

Corzo Zamora, Carmen Cecilia

[cecicorzoxd@gmail.com](mailto: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  
}

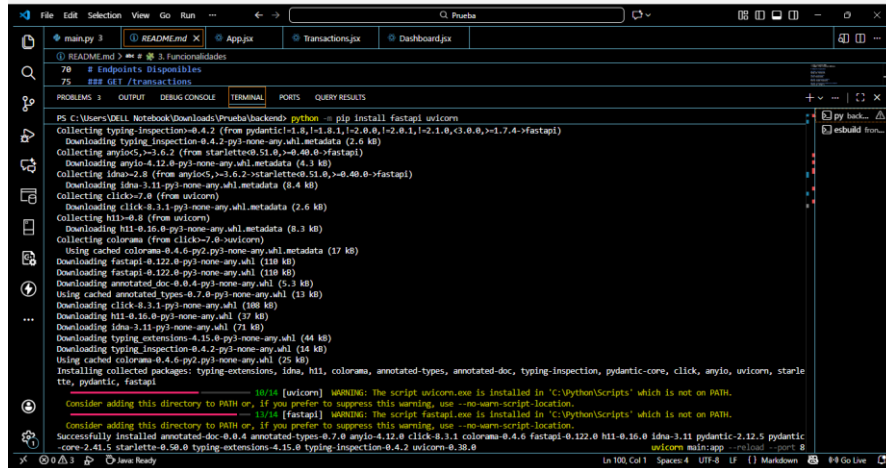
#### Bonus (Opcional):

- 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:

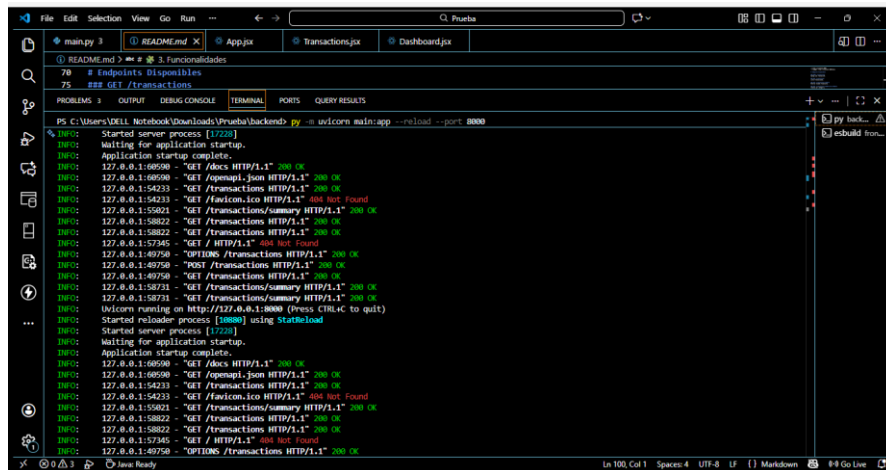
- Abrir el backend
- Crear un entorno virtual: **py -m venv venv**
- Instalar FastAPI y Uvicorn con: **python -m pip install fastapi uvicorn**



```
PS C:\Users\DELL\Notebook\Downloads\Vueba\backend> python -m pip install fastapi uvicorn
Collecting typing-inspection<0.4.2 (from pydantic<1.8.1,!=1.8.1,!=2.0.0,!=2.0.1,!=2.1.0,!=3.0.0,!=1.7.4->fastapi)
  Downloading typing_inspection-0.4.2-py3-none-any.whl.metadata (2.6 kB)
Collecting anyio<5,>=3.6.2 (from starlette<0.51.0,!=0.40.0->fastapi)
  Downloading anyio-4.12.0-py3-none-any.whl.metadata (4.3 kB)
Collecting idna<3.11-py3-none-any.whl.metadata (8.4 kB)
Collecting click<7.8 (from uvicorn)
  Downloading click-8.3.1-py3-none-any.whl.metadata (2.6 kB)
Collecting h11<0.16.0-py3-none-any.whl.metadata (8.3 kB)
Collecting colorama (from click<7.8->uvicorn)
  Using cached colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading fastapi-0.122.0-py3-none-any.whl (110 kB)
Downloading annotated-doc-0.0.4-py3-none-any.whl (5.3 kB)
Using cached annotated-types-0.7.0-py3-none-any.whl (13 kB)
Downloading click-8.3.1-py3-none-any.whl (108 kB)
Downloading h11-0.16.0-py3-none-any.whl (17 kB)
Downloading idna-3.11-py3-none-any.whl (71 kB)
Downloading typing_extensions-4.15.0-py3-none-any.whl (44 kB)
Downloading typing_inspection-0.4.2-py3-none-any.whl (14 kB)
Using cached colorama-0.4.6-py2.py3-none-any.whl (15 kB)
Installing collected packages: typing-extensions, idna, h11, colorama, annotated-types, annotated-doc, typing-inspection, pydantic-core, click, anyio, uvicorn, starlette, pydantic, fastapi
Successfully installed annotated-doc-0.0.4 annotated-types-0.7.0 anyio-4.12.0 click-8.3.1 colorama-0.4.6 fastapi-0.122.0 h11-0.16.0 idna-3.11 pydantic-2.12.5 pydantic-core-2.41.5 starlette-0.58.0 typing-extensions-4.15.0 typing-inspection-0.4.2 uvicorn-0.38.0
WARNING: The script uvicorn.exe is installed in 'C:\Python\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
WARNING: The script fastapi.exe is installed in 'C:\Python\Scripts' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
```

### 2. Ejecutar el servidor (necesario NO estar en venv)

- En la terminal levantar server: **py -m uvicorn main:app --reload --port 8000**



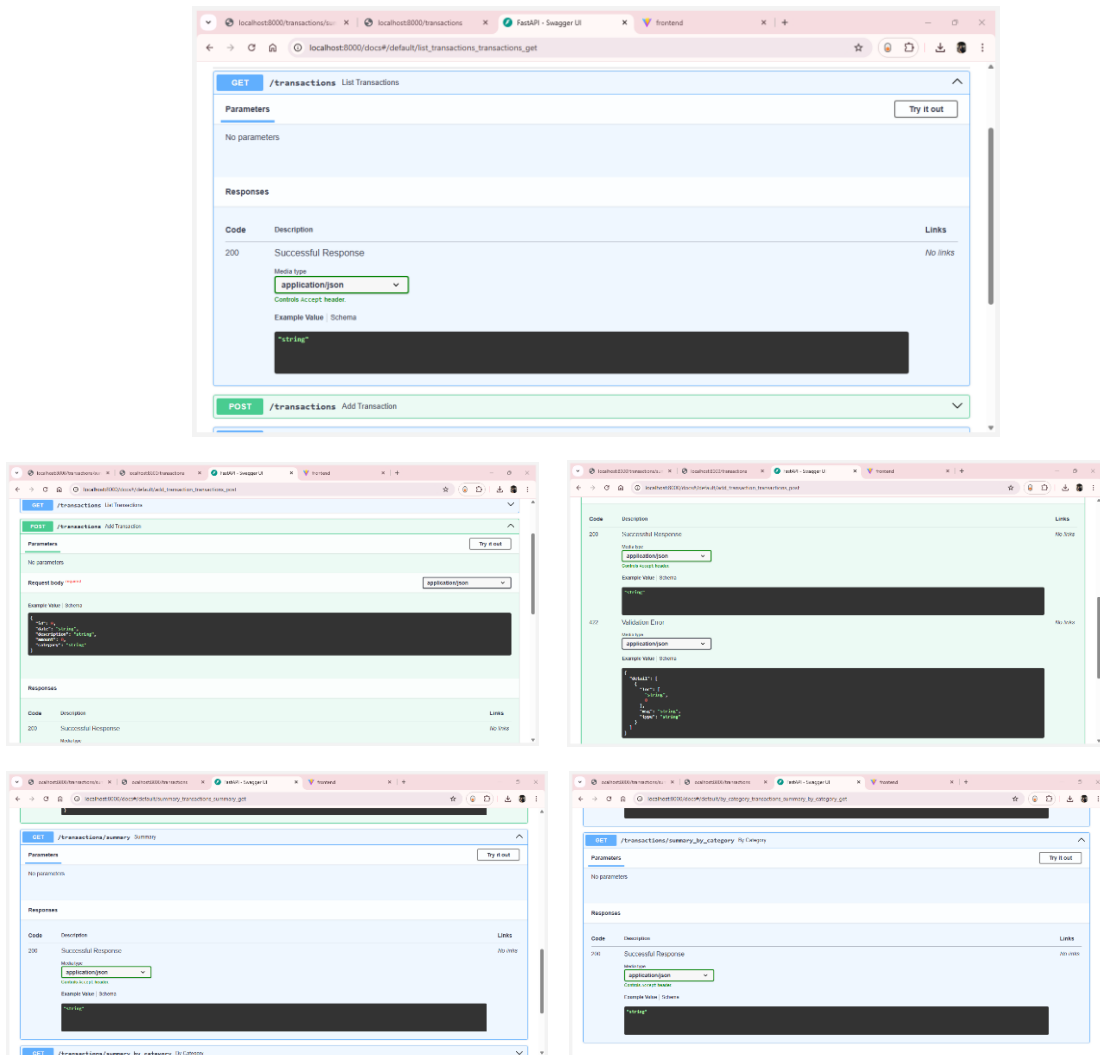
```
PS C:\Users\DELL\Notebook\Downloads\Vueba\backend> py -m uvicorn main:app --reload --port 8000
INFO: Started server process [17228]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:40950 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:40950 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:54213 - "GET /transactions HTTP/1.1" 200 OK
INFO: 127.0.0.1:54213 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:55021 - "GET /transactions/summary HTTP/1.1" 200 OK
INFO: 127.0.0.1:58822 - "GET /transactions HTTP/1.1" 200 OK
INFO: 127.0.0.1:58822 - "GET /transactions HTTP/1.1" 200 OK
INFO: 127.0.0.1:57345 - "GET / HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:40950 - "OPTIONS /transactions HTTP/1.1" 200 OK
INFO: 127.0.0.1:40950 - "POST /transactions HTTP/1.1" 200 OK
INFO: 127.0.0.1:40950 - "GET /transactions HTTP/1.1" 200 OK
INFO: 127.0.0.1:52731 - "GET /transactions/summary HTTP/1.1" 200 OK
INFO: 127.0.0.1:54731 - "GET /transactions/summary HTTP/1.1" 200 OK
INFO: Uvicorn running on http://127.0.0.1:8000 (Press Ctrl+C to quit)
INFO: Started reload process [18080] using Starlette
INFO: Started server process [17228]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:40950 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:40950 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:54213 - "GET /transactions HTTP/1.1" 200 OK
INFO: 127.0.0.1:54213 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:55021 - "GET /transactions/summary HTTP/1.1" 200 OK
INFO: 127.0.0.1:58822 - "GET /transactions HTTP/1.1" 200 OK
INFO: 127.0.0.1:58822 - "GET /transactions HTTP/1.1" 200 OK
INFO: 127.0.0.1:57345 - "GET / HTTP/1.1" 404 Not Found
INFO: 127.0.0.1:40950 - "OPTIONS /transactions HTTP/1.1" 200 OK
```

### 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 categoría

## 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/0B5FF6JBKbLCSaW1VbDI2S3prYUE/page/6SXD>

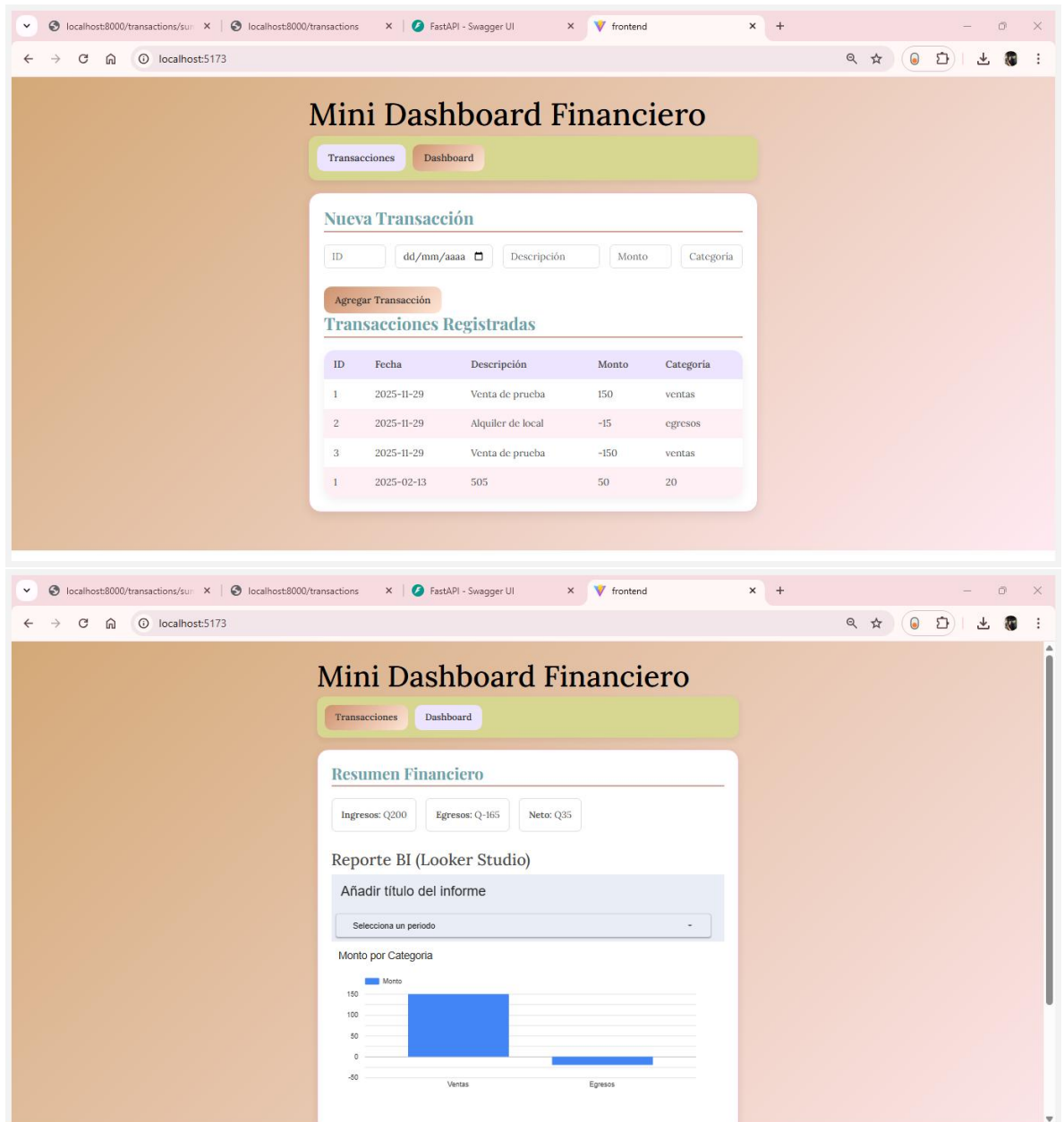
El objetivo es demostrar que puedes integrar un reporte externo de BI dentro de la aplicación de React.

#### 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 copiar: npm run dev
3. **Frontend queda disponible en:**  
<http://localhost:5173>



**IMPORTANTE:**  
PARA ESTA PRUEBA SE UTILIZO VISUAL STUDIO CODE