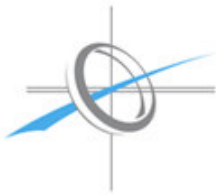
Universidad Politécnica
de Madrid

E.T.S. DE INGENIERÍA
DE SISTEMAS INFORMÁTICOS



Máster en Desarrollo de Aplicaciones y Servicios para Dispositivos Móviles

Trabajo Fin de Máster

Desarrollo de una Aplicación Android para la gestión de gastos e ingresos personales

Autor: Wei Zheng
Tutor(a): Edgar Talavera Muñoz

Madrid, junio 2023

Trabajo Fin de Máster

Máster en Desarrollo de Aplicaciones y Servicios para Dispositivos Móviles

Título: Desarrollo de una Aplicación Android para la gestión de gastos e ingresos personales

junio 2023

Autor: Wei Zheng

Tutor: Edgar Talavera Muñoz
Departamento de Sistemas Informáticos
ETSI de Sistemas Informáticos
Universidad Politécnica de Madrid

Resumen

En este trabajo se presenta el proceso de desarrollo de una aplicación móvil para la gestión de gastos e ingresos en Android. La aplicación tiene como objetivo ofrecer a las personas una forma conveniente y rápida de registrar sus ingresos y gastos, satisfaciendo así las necesidades de la vida cotidiana. Para su desarrollo, se utilizó Android Studio como herramienta principal, SQLite para el almacenamiento de datos locales y Firebase para implementar las funciones de registro e inicio de sesión, así como la base de datos en la nube. Entre las funciones que ofrece la aplicación se encuentran: registro y cambio de cuenta de usuario, agregación de registro, eliminación de registro y modificación de información de ingresos y gastos, clasificación de ingresos y gastos, estadísticas de ingresos y gastos por categoría, estadísticas de ingresos y gastos totales y eliminación de datos, entre otras. La aplicación cuenta con una interfaz de usuario clara y fácil de usar, lo que la hace amigable para el usuario.

En el proceso de desarrollo, se utilizó un enfoque orientado a objetos y se implementó la base de datos en la nube "Firestore" para proteger los datos de usuario y mejorar la seguridad de la aplicación. En cuanto a las estadísticas, se utilizó una técnica adecuada de visualización de datos para mostrar a los usuarios sus ingresos y gastos de manera más intuitiva.

En el futuro, se planea continuar mejorando la aplicación agregando más funcionalidades y mejorando la experiencia del usuario. Se espera que este trabajo contribuya a la comodidad de las personas en su vida diaria.

Palabras clave: Aplicación Android; Firebase; Gestionar; Presupuesto; Analizar; Gastos; Ingresos.

Abstract

This project presents the development process of a mobile application for managing expenses and income on Android. The application aims to offer people a convenient and fast way to record their income and expenses, thus satisfying the needs of daily life. Android Studio was used as the main development tool, SQLite for local data storage, and Firebase to implement registration and login functions, as well as the cloud database. Among the functions offered by the application are registration and change of user account, addition, deletion, and modification of income and expense information, classification of income and expenses, statistics of income and expenses by category, statistics of total income and expenses, data deletion, among others. The application has a clear and easy-to-use user interface, making it user-friendly.

In the development process, an object-oriented approach was used, and the "Firestore" cloud database was implemented to protect user data and improve application security. As for the statistics, an appropriate data visualization technique was used to show users their income and expenses in a more intuitive way.

In the future, it is planned to continue improving the application by adding more functionality and improving the user experience. It is hoped that this work will contribute to the comfort of people in their daily lives.

Keywords: Application Android; Firebase; Managing; Budget; Analyze; expenses; Incomes.

Agradecimientos

Quiero expresar mi agradecimiento a todas las personas que me ayudaron durante mi TFM. Agradezco al profesor Edgar por su valiosa orientación y consejos. También agradezco a mi familia por su incondicional apoyo y comprensión. A mis compañeros, gracias por sus consejos. A mis amigos, gracias por estar ahí y hacerme enfocado en el trabajo. Por último, agradezco mis propios esfuerzos y perseverancia. Gracias a todos por su apoyo en mi viaje del Máster.

Tabla de contenidos

Índice de figuras

1 Introducción

Hoy en día, la tecnología se está desarrollando a una velocidad vertiginosa, integrando en todas partes y en todos los campos del mundo, su influencia en el mundo es bastante significativa. Su enorme poder ha cambiado la vida humana. El concepto de "móvil" surgió y los productos digitales están cada vez más cerca de la vida de las personas. Hace veinte años, los teléfonos móviles solo se presentaron como un dispositivo de comunicación, pero hoy en día, los teléfonos móviles se han convertido en una "computadora" personal. Los diversos tipos de teléfonos móviles y los programas de software complicados están cambiando constantemente la actitud del usuario y provocando la curiosidad de las personas.

Desde la aparición del sistema operativo Android [1] de en 2007, ha dependido de la optimización y nos ha brindado excelentes funciones en redes, multimedia, comunicaciones, entretenimiento, etc. Android, desde su nacimiento hasta junio del año pasado, con un mercado de "70%" [2], nos muestra u popularidad. La razón por la que es tan popular se debe a los desarrolladores de software de terceros y desarrolladores como nosotros. Podemos realizar una personalizada en él.

En este trabajo se va a realizar una aplicación móvil en el sistema operativo Android, utilizando la herramienta Android Studio [3] y plataforma Firebase [4]. La aplicación nos permites registrar los gastos e los ingresos personales, controlar los presupuestos y analizar los datos gráficamente.

1.1 Motivación

Con la rápida desarrollo económico y social, nuestro ritmo de vida se vuelve cada vez más acelerado. Cuando estamos ocupados con muchas cosas, es fácil olvidar algunos detalles de nuestra vida cotidiana, como la gestión de ingresos y gastos. Para liberar más tiempo y disfrutar mejor de la vida, esperamos tener una aplicación que pueda ayudarnos a administrar estos pequeños datos. La gestión de gastos e ingresos personal basado en el sistema operativo Android nos permite registrar fácilmente estos datos en cualquier momento y en cualquier lugar. Ya no tenemos que preocuparnos por nuestros ingresos y gastos, y lo único que tenemos que hacer es mirar los resultados estadísticos con tranquilidad.

1.2 Objetivos

El objetivo principal es diseña y desarrollar una aplicación Android, además la aplicación está dividida en cincos módulos, que son gestión de usuarios, gestión de ingresos, gestión de gastos, gestión de presupuestos y diagrama linear para analizar los registros.

- Implementación de autenticación de los usuarios con Firebase.
- Implementación de registración de los usuarios con Firebase.
- Añadir ingresos.
- Añadir gastos.
- Implementación de descripción y fecha para cada registro.
- Implementación de presupuestos.
- Visualización de los registros del hoy.
- Eliminar registros.
- Visualización de los registros en un diagrama lineal.
- Categorizar los gastos y los ingresos.
- Historias de los registros.

1.3 Estructura de la memoria

La memoria de este trabajo se organiza en las siguientes secciones principales: Estado de arte, Metodología, Desarrollo e Implementación, Conclusiones, Trabajos Futuros y Análisis de Impacto. A continuación, se describe brevemente el contenido de cada una de estas secciones:

1. **Estado de arte:** En esta sección se realiza un análisis del estado de arte relacionado con la gestión financiera personal y empresarial, así como las aplicaciones móviles y herramientas disponibles en el mercado. Se revisan los conceptos clave, las tecnologías utilizadas y las investigaciones previas relevantes en el campo.

2. **Metodología:** En esta sección se describe la metodología utilizada para el desarrollo de la aplicación. Se detallan las etapas del proceso de desarrollo, las herramientas y tecnologías utilizadas, y se analiza los requisitos también como las decisiones de diseño

3. **Desarrollo e Implementación:** En esta sección se presenta el proceso de desarrollo y la implementación de la aplicación. Se describen las funcionalidades clave implementadas, junto con los desafíos técnicos encontrados y las soluciones adoptadas. Además, se proporcionan detalles sobre la integración con Firebase.

4. **Conclusiones:** En esta sección se presentan las conclusiones obtenidas a partir del desarrollo y la implementación de la aplicación. Se resumen los principales logros alcanzados Se evalúa la eficacia y utilidad de la aplicación, tanto desde la perspectiva del usuario como desde la perspectiva técnica.

5. **Trabajos Futuros:** En esta sección se proponen posibles áreas de mejora y trabajos futuros relacionados con la aplicación de gestión financiera. Se identifican las funcionalidades adicionales que podrían implementarse para enriquecer la experiencia del usuario y ampliar la utilidad de la aplicación.

6. **Análisis de Impacto:** En esta sección se realiza un análisis detallado del impacto generado por la aplicación de gestión financiera, tanto a nivel personal como empresarial, social, económico y medioambiental.

2 Estado de arte

En este apartado se describe las diversas tecnologías, lenguaje y herramientas utilizada para el desarrollo de la aplicación móvil.

2.1 Android

Android [1] es un sistema operativo móvil de código abierto basado en Linux y desarrollado por Google. Fue diseñado principalmente para dispositivos móviles como smartphones y tablets, aunque también se utiliza en otros dispositivos como relojes inteligentes, televisores y automóviles.

Una de las características distintivas de Android es su ecosistema de aplicaciones. Los desarrolladores pueden crear aplicaciones para Android utilizando el kit de desarrollo de software (SDK) de Android, que proporciona herramientas para la creación de aplicaciones, el depurado y la publicación en la tienda de aplicaciones de Google Play.

Android también es conocido por su personalización y flexibilidad. Los usuarios pueden personalizar la interfaz de usuario, incluyendo la pantalla de inicio y los widgets, y descargar e instalar aplicaciones de terceros desde la tienda de aplicaciones de Google Play o desde otras fuentes.

Además, Android se integra con muchos servicios de Google, como Gmail, Google Maps, Google Drive, Google Play Music y otros, lo que permite a los usuarios acceder a estos servicios directamente desde sus dispositivos Android.

Figura 1 - Logo Android

2.2 Android Studio

Android Studio [3]es un entorno de desarrollo integrado (IDE) para desarrollar aplicaciones móviles en el sistema operativo Android. Fue desarrollado por Google y está basado en el popular IDE IntelliJ IDEA [5]. Es la herramienta

oficial para desarrollar aplicaciones de Android y se utiliza para escribir código, depurar, probar y empaquetar aplicaciones.

Android Studio ofrece un amplio conjunto de herramientas y características para los desarrolladores de aplicaciones móviles. Incluye un editor de código avanzado con soporte para autocompletado y análisis de código, un depurador para solucionar errores, una herramienta para diseñar interfaces de usuario, una vista previa en tiempo real para ver cómo se verá la aplicación en diferentes dispositivos móviles, una herramienta para generar informes de errores y estadísticas de uso, y una serie de herramientas de prueba para garantizar la calidad de la aplicación.

Además, Android Studio también incluye una amplia gama de plantillas de aplicación para ayudar a los desarrolladores a comenzar rápidamente con la creación de aplicaciones. Las plantillas incluyen aplicaciones básicas de inicio, juegos, aplicaciones de redes sociales, aplicaciones de noticias, aplicaciones de viajes, entre otras.

Figura 2 - Logo Android Studio

2.3 Java

Java [6] es un lenguaje de programación de alto nivel orientado a objetos, que se utiliza en múltiples aplicaciones y sistemas. Fue desarrollado por Sun Microsystems en la década de 1990 y adquirido posteriormente por Oracle Corporation. Una de las características principales de Java es que se puede ejecutar en diferentes plataformas, incluyendo sistemas operativos como Windows, Linux y macOS, y dispositivos móviles como smartphones y tablets.

Java ha sido ampliamente utilizado en el desarrollo de aplicaciones empresariales y de servidor, así como en la creación de aplicaciones móviles, juegos, herramientas de desarrollo y mucho más. El lenguaje es conocido por su seguridad, portabilidad y robustez, lo que lo hace una elección popular en el mundo de la programación.

Java se basa en la programación orientada a objetos (POO) [7], lo que significa que los programadores pueden diseñar soluciones y sistemas utilizando objetos que interactúan entre sí. El lenguaje también cuenta con una amplia biblioteca de clases y métodos predefinidos, que facilitan la programación y el desarrollo de soluciones complejas.

Java también se destaca por su arquitectura de máquina virtual Java (JVM), que permite que el código fuente se compile en un formato especial llamado bytecode. Este bytecode se puede ejecutar en cualquier plataforma que tenga una JVM instalada, lo que lo hace altamente portátil y evita la necesidad de compilar el código para cada plataforma de forma separada.

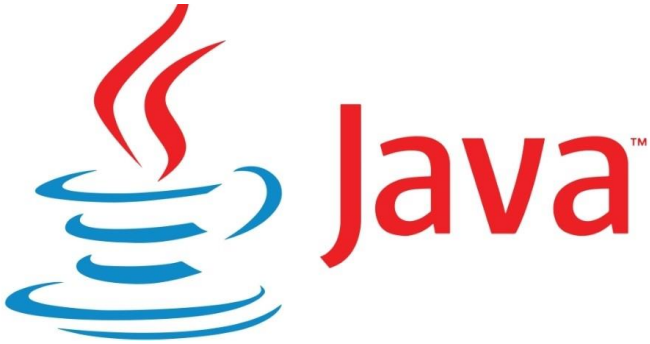


Figura 3 - Logo Java

2.4 Firebase

Firebase [4] es una plataforma en la nube de Google que proporciona una variedad de herramientas y servicios para desarrollar aplicaciones web y móviles de alta calidad. Está diseñado para ayudar a los desarrolladores a crear y escalar aplicaciones de forma rápida y eficiente, sin tener que preocuparse por la gestión de la infraestructura.

Firebase ofrece servicios para autenticación de usuarios, almacenamiento en la nube, bases de datos en tiempo real, mensajería, análisis, entre otros. Estas herramientas permiten a los desarrolladores crear aplicaciones que sean altamente escalables y fáciles de mantener.

Además, Firebase proporciona una amplia variedad de bibliotecas y SDK para varios lenguajes de programación, como Java, Kotlin, Swift, Objective-C, JavaScript, entre otros. Estas herramientas ayudan a los desarrolladores a integrar fácilmente las diferentes funcionalidades de Firebase en sus aplicaciones.

Figura 4 - Logo Firebase

2.4.1 Firebase Authentication

Firebase Authentication [8] es un servicio de Firebase que proporciona una forma fácil de autenticar a los usuarios en una aplicación móvil o web. Con Firebase Authentication, los desarrolladores pueden implementar la autenticación en sus aplicaciones de manera rápida y segura, sin tener que preocuparse por la gestión de usuarios, el almacenamiento de contraseñas y otros aspectos de seguridad.

Firebase Authentication ofrece una amplia gama de opciones de autenticación, que van desde la autenticación por correo electrónico y contraseña hasta la autenticación con redes sociales como Google, Facebook, Twitter y GitHub, y otras opciones de autenticación como la autenticación por SMS y la autenticación con proveedores de identidad personalizados.

Una vez que un usuario se ha autenticado, Firebase Authentication proporciona un token de acceso seguro que puede utilizarse para acceder a otros servicios de Firebase, como Cloud Firestore o Cloud Storage. También permite a los desarrolladores personalizar el flujo de autenticación y el aspecto de la interfaz de usuario para ofrecer una experiencia de autenticación coherente con la marca de su aplicación.

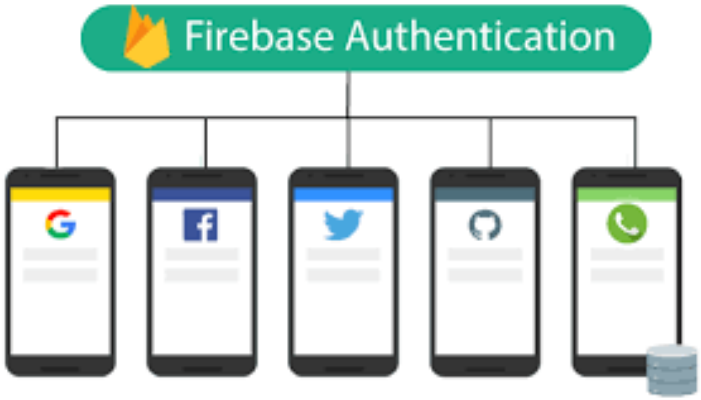


Figura 5 - Proveedores de Firebase Authentication

2.4.2 Firebase Firestore

Firebase Firestore [9] es un servicio de base de datos NoSQL en tiempo real y en la nube, desarrollado por Google como parte de la plataforma Firebase. Firestore es una base de datos escalable que permite almacenar y sincronizar datos en tiempo real en dispositivos móviles y en la web. Además, Firestore ofrece una API sencilla para acceder y actualizar datos en tiempo real desde múltiples clientes. También cuenta con características como consultas complejas, transacciones, índices y seguridad integrada para garantizar la integridad de los datos. Firestore es una excelente opción para aplicaciones que necesitan un almacenamiento de datos en tiempo real y una escalabilidad flexible en la nube.

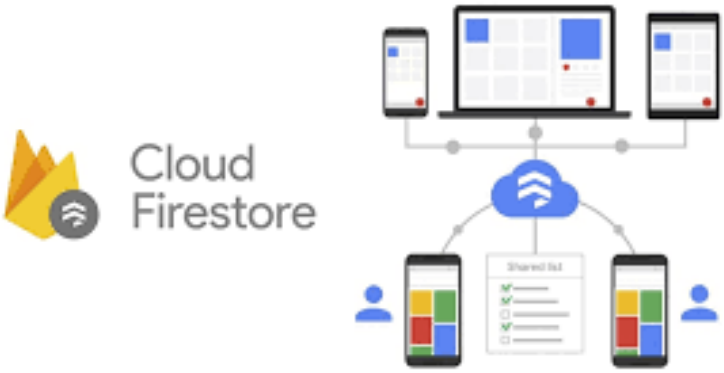


Figura 6 – Base de datos de la nube Firestore

2.5 MPAndroidChart

MPAndroidChart [10] es una biblioteca de gráficos de código abierto para la plataforma Android. Permite a los desarrolladores de aplicaciones mostrar datos en forma de gráficos interactivos en sus aplicaciones. La biblioteca es altamente personalizable y admite varios tipos de gráficos, incluidos gráficos de barras, líneas, área, tarta, radar y dispersión. MPAndroidChart también admite características avanzadas como zoom y desplazamiento, animaciones, múltiples ejes, leyendas personalizables y personalización de colores y estilos. Esta biblioteca es muy popular entre los desarrolladores de Android debido a su facilidad de uso y su capacidad para crear gráficos interactivos y atractivos en aplicaciones móviles.

Figura 7 – Logo MPAndroidChart

2.6 Conocimientos utilizados para el diseño y el desarrollo

2.6.1 Layout

El diseño de una aplicación en Android es una parte importante del desarrollo. Se puede utilizar diferentes métodos, como archivos XML [11], código Java o bibliotecas de terceros. Los archivos XML son el método más común y fácil de mantener, mientras que el código Java es más flexible, pero requiere más habilidades. También existen view group populares como ConstraintLayout, LinearLayout y RelativeLayout que proporcionan una amplia gama de funciones y APIs fáciles de usar. Elegir el método de diseño de un proyecto de Android adecuado es esencial para lograr una interfaz de usuario eficiente y satisfactoria para el usuario.

2.6.2 Activity

Activity es un componente fundamental en el desarrollo de aplicaciones de Android. Representa una pantalla con la que el usuario puede interactuar y puede contener diferentes elementos de la interfaz de usuario, como botones, campos de texto, imágenes, etc. Las actividades son administradas por el sistema operativo Android y se pueden comunicar entre sí mediante intenciones.

2.6.2.1 Ciclo de vida de activity

El ciclo de vida de Activity [12] en Android es importante para administrar recursos y proporcionar una experiencia de usuario fluida y sin errores. Al entender cómo funciona el ciclo de vida de Activity, los desarrolladores pueden crear aplicaciones más estables y eficientes. El ciclo de vida se puede dividir en varios estados (ver Figura 8):

1. onCreate(): Este es el primer método que se llama cuando se crea la actividad. Aquí es donde se inicializan los recursos y se infla la interfaz de usuario.

2. onStart(): Después de onCreate(), la actividad pasa al estado de onStart(). En este estado, la actividad es visible para el usuario, pero aún no está en primer plano.

3. onResume(): Después de onStart(), la actividad entra en el estado de onResume(). En este estado, la actividad está en primer plano y el usuario puede interactuar con ella.

4. onPause(): Cuando el usuario sale de la actividad o se superpone con otra actividad, la actividad pasa al estado de onPause(). En este estado, la actividad aún es visible para el usuario, pero no está en primer plano.

5. onStop(): Si el usuario continúa navegando por la aplicación o sale de ella, la actividad pasa al estado de onStop(). En este estado, la actividad ya no es visible para el usuario.

6. onDestroy(): Si la actividad se cierra o se destruye, se llama al método onDestroy(). Este es el último estado del ciclo de vida de la actividad.

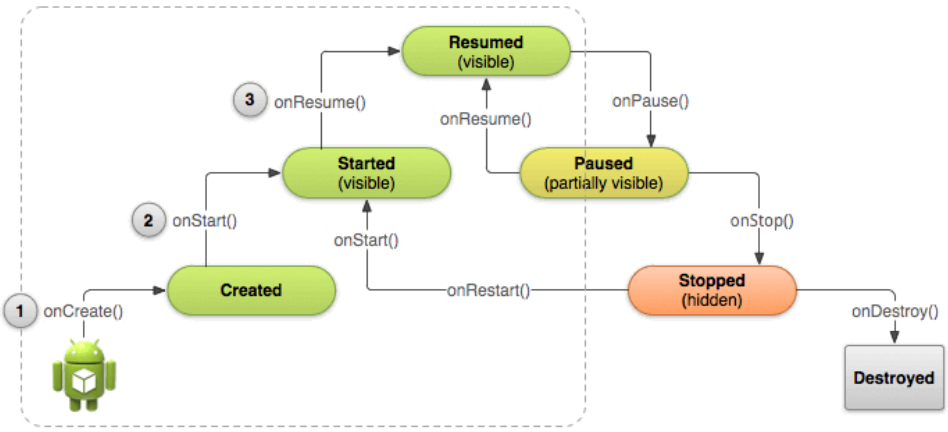


Figura 8 - Ciclo de vida de activity

2.6.3 Fragment

Fragment representa una parte modular de una actividad. Al igual que las actividades, los fragmentos también tienen su ciclo de vida, que incluye estados como onCreate(), onStart(), onResume(), onPause(), onStop() y onDestroy().

Fragment es una forma útil de crear interfaces de usuario reutilizables y dinámicas en Android. Puede combinarse para crear una interfaz de usuario multipanel y también permiten una mejor gestión de los recursos.

2.6.4 Dialog

Dialog es un componente de la interfaz de usuario que se utiliza para mostrar información o solicitar entrada del usuario en una ventana emergente. Los diálogos se crean utilizando la clase Dialog y se pueden personalizar para adaptarse a las necesidades de la aplicación.

2.6.5 ListView/GridView

GridView y ListView son dos componentes comunes en la interfaz de usuario en Android que se utilizan para mostrar una lista de elementos. Aunque ambos componentes tienen funciones similares, hay algunas diferencias clave entre ellos.

ListView es un componente que muestra una lista vertical de elementos en una pantalla. Cada elemento en la lista se puede personalizar utilizando un diseño personalizado. Los elementos se pueden desplazar verticalmente y se pueden incluir en la lista elementos de diferentes tipos, como texto, imágenes, casillas de verificación, etc.

Por otro lado, GridView es un componente que muestra una lista de elementos en una cuadrícula. Cada elemento en la cuadrícula se puede personalizar utilizando un diseño personalizado. Los elementos se pueden desplazar vertical y horizontalmente y se pueden incluir en la cuadrícula elementos de diferentes tipos, como texto, imágenes, casillas de verificación, etc.

2.6.6 SQLite

SQLite [13] es una base de datos relacional de código abierto que se utiliza en una variedad de aplicaciones y sistemas operativos, incluyendo Android. Es una biblioteca que proporciona un motor de base de datos SQL completo que se integra fácilmente en una aplicación.

SQLite es liviano, fácil de usar y no requiere de un servidor externo para almacenar y recuperar datos. Es adecuado para aplicaciones móviles y de escritorio que necesitan almacenar y recuperar datos de forma local.

En Android, SQLite es la base de datos predeterminada para almacenar y recuperar datos en una aplicación. Proporciona una interfaz de programación de aplicaciones (API) completa para interactuar con la base de datos, lo que permite a los desarrolladores crear, actualizar, eliminar y consultar datos en la base de datos.

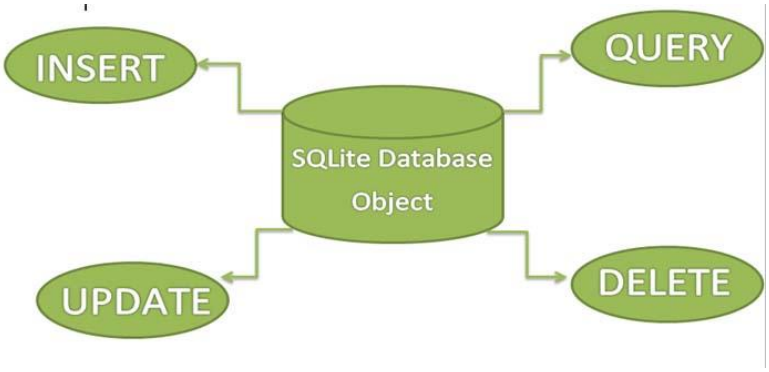


Figura 9 – Query de SQLite

2.7 Aplicaciones similares

A continuación, se mostrarán algunas aplicaciones existentes en el mercado.

2.7.1 Buddy

Buddy [14] es una herramienta de gestión financiera personal que se ejecuta en dispositivos móviles Android. Esta aplicación permite a los usuarios llevar un registro de sus ingresos y gastos diarios y hacer un seguimiento de sus finanzas personales de manera fácil y rápida.

La aplicación Buddy cuenta con una variedad de funciones y herramientas que permiten a los usuarios realizar un seguimiento detallado de sus transacciones financieras, establecer objetivos de ahorro y presupuestos, así como generar informes detallados y gráficos estadísticos para visualizar el desempeño de sus finanzas.

Buddy utiliza una interfaz de usuario intuitiva y fácil de usar que permite a los usuarios agregar y clasificar fácilmente sus ingresos y gastos, establecer alertas y recordatorios para facturas y pagos, y sincronizar sus datos de forma segura en la nube a través de la plataforma Firebase de Google.

Figura 10 - Logo Buddy

Ventajas:

- Fácil de usar: la aplicación es intuitiva y fácil de navegar, accesible para todo tipo de usuarios, incluso aquellos que no están familiarizados con las finanzas personales.
- Personalizable: permite a los usuarios configurar sus propias categorías de gastos e ingresos, adaptando la aplicación a sus necesidades específicas.
- Sincronización en múltiples dispositivos: la aplicación se sincroniza automáticamente en todos los dispositivos del usuario.
- Análisis de datos: Buddy proporciona una visión general de los ingresos y gastos del usuario en un formato visual y fácil de entender, lo que facilita el seguimiento.
- Recordatorios: la aplicación envía recordatorios de pagos pendientes y fechas de vencimiento.

Desventajas:

- Limitaciones en la versión gratuita: la versión gratuita de la aplicación tiene algunas limitaciones, como un número limitado de categorías y transacciones que se pueden agregar.
- Problemas de sincronización: algunos usuarios han informado problemas de sincronización entre dispositivos y la aplicación web.
- Requiere ingreso manual de datos: para mantener la precisión de la información, es necesario ingresar manualmente todos los ingresos y gastos en la aplicación.

- Funciones confusas: algunas funciones avanzadas, como la creación de presupuestos y el seguimiento de objetivos financieros, pueden ser confusas para algunos usuarios y requerir más tiempo para aprender cómo utilizarlas adecuadamente.

2.7.2 Monefy

Monefy [15] es una aplicación móvil útil para el seguimiento de gastos y presupuestos personales, con una interfaz fácil de usar y la posibilidad de sincronizar datos en varios dispositivos. Sin embargo, algunas características avanzadas solo están disponibles en la versión de pago y no tiene una función de seguimiento de ingresos y gastos automático.

Figura 11 - Logo Monefy

Ventajas:

- Permite un seguimiento detallado de los gastos y ayuda a identificar patrones de consumo.
- La interfaz es atractiva y fácil de usar.
- Ofrece una opción de sincronización en la nube para tener los datos disponibles en varios dispositivos.
- Permite establecer un presupuesto mensual y recibir alertas cuando se está cerca de superarlo.

Desventajas:

- Algunas características avanzadas, como la importación de datos bancarios, solo están disponibles en la versión de pago de la aplicación.
- No tiene una función de seguimiento de ingresos y gastos automático.
- No ofrece la posibilidad de establecer objetivos de ahorro específicos.

2.7.3 Money Flow

Money Flow [16] es una aplicación móvil de gestión de gastos y finanzas personales que ayuda a los usuarios a controlar sus gastos diarios y mantener sus finanzas bajo control. Permite a los usuarios realizar un seguimiento de sus ingresos y gastos, crear presupuestos, establecer objetivos de ahorro y recibir notificaciones para recordarles el pago de facturas y otras obligaciones financieras. La aplicación también ofrece funciones de análisis de datos y estadísticas, lo que permite a los usuarios comprender mejor sus hábitos de gastos y hacer ajustes en consecuencia.

Figura 12 - Logo Money Flow

Ventajas:

- Interfaz sencilla e intuitiva, que facilita la gestión de las finanzas personales.
- Registro rápido y fácil de transacciones.
- Posibilidad de establecer y monitorear presupuestos en diferentes categorías.
- Sincronización de datos y copias de seguridad en la nube para evitar la pérdida de información.

Desventajas:

- Algunas funciones están restringidas a la versión premium, lo que puede ser una desventaja para algunos usuarios.
- La aplicación no ofrece la posibilidad de conectar directamente con cuentas bancarias.
- Aunque la aplicación permite personalizar la moneda y el idioma, puede que no esté disponible en algunos idiomas específicos.

3 Metodología

En este apartado se describen los requisitos de la aplicación, las características que debe tener y las funcionalidades que deben estar presentes. Además, se establecen las necesidades de los usuarios y se definen los casos de uso que se deben tener en cuenta. También se detallan los flujos, como las plataformas compatibles y los recursos necesarios para su implementación. En general, esta parte es fundamental para garantizar el éxito del proyecto y la satisfacción de los usuarios.

3.1 Requisitos funcionales

Estos requisitos describen las funcionalidades y características específicas que la aplicación debe tener para satisfacer las necesidades de los usuarios. También ayudará a guiar el proceso de diseño, implementación y prueba de la aplicación.

REQ01	Registrar en la aplicación
Descripción	El usuario podrá registrar una cuenta con su dirección de correo electrónico.
Dependencias	• Permisos internet
comentario	El usuario podrá registrarse en la aplicación utilizando su dirección de correo electrónico.

REQ02	Login de usuarios
Descripción	El usuario podrá iniciar sesión en la aplicación.
Dependencias	• REQ01
comentario	El usuario podrá iniciar sesión si ha registrado previamente. Además, también podrá iniciar sesión con proveedores de terceros como Google.

REQ03	Agregar un registro
Descripción	El usuario podrá agregar un registro
Dependencias	• REQ02
comentario	El usuario podrá agregar un registro de ingreso o gasto deslizando la pantalla o haciendo clic en las opciones. Además, podrá elegir la categoría del registro, agregar una descripción y la fecha correspondiente.

REQ04	Agregar una descripción
Descripción	El usuario podrá agregar una descripción para el registro
Dependencias	• REQ02 • REQ03
comentario	El usuario podrá agregar una descripción para el registro que desea agregar haciendo clic en "Agregar descripción".

REQ05	Agregar una fecha
Descripción	El usuario podrá agregar un registro.
Dependencias	• REQ02 • REQ03
comentario	El usuario podrá agregar una fecha para el registro que desea agregar haciendo clic en "fecha", luego seleccionando el año, mes y día correspondiente, además podrá rellenar la hora y minutos.

REQ06	Visualizar todos los registros de hoy
Descripción	Visualizar todos los registros de hoy.
Dependencias	• REQ02 • REQ03 • REQ05
comentario	El usuario podrá visualizar todos los registros agregados cuya fecha sea la de hoy, deslizando la pantalla para obtener más registros. Además, podrá ver el gasto total y el ingreso total de hoy.

REQ07	Ingresos y gastos totales de mes
Descripción	El usuario podrá obtener los ingresos y gastos totales de mes.
Dependencias	• REQ02 • REQ03 • REQ05
comentario	La aplicación mostrará los ingresos y gastos totales del mes en la interfaz principal.

REQ08	La ocultación de datos
Descripción	El usuario podrá ocultar los datos de la pantalla principal.
Dependencias	• REQ02 • REQ03 • REQ07
comentario	El usuario podrá ocultar los datos de la pantalla principal haciendo clic en el icono de “ojo”.

REQ09	Demostración de datos
Descripción	El usuario podrá mostrar los datos de la pantalla principal.
Dependencias	• REQ02 • REQ03 • REQ07 • REQ08
comentario	El usuario podrá mostrar los datos de la pantalla principal haciendo clic en el icono de “ojo”.

REQ010	Buscar registros
Descripción	El usuario podrá buscar los registros agregados.
Dependencias	• REQ02 • REQ03 • REQ04
comentario	El usuario podrá buscar los registros utilizando las descripciones.

REQ11	Analizar datos
Descripción	El usuario obtener análisis de datos en un diagrama lineal
Dependencias	• REQ02 • REQ03 • REQ07
comentario	El usuario podrá seleccionar un mes para analizar los datos de ingresos o gastos, y la información se mostrará de dos maneras. • Diagrama lineal: muestra los ingresos o gastos totales de cada día del mes. • Lista: muestra los ingres y gastos totales de cada categoría.

REQ12	More
Descripción	El usuario podrá hacer clic en el botón de “More”
Dependencias	• REQ02 • REQ03
comentario	El usuario podrá hacer clic en el botón de “More” para realizar más operaciones.

REQ13	Borrar todos los registros
Descripción	El usuario podrá borrar todos los registros.
Dependencias	• REQ02 • REQ03
comentario	El usuario podrá borrar todos los registros agregados haciendo clic en el botón “borrar”.

REQ14	Ver historial de registros
Descripción	El usuario podrá ver todos los registros agregados.
Dependencias	• REQ02 • REQ03
comentario	El usuario podrá seleccionar un mes para ver todos los registros del mes haciendo clic en “historia”.

REQ15	Eliminar un registro
Descripción	El usuario podrá eliminar un registro.
Dependencias	• REQ02 • REQ03
comentario	Cuando el usuario mantenga presionado un registro durante un tiempo, podrá eliminarlo.

REQ16	Cerrar sesión
Descripción	El usuario podrá cerrar sesión actual.
Dependencias	• REQ02
comentario	El usuario podrá cerrar sesión actual haciendo clic en “salir”.

3.2 Requisitos no funcionales

En esta parte, se describirán los requisitos no funcionales de la aplicación móvil que se está desarrollando, lo que ayudará a garantizar que la aplicación cumpla con los estándares necesarios en cuanto a calidad, usabilidad y rendimiento.

- **Seguridad:** la aplicación debe cumplir con ciertos estándares de seguridad para proteger la información del usuario y evitar el acceso no autorizado.
- **Usabilidad:** la aplicación debe ser fácil de usar y tener una interfaz de usuario intuitiva que permita a los usuarios realizar operaciones de manera efectiva y eficiente.
- **Rendimiento:** la aplicación debe responder rápidamente a las operaciones, sin tiempos de espera excesivos o errores de rendimiento.
- **Disponibilidad:** la aplicación debe estar disponible para su uso en todo momento, sin interrupciones inesperadas en el servicio.
- **Compatibilidad:** la aplicación debe ser compatible con diferentes dispositivos móviles de sistema operativo Android.
- **Mantenibilidad:** la aplicación debe ser fácil de mantener y actualizar, lo que implica que el código debe ser modular y fácil de entender y modificar.
- **Fiabilidad:** la aplicación debe ser confiable y no debe fallar o causar errores en el funcionamiento del dispositivo o la pérdida de datos.

3.3 Estructura del sistema

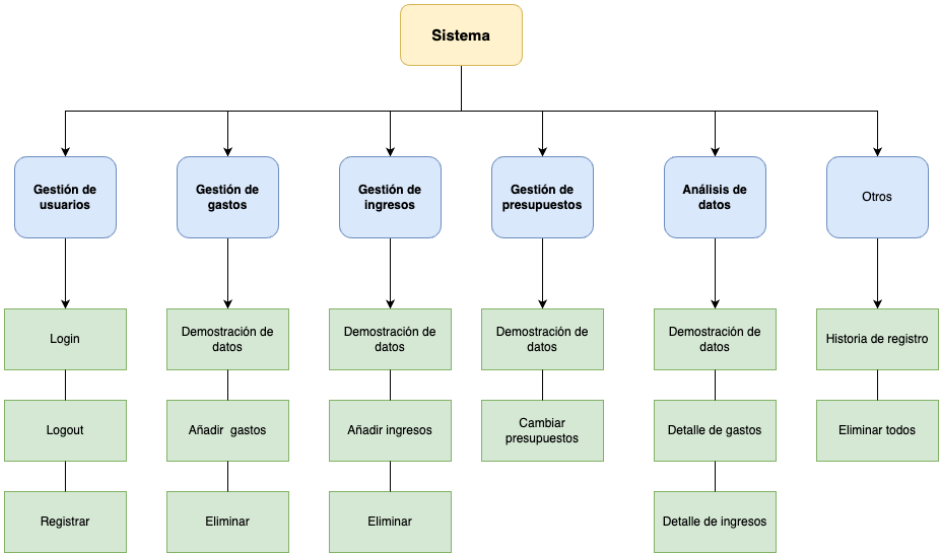


Figura 13 - Diagrama de estructura de sistema, dividido en módulos

El propósito de dividir el sistema en múltiples módulos es reducir la complejidad del sistema, mejorar la legibilidad y el mantenimiento, pero la división de los módulos no puede ser arbitraria y debe mantenerse lo más independiente posible. Es decir, cada módulo solo realiza las subfunciones independientes requeridas por el sistema, y tiene la menor conexión con otros módulos y una interfaz simple, es decir, intenta lograr una alta cohesión y un bajo acoplamiento, mejorar la independencia de los módulos, y diseñar estructuras de software de alta calidad.

3.4 Diseño

En este apartado se explicarán los diseños del sistema. En primer lugar, se utiliza Firebase Auth para implementar la funcionalidad de registro e inicio de sesión, así como para la base de datos en la nube, lo que garantiza la seguridad y fiabilidad de los datos del usuario. En segundo lugar, con respecto al diseño de la interfaz de usuario, se enfoca en mejorar la experiencia del usuario y la claridad del diseño mediante un estilo conciso y claro y una operación sencilla, lo que permite al usuario registrar y gestionar fácilmente sus ingresos y gastos. Finalmente, en cuanto a la estructura del sistema, se ha puesto atención en la escalabilidad y facilidad de mantenimiento del sistema, dividiendo el sistema en viarios módulos de forma razonable y diseñando una estructura lógica que permita ampliar la funcionalidad y actualizar el sistema de manera eficiente.

3.4.1 Firestore

En este trabajo, se utiliza Firebase Firestore [9] para almacenar los datos. Al diseñar el modelo de datos, se consideraron los siguientes puntos: En primer lugar, se definen tres colecciones: una para almacenar información de ingresos y otra para almacenar información de gastos. Cada colección contiene múltiples documentos, cada uno de los cuales representa un registro de ingreso o gasto específico, la tercera colección guarda los datos de los usuarios como presupuestos. En segundo lugar, se definen los campos que deben almacenarse en cada documento, como la cantidad de ingresos, la fecha, la categoría, la descripción y el correo electrónico.

Figura 14 - Colecciones utilizadas en Firestore

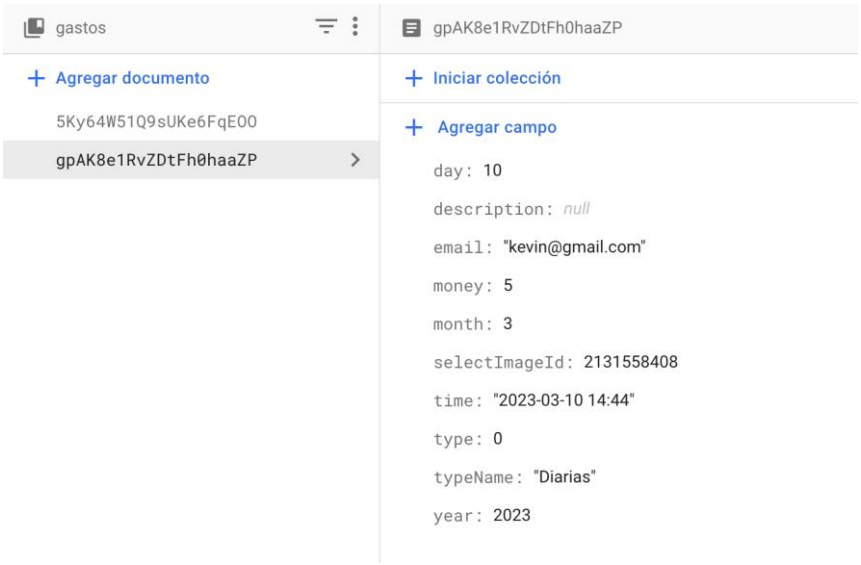


Figura 15 - Documentos de la colección gastos/ingresos, y campos de cada documento

Por último, se considera la seguridad y los permisos de acceso a los datos y se diseñan reglas de seguridad adecuadas para restringir el acceso de los usuarios a los datos y garantizar la seguridad y la integridad de los datos.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
          request.time < timestamp.date(2023, 8, 30);
    }
  }
}
```

Figura 16 - regla de seguridad de para cloud Firestore

3.4.2 Autenticación

Al utilizar “Firebase Authentication” [8] en el sistema, se permite a los usuarios registrarse e iniciar sesión en la aplicación mediante correo electrónico o utilizando proveedores como Google. Al registrarse, los usuarios proporcionan su correo electrónico, contraseña y otra información personal relevante. Por otro lado, se ha diseñado una interfaz de usuario clara y fácil de usar para el proceso de registro e inicio de sesión. Los usuarios pueden elegir entre iniciar sesión con su correo electrónico o con su cuenta de Google, y el proceso de inicio de sesión

es rápido y sencillo. Con esto, se garantiza la seguridad y la privacidad de los datos personales de los usuarios y se les mejora la experiencia de usuario fluida y sin interrupciones.

Figura 17 - Gestión de usuarios desde la consola de Firebase

3.4.3 Módulo de autenticación

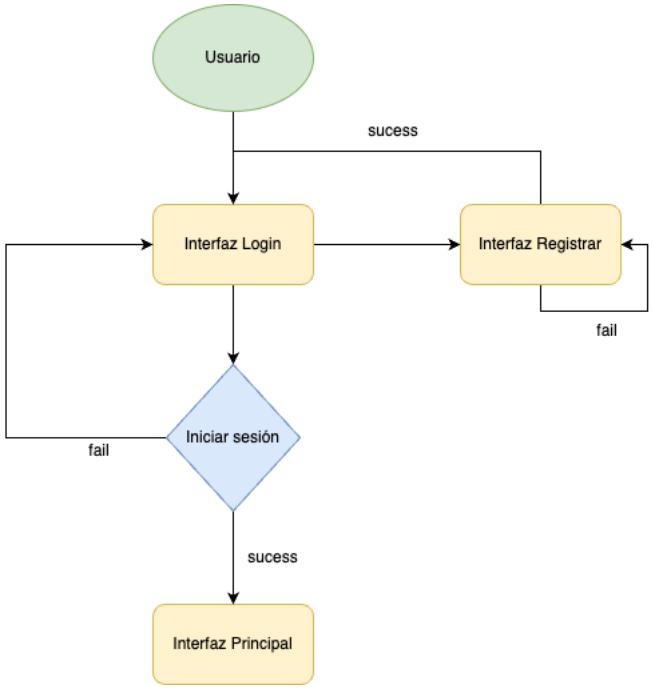


Figura 18 - Diagrama de autenticación,

Este módulo detalla el sistema de inicio de sesión de esta aplicación. Cuando el usuario abre la aplicación, se le redirige a la pantalla de “Login”. En esta pantalla, el usuario puede elegir iniciar sesión con su cuenta de correo electrónico o su cuenta de Google. Este sistema de inicio de sesión utiliza el módulo de autenticación de Firebase para garantizar que la información de la cuenta del usuario esté completamente segura.

Si el usuario elige iniciar sesión con su cuenta de correo electrónico, deberá completar su dirección de correo electrónico registrada y su contraseña. Si la información de la cuenta ingresada es correcta, el sistema verificará

automáticamente la identidad del usuario y lo redirigirá a la pantalla principal. Si la información de la cuenta ingresada no es correcta, el sistema pedirá al usuario que ingrese la información de la cuenta nuevamente hasta que se verifique la identidad del usuario.

Si el usuario elige iniciar sesión con su cuenta de Google, el sistema lo guiará a través de la página de inicio de sesión de Google, donde deberá ingresar la información de su cuenta de Google para completar el inicio de sesión. Si el inicio de sesión es exitoso, el sistema verificará la identidad del usuario y lo redirigirá a la pantalla principal. Si el inicio de sesión falla, el sistema pedirá al usuario que intente iniciar sesión nuevamente hasta que se verifique su identidad.

Si el usuario no tiene una cuenta, puede elegir registrarse y acceder a la pantalla de “Registrar”. En esta pantalla, el usuario debe ingresar su dirección de correo electrónico, contraseña y confirmar la contraseña. Si la información ingresada por el usuario es válida y no se ha registrado anteriormente, el sistema creará automáticamente una nueva cuenta para el usuario y lo redirigirá a la pantalla de inicio de sesión. Si la información ingresada por el usuario es inválida o ya ha sido registrada, el sistema pedirá al usuario que ingrese la información de la cuenta nuevamente.

El diseño de este sistema de inicio de sesión tiene como objetivo permitir que los usuarios inicien sesión en el sistema de manera conveniente y segura, al mismo tiempo que proporciona la funcionalidad de registro de cuenta para usuarios que no tienen una cuenta.

3.4.4 Módulo de agregar registros

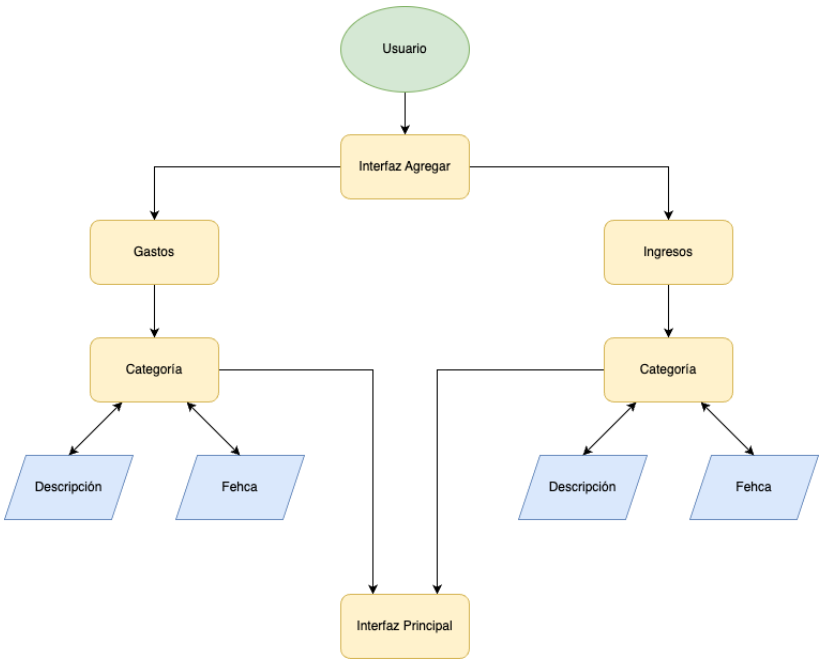


Figura 19 - Diagrama de agregación de registro

Este módulo presenta el sistema de agregar registro, que permite a los usuarios registrar fácilmente sus ingresos y gastos en cualquier momento. Al agregar un registro, los usuarios pueden seleccionar si se trata de un ingreso o un gasto, y luego elegir un tipo específico, como compras, comida, transporte, entre otros. Además, los usuarios pueden optar por agregar una descripción y especificar una fecha para que el registro sea más detallado y preciso. Si no se especifica una fecha, el sistema utilizará la hora actual como fecha del registro. El objetivo de este módulo es permitir a los usuarios registrar sus ingresos y gastos de manera fácil y eficiente, y ayudarlos a administrar mejor sus finanzas. Una vez que se ha agregado un registro, la aplicación volverá automáticamente a la interfaz principal y actualizará los datos.

3.4.5 Módulo de análisis de datos

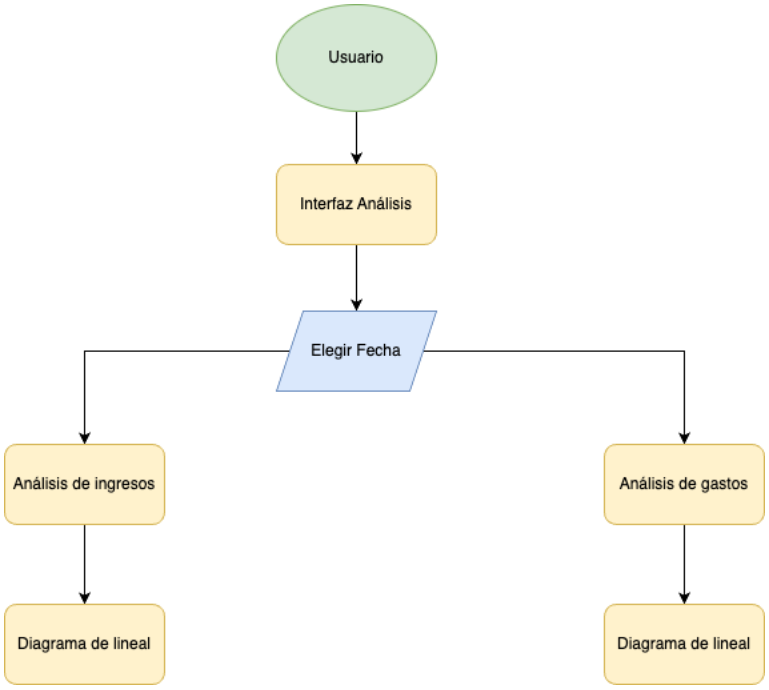


Figura 20 - Diagrama de análisis de datos, para los registros agregados

Este módulo describe la función de estadísticas de datos de la aplicación. Cuando el usuario abre esta pantalla, la aplicación proporcionará al usuario un análisis del mes actual, incluidos los gastos e ingresos totales de cada categoría, además de proporcionar un gráfico lineal para analizar los gastos e ingresos diarios, permitiendo al usuario comprender directamente todos los datos. Por supuesto, aparte del mes actual, el usuario también puede elegir otros meses para su análisis.

3.4.6 Módulo de configuración

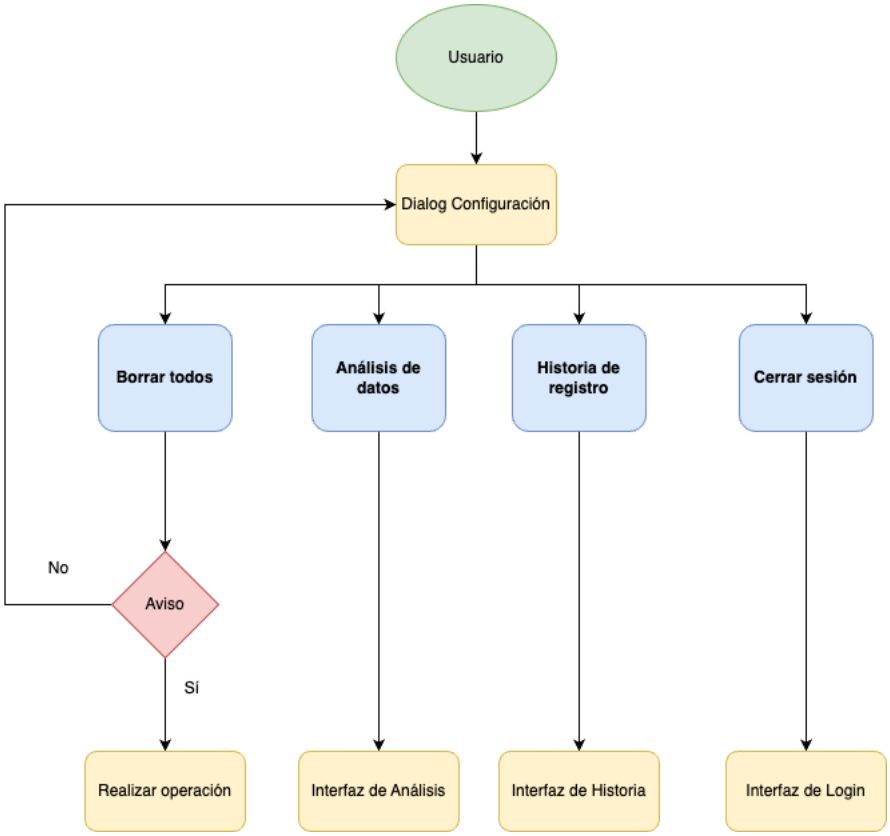


Figura 21 - Diagrama de configuración

Este módulo describe la configuración de esta aplicación. Cuando el usuario hace clic en el botón de configuración en la pantalla principal, la aplicación mostrará un diálogo en la parte inferior de la pantalla. Hay cuatro operaciones diferentes. La primera opción es la eliminación de todos los registros, cuando el usuario hace clic en el botón de eliminación, se borrarán todos los registros existentes. El segundo es el botón de análisis de datos, al hacer clic en él se redirigirá al usuario a la página de análisis de datos. El tercero es el botón de historial, que redirigirá al usuario a la página de historial y mostrará todos los registros agregados. La última opción es el botón de cerrar sesión, al hacer clic en él, el usuario cerrará sesión en su cuenta actual y se redirigirá a la pantalla de Login.

3.4.7 SQLite

En esta aplicación, se utiliza la base de datos SQLite [13] para almacenar las categorías de registro, de manera que los usuarios puedan seleccionar fácilmente un tipo específico para registrar sus ingresos y gastos. Para ello, se crea una tabla llamada "typetb", que contiene los siguientes elementos:

- **typename:** El nombre de la categoría. Este campo almacena el nombre de cada categoría, como "Comida", "Transporte", "Salario", etc. Esto ayuda a los usuarios a seleccionar rápidamente el tipo que desean registrar.

- **unselectImageId:** ID del icono cuando no está seleccionado. Este campo almacena el ID del icono correspondiente al tipo que no está seleccionado. En esta aplicación, se utilizan iconos para representar cada tipo, de manera que los usuarios puedan identificar visualmente cada categoría de forma rápida.

- **selectImageId:** ID del icono cuando está seleccionado. Este campo almacena el ID del icono correspondiente al tipo cuando el usuario selecciona una categoría específica. De esta forma, los usuarios pueden confirmar visualmente el tipo que han seleccionado.

- **kind:** En esta aplicación, se separan los ingresos y los gastos en dos tipos distintos. De esta manera, los usuarios pueden diferenciar fácilmente sus ingresos y gastos registrados.

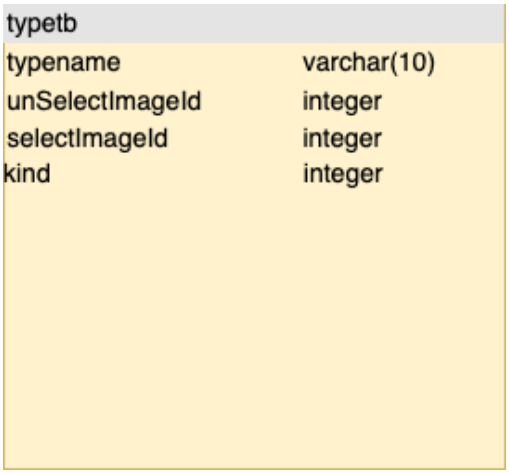


Figura 22 - Tabla de categoría de BBDD

3.4.8 Diseño de las interfaces

La parte de diseño de interfaz de la aplicación presenta una apariencia clara y concisa que satisface las necesidades de los usuarios para registrar sus ingresos y gastos en su vida cotidiana. Para lograr este objetivo, se utiliza el tema AppCompat.Light.NoActionBar.FullScreen, el cual permite ocultar la barra de acciones y proporcionar un modo de pantalla completa, demostrando así más espacio de operación a los usuarios.

En toda la aplicación, se emplea un rango de colores que va desde el blanco hasta el gris claro y oscuro. Estos colores son neutros y contribuyen a que la aplicación sea más equilibrada y cómoda, al mismo tiempo que mejoran la legibilidad y el atractivo visual.

En cuanto al diseño de la estructura y la interfaz de la aplicación, se utilizan botones grandes o menús para ayudar a los usuarios a encontrar las funciones que necesitan. Además, se mantiene una estructura y un diseño consistentes

para asegurarse de que los usuarios puedan comprender y utilizar fácilmente la aplicación. En cada pantalla, se sigue la misma estructura para ubicar la información y las funciones correspondientes, lo cual facilita a los usuarios comprender y utilizar la aplicación de forma rápida.

Por último, el diseño de las interfaces es compatible con una variedad de tamaños de pantalla y tipos de dispositivos Android. De esta manera, se garantiza ofrecer una experiencia de usuario excelente en cualquier dispositivo.

3.4.8.1 Pantalla de login

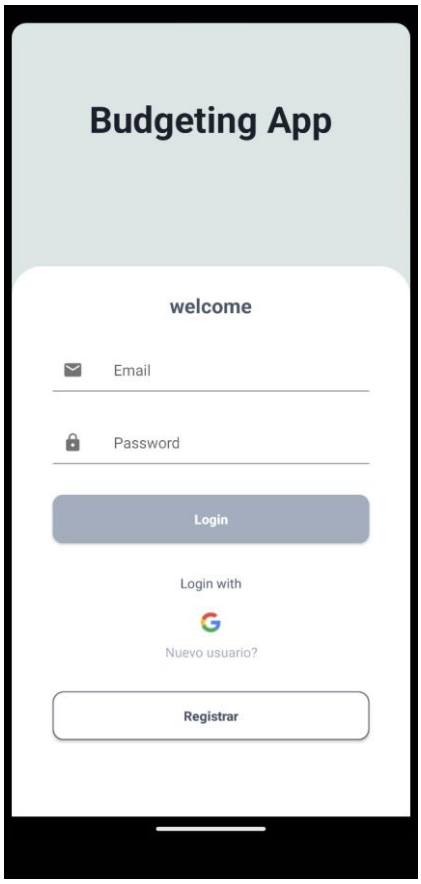


Figura 23 - Diseño de interfaz de Login

Esta pantalla contiene los siguientes elementos:

- Título: El nombre de la aplicación, en un color gris oscuro para destacarlo y llamar la atención del usuario desde el primer momento.

- Input de email: Incluye un icono de correo electrónico en la parte izquierda para indicar al usuario el tipo de información que debe proporcionar en ese campo. Además, cuenta con un texto de ayuda (hint) que aparece en el campo vacío, para que el usuario sepa qué información debe ingresar.

- Input password: Similar al campo de correo electrónico, pero con un icono de bloqueo en lugar del icono de correo electrónico, para indicar

al usuario que debe ingresar su contraseña en ese campo. También cuenta con un texto de ayuda para orientar al usuario.

- Botón de Login: Una vez que el usuario ha ingresado la información necesaria en los campos de correo electrónico y contraseña, puede hacer clic en este botón para iniciar sesión en la aplicación. Es importante que este botón sea lo suficientemente grande y tenga un color que lo haga destacar para que el usuario pueda encontrarlo fácilmente.

- Imagen de Google: Si el usuario prefiere iniciar sesión con su cuenta de Google, puede hacer clic en esta imagen para acceder a esa opción. Esta imagen debe ser clara y fácilmente reconocible para que el usuario pueda identificarla de inmediato.

- Botón de registro: Si el usuario no tiene una cuenta en la aplicación, puede hacer clic en este botón para ir a la pantalla de registro. Este botón también debe ser lo suficientemente grande y tener un color destacado para que el usuario pueda encontrarlo fácilmente.

Además, es importante que el diseño de la pantalla de inicio de sesión sea coherente con el resto del diseño de la aplicación. El uso de colores similares y una disposición clara y ordenada ayudará al usuario a sentirse cómodo y a encontrar la información que necesita de manera rápida y sencilla. También se debe tener en cuenta la usabilidad, asegurándose de que los campos de entrada sean lo suficientemente grandes y estén bien espaciados para que los usuarios puedan ingresar la información sin dificultad.

3.4.8.2 Pantalla de registrar

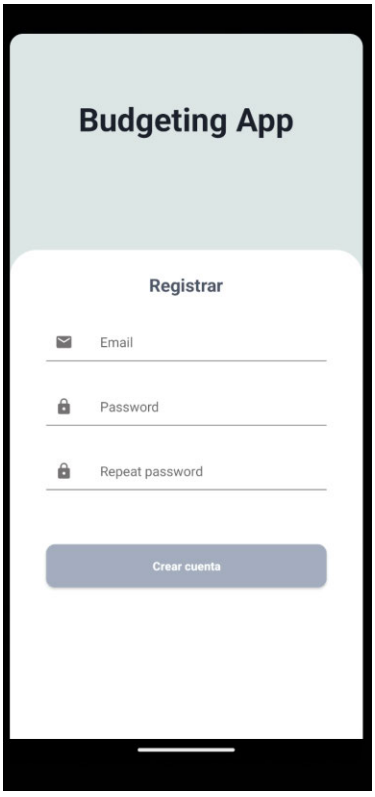


Figura 24 - Diseño de interfaz de registrar cuenta

Esta pantalla de registrar contiene los siguientes elementos:

- Título: El nombre de la aplicación en un color gris oscuro para destacarlo y llamar la atención del usuario desde el primer momento.

- Input de email: En la parte izquierda de este campo se encuentra un icono de correo electrónico, lo que ayuda al usuario a identificar qué tipo de información debe introducir en este campo. Además, cuenta con un texto de ayuda (hint) que aparece en el campo vacío para orientar al usuario sobre qué información debe ingresar.

- Input de password: Este campo es similar al campo de correo electrónico, pero cuenta con un icono de bloqueo en lugar del icono de correo electrónico para indicar al usuario que debe ingresar su contraseña en este campo. También cuenta con un texto de ayuda para orientar al usuario.

- Input repetir password: Para garantizar que la contraseña ingresada sea correcta.

- Botón de registro: Una vez que el usuario ha rellenado los campos necesarios, puede hacer clic en el botón de registro para crear su cuenta. Si los campos se han rellenado correctamente, se redirige al usuario a la pantalla de inicio de sesión.

3.4.8.3 Pantalla principal

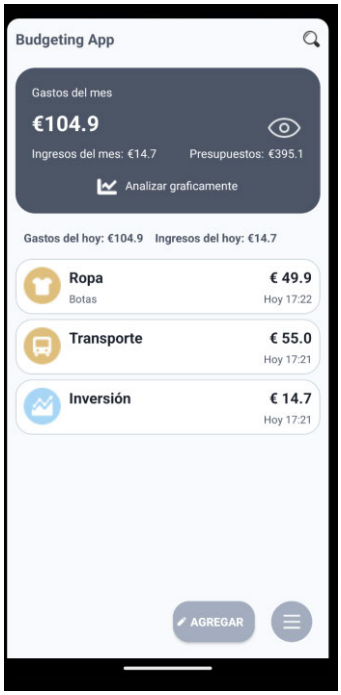


Figura 25 - Diseño de interfaz principal, contiene la lista de los registros

En la pantalla principal contiene los siguientes elementos:

- Icono de búsqueda: Este icono es fundamental para que el usuario pueda buscar registros específicos de manera rápida y sencilla. Al hacer clic en él, se dirige a la pantalla de búsqueda, donde el usuario puede ingresar la descripción correspondiente al registro.

- Información superior: La sección superior de la pantalla es una parte importante de la interfaz de usuario, ya que proporciona información relevante sobre los registros del usuario, como los gastos y los ingresos del mes, así como el presupuesto restante del mes. Si el usuario desea ocultar estas informaciones, puede hacer clic en el icono del ojo. Además, si el usuario desea realizar un análisis más detallado de sus registros, puede hacer clic en el botón "Analizar gráficamente". Se ha utilizado un color gris oscuro para el fondo de esta sección, para resaltarla del resto de la interfaz.

- Lista de registros: Esta sección es donde se muestran los registros de gastos e ingresos del usuario. La lista muestra los totales de gastos e ingresos del día actual en la parte inferior de la pantalla. El usuario puede deslizar la pantalla para navegar por la lista y ver los registros restantes.

- Botón de agregar: El botón de agregar está situado en la parte inferior derecha de la pantalla y permite al usuario agregar nuevos registros de gastos e ingresos con facilidad. Al hacer clic en el botón, se dirige a la pantalla de agregar registros, donde el usuario puede ingresar la información correspondiente.

- Botón de configuración/opción: Este botón proporciona una ventana con varias opciones de configuración y personalización de la aplicación. Al hacer clic en el botón, el usuario puede acceder a las opciones de configuración, como ver historial, borrar todos, cerrar sesión, etc.

También se ha asegurado que el diseño de toda la interfaz de usuario sea coherente con el tema de colores. Utilizando principalmente tonos de gris oscuro y claro para crear un aspecto moderno y elegante.

3.4.8.4 Pantalla de búsqueda

Figura 26 - Diseño de interfaz de búsqueda, resultado en blanco

En la pantalla de búsqueda contiene los siguientes elementos:

- Icono de salida: en la esquina superior izquierda, permite al usuario regresar a la pantalla principal con un solo clic.

- Input de descripción: campo de texto donde el usuario puede ingresar palabras clave para buscar registros específicos. Incluye un icono de búsqueda en la parte derecha y un texto de ayuda (hint) para guiar al usuario sobre qué tipo de información debe ingresar.

- Resultados: Si la búsqueda no devuelve resultados, la aplicación muestra una imagen de "ningún resultado encontrado" para informar al usuario que la búsqueda no ha arrojado ningún registro. Si se encontraron registros, se muestran en una lista debajo del campo de búsqueda.

Además, la pantalla de búsqueda debe mantener la misma paleta de colores que el resto de la aplicación.

3.4.8.5 Ventana de opciones o configuración

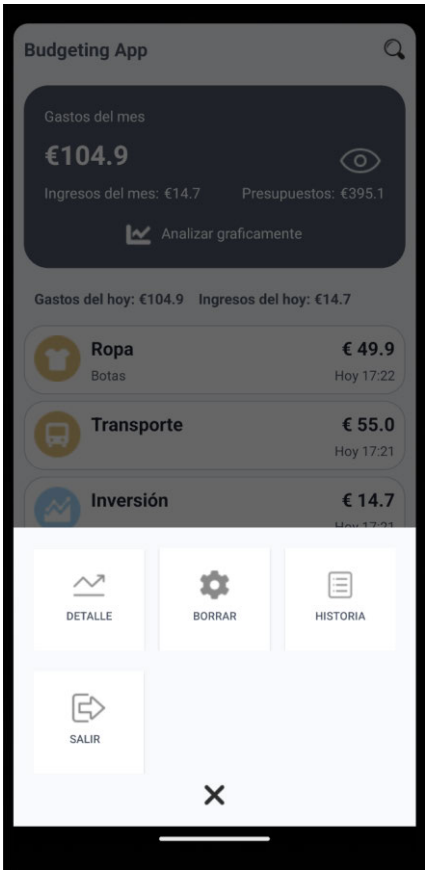


Figura 27 - Diseño de diálogo de configuración de la interfaz principal

En la ventana de opciones puede ser un elemento desplegable situado en la parte inferior de la pantalla, con un icono que indica su presencia. Al hacer clic en el icono, se despliega la ventana de opciones, que ocupa una parte de la pantalla. Cada botón de opción debe ser fácilmente distinguible y tener un icono asociado que lo represente. Además, se puede usar un texto corto debajo del icono para aclarar la función del botón.

- Botón de detalle: permite al usuario acceder a información detallada sobre sus registros.

- Botón de borrar: presenta al usuario un mensaje de confirmación antes de borrar todos los registros.

- Botón de historial: se redirige a la pantalla de historial.

3.4.8.6 Pantalla de agregar

Figura 28 - Diseño de interfaz de agregación de registro de ingreso

La pantalla de agregar registros debe ser lo más fácil de usar posible.

- Icono de salida: en la esquina superior izquierda, permite al usuario regresar a la pantalla principal.

- Tipo de registros: dos títulos para diferenciar el tipo de registros que desea ingresar, por ejemplo, gastos e ingresos. El tipo seleccionado presenta un color y un tamaño de texto diferente para que el usuario pueda identificar fácilmente en qué tipo de registro está agregando.

- Input de cantidad: en la parte izquierda se encuentra el icono y el nombre de la categoría, en la parte derecha se encuentra la cantidad monetaria a introducir.

- Botones de categorías: en esta parte, se muestran todas las categorías disponibles para que el usuario pueda seleccionar la que corresponda a su registro. La categoría seleccionada aparece en color azul, mientras que las no seleccionadas están en gris. Los botones deben ser lo suficientemente grandes para que el usuario pueda seleccionar fácilmente la categoría deseada.

- Campo de descripción: al realizar clic en él, aparecerá una ventana para agregar una descripción correspondiente al registro. La descripción debe ser breve y específica para ayudar al usuario a recordar el propósito del registro.

- Selector de fecha: si el usuario realiza un clic en él, aparecerá una ventana emergente para elegir la fecha del registro.

3.4.8.7 Interfaz de análisis

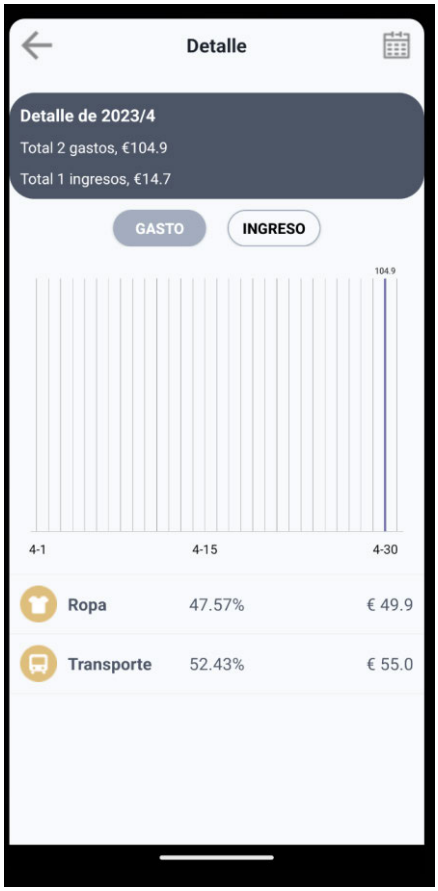


Figura 29 - Diseño de interfaz de análisis de datos de gastos

La pantalla de análisis presenta los siguientes elementos:

- Icono de salida: situado en la esquina superior izquierda, permite al usuario regresar a la pantalla principal.

- Selector de fecha: al hacer clic en él, aparece una ventana que permite al usuario elegir la fecha del mes para analizar.

- Información del mes: en un color gris oscuro para destacar, muestra los gastos e ingresos del mes seleccionado.

- Botones de tipo: dos botones para elegir el tipo de datos a visualizar. El tipo seleccionado se presenta en un color gris claro.

- Diagrama lineal: un diagrama que muestra los días del mes en el eje x y el valor total en el eje y.

- Lista de registros: muestra el valor total de cada categoría agregada, lo que permite al usuario identificar rápidamente las áreas de mayor gasto o ingreso.

4 Desarrollo e implementación

La fase de desarrollo de este proyecto comenzó después de establecer los objetivos, definir los requisitos y el diseño. Como herramienta de desarrollo, se eligió Android Studio, uno de los entornos de desarrollo integrado (IDE) más populares y potentes para la creación de aplicaciones móviles en la plataforma Android.

Durante la fase de desarrollo, se centró en la implementación de las funcionalidades principales, que incluyen el registro y la gestión de ingresos y gastos, la clasificación y las estadísticas de ingresos y gastos, y la gestión de cuentas de usuario y autenticación. Además, se implementó Firebase para agregar capacidades en la nube, como la funcionalidad de inicio de sesión y el almacenamiento en la nube. También se prestó mucha atención a la experiencia del usuario y a la calidad de la aplicación. Se aseguró de que la interfaz de usuario fuera intuitiva, fácil de usar y atractiva. Además, se realizaron pruebas en múltiples dispositivos para garantizar la compatibilidad y la estabilidad.

4.1 Entorno de desarrollo

- **Sistema operativo:** macOS Ventura 13.0
- **IDE:** Android Studio Dolphin | 2021.3.1 Patch
- **Lenguaje de Desarrollo:** JAVA, XML
- **Base de datos:** Firestore

4.2 Dispositivos móviles

- Samsung Galaxy Tab A 8.0 (2019)
  - OS: Android 11
  - CPU: Quad-core 2.0 GHz Cortex-A53
  - Tamaño: 8.0 pulgadas, 800 x 1280 pixels
  - Memoria: 32GB, 2GB RAM
- Emulator: Google pixel 3a
  - OS: Android 13
  - Tamaño: 8.0 pulgadas, 800 x 1280 pixels
  - Memoria: 64GB, 4GB RAM

4.3 Estructura del proyecto

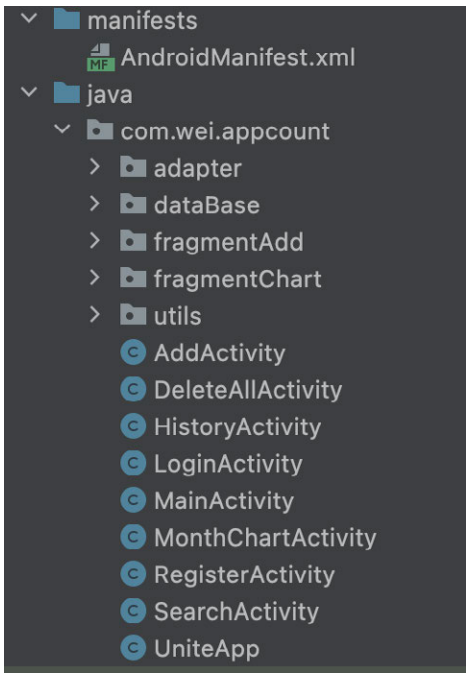


Figura 30 - Estructura de proyecto en Android Studio

En un proyecto de Android, es importante tener una estructura de carpetas bien organizada para el sistema, lo que ayuda a estandarizar la estructura general del sistema. En este proyecto, todas las clases de adapter se colocan en la carpeta "adapter" [17], mientras que la carpeta "dataBase" contiene toda la lógica y las clases de entidad relacionadas con la base de datos. La carpeta "fragmentAdd" contiene fragmentos de diferentes tipos de registros, como fragmento de gasto e ingreso, mientras que "fragmentcChart" incluye todos los fragmentos de análisis de datos. La carpeta "utils" contiene clases de diferentes tipos de dialog, como moreDialog y descriptionDialog, etc. Además, junto con estas carpetas, hay diferentes clases de activity en el mismo nivel, que se utilizan para manejar diversas activity de la aplicación. Con esta estructura, podemos organizar y gestionar mejor los diferentes componentes de la aplicación, y mejorar su mantenibilidad y escalabilidad.

4.4 Integración de Firebase con el proyecto de Android Studio

Antes de poder agregar Firebase a su aplicación de Android, se deben seguir los siguientes pasos: [18]

1. Crear un proyecto en Firebase y configurar la aplicación de Android.
    - Crear un nuevo proyecto en la consola de Firebase.
    - Para registrar la aplicación en el proyecto de Firebase, se debe proporcionar el nombre del paquete de la aplicación.

2. Descargar el archivo "google-services.json" y agregarlo al directorio raíz del módulo (nivel de la app).
    - En el archivo Gradle del nivel raíz (nivel de proyecto), se debe agregar el complemento de servicios de Google como una dependencia de buildscript.

```
buildscript {

  repositories {
   // Make sure that you have the following two repositories
   google()  // Google's Maven repository
   mavenCentral()  // Maven Central repository
  }

  dependencies {
    ...
    // Add the dependency for the Google services Gradle plugin
    classpath 'com.google.gms:google-services:4.3.14'
  }
}

allprojects {
 ...
 repositories {
  // Make sure that you have the following two repositories
  google()  // Google's Maven repository
  mavenCentral()  // Maven Central repository
 }
}
```

    - En el archivo Gradle del módulo (nivel de aplicación),se debe agregar el complemento de servicios de Google:

```
plugins {
  id 'com.android.application'

  // Add the Google services Gradle plugin
  id 'com.google.gms.google-services'
  ...
}
```

2. En el archivo Gradle del módulo (nivel de aplicación), se deben agregar las dependencias para los productos de Firebase que se desean utilizar en la aplicación. Se recomienda utilizar Firebase Android BoM (Bill of Materials) para gestionar el control de versiones de las bibliotecas.

```
dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:31.1.0')

  // When using the BoM, you don't specify versions in Firebase library
dependencies

  // Add the dependency for the Firebase SDK for Google Analytics
  implementation 'com.google.firebase:firebase-analytics'

}
```

- Después de agregar las dependencias, se debe sincronizar el proyecto de Android con los archivos de Gradle. Esto permitirá que las dependencias de Firebase se descarguen y se integren correctamente en el proyecto de Android Studio.

4.5 Login

El inicio de sesión y el registro son funciones básicas en cualquier aplicación, ya que proporcionan una forma para que los usuarios accedan a la aplicación y a su información personal. En este proyecto, usamos la autenticación de Firebase [8] para implementar las funciones de inicio de sesión y registro.

Antes de comenzar a implementar las funciones de inicio de sesión y registro, debemos habilitar el servicio de autenticación （ver Figura 31）. Luego, agregamos las dependencias de autenticación de Firebase a nuestra aplicación.

```
dependencies {
   // Import the BoM for the Firebase platform
   implementation platform('com.google.firebase:firebase-bom:31.1.0')

   // Add the dependency for the Firebase Authentication library
   // When using the BoM, you don't specify versions in Firebase library
dependencies
   implementation 'com.google.firebase:firebase-auth'

   // Also add the dependency for the Google Play services library and specify
its version
   implementation 'com.google.android.gms:play-services-auth:20.4.0'
}
```

Para el registro, se utiliza el proveedor de autenticación de correo electrónico/contraseña proporcionada por Firebase. Esto permite a los usuarios crear nuevas cuentas ingresando su dirección de correo electrónico y contraseña. Además, se puede optar por agregar otros proveedores, como Google, para permitir que los usuarios se registren utilizando sus cuentas existentes.

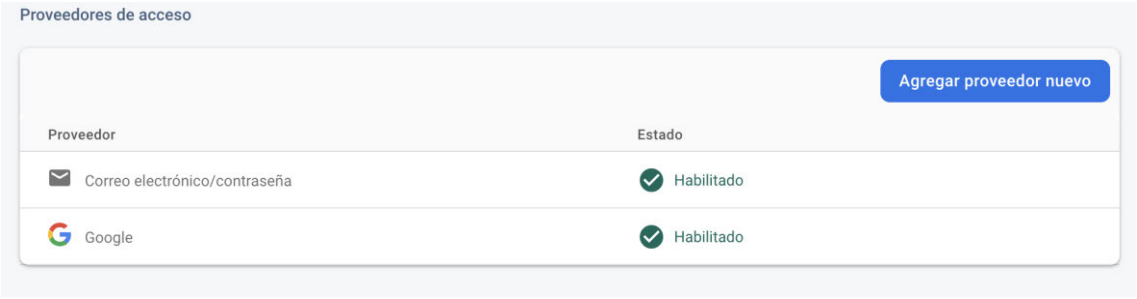


Figura 31 - Proveedores habilitados en Firebase Authentication

4.5.1 Registrar

En los códigos de Java, se obtiene la instancia compartida del objeto "FirebaseAuth" (ver Figura 43) y se crea una nueva cuenta utilizando "createUserWithEmailAndPassword", pasando la dirección de correo electrónico y la contraseña del nuevo usuario. Además, se utiliza un "Toast" para avisar al usuario si el registro de la cuenta se ha realizado correctamente y vuelve a la interfaz de Login, por otra parte, en la consola de Firebase aparecerá el usuario registrado.

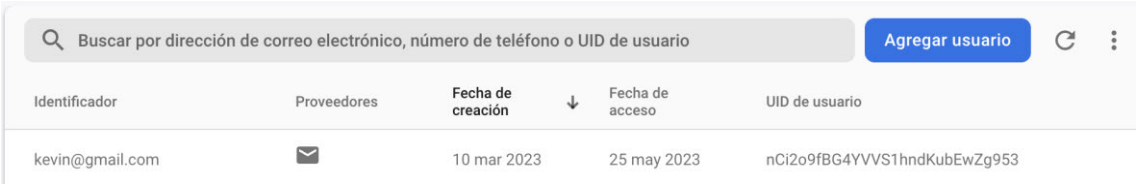


Figura 32 - Cuando un usuario registrado correctamente aparecerá en la consola de Firebase

4.5.2 Autenticación con email

Primero se debe obtener la instancia compartida del objeto "FirebaseAuth". Luego se debe recopilar la dirección de correo electrónico y la contraseña del usuario en la interfaz de inicio de sesión y pasarlos a "signInWithEmailAndPassword"Posteriormente se agrega un listener para la tarea "signInWithEmailAndPassword" con el fin de verificar si el inicio de sesión se realizó correctamente o no. Si el inicio de sesión es exitoso, se redirige al usuario a la pantalla principal de la aplicación utilizando un objeto "Intent". En caso de que falle, se muestra un mensaje de error utilizando "Toast" (ver Figura 44).

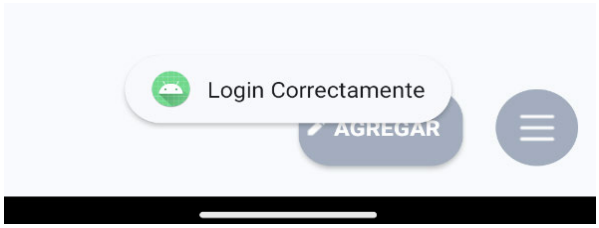


Figura 33 - Notificación de la operación Login(exitosa)

4.5.3 Google sign-in

En esta parte se va a realizar login con la cuenta de Google, simplificando el proceso de autenticación y garantizando la seguridad de la cuenta, para ello hay que seguir estos pasos:

1. Para configurar la autenticación de Google en la aplicación, se debe crear una constante para la opción “DEFAULT_SIGN_IN”. Luego, se debe utilizar un objeto “GoogleSignInOptions.Builder” y pasar la opción creada, junto con el ID de cliente del tipo aplicación web utilizando “requestIdToken”, para obtener la identificación del usuario. Además, para obtener el correo electrónico del usuario, se debe agregar la opción “requestEmail()” (ver Figura 45).

   Una vez que se tenga la configuración, se puede crear un cliente de autenticación mediante el método “GoogleSignIn.getClient” y pasar la configuración. Luego, para mostrar la pantalla de autenticación, se debe llamar al método “startActivityForResult” y pasar los parámetros “googleClient.SignInIntent” y un código.

2. En el método “onActivityResult()”, se obtienen las credenciales de inicio de sesión de Google del usuario y luego se las pasa a Firebase para completar el proceso de autenticación e inicio de sesión.

   En el código anterior, utilizamos el método “addOnCompleteListener()” para agregar un callback que se ejecutará después de que el usuario haya completado el inicio de sesión. Dentro del callback, primero verificamos si la tarea de inicio de sesión se ha completado con éxito. Si es así, mostramos un “Toast” y redirigimos al usuario a la pantalla principal (MainActivity). Si la tarea falla, mostramos otro “Toast” que pide al usuario que intente iniciar sesión de nuevo (ver Figura 46).

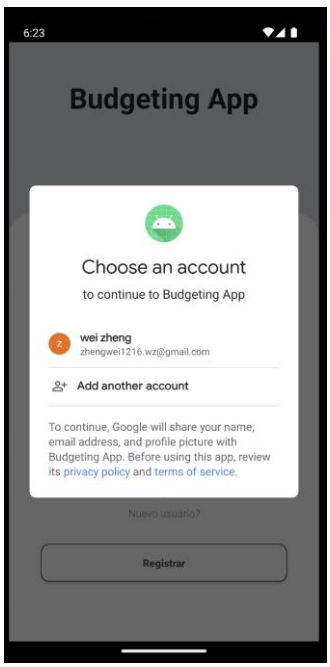


Figura 34 - Demostración de Login con la cuenta del google

4.5.4 SignOut

Si el usuario desea cerrar sesión en la aplicación, puede hacer clic en el botón de configuración de la interfaz principal. A continuación, se debe buscar y seleccionar la opción "Salir". Al seleccionar esta opción, se activará el proceso de cierre de sesión. Para ello, se obtendrá la instancia actual del usuario y se llamará al método "signOut" (ver Figura 50) proporcionado por la plataforma de autenticación Firebase Auth. Este método se encargará de finalizar la sesión actual del usuario. Después de cerrar sesión correctamente, la aplicación redireccionará automáticamente a la interfaz de inicio de sesión, donde el usuario podrá ingresar nuevamente con sus credenciales o seleccionar otras opciones disponibles.

4.6 Funcionalidad de la interfaz principal

Para implementar las funciones de la interfaz principal (ver Figura 25), la aplicación necesita interactuar con Firestore para almacenar y recuperar datos. Para mostrar los ingresos y gastos totales del usuario para el mes actual, la aplicación necesita consultar los datos relacionados con el mes actual almacenados en Firestore y realizar el cálculo final. Además, es necesario mostrar los nuevos registros agregados cuya fecha sea la de hoy.

4.6.1 Gastos e ingresos del mes

Para obtener los gastos totales de este mes en Firestore, se ha utilizado la función de consulta de Firestore y se han definido las condiciones adecuadas. Primero, se debe acceder a la colección "gastos" y luego se puede aplicar la función "whereEqualTo" [19] para filtrar los resultados (ver Figura 47Figura 47). Es necesario definir tres condiciones:

1. "Email": el correo electrónico del usuario actual.
2. "Year": el año actual.
3. "Month": el mes actual.

Al aplicar estas condiciones, se obtendrán todos los registros de gastos correspondientes al mes actual del usuario actual. Para obtener los ingresos totales, se puede seguir el mismo proceso, pero cambiando la colección a "ingresos".

Una vez que se obtienen los resultados de la consulta, se pueden procesar y sumar los valores de todos los registros para obtener el total de gastos o ingresos del mes actual. Esta información se muestra en la interfaz principal.

4.6.2 Presupuesto

En la interfaz principal, se mostrará el presupuesto mensual del usuario. Si el usuario no ha establecido un valor de presupuesto previo, se establecerá automáticamente un valor 0. Si el usuario desea establecer un presupuesto, puede hacer clic en el valor del presupuesto y se mostrará un cuadro de diálogo debajo de la interfaz principal para ingresar la cantidad del presupuesto mensual. Finalmente, el valor del presupuesto se guardará en la colección "datos" de Firestore (ver Figura 48).

Figura 35 – Dialogo para configurar presupuesto de cada mes

Por otra parte, se calculará automáticamente el presupuesto mensual. Primero, se obtendrá la cantidad del presupuesto mensual de la colección "datos" y se restará el total de gastos del mes, luego se mostrará en la interfaz principal. Cuando el usuario agregue nuevos registros de gasto, se recalculará el presupuesto del mes. Del mismo modo, cuando el usuario vuelva a establecer el valor del presupuesto mensual, se recalcula y se muestra en la interfaz principal.

4.6.3 Lista de registro

En la interfaz principal, hay una lista (Listview) que mostrará todos los registros de gastos e ingresos del día actual. Para obtener estos datos, necesitamos agregar cuatro condiciones en las colecciones "gastos" e "ingresos":

1. "Email": el correo electrónico del usuario actual.
2. "Year": el año actual.
3. "Month": el mes actual.
4. “Day”: el día actual.

Aplicando estas condiciones en la consulta, podemos obtener todos los registros del hoy (ver Figura 49).

Además, se ha agregado una función de eliminación para mejorar la interactividad del usuario. Cuando el usuario mantiene presionado un registro en la lista, aparecerá un diálogo que le pregunta si desea confirmar la eliminación de ese registro. Si el usuario confirma la eliminación, el sistema utilizará el parámetro "time" del registro para obtener el "ID" correspondiente en las colecciones "gastos" e "ingresos" y lo eliminará utilizando el método "remove(id)" (ver Figura 51). Después de eliminar el registro, se actualizará la lista para reflejar los cambios realizados. Esta función permite a los usuarios eliminar de manera conveniente y rápida los registros que deseen eliminar de su lista.

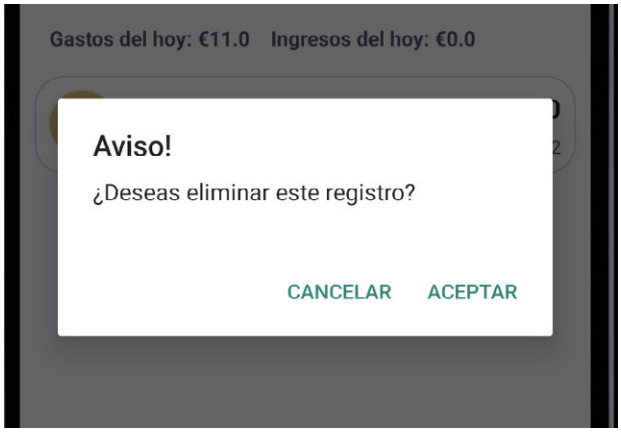


Figura 36 - Dialogo de eliminación de registro de la interfaz principal

De esta manera, los usuarios pueden ver fácilmente los registros de gastos e ingresos del día actual y administrar sus datos mediante la función de eliminación al mantener presionado sobre un registro.

4.7 Buscar registro

En la interfaz de búsqueda, se presenta un campo de entrada con una hint para que el usuario ingrese la descripción de los registros que desea buscar. Luego, al hacer clic en el icono de búsqueda a la derecha, se realiza la búsqueda de acuerdo con los siguientes pasos:

1. Validación del campo de entrada: Si el usuario deja el campo de descripción vacío, se muestra un mensaje (usando Toast) para indicarle que debe ingresar contenido en el campo, ya que no se permite una búsqueda vacía.

2. Obtención de registros: Si el campo de descripción no está vacío, se realiza una consulta en las colecciones "gastos" e "ingresos" para obtener todos los registros.
3. Comparación de registros: Para cada registro obtenido, se verifica si la descripción contiene la información ingresada por el usuario. Si la descripción coincide, el registro se agrega a la lista de resultados (ver Figura 52).
4. Actualización de la interfaz: Finalmente, se actualiza la interfaz para mostrar los registros encontrados en la lista, proporcionando así una visualización de los registros coincidentes con la descripción buscada.

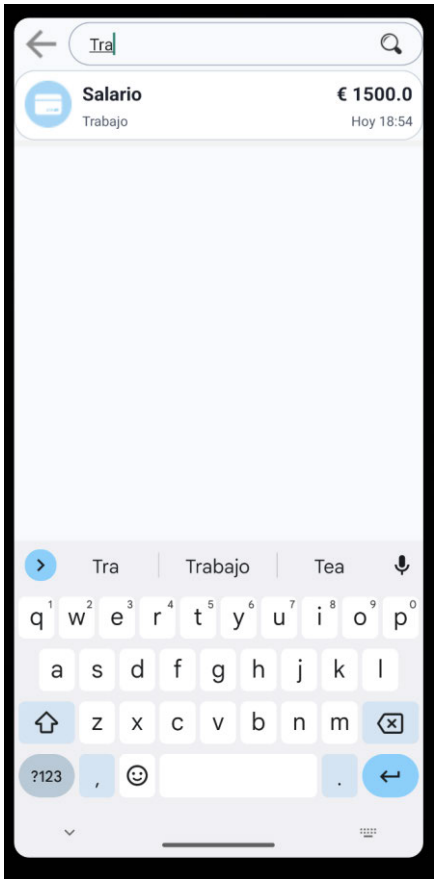


Figura 37 - resultado de una búsqueda

4.8 Agregar registros

La implementación de la funcionalidad de agregar registros ha enfocado en la facilidad de uso y comprensión para los usuarios. A través de una interfaz intuitiva, campos de descripción y selección de fechas, y el almacenamiento adecuado en las colecciones correspondientes.

4.8.1 Añadir descripción

La funcionalidad para agregar registros de gastos e ingresos se centra en la usabilidad del usuario al permitir la descripción de registro correspondientes.

La capacidad de agregar descripciones ayuda a los usuarios una mejor comprensión de los detalles de cada registro y facilita la búsqueda de registros específicos en la interfaz de búsqueda.

Al hacer clic en la opción de agregar descripción, se despliega un diálogo que permite a los usuarios ingresar la descripción del registro. Este diálogo se presenta de forma clara y concisa, con un campo de texto para ingresar la descripción y opciones para guardar o cancelar la operación.

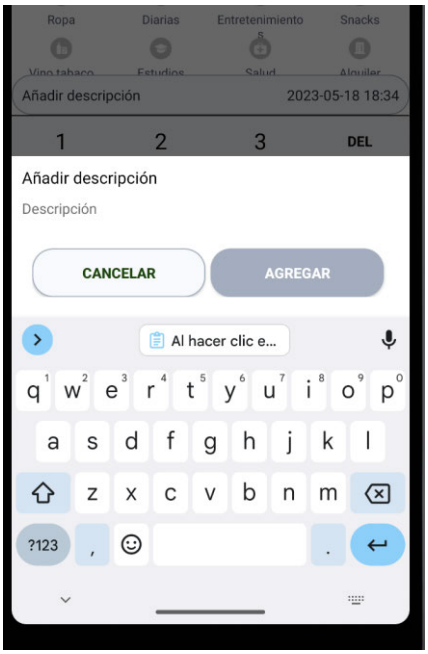


Figura 38 - Dialogo de agregación de descripción para registro

Una vez que el usuario ha ingresado la descripción en el diálogo, esta se guarda en un atributo de registro (ver Figura 53).

4.8.2 Añadir fecha

La funcionalidad de seleccionar y guardar la fecha en los registros de gastos e ingresos ofrece a los usuarios mayor flexibilidad y precisión en la gestión de sus registros. Al hacer clic en la opción de seleccionar fecha, se muestra en la pantalla un selector de fecha que permite a los usuarios elegir el año, mes, día, hora y minuto correspondientes al registro. Si los usuarios que no deseen seleccionar la fecha, se ha implementado la opción de agregar automáticamente la fecha actual al momento de agregar el registro. Esto garantiza que cada registro tenga una fecha asociada y evita la necesidad de seleccionarla manualmente.

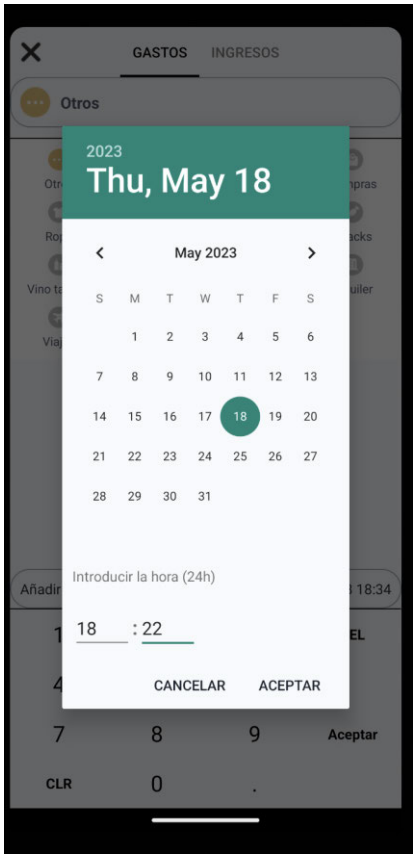


Figura 39 - Selector de fecha en la interfaz de agregación de registro

Una vez que el usuario ha seleccionado la fecha utilizando el selector o se ha añadido automáticamente la fecha actual, esta se guarda en los atributos del registro correspondiente. Luego, la fecha seleccionada se mostrará en el campo "date" de la interfaz de agregar registro.

4.8.3 Agregar registro en Firestore

Al hacer clic en el botón "ACCEPT", se activa la función de guardar el registro en la colección correspondiente según el fragmento actual de la aplicación. Esta función recopila la información ingresada por el usuario, como la descripción, la fecha y el valor, y la almacena en Firestore. Después de que se haya realizado la operación de guardar, se lleva a cabo una verificación para asegurarse de que la agregación de datos se haya realizado de manera exitosa. Esto implica comprobar si no ha habido errores ni problemas de conexión durante la transacción con Firestore.

Una vez completado el proceso de guardar, se utiliza un toast para informar al usuario sobre el resultado de la operación. Si el resultado ha sido exitoso, se muestra un mensaje de confirmación indicando que el registro se ha guardado correctamente. Luego se redirige al usuario de vuelta a la pantalla principal de la aplicación. Esto permite visualizar los datos nuevos como gastos e ingresos totales (ver Figura 54).

4.9 Registro de historial

En la pantalla de historial de registros, los usuarios podrán visualizar todos los registros que han sido agregadas previamente. Al abrir esta pantalla, se llevarán a cabo una serie de pasos para garantizar la correcta obtención y visualización de los datos históricos. A continuación, se detalla el proceso de esta funcionalidad, resaltando la obtención de la fecha actual, el filtrado de datos y la presentación en la interfaz:

1. Obtención de la fecha actual: Al acceder a la pantalla de historial, se obtiene la fecha actual por defecto, el año y el mes en curso. Esto se realiza automáticamente al cargar la pantalla y permite filtrar los datos de acuerdo con el año y mes correspondientes.
2. Acceso a la colección de gastos e ingresos: Una vez obtenida la fecha actual, se accede a las colecciones de gastos e ingresos en Firestore. Utilizando el año y el mes actual, se realiza un filtrado de los datos almacenados en Firestore. Se seleccionan únicamente aquellos registros que corresponden al año y mes obtenidos anteriormente. Esto permite presentar al usuario solo los registros relevantes para el período en el que se encuentra.
3. Visualización en la interfaz: Una vez filtrados los datos, se procede a mostrarlos en la interfaz de la pantalla de historial. Esto puede lograrse mediante la creación de una lista o tabla que presente de manera clara y organizada los registros, mostrando información como la descripción, la fecha y cualquier otro dato relevante asociado a cada entrada.

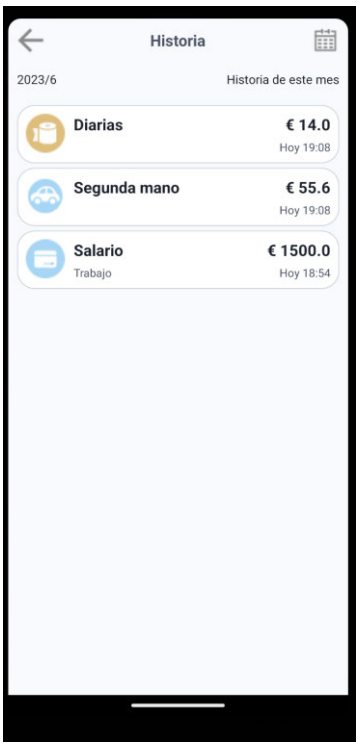


Figura 40 - Resultado de la interfaz historial con registros

En la pantalla de historial de registros, se ha implementado una funcionalidad adicional que permite a los usuarios seleccionar el año y mes específico que desean visualizar. Para lograr esto, se ha incorporado un icono de calendario que, al hacer clic, despliega un de diálogo i donde los usuarios pueden elegir el año y el mes.

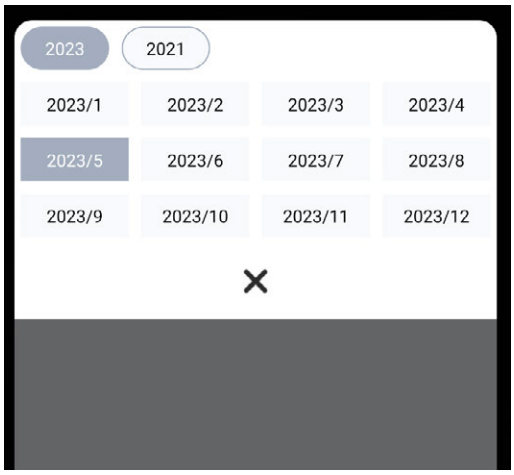


Figura 41 - Dialogo de elegir mes

Cuando se activa el diálogo, se realiza un proceso que garantiza la disponibilidad de los años para seleccionar. Primero, se recopilan todos los registros agregados anteriormente y se extrae el año de cada una de ellas. A continuación, estos años se presentan al usuario en la interfaz del diálogo para que puedan realizar su elección. Solamente se muestra los años asociados a cada registro.

Al permitirles seleccionar el año y mes deseados, se les proporciona un mayor control sobre la visualización de los datos históricos y una forma más eficiente de acceder a la información relevante para sus necesidades.

4.10 Analizar registros

En la pantalla de análisis de datos permite a los usuarios realizar un análisis detallado de sus registros financieros (ver Figura 29). Para lograr esto, se llevan a cabo una serie de pasos para obtener y presentar la información relevante de manera clara y concisa.

En primer lugar, se realiza una consulta a Firestore para obtener los datos de los registros correspondientes al mes actual. Esto implica acceder a la colección de gastos o ingresos, y aplicar condiciones de búsqueda para obtener únicamente los registros del mes en curso. A partir de estos registros, se calcula y obtiene el total de los gastos y los ingresos para el mes actual. Estos valores se presentan en la pantalla para que los usuarios puedan visualizar de manera rápida.

Además de los totales mensuales, también se realiza un análisis más detallado a nivel diario. Se obtiene la suma de los ingresos y los gastos para cada día del mes actual.

Estos valores se utilizan para generar un diagrama de barra, que ilustran la distribución de los ingresos y gastos a lo largo del mes. Para implementar esta funcionalidad, se utiliza una biblioteca de terceros llamada MPAndroidChart, que proporciona las herramientas necesarias para la visualización gráfica de datos.

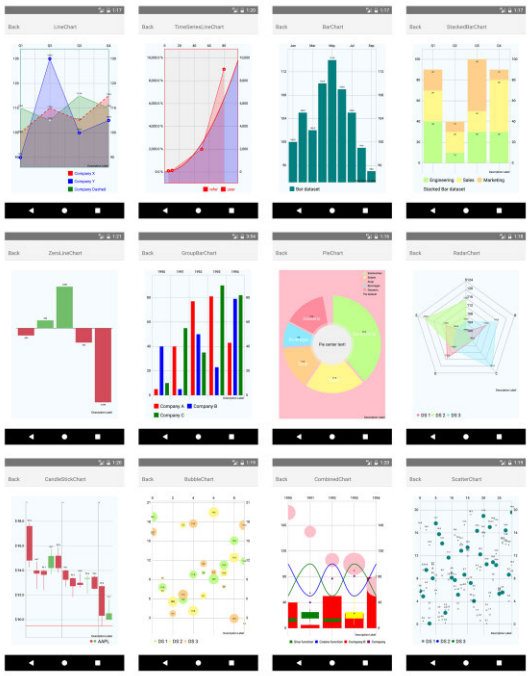


Figura 42 - Diferentes graficas proporcionadas de la librería MPAndroidchart

Además, se ha incorporado un icono de calendario en la interfaz como la interfaz de historial de registro, lo que permite a los usuarios seleccionar el año y el mes que desean analizar.

5 Conclusiones

En este trabajo de fin de máster, se ha desarrollado una aplicación de gestión financiera que permite a los usuarios llevar un seguimiento y análisis de sus registros financieros. Mediante la implementación de diversas funcionalidades, como la agregación de gastos e ingresos, visualización de historiales, búsqueda de registros y análisis de datos, se ha logrado crear una aplicación efectiva para la administración de las finanzas personales. El acceso a Firestore ha permitido una gestión eficiente de los datos financieros, permitiendo consultas y filtros precisos para obtener información relevante. Además, se ha utilizado la biblioteca MPAndroidChart para la visualización gráfica de datos, brindando a los usuarios una representación visual clara y concisa de sus registros financieros.

Por otro lado, el desarrollo de la aplicación me ha permitido mejorar mis habilidades de programación en Android. En este proceso, me enfrenté a pequeños problemas y detalles debido a la falta de experiencia en programación Android y conocimiento incompleto en algunos aspectos, lo que resultó en errores de uso. Gran parte del tiempo de desarrollo se dedicó a resolver estos errores, algunos de los cuales no eran visibles durante el análisis inicial, sino que surgían una vez que comencé a programar y me enfrenté a diversas dificultades que tuve que resolver paso a paso.

Estos desafíos y obstáculos han sido valiosas experiencias y conocimientos. He aprendido a revisar detalladamente el código en busca de posibles errores y a utilizar herramientas de depuración para localizar problemas y encontrar soluciones. A través de este proyecto, he logrado superar los desafíos y completar exitosamente el desarrollo de la aplicación. Esta experiencia no solo ha ampliado mi comprensión de la programación en Android, sino que también me ha enseñado la importancia de aprender de forma continua y practicar de manera constante. Estoy acuerdo de que estas habilidades y conocimientos adquiridos serán de gran utilidad en mi crecimiento personal y profesional.

6 Trabajo futuro

A continuación, muestra varias mejoras y adiciones que podrían considerarse para el futuro desarrollo de la aplicación. Estos trabajos se centran en ampliar la disponibilidad de la aplicación, mejorar las opciones de registro e inicio de sesión, mejorar la experiencia visual del usuario y servicios de pagos. A continuación, se detallan algunas de estas posibles mejoras:

1. Ampliar la disponibilidad de la aplicación: Dado que Firebase es una plataforma multiplataforma, sería beneficioso considerar la posibilidad de desarrollar versiones de la aplicación para iOS y también una versión web. Esto permitiría a los usuarios acceder y gestionar sus registros de ingresos y gastos desde diferentes dispositivos y plataformas, lo que aumentaría la popularidad y la usabilidad de la aplicación.
2. Mejorar las opciones de inicio de sesión y registro: Además de la opción de inicio de sesión con correo electrónico y contraseña, se podría implementar la opción de inicio de sesión con el teléfono móvil sin registrar proporcionando una manera más rápida. brindar a los usuarios una alternativa más conveniente y rápida. También se podría considerar la inclusión del login mediante Facebook.
3. Mejorar la interfaz de la aplicación: Para mejorar la experiencia del usuario, se podría trabajar en la mejora de la interfaz de usuario y el efecto visual de la aplicación. Esto incluiría la revisión del diseño de la interfaz para hacerlo más intuitivo y fácil de navegar, así como la incorporación de elementos visuales atractivos, como iconos, colores y animaciones, para hacer que la aplicación sea más agradable a la vista y emocionalmente atractiva.
4. Recordatorios y notificaciones: Permitir a los usuarios configurar recordatorios y recibir notificaciones para realizar un seguimiento de sus pagos pendientes, fechas de vencimiento de facturas o alcanzar sus metas de ahorro. Esto les ayudaría a mantenerse al tanto de sus compromisos financieros y mantener un control adecuado de sus finanzas.
5. Integración con servicios de pago: Permitir a los usuarios vincular sus cuentas bancarias o servicios de pago para importar automáticamente sus transacciones y facilitar el seguimiento de sus registros financieros.

7 Análisis de Impacto

En septiembre de 2015 se establecieron una serie de objetivos globales, una oportunidad para que los países y sus sociedades emprendan un nuevo camino para mejorar la vida de todos. Con metas específicas a conseguir en 15 años, para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos.

El desarrollo de esta aplicación tiene un impacto significativo en varios aspectos, incluyendo el ámbito personal, empresarial, social, económico y medioambiental. A continuación, se presenta un análisis detallado de cada uno de estos impactos:

- **Impacto Personal:** Esta aplicación tiene un impacto positivo en el ámbito personal, ya que proporciona a los usuarios una plataforma para administrar sus finanzas de manera más eficiente. Al presentar una interfaz intuitiva y funciones útiles, los usuarios pueden llevar un registro detallado de sus ingresos y gastos, ayuda a tomar decisiones financieras más informadas, mejorar sus hábitos de gasto y lograr una mayor estabilidad financiera en sus vidas personales.

- **Impacto Empresarial:** Desde una perspectiva empresarial, esta aplicación puede ser beneficiosa para las pequeñas y medianas empresas que desean llevar un control preciso de sus finanzas. Al utilizar esta aplicación, los empresarios pueden realizar un seguimiento de sus ingresos, gastos y pagos pendientes de manera más eficiente, lo que puede conducir a una mayor rentabilidad y estabilidad financiera para el negocio.

- **Impacto Social:** En el ámbito social, esta aplicación contribuye a fomentar una mayor conciencia financiera entre los usuarios. Esta aplicación mejora la educación financiera y se ayuda a los individuos a adquirir habilidades para administrar sus recursos de manera más efectiva. Esto puede tener un impacto positivo en la sociedad en general, ya que la estabilidad financiera individual se traduce en una mayor estabilidad y bienestar social.

- **Impacto Económico:** Al ayudar a los individuos y las empresas a administrar sus finanzas de manera más eficiente, se fomenta el ahorro, la inversión y el gasto responsable. Esto puede tener un efecto multiplicador en la economía, impulsando el crecimiento económico y generando oportunidades de empleo.

- **Impacto Medioambiental:** En cuanto al impacto medioambiental, esta aplicación, al facilitar la gestión financiera electrónica, puede contribuir a la reducción del uso de papel y la generación de desechos relacionados con la gestión de registros financieros en formato físico, se puede lograr una mayor eficiencia en la gestión de recursos y una reducción de materiales asociada a la administración financiera.

8 Bibliografía

[1] «Android,» [En línea]. Available: https://www.android.com/. [Último acceso: 06 2023].

[2] «Cuota de mercado de Android,» [En línea]. Available: https://root-nation.com/es/noticias-es/es-cuota-mercado-android-ios-estadisticas-publicadas-2022/. [Último acceso: 06 2023].

[3] «Android studio,» [En línea]. Available: https://developer.android.com/studio. [Último acceso: 2023].

[4] «Firebase,» Google, [En línea]. Available: https://firebase.google.com/?hl=es-419. [Último acceso: 06 2023].

[5] «IDEA,» Jetbrains, [En línea]. Available: https://www.jetbrains.com/es-es/idea/.

[6] «JAVA,» Oracle, [En línea]. Available: https://www.java.com/es/.

[7] «Programación orientado a objeto,» Profile, [En línea]. Available: https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/.

[8] «Firebase Authentication,» Google, [En línea]. Available: https://firebase.google.com/docs/auth?hl=es-419. [Último acceso: 2023].

[9] «Cloud Firestore,» Google, [En línea]. Available: https://firebase.google.com/docs/firestore?hl=es-419. [Último acceso: 04 2023].

[10] P. Jay, «MPAndroidChart,» [En línea]. Available: https://github.com/PhilJay/MPAndroidChart. [Último acceso: 04 2023].

[11] «XML W3,» [En línea]. Available: https://www.w3.org/standards/xml/core.

[12] «Ciclo de vida de activity,» Google, [En línea]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es-419. [Último acceso: 2023].

[13] «SQLite,» [En línea]. Available: https://sqlite.org/index.html.

[14] «Buddy,» [En línea]. Available: https://buddy.download/. [Último acceso: 2023].

[15] «MOnefy,» [En línea]. Available: https://monefy.me/. [Último acceso: 2023].

[16] «Money flow,» [En línea]. Available: https://www.moneyflow.app/. [Último acceso: 2023].

[17] «Android Adapter,» [En línea]. Available: https://developer.android.com/reference/android/widget/Adapter. [Último acceso: 2023].

[18] «Agregar Firebase a una app Android,» [En línea]. Available: https://firebase.google.com/docs/android/setup?hl=es-419. [Último acceso: 02 2023].

[19] «consulta de query a Firestore,» [En línea]. Available: https://firebase.google.com/docs/firestore/query-data/queries?hl=es-419. [Último acceso: 04 2023].

9 Anexo

Función para crear una cuenta con el correo electrónico, utilizando Toast para notificar al usuario sobre el resultado de la operación.

```
FirebaseAuth.getInstance().createUserWithEmailAndPassword(email.getText().toString(),
        password.getText().toString()).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()){
            Toast.makeText( context: RegistrarActivity.this, text: "Usuario registrado Correctamente",Toast.LENGTH_LONG).sho
            onBackPressed();
        }else {
            Toast.makeText( context: RegistrarActivity.this, text: " Registrar error!",Toast.LENGTH_LONG).show();
        }
    }
```

Figura 43 - Función para registrar una cuenta con el correo electrónico

Autenticación de Firebase con el correo electrónico. Si la operación es exitosa, se redirigirá al usuario a la interfaz principal, y se utilizará un Toast para notificar al usuario sobre el resultado.

```
FirebaseAuth.getInstance().signInWithEmailAndPassword(edit_email.getText().toString(),
        edit_password.getText().toString()).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
            Toast.makeText( context: LoginActivity.this,  text: "Login Correctamente", Toast.LENGTH_LONG).show();
            Intent intent = new Intent( packageContext: LoginActivity.this, MainActivity.class);
            intent.putExtra( name: "email", task.getResult().getUser().getEmail());
            startActivity(intent);
        } else {
            Toast.makeText( context: LoginActivity.this,  text: "Login Error!", Toast.LENGTH_LONG).show();
        }
    }
});
```

Figura 44 - Instancia de Firebase para realizar Login con el email

Autenticación de Firebase con la cuenta de Google utilizando “DEFAULT_SIGN_IN”

```
case R.id.im_login_google:
    GoogleSignInOptions googleConf = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken(getString(R.string.default_web_client_id))
            .requestEmail()
            .build();

    GoogleSignInClient googleSignInClient = GoogleSignIn.getClient( activity: LoginActivity.this, googleConf);
    googleSignInClient.signOut();
    startActivityForResult(googleSignInClient.getSignInIntent(), GOOGLE_CODE);
    break;
```

Figura 45 - Función de Login con Google con el signInOptions

En la función onActivityResult, se realiza una verificación del código. Si el código coincide, se realiza la autenticación con signInWithCredential de Google. Si la operación es exitosa, se redirige al usuario a la interfaz principal y se utiliza un Toast para notificar el resultado al usuario.

```
if (requestCode == GOOGLE_CODE) {
    Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
    try {
        GoogleSignInAccount account = task.getResult(ApiException.class);
        AuthCredential credential = GoogleAuthProvider.getCredential(account.getIdToken(),  accessToken: null);
        if (account != null) {
            FirebaseAuth.getInstance().signInWithCredential(credential).addOnCompleteListener(new OnCompleteListener<AuthResult
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        Toast.makeText( context: LoginActivity.this,  text: "Login with google sucess", Toast.LENGTH_LONG).show();
                        Intent intent = new Intent( packageContext: LoginActivity.this, MainActivity.class);
                        intent.putExtra( name: "email", task.getResult().getUser().getEmail());
                        startActivity(intent);
                    } else {
                        Toast.makeText( context: LoginActivity.this,  text: "Login with gogle Fail", Toast.LENGTH_LONG).show();
                    }
                }
```

Figura 46 - Comprobación de Google Login en onActivityResult

Realiza una consulta a Firestore para obtener los gastos totales del mes y luego actualiza el presupuesto restante del mes.

```
db.collection( collectionPath: "gastos").whereEqualTo( field: "email", email).whereEqualTo( field: "year", year)
        .whereEqualTo( field: "month", month)
        .get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    Float totalGastosDelMes = 0.0f;
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        Registro registro = document.toObject(Registro.class);
                        totalGastosDelMes = totalGastosDelMes + registro.getMoney();
                    }
                    topOutTv.setText("€" + totalGastosDelMes);
                    //configurar calculo de presupuesto restante
                    setPresupuesto(totalGastosDelMes);
```

Figura 47 - Función para obtener gastos totales del mes actual

El valor de presupuesto se guarda en la colección de datos, luego se actualizará el valor restante.

```
db.collection( collectionPath: "datos").document(email).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if (task.isSuccessful()) {
            DocumentSnapshot document = task.getResult();
            if (document.exists()) {
                float budgetMoney = Float.valueOf(document.getData().get("presupuesto").toString());//valor del presupuesto
                float restanteMoney = budgetMoney - totalGastosDelMes;//presupuesto restantes
                topBudgetTv.setText("€" + restanteMoney);//mostrar en la interfaz principal
            } else {
                topBudgetTv.setText("€0");//valor presupuesto por defecto
            }
```

Figura 48 - Función de configurar presupuesto en la colección datos.

Realiza una consulta a Firestore para obtener los gastos del hoy, y los añaden en la lista.

```
//Cargar los registros del hoy
private void loadLVData(int year, int month, int day) {
    List<Registro> list = new ArrayList<>();
    db.collection( collectionPath: "gastos").whereEqualTo( field: "email", email).whereEqualTo( field: "year", year)
            .whereEqualTo( field: "month", month)
            .whereEqualTo( field: "day", day)
            .get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        for (QueryDocumentSnapshot document : task.getResult()) {
                            Registro registro = document.toObject(Registro.class);
                            list.add(registro);//añadir en la lista
                        }
```

Figura 49 - Función de obtener los registros del día actual

Con la instancia de autenticación, se puede llamar a "signOut" para cerrar la sesión.

```
case R.id.more_logout_btn:
    FirebaseAuth.getInstance().signOut();
    intent.setClass(getContext(), LoginActivity.class);
    getContext().startActivity(intent);
    break;
```

Figura 50 - Salir sesión utilizando signOut

Utilizando el id del registro para eliminar el documento de la colección gastos/ingresos

```
AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
db.collection( collectionPath: "gastos").whereEqualTo( field: "time", registro.getTime()).get()
        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        String id = document.getId();
                        db.collection( collectionPath: "gastos").document(id).delete();
                        builder.setTitle("Aviso!").setMessage("¿Deseas eliminar este registro?")
                                .setNegativeButton( text: "cancelar",  listener: null).setPositiveButton( text: "aceptar",
                                    @Override
                                    public void onClick(DialogInterface dialogInterface, int i) {
                                        db.collection( collectionPath: "gastos").document(id).delete();
                                        mDatas.remove(registro);
                                        adapter.notifyDataSetChanged();
```

Figura 51 -Función de eliminación de registro utilizando id de documento

Realiza una consulta a Firestore para comparar la descripción de cada registro, luego se guardan en la lista de resultado

```
db.collection( collectionPath: "gastos").whereEqualTo( field: "email", email)
        .get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
        if (task.isSuccessful()) {
            for (QueryDocumentSnapshot document : task.getResult()) {
                Registro registro = document.toObject(Registro.class);//obtener registro
                //si el registro contiene la descripcion a buscar
                if(registro.getDescription()!=null&&registro.getDescription().contains(msg)){
                    resList.add(registro);
                }
            }
            mDatas.clear();//borrar la lista anterior
            mDatas.addAll(resList);//añadir en la lista
            adapter.notifyDataSetChanged();//actualizar la lista
        }
    }
```

Figura 52 - Función de búsqueda de registro mediante descripción.

Convierte la entrada de texto en descripción en caso no vació.

```
descriptionDialog.setOnAceptarListener(new DescriptionDialog.OnAceptarListener() {
    @Override
    public void OnAceptar() {
        String msg=descriptionDialog.getEditText();
        if(!msg.isEmpty()){
            descTv.setText(msg);
            registro.setDescription(msg);
        }
        descriptionDialog.cancel();
```

Figura 53 - Dialogo de guardar descripción en el registro

Utilizando “add” para agragar registro en la colección gastos.

```
@Override
public void saveAccountToDB() {
    email=getActivity().getIntent().getStringExtra( name: "email");
    registro.setType(0);//tipo del registro gastos
    registro.setEmail(email);//usuario del registro correspondiente
    db.collection( collectionPath: "gastos").add(registro).addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
            @Override
            public void onSuccess(DocumentReference documentReference) {
            Toast.makeText(getContext(), text: "Registro insertado correctamente", Toast.LENGTH_SHORT).show();

            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Toast.makeText(getContext(), text: "Error al insertar un registro", Toast.LENGTH_SHORT).show();
            }
        });
```

Figura 54 - Agregar registros en la colección de Firestore Utilizando add