API de Análisis de Rendimiento de E-Commerce

1. Objetivo del Proyecto

El objetivo es crear una API REST profesional utilizando **FastAPI** que sirva como una capa de **Inteligencia de Negocio (BI)**. La API debe ser capaz de:

1. Gestionar la **Autenticación** y **Autorización** de usuarios (usando SQLAlchemy y JWT).
2. Obtener datos brutos de una fuente externa (**Google Sheets**).
3. Procesar y analizar los datos con **Pandas** para calcular métricas clave de e-commerce.
4. Exponer los resultados limpios en formato JSON para ser consumidos por un **Dashboard Web** y **Power BI**.

2. Arquitectura de Capas Híbrida

El proyecto sigue el patrón **Router -> Service -> Repository** y utiliza una doble capa de persistencia:

Capa	Herramienta	Rol
Auth Repository	SQLAlchemy	Persistencia de Usuarios (User, Password Hash).
Data Repository	Pandas / gspread	Obtención y limpieza de datos brutos de Google Sheets.
User Service	Python / bcrypt	Lógica de Login, Registro, Generación de JWT/Cookie.
Metrics Service	Pandas	Lógica de Análisis (agregación, cálculos de series de tiempo).
Router	FastAPI	Gestión de rutas HTTP y Serialización (DTOs).

3. Endpoints a Implementar

Se deben implementar dos grupos de *endpoints*: Autenticación (públicos) y Análisis (privados,

requieren JWT).

Grupo A: Autenticación (/auth)

Estas rutas deben ser **públicas** (no requieren *cookie*).

Método	Ruta	Propósito	DTOs Requeridos
POST	/auth/register	Crea un nuevo usuario.	RegisterUserDTO (Entrada), UserDTO (Salida)
POST	/auth/login	Autentica al usuario. **Establece la Cookie JWT.**	LoginUserDTO (Entrada), UserDTO (Salida)
POST	/auth/logout	Cierra la sesión. **Elimina la Cookie JWT.**	Ninguno (solo Response para eliminar *cookie*)

Grupo B: Análisis de Métricas (/analysis)

Estas rutas deben ser **privadas** y estar protegidas con la dependencia Depends(get_current_user).

Método	Ruta	Propósito	Lógica de Pandas
GET	/analysis/kpi_summary	Retorna KPIs clave del negocio (Ingreso Total, AOV).	Filtrar y Sumar (df.sum(), df.mean()).
GET	/analysis/series	Retorna series de tiempo de ingresos por mes/semana.	Conversión a fecha (pd.to_datetime), Agrupación por tiempo (df.groupby()), Resampling.
GET	/analysis/top_countries	Retorna los ingresos totales por país (Top N).	Agrupación y Ordenamiento (df.groupby().sum().sort_values()).

GET	/analysis/page	Retorna transacciones en formato paginado.	Filtrado, df.iloc[offset:limit].

4. Workflow y Estándares de Implementación

4.1. Estándares de Código

- **Paginación:** Usar la dependencia **PaginationParams** para todos los *endpoints* de listado/paginación.
- **Seguridad:** Implementar la detección de rutas públicas y privadas usando la Inyección de Dependencias selectiva (Depends(get_current_user)).
- **Typing:** Uso estricto de anotaciones de tipo (-> list[UserDTO], :UserService).

4.2. Flujo de Datos y Pandas

Capa	Input	Acción de Datos	Output
Data Repository	Nada (conexión a Sheets)	1. Conexión gspread. 2. Lectura de CSV/Hoja.	pd.DataFrame (Datos Brutos)
Metrics Service	pd.DataFrame (Bruto)	1. Cálculo de Revenue (Quantity * UnitPrice). 2. Agrupaciones y Series de Tiempo.	pd.DataFrame (Datos Analizados/Resumidos)
Router	pd.DataFrame (Analizado)	Conversión a lista de dicts: **df.to_dict(orient='records')**.	JSON (Lista de objetos)

4.3. Estrategia para Power BI

Para asegurar que los datos sean consumibles por Power BI:

- **Formato de Salida:** Los *endpoints* analíticos (/analysis/...) deben devolver una **lista simple de objetos JSON** ([{"date": "2025-01", "revenue": 1000}, ...]). Power BI interpreta esto como una tabla lista para graficar.
- **Tipificación:** No retornar DataFrames anidados o estructuras complejas. FastAPI debe recibir el DataFrame del *Service*, convertirlo a JSON simple y validar que cumpla con un Schema de salida (aunque sea solo para la estructura).