Este tarea sirve como un punto de referencia para la versión 2.2 de la aplicación ExpenseFlow. La funcionalidad de OCR ha sido desactivada en el frontend para evitar errores con la API de Google Cloud Vision, ya que la facturación no está habilitada por el momento.

A continuación, se incluye la documentación detallada del estado de la aplicación correspondiente a la versión 2.1, que sigue siendo mayormente válida para la v2.2 (con la salvedad del OCR desactivado):

**Documentación Detallada ExpenseFlow v2.1 (Base para v2.2)**

**1. Estructura General de Ficheros**

La aplicación ExpenseFlow se divide en dos partes principales: el backend (Python/FastAPI) y el frontend (TypeScript/React).

**Backend (**/app/src**)**

* main.py: (Solo lectura) Punto de entrada de la aplicación FastAPI. Carga automáticamente todos los routers de las APIs.
* requirements.txt: (Solo lectura) Lista de dependencias Python para el backend.
* app/
  + apis/: Contiene los diferentes módulos de la API.
    - auth\_google\_api/\_\_init\_\_.py: Endpoints para la autenticación con Google.
    - auth\_service/\_\_init\_\_.py: Endpoints para autenticación general y gestión de sesión.
    - company\_profile\_api/\_\_init\_\_.py: API para gestionar el perfil de la empresa (logo).
    - expense\_api/\_\_init\_\_.py: API principal para CRUD de hojas de gastos y gastos.
    - export\_service/\_\_init\_\_.py: Endpoints para exportar datos (Excel, ZIP de recibos).
    - image\_upload/\_\_init\_\_.py: API para subir imágenes de recibos a Google Drive.
    - ocr\_service/\_\_init\_\_.py: API para OCR de recibos (Google Cloud Vision) - **Actualmente desactivada en frontend.**
    - user\_api/\_\_init\_\_.py: API para la gestión de perfiles de usuario.
    - user\_deletion\_service/\_\_init\_\_.py: Servicio para la eliminación de datos de usuario.
  + auth/: (Solo lectura) Módulos de Databutton para Firebase Auth en backend.
  + libs/: Para código Python reutilizable.

**Frontend (**/app/ui**)**

* index.html: (Solo lectura) Punto de entrada HTML.
* package.json: (Solo lectura) Dependencias NPM y scripts.
* tailwind.config.js: Configuración de Tailwind CSS.
* src/: Código fuente del frontend.
  + main.tsx: (Solo lectura) Punto de entrada React.
  + router.tsx: (Solo lectura) Rutas de la aplicación.
  + app/auth/: (Solo lectura) Componentes y hooks de Databutton para Firebase Auth en UI.
  + brain/: Cliente TypeScript generado para interactuar con el backend.
    - data-contracts.ts: Interfaces TypeScript para datos de API.
  + components/: Componentes React reutilizables.
    - AddExpenseEntryForm.tsx: Formulario para añadir/editar gastos. **Llamada a OCR comentada.**
    - CreateExpenseSheetForm.tsx: Formulario para crear hojas de gastos.
    - ExpenseSheetList.tsx: Lista las hojas de gastos.
    - ui/: Componentes de UI de shadcn/ui.
  + pages/: Páginas de la aplicación.
    - App.tsx: Página principal/landing.
    - ExpenseDashboardPage.tsx: Dashboard de gastos.
    - ExpenseSheetDetailPage.tsx: Detalle de hoja de gastos.
    - UserProfilePage.tsx: Perfil de usuario.
  + utils/: Utilidades y hooks personalizados.

**2. APIs del Backend (Servicios FastAPI)**

* auth\_google\_api: Maneja flujo OAuth2 con Google (/auth/google/login, /auth/google/callback). Abierta.
* auth\_service: Gestión de autenticación general (/auth/user, /auth/logout, /auth/request-account-deletion). Mayormente protegida.
* company\_profile\_api: Gestión de perfil de empresa (POST /company/logo para subir logo). Protegida.
* expense\_api: CRUD para ExpenseSheet y ExpenseEntry. Endpoints como POST /expense-sheets, GET /expense-sheets/{id}, POST /expense-sheets/{id}/entries. Datos en Firestore. Protegida.
* export\_service: Exportaciones (POST /export/excel/upload-template, POST /export/expense-sheet/{id}/excel, GET /export/expense-sheet/{id}/receipts/zip). Usa openpyxl. Protegida.
* image\_upload: Subida de recibos a Google Drive (POST /upload/receipt-to-drive). Organiza en carpetas Gastos-YYYY-MM. Protegida.
* ocr\_service: OCR de recibos (POST /ocr/process-receipt-gcv con Google Cloud Vision). **La UI no lo llama actualmente.** Protegida.
* user\_api: Perfiles de usuario extendidos desde Firestore (GET /users/{id}/profile). Protegida.
* user\_deletion\_service: Lógica interna para eliminar datos de usuario (Firebase Auth, Firestore, Drive). No expone endpoints directos.

**3. Componentes Principales del Frontend (React)**

El frontend está construido con React, TypeScript y utiliza la librería de componentes shadcn/ui sobre Tailwind CSS. La navegación se gestiona con React Router.

* **P��ginas Principales (**ui/src/pages**)**:
  + App.tsx (Ruta /): Actúa como la página de inicio o landing page.
  + ExpenseDashboardPage.tsx (Ruta /expense-dashboard-page): Panel principal donde los usuarios ven un resumen de sus hojas de gastos. Muestra ExpenseSheetList y ExpenseSummaryStats.
  + ExpenseSheetDetailPage.tsx (Ruta /expense-sheet-detail-page): Muestra los detalles de una hoja de gastos específica, incluyendo ExpenseEntryList.
  + UserProfilePage.tsx (Ruta /user-profile-page): Permite al usuario ver/editar su perfil.
  + FirebaseSignInPage.tsx (Ruta /firebase-sign-in-page): Página para el flujo de inicio de sesión/registro con Firebase.
* **Componentes Reutilizables Clave (**ui/src/components**)**:
  + Header.tsx: Cabecera de la aplicación.
  + AddExpenseEntryForm.tsx: Dialogo modal para crear/editar gastos. Incluye subida/asociación de recibos. **La funcionalidad OCR está desactivada en el código de este componente.**
  + CreateExpenseSheetForm.tsx: Dialogo modal para crear hojas de gastos.
  + ExpenseSheetList.tsx: Muestra una lista de hojas de gastos.
  + ExpenseEntryList.tsx: Muestra una lista de gastos individuales.
  + AppProvider.tsx: Envuelve la aplicación para ThemeProvider y Toaster.
* **Gestión de Estado**:
  + Firebase para autenticación y estado de usuario (useCurrentUser).
  + Datos de Firestore se obtienen mediante APIs del backend (cliente brain).
* **Interacción con Backend**: A través del cliente TypeScript brain.

**4. Flujo de Datos Principal**

1. **Autenticación**: Firebase Auth maneja el proceso. Token JWT se usa para llamadas protegidas.
2. **Gestión de Hojas de Gastos**: UI -> expense\_api -> Firestore.
3. **Gestión de Gastos (con Recibo)**: UI -> image\_upload (para recibo a Drive) -> expense\_api (para datos del gasto a Firestore). **El paso de OCR intermedio está desactivado.**
4. **Exportación Excel**: UI -> export\_service -> Backend obtiene datos de Firestore, genera Excel con openpyxl -> Descarga en UI.
5. **Almacenamiento**:
   * Firebase Authentication: Usuarios.
   * Firestore: Perfiles, hojas de gastos, gastos.
   * Google Drive: Imágenes de recibos.
   * db.storage: Posiblemente para plantillas Excel o logo.