BUDGET BUDDY

Álvaro Aparicio Montoto & Armando Rafael Guzmán

2ºDAM

Tutor: Antonio Jesús Carmona Lara

RESUMEN

"Budget Buddy" es una aplicación móvil diseñada para simplificar la gestión de gastos compartidos entre grupos de individuos, como amigos, compañeros de piso o familiares. Inspirados por la complejidad de dividir los gastos equitativamente, buscamos proporcionar una solución práctica y accesible que promueva la transparencia y la colaboración en la administración financiera compartida. Entre sus funcionalidades se destacan la creación de cuentas conjuntas, una interfaz intuitiva y un sistema de chat interno que facilita la comunicación entre los usuarios. "Budget Buddy" ofrece una experiencia fluida para dividir los gastos de manera equitativa.

ABSTRACT

"Budget Buddy" is a mobile application developed to simplify the management of shared expenses among groups of individuals. Based on the need to address the complexity of equitably dividing expenses, it focuses on promoting transparency and collaboration in shared financial management. The application offers features such as joint account creation, an intuitive interface, and an internal chat system that facilitates communication between users. "Budget Buddy" enhances organization and clarity in managing shared expenses, promoting healthier relationships among users.

INDICE

JUSTIFICACIÓN

La creación de "Budget Buddy" responde a la necesidad creciente de simplificar la gestión de gastos compartidos en la vida cotidiana. Ante los desafíos de dividir equitativamente los gastos entre amigos, compañeros de piso o familiares, se

busca ofrecer una solución práctica y accesible que promueva la transparencia y la colaboración en la administración de las finanzas compartidas.

Fomentando la Organización y Transparencia en la Gestión de Gastos:

En la vida moderna, compartir gastos es común en diversas situaciones, desde salidas con amigos hasta gastos del hogar compartido. Sin embargo, esta práctica puede generar confusiones y malentendidos. "Budget Buddy" se ha concebido como una herramienta para simplificar y agilizar este proceso, permitiendo llevar un control claro y equitativo de los gastos compartidos. Con características intuitivas y una interfaz amigable, esta aplicación busca fomentar la organización y la transparencia en la gestión de las finanzas compartidas, promoviendo así relaciones saludables y colaborativas.

Empoderando a los Usuarios para una Mejor Gestión Financiera:

Es crucial que los individuos tengan control total sobre sus finanzas compartidas. "Budget Buddy" no solo ofrece una plataforma para registrar y dividir gastos, sino que también proporciona herramientas y recursos para ayudar a tomar decisiones financieras informadas. Al empoderar a los usuarios con información y control, la aplicación busca promover una mayor responsabilidad financiera y una mejor comprensión de los hábitos de gasto, contribuyendo así al bienestar financiero.

En resumen, "Budget Buddy" representa una visión de transformar la gestión de gastos compartidos en una experiencia más sencilla, transparente y colaborativa. Esta aplicación tiene el potencial de mejorar significativamente la forma en que se gestionan las finanzas compartidas, promoviendo relaciones más saludables y equitativas en la vida diaria. Se confía en que esta iniciativa tendrá un impacto positivo en la vida de los usuarios.

Tecnologías utilizadas:

La decisión de desarrollar esta aplicación como una aplicación móvil en vez de otras plataformas como una aplicación de escritorio o página web, se tomó basándose en lo siguiente:

Gran mercado potencial: Según un artículo publicado por (Shanahan & Bahia, 2023), más de la mitad de la población actual posee un dispositivo móvil, lo que representa un gran mercado potencial para la aplicación.

Comodidad y eficiencia: La naturaleza de la aplicación, que gestiona gastos, hace que sea más cómodo y eficiente poder añadir y editar dichos gastos rápidamente desde el móvil.

Elección del sistema operativo:

En relación con los dispositivos móviles, se optó por desarrollar la aplicación para el sistema operativo Android, ya que, como se refleja en los datos expuestos por (Howarth, 2023), Android representa aproximadamente el 70,29% del mercado global actual, en comparación con el sistema operativo iOS de Apple, que tiene un porcentaje cercano al 29,99%.

Lenguaje de programación:

Dentro del desarrollo en Android, los lenguajes de programación posibles son Java o Kotlin. Se decidió implementar la aplicación con este último, ya que, como Google indica en su documentación (Google, 2024), será el lenguaje prioritario para el desarrollo en Android. Además, Kotlin cuenta con una serie de características que lo convierten en un buen lenguaje para el desarrollo móvil, como la seguridad frente a valores nulos, la interoperabilidad con código Java existente y la escritura de código asincrónico más sencilla (Google, 2024).

Arquitectura del software:

Se utilizará la arquitectura de software Model-View-ViewModel, ya que es la arquitectura recomendada por Google (Google, 2023) . Además, esta arquitectura ofrece ventajas como la posibilidad de usarse para comunicar

diferentes fragmentos entre sí y sobrevivir a la creación y destrucción durante el ciclo de vida de los fragmentos (Boudjnah et al., 2023).

Backend:

Se utilizará Firebase como backend, ya que, como se expone en el artículo escrito por (Khawas, 2018), este cuenta con un conjunto de funciones que agilizarán el desarrollo de la aplicación, como autenticación, base de datos en tiempo real, almacenamiento de archivos, entre muchas otras.

INTRODUCCIÓN

"Budget Buddy" es una aplicación diseñada para simplificar la gestión de gastos compartidos entre grupos de individuos. Nuestro principal objetivo es abordar la creciente necesidad de organizar y administrar de manera equitativa los gastos compartidos en entornos colaborativos, como entre amigos, compañeros de piso o familiares. Inspirados por la complejidad que conlleva dividir los gastos, buscamos proporcionar una solución práctica y accesible que promueva la transparencia y la colaboración en la administración de las finanzas compartidas.

Entre las funcionalidades más destacadas de "Budget Buddy", se encuentra la posibilidad de crear cuentas conjuntas, donde todos los participantes pueden visualizar los gastos compartidos de manera clara y transparente. Esto simplifica la gestión y clarificación de las cuentas compartidas, ofreciendo una experiencia fluida y eficiente para todos los usuarios.

El nombre "Budget Buddy" refleja nuestra idea de brindar una herramienta que actúe como un aliado en la gestión de los presupuestos compartidos. "Budget" hace referencia al conjunto de gastos compartidos, mientras que "Buddy" sugiere la idea de tener un compañero o amigo que te ayuda en esa gestión. Así, "Budget Buddy" se convierte en un compañero confiable que facilita el proceso de manejar y dividir los gastos compartidos de manera equitativa y transparente.

OBJETIVOS

Objetivo General:

Desarrollar la aplicación móvil "Budget Buddy" para simplificar la gestión de gastos compartidos entre grupos de individuos, promoviendo la transparencia y la colaboración en la administración financiera compartida.

Objetivos Específicos:

Implementar un sistema de inicio de sesión seguro que garantice el acceso controlado a las cuentas de usuario.

Diseñar una interfaz de usuario intuitiva y fácil de usar que permita a los usuarios registrar y dividir los gastos compartidos de manera eficiente.

Integrar inteligencia artificial para reconocer automáticamente el monto total de los recibos o facturas capturados mediante una foto.

Establecer un sistema de comunicación interna que facilite la interacción entre los usuarios para discutir y aclarar los detalles de los gastos compartidos.

Implementar un sistema de notificaciones push que alerte a los usuarios sobre nuevos gastos registrados, actualizaciones en las cuentas compartidas y recordatorios de pagos pendientes.

ANÁLISIS DE MERCADO

Definición del Mercado Objetivo:

"Budget Buddy" se dirige a grupos de individuos que comparten gastos en diversas situaciones, como amigos, compañeros de piso o familiares.

El público objetivo incluye a personas de diferentes rangos de edad y estilos de vida, con un enfoque particular en aquellos que valoran la transparencia y la facilidad de uso en la gestión de sus finanzas compartidas.

Tamaño del Mercado:

El mercado potencial incluye a millones de personas en todo el mundo que comparten gastos regularmente en su vida cotidiana.

El crecimiento del mercado está impulsado por cambios en los estilos de vida, como la tendencia hacia la cohabitación y la colaboración financiera entre amigos y familiares.

Competencia:

La competencia incluye aplicaciones similares como Tricount, Splitwise y Settle Up, que ofrecen servicios de gestión de gastos compartidos.

"Budget Buddy" se diferencia al ofrecer mejoras en la experiencia del usuario, como una interfaz más intuitiva y funciones adicionales.

Análisis FODA:

Fortalezas: Mejoras en la experiencia del usuario y funcionalidades adicionales.

Oportunidades: Expansión a nuevos mercados, asociaciones estratégicas con instituciones financieras o plataformas de pago.

Debilidades: Posible falta de conciencia de marca en comparación con competidores establecidos.

Amenazas: Competencia fuerte y cambios en las preferencias del consumidor.

Tendencias del Mercado: La tendencia hacia la digitalización y la simplificación de las finanzas personales ha impulsado la demanda de aplicaciones de gestión de gastos compartidos.

Precio y Modelo de Negocio:

"Budget Buddy" puede seguir un modelo de negocio freemium, ofreciendo una versión básica gratuita y características premium por una tarifa mensual o anual.

Los precios competitivos y las opciones flexibles pueden aumentar la aceptación del mercado y generar ingresos sostenibles.

Regulaciones y Consideraciones Legales:

Se deben cumplir las regulaciones locales relacionadas con la protección de datos financieros y la privacidad del usuario.

Es necesario garantizar la seguridad y la integridad de la información financiera del usuario para construir confianza en la aplicación.

METODOLOGÍA

1- Planificación y gestión de tareas

El desarrollo de la aplicación comienza con una fase de planificación exhaustiva. Utilizando la aplicación Notion como herramienta principal. Dentro de esta se elaboró un diagrama el cual incluía los siguientes elementos:

- Requisitos del proyecto: Cada requisito necesario de definió claramente, cada uno se desglosó en mini tareas específicas. Para cada mini tarea se indicó:
  - Estado: El cual podría ser “No empezado”, “Empezado” y “Terminado”.
  - Tiempo invertido: Registro del tiempo dedicado a la realización de la mini tarea, usado para calcular el tiempo total del requisito.
- Estado de progreso: Para cada requisito, se detalló el estado actual de realización de igual manera que con las mini tareas usando los mismos estados.
- Tiempo de ejecución: Cada requisito cuenta también con el tiempo que ha tomado realizarlo, calculado sumando el tiempo de cada mini tarea del requisito.

Esto permitió una monitorización detallada del progreso y una estimación precisa del tiempo total necesario para completar cada requisito del proyecto.

2- Colaboración y comunicación:

Para asegurar una colaboración eficiente entre los miembros, se implementaron las siguientes practicas:

- Llamadas de Teams: Se realizaron reuniones cada cierto tiempo a través de Microsoft Teams. Durante estas, se discutieron los avances realizados, los obstáculos encontrados y los siguientes pasos a seguir.
- Uso de Git: Para la gestión del código, se utilizó Git con GitHub. Cada desarrollador trabajó en su propia rama, lo que permitió una integración organizada y libre de conflictos. Las ramas se fusionaron periódicamente para asegurar que el código estuviera actualizado.

REQUISITOS

-**Registro**: Los usuarios deben poder registrarse en la aplicación mediante correo electrónico y contraseña, así como también mediante su cuenta de Google.

-**Inicio de sesión**: Los usuarios deben poder iniciar sesión en la aplicación ya sea utilizando un correo electrónico y una contraseña, o mediante su cuenta de Google.

-**Mantener sesión abierta**: La aplicación debe recordar qué usuario ha iniciado sesión.

-**Gestionar cuenta**: Los usuarios deben tener la capacidad de gestionar su perfil, ya sea cambiar su nombre de usuario, email, contraseña, foto de perfil, así como también tener la opción de borrar su cuenta en cualquier momento.

-**Gestión de grupos**: Los usuarios deben poder crear grupos de gastos compartidos e invitar a otros usuarios a unirse a esos grupos. También deben poder abandonar grupos existentes si así lo desean.

-**Invitaciones**: Los usuarios deben poder recibir, aceptar y rechazar invitaciones de parte de otros usuarios, tanto para ser amigos o para unirse a grupos.

-**Comunicación interna**: Los usuarios deben poder comunicarse entre ellos dentro de la aplicación, a través de mensajes, para discutir y aclarar detalles relacionados con los gastos compartidos.

-**Tema oscuro**: La aplicación debe tener tema claro y oscuro.

-**Compatibilidad con múltiples idiomas**: La aplicación debe estar disponible en varios idiomas para adaptarse a las preferencias lingüísticas de los usuarios de diferentes regiones.

-Responsive: La aplicación debe poder usarse tanto desde móviles como desde tabletas.

-**Amigos**: El usuario debe poder agregar a otros usuarios como amigos, para crear grupos con ellos.

-**Notificaciones**: Los usuarios deben recibir notificaciones en tiempo real sobre nuevas transacciones, invitaciones a grupos y recordatorios de pagos pendientes.

-**Registro de gastos**: Los usuarios deben poder registrar los gastos compartidos dentro de cada grupo, especificando el monto del gasto, la descripción y quién participó en la transacción.

-**División de gastos**: La aplicación debe calcular automáticamente la división equitativa de los gastos entre los miembros del grupo, permitiendo a los usuarios ajustar manualmente la distribución si es necesario.

-**Integración de inteligencia artificial**: La aplicación debe integrar inteligencia artificial para ofrecer a los usuarios la oportunidad de capturar recibos o facturas mediante una foto, y el sistema será capaz de reconocer automáticamente el monto total de la transacción. A partir de esta información, la aplicación realizará las acciones necesarias para distribuir equitativamente los gastos entre los participantes del grupo, facilitando así el proceso de registro y división de gastos.

Esta característica agilizará significativamente la entrada de datos y mejorará la eficiencia en la gestión financiera compartida.

RFTP

R01: Los usuarios deben poder registrarse.

- **R01F01:** La aplicación debe contar con la funcionalidad de crear una cuenta rellenando en un formulario los datos necesarios y enviando dichos datos a la base de datos.
    - **R01F01T01:** Diseñar el fragmento de registro.
    - **R01F01T02:** Crear y configurar proyecto en Firebase.
    - **R01F01T03:** Integrar proyecto de Firebase con proyecto de Android.
    - **R01F01T04:** Definir clase base de todas las clases de validaciones.
    - **R01F01T05:** Crear clases de validaciones que implementen los métodos definidos en la clase base.
    - **R01F01T06:** Definir clase abstracta de la cual heredarán las demás clases validadoras.
    - **R01F01T07:** Crear las clases validadoras para cada tipo de dato requerido en el formulario.
    - **R01F01T08:** Crear el RegisterViewModel con atributos a los cual vincular cada campo del formulario.
    - **R01F01T09:** Implementar inyección de dependencias e inyectar el ViewModel en el fragmento.
    - **R01F01T10:** Implementar dataBinding entre el ViewModel y la vista de registro.
    - **R01F01T11:** Implementar métodos para validar cada campo.

- **R01F01T12:** Implementar métodos de navegación para ir al fragmento de login o al activity de Home una vez se haya registrado.
- **R01F01T13:** Vincular los eventos producidos en la vista con los métodos definidos en el ViewModel.
- **R01F01T14:** Implementar método para crear cuenta en Firebase con correo y contraseña.
- **R01F01T15:** Crear clase modelo de usuario.
- **R01F01T16:** Crear clase repositorio de usuarios con método para guardar un usuario en base de datos.
- **R01F01T17:** Implementar lógica para guardar el usuario en base de datos después de crear la cuenta en Firebase.
- **R01F01T18:** Implementar un ProgressBar y hacer que este sea visible cuando se está procesando el registro.
- **R01F01T19:** Implementar método en repositorio para hallar un usuario mediante su nombre de usuario.
- **R01F01T20:** Implementar validación en ViewModel para determinar si el nombre de usuario con el cual se quiere registrar el usuario ya está tomado o no.
- **R01F01T21:** Crear clase de datos que contengan los datos a mostrar en una ventana emergente.
- **R01F01T22:** Diseñar las ventanas emergentes de error y éxito.
- **R01F01T23:** Implementar lógica para mostrar una ventana emergente en caso de éxito o error en el registro.

PRUEBAS:

Tareas	Contenido
R01F01T10	Se fue cambiando el valor de los atributos del ViewModel vinculados a la vista y se pudo observar que estos

	se actualizaban automáticamente reflejando dicho cambio.
R01F01T11	Se le paso como argumento a cada uno de estos métodos una serie de datos tanto correctos como incorrectos y se comprobaba el que el valor devuelto era el esperado.
R01F01T12	Se vinculó cada uno de estos métodos a botones de la interfaz y se pudo navegar a las diferentes destinaciones dentro de la aplicación.
R01F01T13	Se hizo que cada vez que se produjera uno de estos eventos, se mostrase por consola un Log con un mensaje identificativo.
R01F01T14	Para validar que se creaba la cuenta, se mostraba por consola mediante un Log el objeto almacenado en la variable de Firebase auth.currentUser
R01F01T17	Se pudo observar desde el panel de administración de Firebase, que el objeto de usuario se había creado correctamente con cada uno de los datos introducidos por el usuario.
R01F01T20	Se mostraba un valor booleano por consola mediante Logs indicando si el nombre de usuario introducido por el usuario ya estaba tomado o no.
R01F01T23	Si el usuario nuevo se mostraba en la base de datos, se mostraba la ventana de éxito y en caso de que no se hubiera creado, se mostraba la de error.

R02: Los usuarios deben poder iniciar sesión.

- **R02F01:** La aplicación debe contar con la función de poder iniciar sesión introduciendo el correo electrónico y contraseña de una cuenta creada previamente.
    - **R02F01T01:** Diseñar fragmento de Login.
    - **R02F01T02:** Implementar método para iniciar sesión mediante correo y contraseña.
    - **R02F01T03:** Vincular evento en la vista para recoger los datos introducidos e iniciar la tarea de inicio de sesión.
    - **R02F01T04:** Implementar un ProgressBar para indicar que se está llevando a cabo la tarea de inicio de sesión.
    - **R02F01T05:** Implementar lógica para mostrar ventana emergente indicando si se ha iniciado sesión o no.
    - **R02F01T06:** Implementar lógica para navegar al Home una vez se ha iniciado sesión.
    - **R02F01T07:** Implementar lógica para poder navegar al fragmento de registro.
- **R02F02:** La aplicación debe contar con la función de poder iniciar sesión mediante una cuenta existente de Google.
    - **R02F02T01:** Añadir botón en la vista de Login para iniciar sesión mediante cuenta de Google.
    - **R02F02T02:** Implementar métodos para iniciar sesión mediante cuenta de Google.
    - **R02F02T03:** Vincular evento de botón al método para iniciar sesión con Google.
    - **R02F02T04:** Implementar lógica para mostrar el ProgressBar durante el procesamiento del inicio sesión con Google.

- **R02F02T05:** Implementar lógica para mostrar ventana emergente cuando termine el proceso de inicio de sesión.
- **R02F02T06:** Implementar validación para determinar si ya existen datos asociados a la cuenta con la cual se ha iniciado sesión o no.
- **R02F02T07:** Diseñar fragmento auxiliar 'PersonalDataFragment 'para que el usuario introduzca los datos necesarios para crearse una cuenta después de iniciar sesión mediante Google por primera vez.
- **R02F02T08:** Validar y recoger los datos introducidos en el formulario de 'PersonalData' y crear el objeto de usuario dentro de la base de datos con RegisterViewModel.
- **R02F02T09:** Implementar progressBar en 'PersonalDataFragment' para indicar que se está realizando la tarea de crear el usuario en la base de datos.
- **R02F02T10:** Implementar lógica para mostrar ventana emergente cuando termine la tarea informando si se ha creado o no el usuario en la base de datos.
- **R02F02T11:** Implementar navegación para que el usuario sea redirigido al 'PersonalDataFragment' cuando se valide que no hay datos guardados en la base de datos asociados a su cuenta de Google, o redirigido al Activity de Home en caso de que si los haya.
- **R02F02T12:** Implementar navegación en 'PersonalDataFragment' para que el usuario sea redirigido al Activity Home una vez se haya creado el usuario en la base de datos.

PRUEBAS:

Tareas	Contenido
R02F01T03	Los datos recogidos se muestran mediante Logs por consola y también se muestra el valor de FirebaseAuth.current user para comprobar que se ha iniciado sesión.
R02F01T05	Se comprueba que cuando el valor de FirebaseAuth.currentUser no es nulo, la ventana de éxito se muestra, y se muestra la de error cuando si lo es.
R02F01T06	Se comprueba que al iniciar sesión correctamente, el usuario es redirigido al Activity de Home
R02F01T07	Se comprueba que al presionar el botón para ir al registro se navega correctamente a dicha vista.
R02F02T03	Se observa que al presionar sobre el botón se abre una ventana desde donde iniciar sesión con una cuenta de Google
R02F02T04	Se comprueba que mientras se está procesando la tarea, ProgressBar es visible, una vez que termina esta desaparece.
R02F02T05	Se comprueba que cuando se inicia sesión correctamente y el valor de FirebaseAuth.currentUser no distinto de nulo, se muestra la ventana de

	éxito y cuando si lo es, se muestra la de error.
R02F02T06	Se muestra por pantalla el valor devuelto por el método de 'UsersRepository' para encontrar un usuario mediante su email.
R02F02T08	Se muestran los datos introducidos mediante Logs y se muestra también el resultado devuelto por el método de crear usuario de 'UsersRepository'. También desde el panel de administración de Firebase se observa el nuevo usuario creado.
R02F02T10	Se comprueba que, si el valor devuelto por la tarea de creación de registro en la base de datos es positivo, se muestra la ventana de éxito, y si es negativo se muestra la de error.
R02F02T11	Se comprueba que cuando no hay datos asociados a la cuenta de Google, el usuario es redirigido al fragmento de 'PersonalData' para que introduzca sus datos, así como que, cuando ya existen dichos datos previamente, este es redirigido directamente hacia el Activity de Home.
R02F02T12	Se comprueba que cuando se crea correctamente el usuario en la base de datos, el usuario es redirigido correctamente hacia el Activitiy de Home.

R03: La aplicación debe recordar el usuario que actualmente ha iniciado sesión.

- **R03F01**: La aplicación tendrá un 'Splash Screen' el cual se mostrará cada vez que se inicie la aplicación y el cual realizará las tareas de determinar el usuario que actualmente ha iniciado sesión.
    - **R03F01T01**: Diseñar 'Splash Screen'.
    - **R03F01T02**: Implementar lógica para determinar el usuario que actualmente ha iniciado sesión y redirigirlo a la vista correspondiente.

Pruebas:

Tareas	Contenido
R03F01T02	Se comprueba que cuando ningún usuario ha iniciado sesión, después de mostrar el splash screen, el usuario es redirigido hacia la vista de Login. En el caso de que, si haya iniciado sesión, se valida si existen datos dentro de la base de datos asociados, si los hay es redirigido hacia la actividad 'HomeActivity', en el caso contrario es redirigido hacia el fragmento 'PersonalDataFragment'.

R04: Los usuarios deben poder gestionar su perfil dentro de la aplicación, ya sea cambiando su foto de perfil, nombre de usuario, email, contraseña o borrando su cuenta si así lo desean

- **R04F01**: La aplicación debe contar con un fragmento dentro de la actividad de 'HomeActivity' en donde se expongan los datos del usuario que actualmente ha iniciado sesión, así como todas las operaciones que este dispone para realizar relativas a su propio perfil.

- **R04F01T01**: Diseñar el fragmento ‘ProfileFragment’.
- **R04F01T02**: Implementar animación al desplegar la lista de acciones que puede realizar el usuario.
- **R04F01T03**: Implementar lógica para cargar los datos del usuario actual.
- **R04F01T04**: Implementar ventana emergente para que el usuario introduzca un dato solicitado por teclado.
- **R04F01T05**: Implementar ventana emergente para que el usuario introduzca dos datos solicitados por teclado.
- **R04F01T06**: Implementar lógica para cerrar sesión.
- **R04F01T07**: Implementar lógica para mostrar ventana emergente que pida email y contraseña para volver a autenticarse.
- **R04F01T08**: Implementar lógica para volver a autenticarse mediante Google.
- **R04F01T09**: Implementar métodos tanto en ‘UsersRepository’ como ‘ProfileViewModel’ para borrar la cuenta.
- **R04F01T10**: Implementar lógica para mostrar ventana emergente que pida nueva contraseña y la valide, para cambiarla.
- **R04F01T11**: Implementar lógica para mostrar ventana emergente que pida un nuevo nombre de usuario, lo valide y usar métodos tanto en ‘UsersRepository’ como ‘ProfileViewModel’ para cambiarlo.
- **R04F01T12**: Implementar lógica para cambiar correo electrónico.
- **R04F01T13**: Diseñar ventana emergente que muestre las opciones para cargar una foto desde la galería o desde la cámara.
- **R04F01T14**: Implementar lógica para cargar foto desde la galería.

- **R04F01T15**: Implementar lógica para cargar foto desde la cámara.
- **R04F01T16**: Implementar la clase 'StorageRepository' y método para subir fotos a Firebase storage.
- **R04F01T17**: Implementar lógica para cargar foto desde foto desde Firebase.
- **R04F01T18**: Implementar lógica para cargar foto desde Google.
- **R04F01T19**: Vincular los eventos que se producen en la vista con los métodos definidos en el 'ProfileViewModel'.

Pruebas:

Tareas	Contenido
R04F01T03	Se comprueba que se cargan correctamente los datos del usuario que actualmente ha iniciado sesión, mostrando en la vista su nombre de usuario y foto de perfil.
R04F01T06	Se comprueba que después de cerrar sesión el valor de FirebaseAuth.currentUser es nulo y el usuario es redirigido hacia el fragmento de Login
R04F01T07	Se muestra por consola mediante Logs, el valor 'true' si el resultado de la tarea de re-autenticación fue exitoso y el valor 'false' si no lo fue.
R04F01T08	Se muestra por consola mediante Logs, el valor 'true' si el resultado de

	la tarea de re-autenticación fue exitoso y el valor ‘false’ si no lo fue.
R04F01T09	Se valida que, al borrar la cuenta, desde el panel de administración de Firebase no aparece la cuenta en el panel de autenticación ni tampoco dentro de la base de datos.
R04F01T10	Para comprobar que la contraseña cambio, se cerró sesión y se volvió a iniciar sesión con la contraseña nueva.
R04F01T11	Se comprueba que el nombre de usuario ha cambiado dentro de la base de datos en el panel de administración de Firebase.
R04F01T12	Para comprobar que funciona, se valida que se recibe un correo de verificación para cambiar el correo en el buzón del nuevo email introducido, y una vez cambiado se vuelve a iniciar sesión usando el nuevo correo.
R04F01T14	Para confirmar que funciona, se muestra la foto cargada en el ‘ImageView’ en donde se muestra la foto de perfil del usuario.
R04F01T15	Para confirmar que funciona, se muestra la foto cargada en el ‘ImageView’ en donde se muestra la foto de perfil del usuario.
R04F01T16	Se confirma que la foto se ha subido y se puede abrir desde el panel de administración de Firebase storage

R04F01T17	Se muestra la foto cargada desde Firebase en el 'ImageView' en donde se muestra la foto de perfil del usuario.
R04F01T18	Se muestra la foto cargada desde Google en el 'ImageView' en donde se muestra la foto de perfil del usuario.

R05: El usuario debe poder crear y gestionar grupos.

- **R05F01:** La aplicación debe contar con un fragmento desde donde el cual, el usuario podrá crear un grupo e invitar a sus amigos a unirse.
  - **R05F01T01:** Diseñar el fragmento 'NewGroupFragment'.
  - **R05F01T02:** Implementar validadores para los campos de la creación de grupo.
  - **R05F01T03:** Implementar ventana emergente para solicitar fecha.
  - **R05F01T04**: Diseñar layout de amigo para mostrarlo en el formulario.
  - **R05F01T04:** Implementar lógica para mostrar amigos dentro del formulario.
  - **R05F01T05:** Implementar lógica en 'GroupsRepository' y 'NewGroupViewModel' para crear un nuevo grupo.
  - **R05F01T06:** Vincular los eventos que se producen en la vista a la lógica definida en el ViewModel.
  - **R05F01T07:** Implementar lógica para mostrar Progress Bar y ventanas emergentes para informar al usuario del resultado de sus acciones.
  - **R05F01T08:** Implementar la subida de foto para el grupo.
  - **R05F01T09:** Implementar la eliminación de la foto, para evitar uso de memoria innecesario.
  - **R05F01T10:** Implementar la lógica para invitar a aquellos amigos que se hayan seleccionado en el formulario de creación.

- **R05F02:** La aplicación debe contar con un fragmento en donde se mostrarán los grupos a los que pertenece el usuario.
    - **R05F02T01:** Diseñar fragmento ‘GroupsFragment’.
    - **R05F02T02:** Implementar lógica para cargar los grupos a los que pertenece el usuario.
    - **R05F02T03:** Diseñar layout de grupo en donde mostrar las propiedades más relevantes como nombre, descripción, foto, y categoría.
    - **R05F02T04:** Implementar RecyclerView para mostrar los grupos.
    - **R05F02T05:** Implementar lógica para ordenar los grupos para que los que hayan sido modificados más recientemente aparezcan primero.
    - **R05F02T06:** Implementar lógica para filtrar los grupos que aparecen mediante un buscador.
    - **R05F02T07:** Adaptar calendario en ‘HomeFragment’ para que se muestren los días en los cuales hay un grupo.
    - **R05F02T08:** Implementar lógica para filtrar los grupos que aparecen en función de la fecha escogida en el calendario.
    - **R05F02T09:** Implementar RecyclerView en donde mostrar las categorías de grupos existentes.
    - **R05F02T10:** Implementar lógica para filtrar los grupos que aparecen en función de las categorías seleccionadas.
    - **R05F02T11:** Implementar la carga de la foto de cada grupo.
- **R05F03:** La aplicación debe contar con un fragmento desde donde el usuario pueda actualizar los datos relacionados al grupo, así como invitar y eliminar miembros del grupo.
    - **R05F03T01:** Diseñar el fragmento ‘GroupOverviewFragment’.
    - **R05F03T02:** Implementar lógica para recibir por parámetro el grupo seleccionado y así cargar sus datos.

- **R05F03T03:** Implementar métodos para actualizar y borrar el grupo.
- **R05F03T04:** Implementar lógica para cargar los miembros del grupo seleccionado.
- **R05F03T05:** Implementar lógica para eliminar aquellos miembros que hayan sido deseleccionados.
- **R05F03T06:** Implementar lógica para invitar aquellos amigos que hayan sido seleccionados al actualizar.
- **R05F03T07:** Diseñar el 'SpinnerDialog'.
- **R05F03T08:** Implementar lógica para mostrar y ocultar ciertas opciones en función del rol que ocupa el usuario actual en el grupo.
- **R05F03T09:** Implementar lógica para cambiar el rol de los demás usuarios.
- **R05F03T10:** Implementar lógica para que en caso de que el usuario actual sea eliminado del grupo o pierda sus privilegios de administrador, la vista se actualice correctamente.
- **R05F03T11:** Implementar la actualización de foto de grupo al actualizar.
- **R05F03T12:** Implementar la eliminación de la foto al eliminar el grupo.
- **R05F03T13:** Implementar la función de salir del grupo.
- **R05F03T14:** Implementar la función de filtrar los amigos y miembros por nombre.

Pruebas:

Tareas	Contenido

R05F01T02	Se comprueba que, al introducir datos erróneos dentro de los campos, se muestra el mensaje de error correspondiente en la vista.
R05F01T03	Se valida que, al pulsar sobre la opción de escoger fecha, se muestre la ventana emergente y una vez seleccionada la fecha esta se muestra por consola para verificar que es la escogida.
R05F01T04	Se comprueba que, se muestran los amigos del usuario actual dentro del formulario, y que, si estos cambian dentro de la base de datos, el cambio se refleja dentro de la vista.
R05F01T05	Se valida que en la consola de Firebase Realtime Database se crea el grupo con los datos introducidos en el formulario.
R05F01T07	Se comprueba que se muestra el ProgressBar solo durante la ejecución de la tarea y que se muestra la ventana emergente correspondiente al resultado de la operación realizada.
R05F01T08	Se valida que lo foto se ha subido correctamente dentro del panel de administración de Firebase Storage y que tiene como nombre el Uid del grupo creado.
R05F01T09	Se comprueba que la foto ya no aparece dentro del panel de administración de Firebase Storage.

R05F01T10	Se comprueba en el panel de administración de Firebase Realtime Database que se ha creado el objeto de invitación dentro de aquellos amigos seleccionados.
R05F02T02	Se comprueba que se cargan los datos a los que pertenece el usuario dentro de la base de datos de Firebase Realtime Database.
R05F02T05	Se comprueba que los grupos recién creados o modificados son los que aparecen de primero en la lista.
R05F02T06	Se comprueba que solo aparecen aquellos grupos cuyo nombre o descripción contiene el texto introducido en el buscador.
R05F02T07	Se comprueba que solo aquellos días cuya esté contenida dentro de la fecha de inicio y fin de algún grupo se muestra con un círculo.
R05F02T08	Se comprueba que la fecha de los grupos que se muestran coincide con la fecha seleccionada en el calendario.
R05F02T09	Se comprueba que las categorías mostradas son las definidas en la clase de utilidades de la aplicación.
R05F02T10	Se comprueba que, solo aparecen aquellos grupos cuya categoría es, al menos una de las seleccionadas.
R05F02T11	Se valida que se carga correctamente la foto de aquellos grupos que la

	tengan en Storage, y que aquellos que no, se muestre la foto por defecto.
R05F03T02	Se comprueba en la base de datos de Firebase que el uid recibido coincide con el grupo seleccionado en el fragmento de grupos.
R05F03T03	Se comprueba que, al actualizar el grupo, los cambios se reflejen en la base de datos y que al borrarlo este no aparezca dentro de la colección de grupos.
R05F03T04	Se comprueba que se muestran en tiempo real, los miembros del grupo cargado.
R05F03T05	Se comprueba dentro de la base de datos que, los miembros que han sido deseleccionados fueron eliminados del grupo.
R05F03T06	Se comprueba que se crea el objeto de invitación dentro de aquellos amigos que han sido seleccionados dentro del formulario.
R05F03T08	Se valida que cuando el usuario es miembro, solo puede invitar a sus amigos al grupo, y cuando este es administrador, este puede realizar cualquier acción dentro de este.
R05F03T09	Se comprueba dentro de la base de datos que el rol del usuario ha cambiado al rol seleccionado dentro del formulario.
R05F03T10	Se valida que si otro usuario, cambia el rol del usuario actual estando

	dentro del formulario, esta se actualiza en función del rol asignado. También se valida que, si alguien elimina al usuario actual, este es redirigido hacia el fragmento de grupos.
R05F03T11	Se comprueba que, al cambiar la foto del grupo, al actualizar esta también cambie dentro de Firebase Storage.
R05F03T12	Se comprueba que, al eliminar el grupo, dentro de Firebase Storage la foto también es eliminada.
R05F03T13	Se comprueba que, al salir del grupo, el usuario ya no cuenta con la referencia a este dentro de la base de datos al igual que el grupo ya no cuenta con la referencia del usuario.
R05F03T14	Se comprueba que solo se muestran aquellos amigos y miembros cuyo nombre contenga el texto introducido en el campo de búsqueda.

R06: El usuario debe poder recibir solicitudes de otros usuarios ya sea para ser amigos o unirse a un grupo.

- **R06F01:** La aplicación debe contar con un fragmento en donde se mostrarán todas las invitaciones que tiene el usuario.
    - **R06F01T01:** Diseñar fragmento de invitaciones (InvitationsFragment).
    - **R06F01T02:** Diseñar layout de invitación.
    - **R06F01T03:** Implementar clase de adaptador.

- **R06F01T04:** Implementar ‘InvitationViewHolder’.
- **R06F01T05:** Implementar lógica de carga invitaciones tanto en ‘InvitationsViewModel’ como en ‘InvitationsRepository’.
- **R06F01T06:** Implementar lógica para aceptar y rechazar invitaciones.

Pruebas:

Tareas	Contenido
R06F01T05	Se comprueba que se cargan correctamente las invitaciones que tiene el usuario en la base de datos de Firebase Realtima Database.
R06F01T06	Se comprueba que, al aceptar las invitaciones de amistad, se crea la relación dentro de la base de datos, así como cuando la invitación es para unirse a un grupo, el usuario aparece como miembro dentro de este.

R07: Los usuarios, en los grupos que creen, deben poder enviarse mensajes de texto y compartir fotos entre sí.

- **R07F01**: La aplicación debe contar con un fragmento desde donde el cual los miembros de un grupo puedan intercambiar mensajes y fotos entre sí.
  - **R07F01T01:** Diseñar el fragmento de chat (ChatFragment).
  - **R07F01T02:** Diseñar los layout de los mensajes.
  - **R07F01T03:** Implementar lógica para recoger el texto introducido por el usuario y validarlo.

- **R07F01T04:** Añadir lógica para cargar imágenes desde la cámara y la galería.
- **R07F01T05:** Implementar lógica tanto en 'ChatRepository' como en 'ChatViewModel' para guardar mensajes en base de datos.
- **R07F01T06:** Implementar lógica para cargar los mensajes del grupo.
- **R07F01T07:** Implementar 'Listener' para saber en tiempo real si se sigue siendo miembro del grupo.

Pruebas:

Tareas	Contenido
R07F01T03	Se muestra el texto por consola para verificar que es lo introducido y se muestra por el resultado de la validación en pantalla mediante un Toast.
R07F01T04	Se muestra por consola el objeto de la foto cargada y se muestran mensajes informando del éxito y error de la carga por pantalla.
R07F01T05	Se comprueba en el panel de administración de Firebase Realtime Database que se guarda el mensaje correctamente.
R07F01T06	Se comprueba que se muestran los mensajes guardados en el panel de administración de Firebase Realtime Database.
R07F01T07	Se comprueba que, cuando otro usuario borra al usuario actual o elimina el grupo completamente, el

	usuario actual es redirigido desde el chat hacia el fragmento de grupos.

R08: La aplicación debe contar tanto con un modo claro, así como un modo oscuro el cual se activará cuando el dispositivo se encuentre en este modo.

- **R08F01**: La aplicación debe tener definidas dos paletas de colores a usar en función de si el sistema se Encuentra en modo claro u oscuro.
    - **R08F01T01**: Crear dos ficheros de colores dentro del directorio de recursos de la aplicación, uno a usar para el tema claro y otro para el tema oscuro.
    - **R08F01T02**: Definir el tema de la aplicación para que use estas paletas de colores.

Pruebas:

Tareas	Contenido
R08F01T02	Se comprueba que cuando el sistema está en modo claro, la paleta de colores usada es la definida para el modo claro, y cuando este esté en modo oscuro la paleta de colores es la de colores oscuros.

R09: La aplicación debe tener disponible traducción tanto en inglés como en español.

- **R09F01**: La aplicación debe contar con un fichero en donde definir los textos usados dentro de la aplicación tanto para inglés como para el español.
    - **R09F01T01**: Crear el fichero de strings con los textos usados por la aplicación en español.
    - **R09F01T02**: Crear el fichero de strings con los textos usados por la aplicación en inglés.

Pruebas:

Tareas	Contenido
R09F01T02	Se comprueba que cuando el dispositivo tiene como lenguaje el español, la aplicación muestra los textos en este idioma, así como que cuando el dispositivo está en ingles los textos cambian para ajustarse a este.

R10: La aplicación debe poder usarse cómodamente tanto desde un dispositivo móvil como una tableta.

- **R10F01**: La aplicación debe contar con dos ficheros de tamaños usados para las vistas, uno para móviles y otro para tabletas.
    - **R10F01T01**: Crear fichero dimen con los tamaños usados en las vistas por los dispositivos móviles.
    - **R10F01T02**: Crear fichero dimen con los tamaños usados en las vistas por las tabletas.

Pruebas:

Tareas	Contenido
R10F01T02	Se comprueba que al usar la aplicación en un emulador de teléfono móvil los tamaños usados son los establecidos para móvil, y que cuando la aplicación es ejecutada en un emulador de tableta los tamaños se hacen más grandes para ajustarse al tamaño de la pantalla.

R11: El usuario debe poder acceder a una ventana en la que aparezca una lista con todos los usuarios agregados como amigos, así como enviar invitaciones a otros.

- **R11F01:** La aplicación debe contar con un fragmento desde donde el cual, el usuario podrá visualizar una lista, la cual contiene las “cards” de todos los amigos, así como enviar invitaciones.
  - **R11F01T01:** Diseñar el fragmento ‘FragmentFriends’, que contendrá un ‘RecyclerView’ para mostrar la lista de amigos, un ‘SearchView’ para buscar amigos y botón para enviar invitación a otros usuarios.
  - **R11F01T02:** Implementar lógica en ‘FriendsViewModel’ para cargar la lista de amigos del usuario actual desde la base de datos.
  - **R11F01T03:**Implementar‘RecyclerView’en ‘FriendsFragment’ para mostrar la lista de amigos del usuario actual.
  - **R11F01T04**: Implementar lógica para buscar y filtrar amigos en la lista.
  - **R11F01T05:** Implementar lógica para enviar solicitudes de amistad.

- **R11F01T06:** Implementar adaptador ‘FriendsAdapter’ para enlazar los datos de los amigos con las vistas del ‘RecyclerView’.
- **R11F01T07:** Implementar ‘viewHolder’ ‘FriendsViewHolder’ para enlazar los datos de cada amigo con las vistas correspondientes del ‘RecyclerView’.

Pruebas:

Tareas	Contenido
R11F01T01	Se valida que, ‘FriendsFragment’ se carga correctamente.
R11F01T01	Se comprueba que, el ‘RecyclerView’ y el ‘SearchView’ están presentes en la vista.
R11F01T02	Se verifica que, al abrir el ‘FriendsFragment’, se cargan correctamente los amigos del usuario actual desde FireBase
R11F01T02	Se valida que, si no hay amigos, la lista se muestra vacía.
R11F01T02	Se verifica que, al añadir o eliminar un amigo en Firebase, loa cambion se reflejan en la vista en tiempo real.
R11F01T03	Se comprueba que, al cargar el fragmento, el ‘RecyclerView’ muestra la lista de amigos correctamente.
R11F01T03	Se valida que, al hacer scroll, los elementos del ‘RecyclerView’ se desplazan y cargan correctamente.

R11F01T03	Se verifica que los datos de cada amigo se muestran adecuadamente en cada ítem del 'RecyclerView'.
R11F01T04	Se comprueba que, al introducir texto en el 'SearchView', la lista de amigos se filtra correctamente en tiempo real.
R11F01T04	Se valida que, al borrar el texto del 'SearchView', se muestran nuevamente la lista completa de amigos.
R11F01T04	Se verifica que la búsqueda es insensible a mayúsculas y minúsculas.
R11F01T05	Se comprueba que, al pulsar el botón de añadir amigo, se muestra una ventana emergente para introducir el nombre de usuario.
R11F01T05	Se valida que, al introducir el nombre de usuario válido y pulsar OK, se envía una solicitud de amistad.
R11F01T05	Se verifica que, si el nombre de usuario no existe, se muestra un mensaje de error adecuado.
R11F01T06	Se comprueba que, el 'FriendsAdapter' enlaza correctamente los datos con las vistas del 'RecyclerView'.
R11F01T06	Se valida que, al actualizar la lista de amigos en el adaptador, los cambios se reflejan inmediatamente en la vista.
R11F01T06	Se verifica que el método 'filter' del adaptador filtra correctamente los

	elementos según el texto de búsqueda.
R11F01T07	Se comprueba que, el 'FriendsViewHolder' enlaza correctamente los datos de cada amigo con las vistas correspondientes.
R11F01T07	Se valida que las imágenes de perfil de los amigos se cargan correctamente utilizando 'ListItemImageLoader'.
R11F01T07	Se verifica que los nombres de usuario se muestran adecuadamente en la vista.

R12: Los usuarios deben recibir notificaciones sobre los mensajes enviados en los grupos a los que este pertenece.

- **R12F01:** La aplicación debe poder comunicarse mediante peticiones HTTP al backend para poder suscribirse a los mensajes de los grupos, así como enviar mensajes a dichos grupos.
    - **R12F01T01:** Implementar backend local para poder recibir peticiones HTTP en diferentes rutas y comunicarse con Firebase.
    - **R12F01T02:** Implementar clase para poder realizar las peticiones correspondientes al backend local desde la aplicación.
    - **R12F01T03:** Incorporar la llamada al servicio en los métodos que lo requieran.

Pruebas:

Tareas	Contenido
R12F01T01	Se comprueba con postman que el backend recibe correctamente las peticiones, así como que devuelve la respuesta correspondiente.
R12F01T02	Se comprueba que, al realizar las peticiones desde la aplicación, el backend recibe la petición y muestra un mensaje de Log por consola.
R12F01T03	Se valida que el dispositivo recibe las notificaciones recibidas después de realizar la petición a suscribirse, que este deja de recibirlas al realizar la petición de desuscribirse y finalmente que el usuario puede mandar mensajes y los otros usuarios reciben la notificación.

R13: El usuario debe poder acceder a una ventana en la que se le permita crear y gestionar gastos.

R13F01: La aplicación debe contar con un fragmento desde donde el usuario pueda registrar un nuevo gasto.

- **R13F01T01:** Diseñar el fragmento 'NewExpenseFragment'.

- **R13F01T02:** Implementar validadores para los campos del registro del gasto.
- **R13F01T03:** Implementar ventana emergente para seleccionar la fecha del gasto.
- **R13F01T04:** Diseñar el layout para mostrar los detalles del gasto.
- **R13F01T05:** Implementar lógica para mostrar información relevante en el formulario.
- **R13F01T06:** Implementar lógica en 'ExpenseRepository' y 'NewExpenseViewModel' para crear un nuevo gasto.
- **R13F01T07:** Vincular los eventos de la vista a la lógica definida en el ViewModel.
- **R13F01T08:** Implementar lógica para mostrar indicadores de progreso y ventanas emergentes para informar al usuario sobre el resultado de sus acciones.

R13F02: La aplicación debe contar con un fragmento desde doned el usuario pueda registrar un nuevo gasto.

- **R13F02T01:** Diseñar el fragmento 'ExpensesFragment'.
- **R13F02T02:** Implementar lógica para cargar los gastos registrados.
- **R13F02T03:** Diseñar layout para mostrar los detalles de cada gasto, como nombre, descripción, fecha y cantidad.
- **R13F02T04:** Implementar RecyclerView para mostrar los gastos.

Pruebas:

Tareas	Contenido
R13F01T01	Se valida que, ‘NewExpenseFragment se carga correctamente.
R13F01T01	Se comprueba que todos los elementos de la interfaz, como campos de entrada, botones, están presentes en la vista.
R13F01T01	Se verifica que el título del fragmento es correcto
R13F01T02	Se verifica que, al ingresar datos inválidos (por ejemplo, un número de caracteres mayor al permitido en el campo de título), se muestran mensajes de error apropiados.
R13F01T02	Se valida que, al dejar campos obligatorios en blanco y enviar el formulario, se muestran mensajes de error.
R13F01T03	Se verifica que, al hacer clic en el campo de fecha, se abre una ventana emergente para seleccionar la fecha .
R13F01T03	Se valida que, al seleccionar una fecha en la ventana emergente, esta se refleja correctamente en el campo de fecha.
R13F01T03	Se comprueba que la ventana emergente muestra el calendario correctamente.

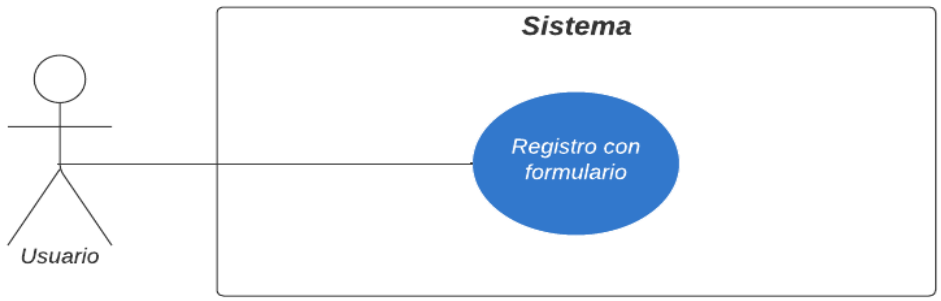
R13F01T04	Se verifica que el layout de 'NewExpenseFragment' muestra todos los detalles necesarios (nombre, descripción, cantidad, etc.).
R13F01T04	Se valida que todos los elementos del layout están alineados y son visibles correctamente.
R13F01T05	Se verifica que el formulario muestra información relevante según la lógica implementada.
R13F01T05	Se valida que, al ingresar información en los campos, estos se actualizan correctamente en la interfaz de usuario.
R13F01T06	Se verifica que, al ingresar datos válidos y enviar el formulario, el gasto se guarda correctamente en el repositorio.
R13F01T06	Se valida que, después de guardar un nuevo gasto, se muestra un mensaje de confirmación al usuario.
R13F01T06	Se comprueba que el nuevo gasto aparece en la lista de gastos registrados.
R13F01T07	Se verifica que los eventos de la vista (como clics en botones) están correctamente vinculados a la lógica del ViewModel.
R13F01T07	Se valida que, al realizar acciones en la vista, estas desencadenan la lógica esperada en el ViewModel.

R13F01T08	Se verifica que, al enviar el formulario, se muestra un indicador de progreso mientras se procesa la solicitud.
R13F01T08	Se valida que, al completar la acción, se muestra una ventana emergente con el resultado (éxito o error).
R13F02T01	Se valida que 'ExpensesFragment' se carga correctamente.
R13F02T01	Se comprueba que todos los elementos de la interfaz, como el RecyclerView, están presentes en la vista.
R13F02T01	Se verifica que el título del fragmento es correcto.
R13F02T02	Se verifica que, al abrir 'ExpensesFragment', se cargan correctamente los gastos registrados desde el repositorio.
R13F02T02	Se valida que, si no hay gastos registrados, se muestra una vista vacía.
R13F02T02	Se comprueba que los datos cargados corresponden a los gastos reales en el repositorio.
R13F02T03	Se verifica que el layout de cada ítem en el RecyclerView muestra todos los detalles necesarios (nombre, descripción, fecha, cantidad).
R13F02T03	Se valida que todos los elementos del layout están alineados y son visibles correctamente.

R13F02T04	Se verifica que el RecyclerView muestra correctamente la lista de gastos.
R13F02T04	Se valida que, al desplazarse por la lista, los ítems se cargan y muestran correctamente sin problemas de rendimiento.
R13F02T04	Se comprueba que el RecyclerView se actualiza correctamente al agregar o eliminar gastos.

CASOS DE USO

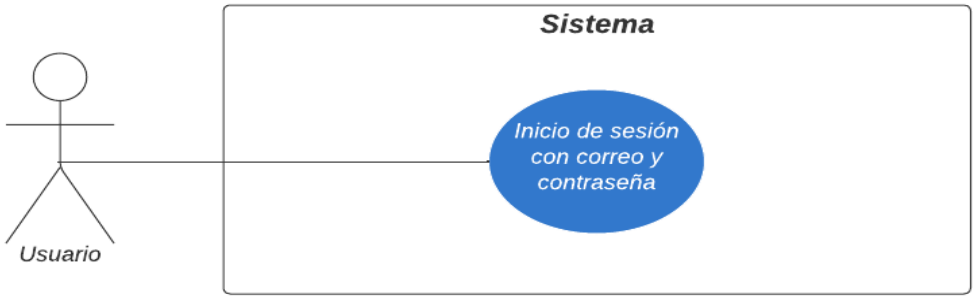
- **Registro mediante formulario**



Precondición	Postcondición
El usuario no debe haber iniciado sesión.	El usuario automáticamente iniciará sesión y pasará estar en Home de la aplicación.
Clases que intervienen	**Colecciones de base de datos afectadas**
RegisterFragment	Users
RegisterViewModel	
UsersRepository	
User	
Result	
AlertDialogFactory	
FirebaseAuth	
MainActivity	

1- El usuario inicia la aplicación y como no ha iniciado sesión, este se encuentra en la ventana de Login.
2- Para ir a la ventana de registro, pulsa sobre la opción de crear cuenta.
3- El usuario rellena todos los campos del formulario hasta que todos estén correctos y el botón de crear cuenta se active.
4- Una vez presionado el botón, todos los datos introducidos se recogen y se crea una cuenta tanto en el sistema se autenticación de Firebase como dentro de la base de datos.
5- Mientras se procesa el registro, se muestra una barra de progreso indicando que se está llevando a cabo la tarea.
6- Una vez finalizado, se le informara al usuario mediante una ventana emergente si el registro ha sido exitoso o no.
7- Si este ha sido exitoso al presionar en 'ok', el usuario será redirigido hacia el 'Home' de la aplicación, en caso de no ser exitoso, el usuario podrá volver a intentarlo.

- **Inicio de sesión con correo y contraseña.**

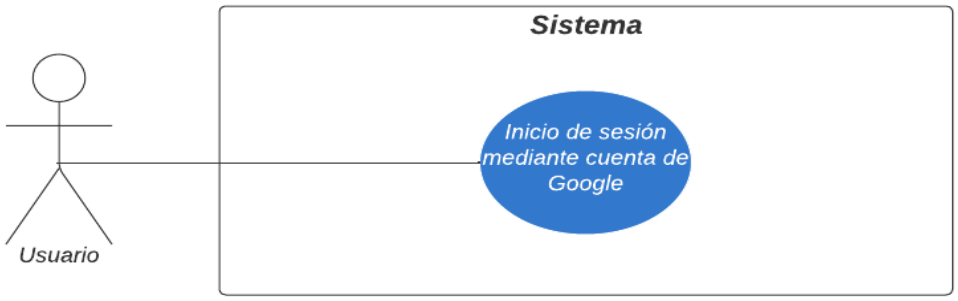


Precondición	Postcondición
El usuario no debe haber iniciado sesión.	Si el inicio de sesión es correcto, este será redirigido hacia el Activity Home, en caso de que no lo sea, lo tendrá que volver a intentar.
Clases que intervienen	**Colecciones de base de datos afectadas**
LoginFragment	Users
RegisterViewModel	
UsersRepository	
User	
Result	
AlertDialogFactory	
FirebaseAuth	
MainActivity	

1- El usuario como aún no ha iniciado sesión, este al iniciar la aplicación se encontrará con la vista de Login.
2- Al rellenar los campos de correo y contraseña y presionar el botón de 'Login', se recogerán los datos introducidos y empezará la tarea de inicio de sesión, mostrando durante el proceso un ProgressBar.
3- Una vez se ha procesado el inicio de sesión, si este es correcto se mostrará una ventana emergente con un mensaje informativo indicando si es el inicio de sesión ha sido correcto o no.

4- Si el inicio de sesión ha sido exitoso al presionar el botón de 'ok' de la ventana emergente, el usuario será redirigido hacia el Activity de Home.
5- Si el inicio de sesión no fue correcto, al usuario se le indicará el error y se le permitirá volver a intentar introducir sus datos.

- Inicio de sesión mediante una cuenta de Google

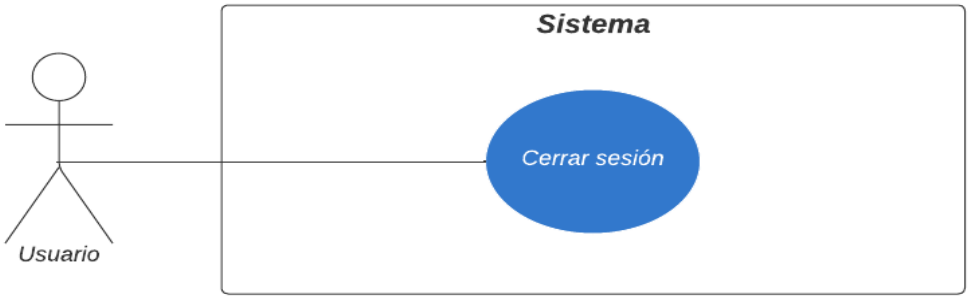


Precondición	Postcondición
El usuario no debe haber iniciado sesión.	Si el inicio de sesión es correcto, en caso de que no existan datos personales guardados del usuario en la base de datos, este será redirigido hacia el fragmento 'PersonalDataFragment' para que los introduzca, en el caso contrario será directamente redirigido hacia el Activity 'HomeActivity'.
Clases que intervienen	**Colecciones de base de datos afectadas**
LoginFragment	Users
RegisterViewModel	
UsersRepository	
User	
Result	

AlertDialogFactory	
FirebaseAuth	
MainActivity	
GoogleSignInClient	
ActivityResultLauncher	

1. Si el usuario no ha iniciado sesión, al iniciar la aplicación, se encontrará con la vista de 'Login'.
2. Al presionar el botón para iniciar sesión mediante Google, se abrirá una nueva actividad en donde al usuario se le pedirá que inicie sesión con una cuenta de Google.
3. Si el proceso de inicio de sesión de Google resulta exitoso, el usuario volverá a la vista de 'Login' y se le mostrará una ventana emergente con un mensaje indicando que ha iniciado sesión correctamente.
4. En caso de que el proceso de inicio de sesión con Google no sea exitoso, el usuario volverá a la vista de 'Login' y se le indicará mediante una ventana emergente que ha ocurrido un error y se le permitirá volverlo a intentar.
5. Una vez que el usuario inicie sesión correctamente, se verificará si existen o no datos asociados a su email dentro de la base de datos.
6. Si existen datos, será directamente redirigido hacia la actividad 'HomeActivity'.
7. Si no existen los datos, el usuario será redirigido hacia el fragmento 'PersonalDataFragment' en donde se le presentará un formulario en donde tiene que introducir su nombre, apellidos y nombre de usuario.
8. Una vez validados dichos datos, se creará un objeto de usuario dentro de la base de datos de igual forma que en el fragmento 'RegisterFragment' y este será finalmente redirigido hacia la actividad 'HomeActivity'.

- **Cerrar sesión.**

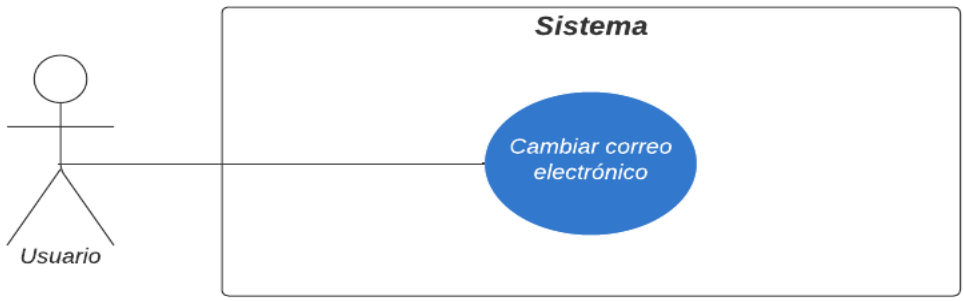


Precondición	Postcondición
El usuario debe haber iniciado sesión y encontrarse en el fragmento de 'ProfileFragment'	El usuario será redirigido hacia el fragmento de 'Login' para que pueda volver a iniciar sesión y el valor de FirebaseAuth.currentUser será nulo.
Clases que intervienen	**Colecciones de base de datos afectadas**
ProfileFragment	
HomeViewModel	
HomeActivity	
Result	
AlertDialogFactory	
FirebaseAuth	
MainActivity	
LoginFragment	

1. El usuario una vez que haya iniciado sesión y navegue hacia el fragmento 'ProfileFragment', se le mostraran su nombre de usuario, foto de perfil, así como una seria de acciones disponibles.
2. Cuando el usuario pulse sobre la opción se cerrar sesión, se mostrará un Progress Bar que indicara que se está llevando a cabo una tarea de larga duración.
3. Una vez el proceso haya culminado, se le mostrará al usuario una ventana emergente indicando si el proceso ha sido exitoso o no.

4. En caso de ser exitoso, el usuario al pulsar sobre ‘ok’, será redirigido hacia la vista de Login. Si no fue exitoso, se le informará y se le permitirá volver a intentarlo.

- **Cambiar correo electrónico.**

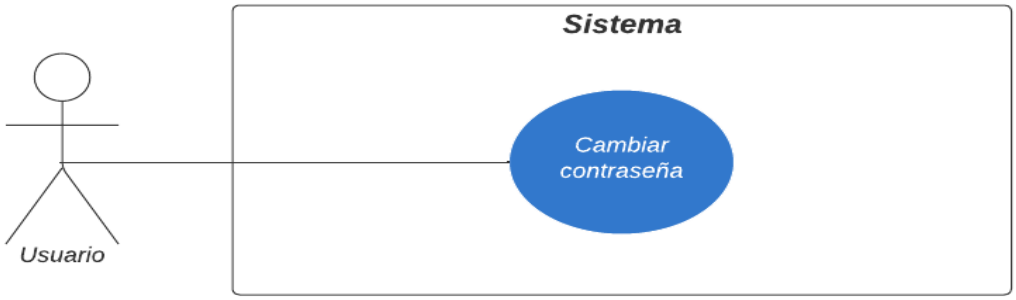


Precondición	**Postcondición**
El usuario debe haber iniciado sesión mediante correo y contraseña, y encontrarse en el fragmento de ‘ProfileFragment’	Al usuario se le indicara mediante una ventana emergente si se ha enviado correctamente el email para cambiar el correo electrónico o no.
Clases que intervienen	**Colecciones de base de datos afectadas**
ProfileFragment	
HomeViewModel	
HomeActivity	
Result	
AlertDialogFactory	
FirebaseAuth	
EmailValidator	

1- El usuario al pulsar sobre la opción de cambiar email, se le mostrará una ventana emergente en donde se le pedirá que vuelva a entrar sus credenciales para re autenticarse.

2- Si la re-autenticación no es exitosa, se le mostrará una ventana emergente indicando el error, en el caso contrario se le mostrará una ventana emergente en donde el usuario deberá introducir el nuevo email.
3- Este nuevo email se validará igualmente que, en el registro, y una vez que este sea correcto y el usuario pulse 'ok' se le enviará un correo a la dirección introducida en donde habrá un enlace desde donde el usuario puede confirmar el cambio de correo electrónico.
4- Durante la ejecución de la tarea de enviar el correo, se mostrará un Progress Bar indicando que se está realizando una tarea de larga duración.
5- Cuando termine la tarea desaparecerá el Progress Bar, y se le mostrará al usuario una ventana indicando si la tarea ha tenido éxito o no.

- Cambiar contraseña

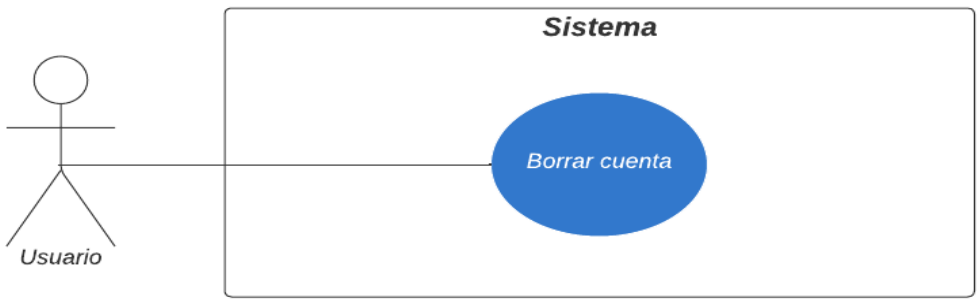


Precondición	Postcondición
El usuario debe haber iniciado sesión mediante correo y contraseña, y encontrarse en el fragmento de 'ProfileFragment'	Al usuario se le indicara mediante una ventana emergente si se ha cambiado la contraseña o no.
Clases que intervienen	**Colecciones de base de datos afectadas**
ProfileFragment	
HomeViewModel	
HomeActivity	

Result	
AlertDialogFactory	
FirebaseAuth	
PasswordValidator	

1- El usuario al pulsar sobre la opción de cambiar contraseña, se le mostrará una ventana emergente en donde se le pedirá que vuelva a entrar sus credenciales para re autenticarse.
2- Si la re-autenticación no es exitosa, se le mostrará una ventana emergente indicando el error, en el caso contrario se le mostrará una ventana emergente en donde el usuario deberá introducir la nueva contraseña.
3- Esta nueva contraseña se validará igualmente que, en el registro, una vez que esta sea correcta y el usuario pulse 'ok', se empezará a llevar a cabo la tarea de cambiar la contraseña.
4- Durante el proceso se mostrará un Progress Bar indicando que se esta llevando a cabo una tarea de larga duración.
5- Una vez que termine, desaparecerá el Progress Bar y se le mostrará al usuario una ventana emergente indicando si se ha cambiado o no la contraseña.

- **Borrar cuenta.**

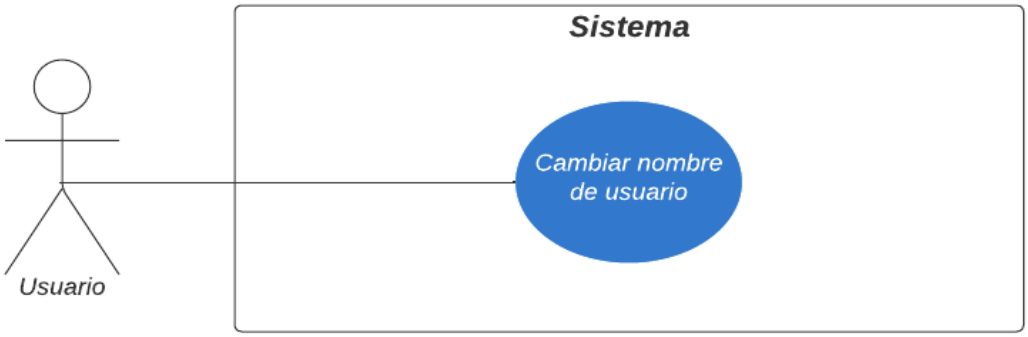


Precondición	Postcondición
El usuario debe haber iniciado sesión y encontrarse en el fragmento de 'ProfileFragment'	Al usuario se le indicara mediante una ventana emergente si se ha borrado su cuenta o no, y en caso de si se haya borrado será redirigido hacia el fragmennto de Login.
Clases que intervienen	**Colecciones de base de datos afectadas**
ProfileFragment	Users
HomeViewModel	Groups
HomeActivity	
Result	
AlertDialogFactory	
FirebaseAuth	
StorageRepository	
UsersRepository	
ProfileViewModel	
MainActivity	
LoginFragment	

1- El usuario al pulsar sobre la opción de borrar cuenta, se le mostrará una ventana emergente en donde se le pedirá que vuelva a entrar sus credenciales para re autenticarse.
2- Si la re-autenticación es incorrecta se le mostrará una ventana de error informando y se le permitirá volver a intentarlo. En el caso contrario, empezará el proceso de borrar la cuenta del usuario, el cual no solo implica borrar la cuenta propia del usuario, sino también borrar su presencia de la lista de amigos y de grupos del usuario.
3- Durante el proceso se mostrará un Progress Bar indicando que se esta llevando a cabo una tarea de larga duración.
4- Una vez termine se le mostrará una ventana emergente indicando si se ha borrado o no.

5- Si se ha borrado, el usuario será redirigido hacia el fragmento de Login.

- **Cambiar nombre de usuario.**

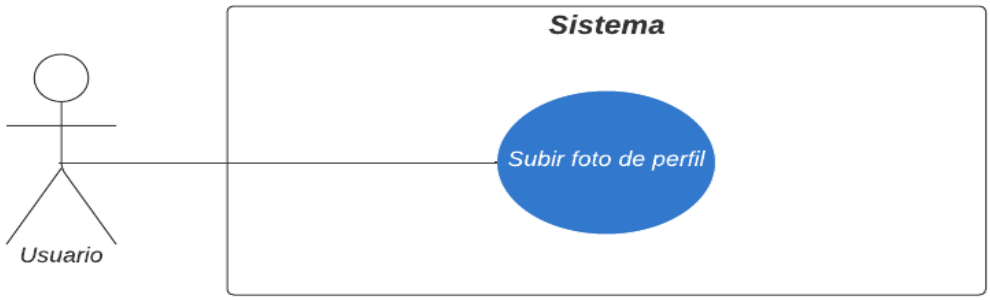


Precondición	**Postcondición**
El usuario debe haber iniciado sesión y encontrarse en el fragmento de 'ProfileFragment'	Al usuario se le indicara mediante una ventana emergente si se ha cambiado su nombre de usuario o no, además de mostrar el nuevo nombre debajo de la foto de perfil.
Clases que intervienen	**Colecciones de base de datos afectadas**
ProfileFragment	Users

HomeViewModel	
HomeActivity	
Result	
AlertDialogFactory	
UsersRepository	
ProfileViewModel	
UsernameValidator	

1- Al presionar sobre la opción de cambiar nombre de usuario, al usuario se le pedirá que introduzca un nuevo nombre de usuario mediante una ventana emergente.
2- Este nuevo nombre de usuario se validará de igual forma que en el registro, una vez que este sea correcto y el usuario presione 'ok' empezará el proceso de cambiar su nombre de usuario.
3- Durante el proceso se mostrará un Progress Bar indicando que se esta llevando a cabo una tarea de larga duración.
4- Una vez que termine, se el mostrará una ventana emergente al usuario indicando si se ha cambiado o no su nombre de usuario y se actualizará el nombre mostrado debajo de su foto de perfil.

- **Subir foto de perfil.**

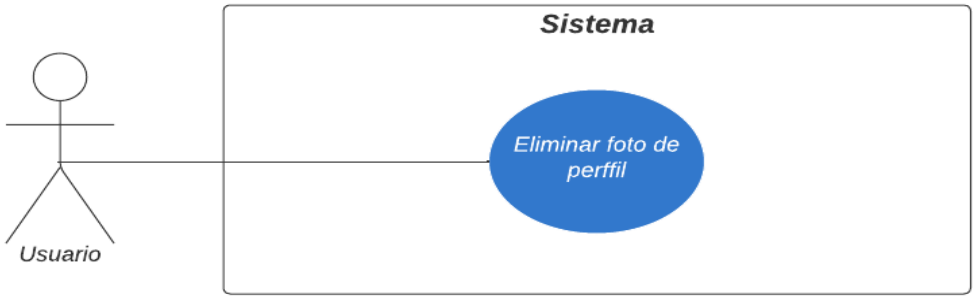


Precondición	Postcondición
El usuario debe haber iniciado sesión mediante correo y contraseña, y	Cambiara la foto de perfil mostrada en el fragmento.

encontrarse en el fragmento de 'ProfileFragment'	
Clases que intervienen	**Colecciones de base de datos afectadas**
ProfileFragment	Users
HomeViewModel	
HomeActivity	
Result	
AlertDialogFactory	
UsersRepository	
ProfileViewModel	
StorageRepository	
ImageLoader	

1- El usuario al presionar sobre el icono sobre la foto de perfil en la vista, se le mostrará una ventana emergente en donde se le mostrarán dos opciones desde donde escoger una foto, la galería o tomar una foto desde la cámara.
2- Si se escoge la galería, esta se abrirá para escoger una imagen, si se escoge la cámara, esta se abrirá para tomar una foto.
3- Cuando se haya seleccionado una foto, el usuario volverá a la aplicación con la foto seleccionada y se iniciará el proceso de subida de la foto.
4- Una vez termine el proceso, si es exitoso la foto de perfil mostrada se actualizará para mostrar la foto seleccionada por el usuario.
5- Si durante el proceso de seleccionar la foto, el proceso se interrumpe, el usuario volverá a la aplicación y se le indicará que ha fallado el proceso de subir la foto de perfil.

- **Eliminar foto de perfil.**

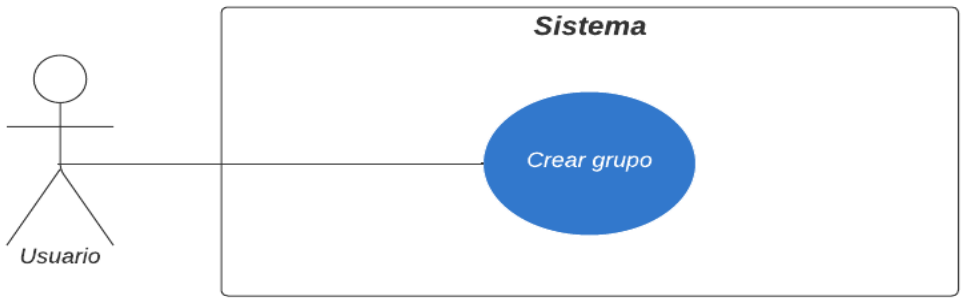


Precondición	Postcondición
El usuario debe haber iniciado sesión mediante correo y contraseña, y encontrarse en el fragmento de 'ProfileFragment'	Cambiará la foto de perfil mostrada en el fragmento a una por defecto.
Clases que intervienen	**Colecciones de base de datos afectadas**
ProfileFragment	Users
HomeViewModel	
HomeActivity	
Result	
AlertDialogFactory	
UsersRepository	
ProfileViewModel	
StorageRepository	

1- El usuario al presionar sobre icono sobre la foto de perfil, si el usuario ya tiene una foto de perfil, además de las demás opciones para cargar una nueva foto, aparecerá la opción de eliminar la foto ya establecida.
2- Al pulsar esta opción, empezará el proceso de eliminar la foto de perfil del usuario.
3- Durante el proceso, se mostrará un Progress Bar indicando que se está llevando a cabo una tarea de larga duración.

4- Una vez que termine el proceso de eliminar la foto de perfil, desaparecerá el Progress Bar y la foto de perfil de usuario cambiara a una foto por defecto indicando que no cuenta con foto de perfil.

- **Crear grupo.**



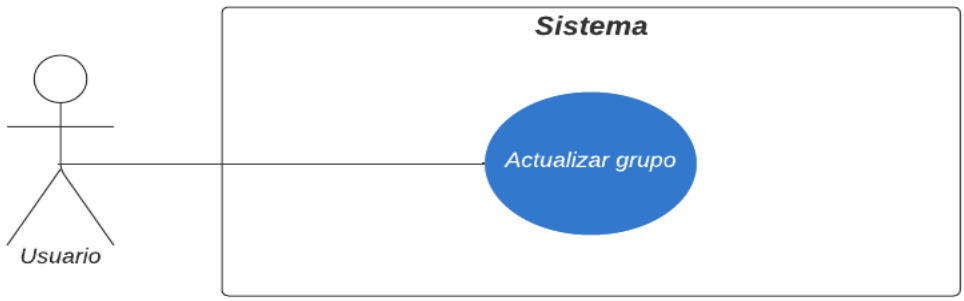
Precondición	Postcondición
El usuario debe haber iniciado sesión.	Si no ocurre ningún error, el usuario será redirigido hacia el fragmento 'GroupsFragment' en donde podrá ver el nuevo grupo creado. Si surge algún error, se le mostrará una ventana emergente informado de este.
Clases que intervienen	**Colecciones de base de datos afectadas**
NewGroupFragment	Users
HomeViewModel	Groups
HomeActivity	
Result	
AlertDialogFactory	
GroupsRepository	
NewGroupViewModel	
StorageRepository	
Group	

User	
GROUP_CATEGORY	
NewGroupFriendsAdapter	
NewGroupFriendViewHolder	
GroupDescriptionValidator	
GroupNameValidator	
Utilities	
GroupsFragment	
FriendsViewModel	
ImageLoader	

1. El usuario al presionar sobre el botón central en el menú de navegación será trasladado hacia el fragmento 'NewGroupFragment'.
2. allí el usuario será presentado con un formulario en donde se le pedirá que introduzca los datos necesarios para crear un nuevo grupo.
3. Estos datos son, nombre, descripción, categoría, fecha de inicio, fecha de fin, y aunque no es obligatorio, se puede marcar que amigos se desean invitar al crear el grupo para que se unan.
4. El usuario también puede subir una foto para identificar el grupo si así lo desea.
5. Cuando todos estos valores sean válidos, el usuario podrá presionar el botón de crear y empezara el proceso de creación.
6. Durante el proceso, se mostrará un Progress Bar indicando que se esta llevando a cabo una tarea de larga duración.
7. El proceso de creación de grupo implica no solo crear el objeto de grupo dentro de la colección de 'Groups' en la base de datos, sino también implica crear la referencia a dicho grupo dentro del usuario que lo crea, así como también crear las invitaciones en cada uno de los amigos que se hayan seleccionado, y finalmente subir la foto al Firebase Storage.
8. Una vez que este proceso haya finalizado, el Progress Bar desaparecerá y se le indicará al usuario el resultado de la operación mediante una ventana emergente.

9. Si el proceso ha sido exitoso, el usuario al quitar la ventana emergente será redirigido hacia el fragmento de 'Groups' en donde podrá ver el nuevo grupo. Si no fue exitoso, el usuario lo podrá volver a intentar al quitar la ventana emergente.

- **Actualizar grupo.**



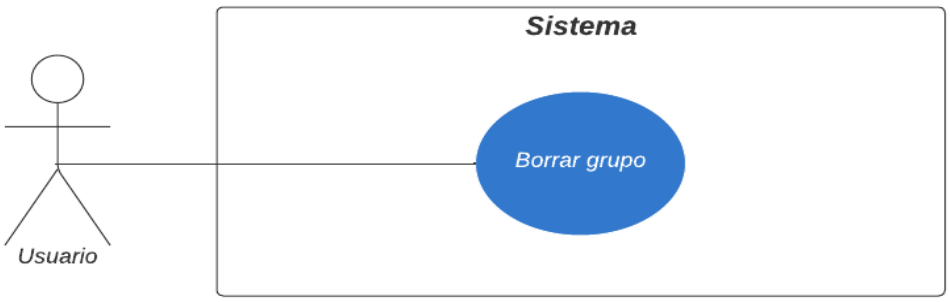
Precondición	Postcondición
El usuario debe haber iniciado sesión y debe haber seleccionado un grupo ya existente en el fragmento 'GroupsFragment'.	Si no ocurre ningún error, el usuario será redirigido hacia el fragmento 'GroupsFragment' en donde podrá ver el grupo actualizado. Si surge algún error, se le mostrará una ventana emergente informado de este.
Clases que intervienen	**Colecciones de base de datos afectadas**
GroupOverviewFragment	Users

HomeViewModel	Groups
HomeActivity	
Result	
AlertDialogFactory	
GroupsRepository	
NewGroupViewModel	
StorageRepository	
Group	
User	
GROUP_CATEGORY	
NewGroupFriendsAdapter	
NewGroupFriendViewHolder	
GroupDescriptionValidator	
GroupNameValidator	
Utilities	
GroupsFragment	
FriendsViewModel	
ImageLoader	

1. El usuario el presionar sobre un grupo que ya exista en el fragmento de es ‘Groups’, el usuario será redirigido hacia el fragmento ‘DetailsFragment’ en el cual en la parte de arriba aparecerán la foto y el nombre el grupo.
2. Al presionar sobre esa parte de la vista, el usuario será redirigido nuevamente hacia el fragmento ‘GroupOverViewFragment’ pasando el grupo seleccionado como argumento.
3. Una vez allí, la vista será la misma que en el fragmento de creación de grupos, pero esta vez con más acciones disponibles y los datos del grupo ya cargados en los campos.
4. En función del rol que ocupe el usuario dentro del grupo, este podrá modificar cualquier dato del grupo y eliminar miembros si es administrador o solo invitar nuevos miembros si es miembro.

5. Una vez que el usuario haya realizado las modificaciones que desee y se hayan validado el usuario podrá presionar el botón de actualizar y al hacerlo empezará la tarea de actualizacion del grupo.
6. Durante el proceso se mostrará un Progress Bar indicando que se está llevando a cabo una tarea de larga duración.
7. La operación de actualizacion del grupo, implica cambiar cualquier dato que haya sido modificado, cambiar o eliminar la foto del grupo en el caso que corresponda, así como eliminar miembros que hayan sido deseleccionados e invitar aquellos amigos que hayan sido seleccionados.
8. Una vez que la tarea de actualizacion termine, el Progress Bar desaparecerá y se le informará al usuario del resultado de la operación mediante una ventana emergente.
9. Si fue exitosa la operación, el usuario al quitar la ventana emergente, será redirigido hacia el fragmento 'Groups' en donde podrá ver el grupo actualizado. En el caso contrario, al usuario se le permitirá volver a intentarlo informado del error que haya ocurrido.

- **Borrar grupo.**

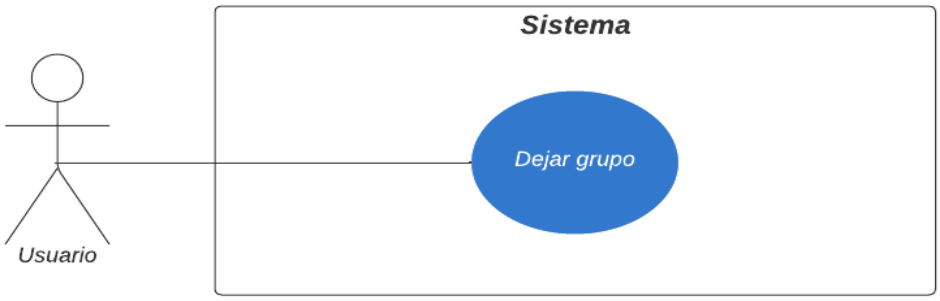


Precondición	Postcondición
El usuario debe haber iniciado sesión y debe haber seleccionado un grupo ya existente en el fragmento	Si no ocurre ningún error, el usuario será redirigido hacia el fragmento 'GroupsFragment' en donde ya no

‘GroupsFragment’ y tener el rol de administrador dentro del grupo.	aparecerá el grupo. Si surge algún error, se le mostrará una ventana emergente informado de este.
Clases que intervienen	**Colecciones de base de datos afectadas**
GroupOverviewFragment	Users
HomeViewModel	Groups
HomeActivity	
Result	
AlertDialogFactory	
GroupsRepository	
NewGroupViewModel	
StorageRepository	
Group	
User	
GroupsFragment	

1. El usuario, el fragmento de ‘GroupOverviewFragment’, en el caso de que tenga el rol de administrador dentro grupo, además de poder actualizar los datos del grupo, tendra disponible la opción borrar el grupo en su totalidad.
2. Al presionar sobre esta opción, empezará la tarea de eliminar el grupo, lo cual implica la eliminación de la referencia al grupo dentro de cada uno de los miembros actuales del grupo, así como la eliminación de la foto del grupo, en el caso de que exista.
3. Una vez que termine el proceso de eliminación, si no ha ocurrido ningún error, el usuario al quitar la ventana emergente, será redirigido hacia el fragmento ‘GroupsFragment’, en donde ya no aparecerá el grupo eliminado.
4. En el caso de que haya ocurrido algún error, se le informará al usuario mediante una ventana emergente, y se le permitirá volver a intentarlo.

- **Dejar grupo.**

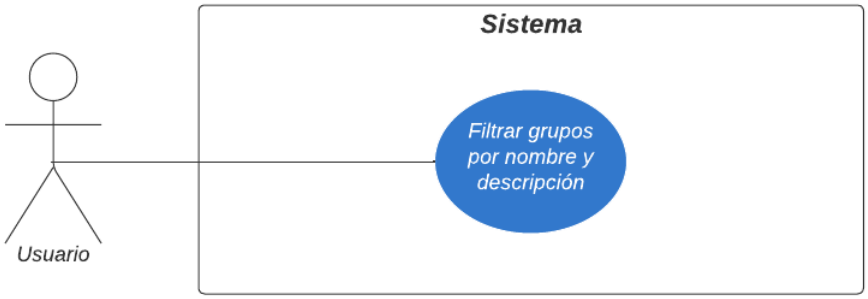


Precondición	**Postcondición**
El usuario debe haber iniciado sesión y debe haber seleccionado un grupo ya existente en el fragmento 'GroupsFragment'.	Si no ocurre ningún error, el usuario será redirigido hacia el fragmento 'GroupsFragment' en donde ya no aparecerá el grupo. Si surge algún error, se le mostrará una ventana emergente informando de este.
Clases que intervienen	**Colecciones de base de datos afectadas**

GroupOverviewFragment	Users
HomeViewModel	Groups
HomeActivity	
Result	
AlertDialogFactory	
GroupsRepository	
NewGroupViewModel	
Group	
GroupsFragment	

1. El usuario, en el fragmento de 'GroupOverviewFragment', independientemente del rol que tenga, tendrá la opción de dejar el grupo, al presionar sobre esta opción empezará la tarea.
2. Esta tarea implica la eliminación de la referencia del usuario como miembro del grupo, así como la eliminación de la referencia al grupo dentro del usuario.
3. Durante el proceso se mostrará un Progress Bar indicando que se está llevando a cabo la tarea.
4. Una vez que termine la tarea, se le informará al usuario del resultado de la tarea mediante una ventana emergente.
5. Si la tarea ha sido exitosa, el usuario será redirigido hacia el fragmento 'Groups' en donde ya no le aparecerá el grupo que acaba de dejar.
6. Si la tarea no ha sido exitosa, al usuario se le informará del error mediante una ventana emergente y se le permitirá volver a intentarlo.

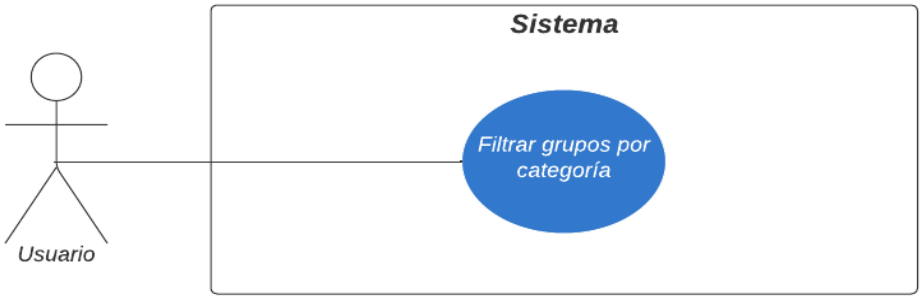
- **Filtrar grupos por nombre y descripción.**



Precondición	Postcondición
El usuario debe haber iniciado sesión y pertenecer por lo menos a un grupo.	Se mostrarán únicamente aquellos grupos que cumplan el filtro.
Clases que intervienen	**Colecciones de base de datos afectadas**
GroupsFragment	
GroupsAdapter	
HomeActivity	
SearchView.OnQueryTextListener	
Group	
GroupsFragment	
ListItemUiModel.Group	
GroupViewHolder	

1. El usuario en el fragmento 'GroupsFragment' tendrá un SearchView desde el cual podrá escribir para filtrar los grupos que se muestran.
2. Cada vez que escribe el usuario, se procederá a filtrar los grupos, para esto primero se recogerá el texto introducido por el usuario.
3. Luego, del adaptador se eliminarán todos aquellos grupos cuyo nombre ni descripción contenga lo que el usuario haya introducido.
4. Cuando el usuario deje vacío el SearchView, en el adaptador se volverán a incluir todos los grupos existentes.
5.

- **Filtrar grupos por categorías.**



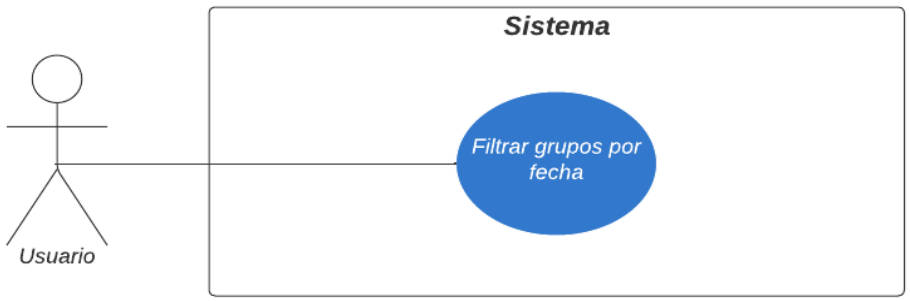
Precondición	Postcondición
El usuario debe haber iniciado sesión y pertenecer por lo menos a un grupo.	Se mostrarán únicamente aquellos grupos que cumplan el filtro.
Clases que intervienen	**Colecciones de base de datos afectadas**
GroupsFragment	
GroupsAdapter	
HomeActivity	
GROUP_CATEGORY	
Group	
GroupCategoryAdapter	
Utilities	
GroupCategoryViewHolder	
ListItemUiModel.CategoryUiModel	
ListItemUiModel.Group	
GroupViewHolder	

1. El usuario en el fragmento ‘GroupsFragment’, además del SearchView que le permite filtrar por nombre, también cuenta con una lista horizontal en donde se muestran las categorías a las cuales puede pertenecer un grupo.
2. El usuario al presionar sobre una categoría hace que esta quede seleccionada y se use para filtrar los grupos, resultando en que solo se

muestren aquellos grupos cuya categoría sea por lo menos una de las seleccionadas.

3. El usuario al volver a presionar sobre alguna categoría que ya se encuentre seleccionada, este dejará de estarlo y ya no se tendrá en cuenta para filtrar los grupos.
4. Una vez que ya no haya ninguna categoría seleccionada, se volverán a mostrar todos los grupos a los que pertenece el usuario.

- **Filtrar grupos por fecha.**

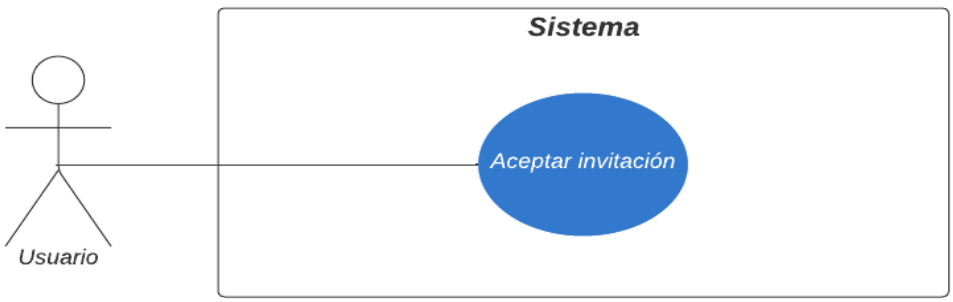


Precondición	**Postcondición**
El usuario debe haber iniciado sesión y pertenecer por lo menos a un grupo.	Se mostrarán únicamente aquellos grupos que cumplan el filtro.
Clases que intervienen	**Colecciones de base de datos afectadas**
GroupsFragment	
GroupsAdapter	
HomeActivity	
HomeFragment	
Group	
CalendarFragment	
LocalDateTime	
ListItemUiModel.Group	
GroupViewHolder	

CalendarAdapter	
CalendarViewHolder	
ListItemUiModel.CalendarDayUiModel	

1. El usuario en la vista de 'HomeFragment', se encontrará con un calendario posicionado en el mes actual, en él se mostrarán con un círculo relleno los días en los cuales haya algún grupo.
2. Cuando el usuario pulse sobre alguno de estos días, este será redirigido hacia el fragmento 'GroupsFragment', en donde se le mostrarán solo aquellos grupos cuya fecha de inicio y fin, incluyan el día pulsado.

- **Aceptar invitación.**

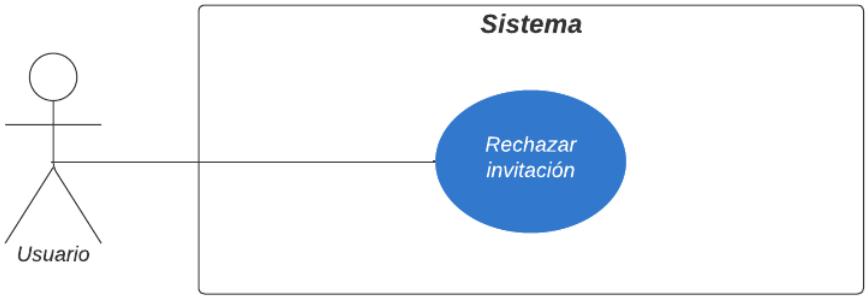


Precondición	Postcondición
El usuario debe haber iniciado sesión y tener alguna invitación ya sea de amistad o de grupo.	El usuario se volverá amigo del usuario que lo haya invitado, si la invitación era de amistad, y se unirá al

	grupo al cual lo hayan invitado si era una invitación de grupo.
Clases que intervienen	**Colecciones de base de datos afectadas**
InvitationsFragment	Users
InvitationsAdapter	Groups
HomeActivity	
HomeViewModel	
InvitationUiModel	
ListItemUiModel.Invitation	
InvitationsViewModel	
InvitationsRepository	
InvitationViewHolder	

1. El usuario en cualquier parte de la aplicación tendrá en la parte superior siempre visible un icono de campana, que, al pulsar sobre él, será redirigido hacia el fragmento de invitaciones (InvitationsFragment) en donde se cargarán las invitaciones que este tenga disponible.
2. Cuando el usuario pulse sobre la opción de aceptar la invitación, se determinará el tipo de esta, el cual puede ser de amistad o de grupo.
3. En el caso de que sea de amistad, al aceptarla el usuario que ha pulsado y el usuario que ha enviado la invitación, se convertirán en amigos.
4. En el caso de la invitación sea para unirse a un grupo, al aceptarla, el usuario se convertirá en miembro de este y a partir de ese momento, le aparecerá dicho grupo en el fragmento de grupos (GroupsFragment).

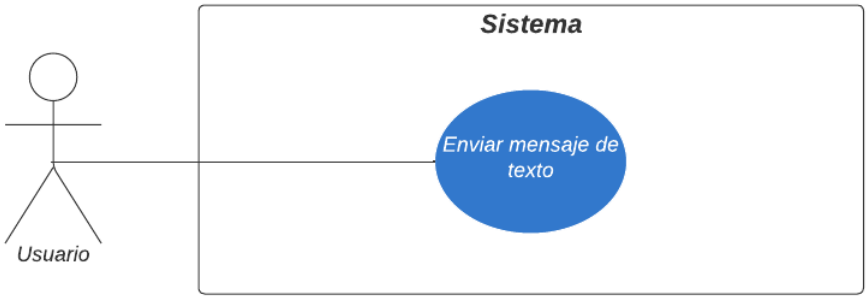
- **Rechazar invitación.**



Precondición	Postcondición
El usuario debe haber iniciado sesión y tener alguna invitación ya sea de amistad o de grupo.	La invitación será eliminada de la base de datos y desaparecerá del fragmento de invitaciones (InvitationsFragment)
Clases que intervienen	**Colecciones de base de datos afectadas**
InvitationsFragment	Users
InvitationsAdapter	
HomeActivity	
HomeViewModel	
InvitationUiModel	
ListItemUiModel.Invitation	
InvitationsViewModel	
InvitationsRepository	
InvitationViewHolder	

1. El usuario, al pulsar sobre la opción de rechazar sobre alguna invitación que haya recibido, simplemente iniciará el proceso de eliminación de dicha invitación en la base de datos.
2. Una vez que termine la eliminación, como la lista mostrada en la vista, se actualiza en tiempo real, la invitación automáticamente desaparecerá de esta.

- **Enviar mensaje de texto.**

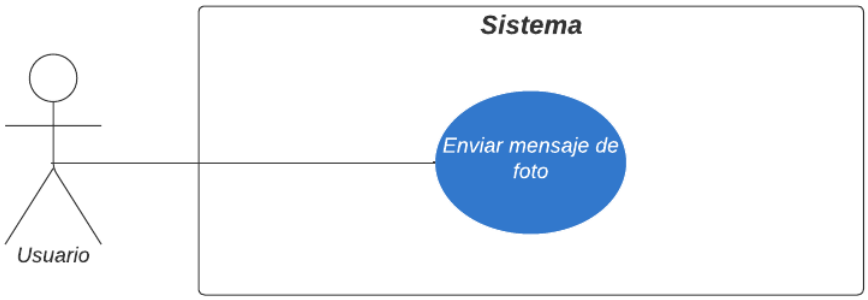


Precondición	Postcondición
El usuario debe haber iniciado sesión y ser miembro del grupo y haberse trasladado hacia el fragmento de chat (ChatFragment).	El mensaje se guardará en la base de datos y se cargará en la vista.
Clases que intervienen	**Colecciones de base de datos afectadas**
ChatFragment	Groups
MessageAdapter	
HomeActivity	
HomeViewModel	
Message	
ListItemUiModel.MessageUiModel	
ChatViewModel	
MessageRepository	
MessageViewHolder	
ListItemImageLoader	

1. El usuario una vez que se encuentre el fragmento (ChatFragment), será presentado con aquellos mensajes que existan en el grupo, ordenados para mostrar los recientes primero.

2. En la vista el usuario tendrá un campo en donde podrá escribir y un botón que al presionar validará que el mensaje no este vacío y tampoco que sea muy largo.
3. Cuando el mensaje sea válido y el usuario presione el botón de enviar, se iniciará la tarea de subir el mensaje a la base de datos, guardando cuando se envió y quien lo hizo.
4. Cuando termine de subirse el mensaje, como los mensajes se actualizan en tiempo real, el mensaje aparecerá automáticamente en la vista, mostrando la fecha de envío y los datos del usuario que lo envió.

- **Enviar mensaje de foto.**

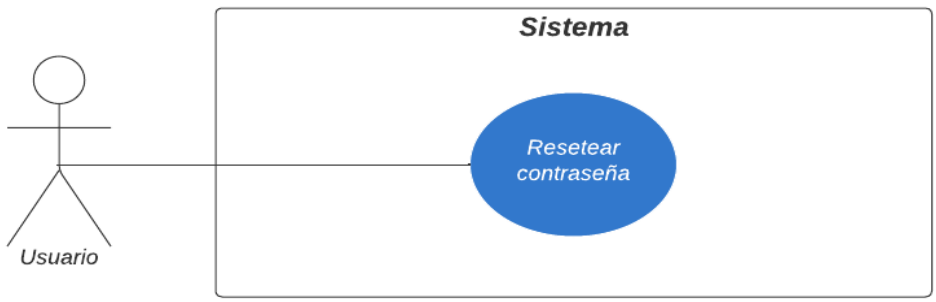


Precondición	Postcondición
El usuario debe haber iniciado sesión y ser miembro del grupo y haberse trasladado hacia el fragmento de chat (ChatFragment).	El mensaje se guardará en la base de datos y se cargará en la vista.
Clases que intervienen	**Colecciones de base de datos afectadas**
ChatFragment	Groups
MessageAdapter	
HomeActivity	
HomeViewModel	

Message	
ListItemUiModel.MessageUiModel	
ChatViewModel	
MessageRepository	
MessageViewHolder	
ListItemImageLoader	
ImageLoader	
StorageRepository	

1. Además de poder enviar mensaje de texto, el usuario contará en el fragmento de chat (ChatFragment) con un botón con un icono de clip, que al pulsarlo se le preguntará con una ventana emergente desde donde quiere cargar una foto, la galería o la cámara.
2. Una vez que el usuario haya seleccionado una imagen, este se cargará a Firebase Storage y una vez que se haya subido se creara un mensaje con una referencia a dicha imagen.
3. Cuando se termine este proceso, el mensaje automáticamente se mostrará en la vista, cargando la imagen para que sea visible para los miembros.

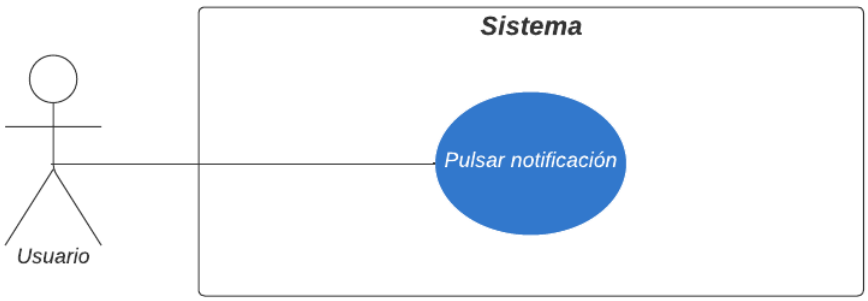
- **Solicitar cambio de contraseña.**



Precondición	Postcondición
El usuario no debe haber iniciado sesión y encontrarse en el fragmento de Login	Se enviará un correo a la dirección proporcionada para cambiar la contraseña de la cuenta.
Clases que intervienen	**Colecciones de base de datos afectadas**
LoginFragment	
FirebaseAuth	
MainActivity	
AlertDialogFactory	
PromptResult	

1. El usuario al entrar en la aplicación se encontrará con el formulario para iniciar sesión, en caso de que este no recuerde su contraseña podrá pulsar sobre la opción para solicitar un cambio.
2. Al pulsar sobre esta opción, se le mostrará una ventana emergente desde donde este podrá introducir la dirección de correo de su cuenta.
3. Una vez introducida la introducida, si la dirección se corresponde con una cuenta registrada, se enviará el mail con un enlace desde donde el usuario podrá cambiar su contraseña.

- ## Pulsar notificación

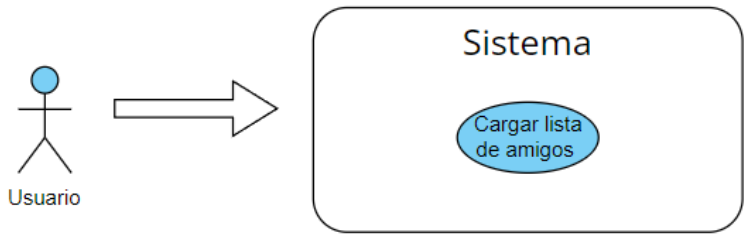


Precondición	Postcondición

El usuario debe encontrarse fuera de la aplicación y pertenecer a un grupo en el cual se envía un mensaje.	La aplicación se abrirá
Clases que intervienen	**Colecciones de base de datos afectadas**
FirebaseMessagingService	
SendMessageRequest	
SubscribeRequest	
UnsubscribeRequest	
Route	
RetrofitModules	
FcmApi	
ChatFragment	

1- Un usuario dentro de un grupo en el cual hay otros miembros aparte de él, envía un mensaje dentro del grupo.
2- Cualquier otro usuario que pertenezca al mismo grupo y no se encuentre dentro de la aplicación recibirá una notificación informándole del mensaje recibido.
3- El usuario al pulsar sobre dicha notificación hará que se abra la aplicación.

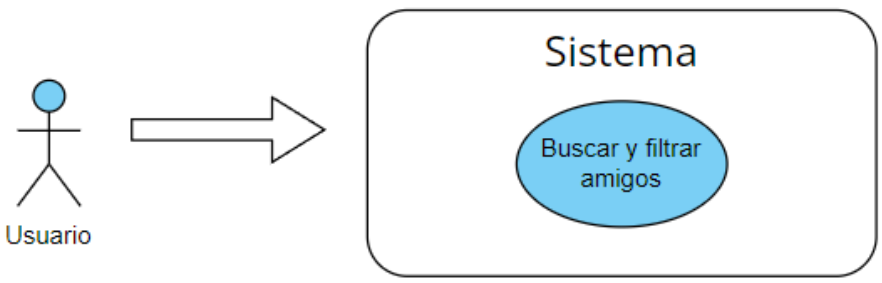
- **Cargar lista de amigos.**



Precondición	**Postcondición**
El usuario debe haber iniciado sesión y debe haberse dirigido a la ventana 'Friends', la cual carga el 'FriendsFragment'.	Si no ocurre ningún error, la lista de amigos se muestra en el RecyclerView.
Clases que intervienen	**Colecciones de base de datos afectadas**
FriendsFragment	Groups/Friends
FriendsAdapter	
StorageRepository	
User	
ImageLoader	

1. El usuario abre el FriendsFragment.
2. La aplicación carga los amigos del usuario desde Firebase.
3. La lista de amigos se muestra en el 'RecyclerView' utilizando 'FriendsAdapter'.
4. Si no hay amigos, la lista se muestra vacía.

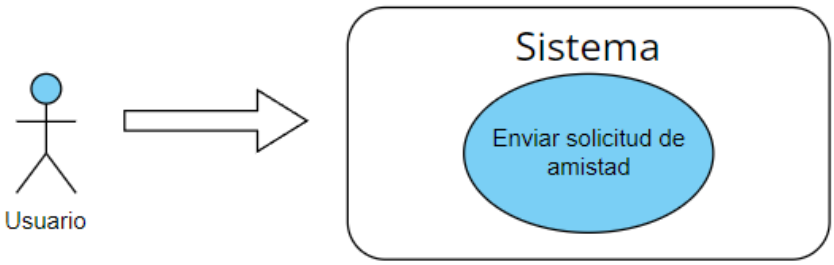
- **Buscar y filtrar amigos.**



Precondición	**Postcondición**
El usuario debe haber iniciado sesión, encontrarse en el fragmento 'FriendsFragment' y la lista debe haber sido cargada.	La lista de amigos se filtra según el texto de búsqueda introducido.
Clases que intervienen	**Colecciones de base de datos afectadas**
FriendsFragment	
FriendsAdapter	
StorageRepository	
User	
ImageLoader	

1. El usuario introduce el texto en el SearchView.
2. La aplicación filtra la lista de amigos según el texto introducido utilizando el método 'filter' en 'FriendsAdapter'.
3. La lista de amigos se actualiza en el RecyclerView.
4. Si no hay coincidencias, la lista se muestra vacia.
5.

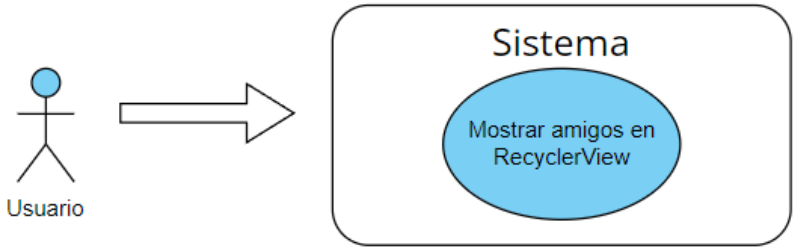
- **Enviar solicitud de amistad.**



Precondición	**Postcondición**
El usuario debe haber iniciado sesión, encontrarse en el fragmento 'FriendsFragment' y haber presionado el botón de añadir amigo, haber escrito el nombre de dicho usuario y posteriormente haber pulsado "OK".	Si el usuario especificado se encuentra en la base de datos, se envía una solicitud de amistad al mismo.
Clases que intervienen	**Colecciones de base de datos afectadas**
FriendsFragment	Users/Invitations
DialogFragment	

1. El usuario pulsa el botón para añadir un amigo.
2. La aplicación muestra una ventana emergente ('DialogFragment') para introducir el nombre de usuario.
3. El usuario introduce el nombre de usuario y pulsa "OK".
4. La aplicación utiliza 'FirebaseDatabaseHelper' para enviar una solicitud de amistad al usuario especificado.
5. Si el nombre de usuario no existe, se envía un mensaje de error.

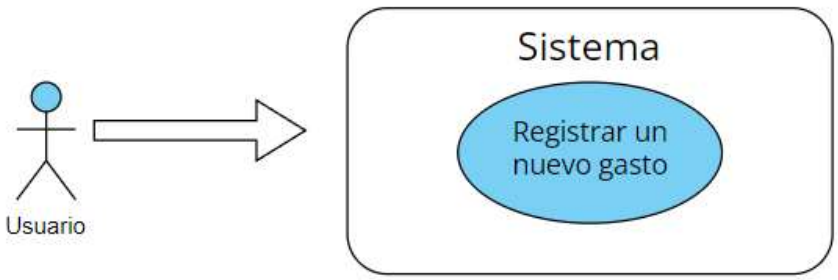
- **Mostrar amigos en RecyclerView.**



Precondición	**Postcondición**
El usuario se encuentra en el fragmento ‘FriendsFragment’ y la lista de amigos ha sido cargada.	La lista de amigos se muestra correctamente en el ‘RecyclerView’
Clases que intervienen	**Colecciones de base de datos afectadas**
FriendsFragment	Ninguna(datos ya cargados)
FriendsAdapter	

1. El usuario abre el ‘FriendsFragment’.
2. La lista de amigos se muestra en el ‘RecyclerView’ utilizando ‘FriendsAdapter’.
3. El usuario puede hacer “scroll” para ver todos los amigos.

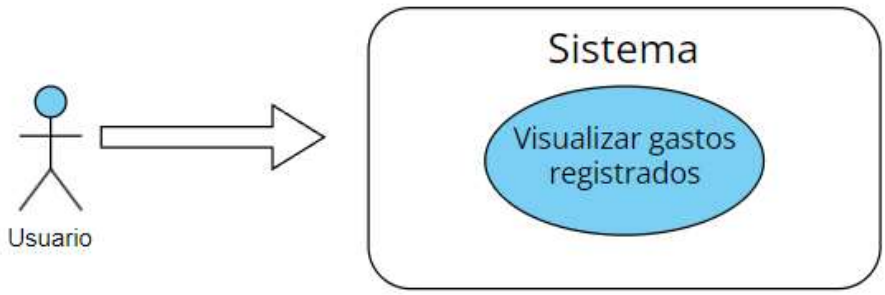
- **Registrar un nuevo gasto**



Precondición	**Postcondición**
El usuario ha iniciado sesión y se encuentra en el fragmento 'NewExpenseFragment'.	El nuevo gasto se guarda en el repositorio y aparece en la lista de gastos registrados.
Clases que intervienen	**Colecciones de base de datos afectadas**
NewExpenseFragment	Expenses
ExpenseRepository	
NewExpenseViewModel	
NewExpenseTitleValidator	
HomeViewModel	
HomeActivity	
Result	
AlertDialogFactory	
StorageRepository	
Expense	
NewExpenseAdapter	
NewExpenseViewHolder	
ExpenseFragment	
ExpenseViewModel	

1. El usuario navega hasta el fragmento 'NewExpenseFragment'.
2. El usuario completa los campos del formulario con la información del gasto (nombre, descripción, cantidad, fecha).
3. El usuario selecciona la fecha del gasto mediante una ventana emergente.
4. El usuario envía el formulario.
5. El sistema valida los campos del formulario.
6. El sistema guarda el nuevo gasto en el repositorio.
7. El sistema muestra un mensaje de confirmación indicando que el gasto ha sido registrado correctamente.
8. El sistema muestra el nuevo gasto en la lista de gastos registrados.

- **Visualizar gastos registrados**



Precondición	**Postcondición**
El usuario ha iniciado sesión y se encuentra en el fragmento 'ExpenseFragment.	El nuevo gasto se guarda en el repositorio y aparece en la lista de gastos registrados.
Clases que intervienen	**Colecciones de base de datos afectadas**
ExpenseFragment	Expenses
ExpenseRepository	
ExpenseViewModel	

1. El usuario navega hasta el fragmento 'ExpenseFragment'.
2. El sistema carga los gastos registrados desde el repositorio. El usuario selecciona la fecha del gasto mediante una ventana emergente.
3. El usuario envía el formulario.
4. El usuario puede ver los detalles de cada gasto (nombre, descripción, fecha, cantidad) en la lista. El sistema guarda el nuevo gasto en el repositorio.

MANUAL DE USUARIO

1. Se debe introducir el correo y la contraseña de una cuenta existente.
2. Si se olvida la contraseña, al pulsar aquí se le enviará un mail al correo de su cuenta para cambiarla.
3. Al pulsar aquí puede iniciar sesión mediante una cuenta de Google directamente.
4. Si aún no tiene una cuenta, al pulsar aquí será llevado al formulario para hacerlo.

1. Se deben rellenar todos los campos hasta que no se muestren errores.
2. Si se pulsa aquí se mostrarán los requisitos que debe cumplir la contraseña.
3. Al pulsar aquí se volverá a la vista de inicio de sesión.
4. Al pulsar aquí se validarán los datos introducidos y comenzará el proceso de registro.

1. Si se inicia sesión con Google y no existen datos dentro de la base de datos, los deberá introducir aquí y al pulsar empezará el proceso de registro.

1. Al pulsar aquí irá a la vista de Home.
2. Al pulsar aquí irá a la vista de Grupos.
3. Al pulsar aquí irá a la vista de amigos.
4. Al pulsar aquí irá a la vista de perfil.
5. Al pulsar aquí irá a la vista de creación de un nuevo grupo.
6. Al pulsar aquí irá a la vista de invitaciones.
7. En este calendario se mostrarán rellenos aquellos días en los que haya un grupo, se puede avanzar y retroceder el mes y al pulsar sobre un día será llevado a la vista de grupos mostrando solo aquellos grupos que ocurran en dicha fecha.

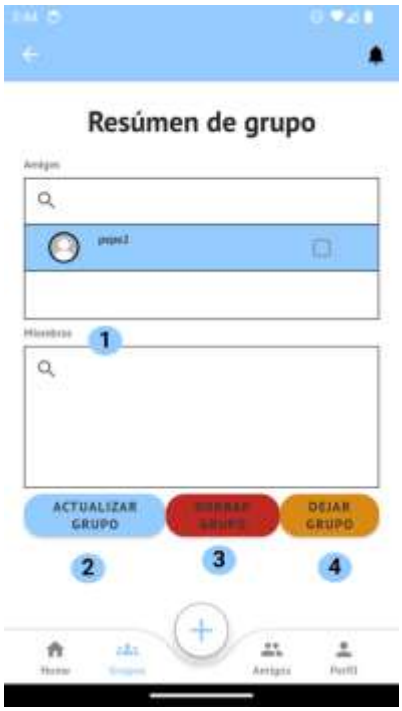
1. Al pulsar aquí podrás seleccionar una foto para establecerla como foto de perfil.
2. Aquí podrás seleccionar una categoría con la cual identificar tu grupo.
3. Deberás escoger tanto una fecha de inicio como una de fin para el grupo, pueden ser la misma si ocurre en un día.
4. Desde aquí podrás seleccionar que amigos quieres invitar al crear el grupo.
5. Al pulsar aquí comenzará el proceso de creación del grupo, y al finaliza serás llevado a la vista de grupos donde estará el grupo creado.

1. Al escribir aquí se filtrarán los grupos que se muestran, mostrando solo aquellos cuyo nombre o descripción coincida con lo introducido.
2. Al pulsar sobre las categorías, esta se seleccionará filtrando los grupos, mostrando solo aquellos cuya categoría sea alguna de las seleccionadas.
3. Al pulsar sobre un grupo, se le llevará a vista de detalles en donde se mostrarán los detalles de dicho grupo.

1. Al pulsar aquí será llevado a la vista desde donde podrá actualizar los datos del grupo si es administrador, o invitar amigos y es miembro.
2. Al pulsar aquí será llevado a la vista de chat del grupo, desde donde podrá comunicarse con los demás miembros.

GroupOverviewFragment

Este fragmento es exactamente igual que el fragmento de creación de un grupo nuevo, solo que ahora los datos están cargados, por lo cual el funcionamiento es igual.



1. Aquí aparecerán los miembros del grupo mostrando su rol, los Administradores podrán cambiar el rol de estos, así como eliminarlos.
2. Al pulsar aquí se harán efectivos los cambios realizados a la información del grupo.
3. Al pulsar aquí se borrará completamente el grupo.
4. Al pulsar aquí se procederá al dejar el grupo.

ChatFragment

1. Al pulsar aquí se podrá escoger desde donde cargar una foto para mandarla.
2. Al pulsar aquí se enviará un mensaje con el texto escrito dentro del campo.
3. Si se pulsa sobre un mensaje con una foto, se abrirá esta en pantalla completa para poder ver toda la foto.

ProfileFragment

1. Al pulsar aquí se podrá seleccionar una foto para subirla como foto de perfil.
2. Al pulsar esta opción se cerrará la sesión actual.
3. Al pulsar sobre esta opción, se podrá introducir un nuevo correo para asociarlo a la cuenta.
4. Al pulsar sobre esta opción, se podrá cambiar la contraseña actual por otro.
5. Al pulsar sobre esta opción se procederá con la eliminación de la cuenta.
6. Al pulsar sobre esta opción, se podrá cambiar el nombre de usuario por otro.

DIAGRAMA DE GANTT

R01:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R01F01T01-Diseño básico de registro	10
Armando Guzmán	R01F01T02-Crear y configurar proyecto en Firebase	30
Armando Guzmán	R01F01T03-Integrar proyecto de Firebase con proyecto de Android	20
Armando Guzmán	R01F01T04-Crear clase base de todas las validaciones	4
Armando Guzmán	R01F01T05-Crear clases de validaciones que implementen la clase base	10
Armando Guzmán	R01F01T06-Definir clase abstracta de la cual heredarán las demás clases validadoras.	4
Armando Guzmán	R01F01T07-Crear clases de validaciones para cada campo	8
Armando Guzmán	R01F01T08-Crear campos públicos y privados para cada campo	10
Armando Guzmán	R01F01T09-Crear módulos que suministren los validadores	6
Armando Guzmán	R01F01T10-Implementar databinding con vista de Registro	5
Armando Guzmán	R01F01T11-Implementar métodos para validar cada campo	10
Armando Guzmán	R01F01T12-Implementar métodos de navegación	3
Armando Guzmán	R01F01T13-Crear eventos para realizar validaciones en cada campo al escribir	5

Armando Guzmán	R01F01T14-Crear método para crear cuenta con correo y contraseña	10
Armando Guzmán	R01F01T15-Crear clase modelo de usuario	5
Armando Guzmán	R01F01T16-Crear clase repositorio de usuarios	10
Armando Guzmán	R01F01T17-Crear método en ViewModel usando el repositorio para crear un nuevo usuario	14
Armando Guzmán	R01F01T18-Añadir progressBar y hacer que sea visible cuando se está procesando el registro	25
Armando Guzmán	R01F01T19-Crear método en repositorio de usuarios para obtener un usuario mediante el nombre de usario	10
Armando Guzmán	R01F01T20-Añadir validación a ViewModel para validar que el nombre de usuario no existe ya	12
Armando Guzmán	R01F01T21-Crear clase que representa el resultado de una tarea	10
Armando Guzmán	R01F01T22-Crear diseño de alertDialog para éxito y error	30
Armando Guzmán	R01F01T23-Implementar lógica para mostrar alertDialog de éxito o error en registro	35

R02:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R02F01T01-Diseño basico de Login	20

Armando Guzmán	R02F01T02-Implementar métodos para el inicio de sesion mediante correo y contraseña	30
Armando Guzmán	R02F01T04-Implementar un ProgressBar para indicar que se está llevando a cabo la tarea de inicio de sesión.	15
Armando Guzmán	R02F01T05-Implementar lógica para mostrar alertDialog de éxito o error en Login	40
Armando Guzmán	R02F01T06-Implementar lógica para navegar al Home una vez se ha iniciado sesión.	10
Armando Guzmán	R02F01T07-Implementar lógica para poder navegar al fragmento de registro.	5
Armando Guzmán	R02F02T01-Añadir botón en la vista de Login para iniciar sesión mediante cuenta de Google.	4
Armando Guzmán	R02F02T02-Crear métodos para iniciar sesión con cuenta de Google	20
Armando Guzmán	R02F02T03-Vincular evento de botón al método para iniciar sesión con Google.	9
Armando Guzmán	R02F02T04-Implementar lógica para mostrar el ProgressBar durante el procesamiento del inicio sesión con Google.	10
Armando Guzmán	R02F02T05-Implementar lógica para mostrar ventana emergente cuando termine el proceso de inicio de sesión.	10
Armando Guzmán	R02F02T06-Implementar validación para determinar si ya existen datos asociados a la cuenta con la cual se ha iniciado sesión o no.	12
Armando Guzmán	R02F02T07-Crear fragmento auxiliar para guardar datos personales al iniciar sesión con Google	50
Armando Guzmán	R02F02T08-Validar y recoger los datos introducidos en el formulario de ‘PersonalData’ y crear el objeto de usuario dentro de la base de datos con RegisterViewModel.	25

Armando Guzmán	R02F02T09-Implementar progressBar en 'PersonalDataFragment' para indicar que se está realizando la tarea de crear el usuario en la base de datos.	12
Armando Guzmán	R02F02T10-Implementar lógica para mostrar ventana emergente cuando termine la tarea informando si se ha creado o no el usuario en la base de datos.	9
Armando Guzmán	R02F02T11-Implementar navegación para que el usuario sea redirigido al 'PersonalDataFragment' cuando se valide que no hay datos guardados en la base de datos asociados a su cuenta de Google, o redirigido al Activity de Home en caso de que si los haya.	8
Armando Guzmán	R02F02T12-Implementar navegación en 'PersonalDataFragment' para que el usuario sea redirigido al Activity Home una vez se haya creado el usuario en la base de datos.	10

R03:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R03F01T01-Diseñar Splash Screen	25
Armando Guzmán	R03F01T02-Implementar lógica para determinar el usuario que actualmente ha iniciado sesión y redirigirlo a la vista correspondiente.	19

R04:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R04F01T01-Diseñar el fragmento 'ProfileFragment'.	40
Armando Guzmán	R04F01T02-Implementar animación al desplegar la lista de acciones que puede realizar el usuario.	14
Armando Guzmán	R04F01T03-Implementar lógica para cargar datos de usuario actual	18
Armando Guzmán	R04F01T04-Implementar PromptAlertDialgo	30
Armando Guzmán	R04F01T05-Implementar TwoPromptAlertDialog	25
Armando Guzmán	R04F01T06-Implementar lógica para cerrar sesión correctamente	12
Armando Guzmán	R04F01T07-Implementar lógica para Re autenticarse	40
Armando Guzmán	R04F01T08-implementar lógica para Re autenticarse mediante Google	34
Armando Guzmán	R04F01T09-Implementar lógica para borrar cuenta	33
Armando Guzmán	R04F01T10-Implementar lógica para cambiar contraseña	28
Armando Guzmán	R04F01T11-Implementar lógica para cambiar nombre de usuario	37
Armando Guzmán	R04F01T12-Implementar lógica para cambiar correo electrónico	45
Armando Guzmán	R04F01T13-Crear dialog que muestra opciones de galería y Cámara	20
Armando Guzmán	R04F01T14-Implementar lógica para cargar foto de galería	23

Armando Guzmán	R04F01T15-Implementar lógica para cargar foto de Cámara	15
Armando Guzmán	R04F01T16-Implementar lógica para cargar foto a firebase	30
Armando Guzmán	R04F01T17-Implementar lógica para recoger foto de firebase	23
Armando Guzmán	R04F01T18-Implementar lógica para recoger foto de Google	16
Armando Guzmán	R04F01T19-Vincular los eventos que se producen en la vista con los métodos definidos en el 'ProfileViewModel'	14

R05:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R05F01T01-Diseñar fragmento de creación de grupo	45
Armando Guzmán	R05F01T02-Implementar validaciones de para creación de grupo	40
Armando Guzmán	R05F01T03-implementar date alert dialog	18
Armando Guzmán	R05F01T04-diseñar card de amigo para añadir grupo	12
Armando Guzmán	R05F01T05-Implementar lógica de creación de grupos	35
Armando Guzmán	R05F01T05-implementar lógica para carga de amigos	29
Armando Guzmán	R05F01T06-Vincular los eventos que se producen en la vista a la lógica definida en el ViewModel.	15

Armando Guzmán	R05F01T07-Implementar lógica para mostrar Progress Bar y ventanas emergentes para informar al usuario del resultado de sus acciones.	20
Armando Guzmán	R05F01T08-implementar la carga de fotos desde la creación de un grupo nuevo	25
Armando Guzmán	R05F01T09-implementar la eliminación de la foto	30
Armando Guzmán	R05F01T10-Implementar invitación de grupo al crear el grupo	25
Armando Guzmán	R05F02T01-implementar lógica de carga de grupos	23
Armando Guzmán	R05F02T01-implementar recyclerview de grupos	36
Armando Guzmán	R05F02T03-Diseñar layout de grupo en donde mostrar las propiedades más relevantes como nombre, descripción, foto, y categoría.	25
Armando Guzmán	R05F02T04-Implementar RecyclerView para mostrar los grupos.	30
Armando Guzmán	R05F02T05-implementar ordenamiento de grupos por más recientes	20
Armando Guzmán	R05F02T06-implementar lógica de filtrado mediante searchView	30
Armando Guzmán	R05F02T07-adaptar calendario	35
Armando Guzmán	R05F02T08-implementar lógica para filtrar grupos en función de la fecha escogida	32
Armando Guzmán	R05F02T09-Crear RecyclerView en donde mostrar las categorias existentes	34
Armando Guzmán	R05F02T10-Implementar lógica para filtrar los grupos que aparecen en función de las categorías seleccionadas	27
Armando Guzmán	R05F02T11-implementar la carga de fotos desde el fragmento de grupos	20

Armando Guzmán	R05F03T01-Diseñar el fragmento ‘GroupOverviewFragment’.	28
Armando Guzmán	R05F03T02-Implementar lógica para recibir por parámetro el grupo seleccionado y así cargar sus datos.	13
Armando Guzmán	R05F03T03-Implementar lógica de CRUD de grupos	60
Armando Guzmán	R05F03T04-implementar lógica para carga de miembros del grupo seleccionado	34
Armando Guzmán	R05F03T05-Implementar eliminación de usuario al actualizar grupo	17
Armando Guzmán	R05F03T06-Implementar invitación al actualizar grupo	25
Armando Guzmán	R05F03T07-Diseñar el SpinnerDialog.	13
Armando Guzmán	R05F03T08-Implementar lógica para mostrar y ocultar ciertas opciones en función del rol que ocupa el usuario actual en el grupo.	33
Armando Guzmán	R05F03T09-Implementar la modificación de los roles dentro del fragmento de GroupOverview	40
Armando Guzmán	R05F03T10-Implementar lógica para que en caso de que el usuario actual sea eliminado del grupo o pierda sus privilegios de administrador, la vista se actualice correctamente.	38
Armando Guzmán	R05F03T11-implementar la carga de fotos desde la actualizacion de un grupo existente	18
Armando Guzmán	R05F03T12-implementar la eliminación de la foto del grupo al borrarlo	15
Armando Guzmán	R05F03T13-Implementar poder salir del grupo	20
Armando Guzmán	R05F03T14-Implementar la función de filtrar los amigos y miembros por nombre.	30

R06:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R06F01T01-Diseñar fragmento de invitaciones	16
Armando Guzmán	R06F01T02-Diseñar layout de invitación	14
Armando Guzmán	R06F01T03-Implementar clase Adapter	23
Armando Guzmán	R06F01T04-Implementar clase ViewHolder	25
Armando Guzmán	R06F01T05-Implementar lógica de cargado de invitaciones	45
Armando Guzmán	R06F01T06-implementar lógica de aceptar y rechazar invitaciones	30

R07:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R07F01T01-Diseñar fragmento de chat	40
Armando Guzmán	R07F01T02-diseñar los layouts de los mensajes	40
Armando Guzmán	R07F01T03-Implementar lógica para recoger el texto introducido por el usuario y validarlo.	10
Armando Guzmán	R07F01T04-Añadir lógica para cargar imagen de cámara o galería	20
Armando Guzmán	R07F01T05-Implementar lógica tanto en 'ChatRepository' como en 'ChatViewModel' para guardar mensajes en base de datos.	35

Armando Guzmán	R07F01T06-Implementar lógica para cargar los mensajes del grupo.	19
Armando Guzmán	R07F01T07-Añadir listener para saber si se sigue siendo miembro	10

R08:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R08F01T01-Crear dos ficheros de colores dentro del directorio de recursos de la aplicación, uno a usar para el tema claro y otro para el tema oscuro.	15
Armando Guzmán	R08F01T02-Definir el tema de la aplicación para que use estas paletas de colores.	5

R09:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R09F01T01-Crear el fichero de strings con los textos usados por la aplicación en español.	13
Armando Guzmán	R09F01T02-Crear el fichero de strings con los textos usados por la aplicación en inglés.	14

R10:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R10F01T01-Crear fichero dimen con los tamaños usados en las vistas por los dispositivos móviles.	13

Armando Guzmán	R10F01T02-Crear fichero dimen con los tamaños usados en las vistas por las tabletas.	16

R12:

Realizado por	Tarea	Tiempo empleado (Minutos)
Armando Guzmán	R12F01T01: Implementar backend local para poder recibir peticiones HTTP en diferentes rutas y comunicarse con Firebase	35
Armando Guzmán	R12F01T02: Implementar clase para poder realizar las peticiones correspondientes al backend local desde la aplicación	28
Armando Guzmán	R12F01T03: Incorporar la llamada al servicio en los métodos que lo requieran	40

Tiempo total empleado por Armando Guzmán (Horas): 44,07.

R11:

Realizado por	Tarea	Tiempo empleado (Minutos)
Álvaro Aparicio Montoto	R11F01T01- Diseñar el fragmento 'FragmentFriends', que contendrá un 'RecyclerView' para mostrar la lista de amigos, un 'SearchView' para buscar amigos y botón para enviar invitación a otros usuarios.	35
Álvaro Aparicio Montoto	R11F01T02- Implementar lógica en 'FriendsViewModel' para cargar la lista de amigos del usuario actual desde la base de datos.	70

Álvaro Aparicio Montoto	R11F01T03- Implementar‘RecyclerView’en ‘FriendsFragment’ para mostrar la lista de amigos del usuario actual.	20
Álvaro Aparicio Montoto	R11F01T04- Implementar lógica para buscar y filtrar amigos en la lista.	65
Álvaro Aparicio Montoto	R11F01T05- Implementar lógica para enviar solicitudes de amistad.	50
Álvaro Aparicio Montoto	R11F01T06- Implementar adaptador ‘FriendsAdapter’ para enlazar los datos de los amigos con las vistas del ‘RecyclerView’.	30
Álvaro Aparicio Montoto	R11F01T07- Implementar ‘viewHolder’ ‘FriendsViewHolder’ para enlazar los datos de cada amigo con las vistas correspondientes del ‘RecyclerView’.	35

R13:

Realizado por	Tarea	Tiempo empleado (Minutos)
Álvaro Aparicio Montoto	R13F01T01- Diseñar el fragmento ‘NewExpenseFragment’.	35
Álvaro Aparicio Montoto	R13F01T02- Implementar validadores para los campos del registro del gasto.	50
Álvaro Aparicio Montoto	R13F01T03- Implementar ventana emergente para seleccionar la fecha del gasto.	45

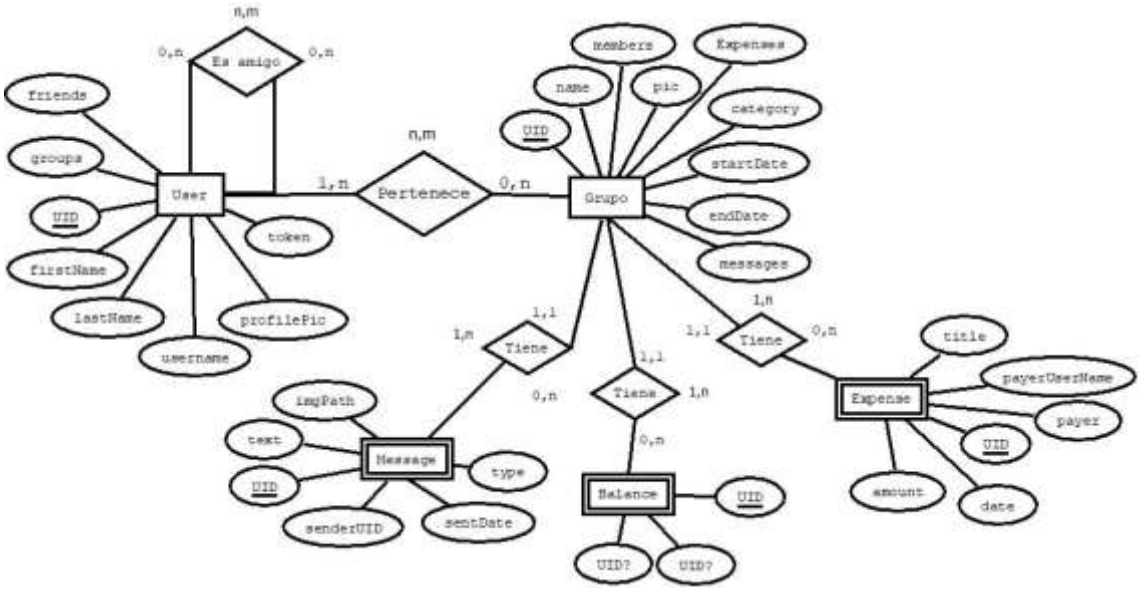
Álvaro Aparicio Montoto	R13F01T04- Diseñar el layout para mostrar los detalles del gasto.	35
Álvaro Aparicio Montoto	R13F01T05- Implementar lógica para mostrar información relevante en el formulario.	25
Álvaro Aparicio Montoto	R13F01T06- Implementar lógica en 'ExpenseRepository' y 'NewExpenseViewModel' para crear un nuevo gasto.	800
Álvaro Aparicio Montoto	R13F01T07- Vincular los eventos de la vista a la lógica definida en el ViewModel.	25
Álvaro Aparicio Montoto	R13F01T08- Implementar lógica para mostrar indicadores de progreso y ventanas emergentes para informar al usuario sobre el resultado de sus acciones.	40
Álvaro Aparicio Montoto	R13F02T01- Diseñar el fragmento 'ExpenseFragment'.	30
Álvaro Aparicio Montoto	R13F02T02- Implementar lógica para cargar los gastos registrados.	250
Álvaro Aparicio Montoto	R13F02T03- Diseñar layout para mostrar los detalles de cada gasto, como nombre, descripción, fecha y cantidad.	30
Álvaro Aparicio Montoto	R13F02T04- Implementar RecyclerView para mostrar los gastos.	70

Tiempo total empleado por Álvaro Aparicio Montoto (Horas): 29.

BASE DE DATOS

Diagrama Entidad-Relación:

Aunque el sistema de base de datos que se piensa implementar para la aplicación seria Firebase Realtime Database, la cual es no relacional basada en documentos, este diagrama de entidad relación resulta útil para entender la dinámica de las relaciones entre las distintas entidades que habrá en la aplicación.



- Un usuario puede tener como amigo a ningún otro usuario o varios, por eso la relación es en ambas direcciones de 0,n con una cardinalidad resultante de n,m.
- Un usuario puede pertenecer a ningún grupo o a varios, y un grupo puede tener como miembros como mínimo uno y máximo varios, por lo que la cardinalidad de la relación quedaría en n, m.
- Un grupo puede contar con ningún mensaje o varios, y un mensaje solo puede pertenecer a un grupo, por ello la cardinalidad entre estas entidades es 1,n.
- Un grupo puede contar con ningún saldo (Balance) o varios y un saldo solo puede pertenecer a un grupo, quedando la cardinalidad de la relación en 1,n.

- Un grupo puede tener ningún gasto (Expense) o varios, y un gasto solo puede pertenecer a un grupo, resultando en que la cardinalidad de la relación sea 1,n.

Las entidades **mensaje**, **balance** y **expense** son entidades débiles ya que están dependen de su relación con un grupo para existir, es decir, están no tiene sentido sin la existencia a grupo al cual pertenecer.

Estructura de las Colecciones de Firebase Realtime Database:

Users:

Dentro de esta colección, cada documento tendrá como clave un UID único que lo identifique, y como valor tendrá un objeto el cual contendrá las siguientes propiedades:

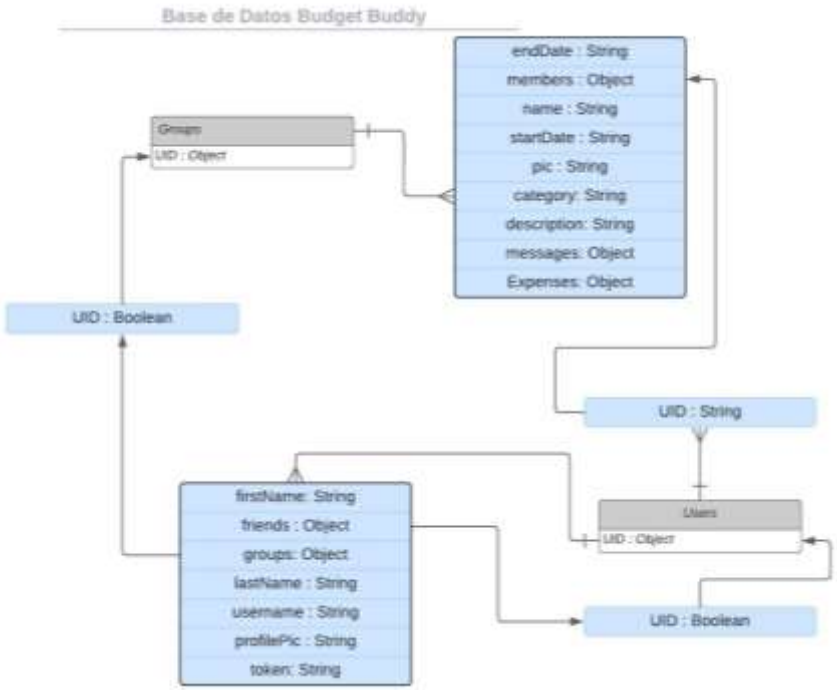
- firstName: Contendrá el primer nombre del usuario.
- lastName: Contendrá los apellidos del usuario
- username: Contendrá el nombre de usuario.
- Friends: Contendrá un objeto, donde cada clave será un UID de otro usuario y como valor tendrá un valor booleano, el valor no importa cuál sea, lo importante es tener la referencia para determinar que son amigos.
- Groups: Contendrá un objeto, donde cada clave será un UID de un grupo, y el valor será un valor booleano, de nuevo el valor no importa, solo es importante que el UID del grupo al que pertenece exista para determinar que es miembro.
- profilePic: Esta clave solo existirá si el usuario ha subido o no su foto de perfil, esta contendrá una referencia a la foto subida a Firebase Storage o directamente el enlace a la foto de perfil que tenga en su cuenta de Google.
- Token: Esta propiedad contendrá el token de registro del dispositivo, usado para mandar notificaciones al dispositivo que tenga esa cuenta.

Groups:

Dentro de esta colección cada documento tendrá como clave un UID único que identifica el grupo, y como valor tendrá un objeto el cual tendrá las siguientes propiedades:

- endDate: Esta clave tendrá como valor la representación en cadena de texto de la fecha en la cual acabará el evento del grupo.
- startDate: Esta clave tendrá como valor la representación en cadena de texto de la fecha en la cual iniciará el evento.
- Name: Contendrá el nombre del grupo.
- Pic: Contendrá una referencia a la foto del grupo guardada en Firebase Storage.
- Members: Contendrá un objeto, en donde cada miembro se verá representado por su UID como clave y como valor tendrá el rol que dicho miembro tiene dentro del grupo.
- Category: Esta propiedad contendrá la categoría del grupo.
- Description: Esta propiedad contendrá la descripción del grupo.
- lastUpdated: Esta propiedad contendrá el momento de la ultima actualizacion del grupo, representado en milisegundos extraídos del servidor.
- Messages: Dentro de esta propiedad habrá un objeto, cuyas claves serán los uid de los mensajes del grupo, y el valor de cada una será un objeto con las siguientes propiedades:
    - imgPath: Contendrá la ruta de la imagen adjunta el mensaje, si el tipo de mensaje es de foto.
    - SenderUID: Contendrá el uid del usuario que ha mandado el mensaje.
    - sentDate: Contendrá el momento en el cual se ha enviado el mensaje, será representado en milisegundos extraídos del servidor.
    - Text: Contendrá el texto del mensaje si el tipo es de mensaje de texto.
    - Type: Contendrá el tipo del mensaje.

- Expenses: Dentro de esta propiedad habrá un objeto cuyas claves serán los uid de cada uno de los gastos, y como valor cada clave tendrá como valor un objeto con las siguientes claves:
    - Amount: contendrá el monto del gasto.
    - Date: Contendrá la representación en cadena de texto de la fecha, en el cual se produjo el gasto.
    - Payer: Contendrá el uid del usuario que he creado el gasto.
    - payerUsername: Contendrá el nombre de usuario del miembro del grupo que ha creado el grupo.
    - Title: Contendrá el nombre del gasto.



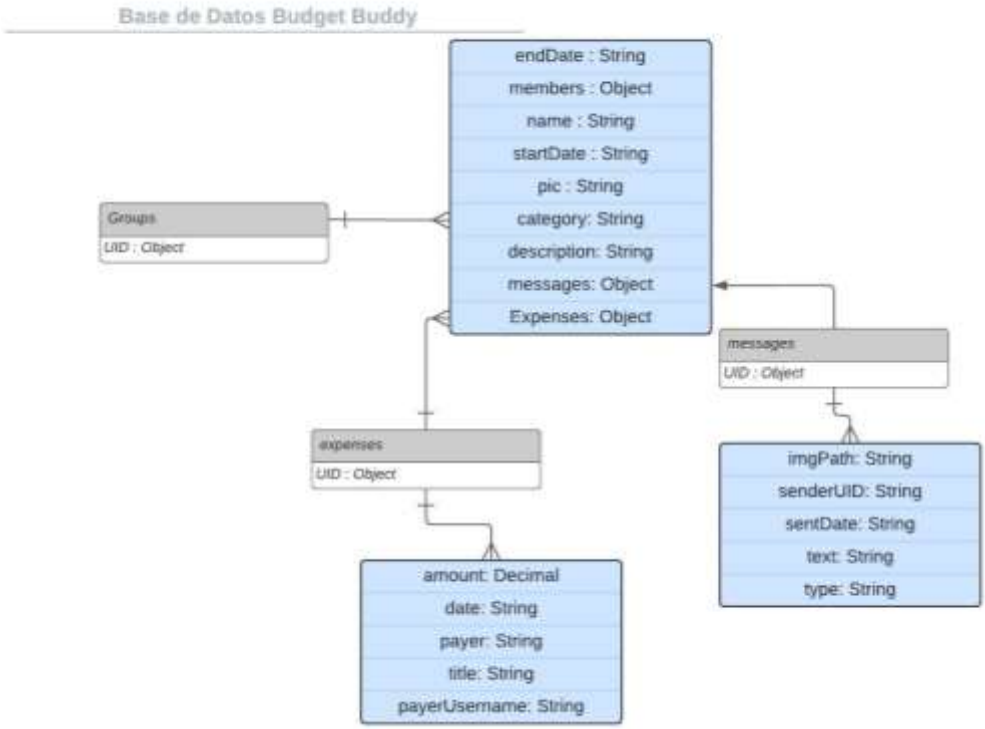
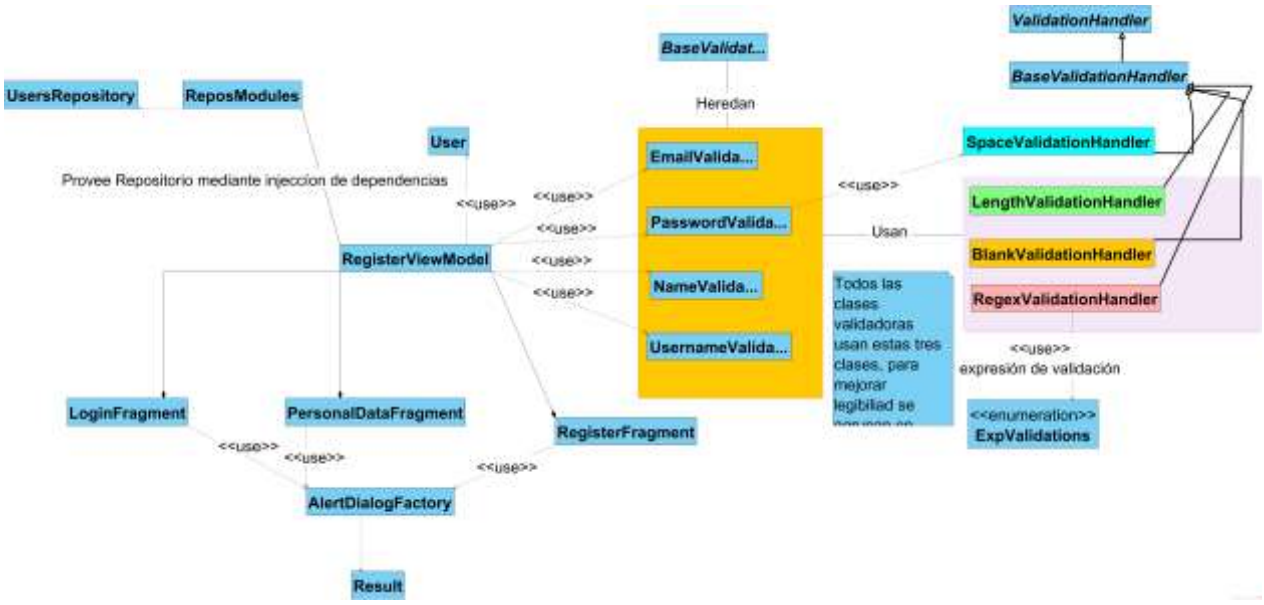
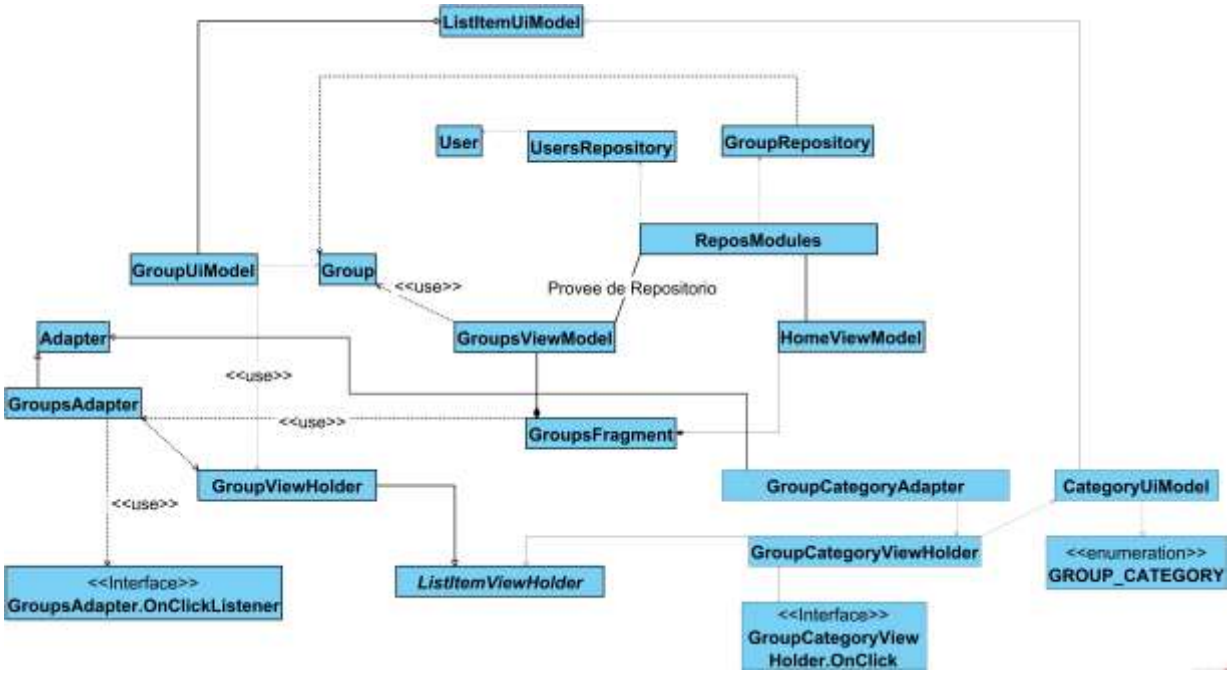


DIAGRAMA DE CLASES

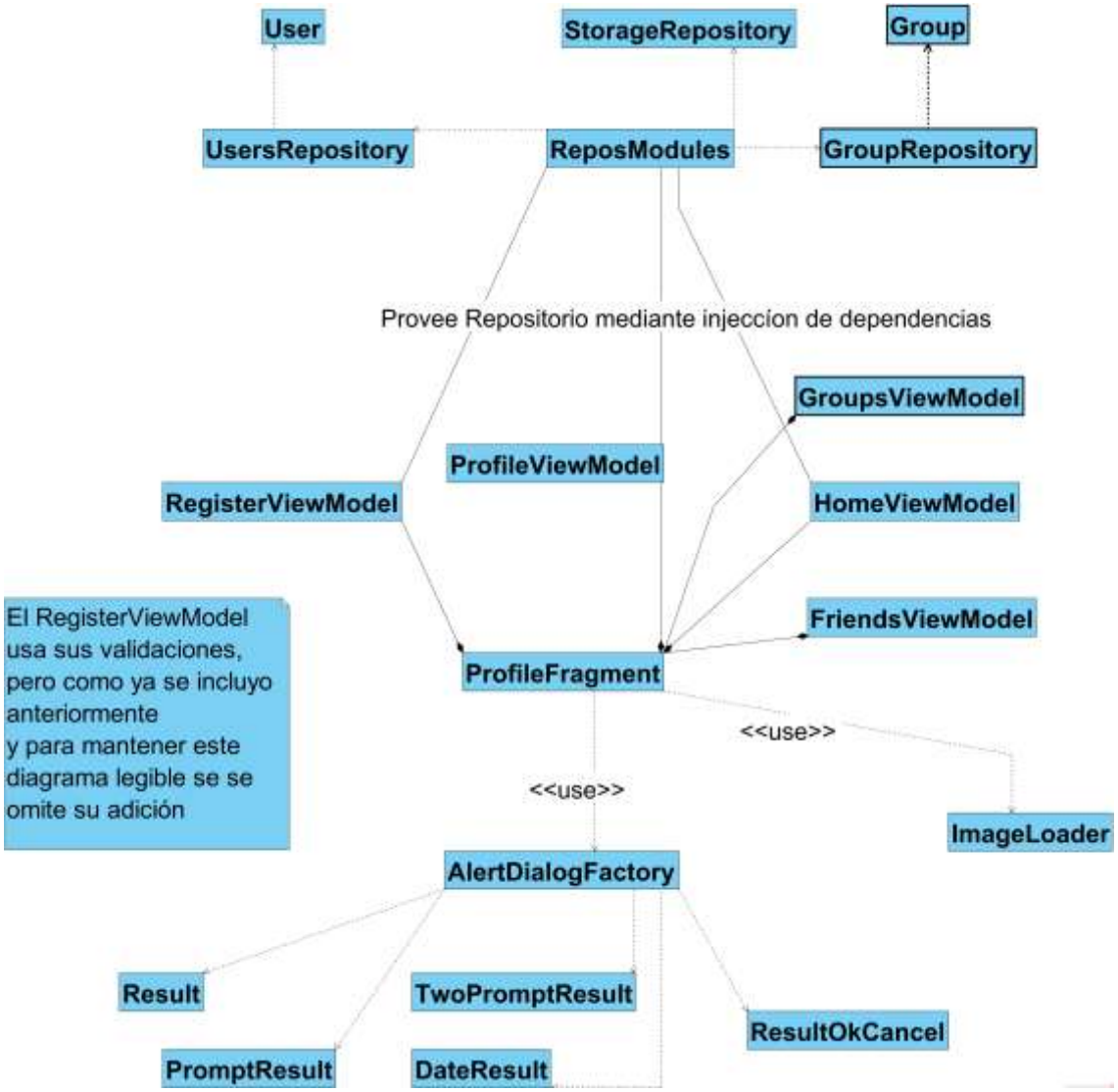
Jerarquía de clases de los fragmentos de Login, Register y PersonalData:



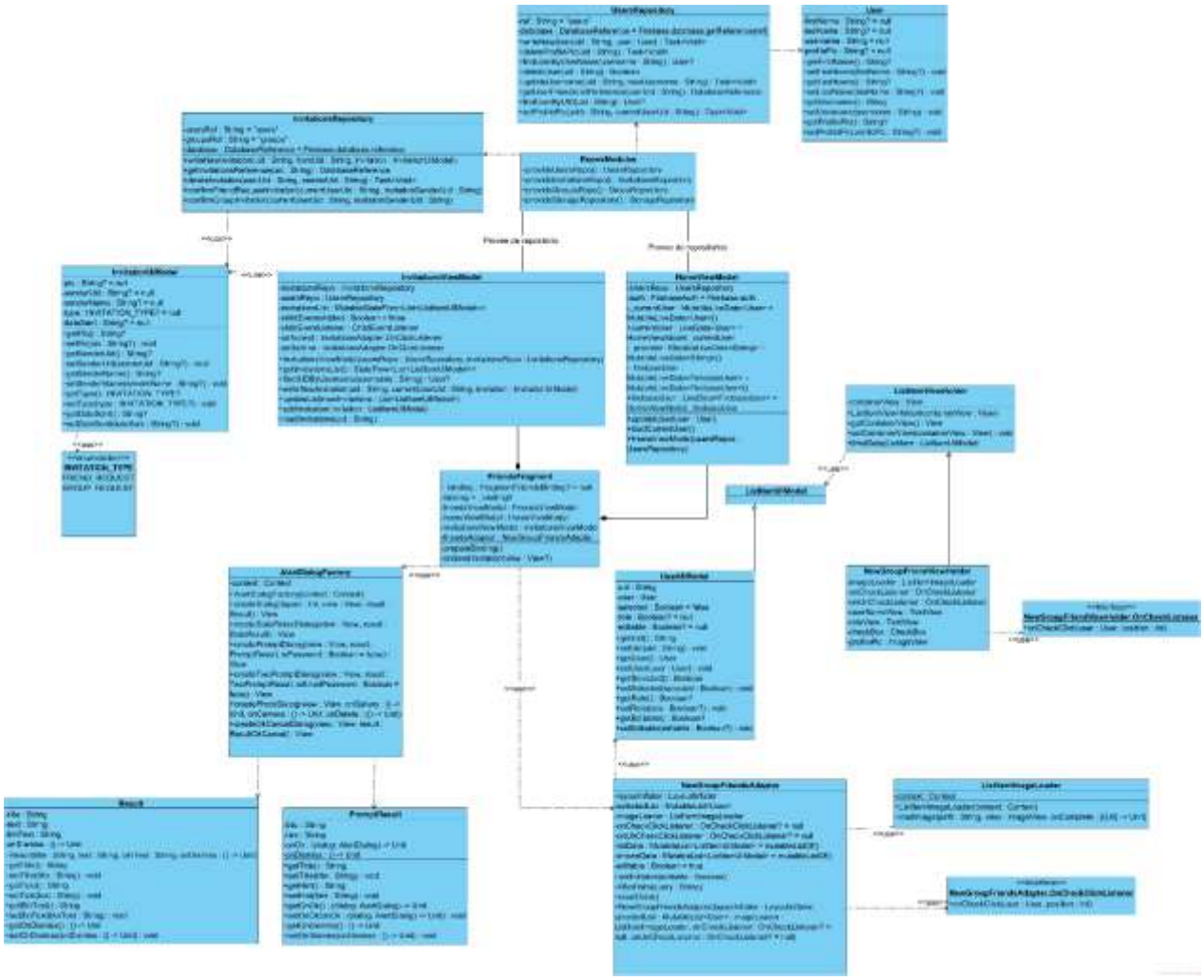
Jerarquía de clases del fragmento de Groups (Grupos):



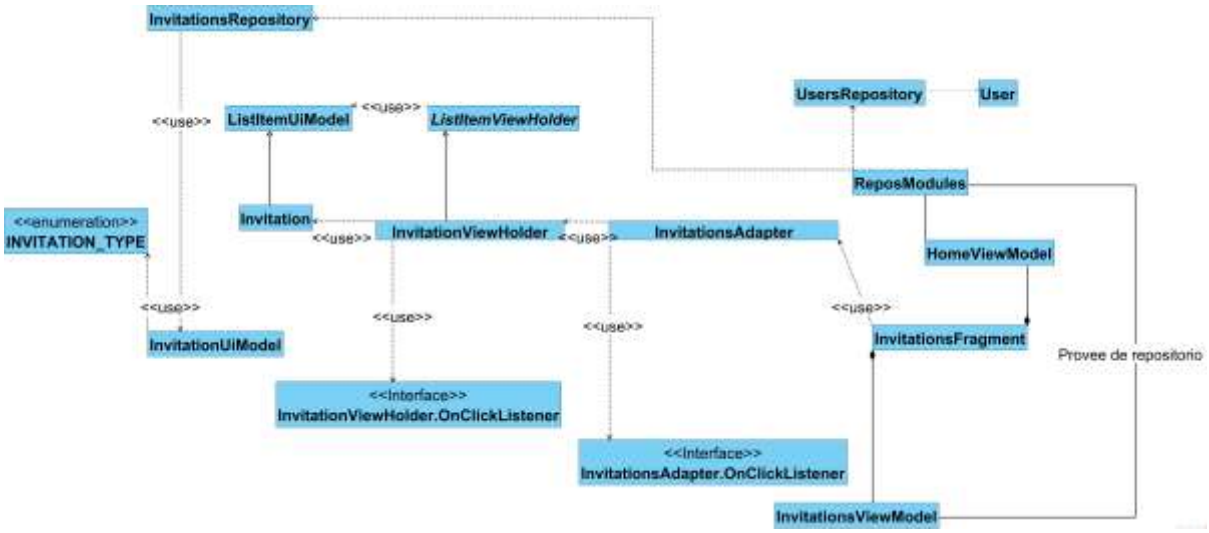
Jerarquía de clases de el fragmento de Profile (perfil):



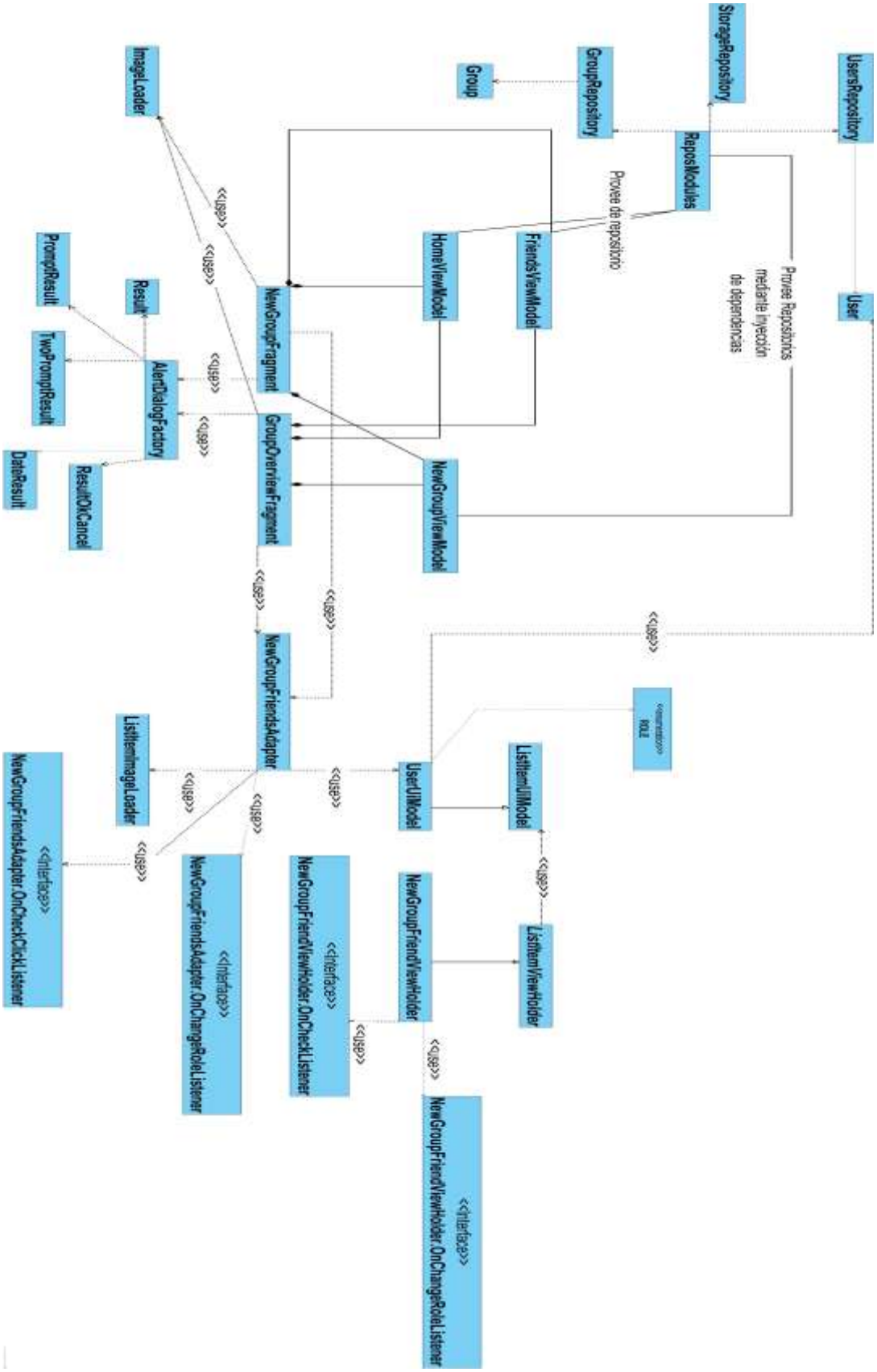
Jerarquía de clases del fragmento de Friends (Amigos):



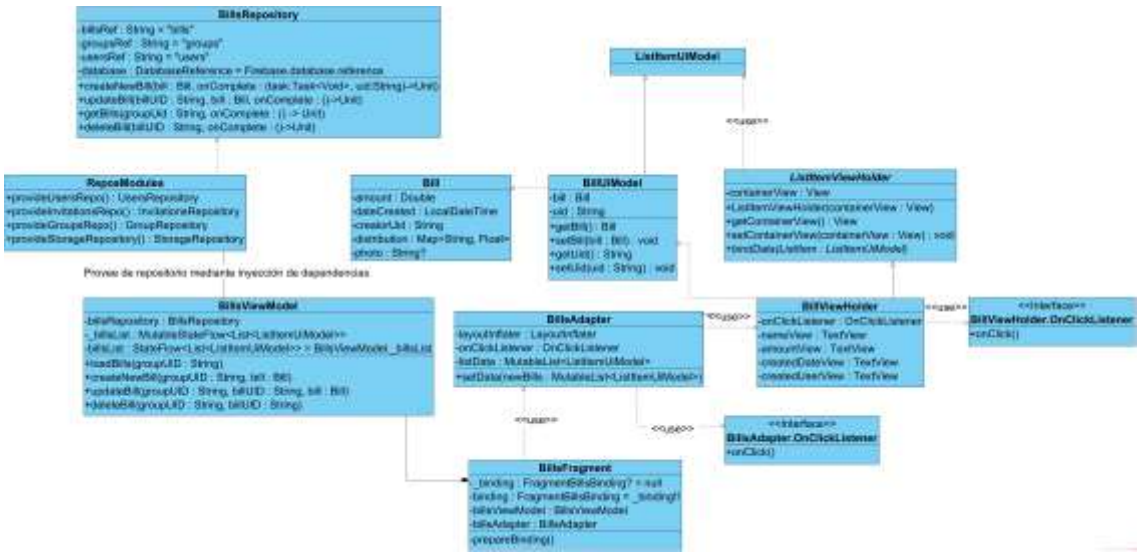
Jerarquía de clases del fragmento Invitations (Invitaciones):



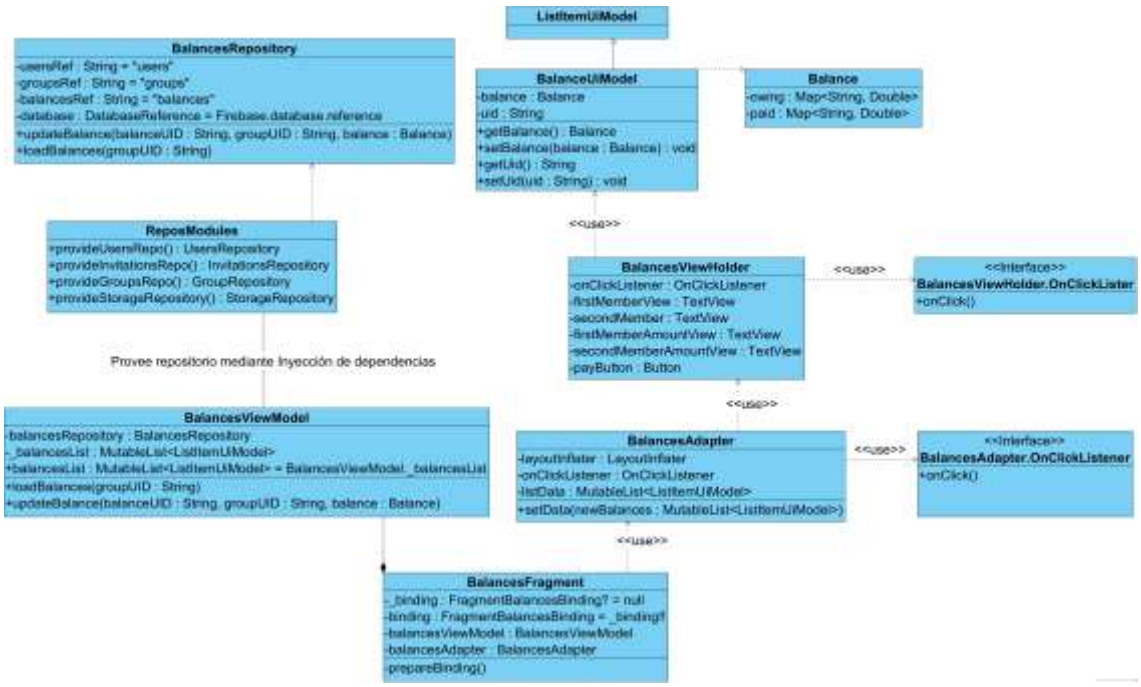
Jerarquía de clases de los fragmentos para crear y actualizar los grupos:



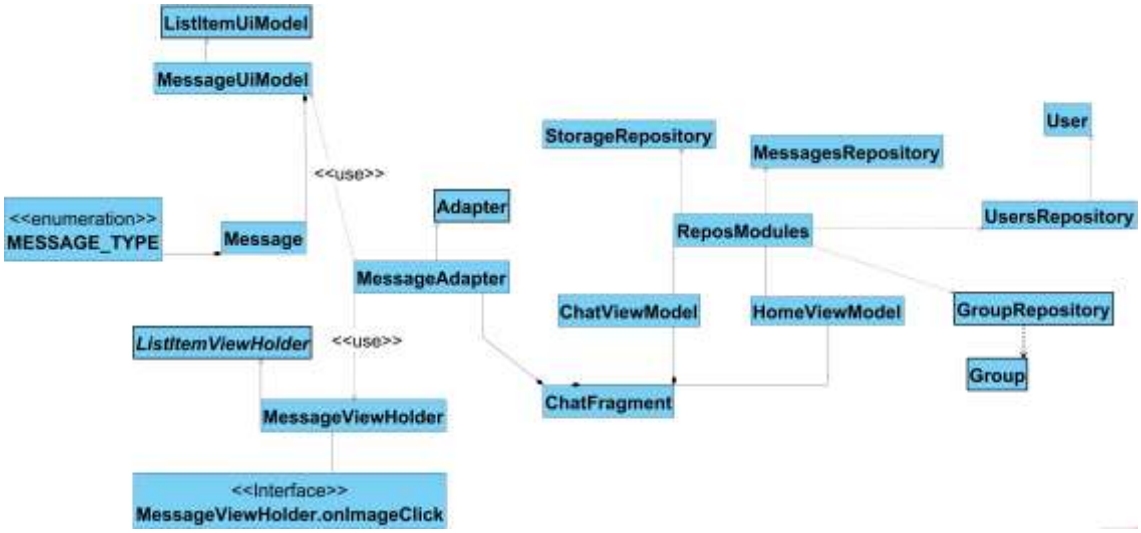
Jerarquía de clases para el fragmento de gastos (Bills):



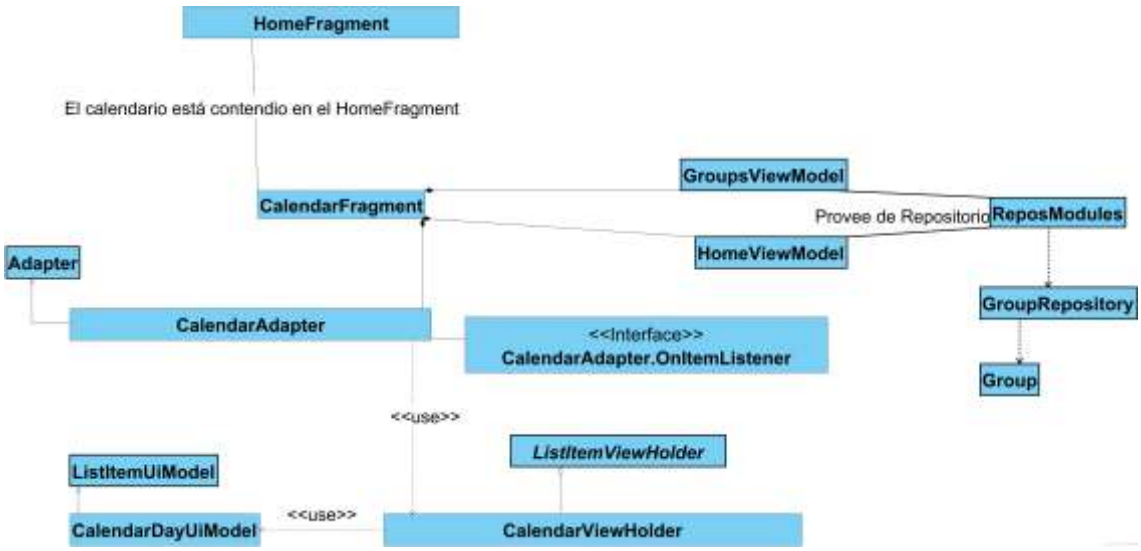
Jerarquía de clases para el fragmento de saldos (Balances):



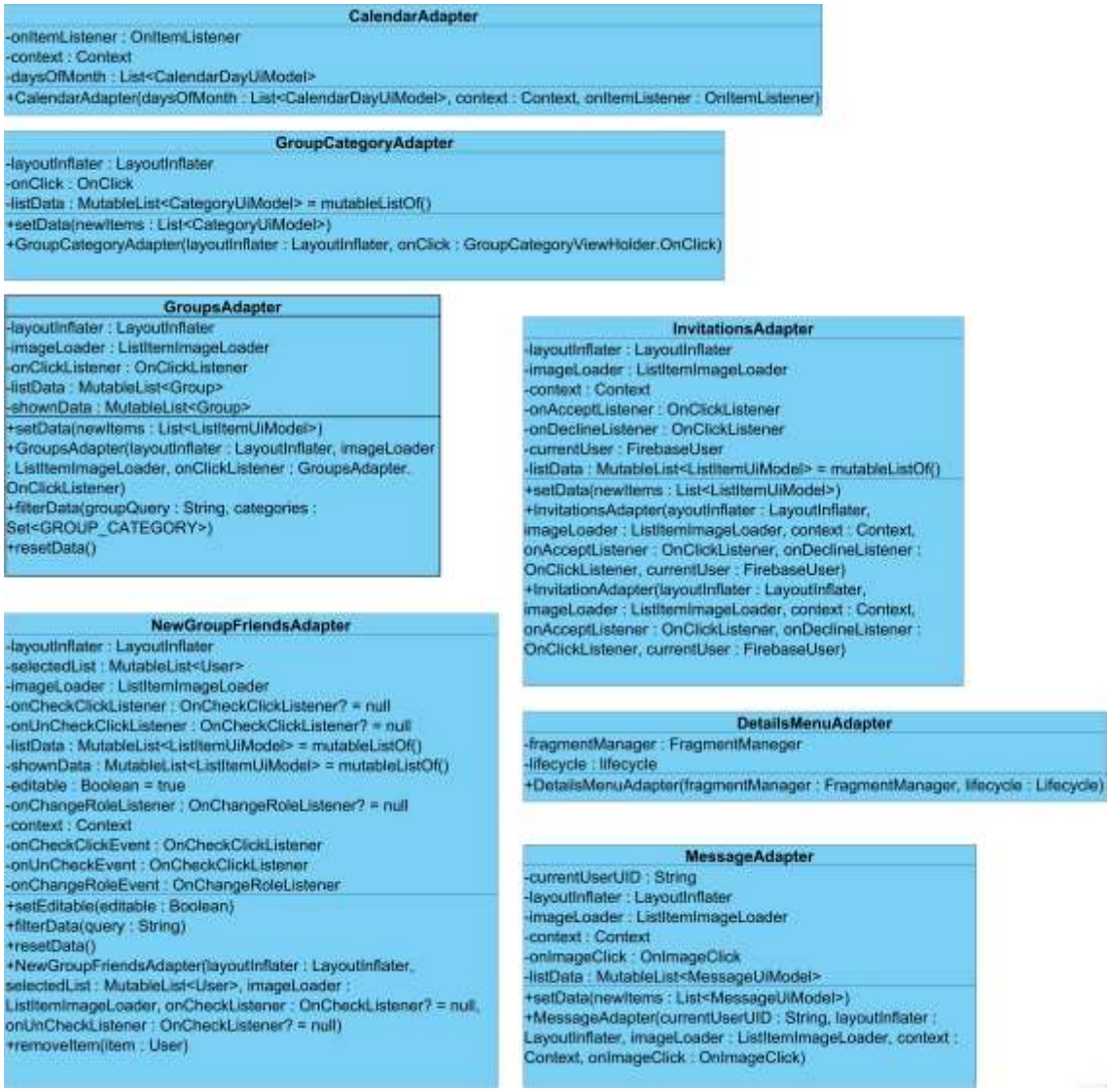
jerarquía de clases del fragmento de chat:



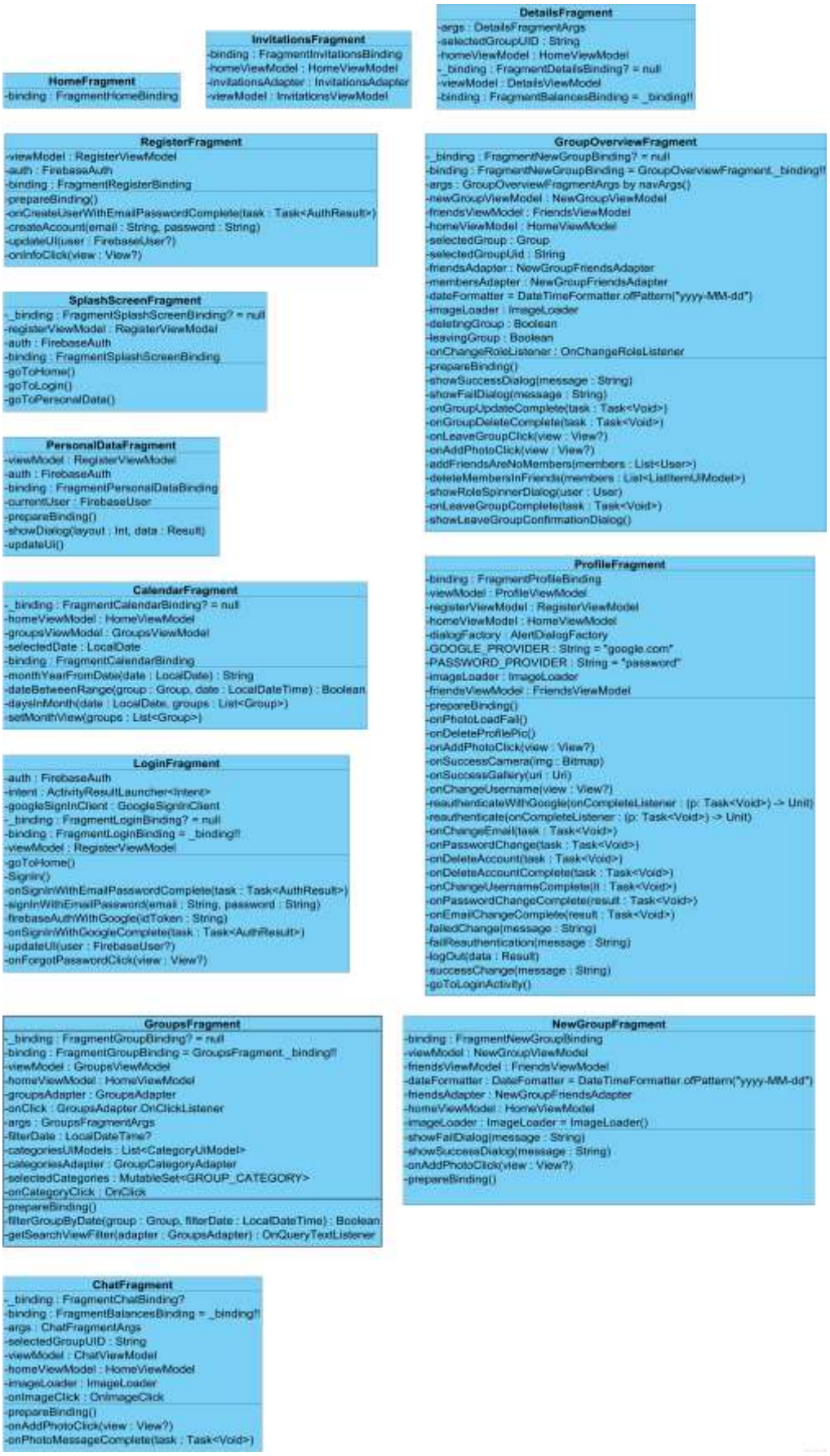
jerarquía de clases del fragmento del calendario.



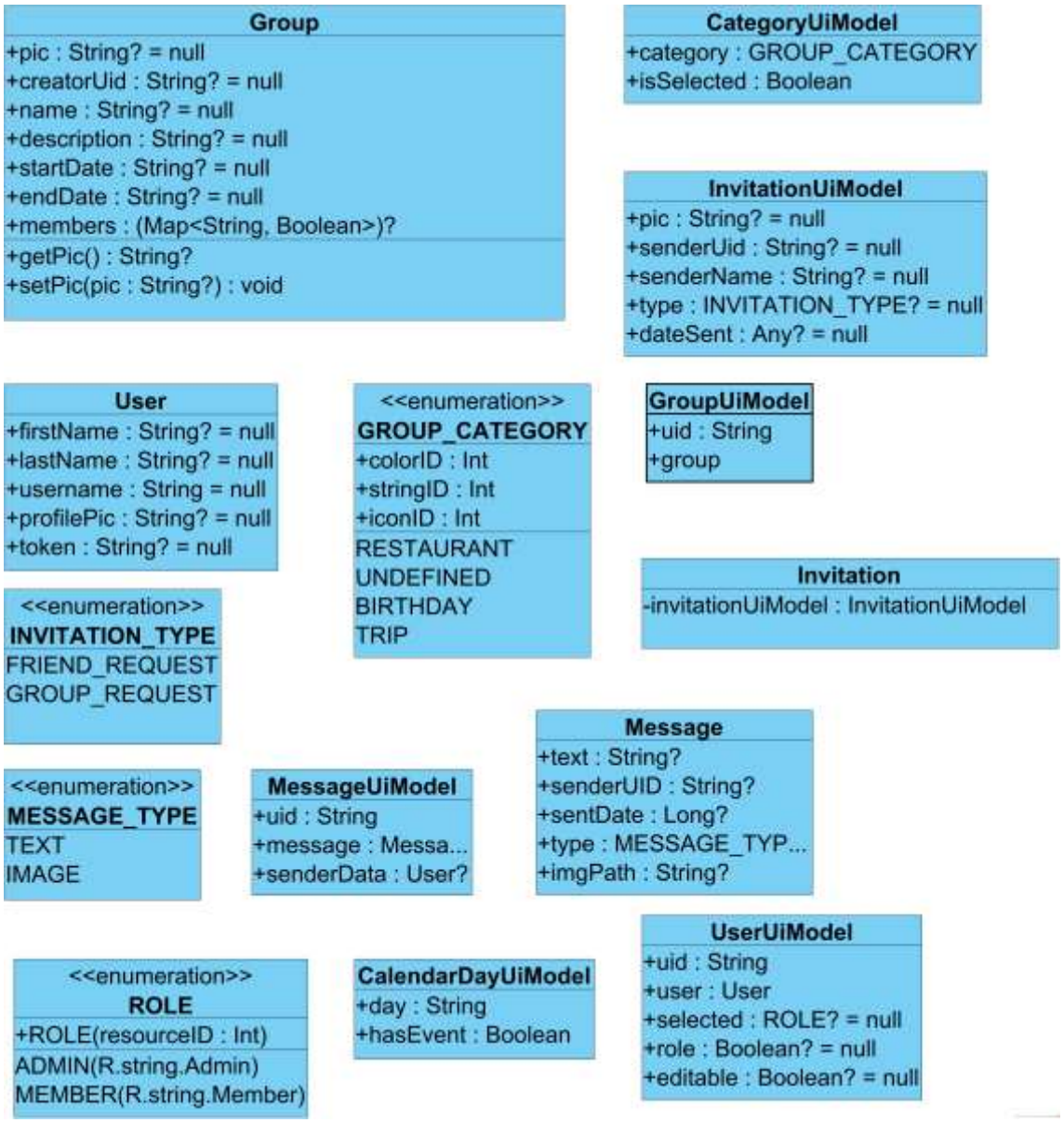
Clases Adapter



Clases Fragmento



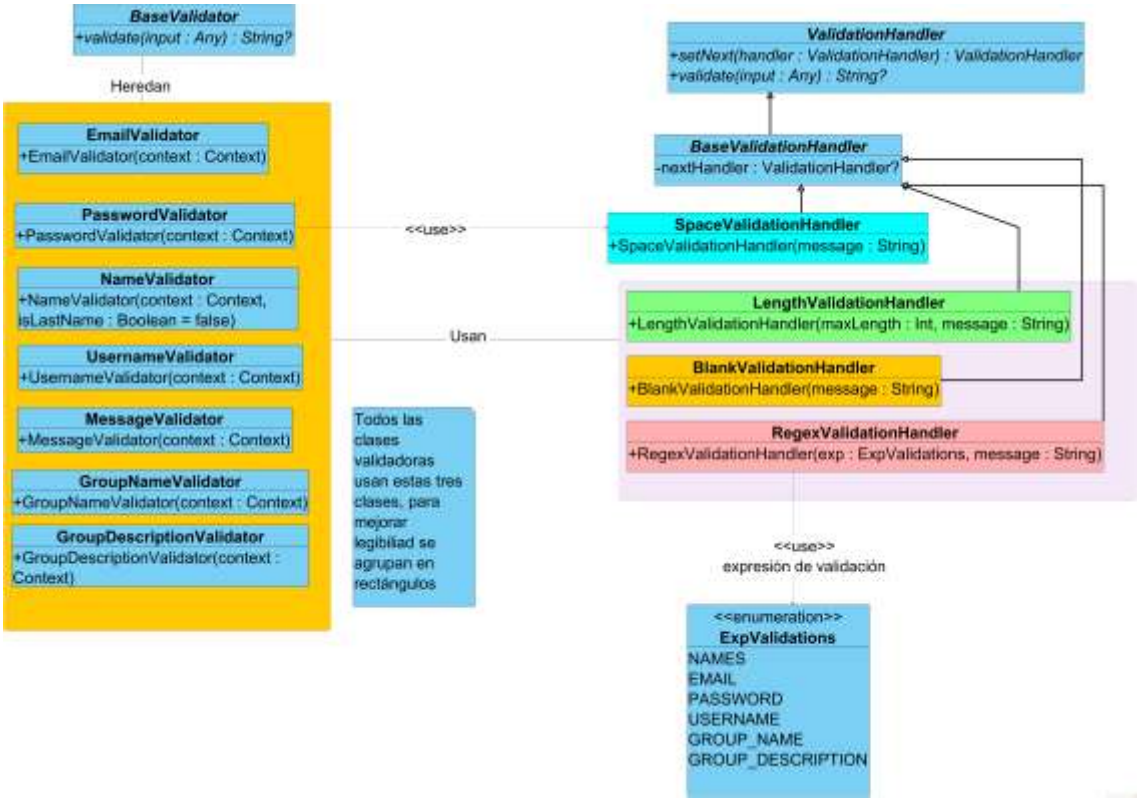
Clases de Modelo



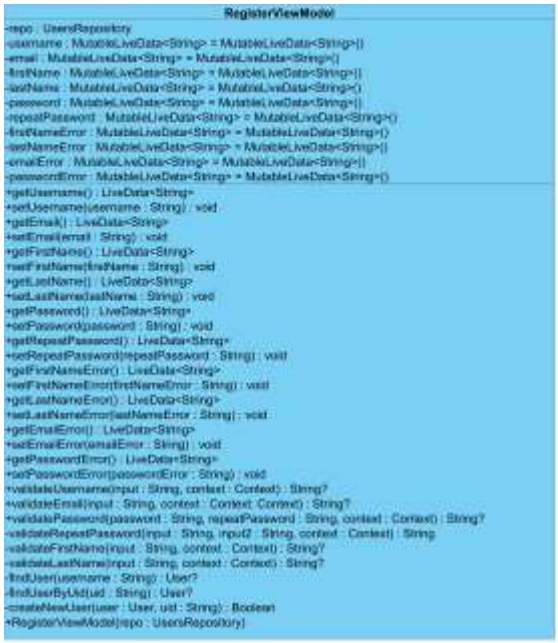
Clases repositorio

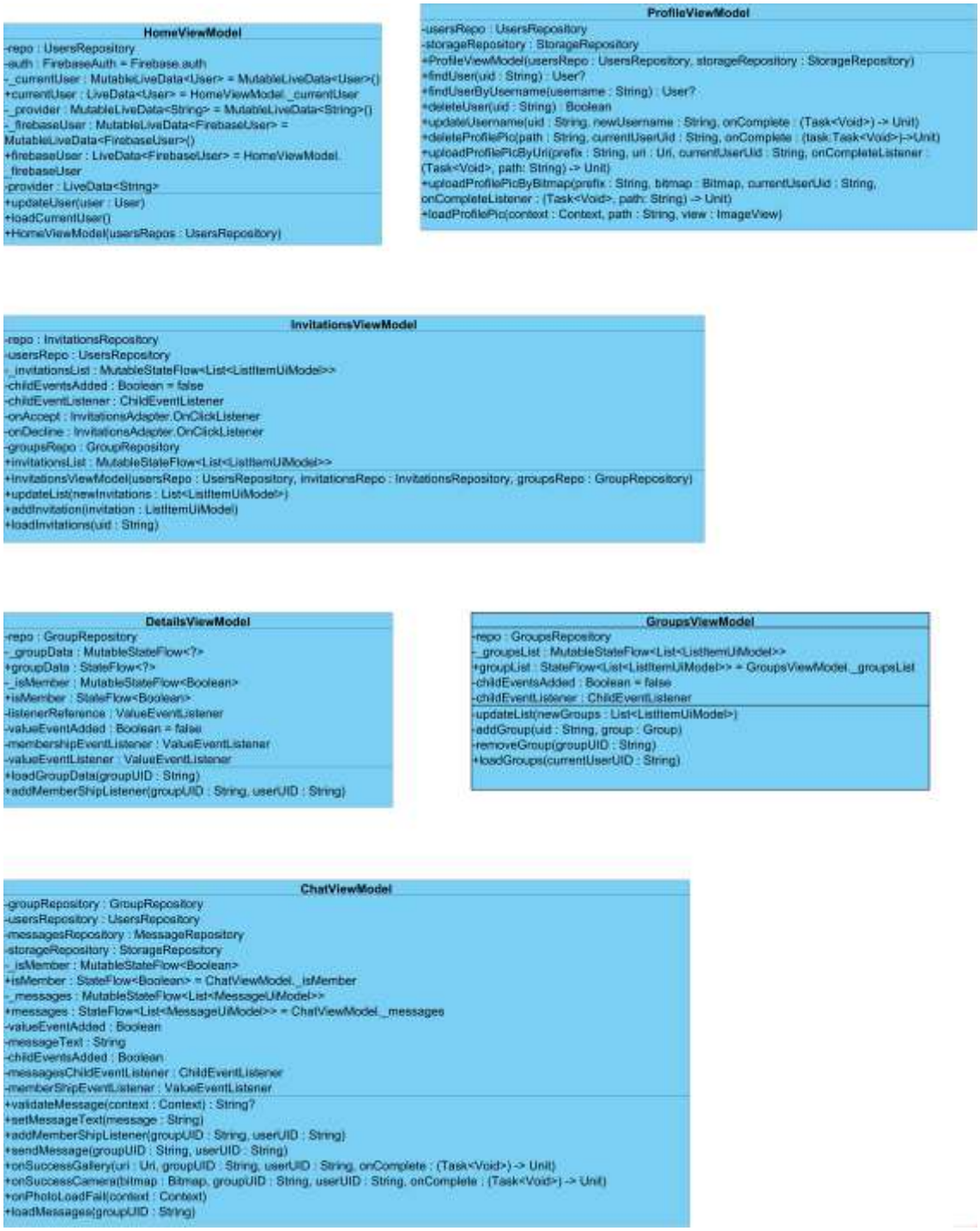
Clases de utilidad

Clases de validaciones

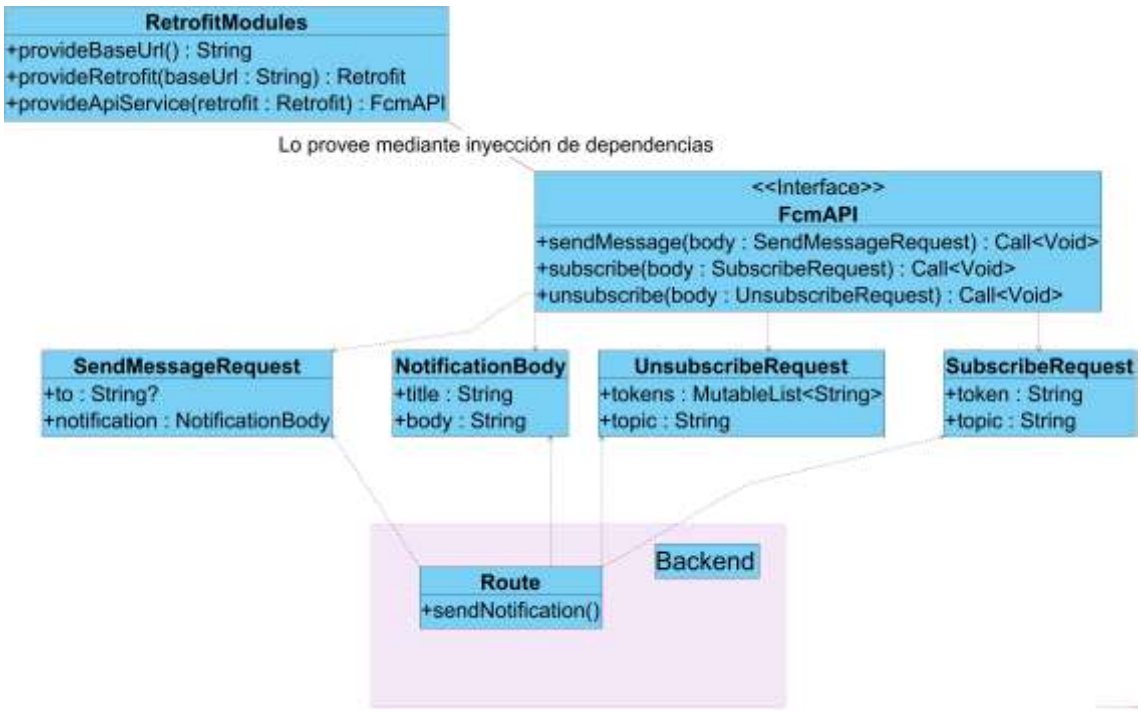


Clases de ViewModel

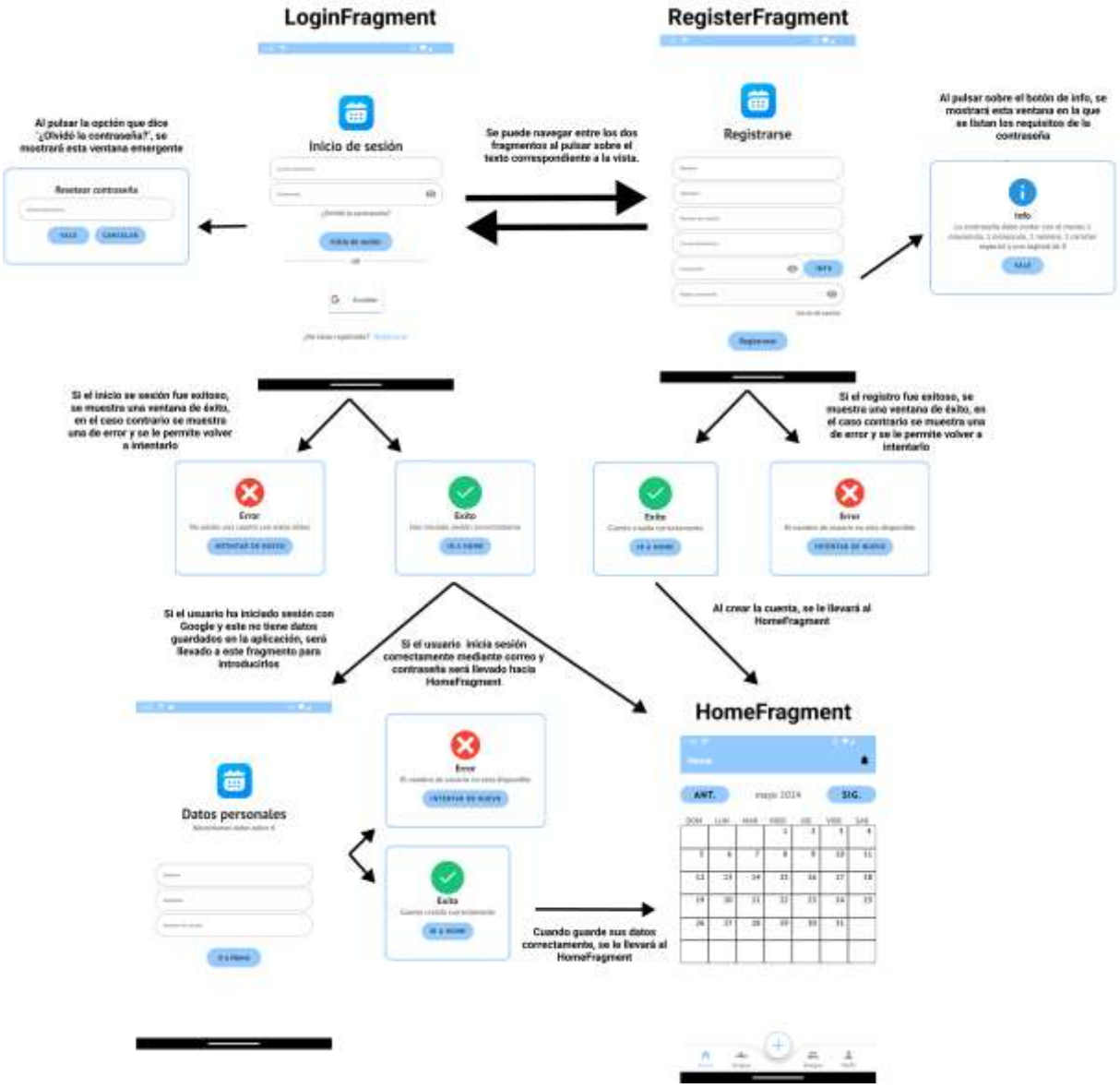



HomeViewModel
-repo : UsersRepository
-auth : FirebaseAuth = Firebase.auth
-_currentUser : MutableLiveData<User> = MutableLiveData<User>()
+currentUser : LiveData<User> = HomeViewModel._currentUser
-_provider : MutableLiveData<String> = MutableLiveData<String>()
-_firebaseUser : MutableLiveData<FirebaseUser> = MutableLiveData<FirebaseUser>()
+firebaseUser : LiveData<FirebaseUser> = HomeViewModel._firebaseUser
-provider : LiveData<String>
+updateUser(user : User)
+loadCurrentUser()
+HomeViewModel(usersRepos : UsersRepository)
ProfileViewModel
-usersRepo : UsersRepository
-storageRepository : StorageRepository
+ProfileViewModel(usersRepo : UsersRepository, storageRepository : StorageRepository)
+findUser(uid : String) : User?
+findUserByUsername(username : String) : User?
+deleteUser(uid : String) : Boolean
+updateUsername(uid : String, newUsername : String, onComplete : (Task<Void>) -> Unit)
+deleteProfilePic(path : String, currentUserUid : String, onComplete : (task:Task<Void>)->Unit)
+uploadProfilePicByUri(prefix : String, uri : Uri, currentUserUid : String, onCompleteListener : (Task<Void>, path: String) -> Unit)
+uploadProfilePicByBitmap(prefix : String, bitmap : Bitmap, currentUserUid : String, onCompleteListener : (Task<Void>, path: String) -> Unit)
+loadProfilePic(context : Context, path : String, view : ImageView)
InvitationsViewModel
-repo : InvitationsRepository
-usersRepo : UsersRepository
-_invitationsList : MutableStateFlow<List<ListItemUiModel>>
-childEventsAdded : Boolean = false
-childEventListener : ChildEventListener
-onAccept : InvitationsAdapter.OnClickListener
-onDecline : InvitationsAdapter.OnClickListener
-groupsRepo : GroupRepository
+invitationsList : MutableStateFlow<List<ListItemUiModel>>
+InvitationsViewModel(usersRepo : UsersRepository, invitationsRepo : InvitationsRepository, groupsRepo : GroupRepository)
+updateList(newInvitations : List<ListItemUiModel>)
+addInvitation(invitation : ListItemUiModel)
+loadInvitations(uid : String)
DetailsViewModel
-repo : GroupRepository
-_groupData : MutableStateFlow<?>
+groupData : StateFlow<?>
-_isMember : MutableStateFlow<Boolean>
+isMember : StateFlow<Boolean>
-listenerReference : ValueEventListener
-valueEventAdded : Boolean = false
-membershipEventListener : ValueEventListener
-valueEventListener : ValueEventListener
+loadGroupData(groupUID : String)
+addMemberShipListener(groupUID : String, userUID : String)
GroupsViewModel
-repo : GroupsRepository
-_groupsList : MutableStateFlow<List<ListItemUiModel>>
+groupList : StateFlow<List<ListItemUiModel>> = GroupsViewModel._groupsList
-childEventsAdded : Boolean = false
-childEventListener : ChildEventListener
-updateList(newGroups : List<ListItemUiModel>)
+addGroup(uid : String, group : Group)
+removeGroup(groupUID : String)
+loadGroups(currentUserUID : String)
ChatViewModel
-groupRepository : GroupRepository
-usersRepository : UsersRepository
-messagesRepository : MessageRepository
-storageRepository : StorageRepository
-_isMember : MutableStateFlow<Boolean>
+isMember : StateFlow<Boolean> = ChatViewModel._isMember
-_messages : MutableStateFlow<List<MessageUiModel>>
+messages : StateFlow<List<MessageUiModel>> = ChatViewModel._messages
-valueEventAdded : Boolean
-messageText : String
-childEventsAdded : Boolean
-messagesChildEventListener : ChildEventListener
-memberShipEventListener : ValueEventListener
+validateMessage(context : Context) : String?
+setMessageText(message : String)
+addMemberShipListener(groupUID : String, userUID : String)
+sendMessage(groupUID : String, userUID : String)
+onSuccessGallery(uri : Uri, groupUID : String, userUID : String, onComplete : (Task<Void>) -> Unit)
+onSuccessCamera(bitmap : Bitmap, groupUID : String, userUID : String, onComplete : (Task<Void>) -> Unit)
+onPhotoLoadFail(context : Context)
+loadMessages(groupUID : String)

Clases del servicio REST y backend



FLUJO DE INTERFACES



Gracias al menú de navegación disponible en la parte inferior, se puede navegar en cualquier momento entre todos estos fragmmentos

Siempre estará disponible el botón de campana en la parte superior para acceder a las invitaciones del usuario

ProfileFragment

HomeFragment

InvitationsFragment

NewGroupFragment

GroupsFragment

FriendsFragment

NewGroupFragment

HomeFragment

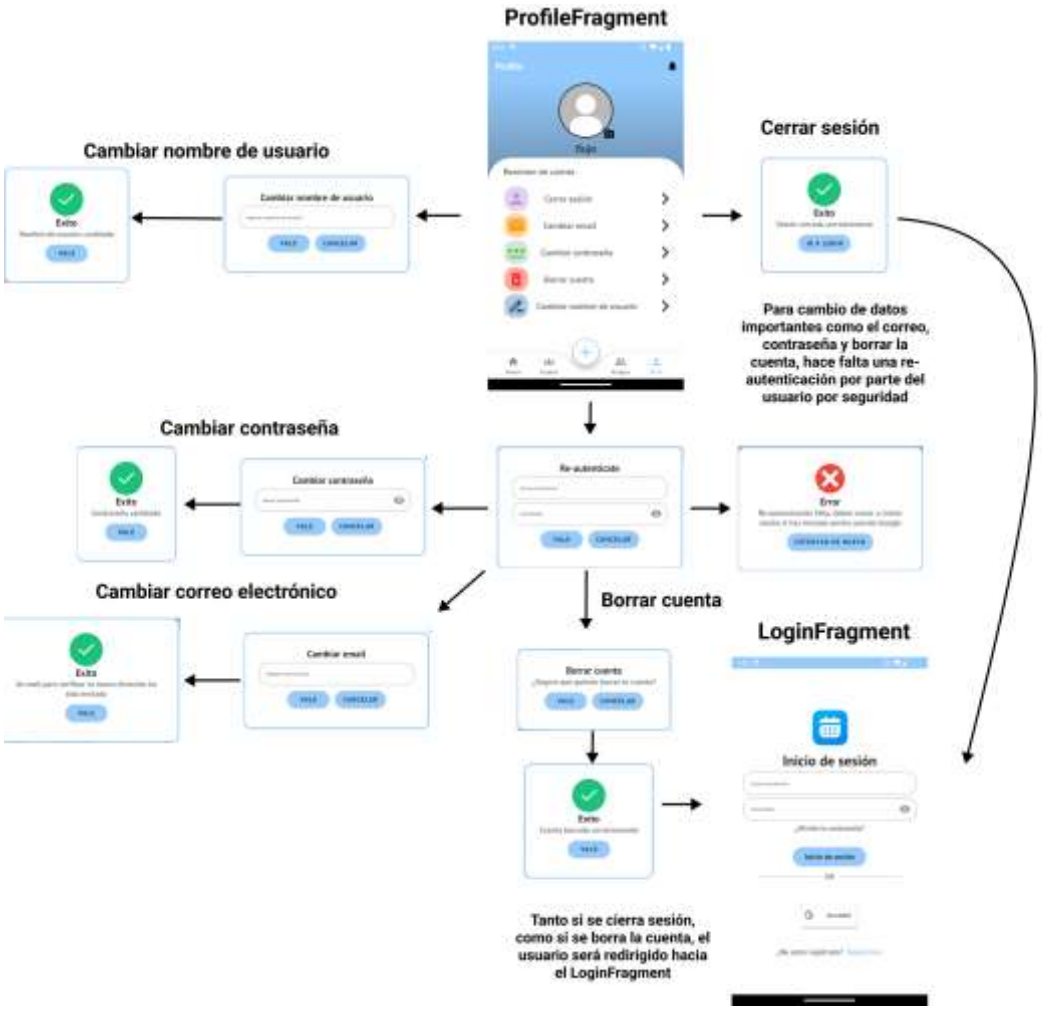
Una vez que el usuario pertenezca a por lo menos un grupo, en el calendario de HomeFragment se le marcarán aquellos días en los cuales hay un grupo, y al pulsar será llevado al GroupsFragment pasando dicha fecha como argumento

Una vez se valide que los datos con los cuales se va a crear el grupo sean correctos, se mostrará un mensaje de éxito y el usuario será llevado hacia el GroupsFragment

GroupsFragment

Al pulsar sobre un grupo el usuario será llevado hacia el DetailsFragment en donde se mostrarán los detalles de este

DetailsFragment


ProfileFragment
Cambiar nombre de usuario
Cerrar sesión
Para cambio de datos importantes como el correo, contraseña y borrar la cuenta, hace falta una re-autenticación por parte del usuario por seguridad
Cambiar contraseña
Cambiar correo electrónico
Borrar cuenta
LoginFragment
Tanto si se cierra sesión, como si se borra la cuenta, el usuario será redirigido hacia el LoginFragment

DetailsFragment

Dentro del DetailsFragment están contenidos ExpensesFragment y BalancesFragment, cambiando cuál se muestra usando las pestañas de la parte superior

ExpensesFragment

BalancesFragment

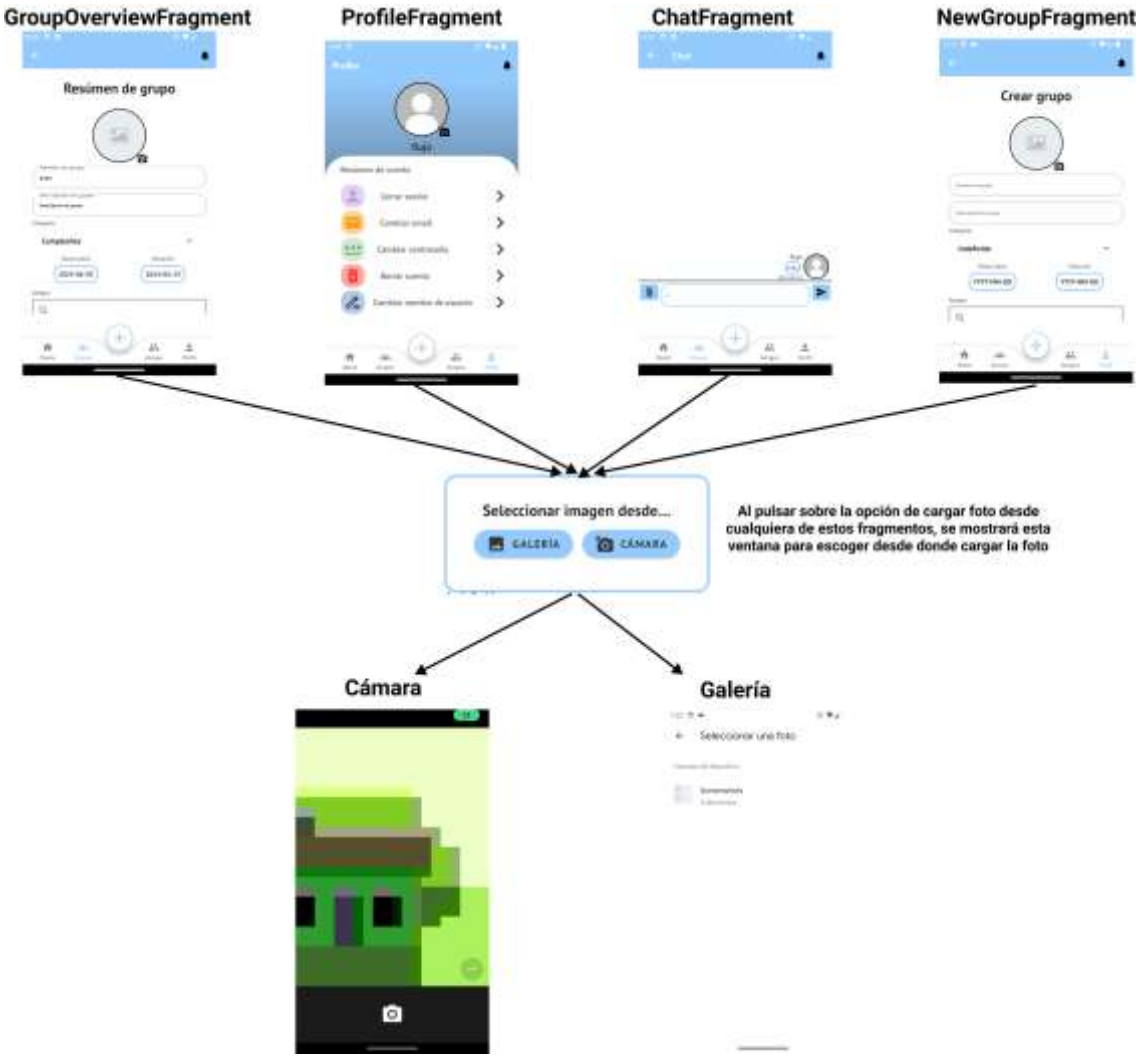
GroupsFragment

Dentro de estos fragmentos si el grupo es borrado o el usuario es expulsado, este será redirigido hacia GroupsFragment

En la parte superior del fragmento al pulsar sobre la información del grupo, el usuario será llevado al GroupOverviewFragment y si pulsa sobre el icono en la derecha al ChatFragment

GroupOverviewFragment

ChatFragment


GroupOverviewFragment
Resúmen de grupo
ProfileFragment
ChatFragment
NewGroupFragment
Crear grupo
Seleccionar imagen desde...
GALERÍA
CÁMARA
Al pulsar sobre la opción de cargar foto desde cualquiera de estos fragmentos, se mostrará esta ventana para escoger desde donde cargar la foto
Cámara
Galería
Seleccionar una foto

CONCLUSIONES

Durante el desarrollo de "Budget Buddy", nuestra aplicación para gestionar gastos compartidos, logramos avanzar bastante bien en la creación de esta.

Implementamos características como la creación de cuentas conjuntas y usamos Firebase como nuestro backend, lo que hizo que la aplicación fuera fácil de usar y eficiente.

Sin embargo, aún tenemos trabajo por hacer. No pudimos terminar completamente algunas partes, como el seguimiento detallado de los saldos de cada grupo y la forma de hacer pagos dentro de la aplicación. Estas son áreas que queremos mejorar en el futuro.

Personalmente, este proyecto fue una gran experiencia de aprendizaje para ambos. Aprendimos mucho sobre cómo planificar y llevar a cabo un proyecto de software. Trabajar en equipo también fue genial; entendimos lo importante que es comunicarse bien para lograr nuestros objetivos. A pesar de los desafíos, estamos orgullosos de lo que logramos y emocionados por el futuro de "Budget Buddy".

BIBLIOGRAFÍA

Boudjnah, E., Forrester, A., & Dumbravan, A. (2023). Android Architecture Components. En E. Boudjnah, A. Forrester, & A. Dumbravan, *How to Build Android Apps with Kotlin - Second Edition: A practical guide to developing, testing, and publishing your first Android apps* (págs. 442-482). Packt Publishing.

Google. (12 de Diciembre de 2023). *Guide to app architecture*. Android Developers. https://developer.android.com/topic/architecture#recommended-app-arch

Google. (22 de 2 de 2024). *Add spinners to your app*. Android Developers. https://developer.android.com/develop/ui/views/components/spinner

Google. (5 de Abril de 2024). *Android's Kotlin-first approach* . Android Developers. https://developer.android.com/kotlin/first

Google. (5 de Abril de 2024). *Android's Kotlin-first approach* . Android Dvelopers. https://developer.android.com/kotlin/first#why

Howarth, J. (6 de Diciembre de 2023). *iPhone vs Android User Stats (2024 Data)*. Exploding Topics. https://explodingtopics.com/blog/iphone-android-users

Khawas, C. (2018). Application of Firebase in Android App Development-A Study. *International Journal of Computer Applications*, 49-53.

Shanahan, M., & Bahia, K. (Octubre de 2023). *The State of Mobile Internet Connectivity*. GSMA.