Grado Superior
Desarrollo de aplicaciones multiplataforma

Aplicación para la Gestión de Tareas y Control de Alimentos en el Hogar

TaskTraker

Autor:
Borja Merchán Mckenna
26/11/2024

Índice

Índice de figuras

1. Metodología, Leyes y Normativas

1.1. Metodología Realizada del Proyecto

Para llevar a cabo el proyecto, se ha empleado una combinación de diferentes técnicas y herramientas adecuadas a las tareas que lo exigían. Se ha utilizado un enfoque iterativo e incremental, lo que permitió adaptarse a los cambios que surgieron a lo largo del desarrollo. Además, se implementó una metodología basada en el uso del diagrama de Gantt, lo cual facilitó la planificación y organización de las tareas de manera flexible.

1.2. Leyes y Normativas

Dado que el proyecto incluye funcionalidades relacionadas con la gestión de usuarios y datos, es fundamental garantizar el cumplimiento de las normativas vigentes en materia de protección de datos y privacidad.

Reglamento General de Protección de Datos (GDPR)

- Consentimiento explícito: Obtener el consentimiento explícito de los usuarios para el tratamiento de sus datos personales, especialmente en funcionalidades como la gestión de tareas, grupos de hogar y ajustes personalizados.
- Derechos de los usuarios: Respetar los derechos de acceso, rectificación, supresión, limitación del tratamiento, portabilidad de datos y oposición en cada interacción de los usuarios con la aplicación.
- Notificación de violaciones de seguridad: En caso de incidentes de seguridad que comprometan los datos personales de los usuarios, garantizar la notificación a los usuarios afectados y a las autoridades competentes de forma inmediata y conforme a los plazos legales.

Ley Orgánica de Protección de Datos Personales y Garantía de los Derechos Digitales (LOPDGDD)

- Como normativa complementaria al GDPR en España, se han implementado medidas específicas para el tratamiento de datos personales almacenados y sincronizados en Firebase, asegurando que su gestión respete los principios de confidencialidad, seguridad y minimización de datos.

2. Introducción

En la actualidad, la gestión eficiente del tiempo y las tareas se ha vuelto fundamental en la vida diaria. "Tasktraker.^{es} una aplicación multiplataforma desarrollada en Flutter que permite creación de usuarios y de grupos de hogares, a los usuarios gestionar sus tareas diarias y llevar un control de los productos en su nevera. Utilizando Firebase como base de datos y sistema de autenticación, esta aplicación busca facilitar la organización personal a través de una interfaz intuitiva y funcionalidades prácticas.

2.1. Planteamiento del Problema

La idea de desarrollar esta aplicación surgió de una necesidad personal: organizar mis tareas diarias de manera más eficiente mediante una lista de pendientes. Además, al vivir con mis hermanos, se presentó la necesidad de llevar un control compartido de los alimentos disponibles en la nevera para facilitar la planificación de las compras y evitar desperdicios. Por ello, decidí crear una solución que integrara ambas funcionalidades, ayudando no solo en la gestión de tareas, sino también en el control del inventario del hogar.

2.2. Objetivos del Proyecto

2.2.1. Objetivo General

Desarrollar una aplicación móvil que permita a los usuarios del mismo domicilio, gestionar tanto sus tareas diarias como el control del inventario de alimentos en sus hogares, facilitando la organización ,la reducción del desperdicio de alimentos y de tiempo.

2.2.2. Objetivos Específicos

- Registrar usurios
- Crear un sistema de gestión de tareas que permita añadir, modificar y eliminar tareas.
- Implementar una funcionalidad para registrar y controlar los alimentos disponibles en la nevera.
- Implementar la autenticación de usuarios para asegurar que cada perfil tenga un control personalizado de sus tareas y alimentos.
- Utilizar **Firebase** y **Firestore** para gestionar la base de datos en tiempo real, garantizando la sincronización de los datos de los usuarios.

3. Tecnologías, recursos y medios utilizados

3.1. Tenologias Ultilizadas:

3.1.1. Editores de Codigo

Figura 1: Visual Studio Code como IDE para el desarrollo.

He utilizado **Visual Studio Code** como entorno de desarrollo integrado (IDE). VS Code es una herramienta ligera, potente y ampliamente utilizada para programar en diversos lenguajes.

Extensiones Utilizadas.

Extensiones esenciales para Flutter

1. Flutter
   - Esta extensión incluye soporte para trabajar con Flutter, depuración, y generación de proyectos.
2. Dart
   - Soporte esencial para el lenguaje Dart, necesario para escribir y depurar código en Flutter.

Extensiones de productividad

1. Awesome Flutter Snippets
   - Ofrece fragmentos de código predefinidos para escribir código más rápido en Flutter y Dart.
2. Bracket Pair Colorizer 2
   - Resalta pares de paréntesis, corchetes y llaves con diferentes colores, facilitando la lectura del código.

3. Error Lens

    - Muestra errores y advertencias directamente en el editor, junto al código, en lugar de depender únicamente del panel de problemas.

4. Flutter Tree

    - Ayuda a visualizar y navegar fácilmente por el árbol de widgets.

Figura 2: Flutlab, editor web utilizado para desarrollo rápido con Flutter.

flutlab.io: Es un editor de código web, Es ligero y potente que proporciona herramientas útiles para la programación en Dart y Flutter, como la integración de un emulador para poder compilar de forma rapida,

3.1.2. Diseño

Figura 3: Figma como herramienta para el diseño gráfico.

Para el diseño gráfico hemos utilizado **Figma**, una herramienta de diseño de interfaz de usuario (UI) y experiencia de usuario (UX) basada en la nube. Permite a los diseñadores de pro- ductos crear, probar y colaborar en diseños de manera efectiva.

3.1.3. Documentación

Latex

Figura 4: LaTeX como herramienta de documentación.

LaTeX se utilizó para crear la documentación del proyecto de manera estructurada y profesional, asegurando una presentación clara y adecuada del contenido.

Listado de dependencias utilizadas en el documento LaTeX

A continuación, se mostrara la tabla de las dependencias

Paquete	Propósito
\usepackage[utf8]{inputenc}	Permite el uso de caracteres UTF-8, esencial para caracteres especiales como tildes y eñes.
\usepackage{graphicx}	Proporciona soporte para incluir gráficos o imágenes (\includegraphics).
\usepackage{fancyhdr}	Permite personalizar encabezados y pies de página (\pagestyle{fancy}).
\usepackage{ragged2e}	Aporta comandos para alineación de texto (\justifying).
\usepackage{multirow}	Permite combinar filas en tablas (\multirow).
\usepackage[hidelinks]{hyperref}	Añade soporte para enlaces clicables en el PDF sin resaltarlos visualmente.
\usepackage[table,xcdraw]{xcolor}	Añade soporte para colores en texto y tablas (\definecolor, tablas con colores personalizados).
\usepackage{ulem}	Proporciona herramientas para subrayar, tachar y estilos similares.
\usepackage{listings}	Permite incluir y resaltar código fuente en diferentes lenguajes de programación.
\usepackage[spanish]{babel}	Configura el idioma del documento en español, adaptando nombres predeterminados como “Índice” o “Figura”.
\usepackage{xcolor}	Facilita la definición y uso de colores (\definecolor, \color).
\usepackage{bbding}	Proporciona símbolos adicionales, como marcas de verificación y cruces (\CheckedBox, \XBox).
\usepackage{pifont}	Similar a bbding, añade más símbolos útiles.

\usepackage{wasysym}	Incluye símbolos adicionales (caras, objetos astronómicos, etc.).
\usepackage{amssymb}	Ofrece acceso a símbolos matemáticos adicionales.
\usepackage{pgfgantt}	Herramienta para crear diagramas de Gantt.
\usepackage{float}	Controla la posición de objetos flotantes como tablas y figuras ([H]).
\usepackage{pdflscape}	Permite girar páginas enteras a orientación horizontal.
\usepackage{longtable}	Facilita la creación de tablas largas que se extienden en varias páginas.
\usepackage{geometry}	Personaliza los márgenes del documento.
\usepackage{translator}	Proporciona soporte adicional para traducciones automáticas en el texto.

Figura 5: PowerPoint utilizado para la presentación del proyecto.

Para la presentación del proyecto TaskTraker, he utilizado **PowerPoint** debido a su facilidad.

3.1.4. Control De Versiones

Figura 6: GitLab para el control de versiones.

Para el control de versiones he utilizado GitLab, una plataforma que permite gestionar repositorios de código de manera eficiente

GitLab es una herramienta esencial para el desarrollo de software, ya que permite mantener un historial detallado de los cambios realizados en el proyecto

- Recuperar versiones anteriores del código en caso de errores.
- Rastrear la evolución del proyecto a lo largo del tiempo.
- Gestionar múltiples ramas para trabajar en nuevas funcionalidades sin afectar la versión principal.

Explicación de ramas

- **Master**
  - Utilizada como la versión más reciente, estable y funcional del proyecto.
- **Dev**
  - Utilizada como la versión de pruebas del proyecto.
- **Versión 1**
  - En esta versión, se implementó la lógica de programación de las funcionalidades principales de la aplicación, integrando las siguientes características:
    - **SplashScreen:** Pantalla inicial que muestra el logotipo mientras la aplicación se carga.

- **Welcome:** Pantalla de bienvenida con información básica sobre la aplicación.
- **Login:** Pantalla para que los usuarios inicien sesión.
- **Register:** Pantalla para el registro de nuevos usuarios.
- **GrupoScreen:** Pantalla para la creación y gestión de grupos de hogar.
- **Home:** Pantalla principal con un menú de navegación para elegir entre las secciones To-Do, Nevera y Ajustes.
- **To-Do:** Pantalla para la gestión de tareas pendientes.
- **Nevera:** Pantalla para el control del inventario de alimentos.
- **Ajustes:** Pantalla para configurar las preferencias del usuario.

- **Versión 2**

  - En esta versión, se implementó la integración con la base de datos utilizando Firebase, lo que permitió almacenar y sincronizar datos de manera eficiente en tiempo real. Las mejoras y funcionalidades añadidas fueron:

    - Configuración e integración de **Firebase Authentication** para el registro e inicio de sesión de los usuarios.
    - Implementación de **Firebase Firestore** para:
      - Guardar los datos relacionados con los grupos de hogar en **GrupoScreen**, permitiendo:
      - Crear un grupo de hogar .
      - Permitir a otros usuarios unirse al grupo utilizando el nombre del domicilio y la contraseña.
      - Almacenar y sincronizar datos de tareas en la pantalla **To-Do**.
      - Gestionar la información de productos en la sección **Nevera**.
      - Guardar los datos relacionados con los grupos de hogar creados en **GrupoScreen**.

- **Versión 3**

  - En esta versión, se centró en implementar mejoras generales en la aplicación y en integrar nuevas funcionalidades para mejorar la experiencia del usuario. Las principales actualizaciones fueron:
    - Implementación de mejoras en la interfaz y experiencia de usuario:
    - que Sí el usuario haya iniciado sesión anteriormente se queda guardado
    - Posibilidad de mostrar y ocultar contraseñas en las pantallas de **Login** y **Register**.
    - Optimización del flujo en **GrupoScreen**, redirigiendo automáticamente al **Home** si el usuario ya pertenece a un grupo de hogar.
  - Integración de la API de Mercadona en la sección **Nevera**:
    - Búsqueda de productos a través de la API para facilitar la incorporación al inventario.
  - Optimización del rendimiento general de la aplicación y corrección de errores menores detectados en las versiones anteriores.

3.1.5. Framework y Lenguaje

Figura 7: Flutter y Dart utilizados para el desarrollo.

He decidido usar Flutter como framework y Dart como lenguaje de programación para el desarrollo de mi aplicación, ya que ofrecen una combinación ideal de herramientas y características que se adaptan perfectamente a los objetivos del proyecto. Entre las razones principales para esta elección destacan su capacidad para desarrollar aplicaciones multiplataforma, su rendimiento nativo y la flexibilidad para crear interfaces modernas y personalizables.

Por qué Flutter:

- Multiplataforma:

  Flutter permite desarrollar una sola base de código que funciona tanto en Android como en iOS, reduciendo significativamente el tiempo y los recursos necesarios para el desarrollo.

- Rendimiento nativo:

  Gracias a su motor gráfico personalizado (Skia), Flutter ofrece un rendimiento similar al de aplicaciones nativas, con animaciones fluidas y tiempos de respuesta rápidos.

- Hot Reload:

  Esta funcionalidad permite realizar cambios en el código0 y ver los resultados al instante, acelerando el proceso de desarrollo y depuración.

Por qué Dart:

- Compilación a código nativo:

  Dart compila directamente a código nativo, optimizando el rendimiento de la aplicación y evitando las limitaciones de los intérpretes.

- Integración con Flutter:

  Dart está diseñado específicamente para Flutter, proporcionando características como programación reactiva, manejo eficiente de estado y soporte para operaciones asíncronas.

- Curva de aprendizaje suave:

  Su sintaxis moderna, similar a lenguajes como JavaScript y Java, facilita el aprendizaje rápido y la implementación eficiente de funcionalidades.

3.1.6. Base De Datos

Figura 8: Firebase utilizado para la base de datos y autenticación.

Firebase Para la gestión de datos hemos utilizado exclusivamente Firebase, es una plataforma de almacenamiento y gestión de datos enfocada al desarrollo de aplicaciones móviles y web proporcionada por Google. Nosotros hemos utilizado dos servicios de Firebase y los hemos monotorizado desde Firebase Console porque Firebase Storage tienes que usar el plan de pago **Plan Blaze**:

- **Firebase Authentication.**
  Este servicio facilita la implementación de un sistema seguro de registro e inicio de sesión. Proporciona una variedad de métodos de autenticación, incluidos el correo electrónico y las redes sociales, lo que garantiza la seguridad de los datos de los usuarios y una experiencia de usuario fluida.

- **Firebase Firestore.**
  Firestore es una base de datos NoSQL que permite el almacenamiento y la sincronización de datos en tiempo real. Su integración con Flutter facilita el desarrollo de aplicaciones que requieren un backend robusto y confiable.

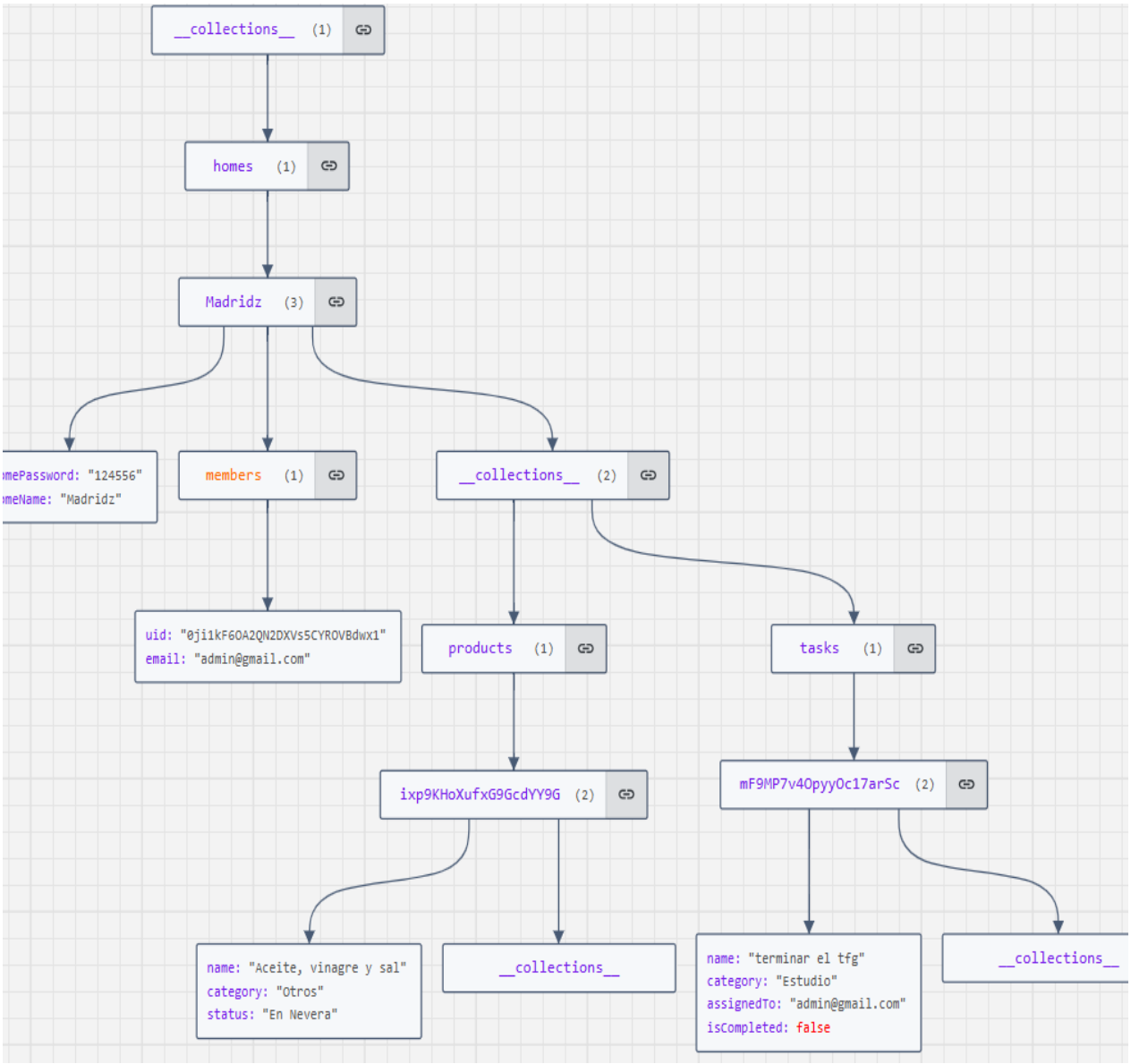

Figura 9: Estructura de datos visualizada en JSON.

4. Planificación Completa del Proyecto

4.1. Planteamiento Principal

Clave	Funcionalidades
	Inicio de la Aplicación
F1	Nombre de la aplicación
F2	Logo de la aplicación
F3	Investigación de herramientas
	Diseño de Aplicación
F4	Diagrama de Gantt
F5	Diagrama de flujo
F6	Diseño en el front
	Desarrollo
F7	Desarrollo de usuarios
F8	Desarrollo de grupo de hogar
F9	Desarrollo del to-do
F10	Desarrollo de gestión de nevera
F11	Desarrollo de ajustes
	Implementación en Firebase
F12	Implementación de Firebase en usuarios
F13	Implementación de Firebase en grupo de hogar
F14	Implementación de Firebase en el to-do
F15	Implementación de Firebase en la gestión de nevera
F16	Implementación de Firebase en ajustes
	Pruebas y Seguridad
F17	Implementación de mejoras
F18	Comprobación del funcionamiento en Android e iOS
	Documentación
F19	Documentación en PDF
F20	Presentación en PowerPoint

Cuadro 2: Tabla de Funcionalidades

4.1.1. Inicio de la Aplicación

- **F1: Nombre de la aplicación** Determinar y definir un nombre distintivo para la aplicación que sea memorable y representativo de su función principal.
- **F2: Logo de la aplicación** Diseñar un logotipo que identifique visualmente la aplicación, alineado con la temática y estilo de la interfaz.
- **F3: Investigación de herramientas** Explorar y seleccionar herramientas de desarrollo, lenguajes, diseño y bases de datos que mejor se adapten al proyecto.

4.1.2. Diseño de la Aplicación

- **F4: Diagrama de Gantt** Crear un cronograma que detalle las fases del proyecto, incluyendo las tareas y sus tiempos de ejecución.
- **F5: Diagrama de flujo** Diseñar un esquema que muestre el flujo lógico y funcional de la aplicación, facilitando la comprensión de su estructura.
- **F6: Diseño en el front** Crear los prototipos o bocetos de las pantallas de la aplicación, definiendo la estética y experiencia de usuario (UX/UI).

4.1.3. Desarrollo

- **F7: Desarrollo de usuarios** Programar la funcionalidad que permita gestionar el registro, inicio de sesión y perfiles de usuario.
- **F8: Desarrollo de grupo de hogar** Crear la función que gestione la creación y administración de grupos, vinculados a hogares o grupos familiares.
- **F9: Desarrollo del to-do** Implementar una funcionalidad para gestionar listas de tareas pendientes (to-do), permitiendo agregar, editar y completar tareas.
- **F10: Desarrollo de gestión de nevera** Programar una herramienta que permita al usuario registrar y organizar los productos almacenados en su nevera, utilizando un enfoque visual.
- **F11: Desarrollo de ajustes** Implementar una sección donde los usuarios puedan personalizar la configuración de la aplicación según sus preferencias.

4.1.4. Implementación en Firebase

- **F12: Implementación de Firebase en usuarios** Integrar Firebase para almacenar y sincronizar datos de los usuarios, como credenciales y perfiles.
- **F13: Implementación de Firebase en grupo de hogar** Configurar Firebase para gestionar la información y datos de los grupos de hogar, permitiendo colaboración entre usuarios.
- **F14: Implementación de Firebase en el to-do** Usar Firebase para guardar y sincronizar las listas de tareas en tiempo real.
- **F15: Implementación de Firebase en la gestión de nevera** Aplicar Firebase para registrar y sincronizar la información sobre los productos disponibles en la nevera.

- **F16: Implementación de Firebase en ajustes** Sincronizar la configuración personalizada de los usuarios a través de Firebase para que se mantenga en todos los dispositivos.

4.1.5. Pruebas y Seguridad

- **F17: Implementación de mejoras** Revisar y ajustar las funcionalidades para optimizar el rendimiento y corregir posibles errores o deficiencias.
- **F18: Comprobación del funcionamiento en Android e iOS** Realizar pruebas exhaustivas para verificar que la aplicación funcione correctamente en ambas plataformas, detectando errores de compatibilidad.

4.1.6. Documentación

- **F19: Documentación en PDF** Generar un documento formal que explique el desarrollo de la aplicación, incluyendo análisis, diseño, implementación y pruebas.
- **F20: Presentación en PowerPoint** Crear una presentación visual para exponer el proyecto, destacando sus funcionalidades y características principales.

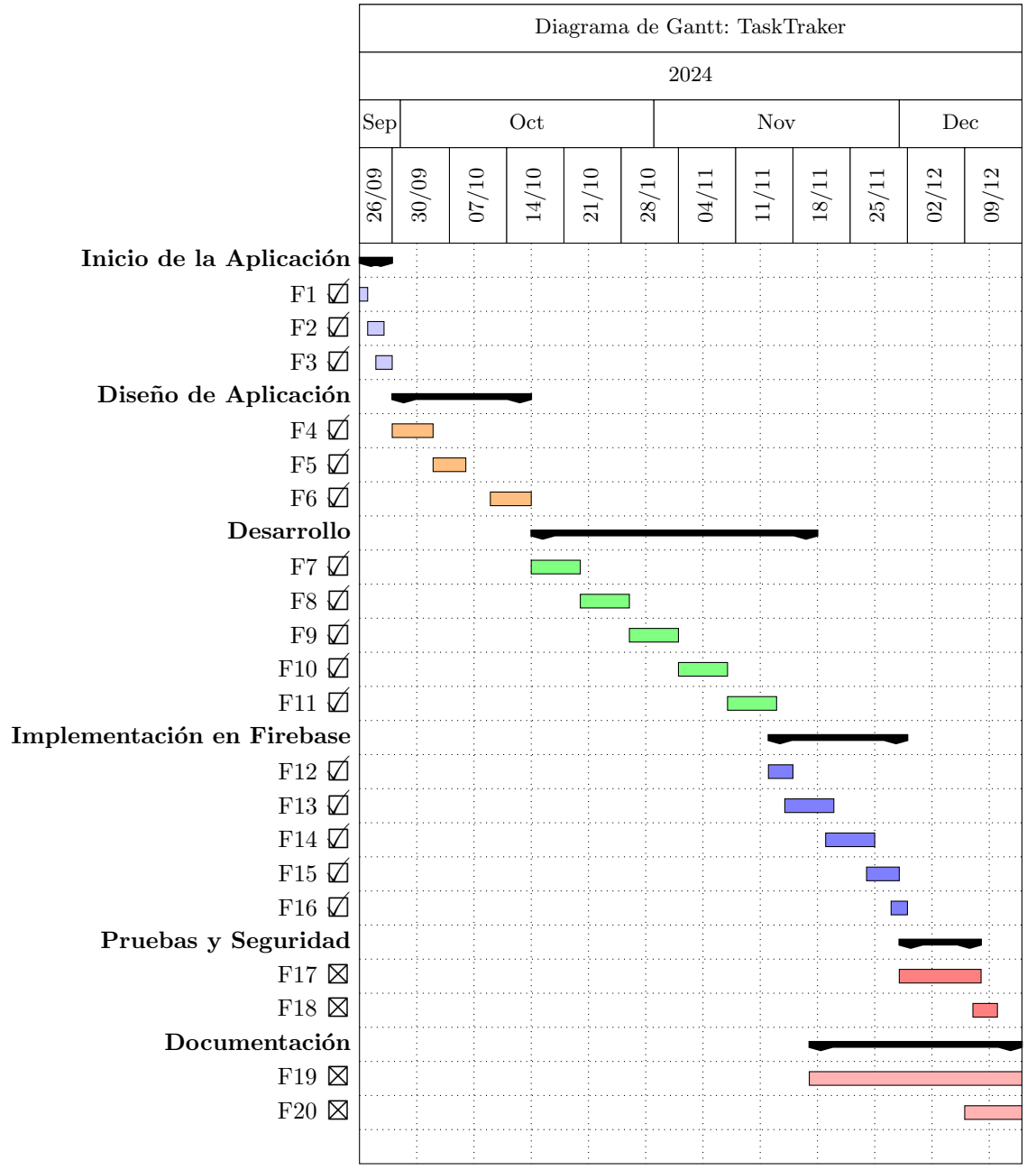

Figura 10: Diagrama de Gantt de **TaskTraker**

4.2. Mejoras o Ideas para Implementar en un Futuro

Pantalla/Actividad	Mejoras o Ideas Futuras	Estado
LoginScreen	- Opción de mostrar contraseña. - Inicio de sesión con Gmail. - Deshabilitar botones si no hay conexión a internet.	En planificación
Register	- Opción de mostrar contraseña. - Registro con Gmail. - Deshabilitar botones si no hay conexión a internet.	En planificación
GrupoScreen	- Detectar si el usuario ya tiene un domicilio registrado y, de ser así, redirigir directamente al *Home Activity*.	Pendiente
To-Do	- Actualizar las tareas al arrastrar hacia abajo. - Añadir la posibilidad de incluir una descripción al presionar en el medio de la tarea.	En planificación
Gestión de Nevera	- Agregar productos mediante una opción que utilice la API de Mercadona. - Añadir opciones de editar y eliminar productos.	Pendiente
Gestión de Ajustes	- Permitir al usuario cargar una imagen como foto de perfil al hacer clic en su foto de usuario.	Pendiente

Cuadro 3: Mejoras o ideas para implementar en el futuro

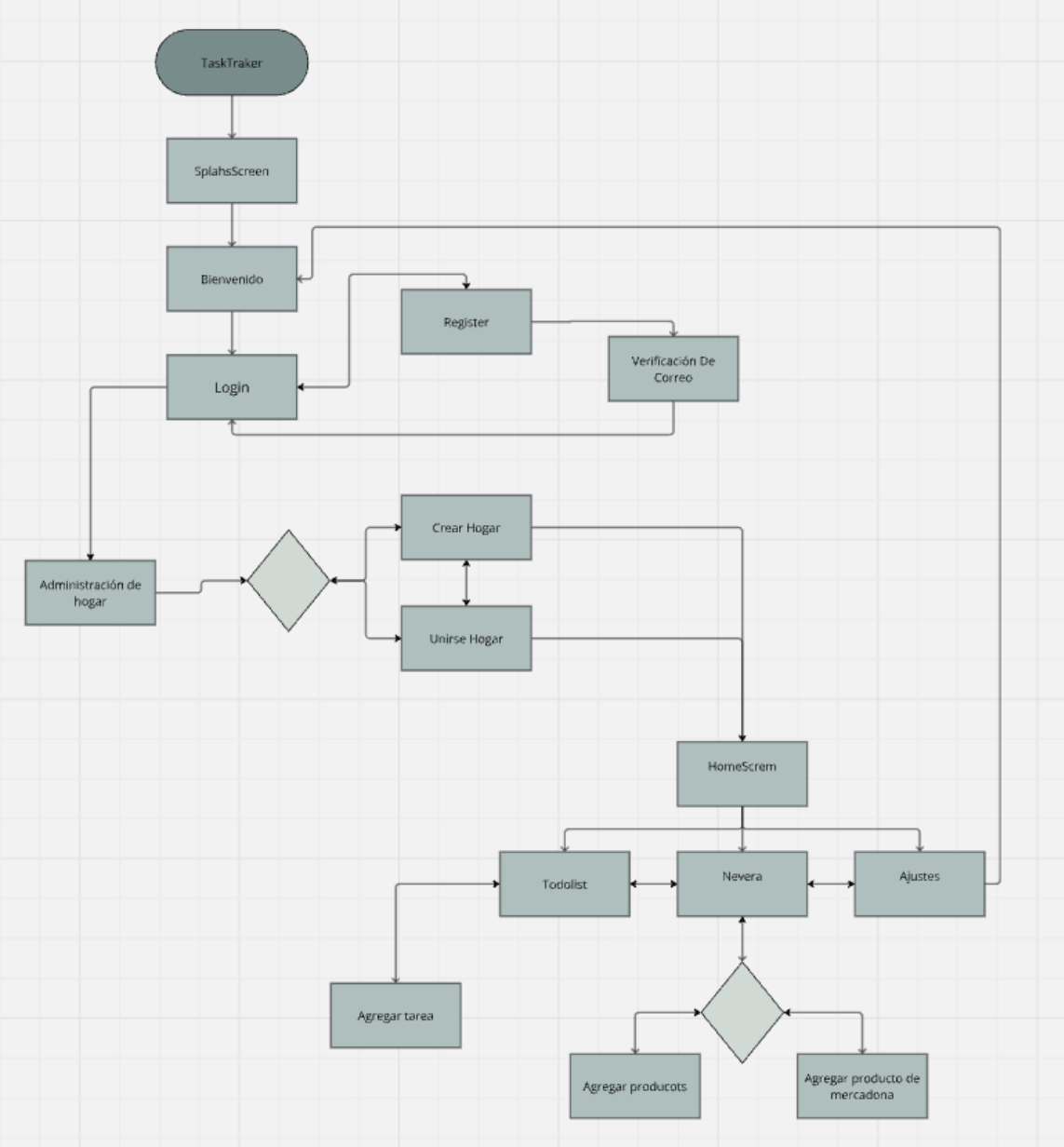

Figura 11: Diagrama De Flujo de (`TaskTraker`)

5. Análisis y Diseño

5.1. Navegación por pantallas

A continuación, iremos viendo la navegación y descripción de cada pantalla para obtener una visión general de la aplicación.

5.1.1. Archivo Principal (`main.dart`)

- **Descripción**: Este archivo actúa como punto de entrada para la aplicación. Inicializa Firebase, configura los temas visuales y gestiona la verificación del estado de autenticación del usuario para redirigirlo a la pantalla correspondiente.
- **Requisitos**:
  - Firebase debe estar configurado e inicializado correctamente en el proyecto.
  - Las pantallas de bienvenida (`WelcomeScreen`) y principal (`HomeScreen`) deben estar implementadas.
  - La clase `DefaultFirebaseOptions` debe estar configurada con las credenciales de Firebase.
- **Componentes principales**:
  - **Clase** `MyApp`:
    - Define el tema global de la aplicación.
    - Especifica la pantalla inicial basada en el estado de autenticación mediante el widget `AuthCheck`.
  - **Widget** `AuthCheck`:
    - Escucha los cambios en el estado de autenticación del usuario a través de `FirebaseAuth`.
    - Redirige al usuario a:
      - `HomeScreen` si el usuario está autenticado.
      - `WelcomeScreen` si el usuario no está autenticado.
    - Muestra un indicador de carga mientras se verifica el estado de autenticación.
- **Flujo de la Aplicación**:
  - La aplicación se inicializa mediante `Firebase.initializeApp`.
  - Se verifica el estado de autenticación del usuario:
    - Si el usuario está autenticado, es redirigido a `HomeScreen`.
    - Si el usuario no está autenticado, es redirigido a `WelcomeScreen`.

5.1.2. Pantalla de Arranque (`SplashScreen`)

- **Descripción**: Es la primera pantalla que se muestra cuando se abre la aplicación. Contiene el logotipo de la aplicación.
- **Navegación**: Esta pantalla lleva al usuario, una vez han transcurrido tres segundos, a la pantalla de bienvenida (`WelcomeScreen`).

Figura 12: Pantalla de Arranque (`SplashScreen`).

5.1.3. Pantalla de Bienvenida (`WelcomeScreen`)

- **Descripción**: La pantalla de bienvenida es la primera interacción del usuario después de la pantalla de arranque. Incluye una imagen representativa, un mensaje de bienvenida y una breve descripción de las principales funcionalidades de la aplicación. Además, cuenta con un botón destacado que facilita la navegación hacia la pantalla de inicio de sesión.
- **Navegación**: El flujo de navegación de esta pantalla es claro y sencillo. Al pulsar el botón `Comenzar`, el usuario es redirigido automáticamente a la pantalla de inicio de sesión (`LoginScreen`).
- **Listado de contenidos**:
  - **Imagen representativa**: Una imagen ubicada en la parte superior que refuerza la identidad visual de la aplicación.
  - **Mensaje de bienvenida**: Un texto amigable que recibe al usuario y destaca el nombre de la aplicación.
  - **Descripción de funcionalidades**: Un breve texto que informa al usuario sobre las principales utilidades de la aplicación, como la organización de tareas y la gestión de productos en la nevera.
  - **Botón de acción `Comenzar`**: Un botón claramente visible que permite al usuario avanzar a la pantalla de inicio de sesión (`LoginScreen`).

Figura 13: Pantalla de Bienvenida (`WelcomeScreen`).

5.1.4. Pantalla de Inicio de Sesión (`LoginScreen`)

Descripción: La pantalla de inicio de sesión permite a los usuarios acceder a su cuenta proporcionando un correo electrónico y contraseña previamente registrados. Incluye validaciones de los campos ingresados, un indicador de carga durante el proceso de inicio de sesión y un mensaje de error en caso de credenciales incorrectas. Además, ofrece un enlace para redirigir a la pantalla de registro en caso de no tener cuenta.

Requisitos:

- El usuario debe ingresar un correo electrónico registrado y válido.
- La contraseña debe coincidir con la registrada previamente.
- El correo electrónico debe estar verificado.

Navegación: Una vez completado el inicio de sesión, el usuario es redirigido a la pantalla principal (`GroupScreen`). Si el correo no está verificado, se muestra un mensaje y se redirige nuevamente a la pantalla de inicio de sesión. También ofrece un enlace a la pantalla de registro (`RegistrarScreen`) para los usuarios que no tienen cuenta.

Listado de contenidos:

- **Formulario de inicio de sesión**: Campos para ingresar el correo electrónico y la contraseña, con validaciones estrictas.
- **Botón Ïngresar"**: Permite enviar la información del formulario para autenticar al usuario.
- **Indicador de carga**: Muestra un indicador mientras se procesa el inicio de sesión.
- **Botón Regístrate"**: Un enlace para los usuarios que no tienen cuenta, que los redirige a la pantalla de registro.
- **Mensaje de error**: Muestra mensajes personalizados en caso de credenciales incorrectas o problemas de verificación.

Funciones principales:

- `_signIn()`:
  - Valida los campos del formulario.
  - Autentica al usuario mediante Firebase Authentication.
  - Verifica si el correo electrónico ha sido validado.
  - Redirige al usuario a la pantalla principal (`GroupScreen`).
- `_buildTextField()`:
  - Construye un campo de texto reutilizable con opciones de validación y visibilidad de contraseña.
- `setState()`:
  - Controla cambios en el estado de carga y visibilidad de la contraseña.

- **Navegación por pantalla**

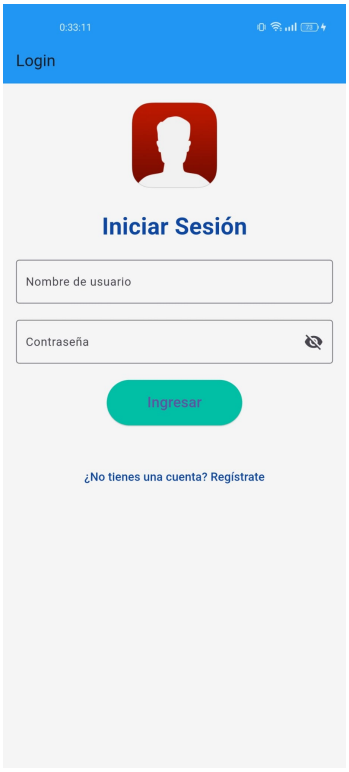

Figura 14: Pantalla de Inicio de Sesión (`LoginScreen`).

- **Gestión de Errores**
  - **Correo electrónico**:
    - El formato debe ser válido (`usuario@dominio.com`).
    - Mensaje de error: `"Por favor, introduce un correo válido."`
  - **Contraseña**:
    - Debe tener al menos 8 caracteres.
    - Debe incluir mayúsculas, minúsculas, números y caracteres especiales.
    - Mensaje de error: `"La contraseña debe cumplir con los requisitos de seguridad."`
  - **Correo electrónico**:
    - Debe tener un formato válido (`usuario@dominio.com`).
    - Mensaje de error: `"Por favor, introduce un correo válido."`
  - **Contraseña**:
    - Debe tener al menos 8 caracteres.

- Debe incluir mayúsculas, minúsculas, números y caracteres especiales.
- Mensaje de error: `"La contraseña debe cumplir con los requisitos de seguridad."`

5.1.5. Pantalla de Registro (`RegistrarScreen`)

- **Descripción**: La pantalla de registro permite a los usuarios crear una nueva cuenta proporcionando su correo electrónico y contraseña. Incluye validaciones de los campos ingresados, instrucciones para crear una contraseña segura y usar un correo electrónico válido, y un mensaje de confirmación tras un registro exitoso. También se envía un enlace de verificación al correo electrónico del usuario.
- **Requisitos**:
  - El usuario debe ingresar un correo electrónico válido.
  - La contraseña debe tener al menos 8 caracteres, incluyendo mayúsculas, minúsculas, números y caracteres especiales.
  - Es necesario verificar la cuenta mediante el enlace enviado al correo electrónico.
- **Navegación**: Una vez completado el registro, el usuario es redirigido a la pantalla de verificación de correo (`EmailVerificationScreen`). Además, ofrece la opción de regresar a la pantalla de inicio de sesión.
- **Listado de contenidos**:
  - **Formulario de registro**: Campos para ingresar el correo electrónico y la contraseña, con validaciones estrictas.
  - **Instrucciones**: Consejos sobre cómo crear una contraseña segura y usar un correo electrónico válido. Estas se muestran de manera opcional mediante un botón de guía.
  - **Botón "Mostrar/Ocultar guía"**: Un botón interactivo que permite al usuario alternar la visibilidad de las instrucciones para el registro.
  - **Botón de acción Registrarse"**: Permite enviar la información del formulario y registrar al usuario.
  - **Indicador de carga**: Muestra un indicador mientras se procesa el registro.
  - **Botón Ïnicia sesión"**: Un enlace para los usuarios que ya tienen cuenta, que los redirige a la pantalla de inicio de sesión.
  - **Enlace de verificación**: Tras el registro exitoso, se envía un correo electrónico con un enlace para verificar la cuenta.
- **Funciones principales**:
  - `_register()`:
    - Valida los campos del formulario.
    - Crea un nuevo usuario con Firebase Authentication.
    - Envía un enlace de verificación al correo electrónico del usuario.
    - Redirige al usuario a la pantalla de verificación de correo.
  - `_buildTextField()`:
    - Construye un campo de texto con personalización de etiqueta, validación y visibilidad de contraseña.

- `setState()`:
  - Controla la visibilidad de la contraseña.
  - Alterna la visibilidad de las instrucciones.
  - Maneja el estado de carga durante el registro.

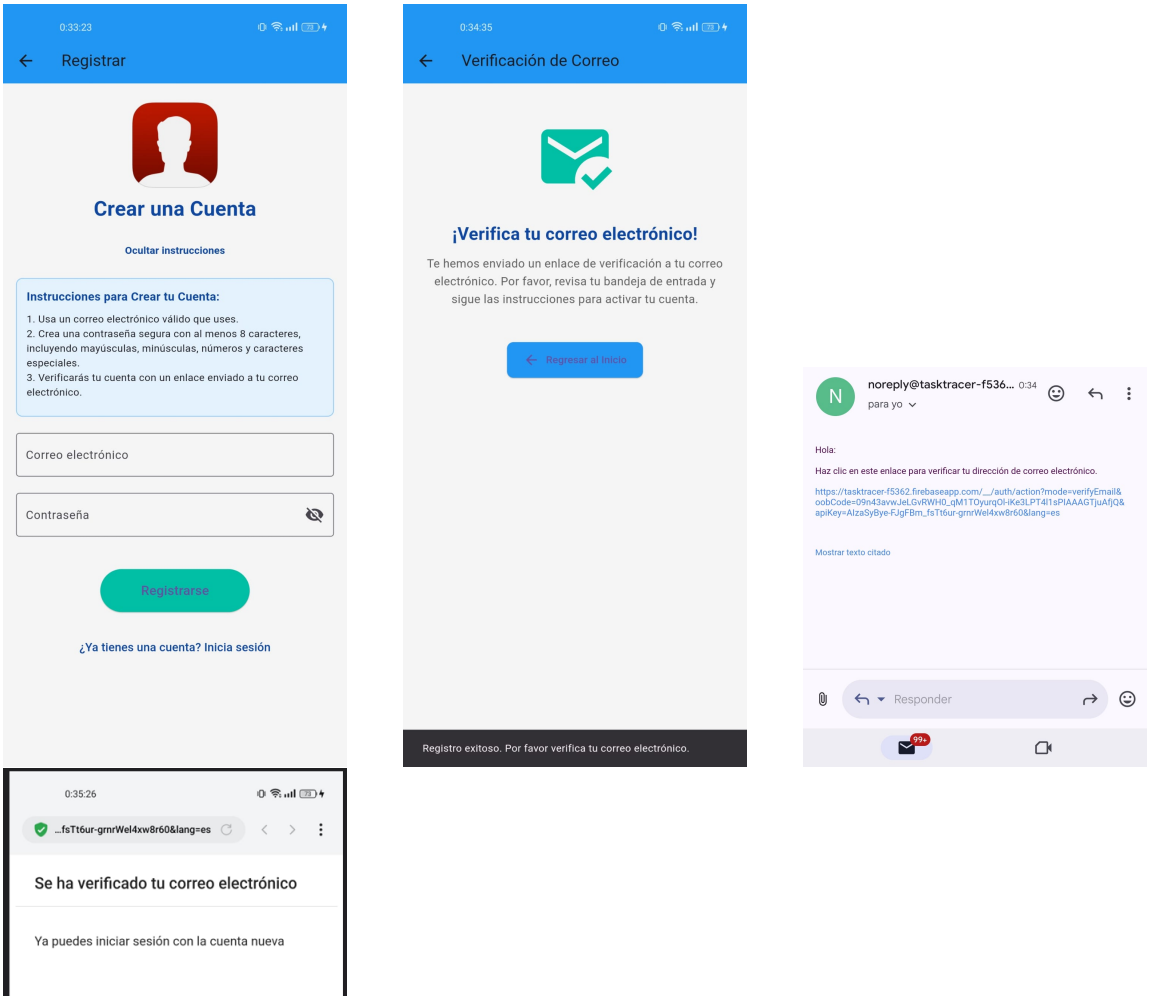

Figura 15: Pantallas de Registro (`RegistrarScreen`).

5.1.6. Pantalla de Grupos de Hogar (`GroupScreen`)

- **Descripción**: La pantalla de grupos de hogar permite a los usuarios gestionar los grupos de los que forman parte, crear nuevos hogares o unirse a un hogar existente. Facilita la conexión de múltiples usuarios a un mismo hogar para compartir y administrar información. Un diseño intuitivo para facilitar la interacción.
- **Requisitos**:
  - El usuario debe estar autenticado en la aplicación.
  - Para crear un hogar, se requiere un nombre único y una contraseña.
  - Para unirse a un hogar, el usuario debe proporcionar el nombre del hogar y la contraseña correspondiente.
- **Navegación**: Tras asociarse a un hogar (ya sea al crear uno o unirse a uno existente), el usuario es redirigido a la pantalla principal (`HomeScreen`).
- **Listado de contenidos**:
  - **Formulario para crear un hogar**: Permite ingresar un nombre y una contraseña para registrar un nuevo hogar.
  - **Formulario para unirse a un hogar**: Permite ingresar el nombre y la contraseña de un hogar existente para unirse a él.
  - **Botón Çrear Hogar"**: Alterna la visibilidad del formulario para registrar un nuevo hogar.
  - **Botón Ünirse a Hogar"**: Alterna la visibilidad del formulario para unirse a un hogar existente.
- **Funciones principales**:
  - `_createHome()`:
    - Crea un nuevo hogar en Firebase Firestore con el nombre y contraseña proporcionados.
    - Añade al usuario autenticado como miembro del hogar creado.
  - `_joinHome()`:
    - Verifica si el hogar y la contraseña proporcionados son válidos.
    - Añade al usuario autenticado como miembro del hogar existente.
  - `_buildHomeForm()`:
    - Genera un formulario dinámico para registrar o unirse a un hogar.
    - Incluye campos de texto con validación y un botón de acción.
  - `setState()`:
    - Controla la visibilidad de los formularios (crear o unirse a un hogar).
    - Alterna la visibilidad de la contraseña en los campos correspondientes.

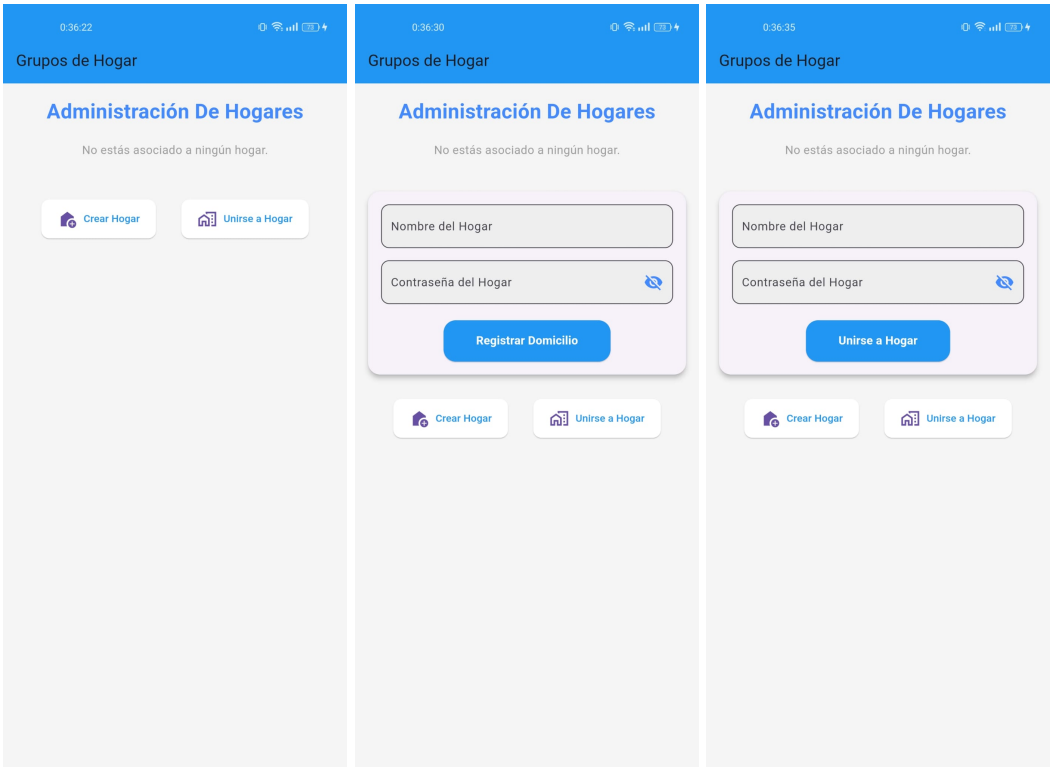

Figura 16: Pantallas de Grupos de Hogar (`GroupScreen`).

5.1.7. Pantalla Principal (HomeScreen)

- **Descripción**: La pantalla principal de la aplicación organiza la navegación entre las funcionalidades clave: la gestión de tareas (`ToDoScreen`), el control de alimentos en la nevera (`NeveraScreen`), y los ajustes de la aplicación (`AjustesScreen`). Utiliza una barra de navegación inferior (`BottomNavigationBar`) para facilitar la interacción del usuario.
- **Navegación**:
  - La navegación entre las pantallas se realiza mediante la barra inferior (`BottomNavigationBar`).
  - Al pulsar un icono, el índice correspondiente cambia y actualiza el contenido mostrado.
- **Listado de contenidos**:
  - **Barra de navegación inferior** (`BottomNavigationBar`):
    - Iconos:
      - Tareas: `Icons.fact_check_outlined`.
      - Nevera: `Icons.production_quantity_limits_rounded`.
      - Ajustes: `Icons.settings`.
    - Colores:
      - Seleccionado: Azul.
      - No seleccionado: Gris.
  - **Lista de pantallas** (`_pages`):
    - `ToDoScreen`: Gestión de tareas pendientes.
    - `NeveraScreen`: Registro y organización de los alimentos disponibles.
    - `AjustesScreen`: Configuración y personalización de la aplicación.
- **Funciones principales**:
  - `_onItemTapped(int index)`:
    - Cambia el índice seleccionado (`_selectedIndex`) según el icono pulsado.
    - Actualiza la pantalla mostrada en el cuerpo principal (`body`).

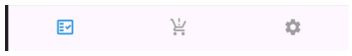

Figura 17: Pantalla nav menu (HomeScreen).

5.1.8. Pantalla de Gestión de Tareas (`ToDoScreen`)

- **Descripción**: La pantalla permite gestionar tareas asociadas a un hogar, incluyendo la creación, edición, eliminación y marcado de tareas como completadas. Incluye funcionalidades para filtrar tareas y asignarlas a los miembros del hogar.
- **Requisitos**:
  - El usuario debe estar autenticado y asociado a un hogar en la base de datos.
  - Firebase Firestore debe estar configurado correctamente para gestionar la colección de tareas.
  - Los miembros del hogar deben estar registrados y visibles en la pantalla.
- **Navegación**:
  - La pantalla muestra una lista de tareas filtradas según el estado seleccionado: `Todas`, `Completadas`, o `Incompletas`.
  - Incluye un botón flotante para añadir nuevas tareas y un menú emergente para editar tareas existentes.
- **Listado de contenidos**:
  - **Barra superior**:
    - Menú desplegable para filtrar tareas (`Todas`, `Completadas`, `Incompletas`).
  - **Lista de tareas**:
    - Muestra tareas con su nombre, categoría y asignación.
    - Permite marcar tareas como completadas o incompletas.
    - Incluye opciones para editar o eliminar tareas.
  - **Botón flotante**:
    - Inicia un diálogo para añadir una nueva tarea, especificando el nombre, categoría y miembro asignado.
- **Funciones principales**:
  - `_fetchUserHomeTasks()`:
    - Obtiene las tareas asociadas al hogar del usuario desde Firebase Firestore.
  - `_loadTasks()`:
    - Carga las tareas del hogar actual y las almacena en la lista de tareas.
  - `_filterTasks()`:
    - Aplica filtros a la lista de tareas según el estado seleccionado.
  - `_addNewTask(String taskName, String category)`:
    - Añade una nueva tarea a la base de datos y la lista de tareas en la pantalla.
  - `_toggleCompleteTask(int index)`:
    - Cambia el estado de una tarea entre completada e incompleta.

- `_deleteTask(int index)`:
  - Elimina una tarea de la base de datos y de la lista en la pantalla.
- `_editTask(int index, String newTaskName, String newCategory)`:
  - Actualiza el nombre y categoría de una tarea existente.
- `_showAddTaskDialog()`:
  - Muestra un diálogo para ingresar detalles de una nueva tarea.
- `_showEditTaskDialog(int index)`:
  - Muestra un diálogo para editar una tarea seleccionada.

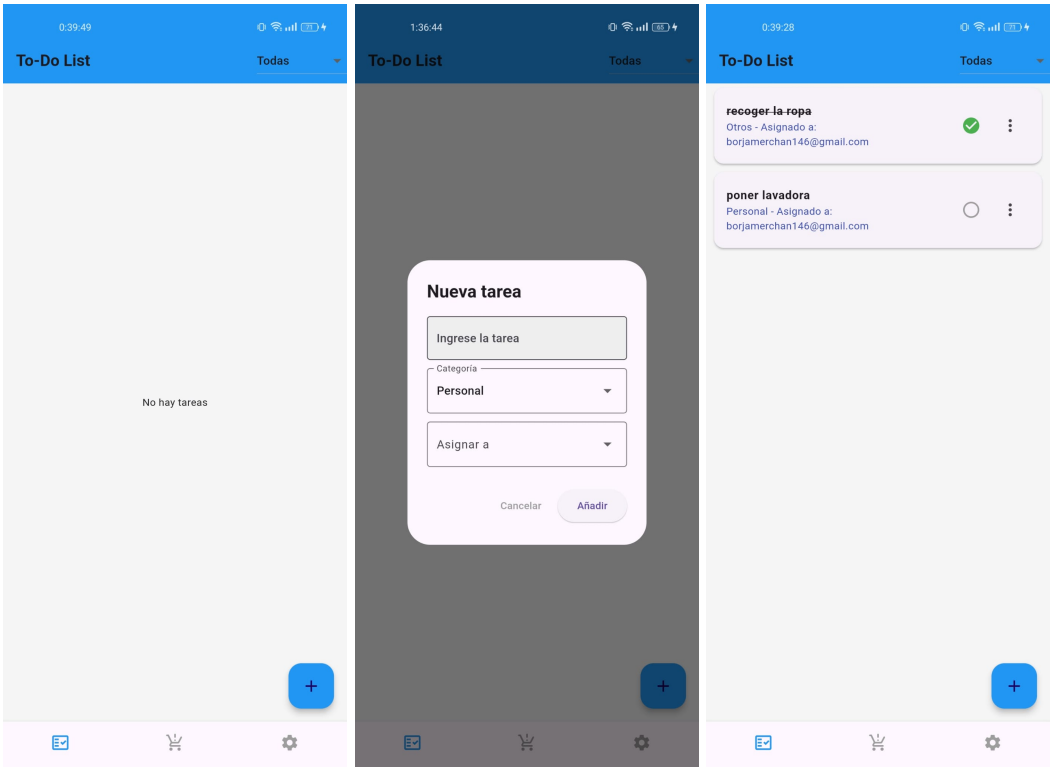

Figura 18: Pantallas de Gestión de Tareas (`ToDoScreen`).

5.1.9. Pantalla de Gestión de Productos (`NeveraScreen`)

- **Descripción**: Esta pantalla permite gestionar productos asociados a un hogar en un formato de tablero Kanban. Los productos pueden estar en diferentes estados (`En Nevera` y `Por Comprar`), ser arrastrados entre columnas, y gestionados mediante opciones como añadir, eliminar o mover productos. Incluye una integración con la API de Mercadona para explorar categorías y subcategorías de productos.
- **Requisitos**:
  - El usuario debe estar autenticado y asociado a un hogar en Firebase Firestore.
  - Firebase Firestore debe estar configurado correctamente para gestionar los productos.
- **Navegación**:
  - Los productos se organizan en dos columnas principales: `En Nevera` y `Por Comprar`.
  - Los productos pueden ser arrastrados entre columnas para cambiar su estado.
  - Un botón flotante permite añadir nuevos productos manualmente o desde la API de Mercadona.
  - Si el usuario realiza un "long click"sobre una tarjeta de producto, este se elimina tanto de la lista visual como de Firebase Firestore.
- **Listado de contenidos**:
  - **Columnas del tablero Kanban**:
    - `En Nevera`: Productos disponibles actualmente en la nevera.
    - `Por Comprar`: Productos que aún deben adquirirse.
  - **Botón flotante**:
    - Abre un diálogo para añadir productos manualmente o desde la API de Mercadona.
  - **Tarjetas de producto**:
    - Muestran el nombre, categoría y número de clics de cada producto.
    - Incluyen opciones para eliminar productos.
- **Funciones principales**:
  - `_fetchUserHomeProducts()`:
    - Obtiene los productos del hogar del usuario desde Firebase Firestore.
  - `_loadProducts()`:
    - Carga los productos en las columnas del tablero Kanban.
    - Actualiza el contador de clics de cada producto desde Firebase.
  - `_addNewProduct(String productName, String category)`:
    - Añade un nuevo producto a la base de datos y lo asigna al estado `Por Comprar`.
  - `_moveProduct(Product product, String newStatus)`:


    - Cambia el estado de un producto (`En Nevera` o `Por Comprar`) y actualiza la base de datos.
- `_deleteProduct(String productId)`:
    - Elimina un producto de Firebase Firestore y de la lista local.
- `_handleProductClick(Product product, String status)`:
    - Incrementa o decrementa el contador de clics del producto según su estado.
- `_showAddProductDialog()`:
    - Muestra un diálogo para añadir nuevos productos manualmente o desde la API de Mercadona.
- `_showMercadonaProductsDialog()`:
    - Integra la API de Mercadona para explorar categorías y subcategorías de productos.

- **Interacción con la API de Mercadona**:
    - Descarga categorías de productos mediante la API de Mercadona.
    - Permite explorar subcategorías y añadir productos directamente desde la API.

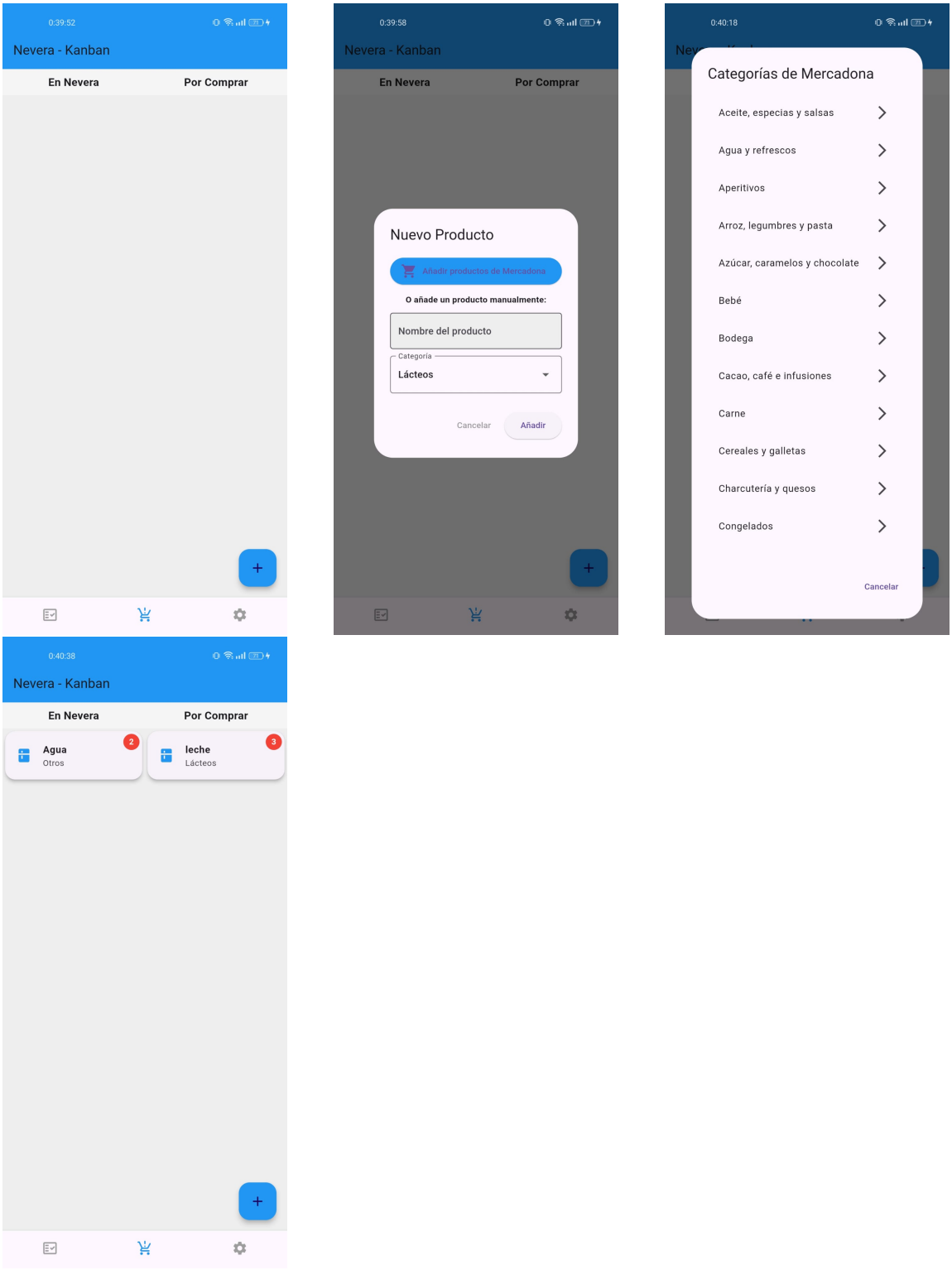

Figura 19: Pantallas de Gestión de Productos (`NeveraScreen`).

5.1.10. Pantalla de Ajustes (`AjustesScreen`)

- **Descripción**: Esta pantalla permite al usuario visualizar y gestionar ajustes relacionados con su cuenta y hogar. Incluye funcionalidades para seleccionar un grupo de hogar, listar los miembros del hogar y cerrar sesión.
- **Requisitos**:
  - El usuario debe estar autenticado en Firebase Authentication.
  - Firebase Firestore debe estar configurado para almacenar y gestionar los grupos de hogar y sus miembros.
- **Navegación**:
  - El usuario puede visualizar el grupo de hogar que se ha registrado.
  - Los miembros del hogar seleccionado se muestran en una lista.
  - Un botón permite cerrar la sesión y redirigir al usuario a la pantalla de inicio de sesión (`LoginScreen`).
- **Listado de contenidos**:
  - **Avatar del usuario**:
    - Muestra una imagen representativa del usuario.
  - **Tarjeta de información del usuario**:
    - Muestra el nombre y correo electrónico del usuario autenticado.
  - **Lista de miembros del hogar**:
    - Muestra los correos electrónicos de los miembros del hogar seleccionado.
    - Si no hay miembros disponibles, se muestra un mensaje informativo.
  - **Botón de cerrar sesión**:
    - Permite al usuario cerrar su sesión y regresar a la pantalla de inicio de sesión.
- **Funciones principales**:
  - `_fetchUserGroups()`:
    - Obtiene los grupos de hogar a los que pertenece el usuario desde Firebase Firestore.
  - `_fetchHomeMembers(String homeName)`:
    - Obtiene los miembros del hogar seleccionado desde Firebase Firestore.
  - `Cerrar sesión`:
    - Cierra la sesión del usuario y redirige a la pantalla de inicio de sesión (`LoginScreen`).

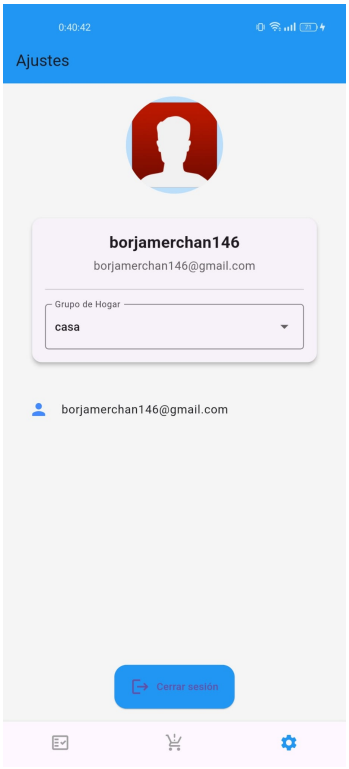

Figura 20: Pantalla de Ajustes (`AjustesScreen`).

6. Biografia

6.1. Documentación

- Documentación Offical de flutter
- Información de Como subir la aplicacion ha app store y play store
- Documentación Official de firebase
- Importar y exportar datos de firebase
- Para exportar usarios de la base de datos de autodentificador
- Tener Dibujado el json en modo visual
- Canal de Flutter
  https://www.youtube.com/@NetNinja