\# 💰 **Aplicación de Seguimiento de Gastos (Proyecto Ordinario)**

Aplicación móvil de vanguardia, desarrollada en Flutter, orientada a la gestión y el monitoreo exhaustivo de las finanzas personales. Su diseño permite a los usuarios registrar, visualizar y analizar su actividad financiera de forma segura e intuitiva.

\#\# ✨ **Funcionalidades Destacadas**

* ***Mecanismos de Autenticación Robustos**\\:
* Registro y acceso mediante credenciales de correo electrónico.
* Implementación de ***Verificación por Código OTP**\\ (Contraseña de Un Solo Uso) para reforzar la seguridad durante el proceso de registro.
* Administración de sesiones a través de Supabase Auth.

* ***Administración Integral de Transacciones**\\:
* Capacidad para ingresar nuevos gastos detallando el monto, descripción, categoría, fecha y el medio de pago (Efectivo, Tarjeta, Transferencia).
* ***Edición y Eliminación**\\ eficiente de registros de gastos preexistentes.
* Presentación de la lista de gastos con paginación y mecanismos de carga optimizada.

* ***Panel de Control y Análisis Financiero**\\:
* ***Síntesis Financiera**\\: Visualización inmediata del desembolso total mensual y diario.
* ***Desglose Pormenorizado Diario**\\: Listado detallado de transacciones agrupadas por día con sus respectivos subtotales.
* ***Opciones de Filtrado Avanzado**\\: Filtrado por rangos de fechas definidos, selección de múltiples categorías y búsqueda por texto.

* ***Estándares de Diseño UI/UX Profesional**\\:
* Interfaz de usuario pulcra y contemporánea, cimentada en las directrices de Material 3.
* Incorporación de animaciones fluidas (Lottie) para indicar estados de carga y listas vacías.
* Provisión de retroalimentación visual al usuario mediante Snackbars y diálogos de confirmación.

\#\# 🛠️ **Plataforma Tecnológica Empleada**

* ***Interfaz de Usuario (Frontend)**\\: \Flutter\ (Lenguaje Dart)
* ***Infraestructura de Soporte (Backend) y Almacenamiento de Datos**\\: \Supabase\ (PostgreSQL + Módulo de Autenticación)
* ***Manejo del Estado**\\: Librería \`provider\\`
* ***Gestión de Navegación**\\: Librería \`go\\_router\\`
* ***Recursos Gráficos y Animaciones**\\: Librería \`lottie\\`
* ***Formatos de Fecha y Moneda**\\: Librería \`intl\\`

\#\# 📂 **Arquitectura del Proyecto**

El proyecto adopta una estructura arquitectónica basada en el concepto de ***Features***\\ (Módulos de Funcionalidad) con el fin de garantizar la organización y escalabilidad del código fuente:

```
\\\\ lib/ ├── core/ # Componentes transversales y configuración general | ├── constants/ # Declaración de constantes (ej. Categorías, Íconos) | ├── models/ # Definición de modelos de datos (Expense) | └── providers/ # Implementación de la lógica de negocio y gestión del estado (AuthProvider, ExpenseProvider) ├── features/ # Módulos centrales de la aplicación | ├── auth/ # Interfaces y lógica concerniente a la autenticación (Login, Registro, Verificación, Perfil de Usuario) | ├── expenses/ # Interfaces de listado, visualización detallada y formulario para el registro de gastos | └── summary/ # Interfaz para la presentación del resumen y métricas estadísticas ├── routes/ # Configuración de las rutas de navegación (GoRouter) └── main.dart # Punto de inicio y ejecución de la aplicación \\\\\
```

\#\# 🚀 Procedimiento de Configuración e Implementación

\#\#\# Requisitos Previos
* Disponer del SDK de Flutter instalado (última versión estable).
* Contar con una cuenta y un proyecto configurado en la plataforma Supabase.

\#\#\# Pasos a Seguir
1\. ***Clonar el Repositorio***\\ (o proceder a la descarga del código fuente).
2\. ***Instalación de Dependencias***\\:
\\\\bash flutter pub get \\\\\
3\. ***Configuración de Variables de Entorno***\\:
Asegurar la correcta configuración de las credenciales de Supabase (URL y Anon Key) en el archivo \lib/main.dart\\.
4\. ***Ejecución de la Aplicación***\\:
\\\\bash flutter run \\\\\

\#\# 🔐 Configuración de Supabase (Servicios de Backend)

Para asegurar la correcta operatividad del envío de códigos OTP:

1\. Acceder al Panel de Control de Supabase -\> ***Authentication***\\ -\> ***Email Templates***\\.
2\. En la plantilla denominada **"Confirm Signup"**\\, emplear la siguiente estructura en el cuerpo del correo para la remisión del código numérico:
\\`\\`\\`html
\\<h2\\>Confirmación de Cuenta\\</h2\\>
\\<p\\>Su código de seguridad es el siguiente:\\</p\\>

\\<h1\\>{{ .Token }}\\</h1\\>
\\`\\`\\`
\#\# 📱 Ilustraciones (Fines de Referencia)

* ***Acceso/Registro***: Punto de acceso seguro con validación de datos.
* ***Registro de Gastos***: Listado con opciones de filtrado horizontal presentadas en formato de "píldora".
* ***Resumen General***: Panel de control que incluye tarjetas de totales y un desglose histórico.

\---
Proyecto desarrollado en el marco del proyecto Ordinario.

Documentación Técnica: Expense Tracker App

Este documento presenta una descripción detallada de la arquitectura, la organización de directorios y las funcionalidades clave de la aplicación móvil de gestión de gastos.-----1. Arquitectura y Estructura del Proyecto

El proyecto se basa en una **arquitectura modular de Características (Features)**, donde el código se estructura según la funcionalidad específica (ej. `auth`, `expenses`) en lugar del tipo de archivo (vistas, modelos). Este enfoque facilita la escalabilidad y el mantenimiento del sistema.Estructura del Directorio Principal (`lib/`)

Directorio	Propósito	Contenido Clave
`core/`	Código esencial y compartido.	`constants/` (categorías, colores), `models/` (estructura de datos como `expense.dart`), `providers/` (lógica de negocio y gestión de estado: `auth_provider.dart`, `expense_provider.dart`).
`features/`	Módulos funcionales y pantallas específicas de usuario.	`auth/` (Login, Registro con OTP, Perfil, Splash), `expenses/` (Listado, Detalle, Formulario, Filtros), `summary/` (Dashboard/Resumen).
`routes/`	Configuración de la navegación.	`app_router.dart` (Definición de rutas usando GoRouter).

`main.dart`	Punto de entrada.	Inicialización de Supabase, Providers y configuración de temas.

-----2. Descripción de Módulos y FuncionalidadesA. Autenticación y Seguridad (`features/auth`)

Se utiliza **Supabase Auth** para la gestión segura de usuarios.

- **Verificación Inicial (`Splash Screen`):** Comprueba automáticamente el estado de la sesión para dirigir al usuario al Login o a la pantalla principal.
- **Registro con OTP:**
  - El usuario se registra con correo/contraseña.
  - Supabase envía un **código PIN de 6 dígitos** por correo.
  - El usuario debe ingresar el código en la pantalla de **Verificación** para activar la cuenta.
- **Gestión de Sesión:** Funcionalidades estándar de Inicio de Sesión y Cierre de Sesión.

B. Gestión de Gastos (`features/expenses`)

Módulo central que coordina la interfaz de usuario con la base de datos a través de `ExpenseProvider`.

- **Visualización:** Lista de gastos con soporte para scroll infinito o paginación. Incluye un **Empty State** amigable si no hay registros.
- **Herramientas de Filtrado:** Búsqueda por texto, selector de rango de fechas, y un carrusel horizontal para filtrar por categorías.
- **CRUD Completo:**
  - **Creación:** Formulario validado para registrar nuevos gastos.
  - **Lectura/Detalle:** Vista detallada de cada gasto.
  - **Actualización:** Función de edición que precarga los datos del gasto seleccionado.
  - **Eliminación:** Opción de borrado.

C. Dashboard Financiero (`features/summary`)

Proporciona una vista analítica y un resumen del comportamiento financiero del usuario.

- **Resumen de Totales:** Tarjetas destacadas que muestran el **Total de Gastos del Mes** y el **Total del Día** actual.
- **Desglose Diario:** Los gastos se agrupan por fecha, mostrando el **subtotal** consumido en cada día, facilitando la identificación de días con mayor gasto.

-----3. Flujo de Datos (Reactividad)

El ciclo de vida de los datos garantiza que la aplicación sea **reactiva** y que la interfaz esté siempre sincronizada con la fuente de datos real (Supabase).

1. **Acción del Usuario:** El usuario interactúa con la UI (ej: guarda un gasto).
2. **Invocación al Provider:** La UI llama al método correspondiente en el **Provider** (ej: `expenseProvider.addExpense(...)`).
3. **Persistencia:** El **Provider** se comunica con **Supabase** para realizar la operación de escritura o lectura.
4. **Notificación de Actualización:** Tras la respuesta exitosa de Supabase, el **Provider** actualiza su estado interno y ejecuta `notifyListeners()`.
5. **Reconstrucción de la UI:** La interfaz de usuario se **reconstruye automáticamente**, reflejando el cambio de datos (ej: el nuevo gasto aparece en la lista).

Estructura Detallada del Proyecto Ordinario

Este documento presenta la organización jerárquica de directorios y archivos que componen el código fuente de la aplicación.

c:/Ordinario/

Archivo/Carpeta	Descripción
lib/	**CÓDIGO FUENTE DE LA APLICACIÓN**
├── **core/**	**NÚCLEO: Lógica y Configuraciones Compartidas**
│ ├── **constants/**	
│ │ └── `categories.dart`	Definición de constantes: categorías (iconos, colores, nombres).
│ ├── **models/**	**MODELOS DE DATOS**
│ │ └── `expense.dart`	Estructura del objeto Gasto (incluye id, monto, fecha, etc.).
│ └── **providers/**	**GESTIÓN DEL ESTADO GLOBAL (Provider)**

│ ├── `auth_provider.dart`	Lógica de autenticación: Login, Registro y verificación OTP.
│ └── `expense_provider.dart`	Lógica CRUD (Crear, Leer, Actualizar, Borrar) y filtrado de Gastos.
├── **features/**	**MÓDULOS DE FUNCIONALIDAD (Pantallas agrupadas)**
│ ├── **auth/**	**Módulo de Autenticación**
│ │ └── **screens/**	
│ │ ├── `login_screen.dart`	Pantalla para iniciar sesión.
│ │ ├── `register_screen.dart`	Pantalla para la creación de cuentas.
│ │ ├── `verify_email_screen.dart`	Pantalla de verificación PIN (código de 6 dígitos).
│ │ ├── `profile_screen.dart`	Pantalla de Perfil de usuario.
│ │ └── `splash_screen.dart`	Pantalla de carga inicial.
│ ├── **expenses/**	**Módulo de Gastos**
│ │ ├── **screens/**	
│ │ │ ├── `expenses_list_screen.dart`	Vista principal con la lista de gastos.
│ │ │ ├── `expense_detail_screen.dart`	Vista para el detalle de un gasto específico.
│ │ │ └── `expense_form_screen.dart`	Formulario para la creación o edición de gastos.
│ │ └── **widgets/**	
│ │ └── `expenses_filters.dart`	Componente de filtros horizontales para gastos.
│ └── **summary/**	**Módulo de Resumen Financiero**
│ └── **screens/**	
│ └── `summary_screen.dart`	Dashboard con el resumen financiero.
├── **routes/**	**CONFIGURACIÓN DE NAVEGACIÓN**

`│   └── app_router.dart`	Configuración centralizada de rutas (utilizando GoRouter).
`└── main.dart`	**PUNTO DE ENTRADA** (Inicialización de la aplicación).
`pubspec.yaml`	**Dependencias** del proyecto (listado de librerías externas).
`README.md`	**Documentación general** del proyecto.