# Manual Técnico – App Farmacia

## Descripción General

La App Farmacia es una aplicación móvil desarrollada con Flutter y Firebase que permite a farmacias de pequeña y mediana escala gestionar productos farmacéuticos, ventas, vencimientos y estantes. Está orientada al entorno Android y utiliza Cloudinary para el manejo de imágenes.

## Requisitos del Sistema

### Hardware

* Dispositivo Android (teléfono o tablet) con Android 10 o superior
* Laptop para desarrollo (mínimo 8GB RAM)

### Software

* Flutter SDK
* Android Studio o Visual Studio Code
* Cuenta de Firebase (plan Spark)
* Cuenta de Cloudinary para manejo de imágenes

## Instalación y Configuración

### Verificar entorno y herramientas

#### Instalar Flutter SDK

Seguir las instrucciones oficiales en: <https://docs.flutter.dev/get-started/install>

#### Verificar instalación con flutter doctor

flutter doctor

Texto

El contenido generado por IA puede ser incorrecto.

Asegúrese de que todos los elementos estén correctamente configurados, en especial:

* Flutter
* Android toolchain
* Android Studio / VS Code
* Connected device

### Clonar el repositorio

git clone <https://github.com/Young-Heredia/PR2-25-APP-FARMACIA.git>

cd app\_farmacia

### Instalar dependencias

flutter pub get

### Configurar Firebase

* Agregar archivo **firebase\_options.dart** generado desde Firebase Console
* Hay que confirmar que está incluido en **main.dart**:

await Firebase.initializeApp(

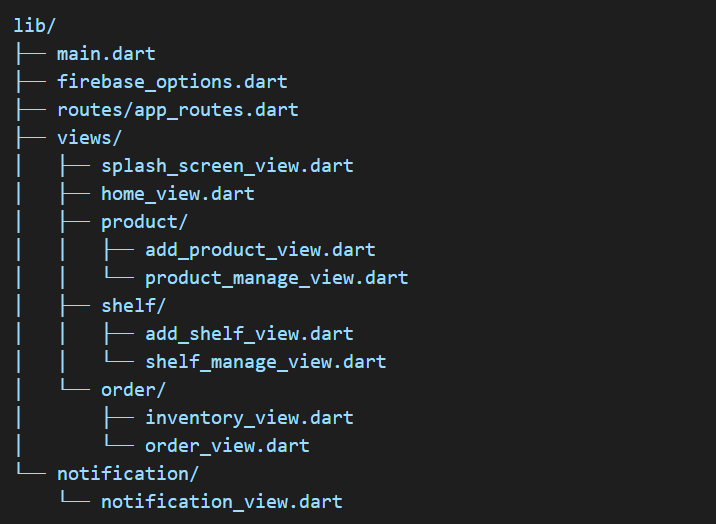
options: DefaultFirebaseOptions.currentPlatform,

);

### Ejecutar la app

flutter run

## Estructura del Proyecto



### Colecciones Principales en Firestore

#### Colecciones Principales en Firestore

**products (colección)**

Contiene el listado completo de productos farmacéuticos registrados en el sistema.

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| name | string | Nombre del producto |
| description | string | Descripción del producto |
| price | double | Precio unitario |
| stock | int | Cantidad disponible |
| imageUrl | string | URL pública de la imagen del producto (Cloudinary) |
| expirationDate | timestamp | Fecha de vencimiento |
| supplier | string | Nombre del proveedor |
| shelfId | string? | ID del estante asignado (nullable) |

**Relación:**

* Un producto puede estar asignado a **un estante** → referencia por shelfId.

**orders (colección)**

Almacena los registros de ventas, incluyendo la lista de productos vendidos, cantidad y monto total.

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| date | timestamp | Fecha y hora de la venta |
| total | double | Monto total de la venta |
| items | Array<OrderItem> | Lista de productos vendidos (ver abajo) |

**OrderItem (dentro de cada order)**

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| productId | string | ID del producto vendido |
| productName | string | Nombre del producto vendido (copia) |
| price | double | Precio unitario en el momento de la venta |
| quantity | int | Cantidad vendida |

**Relación:**

* Cada item.productId referencia indirectamente a un documento en products.

**shelves (colección)**

Define los estantes físicos/digitales a los que pueden asignarse productos.

|  |  |  |
| --- | --- | --- |
| Campo | Tipo | Descripción |
| name | string | Nombre del estante |
| description | string | Descripción del uso del estante |
| productsCount | int | Número total de productos asignados |
| assignedProducts | List<string> | Lista de IDs de productos asignados (productId) |

**Relación:**

* Un estante puede contener múltiples productos → lista assignedProducts.
* ProductModel.shelfId permite navegación inversa hacia su estante.

## Dependencias Clave (pubspec.yaml)

Texto

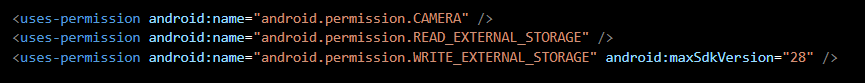
El contenido generado por IA puede ser incorrecto.

## Configuración de Splash Screen

* Implementado en **splash\_screen\_view.dart**
* Animación de carga + transición automática a /home
* Configurado también a nivel nativo en android/app/src/main/res/drawable\*/launch\_background.xml

## Permisos

AndroidManifest.xml incluye:



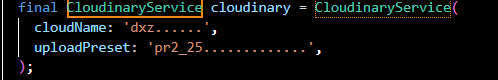
## Almacenamiento de Imágenes

### ¿Qué es Cloudinary?

Cloudinary es una plataforma en la nube para el manejo de imágenes y videos. Permite almacenar, transformar y entregar imágenes optimizadas mediante API.

### Registro y configuración de cuenta

1. Crear una cuenta gratuita en <https://cloudinary.com>
2. Acceder a tu panel de control y copiar las siguientes credenciales:
   * cloud\_name
   * upload\_preset (creado manualmente si es necesario)



### Servicio de subida de imágenes desde Flutter

El archivo lib/services/cloudinary\_service.dart implementa la lógica de subida:

Texto

El contenido generado por IA puede ser incorrecto.

### Integración en la vista de productos

* En add\_product\_view.dart y edit\_product\_view.dart, la imagen seleccionada se sube a Cloudinary automáticamente si se elige la opción de archivo.
* Luego se utiliza el secure\_url como campo imageUrl del modelo ProductModel.

### Consideraciones

* Validar el archivo antes de subir (existencia, extensión, tamaño).
* Mostrar SnackBar para indicar progreso o errores.
* Puedes usar flutter\_dotenv para ocultar tus credenciales.

## Organización del Código y Buenas Prácticas

* Comentarios explicativos en cada archivo de vista.
* Organización modular por entidades (**product**, **shelf**, **order**, etc.).
* Navegación controlada vía **app\_routes.dart**.
* Nombres semánticos y consistentes.

## Notas Finales

* Se recomienda mantener el archivo **README.md** actualizado.
* Hacer commits frecuentes y claros en el repositorio Git.
* Documentar cambios importantes en los servicios (como Cloudinary).

## Contacto Técnico

**Desarrollador Líder:** Heredia Saravia Edwin  
**Repositorio GitHub:** <https://github.com/Young-Heredia/PR2-25-APP-FARMACIA.git>   
**Correo:** [hse1010674@est.univalle.edu](mailto:hse1010674@est.univalle.edu)