2. MANUAL TÉCNICO (PROYECTO 2º DAM)

Introducción Técnica

Chronarc es una aplicación multiplataforma programada con el framework **Flutter** y lenguaje **Dart**. Para toda la parte del servidor (base de datos y usuarios) he usado **Firebase**.

Arquitectura del Sistema

He organizado el código siguiendo una estructura limpia para no volverme loco:

- **`lib/screens/`**: Aquí están todas las vistas (Login, Home, Perfil, etc.).
- **`lib/services/`**: Aquí está la chicha. El `database_service.dart` maneja Firestore y el `weather_service.dart` conecta con una API externa.
- **`lib/widgets/`**: Componentes que reutilizo, como el menú lateral (Drawer) o la página del Códice.

Base de Datos (Firestore)

He usado una base de datos NoSQL. Solo hay una colección llamada `users` donde cada documento es el email del usuario.

- **Campos:** `essence` (int), `xp` (int), `level` (int), `avatar` (string), `unlockedCards` (array de strings).
- **Lógica de Nivel:** He programado que el nivel se calcule automáticamente: `(XP ~/ 100) + 1`. Así cada 100 puntos de XP subes un nivel.

Integración de APIs

Para darle un toque pro, la app conecta con **OpenWeatherMap**.

- Hacemos una petición HTTP GET.
- Recibimos un JSON con el clima.
- Dependiendo del ID del clima, la app cambia los rituales que te ofrece en la pantalla de inicio.

Componentes Clave del Código

1. **`StreamBuilder`**: Lo uso en casi toda la app para que, en cuanto cambie algo en la base de datos (como la esencia), la pantalla se actualice sola sin tener que refrescar.
2. **`IndexedStack`**: En la `ShellPage` lo uso para que, cuando cambies de pestaña en el menú de abajo, no se pierda lo que estabas haciendo en la otra pantalla.
3. **`FutureBuilder`**: Para la Splash Page de carga al principio.

Requisitos de Instalación

1. Tener instalado el SDK de Flutter (versión estable).
2. Configurar el proyecto en la consola de Firebase y descargar el `google-services.json` (aunque ya he dejado el `firebase_options.dart` configurado).
3. Ejecutar `flutter pub get` para bajar las dependencias (firebase_core, cloud_firestore, http, etc.).
4. Lanzar en un emulador o dispositivo real con `flutter run`.