Objetivo:

Crear una API usando FastAPI que permita manejar archivos: listar, crear y ver el contenido de los archivos indicados.

Se recomienda revisar la guía oficial para antes de comenzar
https://fastapi.tiangolo.com/tutorial/

Paso 1: Instalación de dependencias

Primero, chequear si Python y pip están instalados en el entorno local. Si no, instalarlos y luego instalar FastAPI y Uvicorn para poder ejecutar la API.

```
pip install fastapi[standard] uvicorn
```

Alternativamente se puede usar un entorno virtual siguiendo estos pasos:
https://fastapi.tiangolo.com/virtual-environments/

Recomendamos crear un archivo de requerimientos llamado requirements.txt que incluya ambas bibliotecas (y donde se agregará todo lo que requieran importar para la ejecución de sus prácticas).

Paso 2: Crear la API

1. Crear un archivo para la implementación de la API llamado `main.py`
2. Este archivo tiene que servir para exponer lo siguientes endpoints:
    a. **GET `/files`**: Devuelve una lista de los archivos disponibles en el directorio `./files`.

    b. **POST `/files`**: Crea un archivo con el nombre y contenido proporcionado en el cuerpo de la solicitud.

    Esta parte de la documentación puede resultar util:
    https://fastapi.tiangolo.com/tutorial/request-files/

    c. **GET `/files/{file_name}`**: Devuelve el contenido de un archivo especificado en la URL.

Por ejemplo, para crear un endpoint en la raíz (“/”), deberían incluir en su archivo `main.py` el

siguiente código:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def prueba_root():
    return {"message": "Accediste al endpoint de prueba"}
```

Paso 3: Probar localmente

Para probar la API en tu máquina local:

1. Ejecutar el servidor: `fastapi dev main.py`

2. Acceder a la documentación interactiva de la API a través de http://localhost:8000/docs o http://localhost:8000/redocs

3. Hay distintas opciones para probar los diferentes endpoints creados anteriormente:
   - Los endpoints **GET** `/files` y **GET** `/files/{file_name}` pueden probarse directamente escribiendo directamente en la barra de navegación (Ej: http://localhost:8000/files)
   - Utilizar la interfaz provista por la documentación http://localhost:8000/docs
   - Desde Python se puede usar el módulo `requests` apuntado al endpoint.
   - Desde linea de comandos también se puede utilizar `curl`:

     Listar archivos: `curl -X 'GET' 'http://127.0.0.1:8000/files'`
     Ver archivo: `curl -X 'GET' 'http://127.0.0.1:8000/files/test.txt'`

     Crear archivo: `curl -X 'POST' 'http://127.0.0.1:8000/files' -H 'Content-Type: application/json' -d '{"name": "test.txt", "content": "Este es un archivo de prueba"}'`
     ACLARACION: El upload del archivo se puede implementar de distintas formas por lo cual este comando de curl puede que no aplique en todos los casos.

Paso 4: Vincular su repositorio con Render.com para disponibilizar la API online

1. **Crear una cuenta en Render.com:**
   Registrar (o acceder) una cuenta en render.com.

2. Crear un nuevo Web Service
   https://render.com/docs/web-services

3. Vincularlo con el repositorio de GitHub y sumar los parámetros:

   Build Command: `pip install -r requirements.txt`

   Start Command `uvicorn main:app --host 0.0.0.0 --port 10000`

   En la pantalla de configuración se pueden definir más parámetros, que podrán explorar para hacer otras integraciones, pero para el fin de esta actividad es suficiente con definir los parámetros indicados.

4. Al aceptar, en la vista de su Web Service tendrán la siguiente información:
   - Service ID
   - El repositorio
   - La url creada para poder acceder a su API, algo del estilo: https://mi_repo.onrender.com
   - Logs

Paso 5: Probar la API deployada

1. **Acceder a su API** en Render.com, por ejemplo, `https://mi_repo.onrender.com/files`
2. **Probar las rutas de la API** como lo hicieron localmente utilizando `curl` y/o un navegador web.
3. Si quieren integrar nuevos cambios, luego de pushearlos deben actualizar el deploy en el botón “Manual deploy” de la vista de Render.