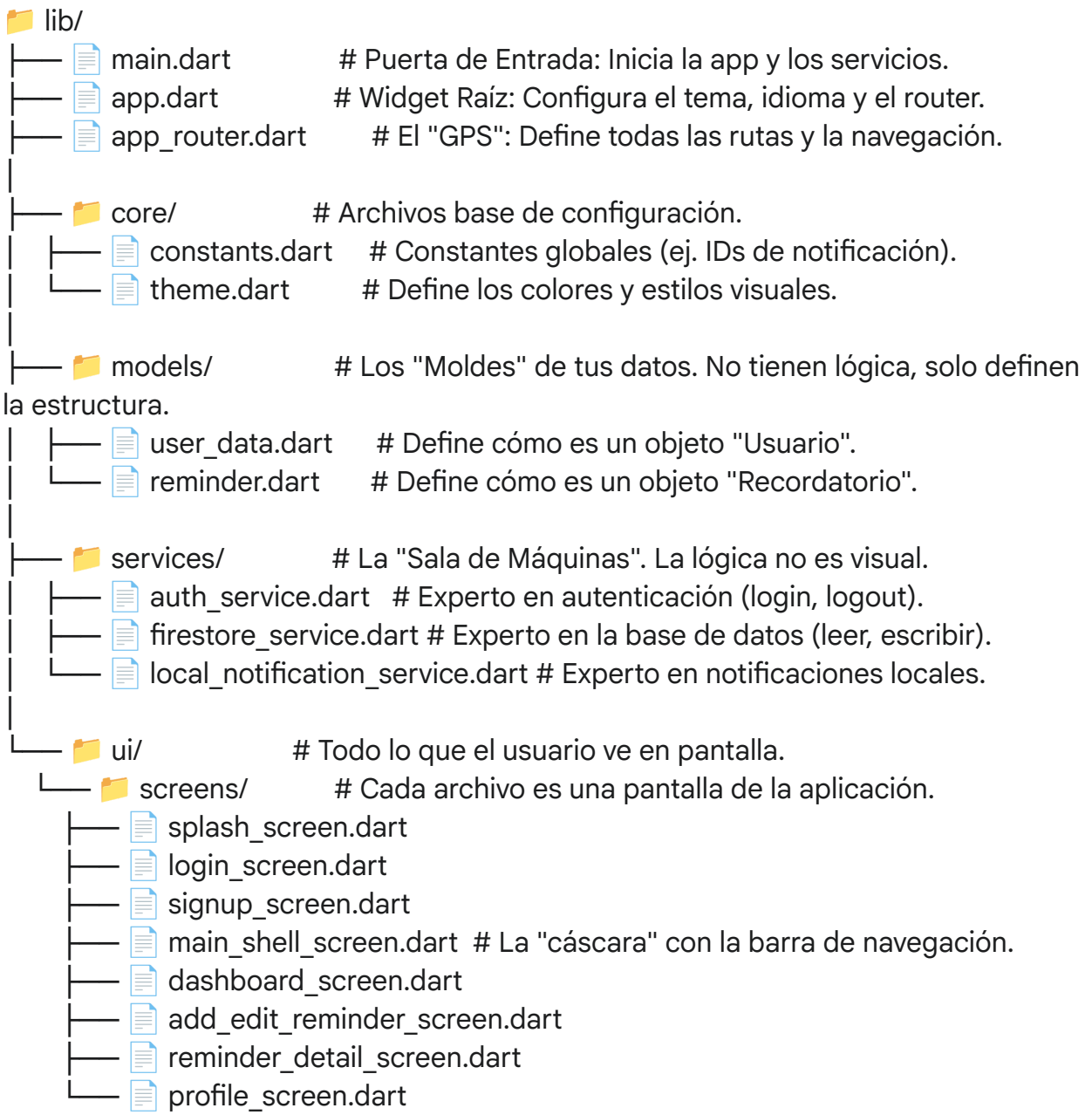
Guía de Estudio Completa: VenceYa

Este documento es tu mapa completo del proyecto. Está diseñado para que entiendas no solo qué hace cada pieza, sino cómo encajan todas juntas.

Parte 1: Estructura Gráfica y Flujo General

Así es como se organiza tu código. Cada carpeta tiene una responsabilidad clara, lo que se conoce como una **arquitectura limpia**.



El Flujo de la Aplicación (La Analogía del Restaurante)

Imagina que tu app es un restaurante de alta tecnología:

1. **main.dart (La Inauguración):** Es la puerta principal. Su única misión es "abrir", preparar al personal (Providers) y encender las luces para que el app.dart pueda empezar.
2. **app.dart (El Diseño y el Menú):** Define el ambiente del restaurante. Establece la decoración (Theme), el idioma del menú (localizations) y le entrega el control al jefe de sala (routerConfig).
3. **app_router.dart (El Jefe de Sala):** Es el cerebro de la operación. Cuando llega un usuario, el jefe de sala (redirect) mira su lista de reservas (authService). Si no tiene reserva (no está logueado), lo manda a la entrada (/login). Si tiene reserva, lo lleva a su mesa principal (/dashboard).
4. **ui/screens/ (Las Mesas):** Son los lugares donde se sientan los usuarios. Cada pantalla es una "mesa". Estas pantallas le piden cosas a los camareros (Provider). Por ejemplo, DashboardScreen le dice al camarero: "Tráeme la lista de recordatorios".
5. **services/ (Los Cocineros):** Son el personal experto que el usuario nunca ve. FirestoreService es el chef que sabe cómo leer y escribir en el libro de recetas (la base de datos). AuthService es el guardia de seguridad que comprueba las identidades.
6. **models/ (El Libro de Recetas):** Define los "ingredientes" de cada plato. reminder.dart dice: "un recordatorio debe tener un título, una fecha, etc.".

Parte 2: Análisis Detallado por Archivo

Aquí desglosamos cada archivo clave, su estructura y las posibles preguntas del profesor.

main.dart (El Arranque)

- **¿Qué Hace?:** Es el punto de entrada de toda la app. Su trabajo es inicializar los servicios más importantes (como Firebase y la localización) en el orden correcto y luego ejecutar la aplicación.
- **Estructura Gráfica:**
  1. imports (Herramientas necesarias para el arranque).
  2. main() (Lógica de inicialización).
     - WidgetsFlutterBinding.ensureInitialized()
     - initializeDateFormatting()

    - Firebase.initializeApp()
    - Creación de instancias de los servicios.
    - runApp(MultiProvider(...))
- **Preguntas del Profe:P: ¿Dónde nos conectamos a la BBDD?**
  R: La conexión inicial con todo el ecosistema de Firebase se establece aquí, en main.dart, con la línea await Firebase.initializeApp(...). Este es el primer "apretón de manos" con el servidor.

app_router.dart (El GPS)

- **¿Qué Hace?:** Define todas las rutas (URLs) de la aplicación y la lógica para protegerlas. Decide qué pantalla mostrar y cuándo redirigir al usuario (por ejemplo, si no ha iniciado sesión).
- **Estructura Gráfica:**
  1. imports (Importa todas las pantallas para poder enrutarlas).
  2. class AppRouter (Lógica de configuración del router).
     - Definición de routes (la lista de todas las URLs y a qué pantalla llevan).
     - Definición de redirect (la función "guardia de seguridad").
     - refreshListenable (el mecanismo que escucha los cambios de login/logout).
- **Preguntas del Profe:**P: ¿Qué hace el import a firebase_auth o cloud_firestore?
  R: Aunque este archivo no los importa directamente, su lógica depende de ellos. El AuthService (que sí los importa) le avisa al router cuándo el usuario inicia o cierra sesión, y el router usa esa información en su redirect para proteger las rutas.

services/auth_service.dart (El Guardia de Seguridad)

- **¿Qué Hace?:** Encapsula toda la lógica de autenticación. Es el único que habla con FirebaseAuth y GoogleSignIn. Provee métodos simples como signInWithEmailAndPassword o signOut al resto de la app.
- **Estructura Gráfica:**
  1. imports.
  2. class AuthService (Lógica pura, sin widgets).
     - Instancias de FirebaseAuth y GoogleSignIn.
     - El stream authStateChanges que avisa de los cambios de sesión.
     - Funciones para cada acción de autenticación.
- **Preguntas del Profe:**P: ¿Qué hace el import a Firebase?
  R: En este archivo, import 'package:firebase_auth/...' nos da las herramientas para crear usuarios, iniciar sesión y manejar todo lo relacionado con la identidad del usuario directamente con el servicio de autenticación de Firebase.

ui/screens/dashboard_screen.dart (La Pantalla Principal)

- **¿Qué Hace?:** Muestra la lista de recordatorios del usuario en tiempo real, organizados en pestañas.
- **Estructura Gráfica:**
  1. imports.
  2. class DashboardScreen extends StatefulWidget (La definición del widget dinámico).
  3. class _DashboardScreenState extends State<...> (La lógica y la UI).
     - Variables de estado (_tabController).
     - Lógica (initState, dispose, _showPostSignupMessage).
     - UI (@override build(...)).
- **Preguntas del Profe:**P: ¿Dónde está el widget dinámico? ¿Y la línea que lo define?
  R: Esta pantalla es un widget dinámico. La línea es class DashboardScreen extends StatefulWidget. Es dinámico porque necesita un TabController para manejar el estado de las pestañas, y este controlador necesita ser inicializado y destruido, algo que solo se puede hacer en un StatefulWidget.
  P: ¿Cuál es el "widget init"?
  R: Es el método initState(). Aquí lo usamos para crear el _tabController y para llamar a la función que revisa si hay que mostrar el mensaje de "cuenta creada".
  P: ¿Cuál es el widget de los recordatorios?
  R: El widget que dibuja la lista es ListView.builder. Y el widget que representa un solo recordatorio en esa lista es el ListTile que está dentro de un Card en el método _buildReminderList.

ui/screens/login_screen.dart (Ejemplo de Widget Dinámico)

- **¿Qué Hace?:** Provee un formulario para que el usuario inicie sesión.
- **Estructura Gráfica:**
  1. imports.
  2. class LoginScreen extends StatefulWidget.
  3. class _LoginScreenState extends State<...>
     - Variables de estado (_emailController, _passwordController, _isLoading).
     - Lógica (_signInWithEmail, dispose).
     - UI (@override build(...)).
- **Preguntas del Profe:**P: Widgets estáticos y dinámicos.
  R: Un ejemplo perfecto de widget dinámico es LoginScreen. Es StatefulWidget porque necesita "recordar" lo que el usuario escribe en los campos de texto, si debe mostrar un mensaje de error, y si está en un estado de carga (_isLoading).

Parte 3: Respuestas Rápidas para el Profe

- **Diferencia entre SQL y NoSQL y por qué la uso:**
  - **SQL** es rígido como una tabla de Excel. **NoSQL** (Firestore, la que usamos) es flexible como guardar documentos. La usamos porque es **más rápida para desarrollar** y **más fácil de escalar** para una app móvil.
- **¿Dónde se guarda el código y cómo se modifica?**
  - Todo el código fuente está en la **carpeta lib**. Se modifica abriendo el proyecto en **Visual Studio Code** y editando los archivos .dart.
- **¿Dónde se guardan las APKs generadas?**
  - Al compilar, el archivo .apk se guarda en la ruta: **build/app/outputs/flutter-apk/app-release.apk** dentro de la carpeta del proyecto.
- **¿Qué hace el import a material.dart?**
  - Es la caja de herramientas **esencial** de Flutter. Nos da todos los widgets visuales como Scaffold, AppBar, Text, Button, etc. Sin él, la pantalla estaría en negro.