En los tres archivos que hemos creado, cada uno tiene un propósito específico. Aquí está la explicación de

1. Backend

Archivo que actúa como backend: main.py y datos.py (juntos forman el backend).

Qué es el backend:

- El backend es la lógica detrás del sistema.
- Se encarga de procesar datos, realizar cálculos, y manejar la interacción entre la base de datos (en est
- No es visible para el usuario final; trabaja "detrás de escena".

En nuestro caso:

- datos.py maneja los datos:
  - Define cómo se inicializan (con un DataFrame de ejemplo).
  - Define cómo se convierten a JSON para enviarlos a otros componentes.
- main.py expone la lógica del backend a través de una API:
  - Responde a solicitudes HTTP (por ejemplo, GET y POST).
  - Coordina la lógica de negocio usando las funciones de datos.py.

2. API

Archivo que actúa como API: main.py.

Qué es una API:

- Una API (Application Programming Interface) es un puente entre dos sistemas.
- Permite que otros programas (como el frontend) interactúen con el backend.
- Las APIs utilizan rutas y métodos HTTP (GET, POST, etc.) para recibir solicitudes y devolver respuesta

En nuestro caso:

- main.py usa FastAPI para exponer funciones del backend como rutas HTTP:

  - GET /datos: Devuelve los datos en JSON.

  - POST /datos: Permite agregar nuevos datos.

- main.py también gestiona la lógica de la API, como la validación de datos usando Pydantic.

Analogía nueva: Piensa en la API como un camarero en un restaurante:

- Tú (frontend) pides algo del menú (ruta GET o POST).

- El camarero (API) recibe tu pedido y lo lleva a la cocina (backend).

- La cocina (backend) prepara tu pedido (datos o lógica) y el camarero (API) te lo trae.

3. Frontend

Archivo que actúa como frontend: frontend.html.

Qué es el frontend:

- El frontend es la interfaz que el usuario final ve e interactúa.

- Es la parte "visible" de la aplicación: botones, formularios, tablas, etc.

- Usa tecnologías como HTML, CSS y JavaScript para mostrar y manejar la interacción.

En nuestro caso:

- frontend.html muestra:

  - Una tabla para visualizar los datos.

  - Un formulario para enviar nuevos datos.

- Usa JavaScript (con la función fetch) para comunicarse con la API:

  - Envía solicitudes (POST) a http://127.0.0.1:8000/datos.

  - Recibe datos (GET) desde la misma URL para mostrarlos en la tabla.

Analogía nueva: El frontend es como el menú y la mesa del restaurante:

- El menú te muestra las opciones (datos disponibles).

- La mesa te da un espacio donde interactúas (enviar o recibir información).

- Cuando decides algo, se lo pides al camarero (API).

Resumen de roles:

Archivo	Rol	Propósito
datos.py	Backend	Maneja los datos (almacén temporal, funciones para manipulación).
main.py	API	Actúa como puente entre el backend (datos) y el frontend. Expone rutas HTTP.
frontend.html	Frontend	Proporciona una interfaz para que los usuarios interactúen con la API y vean

Conexión entre ellos (flujo completo):

1. El usuario abre frontend.html en su navegador.
2. El usuario hace clic en "Cargar datos" (GET):
   - JavaScript envía una solicitud GET a la API (http://127.0.0.1:8000/datos).
   - La API llama al backend (datos.py) para obtener el DataFrame y lo devuelve en JSON.
   - El frontend actualiza la tabla con los datos.
3. El usuario envía un formulario (POST):
   - JavaScript envía una solicitud POST con los datos ingresados.
   - La API llama al backend para agregar esos datos al DataFrame.
   - La tabla se puede actualizar llamando a la API nuevamente con GET.

¿Te gustaría profundizar en algún aspecto o tienes dudas sobre cómo se conectan estos componentes?