IES RIBERA DE CASTILLA

PROYECTO FINAL

CFGS Desarrollo de Aplicaciones Multiplataforma
Informática y Comunicaciones

Food Manager

Tutor individual: *Rodrigo Iglesias Gorron*
Tutor colectivo: *Cristina Silván Pardo*
Año: *2024/2025*
Fecha de presentación: *12/12/2025*

Nombre y Apellidos: Daniel Eugenio Piana Salviejo
Email: daniele.piasal@educa.jcyl.es

ÍNDICE

1. Identificación del proyecto

Este proyecto trata sobre una aplicación multiplataforma para gestionar todo lo relacionado con la **compra** y la **comida** del hogar. La app tiene cuatro partes principales: gestión de productos con escaneo de códigos de barras, lista de la compra donde puedes marcar lo que ya has comprado, un registro de facturas y una sección para guardar recetas. Todo esto para tener centralizado en una sola aplicación lo que antes tenía repartido en varios sitios.

Durante el desarrollo he trabajado en cosas como conectar la app con la **API de Open Food Facts** para identificar productos automáticamente, hacer una interfaz que se adapte con modo oscuro y que funcione en español e inglés, crear un sistema para compartir e importar recetas usando códigos únicos, y sincronizar todo en tiempo real con Supabase. También he tenido que crear desde cero un componente para los pasos de las recetas porque las librerías que había no me servían, y he usado **Optimistic Updates** para que la app se sienta más rápida.

El objetivo era crear una aplicación que funcione bien y que realmente sea útil para organizar las compras, poniendo en práctica lo que he aprendido en el Ciclo de Desarrollo de Aplicaciones Multiplataforma.

2. Organización de la memoria

La memoria está dividida en varias:

1. **Descripción general del proyecto**: Aquí cuento los objetivos (lista de la compra, recetas, productos y facturas), cómo he ido desarrollándolo (de forma ágil, haciendo cambios sobre la marcha) y qué herramientas he usado (Flutter, Supabase, Open Food Facts).
2. **Descripción general del producto**: Explico qué hace la app, para quién es (principalmente para mí y mi familia, pero puede usarla cualquiera), qué puedes hacer en cada módulo y cómo se conecta con Supabase y Open Food Facts.
3. **Planificación y presupuesto**: Muestro cuánto tiempo me ha llevado (168 horas en 3 meses) y cuánto costaría el proyecto (desarrollo, software, hardware, etc.).
4. **Documentación técnica**: Esta es la parte más extensa. Incluye qué puede hacer cada módulo con todo detalle, cómo está montada la base de datos con las 7 tablas en PostgreSQL, cómo navegar por la app con capturas, la arquitectura del código (MVC con Provider), y dos cosas que destaco: los

Optimistic Updates para que vaya más fluida y el stepper que hice a medida para los pasos de las recetas.

5. **Manuales de usuario e instalación**: Primero explico paso a paso cómo instalar todo lo necesario para desarrollar (Flutter, IntelliJ, Android Studio, Visual Studio). Luego viene un manual de uso completo con capturas de todas las pantallas: cómo registrarte, crear productos, hacer la lista de la compra, generar facturas y gestionar recetas.
6. **Conclusiones y posibles ampliaciones**: Reflexiono sobre lo que he aprendido: que la planificación es muy importante, que he conseguido los objetivos, que me he dado cuenta de que la reutilización de componentes podría estar mejor, y sobre todo que ahora cuando veo algo que me gustaría que existiera pienso en si podría hacerlo yo. También pongo ideas de mejoras futuras como refactorizar el código, añadir web scraping para precios y hacer que funcione sin conexión.
7. **Bibliografía**: Lista de lo que he consultado: documentación de Flutter, ChatGPT y Claude, apps similares que miré (InvenDo, Listonic...) y vídeos de YouTube sobre Flutter.
8. **Anexos**: Aquí van enlaces al GitHub con el código, al Figma con el prototipo y a la documentación del proyecto.

3. Descripción general del proyecto

Objetivos

Este proyecto nace por una necesidad personal: juntar en una única aplicación la gestión de información relacionada con la compra y alimentación. Antes de desarrollar esta aplicación, tenía cada cosa apuntada en un sitio diferente, lo que me complicaba bastante la organización. Tras probar varias aplicaciones dedicadas a este tema en el mercado, ninguna me convenció: o tenían demasiadas funcionalidades que no necesitaba y complicaban su uso, o se quedaban cortas y no cubrían todo lo que buscaba. Por eso decidí crear mi propia aplicación que se ajustase exactamente a lo que necesitaba.

Aunque la aplicación está pensada principalmente para uso personal y familiar, cualquier usuario con necesidades similares de organización de compras podría utilizarla.

Objetivos principales:

- **Gestión de lista de la compra:** Gestión de los productos que se quieren comprar, con interfaz visual para marcar y desmarcar para facilitar su uso.
- **Gestión de recetas:** Gestión de recetas para facilitar su planificación y realización, así como la relación entre ingredientes y productos.

Objetivos secundarios:

- **Gestión de productos:** Mantener un catálogo de productos del supermercado que sirva de base para las listas de la compra y las recetas.
- **Gestión de facturas:** Registrar y guardar las facturas de compra, permitiendo llevar un control del gasto.

Desde el punto de vista técnico, el proyecto incluye la integración con una API externa para escanear códigos de barras, facilitando la identificación rápida de productos. También se ha implementado un sistema de sincronización de datos para garantizar que la información esté siempre disponible. La aplicación funciona en Android y Windows.

Cuestiones metodológicas

Metodología de desarrollo

Para el desarrollo de este proyecto se ha seguido una metodología **mixta** que combina una planificación inicial básica con desarrollo iterativo e incremental.

En la fase inicial se realizó un prototipado básico en Figma donde se definieron los elementos principales de la interfaz: las pestañas de navegación, el buscador, los títulos y la estructura general de las funcionalidades. Sin embargo, durante el desarrollo el diseño visual ha cambiado considerablemente y se han añadido más funcionalidades que no estaban contempladas en el prototipo inicial. Por tanto, la metodología ha sido más cercana a un **desarrollo ágil e iterativo**, donde las funcionalidades se han ido implementando y refinando en base a las necesidades que surgían durante el proceso.

El orden de desarrollo de las funcionalidades ha sido el siguiente:

1. Gestión de productos
2. Gestión de lista de la compra
3. Gestión de facturas
4. Gestión de recetas

Durante el desarrollo ha sido necesario volver atrás y replantear decisiones técnicas importantes. El cambio más significativo fue pasar de utilizar una base de datos local a implementar **Supabase**, lo que permite compartir una cuenta entre varios usuarios.

Para el control de versiones se ha utilizado **GitHub**, lo que ha facilitado el seguimiento de cambios y la gestión del código fuente. El desarrollo se ha realizado de forma individual, contando con una persona como tester para validar las funcionalidades implementadas y encontrar errores.

Entorno de trabajo (tecnologías de desarrollo y herramientas)

Modelo arquitectónico

El proyecto sigue un **modelo cliente-servidor**, donde la aplicación desarrollada en Flutter actúa como cliente y **Supabase** como servidor backend que gestiona la base de datos, autenticación y almacenamiento de archivos.

Lenguajes de programación y frameworks

- **Flutter:** Framework multiplataforma de código abierto desarrollado por Google para crear aplicaciones nativas. Versión: Flutter 3.32.8
- **Dart:** Lenguaje de programación utilizado por Flutter, también desarrollado por Google. Versión: Dart Sdk 3.8.1

Entorno de desarrollo

- **IntelliJ IDEA:** IDE (Entorno de Desarrollo Integrado) utilizado para la programación y desarrollo de la aplicación.
- **GitHub:** Plataforma de control de versiones basada en Git.
- **Figma:** Herramienta de diseño de interfaces y prototipado.
- **DartDoc:** Herramienta de generación automática de documentación a partir del código fuente.

Backend y servicios externos

- **Supabase:** Plataforma backend que proporciona:
    - Base de datos PostgreSQL.
    - Sistema de autenticación.
    - Storage para almacenamiento de fotos.

- **Open Food Facts:** API externa para escaneo y consulta de información de productos mediante códigos de barras, gratis y de código abierto.

Paquetes y librerías principales de Flutter

Gestión de estado:

- **provider** (^6.1.2)

Backend y base de datos:

- **supabase_flutter** (^2.8.4)
- **shared_preferences** (^2.4.0)

Escaneo y selección de archivos:

- **mobile_scanner** (^6.0.2)
- **image_picker** (^1.0.4)
- **file_picker** (^10.3.0)

APIs externas y networking:

- **openfoodfacts** (^3.2.0)
- **http** (^1.2.0)
- **html** (^0.15.4)

Interfaz de usuario:

- **flutter_slidable** (^3.0.0)
- **flutter_speed_dial** (^7.0.0)
- **awesome_snackbar_content** (^0.1.7)

Compartir y portapapeles:

- **share_plus** (^10.0.2)
- **super_clipboard** (^0.9.1)
- **url_launcher** (^6.3.2)

Internacionalización:

- **flutter_localizations** (SDK)
- **intl**

Utilidades:

- **flutter_dotenv** (^5.2.1)
- **path_provider** (^2.1.3)
- **path** (^1.2.0)

4. Descripción general del producto

Visión general del sistema

La aplicación es un sistema de gestión personal para la organización de compras y alimentación. Está diseñada como una aplicación cliente que interactúa con servicios externos: Supabase como backend para almacenamiento y autenticación, y Open Food Facts como API para obtener información de productos mediante códigos de barras.

Límites del sistema

La aplicación no es completamente autónoma, ya que depende de supabase externos para su funcionamiento.

Sin conexión a internet, la aplicación no puede funcionar correctamente, ya que toda la información se almacena en la nube.

Funcionalidades básicas

El sistema se divide en cuatro módulos principales:

Gestión de productos:

- Crear, editar y eliminar productos
- Escanear códigos de barras para añadir o actualizar productos
- Búsqueda de productos por nombre o supermercado
- Añadir productos directamente a la lista de la compra

Gestión de lista de la compra:

- Añadir y eliminar productos de la lista
- Marcar y desmarcar productos como comprados
- Búsqueda de productos en la lista por nombre o supermercado
- Compartir la lista (por WhatsApp en móvil, o copiando al portapapeles en PC)
- Generar factura a partir de la lista

Gestión de facturas:

- Visualización de facturas
- Eliminar facturas
- Búsqueda de facturas por día, mes, año, mes en formato texto, o supermercado

Gestión de recetas:

- Crear, eliminar, buscar recetas (por duración o nombre) e importar recetas de otra persona.
- Abrir detalles de cada receta para:
    - Actualizar información de la receta
    - Crear y editar pasos de preparación
    - Vincular productos necesarios
    - Añadir todos los productos de la receta a la lista de la compra

Funcionalidades extra:

- Sistema de búsqueda personalizado para cada módulo
- Autenticación de usuarios
- Sincronización de datos en tiempo real

Características de la interfaz:

- Modo oscuro
- Traducción al inglés
- Soporte de orientación horizontal y vertical

Usuarios

La aplicación requiere registro obligatorio mediante correo electrónico y contraseña. No existe diferenciación de roles: todos los usuarios tienen los mismos permisos y acceso a las mismas funcionalidades.

El sistema permite que varias personas compartan la misma cuenta sin problemas, facilitando que una familia pueda gestionar conjuntamente las compras y recetas del hogar.

Sistemas operativos compatibles

La aplicación es multiplataforma y ha sido probada en:

- **Android:** varios dispositivos móviles y tablets
- **Windows:** ordenadores de escritorio

Aunque Flutter permite compilar para otras plataformas (iOS, macOS, Linux), no se ha verificado su funcionamiento en estos sistemas operativos.

5. Planificación y presupuesto

Planificación

El proyecto se ha desarrollado a lo largo de aproximadamente 3 meses de trabajo efectivo, distribuyendo las tareas de la siguiente manera:

Fase	Horas dedicadas
Diseño inicial (Figma)	4 horas
Desarrollo - Gestión de productos	22 horas
Desarrollo - Lista de la compra	14 horas
Desarrollo - Facturas	8 horas
Desarrollo – Recetas	40 horas
Testing y correciones	60 horas
Documentación	20 horas
TOTAL	168 horas

Las fases que más tiempo han requerido han sido el desarrollo del módulo de recetas y el proceso de migración de base de datos local a Supabase, que implicó reestructurar gran parte del código ya desarrollado.

Presupuesto

Coste de desarrollo:

- Horas totales: 168 horas
- Precio hora programador junior: 12€/hora
- **Total desarrollo: 2.016€**

Coste de software:

- Flutter SDK: Gratuito (open source)
- Dart SDK: Gratuito (open source)
- IntelliJ IDEA Community Edition: Gratuito
- GitHub: Plan gratuito
- Figma: Plan gratuito
- Supabase: Plan gratuito
- Open Food Facts API: Gratuito
- **Total software: 0€**

Coste de hardware (amortizado):

- Ordenador de desarrollo: 1.500€
- Móvil para testing: 250€
- **Total hardware: 1750€**

Coste de hosting:

- Supabase (plan gratuito): 0€
- **Total hosting: 0€**

COSTE TOTAL DEL PROYECTO: 3.766€

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

Análisis del sistema

Especificación de requisitos funcionales

A continuación, se detalla el funcionamiento de cada módulo: operaciones disponibles, campos, validaciones y comportamientos.

GESTIÓN DE PRODUCTOS

Alta de productos

Los productos contienen los siguientes campos:

- ID (generado automáticamente)
- Nombre
- Descripción
- Precio
- Supermercado
- Código de barras (obligatorio solo si se escanea)
- Foto
- Usuario (asociado automáticamente al usuario logueado)

Campos obligatorios: nombre, precio y supermercado. Si se añade mediante escáner de código de barras, el código de barras también es obligatorio, pero este se autocompleta y no se puede modificar.

Validaciones:

- Ningún campo puede estar vacío (excepto descripción y foto)
- El precio debe ser un número válido (entero o decimal)
- El precio no puede ser negativo

Escaneo de código de barras:

- Si el código es encontrado en la API de Open Food Facts:
    - Se rellenan automáticamente: nombre, foto y descripción (con la marca del producto)
- Si no se encuentra el producto:
    - Aparece un mensaje indicando que no se ha encontrado y se debe rellenar manualmente

Modificación de productos

Todos los campos pueden ser modificados, excepto el código de barras que solo se puede actualizar mediante escáner.

Al actualizar con escáner, se rellenan automáticamente los mismos campos que en el alta: nombre, foto y descripción.

Eliminación de productos

Se solicita confirmación antes de eliminar el producto.

Al borrar un producto:

- Se elimina de la lista de la compra
- Se elimina de las recetas vinculadas
- NO se elimina de las facturas ya generadas (para mantener el histórico)

Búsqueda de productos

La búsqueda se realiza en tiempo real mientras el usuario escribe, permitiendo filtrar por:

- Nombre del producto
- Supermercado

Añadir a lista de la compra

Desde la gestión de productos se puede añadir directamente cualquier producto a la lista de la compra.

GESTIÓN DE LISTA DE LA COMPRA

Añadir productos

Los productos se pueden añadir de dos formas:

- Individualmente desde la gestión de productos
- Todos los productos de una receta desde los detalles de una receta

Al añadir un producto se especifica:

- Cantidad

Nota: Si se crean dos productos diferentes con datos idénticos (sin escanear), es posible tener productos duplicados en la lista.

Marcar/desmarcar productos

Cada producto tiene un checkbox a la izquierda que permite:

- Marcar con un tick cuando ha sido comprado
- Desmarcar dejándolo vacío
- El marcado/desmarcado se puede alternar libremente

Eliminar productos

Se solicita confirmación antes de eliminar.

Al eliminar, el producto se borra únicamente de la lista de la compra, pero permanece en el catálogo de productos.

Compartir lista

La lista se puede compartir con el siguiente formato:

```
---- Lista de la compra ----
[Supermercado 1]
 [ ] • Producto 1 — X,XX €/u × cantidad = total €
 [ ] • Producto 2 — X,XX €/u × cantidad = total €
[Supermercado 2]
 [ ] • Producto 3 — X,XX €/u × cantidad = total €
Total: XXX,XX €
```

Solo se comparten los productos marcados.

En dispositivos móviles se comparte por WhatsApp, en PC se copia al portapapeles.

Generar factura

Al generar una factura:

- Se guardan todos los productos de la lista con su cantidad, precio unitario y precio total
- Se calcula y almacena el precio total de la compra
- La lista debe contener al menos un producto (no se puede generar con lista vacía)
- Todos los productos de la lista se eliminan automáticamente tras generar la factura

Búsqueda en lista

La búsqueda se realiza en tiempo real, filtrando por:

- Nombre del producto
- Supermercado

GESTIÓN DE FACTURAS

Campos de una factura

Cada factura contiene:

- ID (generado automáticamente)
- Fecha (automática al momento de creación)
- Precio total
- Usuario (asociado al usuario logueado)
- Lista de productos con cantidad y precios

Creación

Las facturas solo se pueden crear desde la lista de la compra mediante la función "Generar factura".

Visualización

Las facturas se muestran en una lista expansible donde:

- El título muestra la fecha de la factura
- Al expandir se muestran todos los productos incluidos

Eliminación

Se solicita confirmación antes de eliminar una factura.

Búsqueda

La búsqueda es combinada, en tiempo real y permite filtrar por:

- Día
- Mes (numérico)
- Año
- Mes (en formato texto)
- Supermercado

GESTIÓN DE RECETAS

Campos de una receta

Cada receta contiene:

- ID (generado automáticamente)
- Nombre
- Descripción
- Foto
- Tiempo de preparación
- Usuario (asociado al usuario logueado)
- Pasos (tabla relacionada)
- Productos vinculados (tabla relacionada)
- Código para compartir
- Código con el que se ha importado la receta, si es que se ha importado y no creado manualmente

Campos obligatorios: nombre, descripción y tiempo.

Validaciones: los campos obligatorios no pueden estar vacíos.

Alta de recetas

Se crea una nueva receta con los datos básicos. Los pasos y productos se añaden posteriormente desde la vista de detalles.

Detalles de receta (edición)

Gestión de pasos:

- Los pasos se añaden desde la pestaña de detalles mediante un botón FAB (Floating Action Button)
- Al añadir un paso se entra en modo edición, se crea el paso vacío y el usuario rellena los datos
- Si es el primer paso, se muestra un texto que dice "Para añadir tu primer paso haz click aquí"
- Los pasos solo contienen texto (sin imágenes)
- Los pasos tienen un orden

Vincular productos:

- Los productos se seleccionan de la lista de productos existentes
- No se especifica cantidad en la vinculación

Añadir productos a lista de la compra:

- Desde los detalles de la receta se pueden añadir todos los productos vinculados a la lista de la compra
- Se añaden con cantidad predeterminada (1)

Eliminación

No se solicita confirmación para eliminar.

Búsqueda

La búsqueda se realiza en tiempo real, filtrando por:

- Nombre de la receta
- Tiempo de preparación

Compartir e importar recetas

Compartir receta:

Desde la lista de recetas, cada receta tiene un botón de compartir que:

- Genera un código único asociado a esa receta
- Permite compartir el código por WhatsApp
- El código se guarda en el campo "Código para compartir" de la receta

Importar receta:

Desde el appbar de la lista de recetas se puede importar una receta mediante un código compartido:

1. Se pulsa el botón de importar
2. Aparece una alerta solicitando el código
3. Al introducir el código, se muestra una vista previa con:
    a. Nombre de la receta
    b. Productos vinculados que se importarán
    c. Aviso de que los productos se crearán en una carpeta especial con el nombre de la receta
    d. Pasos de la receta
4. Al confirmar la importación:
    a. Se importa la receta con todos sus datos básicos
    b. Se importan todos los pasos de preparación

c. Se crean los productos vinculados en una carpeta especial
d. Se genera un nuevo código único para esta receta (diferente al original)
e. Se guarda el código con el que se importó en el campo "Código con el que se ha importado"

Validación de duplicados: Si se intenta importar una receta que ya fue importada anteriormente (mismo código), el sistema lo detecta mediante el campo "Código con el que se ha importado" y evita la duplicación de recetas.

La duplicación de productos para claridad del usuario es necesaria, el usuario ya decide si actualizar el supermercado o no.

AUTENTICACIÓN DE USUARIOS

Registro

Campos:

- Correo electrónico
- Contraseña
- Confirmación de contraseña

Validaciones:

- El correo debe tener formato de email válido
- La contraseña debe tener un mínimo de 6 caracteres
- La confirmación de contraseña debe coincidir

Comportamiento:

- Si el correo ya está registrado, aparece un mensaje indicándolo y sugiriendo iniciar sesión
- Si el usuario se crea una cuenta, automáticamente se iniciará sesión con esos datos.

Inicio de sesión

Comportamiento:

- Si las credenciales son incorrectas, aparece el mensaje "Contraseña incorrecta"
- La sesión se recuerda automáticamente hasta que el usuario cierre sesión manualmente
- No existe opción de recuperar contraseña

Cierre de sesión

Disponible desde el menú lateral (drawer), permite cerrar la sesión actual.

CONFIGURACIÓN Y AJUSTES

Accesibles desde el menú lateral (drawer).

Modo oscuro

- Se activa/desactiva mediante un switch
- La preferencia se guarda y persiste entre sesiones

Idioma

- Se selecciona mediante un DropDownButton (español/inglés)
- La preferencia se guarda y persiste entre sesiones

Diseño del sistema

Diseño de la Base de Datos

La base de datos se ha diseñado utilizando PostgreSQL a través de Supabase. A continuación, se describe el modelo de datos implementado.

Modelo Entidad-Relación

El sistema cuenta con 7 tablas principales que gestionan toda la información de la aplicación:

Tabla USUARIO

- **idusuario** (INTEGER, PK): Identificador único del usuario
- **usuariouuid** (UUID, UNIQUE): UUID generado por Supabase para autenticación
- **nombreusuario** (TEXT, UNIQUE): Nombre de usuario
- **correo** (TEXT, UNIQUE): Correo electrónico
- **fechacreacion** (TIMESTAMP): Fecha de registro

Esta tabla almacena la información básica de los usuarios registrados en la aplicación.

Tabla PRODUCTOS

- **id** (INTEGER, PK): Identificador único del producto
- **codbarras** (TEXT, UNIQUE): Código de barras del producto
- **nombre** (TEXT): Nombre del producto
- **descripcion** (TEXT): Descripción o marca
- **precio** (FLOAT): Precio del producto
- **supermercado** (TEXT): Supermercado donde se vende
- **foto** (TEXT): URL de la imagen del producto
- **usuariouuid** (UUID, FK): Referencia al usuario propietario

Esta tabla contiene el catálogo de productos disponibles para cada usuario.

Tabla COMPRA

- **idproducto** (INTEGER, PK, FK): Referencia al producto
- **nombre** (TEXT): Nombre del producto (desnormalizado para optimizar consultas)
- **precio** (FLOAT): Precio del producto
- **marcado** (INTEGER): Estado del producto (0=sin comprar, 1=comprado)
- **cantidad** (INTEGER): Cantidad a comprar (por defecto 1)
- **total** (FLOAT): Precio total (precio × cantidad)
- **usuariouuid** (UUID): Referencia al usuario

Almacena la lista de la compra actual del usuario. Al generar una factura, los productos se eliminan de esta tabla.

Tabla FACTURAS

- **id** (INTEGER, PK): Identificador único de la factura
- **precio** (FLOAT): Precio total de la factura
- **fecha** (TEXT): Fecha de creación
- **usuariouuid** (UUID): Referencia al usuario

Registra las facturas generadas desde la lista de la compra.

Tabla PRODUCTO_FACTURA

- **idproducto** (INTEGER, FK): Referencia al producto
- **idfactura** (INTEGER, FK): Referencia a la factura
- **cantidad** (INTEGER): Cantidad comprada
- **preciounidad** (FLOAT): Precio unitario en el momento de la compra
- **total** (FLOAT): Precio total del producto
- **usuariouuid** (UUID): Referencia al usuario

Tabla intermedia que relaciona productos con facturas (relación N:M), almacenando el detalle de cada compra.

Tabla RECETAS

- **id** (INTEGER, PK): Identificador único de la receta
- **nombre** (TEXT): Nombre de la receta
- **descripcion** (TEXT): Descripción de la receta
- **foto** (TEXT): URL de la imagen
- **tiempo** (TEXT): Tiempo de preparación
- **usuariouuid** (UUID): Referencia al usuario
- codigo_compartir: Código para compartir la receta
- codigo_importado: Código que indica el código con el que se ha importado la receta.

Almacena las recetas creadas por el usuario.

Tabla PASOS_RECETA

- **id** (INTEGER, PK): Identificador único del paso
- **receta_id** (INTEGER, FK): Referencia a la receta
- **numero_paso** (INTEGER): Orden del paso (por defecto 0)
- **titulo** (TEXT): Título del paso
- **descripcion** (TEXT): Descripción detallada del paso

Contiene los pasos de preparación de cada receta (relación 1:N con RECETAS).

Tabla RECETA_PRODUCTO

- **id** (INTEGER, PK): Identificador único
- **idreceta** (INTEGER, FK): Referencia a la receta
- **idproducto** (INTEGER, FK): Referencia al producto
- **cantidad** (TEXT): Cantidad del ingrediente

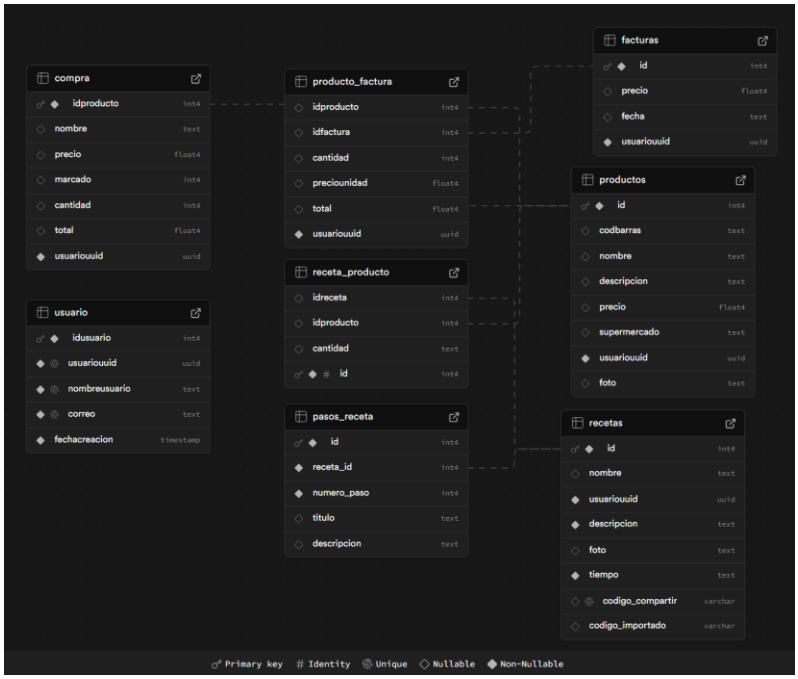
Tabla intermedia que vincula productos con recetas (relación N:M), representando los ingredientes necesarios.

Relaciones entre tablas

Las principales relaciones del modelo son:

- **USUARIO - PRODUCTOS**: 1:N (un usuario puede tener muchos productos)
- **USUARIO - COMPRA**: 1:N (un usuario tiene una lista de compra)
- **USUARIO - FACTURAS**: 1:N (un usuario puede tener muchas facturas)
- **USUARIO - RECETAS**: 1:N (un usuario puede tener muchas recetas)
- **PRODUCTOS - COMPRA**: 1:N (un producto puede estar en la lista de compra)
- **PRODUCTOS - FACTURAS**: N:M a través de PRODUCTO_FACTURA
- **FACTURAS - PRODUCTO_FACTURA**: 1:N (una factura contiene varios productos)
- **RECETAS - PASOS_RECETA**: 1:N (una receta tiene varios pasos)
- **RECETAS - PRODUCTOS**: N:M a través de RECETA_PRODUCTO

Diagrama de la Base de Datos

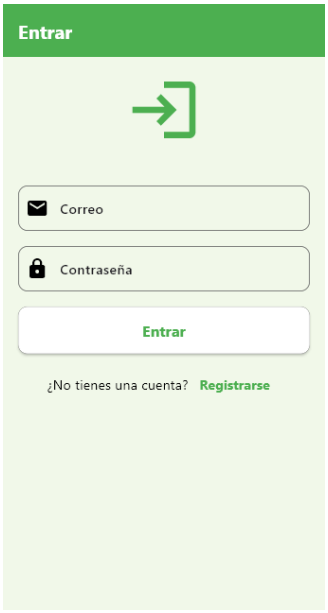


Diseño de la Interfaz de Usuario

La aplicación sigue un diseño basado en navegación por pestañas (tabs) con un menú lateral (drawer) para opciones de configuración.

Flujo de navegación

1. Autenticación



La aplicación inicia en la pantalla de **Login/Registro**, donde el usuario puede:

- Iniciar sesión con credenciales existentes
- Registrarse como nuevo usuario

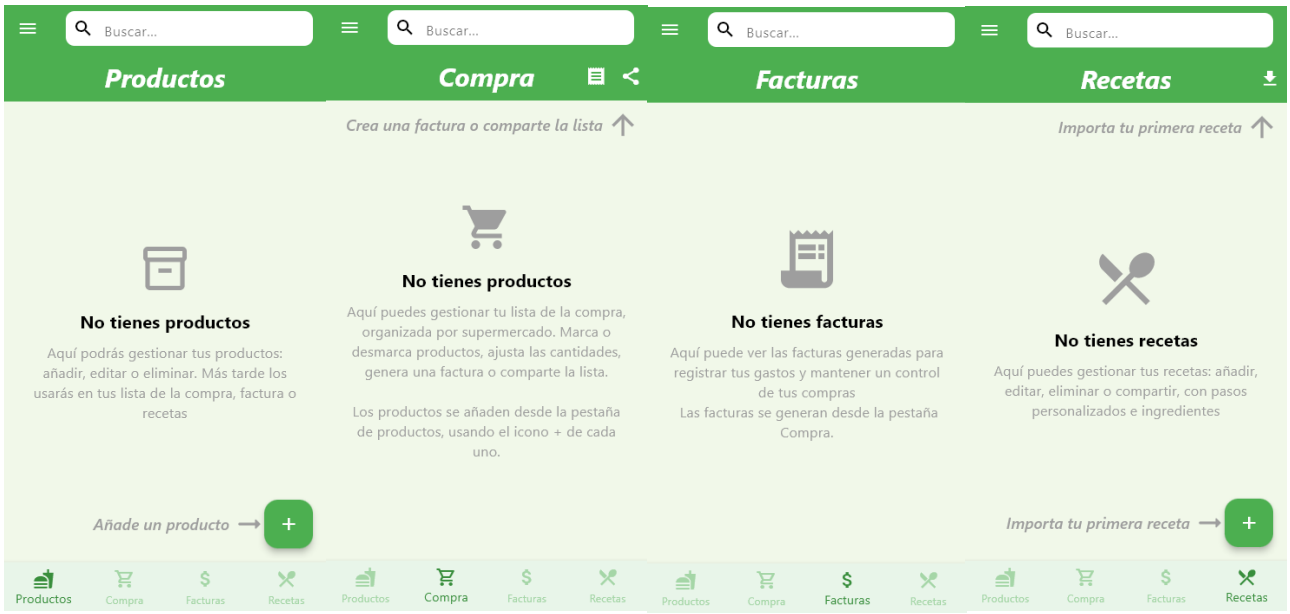
Una vez autenticado, el usuario es redirigido automáticamente a la pantalla principal.

2. Pantalla Principal

La pantalla principal utiliza un **BottomNavigationBar** con 4 pestañas principales:

- **Productos**: Lista de todos los productos del catálogo
- **Compra**: Lista de la compra actual
- **Facturas**: Histórico de facturas generadas
- **Recetas**: Lista de recetas guardadas

Desde cualquier pestaña se puede acceder al **Drawer** (menú lateral) deslizando desde el borde izquierdo o pulsando el icono del menú.



3. Pestaña Productos

Muestra una lista expansible de productos agrupada por supermercados con:

- Buscador en tiempo real (por nombre o supermercado)
- Botón FAB para añadir nuevo producto
- Al pulsar sobre un producto se accede a sus detalles

4. Pestaña Compra

Lista expansible de productos agrupada por supermercados comprar con:

- Checkbox para marcar/desmarcar productos
- Buscador en tiempo real
- Botones para:
  - Compartir lista (WhatsApp/portapapeles)
  - Generar factura

5. Pestaña Facturas

Lista expansible de facturas:

- El título muestra la fecha
- Buscador por múltiples criterios (fecha, supermercado)
- Al expandir se muestra el detalle de productos

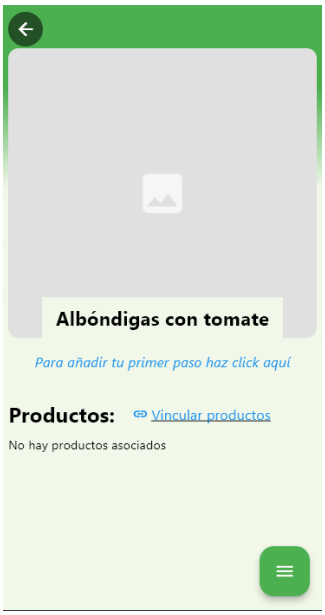
6. Pestaña Recetas

Lista de recetas con:

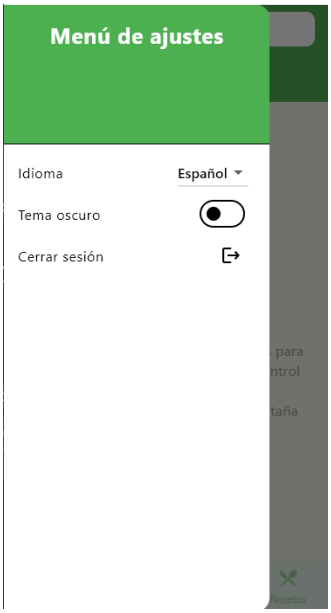
- Buscador en tiempo real (por nombre o duración)
- Botón FAB para crear nueva receta
- Al pulsar sobre una receta se accede a sus detalles
- Botón para importar receta

Detalles de Receta:

- Información general de la receta
- Lista de pasos de preparación
- Lista de productos vinculados
- Opciones para:
  - Editar información
  - Añadir/editar/eliminar pasos
  - Vincular productos
  - Añadir todos los productos a la lista de la compra
  - Eliminar receta



7. Drawer (Menú Lateral)



El drawer contiene:

- Switch para activar/desactivar modo oscuro
- ComboBox para cambiar idioma (Español/Inglés)
- Opción para cerrar sesión

Características extras de la interfaz

- **Modo oscuro/claro**: Toda la aplicación adapta su tema
- **Multiidioma**: Textos en español e inglés
- **Orientación**: Funciona en horizontal y vertical
- **Responsive**: Se adapta a diferentes tamaños de pantalla (móvil/tablet/desktop)

Diseño de la Aplicación

La aplicación sigue el patrón de arquitectura **Modelo-Vista-Controlador (MVC)** adaptado a Flutter, utilizando **Provider** como gestor de estado.

Estructura del proyecto

```
lib/
├── i10n/           # Archivos de internacionalización
├── models/         # Modelos de datos (clases)
├── Providers/      # Providers para gestión de estado
├── services/       # Servicios (comunicación con Supabase, APIs)
├── Utils/          # Utilidades y funciones auxiliares
├── View/           # Vistas/Pantallas de la aplicación
├── Widgets/        # Widgets reutilizables
└── main.dart       # Punto de entrada de la aplicación
```

Descripción de las capas

Capa de Modelos (models/)

Contiene las clases que representan las entidades de la base de datos:

- `Usuario`: Representa un usuario del sistema
- `Producto`: Representa un producto del catálogo
- `Compra`: Representa un producto en la lista de la compra
- `Factura`: Representa una factura generada
- `ProductoFactura`: Representa la relación producto-factura
- `Receta`: Representa una receta
- `PasoReceta`: Representa un paso de una receta
- `RecetaProducto`: Representa la relación receta-producto

Cada modelo incluye:

- Atributos correspondientes a los campos de la tabla
- Constructor
- Método `fromJson()/fromMap()` para deserializar datos de Supabase

Capa de Servicios (services/)

Contiene las clases que se comunican con servicios externos:

- `OpenFoodFactsService`: Gestiona las consultas a la API de códigos de barras

Capa de Providers (Providers/)

Implementa el patrón Provider para gestión de estado. Cada provider es responsable de:

- Mantener el estado de su dominio
- Notificar cambios a los widgets que lo escuchan
- Coordinar las llamadas a los servicios

Providers implementados:

- `user_provider`: Gestiona el estado del usuario autenticado
- `products_provider`: Gestiona la lista de productos
- `shopping_list_provider`: Gestiona el estado de la lista de la compra
- `receipts_provider`: Gestiona las facturas
- `recipe_provider`: Gestiona las recetas, pasos y productos vinculados
- `theme_provider`: Gestiona el tema (modo oscuro/claro)

- `language_provider`: Gestiona el idioma de la aplicación
- recipe_detail_provider: Gestiona los atributos de la receta concreta
- recipe_steps_provider: Gestiona los pasos de las recetas
- products_recipe_provider: Gestiona los productos de una receta

Capa de Vistas (View/)

Contiene las pantallas principales de la aplicación:

- `login_view`: Pantalla de inicio de sesión y registro
- `product_view`: Vista de la pestaña de productos
- `Shopping_list_view`: Vista de la lista de la compra
- `receipts_view`: Vista de facturas
- `recipes_view`: Vista de la lista de recetas
- `recipe_detail_view`: Vista de detalles de una receta

Capa de Widgets (Widgets/)

Contiene componentes reutilizables:

- awesone_snackbar: Constructor para las alertas
- acan_barcode_screen: Widget que carga la cámara para el escaneo
- products_placeholder: Widget a cargar si no tiene productos el usuario
- receipts_placeholder: Widget a cargar si no tiene facturas el usuario
- shopping_list_placeholder: Widget a cargar si no tiene lista de la compra el usuario
- recipe_placeholder: Widget a cargar si no tiene recetas el usuario
- custom_stepper: Widget para cargar los pasos de las recetas

Utilidades (Utils/)

Contiene funciones auxiliares y constantes:

- Formateadores de texto para comparar al buscar
- Constructor de nombre de imagen para guardar en Supabase
- Capitalizador para insertar textos en mayúsculas
- Selector de método para elegir imagen, dependiendo del dispositivo

Internacionalización (i10n/)

Gestiona las traducciones:

- Archivos JSON con strings en español e inglés

Flujo de datos

El flujo de datos sigue este esquema:

1. **Vista** → **Provider**: El usuario interactúa con la interfaz
2. **Provider** → **Supabase**: El provider solicita datos o realiza operaciones
3. **Supabase/API** → **Provider**: Devuelve los datos
4. **Provider**: El provider actualiza su estado
5. **Provider** → **Vista**: La vista se reconstruye con los nuevos datos

Este diseño permite:

- Separación de responsabilidades
- Facilidad para testear y debugear
- Escalabilidad del código
- Reutilización de componentes
- Gestión eficiente del estado de la aplicación

Implementación

Entorno de desarrollo

El desarrollo de la aplicación se ha realizado utilizando las siguientes herramientas:

- **Flutter SDK:** Versión 3.32.8
- **Dart SDK:** Versión 3.8.1
- **IntelliJ IDEA:** Community Edition 2024.2.2
    - Build #IC-242.22855.74 (18 de septiembre de 2024)
    - Runtime: OpenJDK 64-Bit Server VM 21.0.3
    - Plugins instalados:
        - Dart
        - Flutter
        - Android Tools
- **Sistema Operativo:** Windows 10.0

Estructura del código

La estructura del código sigue el patrón MVC (Modelo-Vista-Controlador) con Provider como gestor de estado, tal como se detalló en el apartado de Diseño de la Aplicación. El proyecto está organizado en las siguientes carpetas principales:

- `i10n/`: Internacionalización
- `models/`: Modelos de datos
- `Providers/`: Gestión de estado
- `services/`: Comunicación con servicios externos
- `Utils/`: Funciones auxiliares
- `View/`: Pantallas de la aplicación
- `Widgets/`: Componentes personalizados

Librerías y paquetes utilizados

Librerías agrupadas por funcionalidad:

Backend y autenticación:

- **supabase_flutter** (^2.8.4): Cliente de Supabase para Flutter, gestiona autenticación y base de datos
- **flutter_dotenv** (^5.2.1): Gestión de variables de entorno

Persistencia local:

- **shared_preferences** (^2.4.0): Almacenamiento de preferencias del usuario
- **path_provider** (^2.1.3): Acceso a rutas del sistema de archivos
- **path** (^1.8.3): Manipulación de rutas de archivos

Gestión de estado:

- **provider** (^6.1.2): Patrón de gestión de estado reactivo

Captura y selección de archivos:

- **image_picker** (^1.0.4): Selección de imágenes desde galería o cámara
- **file_picker** (^10.3.0): Selección de archivos del dispositivo
- **mobile_scanner** (^6.0.2): Escaneo de códigos de barras usando la cámara

APIs externas:

- **openfoodfacts** (^3.2.0): Cliente de la API de Open Food Facts para información de productos

Interfaz de usuario:

- **flutter_slidable** (^3.0.0): Acciones deslizables en elementos de lista
- **flutter_speed_dial** (^7.0.0): Botones de acción flotantes (FAB) con opciones múltiples
- **awesome_snackbar_content** (^0.1.7): Snackbars personalizadas

Compartir y portapapeles:

- **share_plus** (^10.0.2): Compartir contenido con otras aplicaciones
- **super_clipboard** (^0.9.1): Acceso avanzado al portapapeles del sistema
- **url_launcher** (^6.3.2): Abrir URLs en navegador o aplicaciones externas

Internacionalización:

- **flutter_localizations** (SDK): Soporte de localización de Flutter
- **intl**: Internacionalización y formateo de fechas/números

Todas las librerías utilizadas son de código abierto y gratuitas.

Librería de funciones de usuario

Se han desarrollado funciones auxiliares propias ubicadas en la carpeta `Utils/`:

- **capitalize**: Capitaliza la primera letra de un texto
- **imageNameNormalizer**: Normaliza y genera nombres de archivos de imagen
- **imagePicker**: Gestiona la selección y procesamiento de imágenes
- **textNormalizer**: Normaliza textos para búsquedas y comparaciones

Estas funciones encapsulan lógica reutilizable en diferentes partes de la aplicación.

Enfoque de programación

El código implementa una **combinación**:

- **Orientado a Objetos**: Los modelos, servicios y providers están implementados como clases con sus respectivos métodos y propiedades. Se aplican principios como encapsulación y abstracción.
- **Orientado a Eventos**: Flutter funciona de manera reactiva, donde los widgets responden a eventos del usuario (clics, deslizamientos, cambios de texto) y a cambios en el estado gestionado por los Providers mediante el patrón Observer.
- **Funcional**: Se utilizan conceptos de programación funcional como funciones de orden superior (map, where, forEach) y expresiones lambda, especialmente en el procesamiento de listas y transformación de datos.

Cuestiones de diseño e implementación reseñables

Durante el desarrollo del proyecto se han implementado dos elementos especialmente destacables:

1. Optimistic Updates

Se ha implementado el patrón de **Optimistic Updates** para mejorar la experiencia de usuario. Este patrón consiste en actualizar la interfaz inmediatamente cuando el usuario realiza una acción, antes de recibir confirmación del servidor.

Gracias a las validaciones implementadas en la interfaz, se controla que siempre se introduzcan datos correctos desde el frontend, minimizando así las posibilidades de error. Sin embargo, aunque los datos sean válidos, pueden ocurrir fallos ajenos a la aplicación como problemas de conexión a internet, caídas del servidor de Supabase, sesiones expiradas o errores internos del servidor.

Funcionamiento:

1. El usuario realiza una acción (por ejemplo, crear un producto)
2. La interfaz se actualiza inmediatamente mostrando el cambio
3. Paralelamente, se envía la petición a Supabase
4. Si la petición falla por cualquier motivo, se revierte el cambio en la interfaz y se notifica al usuario

Este patrón se aplica principalmente en:

- Creación de elementos
- Eliminación de elementos
- Actualización de elementos

Ventajas:

- La aplicación se siente más rápida y fluida
- Reduce la percepción de latencia de red
- Mejora significativamente la experiencia de usuario

Desventajas:

- Mayor complejidad en la gestión de errores
- Necesidad de revertir cambios si la operación falla

2. Stepper personalizado para recetas

Para la gestión de pasos de las recetas se necesitaba un componente stepper (paso a paso). Aunque existen librerías que proporcionan steppers, ninguna se adaptaba completamente a los requisitos que necesitaba:

Requisitos específicos:

- Permitir añadir, editar y eliminar pasos dinámicamente
- Mostrar un indicador visual del orden de los pasos
- Modo edición in-place para modificar pasos
- Diseño coherente con el resto de la aplicación
- Soporte para tema claro y oscuro
- Soporte de textos largos

Solución implementada:

Se desarrolló un widget `CustomStepper` completamente personalizado que:

- **Sistema de navegación por pasos**: Implementa navegación mediante botones "Anterior/Siguiente" en lugar de mostrar todos los pasos a la vez, mostrando un único paso en pantalla
- **Modo lectura/edición con animaciones**: Utiliza `AnimatedCrossFade` para alternar suavemente entre vista de lectura (solo visualización) y edición (campos de texto editables)
- **Indicadores visuales animados**: Emplea `AnimatedContainer` para mostrar puntos de progreso que cambian de tamaño y color según el estado del paso (actual, completado, pendiente)
- **Gestión automática de foco**: Al entrar en modo edición, el foco se establece automáticamente en el campo de título mediante un `FocusNode`
- **Creación y eliminación dinámica**: Permite añadir nuevos pasos vacíos y eliminar pasos existentes con confirmación
- **Actualización en tiempo real**: Se integra con `recipe_steps_provider` y `recipe_detail_provider` para sincronizar cambios instantáneamente

Este componente demuestra la flexibilidad de Flutter para crear interfaces complejas cuando las soluciones existentes no se ajustan a las necesidades específicas del proyecto.

7. Manuales de usuario

Manual de instalación

1- Instalación de Git: https://git-scm.com/install/windows

2- Instalación de IDE, en este caso IntelliJ: https://www.jetbrains.com/es-es/idea/download/?section=windows

3-Instalación y configuración de Flutter:

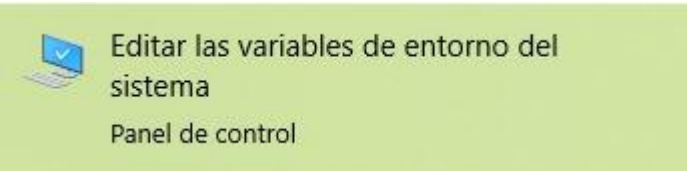
Para usarlo con VS Code: https://docs.flutter.dev/get-started/quick

Para usarlo con IntelliJ: https://docs.flutter.dev/install/manual

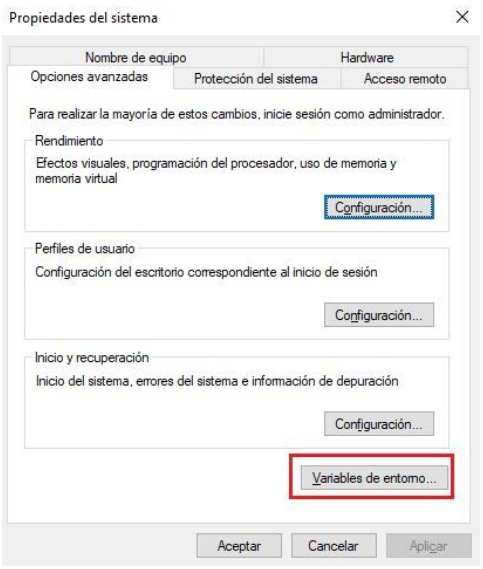
Como indican los pasos de la web, creamos una carpeta llamada develop en la ruta C:\Usuarios\{nombre_usuario}\ y ahí descomprimimos el archivo

Tan sencillo como darle click derecho al .zip que nos hemos descargado, opción de 7-Zip, extraer en, y seleccionar la carpeta develop que hemos creado

Luego abrimos las variables de entorno

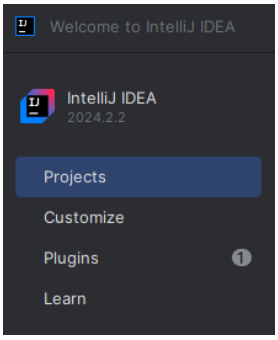


Le damos a Variables de entorno



Y en la parte de arriba donde dice variables de usuario para {usuario}, buscamos la variable Path, hacemos doble click, y en una fila vacía añadimos la ruta del bin de lo que hemos descomprimido en develop, en mi caso seria C:\Users\Formacion\develop\flutter_windows_3.35.4-stable\flutter\bin

Para comprobar que esta hecho correctamente, tenemos que abrir la consola y escribir flutter --version y dart --version, si te da error y no ha encontrado nada, hay que revisar los pasos de nuevo

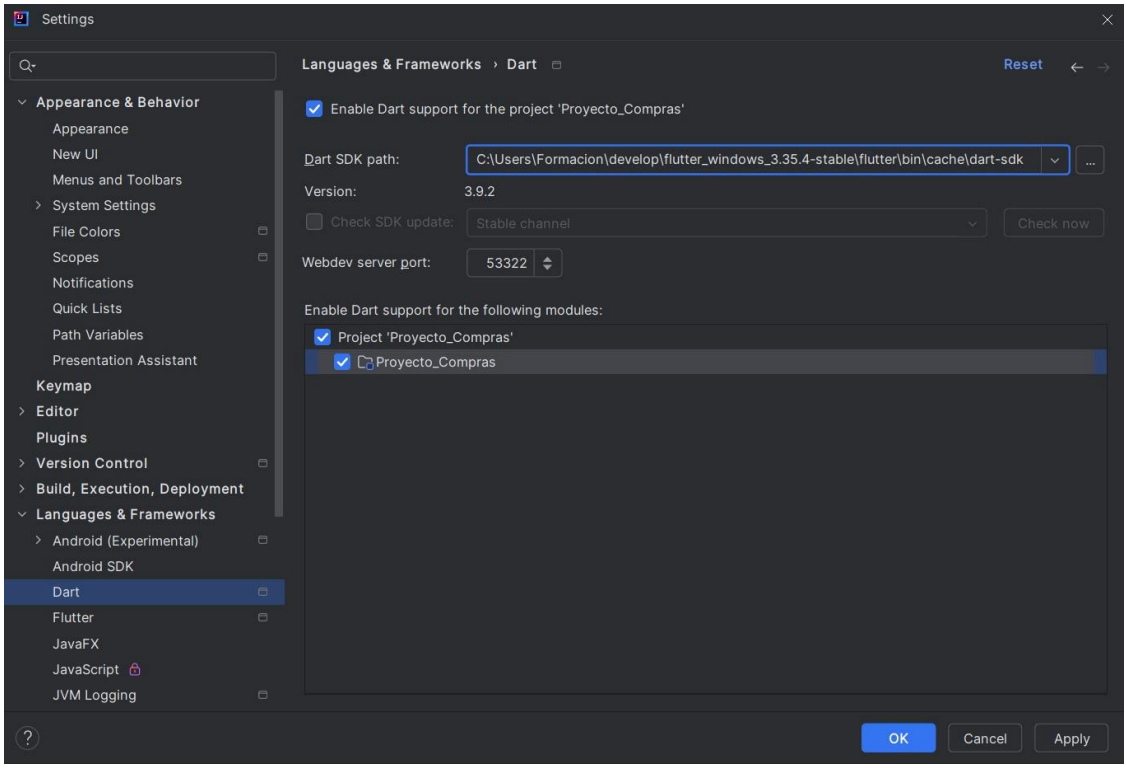


Ahora, al abrir IntelliJ nos saldrá una pestaña donde podremos ver lo siguiente

Lo primero que tenemos que hacer es darle a Plugins e instalar Flutter y Dart, después le damos a reinciar IDE.

Una vez tengamos el proyecto abierto en con IntelliJ, podemos activar el soporte de Dart en: File > Settings > Languages & Frameworks > Dart y marcamos la opción eneable dart support for the project

En la ruta del sdk esta donde hemos descomprimido la descarga manual de flutter C:\Users\Formacion\develop\flutter_windows_3.35.4-stable\flutter\bin\cache\dart-sdk



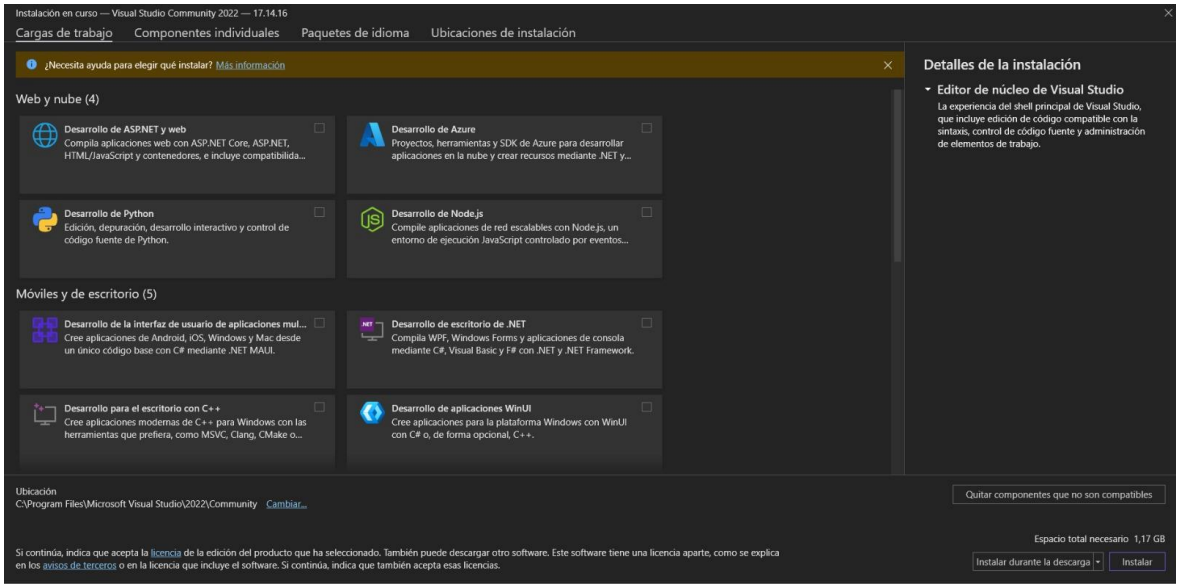
4- Instalación de Android Studio para desarrollar en Android: https://developer.android.com/studio?hl=es-419#get-android-studio

Solo tenemos que instalarlo y añadir una variable del sistema llamada ANDROID_HOME y cuyo valor sea la ruta del SDK de Android, en mi caso es:

C:\Users\Formacion\AppData\Local\Android\Sdk

5- Instalación de Visual Studio para desarrollar en Windows: https://visualstudio.microsoft.com/es/downloads/

Cuando hagamos doble click sobre el instalador nos saldrá esta pestaña, donde tendremos que seleccionar Desarrollo para el escritorio con C++ y Desarrollo de escritorio .NET



Una vez lo tengamos instalado, nos pedirá reiniciar el ordenador, y ya podríamos ejecutar el proyecto (Asegurarse de activar el modo desarrollador en el PC o ejecutar como administrador el IDE)

Es necesario crear un archivo .env en la raíz del proyecto y poner lo siguiente dentro.

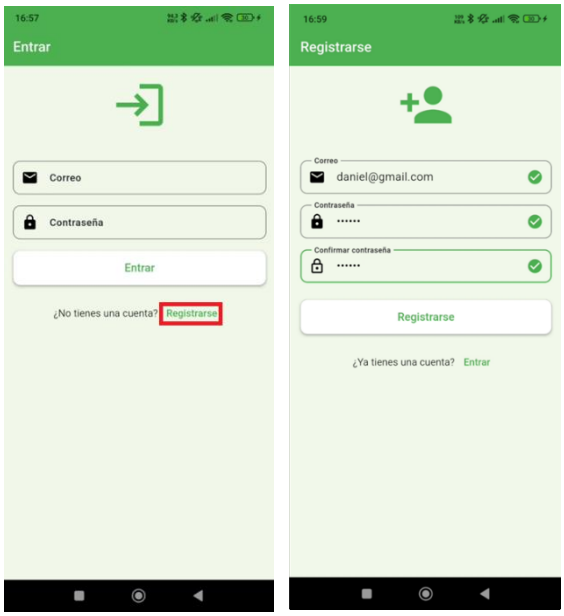
PROJECT_URL=x

API_KEY=x

Y ya tenemos el proyecto para ejecutar en nuestro ordenador.

Manual de uso

Lo primero que nos vamos a encontrar es la ventana de login, tendremos que darle a registrarse porque no tenemos cuenta, y procedemos a crearnos una cuenta

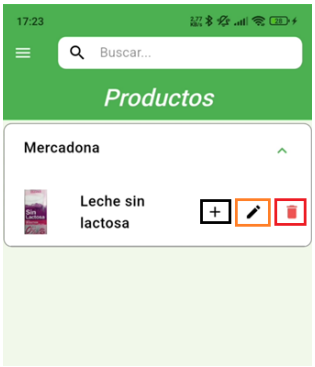


Una vez nos registremos, iniciará sesión automáticamente y veremos la pestaña de productos, con un pequeño “tutorial” de lo que podemos hacer ahí.

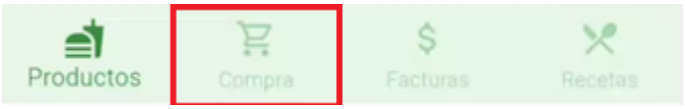
Si le damos al botón de añadir producto, nos saldrá una ventana para rellenar los datos de un producto, donde tendremos la opción de escanear un producto, o rellenar los datos manualmente.

Una vez rellenados los datos, le damos a guardar y aparecerá en la lista, en una carpeta cuyo nombre es el supermercado, donde podremos:

- Filtrar mediante barra de búsqueda
- Icono negro: Añadir producto a la lista de la compra
- Icono naranja: Editar producto
- Icono rojo: Eliminar producto

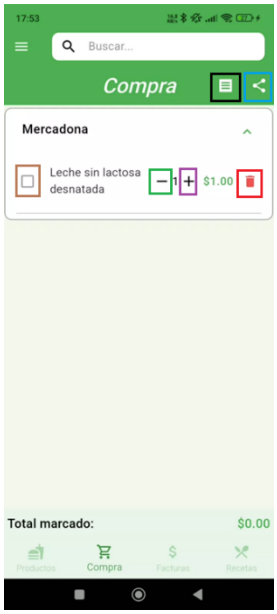


Antes de ir al siguiente apartado vamos a añadir el producto que acabamos de crear a la lista de la compra, y ya podemos navegar a la pestaña de la lista de la compra mediante la navegación inferior



Donde veremos algo muy similar a la pestaña de productos, una pequeña guía de que podemos hacer en esta ventana, en este caso como hemos añadido el producto, vamos a ver eso, en esta ventana se puede:

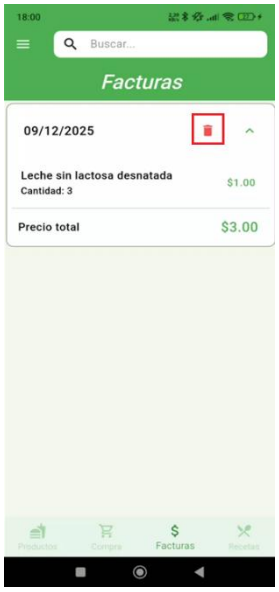
- Filtrar mediante barra de búsqueda
- Icono negro: Crear factura con los productos marcados
- Icono azul: Compartir lista de la compra
- Icono marrón: Marcar o desmarcar el producto
- Icono verde: Reducir cantidad del producto
- Icono morado: Aumentar cantidad del producto
- Icono rojo: Eliminar producto de la lista de la compra



Para ver bien la siguiente pestaña, vamos a sumar la cantidad que queramos al producto y marcarlo, una vez hecho eso, damos al botón de compartir (azul), si estamos en móvil te abrirá una lista de aplicaciones para compartir y si estamos en PC, se te copiará al portapapeles la lista de la compra.

Antes de pasar al menú de Facturas, dale al botón negro para crear una factura de los productos marcados

La pestaña de facturas es similar a las demás, un fondo con unas instrucciones básicas de lo que se puede hacer, al haber creado una factura, nos saldrá una lista de las facturas que tenemos creadas, esta pestaña es de visualización, y se puede:



- Filtrar mediante barra de búsqueda
- Icono rojo: Borrar factura

Para la última pestaña, no necesitamos hacer nada especial, navegar mediante la navegación inferior, donde veremos un pequeño tutorial con todo lo que podamos hacer:

- Filtrar mediante barra de búsqueda
- Icono morado: Importar receta mediante un código
- Icono rojo: Crear una receta

Crear una receta es muy similar a crear un producto, rellenamos datos, ponemos una foto que es opcional y lo creamos, nos saldrá en formato lista.

Al tener algo en la lista, tenemos 2 funciones nuevas:

Compartir: Compartir código para importar receta

Eliminar: Eliminar receta.

Antes de probar la función de importar receta, vamos a darle click a la receta, que nos llevará a una pestaña nueva para ver los detalles de la receta, donde podremos ver y editar:

- La foto
- El nombre
- Los pasos a seguir para preparar la receta
- Los productos vinculados a esa receta

Si le damos al botón verde de abajo a la derecha, nos saldrán 3 opciones para editar la receta:

- Editar nombre
- Editar foto
- Editar pasos

Vamos a añadir nuestro primer paso, le damos click al texto azul que dice “Para añadir tu primer paso haz click aquí”, y ya podremos ponerle título y una descripción al paso, para confirmar y guardar, daremos click al botón verde que tiene un tick abajo del todo, y para cancelar, al botón rojo con la X.

Si le damos al botón que dice “Vincular productos”, nos saldrá una lista de todos los productos que tenemos creados para poder asignarlos a esa receta, lo cual es importante para:

- Importar receta con los productos
- Añadir todos los productos (marcados) a la lista de la compra de golpe.

Por último, vamos a probar la función de importar receta, para esto necesitaremos crear una nueva cuenta, dado que hay una restricción para no poder importar nuestra propia receta, si le damos al icono de importar y ponemos el siguiente código: PKTYX6, nos saldrá una pestaña con un pequeño preview de la receta a importar.

Si le damos a importar, se importará la receta, con todos sus datos, incluyendo productos asociados que se crearán en una carpeta con el nombre de la receta, por si el usuario ya los tiene, poder diferenciarlos y que los borre o los mueva de carpeta editando el supermercado.

8. Conclusiones y posibles ampliaciones

Conclusiones

He conseguido cumplir los objetivos que me había propuesto.

La buena planificación del proyecto es clave para no tener que estar refactorizando completamente, aunque es inevitable en algún momento, especialmente si estás aprendiendo. Sin embargo, es complicado planificar bien un proyecto cuando no sabes exactamente lo que quieres hacer. Es más fácil ver cómo se hace bien cuando ya lo has hecho mal.

He conseguido anticipar problemas y automáticamente pienso soluciones que son viables de implementar y que los resuelven.

Soy consciente de que, sobre todo, la reutilización de componentes y gestión de estados puede ser mejor.

Con lo que me quedo es que, antes del proyecto, simplemente deseaba que existiesen aplicaciones para hacer X o Y, ahora pienso en cómo se podría hacer y si lo podría hacer yo.

Posibles ampliaciones

- Cambio de estructura para reutilización de componentes para mejora de rendimiento
- Automatización de precio con web scrapping
- Modo offline con sincronización posterior
- Un calendario para asignar recetas
- https://youtu.be/iLxjbJtI_2w
  Implementar acciones rápidas desde el acceso directo.

9. Bibliografía

Documentación oficial de flutter

ChatGPT y Claude

Algunas aplicaciones ya existentes que hacen cosas similares como InvenDo, Listonic, Stock e Inventario, Bring!

Videos de youtube como:

https://youtu.be/OzMs0GoVC5I

https://youtu.be/eEt6JrMuPZw

https://youtu.be/5vDq5DXXxss

https://youtu.be/G1LGOH424lo

https://youtu.be/sEID7kTP_hE

10. Anexos

GitHub

Enlace al repositorio de GitHub donde se encuentra el código fuente completo de la aplicación.

Figma

Enlace al diseño y prototipado de la aplicación realizado en Figma.

Página de documentación

Enlace a la página desplegada que contiene la documentación completa del proyecto.