Guía de Estudio Definitiva: 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**.

```
📁 lib/
├── 📄 main.dart              # Puerta de Entrada: Inicia la app y los servicios.
├── 📄 app.dart               # Widget Raíz: Configura el tema, idioma y el router.
├── 📄 app_router.dart        # El "GPS": Define todas las rutas y la navegación.
│
├── 📁 core/                  # Archivos base de configuración.
│   ├── 📄 constants.dart     # Constantes globales (ej. IDs de notificación).
│   └── 📄 theme.dart         # Define los colores y estilos visuales.
│
├── 📁 models/                # Los "Moldes" de tus datos. No tienen lógica, solo definen la estructura.
│   ├── 📄 user_data.dart     # Define cómo es un objeto "Usuario".
│   └── 📄 reminder.dart      # Define cómo es un objeto "Recordatorio".
│
├── 📁 services/              # La "Sala de Máquinas". La lógica que no es visual.
│   ├── 📄 auth_service.dart  # Experto en autenticación (login, logout).
│   ├── 📄 firestore_service.dart # Experto en la base de datos (leer, escribir).
│   └── 📄 local_notification_service.dart # Experto en notificaciones locales.
│
└── 📁 ui/                    # Todo lo que el usuario ve en pantalla.
    └── 📁 screens/           # Cada archivo es una pantalla de la aplicación.
        ├── 📄 splash_screen.dart
        ├── 📄 login_screen.dart
        ├── 📄 signup_screen.dart
        ├── 📄 main_shell_screen.dart  # La "cáscara" con la barra de navegación.
        ├── 📄 dashboard_screen.dart
        ├── 📄 add_edit_reminder_screen.dart
        ├── 📄 reminder_detail_screen.dart
        └── 📄 profile_screen.dart
```

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, una descripción, etc.".

Parte 2: Preguntas Clave del Profesor (y Dónde Encontrar la Respuesta)

Aquí están las preguntas que te hicieron, con respuestas directas y apuntando a los archivos correctos de tu proyecto.

Pregunta: Widgets estáticos y dinámicos

- **Respuesta:** Un **widget estático (StatelessWidget)** es como una foto, no cambia una vez dibujado (ej: MainShellScreen). Un **widget dinámico (StatefulWidget)** es como un video, puede cambiar su estado y apariencia para responder al usuario (ej: LoginScreen, que necesita recordar lo que se escribe en los campos).
- **Dónde Mirar:**
  - **Dinámico:** lib/ui/screens/login_screen.dart -> class LoginScreen extends

StatefulWidget.
    - **Estático:** lib/ui/screens/main_shell_screen.dart -> class MainShellScreen extends StatelessWidget.

Pregunta: Cuál es el "widget init" (initState)

- **Respuesta:** Es un método especial de los widgets dinámicos que se ejecuta **una sola vez** cuando la pantalla se crea. Lo usamos para hacer configuraciones iniciales, como cargar datos.
- **Dónde Mirar:** lib/ui/screens/dashboard_screen.dart -> Dentro de la clase _DashboardScreenState, encontrarás el método @override void initState() { ... } donde llamamos a _showPostSignupMessage() y _updateLastLogin().

Pregunta: ¿Qué hacen los import del framework?

- **Respuesta:**
    - **material.dart:** La caja de herramientas visual. Nos da Scaffold, Text, Button, etc.
    - **firebase_core.dart:** El "enchufe" principal para conectar con Firebase.
    - **provider.dart:** El "delivery" que nos permite acceder a nuestros servicios desde cualquier pantalla.
    - **intl.dart:** El "traductor" que nos permite mostrar las fechas en español.

Pregunta: ¿Por qué notificaciones locales y no push?

- **Respuesta:** Porque nuestra app es como un **despertador personal**. La principal ventaja es que **funcionan sin internet**. Una vez programada, la alerta sonará sí o sí, lo cual es fundamental para una app de vencimientos. Además, es más simple y no requiere un servidor externo.

Pregunta: Diferencia entre SQL y NoSQL y por qué la uso

- **Respuesta: SQL** es rígido como una tabla de Excel. **NoSQL** (Firestore, la que usamos) es flexible, guarda "documentos" (similares a JSON). La usamos porque es **más rápida para desarrollar y más fácil de adaptar**. Si mañana queremos añadir un campo nuevo (como hicimos con "descripción"), simplemente lo hacemos, sin reestructurar todo.

Pregunta: ¿Dónde nos conectamos a la base de datos?

- **Respuesta:** La conexión tiene dos partes: la **inicialización general** en main.dart con Firebase.initializeApp(...), y el **uso diario** a través de nuestro FirestoreService, que es el único que tiene permiso para hablar con la base de datos.
- **Dónde Mirar:** lib/main.dart y lib/services/firestore_service.dart.

Pregunta: ¿Dónde está el código y dónde se guarda el APK?

- **Respuesta:** Todo nuestro código está en la carpeta lib. Se modifica con VS Code. El APK final se genera en la carpeta build/app/outputs/flutter-apk/app-release.apk.

Pregunta: ¿Cuál es el widget de los recordatorios?

- **Respuesta:** El widget que dibuja la lista entera es el StreamBuilder dentro de DashboardScreen. El widget que representa **una sola fila** de un recordatorio es el ListTile que está dentro de un Card en la función _buildReminderList.
- **Dónde Mirar:** lib/ui/screens/dashboard_screen.dart -> _buildReminderList().