Corzo Zamora, Carmen Cecilia

cecicorzoxd@gmail.com

Documentación de prueba técnica – ERP/BI

Instrucciones generales:

El objetivo es crear un mini-dashboard que muestre transacciones financieras y un reporte de Business Intelligence.

Ejemplo sería registrar compras de algún tipo de inventario y la venta de parte de él. Un sistema de compras y ventas y una pestaña donde se pueda ver estadísticas gráficas de los movimientos.

Backend (Python con FastApi)

Instrucciones:

Utilizar Python con FastAPI.

Los datos pueden almacenarse en memoria (una lista de diccionarios) o en un archivo JSON. No es necesaria una base de datos real para esta prueba.

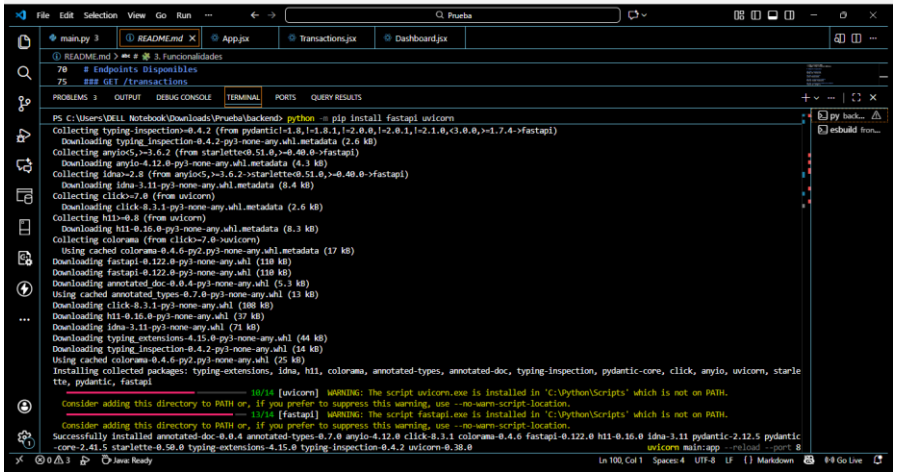
1. El modelo de la transacción debe tener (al menos):
   - id: int
   - date: str (Ej: "2025-11-07")
   - description: str
   - amount: float (usar números positivos para ingresos, negativos para egresos)
   - category: str (Ej: "Ventas", "Salarios", "Marketing")
2. Crear los siguientes endpoints:
   - POST /transactions: Agrega una nueva transacción.
   - GET /transactions: Lista todas las transacciones.
   - GET /transactions/summary: (Endpoint de analítica) Debe retornar un resumen de los datos, como:
     JSON {
     "total_income": 15000.00,
     "total_expense": -8000.00,
     "net_total": 7000.00
     }

Bonus (Opcional):

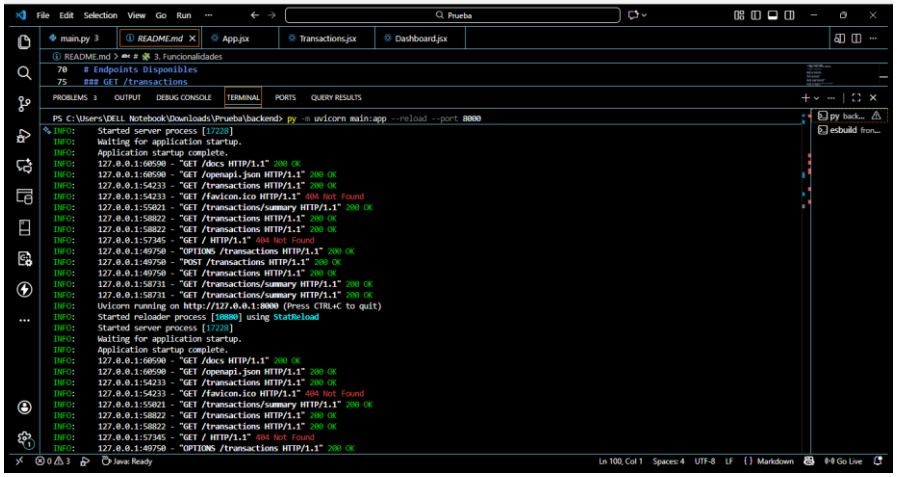
- Validación de datos usando Pydantic.
- GET /transactions/summary_by_category: Un endpoint que agrupe los totales por categoría.

Los pasos a seguir fueron:

1. **Instalar dependencias:**
   - Abrir el backend
   - Crear un entorno virtual: **py -m venv venv**
   - Instalar FastAPI y Uvicorn con: **python -m pip install fastapi uvicorn**

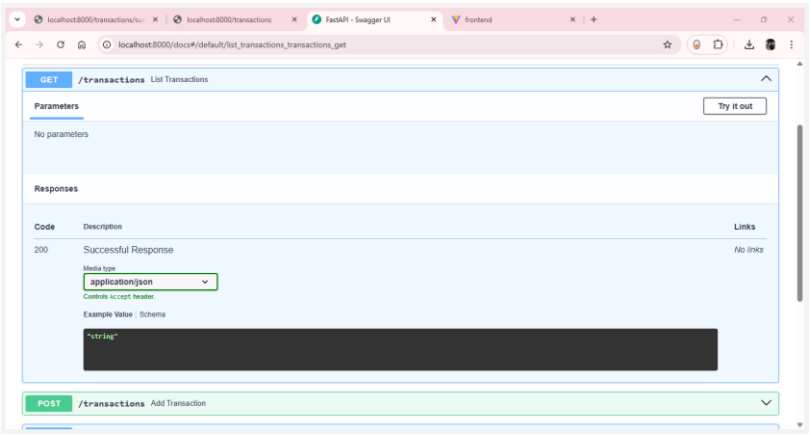


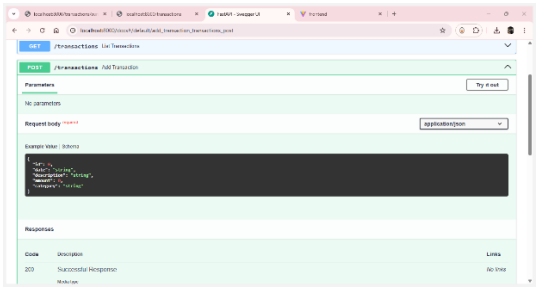
2. **Ejecutar el servidor (necesario NO estar en venv)**
   - En la terminal levantar server: **py -m uvicorn main:app --reload --port 8000**

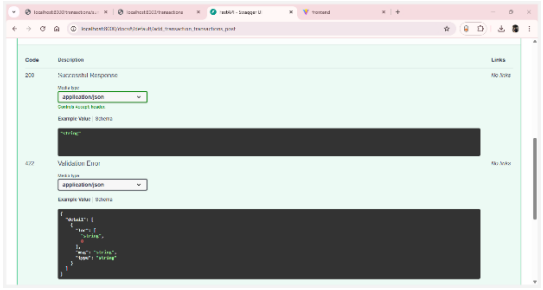


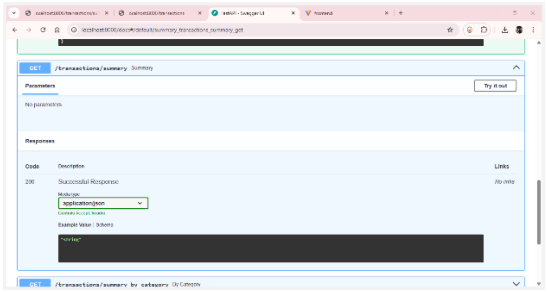
3. **Encontrar backend en los links:**
   - http://localhost:8000
   - Swagger: http://localhost:8000/docs

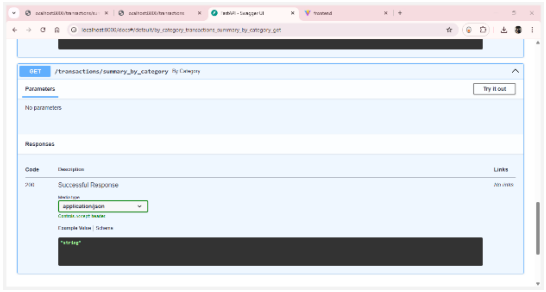
http://localhost:8000/docs#/











Explicación:

- **POST /transactions:** Agregar una transacción
- **GET /transactions:** Lista de transacciones
- **GET /transactions/summary:** Retorna el resumen financiero
- **GET /transactions/summary by category:** Totales por categoria

Frontend (React)

Instrucciones:

Usar React con functional components y hooks (useState, useEffect).

1. Crear dos secciones o "páginas" (puedes usar un simple router o solo pestañas):
   - Sección "Transacciones":
     - Un formulario simple para agregar nuevas transacciones
     - (consumiendo POST /transactions).
     - Una tabla o lista que muestre todas las transacciones
     - (consumiendo GET /transactions).
   - Sección "Dashboard":
     - Mostrar los datos del resumen (Total Ingresos, Total Egresos) consumiendo el endpoint GET /transactions/summary del backend.
     - Embeber un reporte de Looker Studio.
2. Requisito Específico (Looker Studio):
   Para esta prueba, no necesitas crear un reporte. Simplemente embebe este reporte público de ejemplo usando un <iframe>:
   - URL de ejemplo para embeber:

   HTML

   https://lookerstudio.google.com/embed/reporting/0B5FF6JBKbLCSaW1VbDl2S3prYUE/page/6SXD

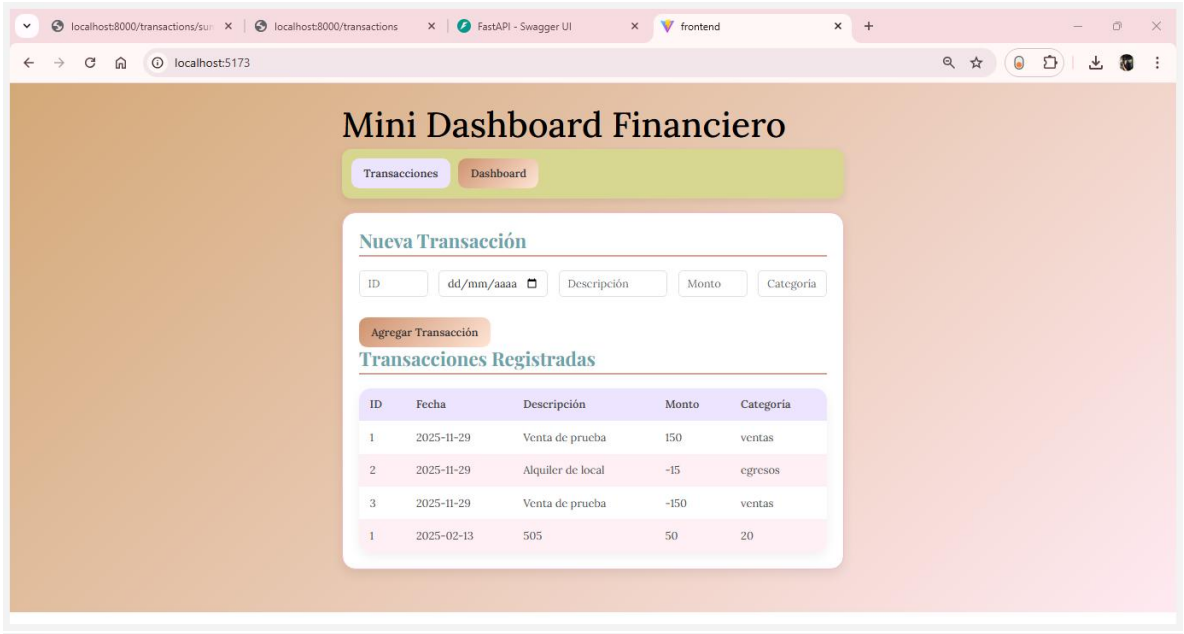
   El objetivo es demostrar que puedes integrar un reporte externo de BI dentro de la aplicación de React.

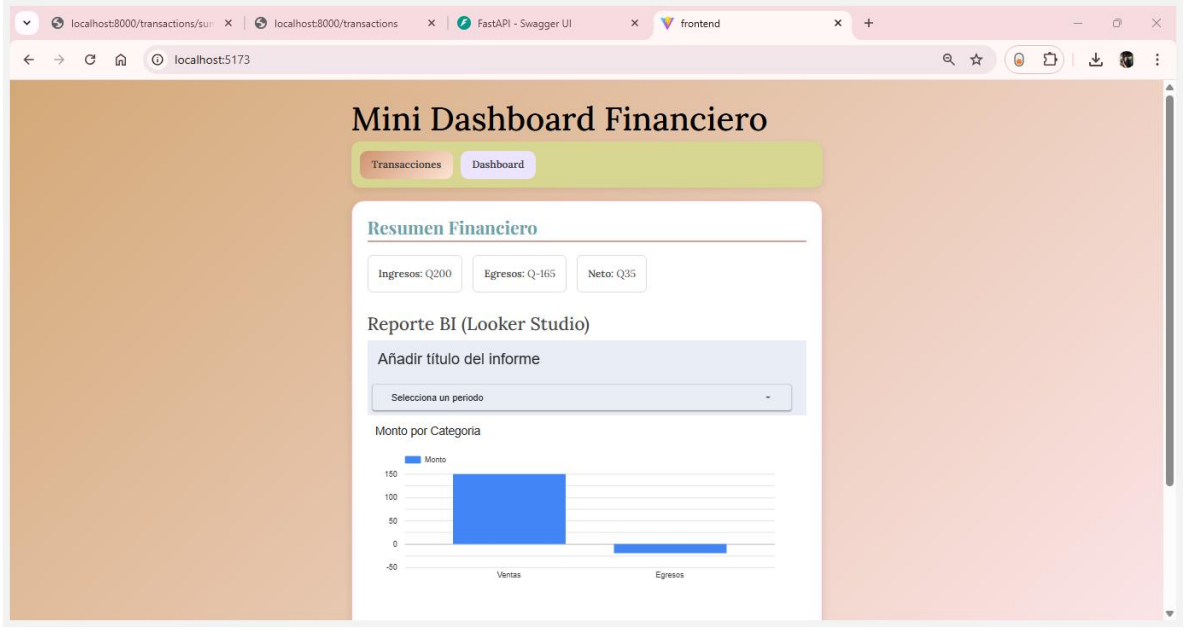
Bonus (Opcional):

- Usar una librería de componentes simple (como Bootstrap o Tailwind CSS) para que se vea ordenado.
- Mostrar un mensaje de "Cargando..." mientras se obtienen los datos.

Los pasos a seguir fueron:

1. **Instalar dependencias para node**
   - En la terminal: npm install
2. **Ejecutar el servidor de desarrollo**
   - En la terminal: cd "C:\Users\DELL Notebook\Downloads\Prueba\frontend"
   - Luego copilar: npm run dev
3. **Frontend queda disponible en:**
   http://localhost:5173





IMPORTANTE:
PARA ESTA PRUEBA SE UTILIZO VISUAL STUDIO CODE