.username)
                viewModel.insertEventUser(participant)
```

Figura 4.18: Código de la aplicación para crear el evento

Tras ello, inicia la ejecución de la función en Cloud Functions. Primero se obtienen los datos enviados por la aplicación a partir de la variable ”data”. Posteriormente, se obtiene la lista de usuarios a los que enviar el mensaje y, para cada uno de ellos, se va generando un documento en la colección ”Events” de Firestore Database con los datos pertinentes. Finalmente, se comprueba que se haya generado un documento para cada usuario de la lista y se envían los mensajes a los dispositivos de cada uno (figuras 4.19 y 4.20).

```
exports.createEvent = functions.https.onCall(async (data, context) => {
  const usersRef = firestore.collection("users");

  const event_idValue = data.event_id;
  const nameValue = data.name;
  const event_datetimeValue = data.event_datetime;
  const descriptionValue = data.description;
  const locationValue = data.location;
  const participantsValue = data.participants;
  const listSendToParticipantsValue = data.list_send_to_participants;

  const listSendToParticipants = listSendToParticipantsValue.split(",");

  const docIds = [];
  const usernamePayloads = [];

  const insertEventPromises = listSendToParticipants.map(async (username) => {
    const userDoc = await usersRef.doc(username).get();
    const userToken = userDoc.get("token");

    const payload = {
      data: {
        type: "events",
        operation: "creation",
        event_to: username,
        event_id: event_idValue,
        name: nameValue,
        event_datetime: event_datetimeValue,
        description: descriptionValue,
        location: locationValue,
        owner: username,
        participants: participantsValue,
      },
    };

    const documentData = {
      type: "events",
      operation: "creation",
      event_to: username,
      event_id: event_idValue,
      name: nameValue,
      event_datetime: event_datetimeValue,
      description: descriptionValue,
      location: locationValue,
      owner: username,
      participants: participantsValue,
    };

    try {
      const docRef = await firestore.collection("events").add(documentData);
      const documentId = docRef.id;
      docIds.push(documentId);

      if (userToken) {
        payload.data.docId = documentId;
        usernamePayloads.push({ userToken: userToken, payload: payload });
      }
```

Figura 4.19: Primera parte del código de Cloud Functions para crear el evento

```
        payload.data.docId = documentId;
        usernamePayloads.push({ userToken: userToken, payload: payload });
      }
    } catch (error) {
      // Manejar errores aquí si es necesario
    }
  });

  await Promise.all(insertEventPromises);

  if (docIds.length == listSendToParticipants.length) {
    //ENVIAR LOS MENSAJES AL DISPOSITIVO DEL USUARIO
    for (const usernamePayload of usernamePayloads) {
      await admin
        .messaging()
        .sendToDevice(usernamePayload.userToken, usernamePayload.payload);
    }
    return { message: "Exito" };
  } else {
    //ELIMINAR EL RESTO DE DOCS QUE SI HAN TENIDO EXITO AL GUARDARSE
    for (const docId of docIds) {
      await firestore.collection("events").doc(docId).delete();
    }
    return { message: "Error" };
  }
});
```

Figura 4.20: Segunda parte del código de Cloud Functions para crear el evento

Para concluir, a todos los usuarios exceptuando el emisor se les ejecuta el método *onMessageReceived(message: RemoteMessage)* del servicio MessagingService al recibir el mensaje enviado por Messaging. Allí se filtra el mensaje en base al valor de los atributos "type" y "events" ejecutando así el código de la figura 4.21. En ella se obtienen los datos del evento y se ejecuta la función "deleteDocument" de Cloud Functions para eliminar el documento de la colección (figura 4.22). Una vez tiene éxito lo anterior se guarda el evento y los participantes de este en la base de datos de la aplicación móvil.

```
if (type == "events") {
    val operation = message.data["operation"]
    if (operation == "creation") {
        val eventId = message.data["event_id"]!!
        val name = message.data["name"]!!
        val eventDatetime = message.data["event_datetime"]!!
        val description = if (message.data["description"]!! == "") { null } else { message.data["description"]!! }
        val location = if (message.data["location"]!! == "") { null } else { message.data["location"]!! }
        val owner = message.data["owner"]!!
        val participants = message.data["participants"]!!
        val documentId = message.data["docId"]!!

        val participantsList = participants.split( ...delimiters: ",")

        val event = Event(eventId, name, eventDatetime, if (description == " ") null else description, if (location == " ") null else location, owner)

        serviceScope.launch { this: CoroutineScope
            withContext(Dispatchers.IO) { this: CoroutineScope

                //ELIMINAR DOCUMENTO DE FIRESTORE
                val functions = FirebaseFunctions.getInstance()
                val data = hashMapOf(
                    "documentId" to documentId,
                    "collectionName" to "events"
                )
                val result = functions
                    .getHttpsCallable( name: "deleteDocument") HttpsCallableReference
                    .call(data) Task<HttpsCallableResult!>
                    .await()

                val message = result.data as? Map<*, *>
                val successMessage = message?.get("message") as? String
                if (successMessage == "Exito") {
                    //INSERTAR EVENTO EN BBDD
                    messagingServiceRepository.insertEvent(event)
                    for (participant in participantsList) {
                        //INSERTAR PARTICIPANTE EN BBDD
                        messagingServiceRepository.insertEventUser(EventUser(eventId = event.eventId, user = participant))
                    }
                }
```

Figura 4.21: Código del servicio para crear el evento

```
exports.deleteDocument = functions.https.onCall((data, context) => {
  const documentId = data.documentId;
  const collectionName = data.collectionName;

  const documentRef = firestore.collection(collectionName).doc(documentId);

  return documentRef
    .delete()
    .then(() => {
      // El documento se ha eliminado correctamente
      return { message: "Exito" };
    })
    .catch((error) => {
      // Error al eliminar el documento
      return { message: "Error" };
    });
});
```

Figura 4.22: Código de Cloud Functions para eliminar un documento de una colección

CAPÍTULO 5
Resultados

Una vez finalizada la implementación de la aplicación es necesario mostrar el resultado del producto final con el objetivo de comprobar que todo opere como es debido. Para ello, se ejemplificará su uso mediante el relato de una historia.

Alice le comenta a Bob que no termina de estar contenta con la aplicación de mensajería que utiliza normalmente y Bob le pregunta si conoce Socialme. Alice se interesa por la aplicación ya que incluye una funcionalidad de traducción de mensajes. Por lo tanto, decide instalársela.

Una vez la inicia, le sale una pantalla para introducir credenciales (figura 5.1). Como Alice no tiene ninguna cuenta creada pulsa sobre el texto ”Crear cuenta” y la aplicación la redirige a una pantalla en la cual debe ingresar datos para crearse una nueva cuenta (figura 5.2). Una vez realizado lo anterior, la aplicación le indica que debe verificar el correo electrónico. Alice ingresa en su correo y clica sobre el enlace eligiendo abrirlo con Socialme (figuras 5.3 y 5.4). Tras realizar aquello, Socialme le informa del éxito al verificar el correo (figura 5.5), Alice introduce sus credenciales y pulsa sobre el botón ”Iniciar sesión”.

Figura 5.1: Alice inicia la aplicación

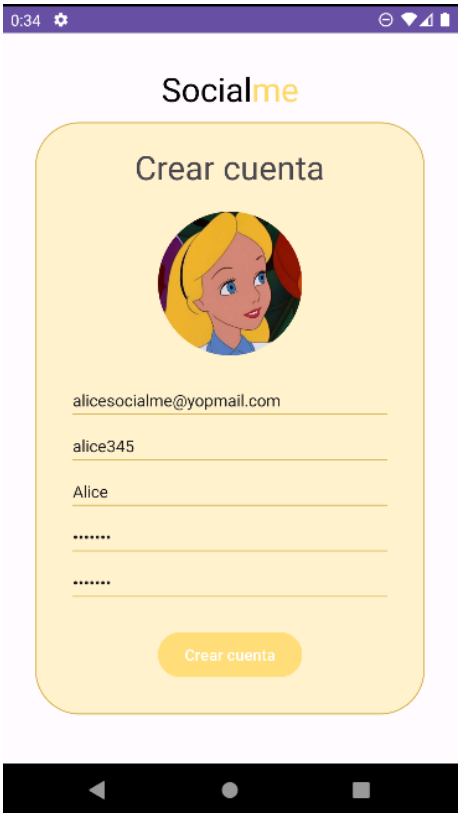

Figura 5.2: Pantalla de crear cuenta con los datos de Alice

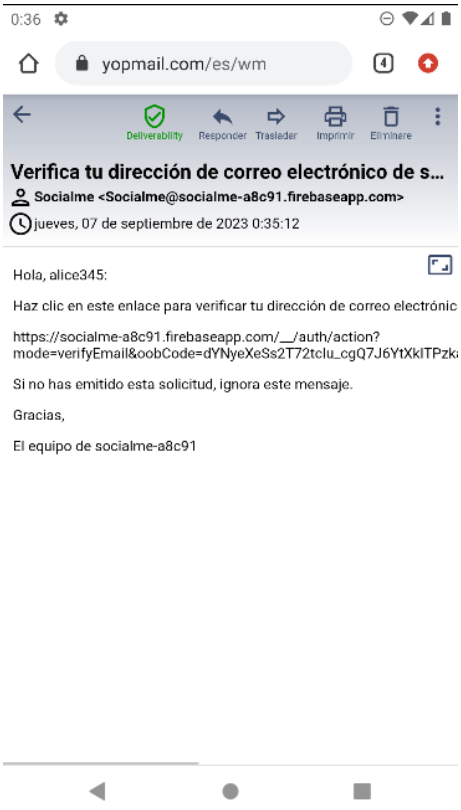

Figura 5.3: Email para verificar el correo electrónico

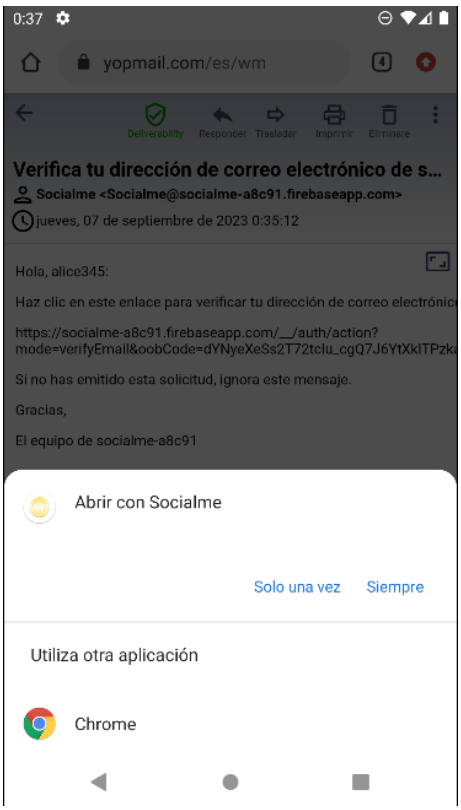

Figura 5.4: Alice selecciona abrir el enlace con Socialme

Figura 5.5: La aplicación informa a Alice del éxito de la verificación del correo

A continuación, la aplicación redirige a Alice a la pantalla principal (figura 5.6). Allí decide dirigirse al apartado "SMS" desencadenando que se analicen todos los SMS de su dispositivo con el propósito de determinar su nivel de peligrosidad (figura 5.7). Posteriormente, decide buscar a Bob en la aplicación pulsando sobre el botón circular flotante de la figura 5.6 llevándola a la pantalla de la imagen 5.8. Luego, Alice pulsa sobre la lupa y teclea el nombre de usuario de Bob (que anteriormente se lo había comentado) como se puede apreciar en la figura 5.9 con el objetivo de añadirle a favoritos (figura 5.10).

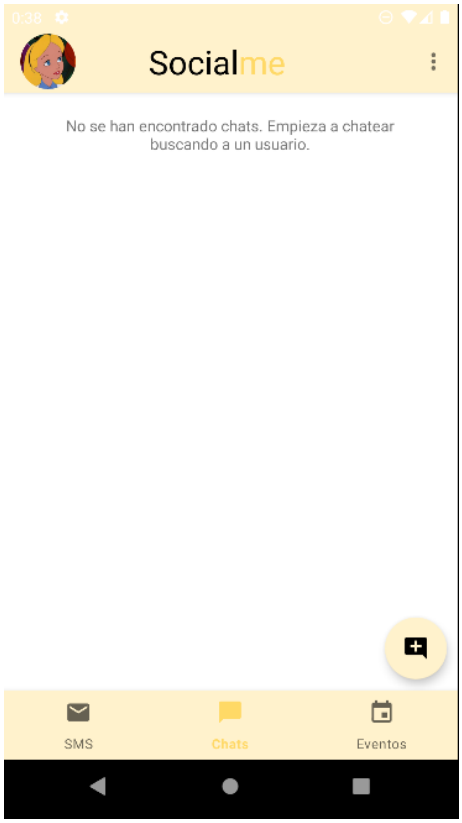

Figura 5.6: Pantalla principal de Socialme

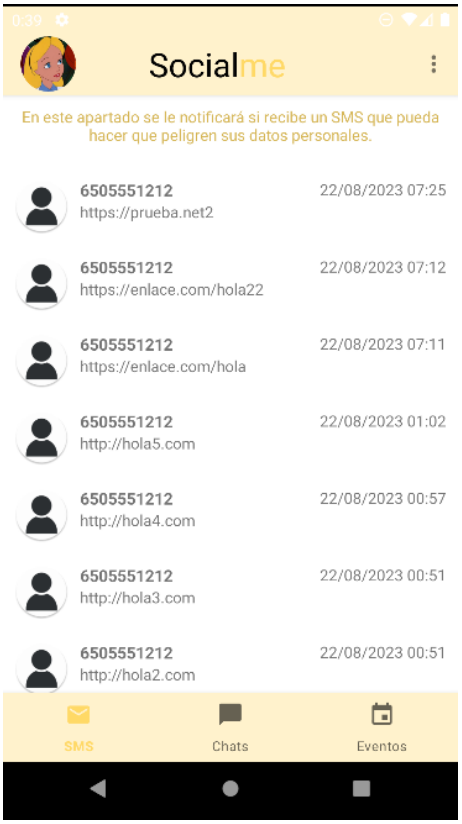

Figura 5.7: SMS analizados y detectados como peligrosos

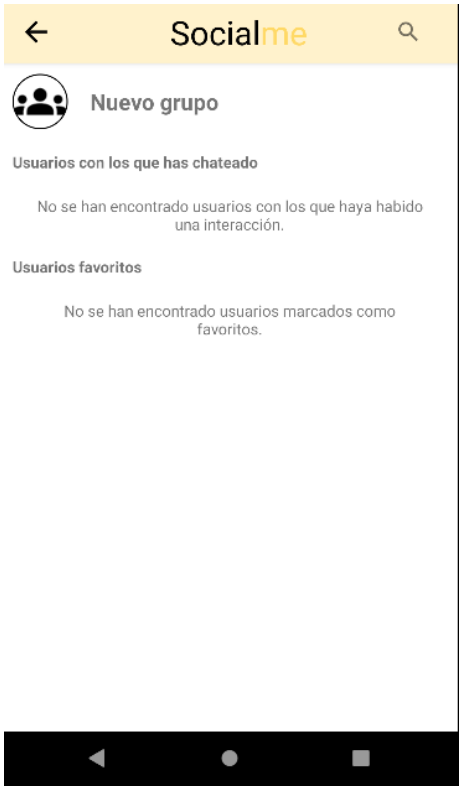

Figura 5.8: Pantalla de nuevo chat de la aplicación

Figura 5.9: Alice busca a Bob

Figura 5.10: Alice añade a Bob a favoritos

Seguidamente, Alice accede al chat con Bob y le envía un mensaje (figura 5.11). Tres minutos más tarde le llega una respuesta de Bob tal y como se puede ver en la figura 5.12. A Alice le gusta la foto de perfil de Bob y clica sobre la misma accediendo así a la pantalla de la figura 5.13. Posteriormente, quiere probar la función de traducción así que elige un idioma (el italiano) y escribe un mensaje (figura 5.14) para luego pulsar el icono de traducir y obtener el mensaje traducido (figura 5.15).

Figura 5.11: Alice le envía un mensaje a Bob

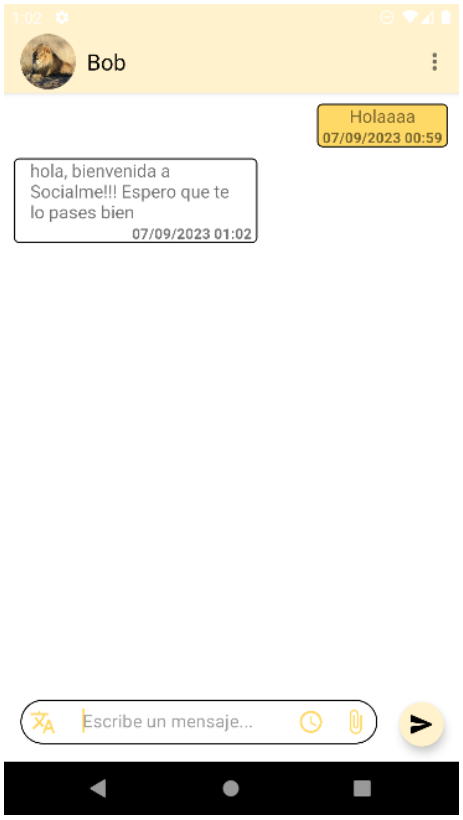

Figura 5.12: Bob responde a Alice

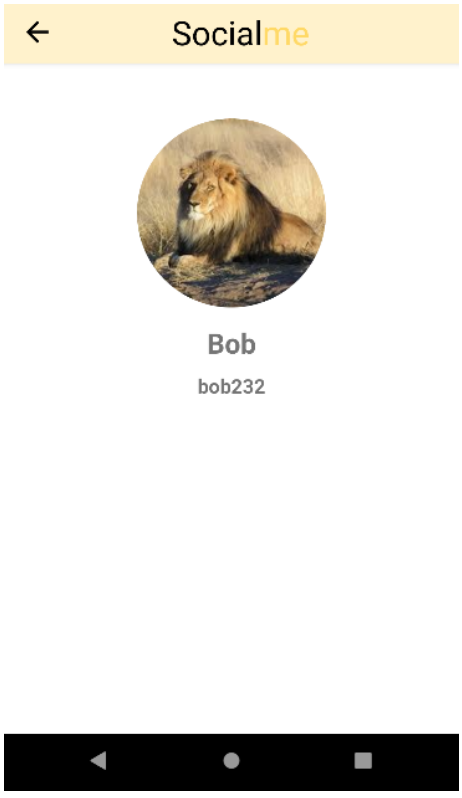

Figura 5.13: Alice selecciona la opción de ver más información sobre Bob

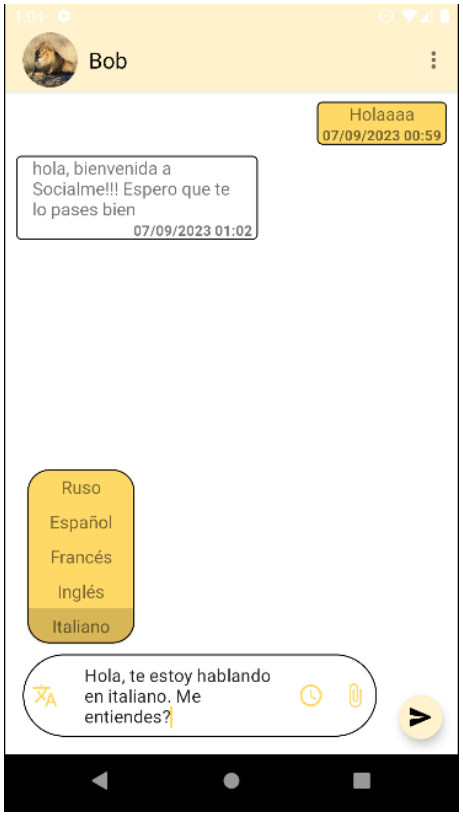

Figura 5.14: Alice selecciona un idioma y escribe un mensaje para que se traduzca

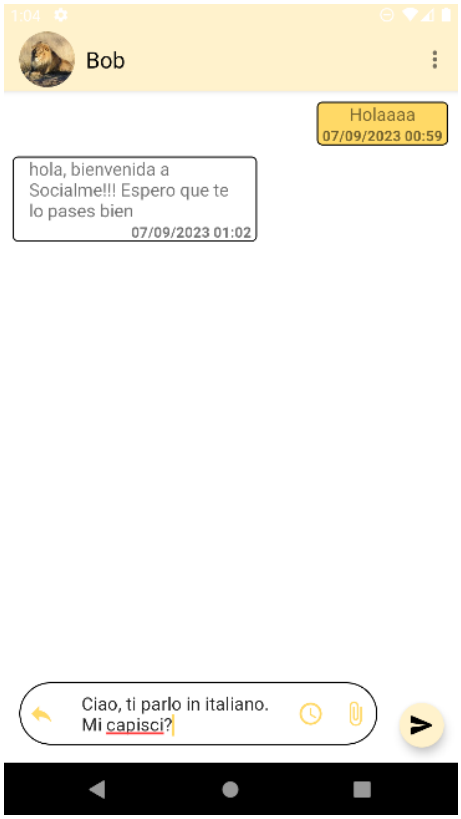

Figura 5.15: Se traduce el mensaje al idioma seleccionado

Ahora Alice quiere ver las otras opciones que ofrece Socialme, por eso accede a la pantalla de ”Eventos” (figura 5.16). Una vez allí, clica sobre la opción de ”Crear nuevo evento” y la aplicación la redirige a una pantalla para que seleccione a las usuarios que desea añadir al evento (figura 5.17). Alice elige a Bob y completa el resto de los datos del evento (figura 5.18). Tras pulsar sobre el botón flotante la aplicación la devuelve a la pantalla de ”Eventos” pudiendo ver el evento recién creado (figura 5.19) y pulsándolo para acceder a él (figura 5.20).

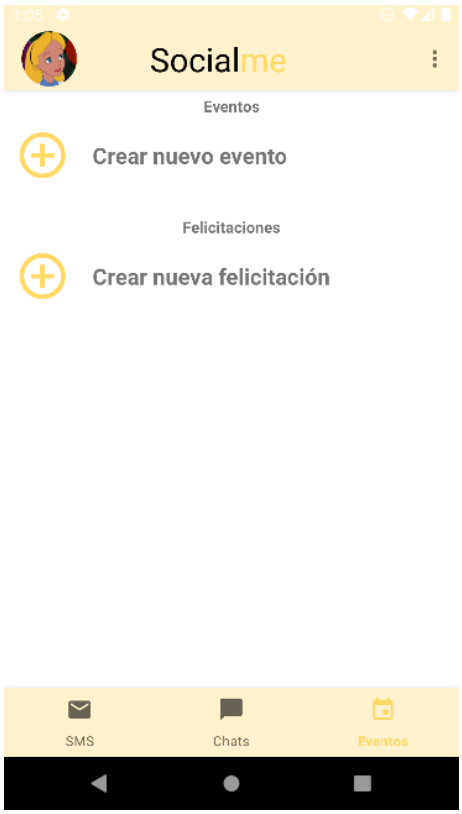

Figura 5.16: Alice accede al apartado "Eventos"

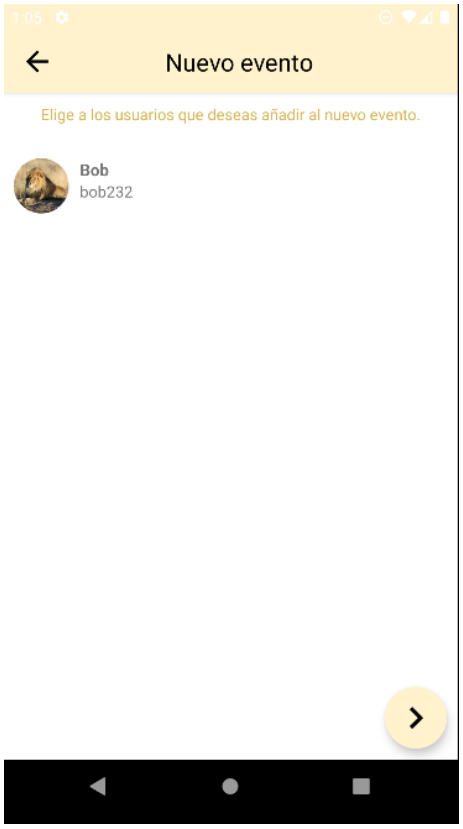

Figura 5.17: La aplicación insta a Alice a seleccionar usuarios

Figura 5.18: Alice completa los datos relativos al evento

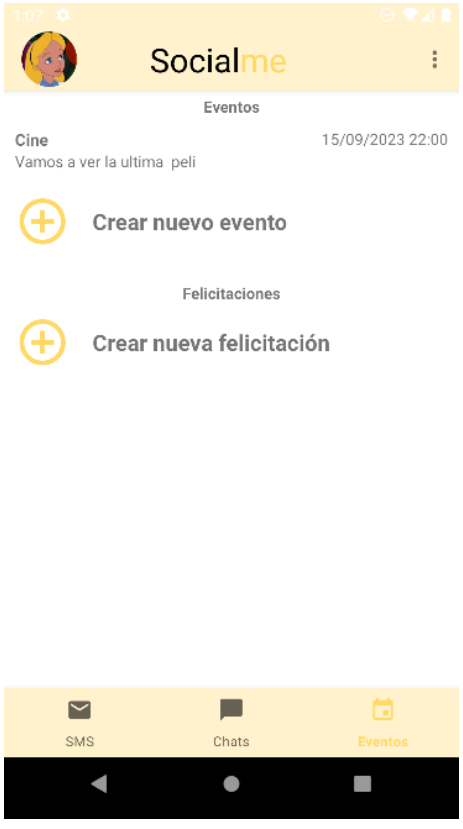

Figura 5.19: Aparece listado el evento recién creado

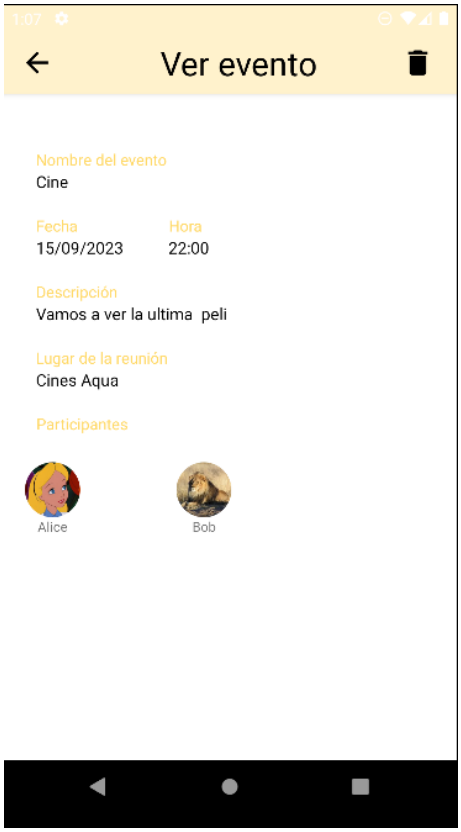

Figura 5.20: Acceso a los datos del evento recién creado

Finalmente, Alice desea cerrar sesión así que clica sobre los tres puntos verticales, selecciona la opción de ”Cerrar sesión” tal y como se puede ver en la figura 5.21 y confirma la operación (figura 5.22).

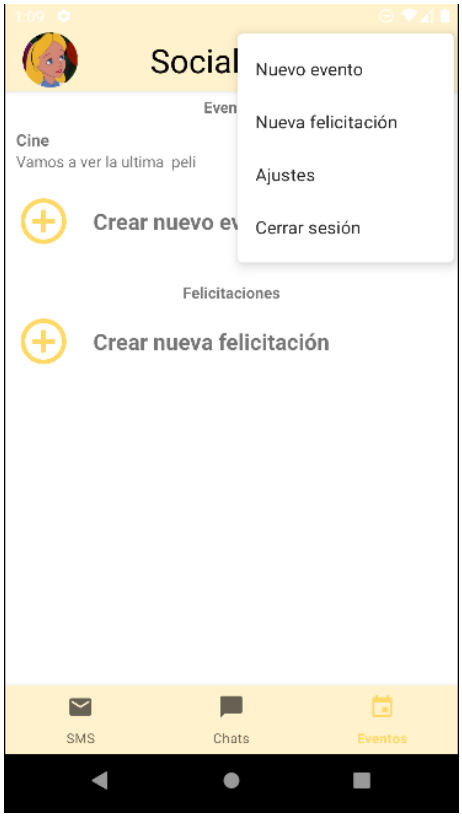

Figura 5.21: La aplicación ofrece varias opciones a Alice

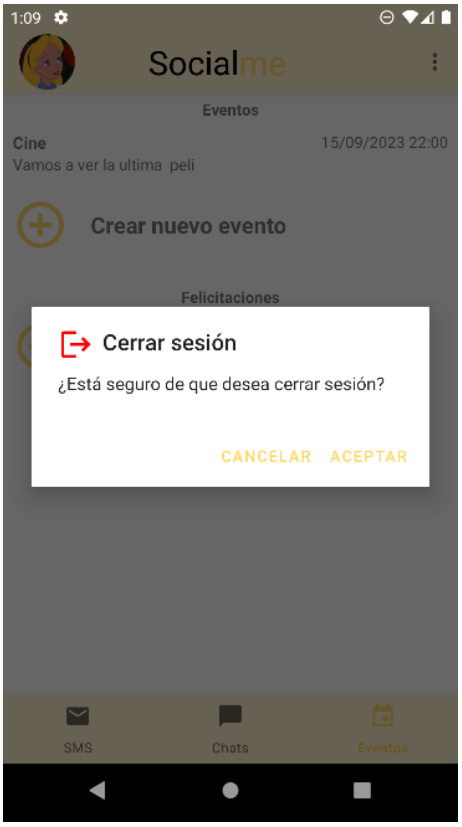

Figura 5.22: Alice solicita cerrar sesión

CAPÍTULO 6
Conclusiones

Siempre a la hora de concluir un proyecto es necesario revisar si se han cumplido los objetivos definidos al principio. En este caso, se ha logrado implementar una aplicación móvil Android que intercambia información con un servidor *backend* que ofrece los 5 servicios descritos inicialmente.

Cabe mencionar que la creación de esta aplicación móvil ha sido un viaje apasionante y desafiante al mismo tiempo en la que ha hecho falta salir de la zona de confort para poder afrontar las nuevas tecnologías y los problemas que han ido surgiendo. Es importante mencionar que en el mundo de la programación es necesario perseverar para poder llegar a cumplir el objetivo establecido.

Por último, además de aprender nuevas tecnologías, se ha aprendido una metodología de trabajo que resultará útil en los proyectos futuros.

Bibliografía

[1] Wikipedia. Historia de internet, 2023. Disponible en https://es.wikipedia.org/wiki/Historia_de_Internet, consultada el 30/08/2023.

[2] Tania Wilches. Impacto de la mensajería instantánea en la vida. Technical report, 2023. Disponible en https://www.b2chat.io/blog/mensajeria-instantanea/mensajeria-instantanea-impacto-vida/.

[3] Wikipedia. Kotlin (lenguaje de programación), 2023. Disponible en https://es.wikipedia.org/wiki/Kotlin_(lenguaje_de_programaci%C3%B3n), consultada el 15/06/2023.

[4] Wikipedia. Firebase, 2023. Disponible en https://es.wikipedia.org/wiki/Firebase, consultada el 16/06/2023.

[5] Android Developers. Aspectos fundamentales de la aplicación, 2023. Disponible en https://developer.android.com/guide/components/fundamentals?hl=es-419, consultada el 27/08/2023.

[6] Android Developers. Descripción general de proyectos, 2023. Disponible en https://developer.android.com/studio/projects?hl=es-419, consultada el 27/08/2023.

[7] Android Developers. Inserción de dependencias con hilt, 2023. Disponible en https://developer.android.com/training/dependency-injection/hilt-android?hl=es-419, consultada el 29/08/2023.

[8] Android Developers. Cómo guardar contenido en una base de datos local con room, 2023. Disponible en https://developer.android.com/training/data-storage/room?hl=es-419, consultada el 31/08/2023.

[9] Google. Primeros pasos: Escribe, prueba e implementa tus primeras funciones, 2023. Disponible en https://firebase.google.com/docs/functions/get-started?hl=es&authuser=0&gen=2nd, consultada el 30/06/2023.

Anexos

En este apartado se presentará información adicional y complementaria que respalda y enriquece el contenido principal del documento. Estos datos y materiales adicionales están diseñados para proporcionar a los lectores una visión más completa y detallada sobre ciertos aspectos relacionados con el proyecto que no se incluyeron en el cuerpo principal del documento para no interrumpir su flujo.

6.1 Descripción detallada de los casos de uso

6.1.1. Usuario no autenticado

Tabla 6.1: Descripción del caso de uso - Iniciar sesión

Caso de Uso	Iniciar sesión
Actores	Usuario no autenticado
Resumen	El usuario inicia la aplicación e introduce sus credenciales (correo electrónico y contraseña). El sistema comprueba su validez y gestiona la sesión creada. Al acabar, el usuario dispone de una sesión.
Precondiciones	-
Postcondiciones	El usuario recibe el token de la sesión iniciada
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario inicia la aplicación 2. El sistema solicita al usuario los siguientes datos: correo electrónico y contraseña 3. El usuario ingresa los datos requeridos 4. El sistema comprueba la veracidad de los datos recibidos, crea un token de sesión y se lo asigna al usuario	
Excepciones en el flujo de eventos	
#1. Si en 3 el usuario no introduce algún campo requerido el sistema genera un mensaje de error #2. Si en 4 los datos recibidos son erróneos el sistema genera un mensaje de error	

Tabla 6.2: Descripción del caso de uso - Crear nueva cuenta

Caso de Uso	Crear nueva cuenta
Actores	Usuario no autenticado
Resumen	El usuario crea las credenciales necesarias para poder iniciar sesión. El sistema comprueba si cumplen todos los requisitos y las almacena. Al acabar, el usuario dispone de una cuenta.
Precondiciones	-
Postcondiciones	La nueva cuenta se registra en el sistema
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario indica que desea crear una nueva cuenta 2. El sistema solicita al usuario los siguientes datos: correo electrónico, nombre de usuario, nombre completo, contraseña, confirmar la contraseña y foto de perfil (opcional) 3. El usuario introduce los datos solicitados 4. El sistema comprueba si los datos recibidos cumplen todos los requisitos (longitud, número, ...), almacena la nueva cuenta y genera un mensaje informando al usuario del éxito	
Excepciones en el flujo de eventos	
#1. Si en 3 el usuario no introduce algún campo requerido o al verificar la contraseña ésta es diferente de la introducida el sistema genera un mensaje de error #2. Si en 4 los datos recibidos no cumplen algún requisito el sistema genera un mensaje de error	

Tabla 6.3: Descripción del caso de uso - Cambiar contraseña

Caso de Uso	Cambiar contraseña
Actores	Usuario no autenticado
Resumen	El usuario solicita cambiar la contraseña de una cuenta. Para ello, anteriormente verifica ser poseedor de esta mediante un desafío. El sistema comprueba si cumple todos los requisitos y la almacena. Al acabar, la cuenta del usuario dispone de una nueva contraseña.
Precondiciones	La existencia del usuario del que se quiere modificar la contraseña y un correo electrónico vinculado a éste
Postcondiciones	La nueva contraseña sustituye a la antigua en el sistema
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario indica que se ha olvidado de la contraseña de su cuenta 2. El sistema solicita al usuario su correo electrónico 3. El usuario introduce el correo 4. El sistema genera un desafío para comprobar si el usuario efectivamente es el dueño de la cuenta y le informa al mismo la forma de completarlo 5. El usuario completa el desafío 6. El sistema comprueba la compleción del desafío y le solicita al usuario los siguientes datos: nueva contraseña y verificar nueva contraseña 7. El usuario introduce los datos solicitados 8. El sistema comprueba si los datos recibidos cumplen todos los requisitos (longitud, número, ...), almacena la nueva contraseña y genera un mensaje informando al usuario del éxito	
Excepciones en el flujo de eventos	
#1. Si en 6 no se ha superado el desafío el sistema genera un mensaje de error #2. Si en 8 al verificar la contraseña ésta es diferente de la introducida el sistema genera un mensaje de error #3. Si en 8 los datos recibidos no cumplen algún requisito el sistema genera un mensaje de error	

6.1.2. Usuario autenticado

Tabla 6.4: Descripción del caso de uso - Ver lista de últimos chats

Caso de Uso	Ver lista de últimos chats
Actores	Usuario autenticado
Resumen	El usuario solicita ver las últimas conversaciones que ha tenido. El sistema le muestra una lista con las mismas.
Precondiciones	-
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario ha realizado alguna de las siguientes acciones: ha iniciado la aplicación (su sesión sigue activa), ha iniciado sesión o ha seleccionado la opción de "Chats" (si ya se encontraba identificado y en otro caso de uso) 2. El sistema muestra una lista con las últimas conversaciones mantenidas por el usuario	
Excepciones en el flujo de eventos	
-	

Tabla 6.5: Descripción del caso de uso - Ver un chat

Caso de Uso	Ver un chat
Actores	Usuario autenticado
Resumen	El usuario solicita ver los mensajes que ha intercambiado con otro usuario o grupo. El sistema le muestra los últimos mensajes enviados por ambas partes.
Precondiciones	-
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario solicita ver los mensajes intercambiados con otro usuario o grupo. Para ello, selecciona un chat existente o un usuario 2. El sistema le muestra los últimos mensajes enviados y recibidos con el otro usuario o grupo	
Excepciones en el flujo de eventos	
-	

Tabla 6.6: Descripción del caso de uso - Enviar mensaje

Caso de Uso	Enviar mensaje
Actores	Usuario autenticado
Resumen	El usuario redacta un mensaje para otro usuario o grupo. El sistema lo envía al destinatario especificado.
Precondiciones	El mensaje ha de tener como mínimo 1 carácter
Postcondiciones	El mensaje ha sido enviado al usuario o grupo destinatarios
Incluye	-
Extiende	Ver un chat
Hereda de	-
Flujo de Eventos	
1. El usuario introduce el mensaje que desea enviar 2. El sistema almacena el mensaje, lo envía al usuario o grupo destinatarios e informa al usuario del éxito del envío	
Excepciones en el flujo de eventos	
#1. Si en 2 el sistema no ha conseguido hacer llegar el mensaje al destinatario esperará a que el mismo esté disponible	

Tabla 6.7: Descripción del caso de uso - Enviar archivo

Caso de Uso	Enviar archivo
Actores	Usuario autenticado
Resumen	El usuario solicita enviar un archivo a otro usuario o grupo. El sistema lo envía al destinatario especificado.
Precondiciones	-
Postcondiciones	El archivo ha sido enviado al usuario o grupo destinatarios
Incluye	-
Extiende	Enviar mensaje
Hereda de	-
Flujo de Eventos	
1. El usuario le indica al sistema que desea enviar un archivo 2. El sistema le solicita que seleccione el archivo que desea remitir 3. El usuario elige el archivo a enviar 4. El sistema almacena el archivo, lo envía al usuario o grupo destinatarios e informa al usuario del éxito del envío	
Excepciones en el flujo de eventos	
#1. Si en 4 el sistema no ha conseguido hacer llegar el archivo al destinatario esperará a que el mismo esté disponible	

Tabla 6.8: Descripción del caso de uso - Traducir mensaje

Caso de Uso	Traducir mensaje
Actores	Usuario autenticado
Resumen	El usuario solicita la traducción del mensaje que ha introducido. El sistema lo traduce.
Precondiciones	El usuario ha de haber introducido un mensaje para el destinatario
Postcondiciones	El mensaje introducido por el usuario ha sido traducido a uno de los idiomas disponibles
Incluye	-
Extiende	Enviar mensaje
Hereda de	-
Flujo de Eventos	
1. El usuario solicita al sistema traducir el mensaje introducido a uno de los idiomas disponibles 2. El sistema traduce el mensaje al idioma solicitado y reemplaza el mensaje introducido por el usuario por el mensaje traducido	
Excepciones en el flujo de eventos	
-	

Tabla 6.9: Descripción del caso de uso - Programar envío

Caso de Uso	Programar envío
Actores	Usuario autenticado
Resumen	El usuario solicita programar el envío de un mensaje para una fecha y hora concretas. El sistema lo almacena y lo hará llegar al destinatario en el momento pertinente.
Precondiciones	El usuario ha de haber introducido un mensaje para el destinatario
Postcondiciones	El mensaje introducido ha sido almacenado en el sistema para su posterior envío en la fecha determinada
Incluye	-
Extiende	Enviar mensaje
Hereda de	-
Flujo de Eventos	
1. El usuario solicita al sistema programar para una fecha y hora concretas el envío del mensaje introducido 2. El sistema solicita al usuario los siguientes datos: fecha y hora 3. El usuario introduce los datos pedidos 4. El sistema almacena el mensaje e informa al usuario del éxito al programar el mismo. Una vez se alcance la fecha de entrega, éste remitirá el mensaje al destinario pertinente cuando éste esté disponible	
Excepciones en el flujo de eventos	
-	

Tabla 6.10: Descripción del caso de uso - Ver más información sobre el chat

Caso de Uso	Ver más información sobre el chat
Actores	Usuario autenticado
Resumen	El usuario solicita ver más detalles sobre el usuario o grupo con el que está conversando. El sistema le remite dicha información.
Precondiciones	-
Postcondiciones	-
Incluye	-
Extiende	Ver un chat
Hereda de	-
Flujo de Eventos	
1. El usuario indica al sistema que desea ver más información sobre el usuario o grupo con el que está intercambiando mensajes 2. El sistema le muestra datos detallados sobre el usuario o grupo pedido	
Excepciones en el flujo de eventos	
-	

Tabla 6.11: Descripción del caso de uso - Eliminar participante

Caso de Uso	Eliminar participante
Actores	Usuario autenticado
Resumen	El usuario selecciona a un participante para eliminarlo del grupo. El sistema lo elimina y le notifica de su expulsión.
Precondiciones	El chat se ha de tratar de un grupo
Postcondiciones	Se ha eliminado el participante del grupo deseado
Incluye	-
Extiende	Ver más información sobre el chat
Hereda de	-
Flujo de Eventos	
1. El usuario indica al sistema que desea eliminar a un participante del grupo 2. El sistema le solicita que seleccione al usuario del grupo que desea expulsar 3. El usuario selecciona al usuario 4. El sistema elimina del grupo al usuario indicado y notifica a éste de su expulsión	
Excepciones en el flujo de eventos	
#1. Si en 4 el usuario expulsado no está disponible el sistema se lo notificará cuando lo esté	

Tabla 6.12: Descripción del caso de uso - Crear nuevo chat

Caso de Uso	Crear nuevo chat
Actores	Usuario autenticado
Resumen	El usuario desea iniciar una nueva conversación. El sistema le muestra todos los contactos que tiene agregados como favoritos. El usuario elije uno de ellos.
Precondiciones	-
Postcondiciones	-
Incluye	Listar usuarios favoritos
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario le indica al sistema que desea crear un nuevo chat 2. El sistema le muestra una lista con los usuarios que ha guardado como "favoritos" (caso de uso "Listar usuarios favoritos") y le solicita que elija a uno de ellos con el que establecer comunicación 3. El usuario selecciona a ese otro usuario 4. El sistema inicia el caso de uso "Ver un chat"	
Excepciones en el flujo de eventos	
-	

Tabla 6.13: Descripción del caso de uso - Buscar usuario

Caso de Uso	Buscar usuario
Actores	Usuario autenticado
Resumen	El usuario desea buscar a otro usuario empleando para ello el nombre de usuario de este último. El sistema le remite las coincidencias.
Precondiciones	-
Postcondiciones	El usuario obtiene un listado con todos los usuarios existentes que incluyen la cadena indicada
Incluye	-
Extiende	Crear nuevo chat
Hereda de	-
Flujo de Eventos	
1. El usuario solicita al sistema que desea buscar a un usuario 2. El sistema le pide que introduzca el nombre de usuario del contacto que está buscando 3. El usuario introduce el dato solicitado 4. El sistema busca al usuario que tenga como nombre de usuario el dato recibido y le muestra una lista al usuario con todos los usuarios que tienen o contienen el nombre de usuario especificado anteriormente	
Excepciones en el flujo de eventos	
-	

Tabla 6.14: Descripción del caso de uso - Guardar usuario como favorito

Caso de Uso	Guardar usuario como favorito
Actores	Usuario autenticado
Resumen	El usuario solicita guardar a otro usuario como "favorito". El sistema lo almacena.
Precondiciones	El usuario que se desea guardar no ha de estar ya guardado
Postcondiciones	Se ha almacenado la información del usuario deseado
Incluye	-
Extiende	Buscar usuario
Hereda de	-
Flujo de Eventos	
1. El usuario solicita al sistema guardar el usuario que se ha buscado 2. El sistema almacena los datos de este	
Excepciones en el flujo de eventos	
-	

Tabla 6.15: Descripción del caso de uso - Crear nuevo grupo

Caso de Uso	Crear nuevo grupo
Actores	Usuario autenticado
Resumen	El usuario desea crear un grupo. El sistema le muestra todos los contactos que tiene agregados. El usuario elije uno o varios de ellos y le pone un nombre al grupo. El sistema crea el grupo.
Precondiciones	-
Postcondiciones	Se crea el nuevo grupo con los miembros seleccionados
Incluye	Listar usuarios favoritos
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario le indica al sistema que desea crear un nuevo grupo 2. El sistema le muestra una lista con los usuarios que ha guardado como "favoritos" (caso de uso "Listar usuarios favoritos") y le solicita que elija a uno o más de ellos a los que añadir al grupo 3. El usuario selecciona al usuario/usuarios 4. El sistema solicita al usuario que indique los siguientes datos: nombre del grupo e imagen (opcional) 5. El usuario introduce los datos pedidos 6. El sistema crea el grupo y notifica a todos los usuarios seleccionados que han sido añadidos a ese grupo 7. El sistema inicia el caso de uso "Ver un chat"	
Excepciones en el flujo de eventos	
#1. Si en 3 el usuario no selecciona a nadie el sistema no le permitirá continuar con el caso de uso hasta que lo haga #2. Si en 6 algún usuario no está disponible, el sistema le notificará cuando lo esté	

Tabla 6.16: Descripción del caso de uso - Listar usuarios favoritos

Caso de Uso	Listar usuarios favoritos
Actores	Usuario autenticado
Resumen	El usuario quiere ver los usuarios que tiene guardados como favoritos. El sistema se los devuelve en forma de listado.
Precondiciones	-
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario indica al sistema que desea ver todos los usuarios que tiene agregados como favoritos 2. El sistema le devuelve un listado con todos los usuarios que tiene agregados	
Excepciones en el flujo de eventos	
-	

Tabla 6.17: Descripción del caso de uso - Eliminar usuario de favoritos

Caso de Uso	Eliminar usuario de favoritos
Actores	Usuario autenticado
Resumen	El usuario solicita eliminar a un usuario de "favoritos". El sistema lo elimina.
Precondiciones	-
Postcondiciones	Se ha eliminado la información del usuario indicado
Incluye	-
Extiende	Listar usuarios favoritos, Buscar usuario
Hereda de	-
Flujo de Eventos	
1. El usuario solicita al sistema eliminar el usuario seleccionado de "favoritos" 2. El sistema elimina los datos de este	
Excepciones en el flujo de eventos	
-	

Tabla 6.18: Descripción del caso de uso - Ver eventos y felicitaciones

Caso de Uso	Ver eventos y felicitaciones
Actores	Usuario autenticado
Resumen	El usuario solicita ver los eventos en los que está incluido y las felicitaciones que ha creado. El sistema se los muestra en forma de listado.
Precondiciones	-
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario indica al sistema que desea ver todos los eventos en los que esté incluido y todas las felicitaciones que ha creado 2. El sistema le muestra dos listados con la información solicitada	
Excepciones en el flujo de eventos	
-	

Tabla 6.19: Descripción del caso de uso - Crear evento

Caso de Uso	Crear evento
Actores	Usuario autenticado
Resumen	El usuario introduce los datos relativos al evento. El sistema lo almacena y envía a todos los usuarios partícipes.
Precondiciones	-
Postcondiciones	Se crea el evento deseado
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario le indica al sistema que desea crear un nuevo evento 2. El sistema le solicita que introduzca los siguientes datos: nombre del evento, participantes, fecha y hora, descripción (opcional) y lugar (opcional) 3. El usuario introduce los datos solicitados 4. El sistema almacena el evento y se lo envía al usuario o usuarios participantes cuando estén disponibles	
Excepciones en el flujo de eventos	
-	

Tabla 6.20: Descripción del caso de uso - Crear felicitación

Caso de Uso	Crear felicitación
Actores	Usuario autenticado
Resumen	El usuario introduce los datos relativos a la felicitación. El sistema la almacena y la hará llegar al destinatario en el momento pertinente.
Precondiciones	-
Postcondiciones	Se crea la felicitación deseada
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario le indica al sistema que desea crear una felicitación 2. El sistema le solicita que introduzca los siguientes datos: mensaje, destinatario, fecha y hora de entrega y archivo (opcional) 3. El usuario introduce los datos solicitados 4. El sistema almacena la felicitación. Una vez se alcance la fecha de entrega, éste remitirá el mensaje al destinario pertinente cuando éste esté disponible	
Excepciones en el flujo de eventos	
-	

Tabla 6.21: Descripción del caso de uso - Eliminar evento

Caso de Uso	Eliminar evento
Actores	Usuario autenticado
Resumen	El usuario solicita eliminar un evento. El sistema lo elimina y notifica a los partícipes de éste.
Precondiciones	-
Postcondiciones	El evento se ha eliminado
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario le indica al sistema que desea eliminar el evento seleccionado 2. El sistema elimina el evento y se lo notifica al usuario o usuarios participantes cuando estén disponibles	
Excepciones en el flujo de eventos	
-	

Tabla 6.22: Descripción del caso de uso - Eliminar felicitación

Caso de Uso	Eliminar felicitación
Actores	Usuario autenticado
Resumen	El usuario solicita eliminar una felicitación. El sistema la elimina.
Precondiciones	-
Postcondiciones	La felicitación se ha eliminado
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario le indica al sistema que desea eliminar la felicitación seleccionada 2. El sistema elimina la felicitación.	
Excepciones en el flujo de eventos	
-	

Tabla 6.23: Descripción del caso de uso - Ver evento

Caso de Uso	Ver evento
Actores	Usuario autenticado
Resumen	El usuario solicita ver más detalles sobre un evento. El sistema se los muestra.
Precondiciones	-
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario le indica al sistema que desea ver más datos sobre un evento en concreto 2. El sistema le muestra una pantalla con todos los datos pedidos	
Excepciones en el flujo de eventos	
-	

Tabla 6.24: Descripción del caso de uso - Ver felicitación

Caso de Uso	Ver felicitación
Actores	Usuario autenticado
Resumen	El usuario solicita ver más detalles sobre una felicitación. El sistema se los muestra.
Precondiciones	-
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario le indica al sistema que desea ver más datos sobre una felicitación en concreto 2. El sistema le muestra una pantalla con todos los datos pedidos	
Excepciones en el flujo de eventos	
-	

Tabla 6.25: Descripción del caso de uso - Cerrar sesión

Caso de Uso	Cerrar sesión
Actores	Usuario autenticado
Resumen	El usuario solicita cerrar la sesión. El sistema elimina el token de la sesión.
Precondiciones	El usuario ha de tener una sesión abierta
Postcondiciones	El token de la sesión se anula en el sistema
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario indica que desea cerrar sesión 2. El sistema anula el token de la sesión del usuario y genera un mensaje informando al mismo del éxito	
Excepciones en el flujo de eventos	
-	

Tabla 6.26: Descripción del caso de uso - Revisar SMS con phishing

Caso de Uso	Revisar SMS con phishing
Actores	Usuario autenticado
Resumen	El usuario solicita ver los SMS con posible phishing que ha recibido. El sistema se los lista.
Precondiciones	-
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario solicita ver los SMS que puedan ser en realidad un ataque de ingeniería social (phishing) 2. El sistema le muestra una lista con los SMS que hayan sido detectados como peligrosos	
Excepciones en el flujo de eventos	
-	

Tabla 6.27: Descripción del caso de uso - Ver más detalles de un SMS con phishing

Caso de Uso	Ver más detalles de un SMS con phishing
Actores	Usuario autenticado
Resumen	El usuario solicita ver con más detalle un SMS con posible phishing. El sistema le muestra el SMS completo.
Precondiciones	-
Postcondiciones	-
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario le indica al sistema que desea ver con más detalle un SMS con posible phishing 2. El sistema le muestra el SMS completo	
Excepciones en el flujo de eventos	
-	

Tabla 6.28: Descripción del caso de uso - Modificar ajustes de la cuenta

Caso de Uso	Modificar ajustes de la cuenta
Actores	Usuario autenticado
Resumen	El usuario modifica el valor de alguna variable de su cuenta. El sistema almacena los cambios.
Precondiciones	-
Postcondiciones	Los ajustes modificados han sido almacenados en el sistema
Incluye	-
Extiende	-
Hereda de	-
Flujo de Eventos	
1. El usuario solicita modificar ajustes de la cuenta 2. El sistema le muestra al usuario las variables que puede modificar: nombre completo, foto de perfil y contraseña 3. El usuario elige la variable que desea modificar e introduce su nuevo valor 4. El sistema recupera el nuevo valor, comprueba que la variable recibida cumpla con todos los requisitos, la almacena sustituyendo el viejo valor y genera un mensaje de éxito	
Excepciones en el flujo de eventos	
#1. Si en 4 los datos recibidos no cumplen algún requisito el sistema genera un mensaje de error	

6.2 Descripción a detalle de los diagramas de bases de datos

El atributo ”owner” de todas las tablas (exceptuando la tabla ”SMS”, ”GroupsUsers” y ”EventsUsers”): Indica a qué usuario autenticado en la aplicación pertenecen esos datos ya que la misma permite iniciar sesión como varios usuarios.

6.2.1. Tabla ”Users”

- id: atributo único para distinguir cada fila
- username: nombre de usuario
- fullname: nombre completo
- image: nombre de la imagen de perfil del usuario
- is_favourite: indica si el usuario ha sido guardado como favorito o ha sido guardado por haber intercambiado mensajes con él

6.2.2. Tabla "Groups"

- group_id: identificador de 32 caracteres para identificar inequívocamente a un grupo ya que se permite crear dos grupos con el mismo nombre
- name: nombre del grupo
- image: nombre de la imagen del grupo
- created_by: nombre de usuario del creador del grupo

6.2.3. Tabla "GroupsUsers"

- id: atributo único para distinguir cada fila
- group_id: identificador de 32 caracteres para identificar el grupo al que pertenece el usuario
- user: nombre de usuario del miembro del grupo

6.2.4. Tabla "Messages"

- id: atributo único para distinguir cada fila
- message_to: indica el destinatario del mensaje
- message_from: indica el emisor del mensaje
- chatId: indica el identificador del chat. Corresponde a un nombre de usuario o a un identificador de grupo
- message: el mensaje enviado
- message_datetime: fecha y hora del envío del mensaje
- is_group: indica si el mensaje pertenece a un grupo
- is_read: indica si el mensaje ha sido leído. Si es un mensaje que se ha enviado, este valor será TRUE. Si es un mensaje recibido, este valor será FALSE hasta que el usuario receptor acceda al chat para leerlo
- is_programmed: indica si el mensaje ha sido programado para una fecha y hora concretas
- is_file: indica si el mensaje en realidad es un fichero
- file_type: indica el tipo del fichero enviado/recibido
- congratulation_id: indica el identificador de la felicitación en el caso de que el mensaje haya sido creado en una
- type: indica el tipo del mensaje. Para los mensajes enviados por el usuario el tipo será "normal" y para los mensajes destinados a informar (como la eliminación de un participante de un grupo) el tipo será "info"

A continuación, se podrán los diferentes casos que pueden darse para un mensaje:

Si es un mensaje para un usuario:

- id: atributo único para distinguir cada fila
- message_to: nombre del usuario destinatario del mensaje
- message_from: nombre de usuario emisor del mensaje
- chatId: el valor de este atributo cambiará de la siguiente forma: nombre de usuario del receptor para el emisor y nombre de usuario del emisor para el receptor
- message: el mensaje enviado
- message_datetime: fecha y hora del envío del mensaje
- is_group: en este caso será FALSE
- is_read: será TRUE si el mensaje ha sido enviado por el usuario y FALSE si el mensaje ha sido recibido y no ha sido leído en el chat pertinente
- is_programmed: en este caso será FALSE
- is_file: en este caso será FALSE
- file_type: en este caso será NULL
- congratulation_id: en este caso será NULL
- type: en este caso será ″normal″

Si es un mensaje para un grupo:

- id: atributo único para distinguir cada fila
- message_to: id del grupo destinatario del mensaje
- message_from: nombre de usuario emisor del mensaje
- chatId: id del grupo del mensaje
- message: el mensaje enviado
- message_datetime: fecha y hora del envío del mensaje
- is_group: en este caso será TRUE
- is_read: será TRUE si el mensaje ha sido enviado por el usuario y FALSE si el mensaje ha sido recibido y no ha sido leído en el chat pertinente
- is_programmed: en este caso será FALSE
- is_file: en este caso será FALSE
- file_type: en este caso será NULL
- congratulation_id: en este caso será NULL
- type: en este caso será ″normal″

Si es un archivo para un usuario:

- id: atributo único para distinguir cada fila

- message_to: nombre del usuario destinatario del archivo
- message_from: nombre de usuario emisor del archivo
- chatId: el valor de este atributo cambiará de la siguiente forma: nombre de usuario del receptor para el emisor y nombre de usuario del emisor para el receptor
- message: el valor de este atributo cambiará de la siguiente forma: para el emisor será el path del archivo y para el receptor será el nombre del archivo
- message_datetime: fecha y hora del envío del archivo
- is_group: en este caso será FALSE
- is_read: será TRUE si el mensaje ha sido enviado por el usuario y FALSE si el mensaje ha sido recibido y no ha sido leído en el chat pertinente
- is_programmed: en este caso será FALSE
- is_file: en este caso será TRUE
- file_type: indica si es un vídeo, audio, foto u otro tipo de archivo
- congratulation_id: en este caso será NULL
- type: en este caso será ”normal”

Si es un archivo para un grupo:

- id: atributo único para distinguir cada fila
- message_to: id del grupo destinatario del archivo
- message_from: nombre de usuario emisor del archivo
- chatId: id del grupo del mensaje
- message: el valor de este atributo cambiará de la siguiente forma: para el emisor será el path del archivo y para los receptores será el nombre del archivo
- message_datetime: fecha y hora del envío del archivo
- is_group: en este caso será TRUE
- is_read: será TRUE si el mensaje ha sido enviado por el usuario y FALSE si el mensaje ha sido recibido y no ha sido leído en el chat pertinente
- is_programmed: en este caso será FALSE
- is_file: en este caso será TRUE
- file_type: indica si es un vídeo, audio, foto u otro tipo de archivo
- congratulation_id: en este caso será NULL
- type: en este caso será ”normal”

Si es un mensaje programado para un usuario:

- id: atributo único para distinguir cada fila

- message_to: nombre del usuario destinatario del mensaje
- message_from: nombre de usuario emisor del mensaje
- chatId: el valor de este atributo cambiará de la siguiente forma: nombre de usuario del receptor para el emisor y nombre de usuario del emisor para el receptor
- message: el mensaje enviado
- message_datetime: fecha y hora en las que se desea que se envíe el mensaje al destinatario
- is_group: en este caso será FALSE
- is_read: será TRUE si el mensaje ha sido enviado por el usuario y FALSE si el mensaje ha sido recibido y no ha sido leído en el chat pertinente
- is_programmed: en este caso será TRUE
- is_file: en este caso será FALSE
- file_type: en este caso será NULL
- congratulation_id: en este caso será NULL
- type: en este caso será ″normal″

Si es un mensaje programado para un grupo:

- id: atributo único para distinguir cada fila
- message_to: id del grupo destinatario del mensaje
- message_from: nombre de usuario emisor del mensaje
- chatId: id del grupo del mensaje
- message: el mensaje enviado
- message_datetime: fecha y hora en las que se desea que se envíe el mensaje al destinatario
- is_group: en este caso será TRUE
- is_read: será TRUE si el mensaje ha sido enviado por el usuario y FALSE si el mensaje ha sido recibido y no ha sido leído en el chat pertinente
- is_programmed: en este caso será TRUE
- is_file: en este caso será FALSE
- file_type: en este caso será NULL
- congratulation_id: en este caso será NULL
- type: en este caso será ″normal″

6.2.5. Tabla "Events"

- event_id: identificador de 32 caracteres para identificar inequívocamente a un evento ya que se permite crear dos eventos con el mismo nombre
- name: nombre del evento
- events_datetime: fecha y hora en el cual será llevado a cabo el evento/reunión acordados
- description: descripción del evento
- location: lugar en el cual será llevado a cabo el evento/reunión

6.2.6. Tabla "EventsUsers"

- id: atributo único para distinguir cada fila
- event_id: identificador de 32 caracteres para identificar el evento al que pertenece el usuario
- user: nombre de usuario del miembro del evento

6.2.7. Tabla "Congratulations"

- congratulation_id: identificador de 32 caracteres para identificar inequívocamente una felicitación
- congratulation_to: nombre del usuario destinatario de la felicitación
- congratulation_from: nombre del usuario emisor de la felicitación
- message: mensaje de la felicitación
- congratulation_datetime: fecha y hora en el cual será enviada la felicitación
- are_there_files: indica si el usuario ha adjuntado un archivo
- files_path: indica el path del archivo adjuntado
- files_type: indica el tipo del archivo adjuntado

6.2.8. Tabla "SMS"

- id: identificador único del SMS analizado
- sms_from: número de teléfono emisor del SMS
- sms_datetime: fecha y hora de recepción del SMS
- message: SMS recibido
- dangerous: indica si el SMS ha sido detectado como peligroso

A continuación, se describirán los dos casos que pueden darse para un SMS:

Si es un SMS no detectado como peligroso:

- id: identificador único del SMS analizado
- sms_from: en este caso será NULL
- sms_datetime: en este caso será NULL
- message: en este caso será NULL
- dangerous: en este caso será FALSE

Si es un SMS detecado como peligroso:

- id: identificador único del SMS analizado
- sms_from: número de teléfono emisor del SMS
- sms_datetime: fecha y hora de recepción del SMS
- message: SMS recibido
- dangerous: en este caso será TRUE

A continuación, para evitar redundar, se describirán únicamente los atributos que no hayan sido explicados anteriormente ya que el resto se mantiene igual:

6.2.9. Tabla ”Users” del *backend*

- email: correo electrónico
- token: identificador único de dispositivo que sirve en el momento de enviar los mensajes a los usuarios

6.2.10. Tabla ”Groups” del *backend*

- type: en este caso será ”groups”
- operation: será ”creation” al crear un grupo y ”delete” al eliminar a un participante
- participants: lista de nombres de usuario de los miembros del grupo (solo se usa cuando operation es ”creation”)
- delete_member: nombre de usuario del miembro que se va a eliminar (solo se usa cuando operation es ”delete”)

6.2.11. Tabla ”Events” del *backend*

- type: en este caso será ”events”
- operation: será ”creation” al crear un evento y ”delete” al eliminar a un participante
- event_to: nombre de usuario del destinatario del mensaje
- participants: lista de nombres de usuario de los miembros del evento (solo se usa cuando operation es ”creation”)
- delete_member: nombre de usuario del miembro que se va a eliminar (solo se usa cuando operation es ”delete”)

6.3 Objetivos de desarrollo sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				✗
ODS 2. **Hambre cero.**				✗
ODS 3. **Salud y bienestar.**			✗	
ODS 4. **Educación de calidad.**				✗
ODS 5. **Igualdad de género.**				✗
ODS 6. **Agua limpia y saneamiento.**				✗
ODS 7. **Energía asequible y no contaminante.**				✗
ODS 8. **Trabajo decente y crecimiento económico.**				✗
ODS 9. **Industria, innovación e infraestructuras.**		✗		
ODS 10. **Reducción de las desigualdades.**				✗
ODS 11. **Ciudades y comunidades sostenibles.**				✗
ODS 12. **Producción y consumo responsables.**				✗
ODS 13. **Acción por el clima.**				✗
ODS 14. **Vida submarina.**				✗
ODS 15. **Vida de ecosistemas terrestres.**				✗
ODS 16. **Paz, justicia e instituciones sólidas.**				✗
ODS 17. **Alianzas para lograr objetivos.**				✗

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Socialme es una aplicación de mensajería destinada a facilitar la comunicación entre las personas mediante la inclusión de nuevas características. Partiendo de esa base, podría considerarse que la aplicación participa en la mejora de los dos siguientes objetivos de desarrollo sostenible: "Salud y bienestar" e "Industria, innovación e infraestructuras". A continuación, se sustentará la anterior afirmación.

Salud y bienestar

Se ha considerado la aplicación para este objetivo por el hecho de poder ayudar a las personas solitarias a comunicarse con otros por la facilidad que ofrece de buscar usuarios. Además, con sus nuevas características potencia la sociabilidad al facilitar la planificación de reuniones o eventos. Esto puede llevar a incrementar el bienestar de esas personas y, por lo tanto, de su salud.

Industria, innovación e infraestructuras

Este objetivo se alinea más con la aplicación por el desarrollo de esas nuevas características anteriormente mencionadas que podrían implicar innovaciones tecnológicas.