



MÓDULO PROYECTO

CFGS Desarrollo de Aplicaciones
Multiplataforma
Informática y Comunicaciones

NestApp

Tutor individual: José María Rojo Zumel

Tutor colectivo: María Cristina Silván Pardo

Año: 2024

Fecha de presentación: 15 de Junio del 2024

Nombre y Apellidos: Victor Manuel
González Bajo

Email: victorgonzalezbajo@gmail.com

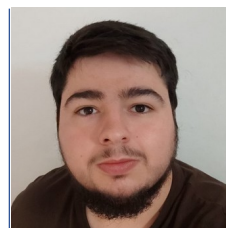


Tabla de contenido

1	Identificación proyecto.....	4
2	Organización de la memoria.....	6
3	Descripción general del proyecto.....	6
3.1	Objetivos.....	6
3.2	Cuestiones metodológicas.....	6
3.3	Entorno de trabajo (tecnologías de desarrollo y herramientas).....	6
4	Descripción general del producto.....	6
4.1	Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.....	6
4.2	Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.....	6
4.3	Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha.....	6
5	Planificación y presupuesto.....	6
6	Documentación Técnica: análisis, diseño, implementación y pruebas.....	6
6.1	Especificación de requisitos.....	6
6.2	Análisis del sistema.....	6
6.3	Diseño del sistema:.....	6
6.3.1	Diseño de la Base de Datos.....	6
6.3.2	Diseño de la Interfaz de usuario.....	6
6.3.3	Diseño de la Aplicación.....	7
6.4	Implementación:.....	7
6.4.1	Entorno de desarrollo.....	7
6.4.2	Estructura del código.....	7

6.4.3	Cuestiones de diseño e implementación reseñables.....	7
6.5	Pruebas.....	7
7	Manuales de usuario.....	7
7.1	Manual de usuario.....	7
7.2	Manual de instalación.....	7
8	Conclusiones y posibles ampliaciones.....	7
9	Bibliografía.....	7
10	Anexos.....	7

1 Identificación proyecto

NestApp es una aplicación móvil desarrollada específicamente para Android cuyo objetivo es la gestión de tareas y gastos de un grupo de compañeros de piso. Esta Desarrollada en Dart con el framework de código abierto Flutter que en resumiendo te permite hacer aplicaciones multiplataforma y usa Firebase para el manejo de la base de datos y gestión de usuarios.

La motivo por el cual elegí este proyecto es personal, ya que en mi experiencia de compartir piso no había encontrado nunca una aplicación que satisfaga todas mis necesidades a la vez, las que tenían las dos funciones no se ajustaban a mis gustos. Así que en mi piso hemos terminado por usar solo una aplicación para compartir los gastos lo que hace que se me olviden realizar mis tareas cuando las tengo programadas. Así que tras ver el hueco en el mercado de aplicaciones móviles y por necesidades personales me vi motivado a realizar el desarrollo de esta aplicación.

Lamentablemente no conseguí todo lo que me propuse al inicio del desarrollo de este proyecto pero conseguí todos los requisitos mínimos.

Respecto a la elección de las tecnologías he elegido usar Dart junto a Flutter y Firebase ya que era tecnologías que ya he usado, por lo tanto que conocía y que me gustan por la flexibilidad con la que te permiten trabajar.

2 Organización de la memoria

1 Identificación proyecto:

En esta sección se detallan los datos esenciales del proyecto.

2 Organización de la memoria:

Este capítulo describe la estructura y el contenido de la memoria del proyecto, proporcionando un esquema general de los capítulos y secciones que la componen.

3 Descripción general del proyecto

3.1 Objetivos:

Se exponen los objetivos principales y específicos del proyecto , los cuales guiarán el desarrollo y las funcionalidades de la aplicación.

3.2 Cuestiones metodológicas:

Se describe la metodología utilizada para el desarrollo del proyecto.

3.3 Entorno de trabajo (tecnologías de desarrollo y herramientas):

Se listan y describen las tecnologías y herramientas empleadas durante el desarrollo del proyecto.

4 Descripción general del producto

4.1 Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.

Se proporciona una visión general del sistema NestApp, incluyendo los límites del sistema, las funcionalidades básicas, los tipos de usuarios y otros sistemas con los que puede interactuar.

4.2 Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.

Se ofrece una descripción concisa de las arquitecturas, métodos y técnicas empleadas en el desarrollo de la aplicación.

4.3 Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha.

Se detalla el proceso de despliegue de la aplicación, incluyendo la plataforma tecnológica (Android e iOS), los pasos para la instalación de la aplicación y la puesta en marcha.

5 Planificación y presupuesto.

En esta sección se presenta la planificación detallada del proyecto y del presupuesto del mismo.

6 Documentación Técnica: análisis, diseño, implementación y pruebas. 6.1 Especificación de requisitos

6.1 Especificación de requisitos:

Se detallan los requisitos funcionales y no funcionales de la aplicación, especificando qué debe hacer el sistema y las restricciones bajo las cuales debe operar.

6.2 Análisis del sistema

Breve explicación de la interacción de los diferentes componentes del sistema.

6.3 Diseño del sistema:

6.3.1 Diseño de la Base de Datos

Se explica el diseño de la base de datos en Firestore, incluyendo la estructura de las colecciones y documentos utilizados para almacenar la información de la aplicación.

6.3.2 Diseño de la Interfaz de usuario:

Se presentan los diseños de la interfaz de usuario , describiendo cómo los usuarios interactuarán con la aplicación.

6.3.3 Diseño de la Aplicación:

Se describe la arquitectura de la aplicación, el diseño de las clases principales.

6.4 Implementación:

6.4.1 Entorno de desarrollo.

Se detalla la configuración del entorno de desarrollo, incluyendo las herramientas y tecnologías utilizadas.

6.4.2 Estructura del código.

Se describe la estructura del proyecto de código, detallando los principales módulos y componentes, y cómo están organizados.

6.4.3 Cuestiones de diseño e implementación reseñables.

Se discuten los desafíos de diseño e implementación más significativos y cómo fueron abordados durante el desarrollo del proyecto.

6.5 Pruebas.

Se presenta el plan de pruebas utilizado, incluyendo los casos de prueba, los resultados obtenidos.

7 Manuales de usuario

7.1 Manual de usuario:

Se proporciona un manual detallado para los usuarios finales, explicando cómo utilizar todas las funcionalidades de la aplicación.

7.2 Manual de instalación:

Se incluye un manual de instalación que describe paso a paso cómo instalar y configurar la aplicación tanto en el entorno de desarrollo como en producción.

8 Conclusiones y posibles ampliaciones:

En esta sección se presentan las conclusiones del proyecto, reflexionando sobre los logros, los desafíos y las lecciones aprendidas. También se proponen posibles mejoras y ampliaciones para futuras versiones de la aplicación.

9 Bibliografía:

Se listan todas las fuentes de información y referencias bibliográficas utilizadas durante el desarrollo del proyecto

10 Anexos

Se incluye información adicional relevante que complementa el contenido de la memoria, como fragmentos de código.

3 Descripción general del proyecto

Como ya hemos comentado el objetivo principal de la aplicación es la gestión de tareas y gastos compartidos entre compañeros de piso.

3.1 Objetivos

La aplicación debía cumplir al menos estos requisitos:

- **Creación y gestión de usuarios:** Permitir el registro, inicio de sesión y administración de usuarios.
- **Creación y gestión de grupos:** Facilitar la creación de grupos donde los usuarios puedan organizarse.
- **Base de datos en la nube:** Utilizar una base de datos en la nube que permita la sincronización en tiempo real entre múltiples dispositivos.
- **Creación y gestión de tareas:** Permitir a los usuarios crear, asignar y gestionar tareas dentro del grupo.
- **Creación y gestión de gastos comunes:** Facilitar el registro y seguimiento de los gastos compartidos entre los miembros del grupo.

Después, como objetivos secundarios me establecí estos requisitos:

- **Chat interno entre el grupo:** Implementar una funcionalidad de chat para la comunicación entre los miembros del grupo.
- **Gestión avanzada de tareas:** Permitir la creación de tareas con repeticiones complejas, como tareas recurrentes cada dos meses el último sábado del mes, y la asignación rotativa de usuarios a estas tareas.
- **Uso de la base de datos tanto offline como online:** Asegurar que la aplicación pueda funcionar sin conexión a internet y sincronizar los datos cuando se restablezca la conexión.

Y por último, como objetivos de baja prioridad estaban:

- **Diseño atractivo:** Crear una interfaz de usuario intuitiva y visualmente atractiva.
- **Documentar el proceso de subir la aplicación al PlayStore:** Incluir una guía detallada sobre cómo publicar la aplicación en Google Play Store.

3.2 Cuestiones metodológicas

Para el proyecto, decidí estructurarlo de una forma rígida. Al principio del desarrollo, diseñé y estructuré toda la lógica del proyecto y, después, a no ser que fuese imperativo, mantuve esta estructura durante todo el desarrollo. Esto me permitió trabajar de manera organizada y coherente, facilitando la implementación y el seguimiento del progreso.

Utilice la aplicación de notion para hacer el seguimiento de las tareas relacionadas con el proyecto junto a Flexcil que es una aplicación de notas, junto a estas dos herramientas diseñe y organice el proyecto.

3.3 Entorno de trabajo (tecnologías de desarrollo y herramientas)

Para el desarrollo de NestApp, utilicé un conjunto de tecnologías y herramientas que permitieron crear una aplicación robusta y eficiente:

Lenguaje de Programación:

- **Dart:** Dart es un lenguaje de programación desarrollado por Google. Es conocido por su rendimiento rápido y su sintaxis fácil de aprender. Dart es el lenguaje principal utilizado en Flutter, lo que permite el desarrollo de aplicaciones móviles nativas con un solo código base.

Framework:

- **Flutter:** Flutter es un framework de código abierto también desarrollado por Google. Permite crear aplicaciones nativas de alta calidad para iOS, MacOS Android, Linux, Windows y Web a partir de un solo código base. Flutter se destaca por su motor de renderizado rápido y su arquitectura basada en widgets, que facilita la creación de interfaces de usuario atractivas y personalizables. Además, ofrece herramientas para pruebas, depuración y compilación rápida de código.

Backend y Base de Datos:

- **Firestore:** Firestore es una base de datos NoSQL en tiempo real que permite el almacenamiento y la sincronización de datos entre los usuarios en tiempo real. Es escalable y fácil de integrar con Flutter.
- **Firestore Authentication:** Este servicio facilita la gestión de usuarios, proporcionando métodos de autenticación seguros, como el inicio de sesión con correo electrónico y contraseña, Google, y otros proveedores.
- **Firestore Cloud Messaging:** Utilizado para enviar notificaciones push a los dispositivos de los usuarios, manteniéndolos informados sobre nuevas tareas, gastos y mensajes en el chat del grupo. (No implementado en la versión presentada).

IDE:

- **Android Studio:** Android Studio es el entorno de desarrollo integrado oficial de Google para la plataforma Android. Proporciona herramientas completas para el desarrollo de aplicaciones, incluyendo un editor de código avanzado, un emulador de Android, herramientas de depuración y pruebas, y soporte para el desarrollo con Flutter y Dart. Su integración con Git permite un control de versiones eficiente y gestión de proyectos colaborativos.

Sistema Operativo de Desarrollo:

- **Ubuntu 23.10:** Ubuntu es una distribución de Linux popular por su estabilidad y seguridad. La versión 23.10 ofrece un entorno robusto y confiable para el desarrollo de software, con soporte para las últimas versiones de herramientas de desarrollo y bibliotecas. Ubuntu facilita la configuración del entorno de desarrollo para Flutter y Android Studio, proporcionando una plataforma eficiente y libre de problemas para el desarrollo de NestApp.

4 Descripción general del producto

4.1 Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.

Límites del sistema: NestApp está diseñada para ser utilizada por compañeros de piso que necesitan gestionar tareas y gastos compartidos. El sistema está limitado a dispositivos móviles que soporten Flutter, principalmente Android e iOS. La base de datos y los servicios de backend están alojados en Firebase, lo que limita la funcionalidad a la disponibilidad y capacidad de Firebase ya que se está usando el plan gratuito por lo que no soporta un gran flujo de datos.

Funcionalidades básicas:

- **Gestión de usuarios:** Registro, inicio de sesión y autenticación mediante Firebase Authentication.
- **Gestión de grupos:** Creación y administración de grupos de usuarios para compartir tareas y gastos.
- **Gestión de tareas:** Creación, asignación y seguimiento de tareas individuales y recurrentes.
- **Gestión de gastos:** Registro y seguimiento de gastos compartidos entre los miembros del grupo.

Usuarios y/o otros sistemas con los que puede interactuar:

- **Usuarios:** Compañeros de piso, cada uno con su perfil individual, que pueden formar y gestionar grupos.
- **Otros sistemas:** Firebase para autenticación, almacenamiento de datos en tiempo real.

4.2 Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.

Arquitectura MVVM (Model-View-ViewModel): Esta arquitectura separa la lógica de negocio, la UI y los datos, lo que facilita el mantenimiento y la escalabilidad del código. El Model maneja los datos y la lógica de negocio, el View representa la UI y el ViewModel actúa como un puente entre los dos. (Pese a haber sido diseñado de esta forma por falta de tiempo en el desarrollo se eliminó la figura de ViewModel incluyendo la lógica en las propias vistas)

- **Autenticación con Firebase:** Utilización de Firebase Authentication para gestionar el registro e inicio de sesión de usuarios, proporcionando seguridad y facilidad de uso.
- **Sincronización en tiempo real con Firestore:** Uso de Firestore para almacenar y sincronizar datos en tiempo real, permitiendo que los cambios se reflejen instantáneamente en todos los dispositivos conectados.
- **Gestión de tareas recurrentes:** Implementación de lógica para la creación de tareas que se repiten según patrones, como tareas mensuales o semanales.

4.3 Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha

Plataforma tecnológica:

- **Flutter y Dart:** Para el desarrollo de la aplicación móvil.
- **Firebase:** Para la autenticación, base de datos y notificaciones.

Instalación de la aplicación:

1. **Descarga:** Aunque el proyecto se ha quedado en la fase de desarrollo la idea es seguir desarrollando la aplicación y publicarla en PlayStore de forma que los usuarios puedan descargar NestApp desde Google Play Store. De momento se comparte el archivo .apk.
2. **Instalación:** El proceso de instalación es el estándar para archivos .apk, donde el usuario descarga y la instala en su dispositivo.
3. **Configuración inicial:** Al abrir la aplicación por primera vez, se solicita a los usuarios que se registren o inicien sesión con su cuenta de Firebase Authentication. Que validen su correo electrónico mediante el Email que recibirán en el correo electrónico con el que se hayan identificado y por último crear o unirse a un nuevo grupo.

Puesta en marcha:

1. **Registro/Iniciar sesión:** Los usuarios deben registrarse con su correo electrónico y una contraseña o iniciar sesión si ya tienen una cuenta.
2. **Creación de grupo:** Una vez autenticados, los usuarios pueden crear un grupo nuevo o unirse a un grupo existente mediante un código de invitación.
3. **Uso de la aplicación:** Los usuarios pueden comenzar a crear y gestionar tareas, registrar gastos y comunicarse a través del chat interno de la aplicación.

5 Planificación y presupuesto

5.1 Presupuesto

Siendo un estudiante de FP y considerando que no se ha gastado dinero en la app, el presupuesto se enfocará en el tiempo invertido y las herramientas gratuitas utilizadas. No se incluyen costos monetarios, ya que todas las herramientas y recursos utilizados son gratuitos o de acceso libre para estudiantes.

Herramientas de desarrollo:

- **Dart y Flutter:** Frameworks de desarrollo gratuitos.
- **Firestore:** Servicios gratuitos bajo el plan Spark (ideal para proyectos pequeños y educativos).
- **Android Studio:** IDE gratuito.

2. Infraestructura:

- **Portátil personal:** Uso de un ordenador portátil personal ya existente.
- **Internet:** Conexión a internet personal.

3. Recursos Humanos:

- **Horas de desarrollo:** Se estima que el desarrollo del proyecto tomó alrededor de 4 semanas, dedicando un promedio de 4 horas semanales al principio del proyecto y las ultimas semanas 14 Horas semanales .

4. Otros recursos:

- **Documentación y aprendizaje:**
 - Lo aprendido durante el curso en el modelo de Programación de Interfaces.
 - Comunidades de desarrolladores (sobre todo Stack Overflow, GitHub).
 - ChatGPT.

Resumen del presupuesto:

- **Costo total en dinero:** 0 €
- **Costo en tiempo (estimación):** 36 horas

6 Documentación Técnica: análisis, diseño, implementación y pruebas.

6.1 Especificación de requisitos

Requisitos funcionales:

Gestión de usuarios:

- Registro de nuevos usuarios con correo electrónico y contraseña.
- Inicio de sesión y autenticación de usuarios.
- Recuperación de contraseña mediante correo electrónico.
- Actualización de perfil de usuario (nombre, correo electrónico)

Gestión de grupos:

- Creación de nuevos grupos.
- unirse a un grupo mediante la id del grupo.
- Eliminación de grupos.
- Abandono de grupos.

Gestión de tareas:

- Creación de tareas con título, descripción y fecha de vencimiento.
- Creación de tareas recurrentes con intervalos (diarios, semanales, mensuales).
- Marcar tareas como completadas.

Gestión de gastos:

- Registro de nuevos gastos con título, descripción, importe y usuario que realizó el pago.
- División de gastos entre los miembros del grupo.
- Visualización del historial de gastos y saldos individuales.

Requisitos no funcionales:

Seguridad:

- Autenticación y autorización mediante Firebase Authentication.
- Encriptación de datos en tránsito y en reposo.

Usabilidad:

- Interfaz de usuario intuitiva y fácil de navegar.
- Diseño responsivo para diferentes tamaños de pantalla.

Rendimiento:

- Sincronización en tiempo real de datos entre dispositivos.
- Respuesta rápida a las interacciones del usuario.

Escalabilidad:

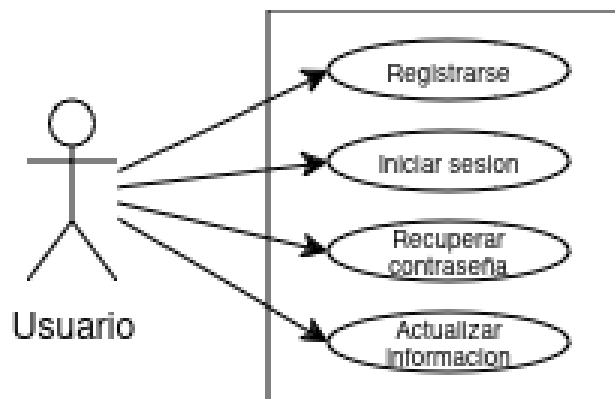
- Capacidad para manejar múltiples usuarios y grupos simultáneamente.
- Escalabilidad vertical y horizontal del backend en Firebase.

6.2 *Análisis del sistema*

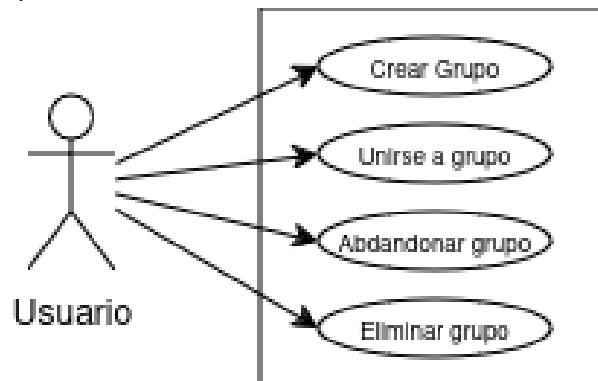
El análisis del sistema se realizó mediante la creación de varios diagramas que ayudan a entender la estructura y el flujo del sistema. A continuación, se presentan los principales diagramas utilizados:

Diagrama de casos de uso:

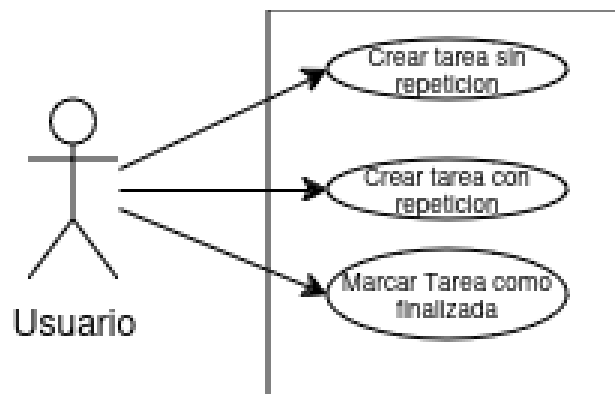
- Gestión de usuarios



- Gestión de grupos



- Gestión de tareas



- Gestión de Gastos

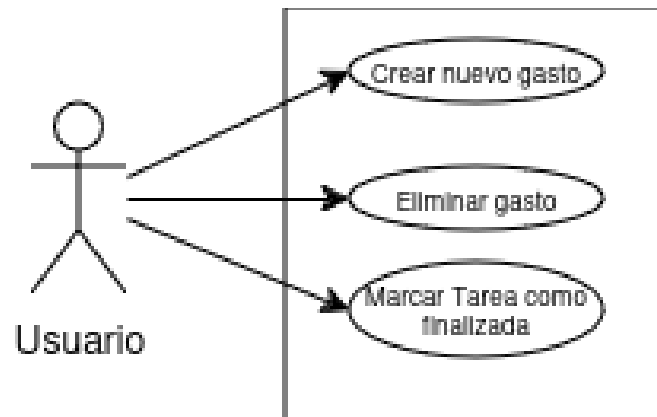
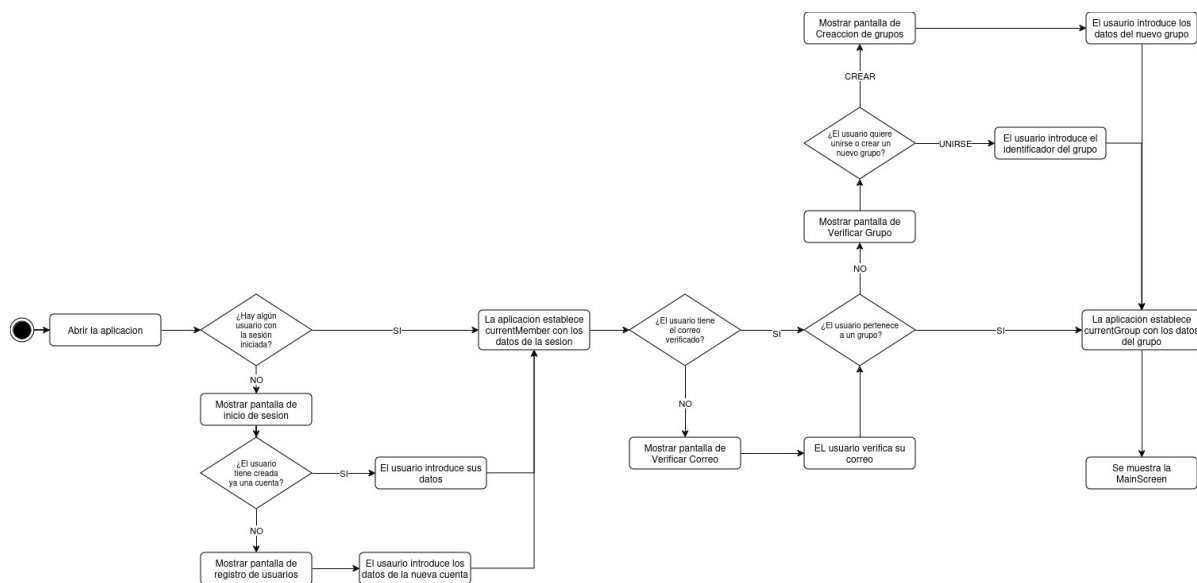


Diagrama de actividad:

EL único diagrama de actividad que he realizado es el de la lógica al entrar a la aplicación. Podréis encontrar el diagrama a mayor escala en el anexo y en github.



6.3 *Diseño del sistema:*

6.3.1 Diseño de la Base de Datos

Para el diseño de la base de datos, se ha utilizado Firestore, una base de datos NoSQL de Firebase que permite el almacenamiento y sincronización de datos en tiempo real. A continuación, se describen las principales colecciones y sus estructuras:

Colecciones principales:

1. **Members:** Representa a los usuarios, usa el UID del user proporcionado por Firebase

- userID: String (ID del usuario, clave primaria)
- name: String (Nombre del usuario)
- email: String (Correo electrónico del usuario)
- groupID: String (ID del grupo al que pertenece, clave foránea)

2. **Groups:**

- id: String (ID del grupo, clave primaria)
- name: String (Nombre del grupo)
- desc: String (Descripción del grupo)
- memberList: List<String> (Lista de IDs de los miembros del grupo)
- expenseList: List<Map> (Lista de gastos del grupo)
- noticeList: List<Map> (Lista de notificaciones del grupo)
- reminderList: List<Map> (Lista de recordatorios del grupo)
- eventList: List<Map> (Lista de eventos del grupo)

3. **Expenses:** Lista dentro de Groups

- id: String (ID del gasto, clave primaria)
- expenseName: String (Nombre del gasto)
- amount: double (Monto del gasto)
- payerUID: String (ID del pagador, clave foránea)
- sharedWithUIDs: List<String> (Lista de IDs de los usuarios con los que se comparte el gasto)

4. **Notices:** Lista dentro de Groups

- id: String (ID de la notificación, clave primaria)
- authorUID: String (ID del autor de la notificación, clave foránea)
- date: DateTime (Fecha de la notificación)

- involvedUIDs: List<String> (Lista de IDs de los usuarios involucrados)
- actionType: String (Tipo de acción notificada)
- message: String (Mensaje de la notificación)

5. Reminders: Lista dentro de Groups

- id: String (ID del recordatorio, clave primaria)
- name: String (Nombre del recordatorio)
- desc: String (Descripción del recordatorio)
- groupID: String (ID del grupo al que pertenece, clave foránea)
- autofinish: bool (Indica si el recordatorio se completa automáticamente)
- repeat: bool (Indica si el recordatorio se repite)
- repeatInterval: Map (Detalles del intervalo de repetición)
- firstDate: DateTime (Fecha inicial del recordatorio)

6. Events: Lista dentro de Groups

- id: String (ID del evento, clave primaria)
- reminderId: String (ID del recordatorio asociado, clave foránea)
- date: DateTime (Fecha del evento)
- assignedUsersUID: List<String> (Lista de IDs de los usuarios asignados)
- autofinish: bool (Indica si el evento se completa automáticamente)
- name: String (Nombre del evento)
- desc: String (Descripción del evento)
- finished: bool (Indica si el evento está finalizado)

6.3.2 Diseño de la Interfaz de usuario.

La interfaz de usuario de NestApp está diseñada con Flutter, utilizando widgets nativos que permiten una experiencia de usuario fluida y responsiva. A continuación se describen las pantallas principales:

1. Pantalla de Inicio de Sesión (LoginScreen):

- Campos de entrada para correo electrónico y contraseña.
- Botones para iniciar sesión, crear cuenta nueva y recuperar contraseña.

2. Pantalla de Registro (RegisterScreen):

- Campos de entrada para nombre, apellidos, correo electrónico, confirmación de correo electrónico, contraseña y confirmación de contraseña.
- Botón para registrar al usuario.

3. Pantalla Principal (MainScreen):

- Barra de navegación inferior con secciones para Gastos, Tareas, Home, Grupo y Ajustes.
- Cada sección muestra información relevante según la funcionalidad.

4. Pantalla de Grupo (GroupScreen):

- Muestra la información del grupo, incluyendo nombre, descripción, miembros y notificaciones.

5. Pantalla de Tareas (TaskScreen):

- Muestra un calendario con las tareas y eventos del grupo.
- Permite crear y gestionar recordatorios y eventos.

6. Pantalla de Gastos (ExpenseScreen):

- Muestra una lista de gastos del grupo.
- Permite crear nuevos gastos y ver el balance de cada miembro.

6.3.3 Diseño de la Aplicación.

El diseño de la aplicación sigue un patrón de arquitectura basado en Model-View-Controller (MVC), donde:

- **Model:** Contiene las clases que representan los datos y la lógica de negocio (e.g., Group, Member, Expense, Reminder, Event).
- **View:** Consiste en las pantallas de la aplicación que interactúan con el usuario (e.g., LoginScreen, MainScreen, GroupScreen, ExpenseScreen).

6.4 Implementación:

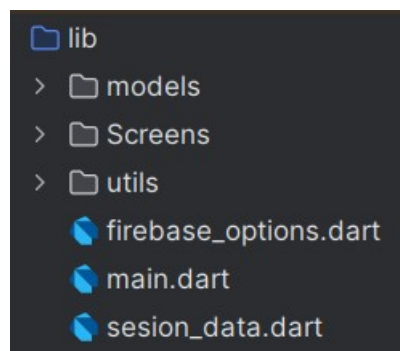
El entorno de desarrollo utilizado para NestApp incluye:

- **Sistema Operativo:** Ubuntu 23.10
- **IDE:** Android Studio
- **Lenguaje de Programación:** Dart (con el framework Flutter)
- **Base de Datos:** Firestore (Firebase)
- **Control de Versiones:** Git (repositorio en GitHub)

6.4.1 Entorno de desarrollo.

6.4.2 Estructura del código.

La estructura del código del proyecto está organizada de la siguiente manera:



6.4.3 Cuestiones de diseño e implementación reseñables.

Algunas cuestiones destacables en el diseño e implementación de NestApp son:

- **Firebase Authentication:** Implementación de autenticación de usuarios utilizando Firebase Auth.
- **Sincronización en tiempo real:** Utilización de Firestore para mantener los datos sincronizados en tiempo real entre todos los dispositivos.
- **Gestión de tareas recurrentes:** Implementación de lógica compleja para la creación y gestión de tareas recurrentes con diferentes intervalos de repetición.
- **Notificaciones:** Generación automática de notificaciones para informar a los miembros del grupo sobre nuevos gastos, tareas y eventos.

6.5 Pruebas.

Caso de Prueba 1: Validación de Correo Electrónico

Objetivo: Verificar que la función `validateEmail` valida correctamente los correos electrónicos.

Entrada:

- Correo electrónico válido: `test@example.com`
- Correo electrónico inválido 1: `test@example`
- Correo electrónico inválido 2: `test.com`

Salida Esperada:

- `true` para `test@example.com`
- `false` para `test@example`
- `false` para `test.com`

Script de Prueba:

```
import 'package:flutter_test/flutter_test.dart';
import 'package:nestapp/utils/Validator.dart';

void main() {
  test('Validar correo electrónico', () {
    expect(Validator.validateEmail('test@example.com'), true);
    expect(Validator.validateEmail('test@example'), false);
    expect(Validator.validateEmail('test.com'), false);
  });
}
```

Caso de Prueba 2: Creación de un Grupo

Objetivo: Verificar que se puede crear un grupo correctamente.

Entrada:

- Nombre del grupo: Grupo de Prueba
- Descripción: Descripción de prueba

Salida Esperada:

- El nombre del grupo debe ser Grupo de Prueba
- La descripción del grupo debe ser Descripción de prueba
- La lista de miembros debe estar vacía inicialmente

Script de Prueba:


```
import 'package:flutter_test/flutter_test.dart';
import 'package:nestapp/models/group.dart';

void main() {
  test('Crear un nuevo grupo', () {
    Group group = Group.new(name: 'Grupo de Prueba', desc: 'Descripción de prueba');
    expect(group.name, 'Grupo de Prueba');
    expect(group.desc, 'Descripción de prueba');
    expect(group.memberList.isEmpty, true);
  });
}
```

Caso de Prueba 3: Añadir Miembro a un Grupo

Objetivo: Verificar que se puede añadir un miembro a un grupo y que la actualización se refleja en Firestore.

Entrada:

- Grupo: Grupo de Prueba
- Miembro:
 - userID: uid123
 - name: Nombre Prueba
 - email: email@prueba.com

Salida Esperada:

- El miembro con userID uid123 debe estar en la lista de miembros del grupo.
- El miembro debe estar en la colección de Firestore del grupo.

Script de Prueba:

```
import 'package:flutter_test/flutter_test.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:nestapp/models/group.dart';
import 'package:nestapp/models/member.dart';

void main() {
  test('Añadir miembro a un grupo', () async {
    Group group = Group.new(name: 'Grupo de Prueba');
    Member member = Member(userID: 'uid123', name: 'Nombre Prueba', email: 'email@prueba.com');

    await group.addMember(member);
    expect(group.memberList.contains('uid123'), true);

    // Verificar en Firestore
    DocumentSnapshot groupDoc = await FirebaseFirestore.instance.collection('Groups').doc(group.id).get();
    expect(groupDoc.exists, true);
    expect(groupDoc['memberList'].contains('uid123'), true);
  });
}
```

Caso de Prueba 4: Crear y Recuperar un Gasto

Objetivo: Verificar que se puede crear un gasto y que se puede recuperar correctamente desde Firestore.

Entrada:

- Grupo: Grupo de Prueba
- Gasto:
 - id: exp123
 - expenseName: Gasto de Prueba
 - amount: 50.0
 - payerUID: uid123
 - sharedWithUIDs: ['uid123', 'uid456']

Salida Esperada:

- El gasto con id exp123 debe estar en la lista de gastos del grupo.
- El gasto debe estar en la colección de Firestore del grupo.

Script de Prueba:

```
import 'package:flutter_test/flutter_test.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:nestapp/models/expense.dart';
import 'package:nestapp/models/group.dart';

void main() {
  test('Crear y recuperar un gasto', () async {
    Group group = Group.new(name: 'Grupo de Prueba');
    Expense expense = Expense(
      id: 'exp123',
      expenseName: 'Gasto de Prueba',
      amount: 50.0,
      payerUID: 'uid123',
      sharedWithUIDs: ['uid123', 'uid456'],
    );

    await group.addExpenseToFirestore(expense);
    Group? retrievedGroup = await Group.getGroupFromFirestore(group.id);
    expect(retrievedGroup?.expenseList.any((e) => e.id == 'exp123'), true);
  });
}
```

Caso de Prueba 5: Pantalla de Inicio de Sesión

Objetivo: Verificar que los elementos de la pantalla de inicio de sesión se muestran correctamente y que las interacciones básicas funcionan.

Entrada:

- Correo electrónico: test@example.com
- Contraseña: password

Salida Esperada:

- Los campos de texto para correo electrónico y contraseña deben estar presentes.
- Los botones para iniciar sesión, crear cuenta y recuperar contraseña deben estar presentes.
- El campo de correo electrónico debe aceptar la entrada test@example.com.
- El campo de contraseña debe aceptar la entrada password.

Script de Prueba:

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'Screens/LoginScreen.dart';

void main() {
  testWidgets('Pantalla de inicio de sesión', (WidgetTester tester) async {
    await tester.pumpWidget(MaterialApp(home: LoginScreen()));

    // Verificar que los campos de texto están presentes
    expect(find.byType(TextField), findsNWidgets(2));
    expect(find.text('Correo electrónico'), findsOneWidget);
    expect(find.text('Contraseña'), findsOneWidget);

    // Verificar que los botones están presentes
    expect(find.text('Iniciar sesión'), findsOneWidget);
    expect(find.text('Crear cuenta'), findsOneWidget);
    expect(find.text('He olvidado la contraseña'), findsOneWidget);

    // Simular una entrada de texto y un clic en el botón de inicio de sesión
    await tester.enterText(find.byType(TextField).first, 'test@example.com');
    await tester.enterText(find.byType(TextField).last, 'password');
    await tester.tap(find.text('Iniciar sesión'));

    // Esperar que se procese la entrada
    await tester.pump();
  });
}
```

Caso de Prueba 6: Crear un Gasto

Objetivo: Verificar que los elementos de la pantalla para crear un gasto se muestran correctamente y que las interacciones básicas funcionan.

Entrada:

- Nombre del gasto: Gasto de Prueba
- Importe: 50

Salida Esperada:

- Los campos de texto para nombre del gasto e importe deben estar presentes.
- El dropdown para seleccionar el pagador debe estar presente.
- El botón para crear el gasto debe estar presente.
- El campo de nombre del gasto debe aceptar la entrada Gasto de Prueba.
- El campo de importe debe aceptar la entrada 50.

Script de Prueba:

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

import 'Screens/CreateExpenseScreen.dart';

void main() {
  testWidgets('Pantalla de crear gasto', (WidgetTester tester) async {
    await tester.pumpWidget(MaterialApp(home: CreateExpenseScreen()));

    // Verificar que los campos de texto están presentes
    expect(find.byType(TextField), findsNWidgets(2));
    expect(find.text('Nombre del Gasto'), findsOneWidget);
    expect(find.text('Importe'), findsOneWidget);

    // Verificar que el dropdown y el botón están presentes
    expect(find.byType(DropdownButton<String>), findsOneWidget);
    expect(find.text('Crear Gasto'), findsOneWidget);

    // Simular una entrada de texto y un clic en el botón de crear gasto
    await tester.enterText(find.byType(TextField).first, 'Gasto de Prueba');
    await tester.enterText(find.byType(TextField).last, '50');
    await tester.tap(find.text('Crear Gasto'));

    // Esperar que se procese la entrada
    await tester.pump();
  });
}
```

7 Manuales de usuario

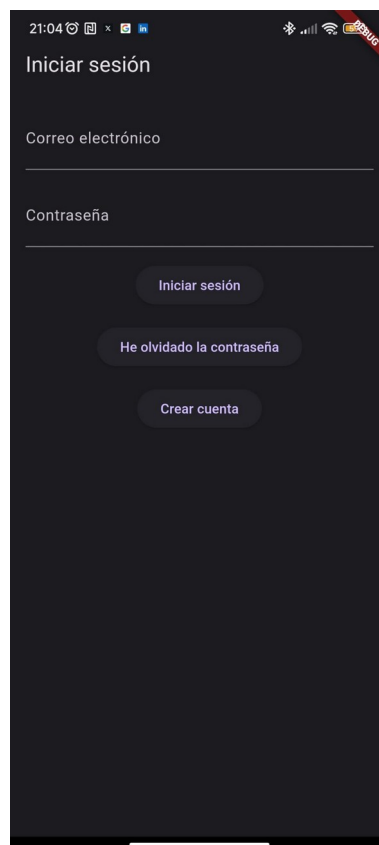
7.1 *Manual de usuario*

Acceder a la aplicación

Para acceder a las funcionalidades de la aplicación necesitaremos una cuenta con el correo electrónico verificado. Veamos el proceso de crear una cuenta, validar el correo y crear un grupo.

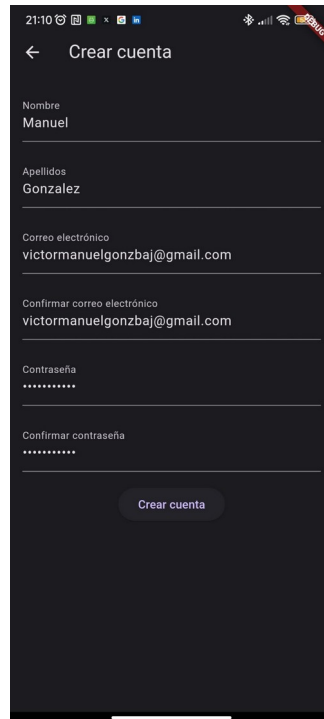
Crear Cuenta

Una vez abramos la aplicación veremos la pantalla de inicio de sesión donde podremos introducir nuestros datos de inicio de sesión o bien crear una cuenta. O también si es necesario restablecer nuestra contraseña, recibiremos un correo con las instrucciones para cambiar la contraseña.



Hacemos clic en Crear cuenta para navegar a la pantalla de creación de un nuevo usuario.

Una vez en la pantalla de creación de Usuarios rellenamos nuestros datos y hacemos clic en crear cuenta. La aplicación verificara si el correo es valido y si la contraseña tiene al menos una mayúscula, una minúscula y un numero.



21:10

← Crear cuenta

Nombre
Manuel

Apellidos
Gonzalez

Correo electrónico
victormanuelgonzbaj@gmail.com

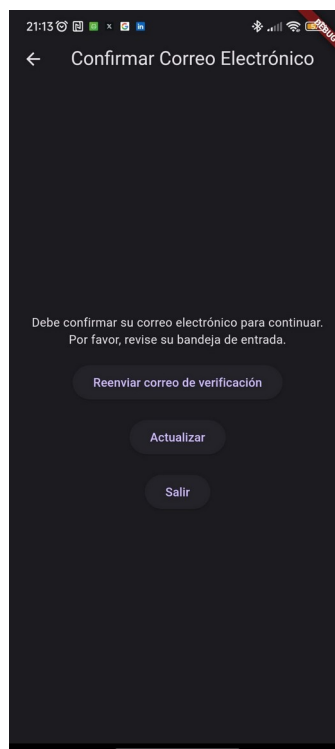
Confirmar correo electrónico
victormanuelgonzbaj@gmail.com

Contraseña
.....

Confirmar contraseña
.....

Crear cuenta

Una vez creado el usuario nos llevara a la pantalla de verificación de Correo, donde no nos dejara avanzar hasta que vayamos a nuestra bandeja de entrada y verifiquemos el correo. Después nos llevara a la pantalla de verificación de grupo.



21:13

← Confirmar Correo Electrónico

Debe confirmar su correo electrónico para continuar.
Por favor, revise su bandeja de entrada.

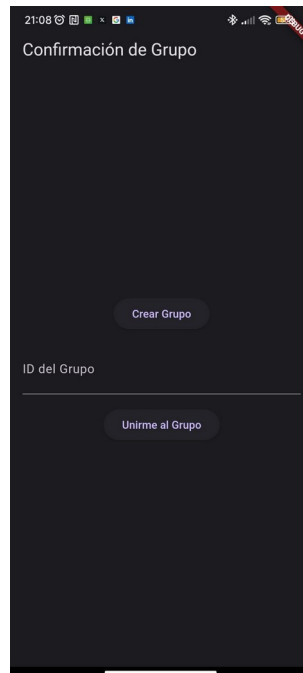
Reenviar correo de verificación

Actualizar

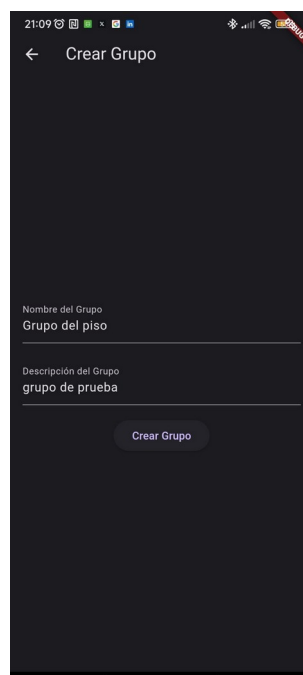
Salir

Crear o unirse a un grupo

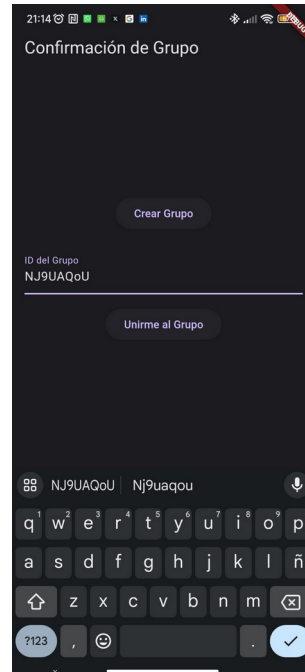
Ya sea que hayamos iniciado sesión sin tener un grupo, al salirnos de un grupo o al crear una cuenta nueva, la aplicación nos llevara a la pantalla de verificación de grupo donde nos dará la opción de crear un grupo o de acceder a uno ya creado introduciendo la ID del grupo.



Primero seleccionemos la opción de crear grupo, donde rellenaremos los datos y una vez creado nos llevara a la pantalla principal.

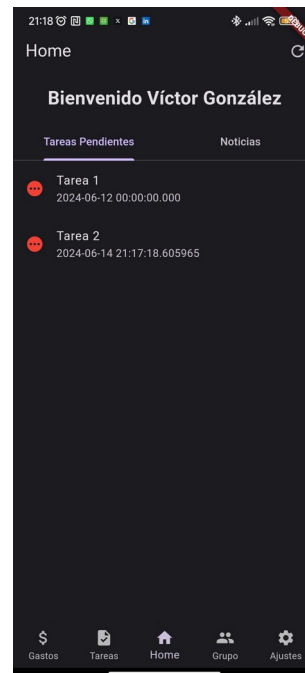
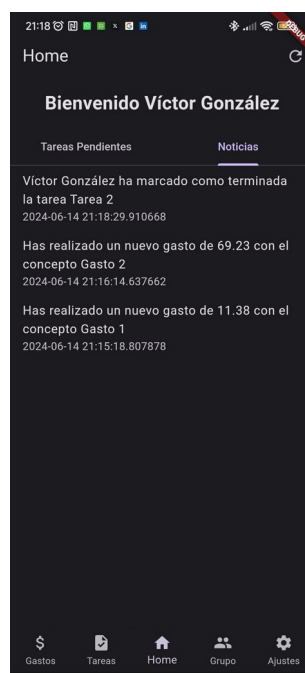


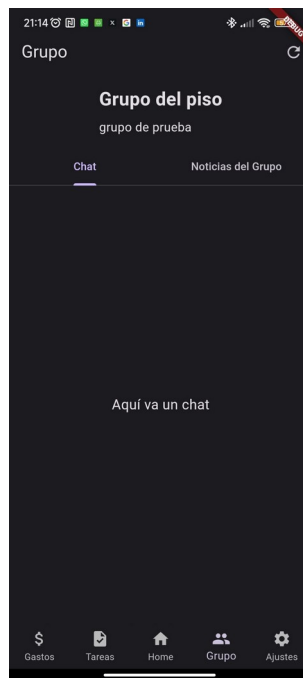
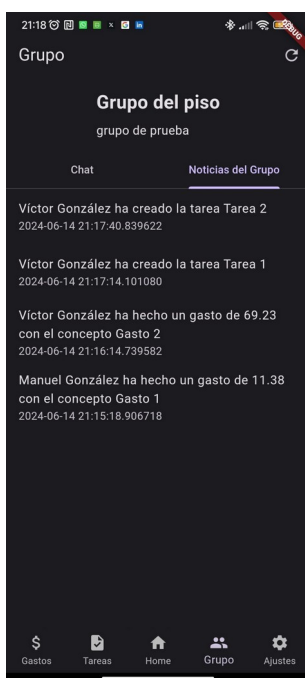
La otra opción que tenemos es rellenar el ID de un grupo ya creado y de igual forma nos llevara a la pantalla principal.



Pantalla principal y pantalla de grupo

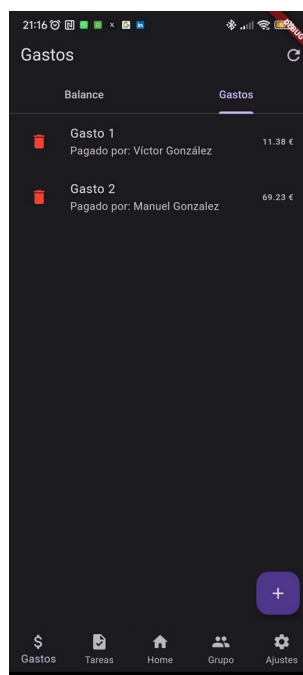
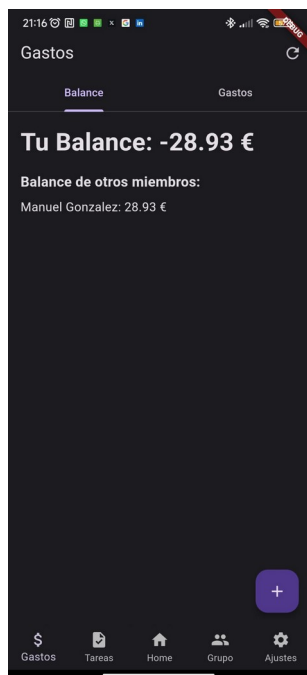
Desde la pantalla principal podremos ver las tareas retrasadas así como las noticias con las que tenga que ver nuestro usuario. De una forma parecida en la pantalla de grupo tenemos un chat de grupo (no esta implementado) y las todas las noticias del grupo.



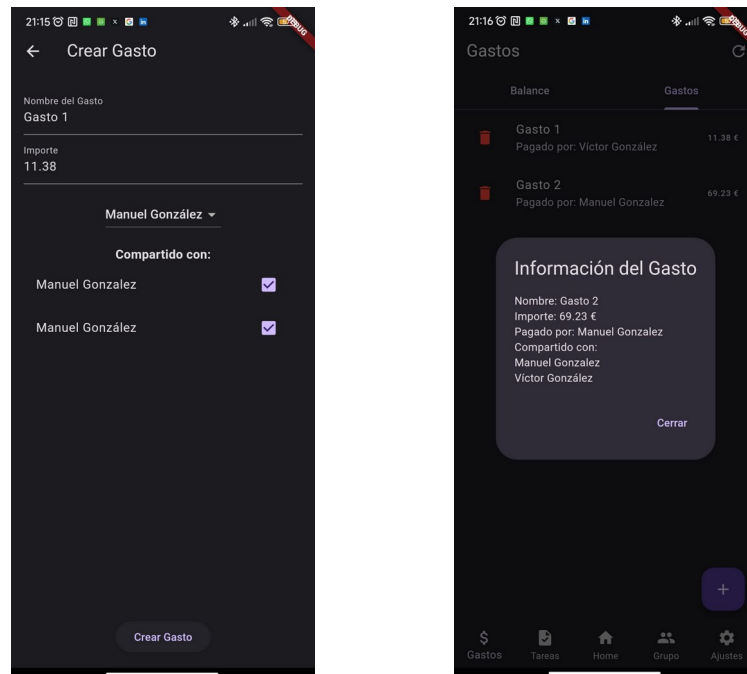


Pantalla de Gastos y de crear gastos

La pantalla de gastos se divide en dos, una con el balance de los miembros del grupo y otra con los gastos.

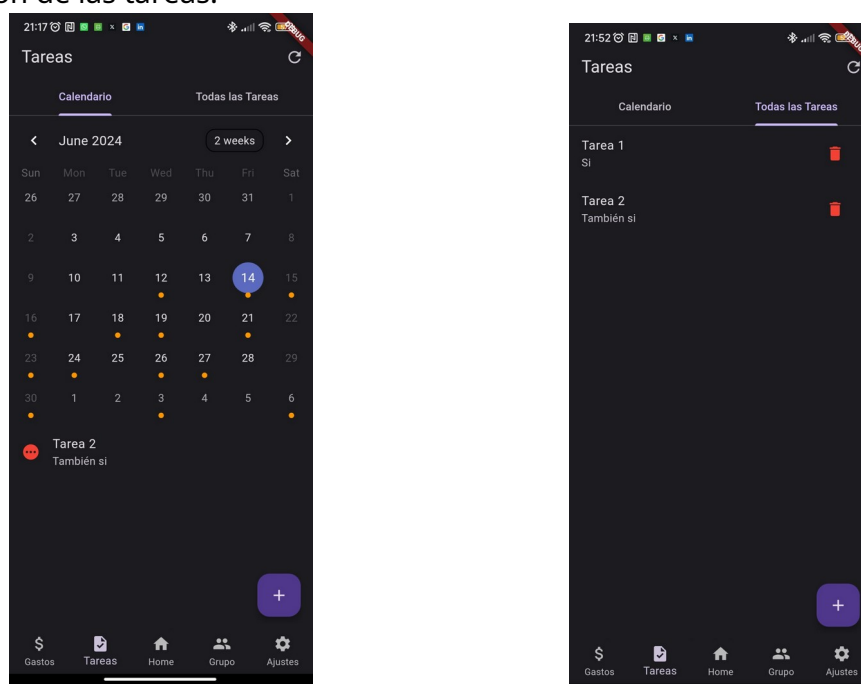


Después tendremos la pantalla de crear gasto donde podremos seleccionar quien ha pagado el gasto, el importe, un nombre y quienes participan en el gasto. Después si hacemos clic en un gasto desde la lista de gastos se nos mostrara la información del mismo.

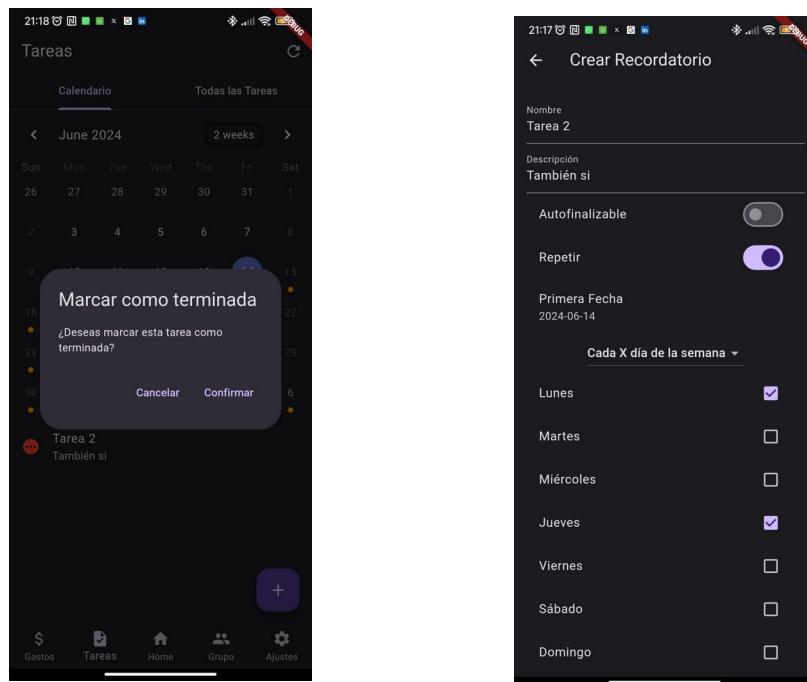


Pantalla de Tareas y de crear tareas.

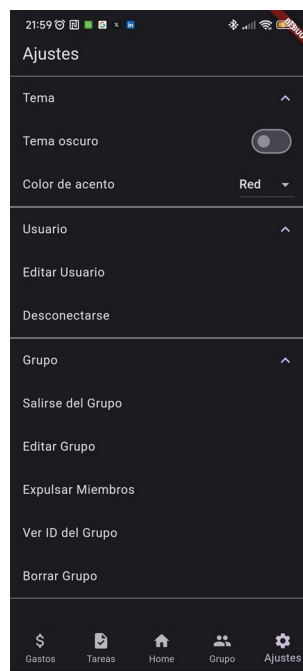
De una forma muy similar a Gastos Tareas tiene dos pestañas una con un calendario y una lista de tareas con las tareas del día seleccionado en el calendario. Y otra con la información de las tareas.



También podremos hacer clic en la lista de tareas del calendario para marcar una tarea como terminada y desde la lista con la información de las tareas si hacemos clic editaremos la tarea. Después si hacemos clic en el botón flotante podremos añadir una tarea , podremos selecciona si se repite o no, si es autofinalizable, nombre descripción y el tipo de repetición entre cada X días, cada día X de la semana o cada día X del mes.



Después tendremos la pantalla de ajustes desde donde podremos editar el grupo, expulsar miembros, salir del grupo, editar nuestro perfil o desconectarse entre otras opciones.



7.2 Manual de instalación

Requisitos Previos

Antes de instalar la aplicación NestApp, asegúrate de cumplir con los siguientes requisitos:

- Un dispositivo Android con al menos 100 MB de espacio libre.
- Tener el archivo nestApp.apk.
- Habilitar la opción para instalar aplicaciones de fuentes desconocidas en tu dispositivo Android.

Habilitar Instalación desde Fuentes Desconocidas

Para instalar aplicaciones que no se encuentran en Google Play Store, necesitas habilitar la instalación desde fuentes desconocidas en tu dispositivo. Sigue estos pasos:

1. Abrir Configuración:

Ve a la aplicación "Configuración" en tu dispositivo Android.

2. Seguridad:

Busca y selecciona la opción "Seguridad" o "Privacidad" (esto puede variar según la versión de Android y la marca del dispositivo).

3. Fuentes Desconocidas:

Habilita la opción "Permitir instalación de aplicaciones de fuentes desconocidas". Esto permitirá la instalación de aplicaciones que no están en la Google Play Store.

Instalar la Aplicación

1. Abrir el archivo .apk:

Abre el archivo .apk desde la ubicación del archivo.

2. Confirmar la instalación:

Aparecerá una pantalla solicitando permiso para instalar la aplicación. Revisa los permisos y presiona el botón "Instalar".

3. Esperar a que finalice la instalación:

El proceso de instalación puede tardar unos momentos. Una vez que la instalación haya finalizado, verás un mensaje de confirmación.

4. Abrir la aplicación:

Presiona "Abrir" para iniciar la aplicación NestApp. También puedes encontrar la aplicación en la lista de aplicaciones instaladas en tu dispositivo.

8 Conclusiones y posibles ampliaciones

Conclusiones

El desarrollo de NestApp ha sido un proyecto enriquecedor que ha permitido la implementación y profundización en diversas tecnologías y conceptos relacionados con el desarrollo de aplicaciones móviles. A lo largo del proceso, se han logrado los objetivos principales y secundarios establecidos inicialmente, proporcionando una aplicación funcional que facilita la gestión de tareas y gastos entre compañeros de piso.

Aspectos Destacables:

1. **Uso de Firebase:** La integración con Firebase ha permitido el manejo de autenticación y base de datos en tiempo real, proporcionando una experiencia de usuario fluida y consistente. Firebase Firestore ha sido crucial para la sincronización de datos entre dispositivos.
2. **Flutter y Dart:** La elección de Flutter y Dart ha demostrado ser una decisión acertada debido a su capacidad para desarrollar interfaces de usuario atractivas y altamente responsivas. La comunidad y la documentación de Flutter han facilitado la resolución de problemas y la implementación de funcionalidades complejas.
3. **Gestión de Tareas y Gastos:** La implementación de la lógica para la gestión de tareas recurrentes y la división de gastos ha cumplido con las expectativas, ofreciendo una herramienta útil para los usuarios finales.
4. **Interfaz de Usuario:** Aunque el diseño de la interfaz de usuario fue un objetivo de baja prioridad, se ha logrado un diseño intuitivo y agradable, mejorando la experiencia del usuario.

Lecciones Aprendidas:

1. **Planificación y Diseño:** La planificación y diseño inicial del proyecto fueron fundamentales para el éxito del desarrollo. Sin embargo, se ha aprendido la importancia de ser flexible y adaptable a los cambios y mejoras que surgieron durante el proceso.
2. **Pruebas y Depuración:** Las pruebas continuas y la depuración fueron esenciales para asegurar la calidad y funcionalidad de la aplicación. La implementación de pruebas unitarias, de integración y de interfaz de usuario ayudó a identificar y corregir errores de manera eficiente.

Posibles Ampliaciones

Aunque NestApp ha alcanzado un estado funcional y útil, existen varias áreas en las que se podría ampliar y mejorar la aplicación:

1. Chat en Tiempo Real:

- Implementar un sistema de chat en tiempo real para mejorar la comunicación entre los miembros del grupo. Utilizar Firebase Realtime Database o Firestore para gestionar los mensajes.

2. Notificaciones Push:

- Integrar notificaciones push para alertar a los usuarios sobre nuevas tareas, gastos y otros eventos importantes en la aplicación. Firebase Cloud Messaging podría ser una herramienta útil para esta funcionalidad.

3. Soporte Multiplataforma:

- Extender la compatibilidad de la aplicación a otras plataformas, como iOS, aprovechando las capacidades multiplataforma de Flutter. Realizar pruebas y ajustes necesarios para asegurar una experiencia consistente en todos los dispositivos.

4. Funcionalidades Adicionales de Tareas:

- Añadir funcionalidades avanzadas de gestión de tareas, como la asignación de prioridades, la categorización y la creación de sub-tareas. Implementar un sistema de recordatorios más flexible y personalizable.

5. Optimización de Rendimiento:

- Realizar optimizaciones adicionales en el rendimiento de la aplicación, especialmente en dispositivos con recursos limitados. Investigar y aplicar técnicas de optimización de Flutter y Firebase para mejorar la eficiencia general.

Conclusión Final

NestApp ha demostrado ser una solución efectiva para la gestión de tareas y gastos compartidos, proporcionando a los usuarios una herramienta práctica y fácil de usar. A través de la implementación de tecnologías modernas y prácticas de desarrollo ágiles, se ha logrado un producto de calidad que puede seguir evolucionando y mejorando con futuras ampliaciones y optimizaciones.

9 Bibliografía

Realmente este proyecto no ha requerido bibliografía, ya que los recursos que he utilizado han sido:

Documentación de Flutter: <https://docs.flutter.dev/>

Documentación de Firebase: <https://firebase.google.com/docs?hl=es>

10 Anexos

Diagrama de actividad al iniciar la aplicación.

