Personal Finance Dashboard (FastAPI + Streamlit)

Plataforma de **gestión de finanzas personales** con frontend en **Streamlit** y backend en **FastAPI**. Incluye 6 vistas exigidas por la guía: Resumen, Gastos, Presupuesto, Patrimonio, Inversiones y Metas.

---

1) Arquitectura

```
[ Streamlit ]  ↔  [ FastAPI ]  ↔  [ CSVs (datos sintéticos + Kaggle) ]
```

- **Frontend:** `Streamlit` (interactivo, Altair/Plotly, caché de consultas a API). - **Backend:** `FastAPI` (endpoints REST para KPIs, agregaciones y series). - **Datos:** CSV generados para cubrir todas las vistas + dataset de clientes de tarjeta (Kaggle).

---

2) Estructura del proyecto

```
FINANCE_DASHBOARD/
├─ backend/
│  ├─ main.py
│  ├─ requirements.txt
│  └─ data/
│     ├─ transactions.csv
│     ├─ budgets.csv
│     ├─ net_worth.csv
│     ├─ investments_holdings.csv
│     ├─ investments_prices.csv
│     └─ goals.csv
├─ frontend/
│  ├─ app.py
│  ├─ requirements.txt
│  └─ .streamlit/
│     └─ secrets.toml
└─ (opcional) README.md
```

---

3) Requisitos

- **Python 3.10+** (probado con 3.12)
- **Pip**

- (Recomendado) **Entorno virtual** `venv`

---

4) Instalación

4.1 Crear y activar entorno virtual

Windows (PowerShell):

```
python -m venv .venv
.venv\Scripts\activate
# Si PowerShell bloquea scripts:
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process
```

4.2 Instalar dependencias

```
pip install -r backend/requirements.txt
pip install -r frontend/requirements.txt
```

4.3 Configurar variables del frontend

Crea `frontend/.streamlit/secrets.toml` con:

```
API_URL = "http://127.0.0.1:8000"
```

> Para despliegue en la nube, cambia la URL por la pública del backend.

---

5) Ejecución en local

5.1 Backend (FastAPI)

```
cd backend
uvicorn main:app --reload
```

- Docs interactivas: **http://127.0.0.1:8000/docs**

5.2 Frontend (Streamlit)

En otra terminal:

```
cd frontend
streamlit run app.py
```

- App: **http://localhost:8501**

5.3 Pruebas rápidas de API (cURL)

```
curl http://127.0.0.1:8000/summary
curl "http://127.0.0.1:8000/expenses_donut?month=2025-06"
```

---

6) Mapeo a los 6 requerimientos

1. **Resumen financiero** (`/summary`)
2. KPIs: ingresos, gastos, neto del mes, patrimonio actual, efectivo, valor de inversiones.
3. **Gráfico de cascada** ingresos vs. gastos (mes seleccionado).
4. **Análisis de gastos** (`/expenses_donut` , `/top_expenses`)
5. **Dona** por categoría.
6. **Tabla** Top N gastos del mes.
7. **Seguimiento de presupuesto** (`/budget_progress`)
8. Medidores por categoría: gasto vs. límite (semáforo: verde ≤80, amarillo 80–100, rojo >100).
9. **Evolución del patrimonio neto** (`/net_worth_series`)
10. **Línea** con efectivo, inversiones y patrimonio.
11. KPIs del último mes y **tabla** con Δ y %Δ mensuales + descarga CSV.
12. **Dashboard de inversiones** (`/investments_history` , `/investments_alloc`)
13. Resumen del valor actual y **retorno acumulado**.
14. **Línea** del valor histórico.
15. **Treemap** de asignación por activo.
16. KPIs y **retorno mensual (%)** con barras verde/rojo + tabla y descarga CSV.
17. **Metas y ahorros** (`/goals`)
18. Lista de metas con **barra de progreso** y métricas.

---

7) Endpoints del backend

- `GET /summary?month=YYYY-MM` – KPIs + datos de cascada del mes.
- `GET /expenses_donut?month=YYYY-MM` – agregados de gastos por categoría.
- `GET /top_expenses?month=YYYY-MM&n=10` – Top N gastos del mes.
- `GET /budget_progress?month=YYYY-MM` – gasto vs límite (por categoría).
- `GET /net_worth_series` – serie mensual: cash, inversiones, patrimonio.
- `GET /investments_history` – valor del portafolio y retorno acumulado.
- `GET /investments_alloc` – asignación por activo (valor y % peso).
- `GET /goals` – metas (objetivo, ahorro actual, % progreso, fecha).

- `GET /transactions?month=YYYY-MM&limit=200` – movimientos crudos del mes.

---

8) Datasets incluidos

- `transactions.csv` – Movimientos diarios de ingresos/gastos (2024-04 → 2025-06). Campos: `date,type,category,amount,description,month`.
- `budgets.csv` – Límites mensuales por categoría.
- `net_worth.csv` – Serie mensual: `net_cash_flow,cumulative_cash,value,net_worth`.
- `investments_holdings.csv` – Posiciones (activos/unidades) del portafolio.
- `investments_prices.csv` – Precios mensuales sintéticos por activo.
- `goals.csv` – Metas de ahorro: objetivo, monto meta, ahorro actual, fecha.

Puedes reemplazar estos CSV por tus datos reales; mantén los mismos nombres de columnas.

---

9) Despliegue en la nube

9.1 Backend en Render (free tier)

1. Subir el repo a GitHub.
2. Crear servicio **Web Service** en Render → conectar repo.
3. **Build Command:** `pip install -r backend/requirements.txt`
4. **Start Command:** `uvicorn main:app --host 0.0.0.0 --port 10000`
5. **Root Directory:** `backend`
6. Añadir los CSV en `backend/data/` dentro del repo.
7. Al terminar, Render te dará una URL pública tipo `https://tu-api.onrender.com`.

9.2 Frontend en Streamlit Community Cloud

1. Entra a share.streamlit.io y conecta tu repo.
2. **Main file path:** `frontend/app.py`
3. **Requirements file:** `frontend/requirements.txt`
4. En **Secrets** del proyecto (en la consola de Streamlit Cloud), define:

```
API_URL = "https://tu-api.onrender.com"
```

5. Deploy. La app quedará pública con URL propia.

---

10) Troubleshooting

- **Error** `secrets.toml` **no encontrado**: crea `frontend/.streamlit/secrets.toml` con `API_URL`. En local usa `http://127.0.0.1:8000`.

- **CORS / conexión API fallida**: verifica que FastAPI esté corriendo y que `API_URL` apunte a la ruta correcta. CORS está habilitado a `*` en `main.py`.
- **Gráficos vacíos**: confirma que los CSV tienen filas para el mes elegido. Cambia el selector de mes en la barra lateral.
- **Windows PowerShell bloquea el venv**: usa `Set-ExecutionPolicy RemoteSigned -Scope Process` y vuelve a activar el entorno.

---

11) Licencia y créditos

- Uso académico. Datos sintéticos generados para cubrir los requerimientos del proyecto.
- Arquitectura y código base por Alexander & ChatGPT.

---

12) Checklist de entrega

- [x] 6 vistas implementadas según la guía.
- [x] API documentada en `/docs`.
- [x] Capturas de cada vista.
- [x] README con pasos de instalación y despliegue.
- [x] Repo en GitHub con carpetas `backend/` y `frontend/`.

¡Éxitos en la socialización! Si quieres, puedo añadir aquí mismo un apartado con **capturas** de cada vista una vez tengas la app corriendo en local o en la nube.