# Proyecto: Mini API REST con Consumo en Streamlit

## Introducción

En este proyecto desarrollaremos una aplicación compuesta por dos partes principales: una API REST creada con FastAPI y una interfaz interactiva desarrollada con Streamlit que consumirá los datos proporcionados por la API. Este enfoque no solo permite practicar el diseño y consumo de APIs, sino también desarrollar habilidades en la presentación y visualización de datos de manera amigable para el usuario.

## Objetivos

1. Diseñar una API REST funcional que proporcione datos ficticios.  
2. Implementar una aplicación en Streamlit para consumir los datos de la API y mostrarlos de manera interactiva.  
3. Practicar conceptos clave como el uso de endpoints, peticiones HTTP y visualización de datos.  
4. Mejorar la comprensión y aplicación de herramientas modernas de desarrollo web y científico.

## Tecnologías Utilizadas

- \*\*Python:\*\* Lenguaje principal para la implementación.  
- \*\*FastAPI:\*\* Framework ligero y rápido para desarrollar la API REST.  
- \*\*Uvicorn:\*\* Servidor ASGI para ejecutar la API.  
- \*\*Streamlit:\*\* Framework para construir aplicaciones web interactivas.  
- \*\*Requests:\*\* Biblioteca para realizar peticiones HTTP desde Streamlit.  
- \*\*Pandas y Matplotlib:\*\* Análisis y visualización de datos en la interfaz.

## Alcance del Proyecto

1. \*\*Creación de la API:\*\*  
 - \*\*Datos ficticios:\*\* Simularemos información sobre usuarios y transacciones financieras.  
 - \*\*Endpoints:\*\*  
 - `/users`: Retornará una lista de usuarios con detalles como nombre, edad y ubicación.  
 - `/transactions`: Proporcionará un historial de transacciones con información sobre el monto, la fecha y el usuario asociado.  
  
2. \*\*Aplicación de Streamlit:\*\*  
 - \*\*Consumo de datos:\*\* La app realizará peticiones a los endpoints para obtener y procesar los datos.  
 - \*\*Visualización:\*\*  
 - Tabla interactiva para explorar los datos.  
 - Gráficos dinámicos, como histográmicas o barras, para representar los datos.  
 - Métricas clave como el gasto promedio por usuario.  
  
3. \*\*Interacción:\*\*  
 - Capacidad de filtrar los datos según diferentes criterios (p. ej., transacciones por rango de fechas).  
 - Diseño simple y limpio para que cualquier usuario pueda entender y utilizar la herramienta.

## Fases del Desarrollo

1. \*\*Diseño de la API:\*\*  
 - Crear la estructura de FastAPI.  
 - Definir los datos ficticios y las respuestas de los endpoints.  
 - Probar la API localmente usando Uvicorn y herramientas como Postman o cURL.  
  
2. \*\*Desarrollo de la Aplicación Streamlit:\*\*  
 - Diseñar la estructura base de la app.  
 - Implementar funciones para consumir la API y procesar los datos.  
 - Crear componentes visuales para presentar los datos.  
  
3. \*\*Integración y Pruebas:\*\*  
 - Conectar la API con la aplicación de Streamlit.  
 - Realizar pruebas end-to-end para asegurar el correcto funcionamiento.  
  
4. \*\*Optimización:\*\*  
 - Mejorar la experiencia de usuario.  
 - Refinar el diseño visual y la organización de los datos en la interfaz.

## Entregables

1. Código fuente de la API y la aplicación Streamlit.  
2. Documentación que describa:  
 - Uso de la API (endpoints, formatos de respuesta, etc.).  
 - Instrucciones para ejecutar la aplicación localmente.  
3. Un despliegue funcional de la API y la aplicación (opcional en una plataforma como Heroku o Streamlit Cloud).

## Próximos Pasos

1. Configurar el entorno de desarrollo.  
2. Implementar los endpoints de la API.  
3. Diseñar la base de la aplicación Streamlit.  
4. Integrar ambas partes y realizar pruebas.