FINANGOALS

Proyecto de Fin de Ciclo

Desarrollo de aplicación Android orientada a la gestión de finanzas personales.

REALIZADO POR:

Carmen Álvarez-Montenegro Piñeiro

Andrés Avendaño Monteagudo

TUTORIZADO POR:

Raquel Cerdá

1. Introducción

En un entorno económico cada vez más desafiante, donde la inflación, el elevado costo de acceso a la vivienda y los salarios estancados complican la gestión financiera personal, nace FinanGoals, una aplicación Android diseñada para simplificar la administración del ahorro personal. FinanGoals ofrece a los usuarios una herramienta intuitiva para monitorear sus ingresos y gastos, establecer objetivos de ahorro y gestionar su progreso financiero de manera eficiente.

La motivación detrás de este proyecto surge de la necesidad de brindar una solución accesible y efectiva que ayude a las personas a manejar sus finanzas de manera más organizada. En un mercado saturado de aplicaciones financieras, FinanGoals se distingue por su enfoque proactivo en la planificación financiera. La aplicación permite a los usuarios registrar sus transacciones y seguir su progreso hacia sus metas de ahorro, ofreciendo una visión clara y detallada de su situación financiera.

Los objetivos generales de FinanGoals son proporcionar una herramienta que facilite la gestión financiera personal, promover hábitos de ahorro saludables y mantener a los usuarios motivados en la consecución de sus metas económicas. A través de notificaciones y mensajes, la aplicación busca mantener a los usuarios comprometidos con sus objetivos, ayudándolos a adoptar buenas prácticas financieras y a alcanzar una mayor estabilidad económica.

En resumen, FinanGoals es una solución práctica que responde a las necesidades actuales de gestión financiera personal, ofreciendo una herramienta fácil de usar y motivadora, contribuyendo así al bienestar y seguridad financiera de sus usuarios.

2. Palabras clave

Gestión financiera, objetivos de ahorro, hábitos de ahorro, finanzas personales, planificación económica, ingresos y gastos, pedagogía financiera, seguimiento de metas, ahorro personal, estabilidad económica, monitoreo financiero, herramientas de ahorro, educación financiera, tips de ahorro, control financiero, Android, Kotlin, Jetpack Compose, Firebase, Material Design 3, notificaciones push.

3. Índice general

4. Módulos formativos aplicados en el trabajo

Programación multimedia y dispositivos móviles

La mayor parte del proyecto se ha basado en los contenidos aprendidos en esta materia, lo que nos ha permitido profundizar más en el entorno de desarrollo Android Studio así como el gestor de bases de datos Firestore de Firebase.

Programación

Si bien es cierto que no hemos desarrollado nuestra aplicación con ningún lenguaje de programación impartido en esta asignatura durante el curso, sí ha sido fundamental contar con conocimientos de programación para aprender un nuevo lenguaje, Kotlin, que será la base del desarrollo de la aplicación.

Entornos de desarrollo

Entornos de desarrollo nos ha servido para trabajar con Github de una manera más cómoda y eficiente, permitiéndonos tener constantemente nuestro proyecto actualizado y unificado.

Además, la elaboración de un diagrama de casos de uso nos ha servido posteriormente para diseñar la funcionalidad de la aplicación.

Desarrollo de interfaces

Los conocimientos adquiridos aquí en relación a material design y la usabilidad han sido claves para el diseño y funcionalidad de la aplicación. Al igual que hemos desarrollado la aplicación en un nuevo lenguaje, el diseño de la interfaz se ha realizado con JetPack Compose. Hay que destacar que en este caso con JetPack Compose no guarda similitud con otra herramienta de diseño que hayamos aprendido antes.

5. Herramientas y Lenguajes utilizados

Todas las tecnologías que los alumnos usarán para el proyecto, como pueden ser, lenguajes de programación, frameworks, bases de datos, IDE, etc.

Android Studio

Entorno de desarrollo para aplicaciones móviles, en este proyecto hemos utilizado la versión Android Studio Iguana 2024.2.1. Este entorno destaca porque puede aplicarse a múltiples lenguajes como Java, C++ o Kotlin, nos proporciona emuladores que podemos elegir y configurar según nuestras necesidades pero sobre todo nos proporciona la funcionalidad de la previsualización en tiempo real.

Kotlin

Es un lenguaje de programación de código abierto que en los últimos años está ganando terreno a Java en el desarrollo de aplicaciones móviles. Destaca entre otras, su curva de aprendizaje que es más sencilla que otros lenguajes, la reducción del número de líneas de código es importante, además Kotlin nos permite trabajar de manera conjunta con Java y viceversa, por ejemplo en nuestro proyecto para el tratamiento de las fechas nos hemos apoyado en clases del paquete de Java.time aun estando el código en Kotlin.

JetPack Compose

Nos ofrece un conjunto de herramientas para la creación de interfaces de usuario en Android. Se integra totalmente con Kotlin , es conciso y fácil de usar ya que aprovecha el paradigma de programación declarativa creando las vistas de forma más rápida y eficiente, es decir, primero describimos cómo queremos que sea la interfaz, ésta está relacionada con un estado de manera que cuando el estado cambia la vista se actualiza automáticamente visualizando la nueva interfaz.

Firebase

Es una plataforma de desarrollo para aplicaciones móviles y web de Google, que para nuestro proyecto nos ha proporcionado los siguientes servicios:

- Firestore Database: como gestor de bases de datos no relacionales, almacena los documentos en estructura JSON.
- Authentication: nos permite verificar la autenticación de los usuarios en las aplicaciones.
- Messaging: nos permite gestionar y enviar notificaciones al usuario.

GitHub

Plataforma que nos permite trabajar de forma colaborativa en distintos proyectos, se apoya en Git (sistema de control de versiones) para llevar un seguimiento de los cambios que se van haciendo en el proyecto.

Draw Io

Es una aplicación que nos ayuda a dibujar distintos tipos de diagramas, para nuestro trabajo nos ha permitido realizar un diagrama de casos de uso. Es fácil de usar, sistema de arrastrar y soltar en el espacio de trabajo además nos permite exportar a prácticamente cualquier herramienta de edición y transformar a imagen o PDF.

Canva

Es una aplicación que nos permite entre otras cosas, diseñar plantillas, carteles, documentos, presentaciones,videos… en nuestro caso se ha utilizado para el diseño del logo de la aplicación y elección de la paleta de colores.

Figma

Figma es una herramienta que nos ayuda en el diseño de aplicaciones móviles y web. Una de sus principales ventajas es que es un programa alojado en la nube, permitiendo el trabajo colaborativo, fácil de usar e intuitivo. El mayor inconveniente que hemos tenido es que la versión gratuíta tiene un número limitado de páginas.

6. Componentes del equipo y aportaciones

Carmen Álvarez-Montenegro

APORTACIONES

- Diseño logo, paleta de colores y nombre app.
- Diagrama de casos de uso.
- Home.
- Consulta de ahorros y aportaciones.
- Configuración barra de navegación.

Andrés Avendaño Monteagudo

APORTACIONES

- Inicio de Sesión y Registro
- Notificaciones Push
- Gastos y Ingresos
- Pantalla de detalle de objetivos
- Homogeneización estética

7. Fases del proyecto

MARZO

Entrega del anteproyecto

Desde el principio tuvimos claro que nuestro proyecto debería de estar relacionado con algún campo en el que alguno de los miembros del equipo tuviese experiencia, aplicar nuestro aprendizaje de estos dos años a la enseñanza o a las finanzas sería culminar nuestra formación.

Quizás ésta ha sido una de las fases más difíciles, ¿qué hacer?, ¿por qué?, ¿qué utilidad tiene?, ¿cómo hacerlo?, ¿qué herramientas y qué lenguajes vamos a utilizar?, ¿cómo se va a llamar?, y ¿el logo?. Sin duda surgieron innumerables preguntas, pero que poco a poco se han ido definiendo en nuestro anteproyecto.

Decidimos que nuestro proyecto sería una aplicación móvil para el control de las finanzas personales, que ayude a fomentar, en la medida de lo posible, el ahorro sobre todo en ese usuario joven con ganas de conseguir muchas cosas pero con falta de planificación y esfuerzo financiero, en definitiva, tal y como lo definimos en el anteproyecto:

“La aplicación busca que los usuarios adopten hábitos de ahorro más eficientes al mismo tiempo que llevan un control de su economía doméstica.”

Figura 1. Logo de FinanGoals

El logo simula un grupo de hojas o un ala en movimiento, recordando a cuando tomábamos notas y echábamos cuentas en la libreta y en el caso del ala recuerda simbólicamente a emprender un vuelo; las alas que adquiere el usuario cuando descarga FinanGoals en su smartphone y empieza a fijar sus metas.

Elegimos como color base de nuestra aplicación el color naranja ya que según BirdCom (agencia dedicada al marketing y publicidad digital).

“El color naranja quiere decir entusiasmo, confianza, éxito, te permite comunicarte con gente joven ya que transmite optimismo, aventura y sociabilidad. El color naranja hace que un producto parezca más accesible”.

Apoyándonos de Balsamiq en esta fase hacemos nuestro primer esbozo de lo que puede ser nuestra aplicación.

Figura 2. Mockups del proyecto

Cuando ya teníamos la idea, el logo, el nombre, el diseño incluso los colores, decidimos que también podría ser un buen momento para seguir formándonos en nuevos lenguajes

y herramientas que están en auge en estos últimos años en el desarrollo de aplicaciones móviles como Kotlin y Jetpack Compose. Y a partir de aquí empezó una montaña rusa de emociones.

ABRIL

Semanas 1-2 : Formación, formación y más formación

Ninguno de los miembros del equipo teníamos conocimiento alguno en Kotlin y JetPack Compose, y esto suponía un gran reto, ya que no solo es la base sino también la estructura de todo el proyecto, por lo que la formación fue crucial para empezar a tener contacto y hacer una inmersión total que nos ha llevado todo este tiempo.

Además de las páginas oficiales de Kotlin y JetPack Compose, nos apoyamos en tutoriales de formadores reconocidos en el mundo de la programación y desarrollo tales como MoureDev de Brais Moure, Programación Android by AristiDev, DevExpert entre otros. Nada más terminar la formación empezamos con la parte más sencilla del proyecto y que nos permitió poner en práctica los conocimientos más básicos de Kotlin y Compose, el diseño y la funcionalidad de las preguntas (*FirstQuestion* y *SecondQuestion*) lanzadas al usuario que se registra por primera vez en FinanGoals.

Paralelamente a la formación vamos diseñando con la ayuda de Figma cada una de las pantallas que conformarán FinanGoals, ya con los colores, el logo y los iconos acordes a la funcionalidad, sin perder de vista la sencillez y usabilidad de la app.

Acordamos en esta fase, definir la estructura que tendrá nuestro proyecto Android, tuvimos dudas de estructurar nuestro proyecto bajo la arquitectura MVVM (View Model) que sin lugar a dudas un proyecto separado por capas nos hubiese ayudado a mantener el código más limpio, estructurado y eficiente, pero finalmente decidimos realizar nuestra propia estructura pues nos resultaba un poco complejo MVVM y

teniendo en cuenta que tanto Kotlin como el JetPack Compose eran nuevos para nosotros y estaba suponiendo un esfuerzo.

Estructura:

- **db**: *FirestoreManager.kt* gestor de las operaciones relacionadas con la metas y transacciones financieras y Firebase Firestore como gestor de base de datos.
- **messaging***: MessagingChannel, MessagingService* se encarga de gestionar el servicio de mensajería y notificaciones con el usuario.
- **ui.theme:**
    - **navigation:** *Navigation.kt, NavRoutes y NavigationBar.kt,* controlador de la navegación, definición de las rutas de navegación de las vistas y diseño y funcionalidad de la barra de navegación de la app para el usuario.
    - **screens**: todas y cada una de las vistas que conforman la aplicación, incluido todo lo relacionado con el login del usuario.
        - *ExpensesIncomeScreen.kt.*
        - *GoalScreen.kt, HomeScreen.kt.*
        - *AuthController.kt,LoginScreen.kt, SignupScreen.kt.*
        - *FirstQuestion.kt, SecondQuestion.kt.*
        - *SavingsSummaryScreen.kt, SummaryScreen.kt.*
    - **theme**: todo lo relacionado con el color y tema de la app.

Elaboramos un diagrama de casos de uso que nos ayudará a definir tanto la funcionalidad de la app así como de todas aquellas acciones que puede o debe realizar el usuario para interactuar con ella.

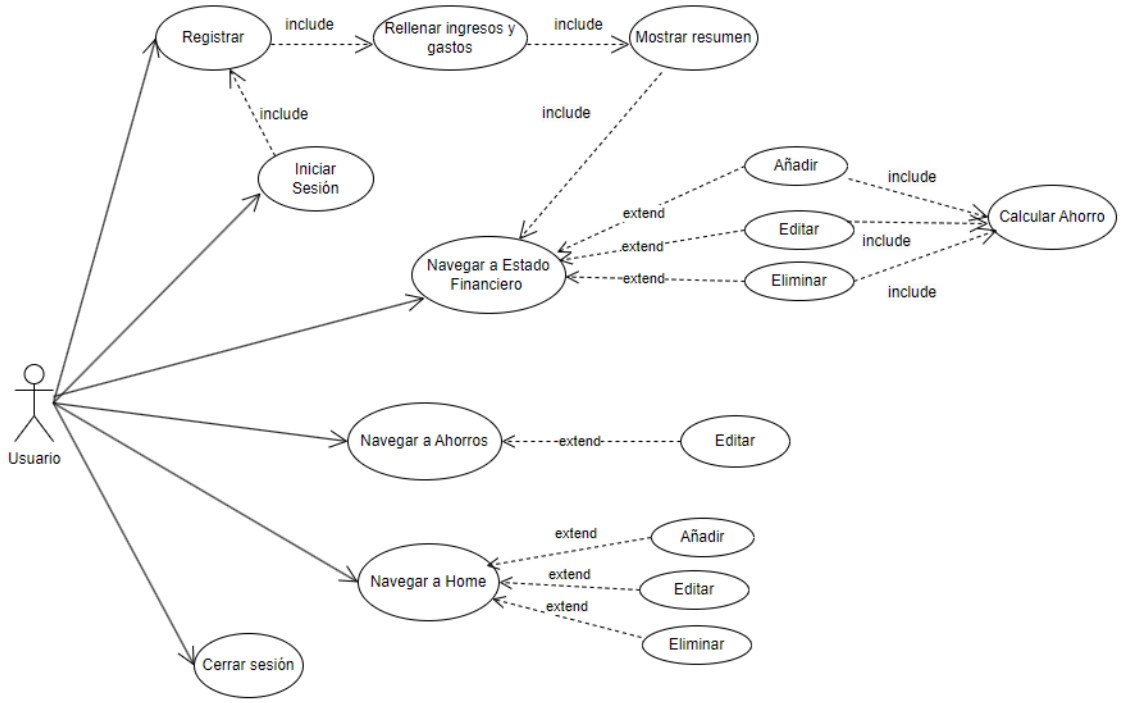

Figura 3. Diagrama de casos de uso

Semana 3:

Durante esta semana, se implementa en *AuthController* el módulo de autenticación utilizando Firebase Authentication. Siguiendo su documentación, configuramos el inicio de sesión con correo electrónico y contraseña y a través de Google, asegurando una experiencia de usuario fluida y segura. Se integraron las pantallas de registro e inicio de sesión con sus respectivas validaciones, procurando que fueran lo más responsive posible, y se realizaron pruebas para garantizar que los usuarios pudieran registrarse y acceder a sus cuentas sin problemas. A este respecto, la gestión del inicio de sesión y registro con Google fue lo más complejo, pues requiere de distintas funciones y objetos como:

- **googleSignInClient:** objeto que se configura con los parámetros de inicio de sesión (email y token).
- **handleSignInResult:** función que maneja el resultado del intento de inicio de sesión.

- **signInWithGoogleCredential:** método asíncrono que maneja el inicio de sesión con las credenciales de Firebase y permite saber si el usuario es nuevo o no.
- **signInWithGoogle:** utiliza el objeto googleSignInClient para obtener el intent (el lanzador) de inicio de sesión.

Simultáneamente, comenzamos también con lo que será la vista principal de la app, la *HomeScreen*, que a lo largo de las semanas y hasta el final se va modificando y adaptando a la funcionalidad de la app.

HomeScreen tiene como función gestionar la pantalla principal, cuenta con un *scaffold*, que incluye un top bar y la barra de navegación, los cuales posteriormente extraeremos al archivo *NavigationBar.kt*, evitando la redundancia de código en el resto de las vistas, además de un *LazzyColum* que permitirá al usuario hacer scroll de todas las metas.

Además, esta *screen* cuenta con un botón flotante, que cuando se acciona por el usuario, generará un *alert dialog* en el que podrá introducir los datos requeridos para añadir cada una de las metas.

- **description**: contiene el nombre del objetivo que asigne el usuario.
- **amountText**: importe del objetivo.
- **endDate**: fecha límite fijada por el usuario para alcanzar el objetivo.

Cada objetivo definido por el usuario se visualizará en una card con la siguiente disposición:

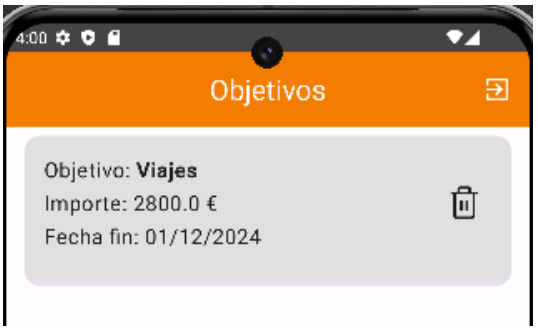

Figura 4. Vista de una meta

En todos los campos se han realizado las funciones de validación necesarias, para evitar la introducción de datos incorrectos por parte del usuario. Inicialmente el campo fecha lo tenía que introducir el usuario en el formato específico de la función, más adelante se cambia por un date picker para darle más funcionalidad y mejorar la usabilidad.

Por otro lado, se lleva a cabo la implementación de las notificaciones push. Para ello, configuramos el servicio de mensajería de Firebase (Firebase Cloud Messaging - FCM) para permitir la entrega de mensajes motivacionales y consejos de ahorro generales a los usuarios. Se crea un *MessagingService* que extiende *FirebaseMessagingService*, para manejar la recepción de mensajes y notificaciones dentro de la app. También se configura el archivo *AndroidManifest.xml* para declarar los permisos necesarios y registrar el servicio de mensajería. Tras varias pruebas iniciales sin éxito, creamos el canal de notificaciones, que es esencial para asegurar que las notificaciones push se manejen adecuadamente en dispositivos con Android 8.0 o superior. Para ello, se define una clase *MessagingChannel*, que se encarga de esta configuración, y se añade una verificación de la versión de Android para asegurar que el canal se configure correctamente en todos los dispositivos. Tras esto, se realizaron pruebas exhaustivas para asegurar que las notificaciones se entregaran correctamente y se mostraran de manera adecuada a los usuarios, tanto en primer plano como en segundo plano.

Así, a través de la gestión que ofrece Firebase a través de la creación de campañas de notificaciones, nos centramos en enviar mensajes generales para motivar a los usuarios y proporcionarles consejos sobre cómo mejorar su ahorro. Con esto, buscamos mantener a los usuarios informados sobre sus metas de ahorro, motivándolos a seguir contribuyendo a sus objetivos.

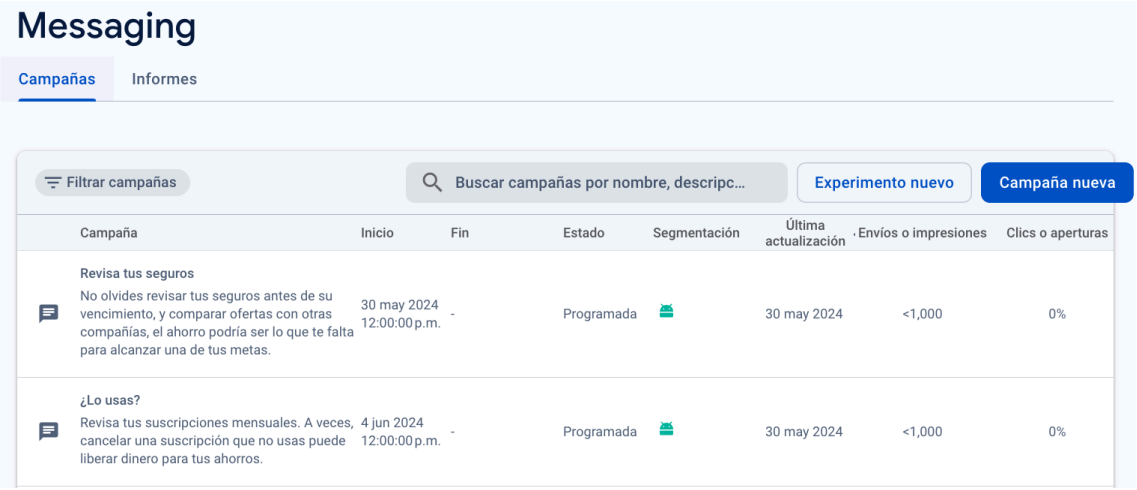

Figura 5. Consola de Firebase Messaging

Semana 4: Entrega intermedia

Configuramos la barra de progreso, que medirá el grado de avance de cada una de las metas del usuario en base a la fecha final y a las aportaciones que vaya haciendo a lo largo del tiempo.

Definimos una función, *calculateProgress,* que recibe como parámetros las variables *endDateGoal*, *amount*, y *aport* (fecha fin, importe del objetivo a alcanzar y aportaciones recurrentes del usuario). La función tiene varias partes complejas:

- Hay que declarar una variable *'df'* que se encargará de formatear la fecha que nos llega como cadena de texto al formato definido dd/mm/yyyy. Y declaramos otra variable *start* que representa el inicio del año.
- Definimos las siguientes variables:
    - por un lado la fecha actual, *todayDate* esta es la fecha actual que le aparece al usuario cuando haga click en el *datepicker* al añadir una nueva meta.
    - la fecha fin, *endDate,* fecha que marca el usuario al añadir la meta.
    - *totalDays* calcula el total de días que van desde el inicio hasta la fecha fin (start → endDate)

- *daysPassed,* rpresenta los días que han pasado desde el inicio hasta la fecha de consulta (start→ todayDate).
- *timeProgress,* calcula el % del año que ha transcurrido desde el inicio hasta la fecha de consulta, de esta manera podemos ver el grado de avance de la barra de progreso.

Esto con respecto al tiempo, pero la barra de progreso también avanza con respecto a la cantidad aportada recurrentemente por el usuario a su objetivo, simplemente comparamos las variables *amount* y *aport* para medir el grado de avance. En cualquier caso, la barra alcanzará el 100%, cuando el usuario haya realizado las aportaciones necesarias para alcanzar su objetivo.

Por otro lado, en esta semana, configuramos Firebase Firestore. Diseñamos la estructura de la base de datos en colecciones para almacenar la información de los usuarios. Inicialmente se valoró la posibilidad de integrar todos los datos en una sola colección, pero enseguida tomó fuerza la idea de segmentar la estructura en dos para beneficiar la claridad en la gestión de los datos y optar por una estructura escalable. Así, el único escollo que hubo que salvar en la implementación fue la recuperación únicamente de los datos del usuario con sesión iniciada, que se solventó incluyendo el *userId* (para almacenar el UID de usuario logueado).

De este modo, la estructura de datos de la aplicación, integrada en Firestore, está organizada en dos colecciones principales: *goals* y *transactions*. Cada documento en la colección *goals* representa una meta de ahorro definida por un usuario. Los campos de cada documento incluyen:

- **id**: Identificador único del documento (auto-generado por Firestore).
- **description**: Descripción de la meta de ahorro.
- **amount**: Cantidad actual ahorrada hacia la meta.
- **endDate**: Fecha límite para alcanzar la meta.
- **userId**: Identificador del usuario propietario de la meta.
- **aport**: Cantidad aportada en cada contribución.

Se desarrollaron las funcionalidades de agregar nuevas metas, actualizar las existentes, listar todas las metas de un usuario y eliminarlas.

Cada documento en la colección *transactions* representa una transacción financiera, ya sea un ingreso o un gasto. Los campos de cada documento incluyen:

- **id**: Identificador único del documento (auto-generado por Firestore).
- **description**: Descripción de la transacción.
- **amount**: Cantidad de la transacción.
- **month**: Mes en que se registró la transacción.
- **type**: Tipo de transacción (INCOME para ingresos, EXPENSE para gastos).
- **userId**: Identificador del usuario propietario de la transacción.

Del mismo modo que con la colección *goals*, desarrollamos las funcionalidades de agregar nuevas transacciones, actualizar las existentes, listar todas las transacciones de un usuario y eliminarlas.

Inicialmente se crearon las clases *FirestoreManager* y *ExpenseIncomeManager*, para gestionar las metas y las transacciones de gastos e ingresos, respectivamente. Sin embargo, incluso antes de probar su funcionamiento, se tomó la decisión de utilizar una sola clase que fuera la encargada de gestionar toda la interacción con la base de datos, la clase encargada de gestionar la interacción con Firestore. Esta clase incluye métodos para agregar, actualizar, listar y eliminar tanto metas como transacciones y utiliza el sistema de autenticación de Firebase para asegurar que cada operación se realiza sobre los datos del usuario autenticado correspondiente. Se han seguido las pautas marcadas por Firebase para Kotlin para la realización del CRUD.

Para empezar a hacer las primeras pruebas no solo con Firebase sino con la navegación entre las distintas vistas de la app, se define la navegación en el archivo *Navigation.kt* donde destaca principalmente el *NavHostController* y *NavHost*; el primero es el controlador de la navegación y el segundo contiene las rutas de las distintas vistas compostables de la aplicación. La pantalla inicial se decide en función de si el usuario ha iniciado sesión a través de la función *determinateStartDestination()*

```
startDestination = determineStartDestination(authController)
```

```
// Verificamos si ya se ha iniciado sesión
andresaven
fun determineStartDestination(authController: AuthController): String {
    return if (authController.isUserAuthenticated()) {
        NavRoutes.Home.route
    } else {
        NavRoutes.Login.route
    }
```

Figura 6. Código. Navegación tras autenticación

Las rutas de la navegación las hemos definido en la clase *NavRoutes* donde cada objeto de la clase representa una ruta tal y como se muestra.

```
sealed class NavRoutes(val route: String) {
    andresaven
    object Login: NavRoutes( route: "login_screen")
    andresaven
    object Signup: NavRoutes( route: "signup_screen")
```

Figura 7. Código.Definición de las rutas

MAYO

Semana 1:

Se crea *ExpenseIncomeScreen* y se hacen las primeras pruebas con la estructura de datos. Esta pantalla permite a los usuarios registrar y visualizar sus transacciones de ingresos y gastos, proporcionando una visión clara de su situación financiera. Para su desarrollo, se parte de la estructura de *HomeScreen* en la UI, empleamos la estructura de

datos ya creada, *TransactionData,* e implementamos una función en *FirestoreManager* para calcular el ahorro total de un usuario restando el total de gastos del total de ingresos. Además, se agregó una funcionalidad para filtrar las transacciones según diferentes períodos de tiempo: mes, trimestre y año. Esto permite a los usuarios visualizar sus ingresos y gastos en distintos intervalos de tiempo, facilitando un análisis más detallado de su situación financiera. Los intervalos son: mes (muestra las transacciones correspondientes al mes actual), trimestre (agrupa las transacciones de los últimos tres meses) y año (incluye todas las transacciones realizadas en el año en curso).

Asimismo, con el fin de proporcionar una representación visual clara y comprensible de las finanzas del usuario, se decidió integrar gráficos que muestren sus datos (incorporando también el filtrado de transacciones). Se realizó una búsqueda exhaustiva de bibliotecas de gráficos adecuadas y se seleccionó *TehrasCharts*, que utiliza una presentación minimalista y evita saturar la UI. Se optó por implementar gráficos de tarta (siguiendo la estética global de la app) para visualizar la distribución de ingresos, gastos y ahorros.

Simultáneamente, se llevaron a cabo pruebas para asegurar que todas las funcionalidades implementadas operaran correctamente. Estas pruebas incluyeron la verificación de la adición, actualización y eliminación de transacciones, así como la correcta visualización y filtrado de las mismas. Esto último resultó la parte más compleja de este punto del desarrollo, pues tras varios intentos de filtrado dentro de la app que recuperaban todas las transacciones, se optó por una medida más eficiente y efectiva: la creación de un índice compuesto en Firestore y la llamada a una consulta que obtiene las transacciones de la base de datos ya filtradas. Además, se realizaron pruebas con los gráficos para asegurarse de que representaran correctamente los datos del usuario.

Además de lo concerniente al desarrollo de *ExpenseIncomeScreen*, se realizaron mejoras en la *HomeScreen* como, por ejemplo, el cambio en la introducción de la fecha de los objetivos de ahorro, implementando un selector de fecha (*DatePicker*). Inicialmente, se encontró con ciertas dificultades en la presentación gráfica de esta funcionalidad. Primero se optó por utilizar un icono clicable; no obstante, esta solución

no fue suficientemente intuitiva ni visualmente agradable y se decidió mantener un *OutlinedTexField* modificado.

La función *datePickerDialog* muestra un cuadro de diálogo de selección de fecha utilizando *android.app.DatePickerDialog*. Este cuadro de diálogo se compone de un tema personalizado y una lambda que maneja la fecha seleccionada, de manera que no sea anterior a la fecha actual y formateando adecuadamente la fecha. La fecha inicial del diálogo se ajusta según el estado actual *selectDate*. Esta implementación garantiza una selección de fecha coherente y evita errores de escritura o selección por parte del usuario.

Por último, para mantener la coherencia estética, se creó un tema personalizado para modificar la apariencia del calendario, lo que benefició su integración visual y, en consecuencia, una experiencia de usuario más agradable.

Semana 2:

Durante la segunda semana de mayo, se realizaron modificaciones estéticas en las pantallas *FirstQuestion* y *SecondQuestion*. Estas pantallas forman parte del cuestionario inicial que los usuarios completan al registrarse en la aplicación, proporcionando información sobre sus ingresos y gastos habituales. Las modificaciones incluyeron mejoras en el diseño de la interfaz de usuario para hacerlas más atractivas y fáciles de usar, alineándose con los principios de Material Design 3.

Después de las pantallas de preguntas, se incorporó la *SummaryScreen*, que muestra un resumen visual de los gastos e ingresos habituales y los ahorros del usuario. Esta pantalla incluye un gráfico que representa los datos financieros recopilados durante el cuestionario inicial, proporcionando una visión clara y comprensible de la situación económica del usuario. En este caso se importó la librería de *PhilJay MPAndroidChart*, una biblioteca muy reconocida en Android y con una variedad de gráficos.

Para soportar la nueva funcionalidad de la *SummaryScreen*, se implementó en *FirestoreManager* una funcionalidad que permite agregar los datos de los gastos e ingresos habituales proporcionados por el usuario durante el cuestionario inicial como

transacciones iniciales en la *ExpenseIncomeScreen*. Esta función asegura que los datos financieros relevantes estén disponibles y visibles para el usuario desde el principio.

Por otro lado, se creó la vista de detalle de los objetivos (*GoalScreen*), que se despliega al hacer *tap* en la *card* correspondiente de un objetivo, bien en la *HomeScreen*, bien en la *SavingsSummaryScreen*. La *GoalScreen* permite a los usuarios ver los detalles de cada objetivo de ahorro, incluyendo la descripción, el importe, las aportaciones, la fecha final y el progreso actual hacia la meta. Esta pantalla proporciona una interfaz intuitiva para que los usuarios puedan editar sus objetivos según sea necesario.

Semana 3:

Durante la tercera semana de mayo, se desarrolló la *SavingsSummaryScreen*, una pantalla que muestra una visión completa de todos los objetivos de ahorro del usuario, destacando el progreso alcanzado y las aportaciones realizadas. Se incluye un gráfico que representa visualmente qué metas requieren mayor aportación para su consecución, es decir, cuales requieren por parte del usuario realizar un mayor esfuerzo financiero. La funcionalidad principal de esta vista es proporcionar al usuario un resumen visual que combine tanto la fecha de finalización de los objetivos como las aportaciones realizadas, brindando una visión holística del estado de sus ahorros, además de calcular no sólo de forma numérica, sino también porcentual lo que le queda al usuario pendiente de aportar para cumplir su objetivo.

En relación con la funcionalidad de esta *screen*, se modificó el cálculo del progreso de los objetivos de ahorro. Anteriormente, el progreso se calculaba únicamente en función de la fecha de finalización establecida por el usuario. Sin embargo, se mejoró esta funcionalidad para incluir también las aportaciones realizadas. Esto proporciona una representación más precisa y motivadora del avance del usuario hacia sus metas financieras. Ahora, el progreso se calcula combinando ambas variables: la fecha de finalización y el importe de las aportaciones, ofreciendo una visión más completa y realista del estado de cada objetivo.

Por último, además de las nuevas funcionalidades, se realizaron diversos ajustes estéticos y de usabilidad para homogeneizar la apariencia de la aplicación y solucionar problemas detectados:

- **Superposición del FAB con el último elemento de las *LazyColumn***: Se introdujo un elemento de espaciado para evitar que el FAB se superponga con el último elemento en las *LazyColumn*, mejorando así la experiencia de usuario y la accesibilidad de los contenidos.
- **Despliegue del teclado numérico en campos numéricos**: Se implementó la configuración para que el teclado numérico se despliegue automáticamente en campos destinados a la introducción de datos numéricos, facilitando la entrada de información y reduciendo errores.
- **Ajustes de tamaño de elementos**: Se realizaron ajustes en el tamaño de diversos elementos de la interfaz, incluyendo tipografías y gráficos, para asegurar una apariencia coherente y una mejor legibilidad en toda la aplicación. Estos cambios también incluyeron la estandarización de los tamaños de texto y elementos gráficos.
- **Mejoras estéticas generales**: Se realizaron ajustes adicionales para unificar el estilo visual de la aplicación, asegurando que todas las pantallas y componentes tengan una apariencia consistente y profesional.

Semana 4: Resultado final

Llega la parte final del proyecto, decidimos unificar a partir de aquí las partes del proyecto, las dos últimas ramas en la que hemos trabajado; se gestionan los conflictos que básicamente son importaciones de librerías en el gradle, se realizan con el emulador los últimos tests para asegurarnos de que la aplicación funciona como se espera y se realiza la puesta en común y análisis de la aplicación: ¿hemos cumplido con el alcance de nuestro proyecto? Sin duda, sí. Se podrían seguir realizando pequeñas mejoras, pero estamos ante una aplicación útil y plenamente funcional. Es el momento de ver el resultado y empezar a escribir nuestra historia.

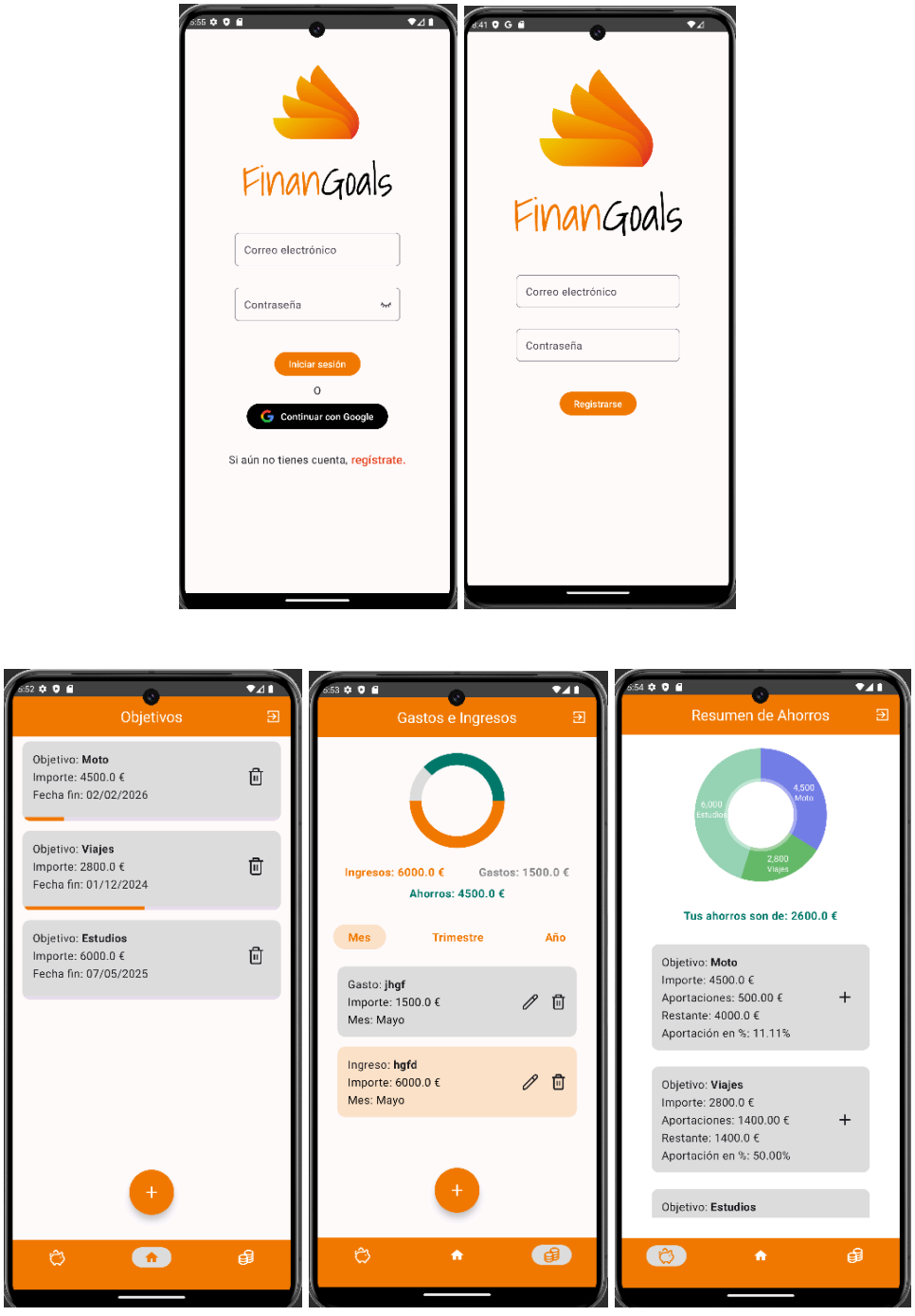

Figura 8. Mockups definitivos de FinanGoals

8. Conclusiones y mejoras

El desarrollo de FinanGoals ha sido un proyecto desafiante que no solo nos ha permitido aplicar y consolidar los conocimientos adquiridos durante el Grado, sino también nos ha posibilitado el hecho de enfrentarnos a nuevos retos y tecnologías emergentes en el desarrollo de aplicaciones móviles.

En este punto del proyecto, podemos decir que FinanGoals se ha convertido en una herramienta robusta y accesible para la gestión del ahorro personal, logrando cumplir con los objetivos planteados en el anteproyecto. La aplicación ha logrado un desarrollo integral utilizando Kotlin y Jetpack Compose, con integración de Firebase para la autenticación y la gestión de bases de datos. Ofrece una funcionalidad completa que permite registrar ingresos y gastos, establecer y seguir objetivos de ahorro, y recibir notificaciones motivacionales, cumpliendo con los propósitos educativos y de gestión financiera planteados. Gracias a Jetpack Compose y a los principios de Material Design 3, se ha conseguido una interfaz de usuario intuitiva, atractiva y fácil de usar, mejorando la experiencia del usuario. Además, la implementación de notificaciones push mantiene a los usuarios comprometidos con sus objetivos de ahorro, proporcionando tips y mensajes motivacionales.

Por otro lado, la realización de este proyecto representa la culminación de nuestra formación en el Grado Superior en Desarrollo de Aplicaciones Multiplataforma, al aplicar una combinación de conocimientos teóricos y prácticos adquiridos durante la formación, incluyendo los fundamentos de programación, diseño de interfaces, y gestión de bases de datos. Aprender Kotlin y Jetpack Compose desde cero ha sido un desafío significativo que nos ha permitido adquirir habilidades actuales y relevantes en el desarrollo de aplicaciones Android. Además, el uso de GitHub ha facilitado el trabajo colaborativo, permitiendo una gestión eficiente del código.

Entre las propuestas de mejora para FinanGoals se incluye la sincronización con las cuentas bancarias del usuario para automatizar la actualización de las transacciones, proporcionando una visión más precisa y en tiempo real de sus finanzas. Además, la integración de Firebase Analytics permitiría personalizar las notificaciones push según

el comportamiento y las necesidades individuales de cada usuario, mejorando la relevancia y efectividad de los mensajes motivacionales. Por último, añadir funcionalidades adicionales como la integración de inteligencia artificial para analizar patrones de gasto o incorporar recomendaciones personalizadas de ahorro y reportes financieros detallados podría aumentar significativamente el valor de la aplicación para los usuarios.

En resumen, FinanGoals no solo ha sido un proyecto exitoso desde el punto de vista técnico, sino también una experiencia de aprendizaje enriquecedora que nos ha preparado para futuros desafíos en nuestra recién estrenada carrera laboral como desarrolladores de software.

9. Bibliografía

Material Design. (2024). Material Design. Obtenido de https://m3.material.io

Android. (2024). Developers. Obtenido de JetPack Compose: https://developer.android.com/develop/ui/compose?hl=es-419

Kotlin. (2024). Obtenido de Kotlin: https://kotlinlang.org/

Firebase. (2024). Firebase. Documentación Firebase-Firestore: https://firebase.google.com/docs/build?hl=es-419

Moure,B.(2024).MoureDev.Obtenidode https://www.youtube.com/watch?v=ebQphhLpJG0

Aris. (2024). AristiDev. Obtenido de https://www.youtube.com/watch?v=vJapzH_46a8

Android. (2024). Developers. Obtenido de DatePickerDialog: https://developer.android.com/reference/android/app/DatePickerDialog

Aranda, R. (2024). Raquel Aranda. Obtenido de Tratamiento de fechas: https://www.youtube.com/watch?v=iaiN2rDtqF0

Jahoda, P. (2024). PhilJay. Obtenido de MPAndroidChart: https://github.com/PhilJay/MPAndroidChart?tab=readme-ov-file#gradle-setup

Weekly Coding. (2024). Obtenido de MPAndroidChart: https://weeklycoding.com/mpandroidchart/

SoftSkilsDevs. (2024). Obtenido de Date Picker: https://www.youtube.com/watch?v=Gg4xRAui7Vw

DevExpert. (2024). Obtenido de Inicio Sesión Protegido: https://www.youtube.com/watch?v=tZe5IGqdckQ&t=339s

DevExpert. (2024). DevExpert. Obtenido de notificaciones Push: https://www.youtube.com/watch?v=W7YbqBX2KcA

DevExpert. (2024). DevExpert. Obtenido de Login Google y Firebase: https://www.youtube.com/watch?v=6OVKnRqzcgo&t=441s

García, G. (2024). Obtenido de Login Creación Usuarios en Firebase: https://www.youtube.com/watch?v=NFot9_bSFhw

García, G. (2024). Gibrán García. Obtenido de Login con Google: https://www.youtube.com/watch?v=Z8VEIDoYWfk&t=244s

Koshkin, T. (2024). tehras. Obtenido de https://github.com/tehras/charts?tab=readme-ov-file

Moure, B. (2024). MoureDev. Obtenido de https://www.youtube.com/watch?v=kZJhB7LlcOU&t=785s

Moure, B. (2024). MoureDev. Obtenido de Login: https://www.youtube.com/watch?v=xjsgRe7FTCU&t=387s

Programming Headache. (2024). Obtenido de DatePicker: https://programmingheadache.com/2023/11/09/custom-material3-jetpack-compose-date-picker/

StackOverflow. (2024). Obtenido de Bottom Navigation View: https://stackoverflow.com/questions/74186634/how-to-set-itemactiveindicatorstyle-on-bottomnavigationview-in-jetpack-compose