Documentación de Prueba Técnica 3

Prueba

Construir un API que coloque a disposición la información contenida en el archivo CSV [enlace]. La información del archivo .csv deberá estar disponible a través de los siguientes endpoints:

a) GET /ejercicio/descarga/
b) POST /ejercicio/descarga/

Existirá un campo obligatorio llamado formato en el payload, mediante el cual, un usuario indicara el formato respuesta deseado, ejemplo:

```
POST /ejercicio/descarga/

{

"formato": "csv"

}

GET /ejercicio/descarga/?formato=json
```

Dicho campo podrá aceptar alguno de los siguientes valores:

- csv: Responderá el archivo en formato csv
- json: Responderá la información del en formato json.

Solución

Las tecnologías para utilizar serán FastAPI y Flask con el lenguaje Python

Código FastAPI

Importar las bibliotecas necesarias y crear una instancia de la aplicación FastAPI:

```
from fastapi import FastAPI, HTTPException

import pandas as pd


app = FastAPI()
```

Definir una ruta GET para descargar el archivo en formato CSV:

```
@app.get("/ejercicio/descarga/")
def descargar_archivo_get(formato: str = "csv"):
```

La ruta /ejercicio/descarga/ está asociada a esta función. El parámetro formato especifica el formato deseado, y por defecto se establece como "csv".

Verificar el formato especificado y lanzar una excepción si no es válido:

```
if formato.lower() != "csv" and formato.lower() != "json":
    raise HTTPException(status_code=400, detail="El formato especificado no es válido. Solo se admiten formatos CSV y JSON.")
```

Se comprueba si el formato es diferente de "csv" y "json". Si es así, se lanza una excepción HTTP con un código de estado 400 (solicitud incorrecta) y un mensaje de error.

Leer el archivo CSV (suponiendo que se llama "data.csv"):

```
try:
    df = pd.read_csv("data.csv")
except FileNotFoundError:
    raise HTTPException(status_code=404, detail="El archivo CSV no se encontró.")
```

Se intenta leer el archivo CSV utilizando pd.read_csv(). Si el archivo no se encuentra, se lanza una excepción HTTP con un código de estado 404 (no encontrado) y un mensaje de error.

Descargar el archivo CSV o convertir el DataFrame a JSON según el formato especificado:

```
if formato.lower() == "csv":
    response = df.to_csv(index=False)
    return Response(content=response, media_type="text/csv", headers={"Content-Disposition": "attachment; filename=data.csv"})
```

```
if formato.lower() == "json":
    response = df.to_json(orient="records")
    return Response(content=response, media_type="application/json")
```

Si el formato es "csv", se utiliza df.to_csv() para convertir el DataFrame en una representación CSV y se devuelve como respuesta con el tipo de contenido "text/csv". También se establece el encabezado "Content-Disposition" para indicar el nombre del archivo a descargar. Si el formato es "json", se utiliza df.to_json() para convertir el DataFrame en una representación JSON y se devuelve como respuesta con el tipo de contenido "application/json".

Definir una ruta POST para descargar el archivo en formato CSV o JSON:

```
@app.post("/ejercicio/descarga/")
def descargar_archivo_post(payload: dict):
```

La ruta /ejercicio/descarga/ está asociada a esta función. Esta ruta permite enviar un payload en una solicitud POST para especificar el formato deseado.

Verificar si el campo "formato" está presente en el payload y lanzar una excepción si no está:

```
if "formato" not in payload:
    raise HTTPException(status_code=400, detail="El campo 'formato' es obligatorio en el payload.")
```

Se comprueba si el campo "formato" está presente en el payload enviado. Si no está presente, se lanza una excepción HTTP con un código de estado 400 (solicitud incorrecta) y un mensaje de error.

Leer el archivo CSV:

```
try:
    df = pd.read_csv("data.csv")
except FileNotFoundError:
    raise HTTPException(status_code=404, detail="El archivo CSV no se encontró.")
```

Se intenta leer el archivo CSV utilizando pd.read_csv(). Si el archivo no se encuentra, se lanza una excepción HTTP con un código de estado 404 (no encontrado) y un mensaje de error.

Descargar el archivo CSV o convertir el DataFrame a JSON según el formato especificado, de manera similar a la ruta GET:

```
if formato.lower() == "csv":
    response = df.to_csv(index=False)
    return Response(content=response, media_type="text/csv", headers={"Content-Disposition": "attachment; filename=data.csv"})
if formato.lower() == "json":
    response = df.to_json(orient="records")
    return Response(content=response, media_type="application/json")
```

Si el formato es "csv", se utiliza df.to_csv() para convertir el DataFrame en una representación CSV y se devuelve como respuesta con el tipo de contenido "text/csv". También se establece el encabezado "Content-Disposition" para indicar el nombre del archivo a descargar. Si el formato es "json", se utiliza df.to_json() para convertir el DataFrame en una representación JSON y se devuelve como respuesta con el tipo de contenido "application/json".

Código Flask

Importar los módulos necesarios:

```
from flask import Flask, request, Response
import pandas as pd
```

Se importa la clase Flask de Flask, el objeto request para manejar las solicitudes HTTP y la clase Response para generar las respuestas HTTP. También se importa pandas para trabajar con el archivo CSV.

Crear una instancia de la aplicación Flask:

app = Flask(__name__)

Se crea una instancia de la clase Flask y se le pasa el nombre del módulo actual (name) como argumento.

Definir la ruta de la API para descargar el archivo:

@app.route('/ejercicio/descarga/', methods=['GET', 'POST'])

def descargar_archivo():

Se utiliza el decorador @app.route para asociar la función descargar_archivo() con la ruta '/ejercicio/descarga/'. La ruta admite los métodos GET y POST.

Obtener el formato del archivo a descargar:

formato = request.args.get('formato', 'csv') if request.method == 'GET' else request.json.get('formato', 'csv')

Se utiliza request.args.get() para obtener el parámetro 'formato' de la URL si el método es GET, o se utiliza request.json.get() para obtener el valor de 'formato' del cuerpo JSON si el método es POST. Por defecto, el formato se establece como 'csv' si no se proporciona.

Validar el formato del archivo:

if formato.lower() != 'csv' and formato.lower() != 'json':

return 'El formato especificado no es válido. Solo se admiten formatos CSV y JSON.', 400

Se verifica si el formato especificado es 'csv' o 'json'. Si no es ninguno de ellos, se devuelve un mensaje de error con un código de estado 400 (Solicitud incorrecta).

Leer el archivo CSV:

try:

```
    df = pd.read_csv("data.csv")
except FileNotFoundError:
    return 'El archivo CSV no se encontró.', 404
```

Se intenta leer el archivo CSV llamado "data.csv". Si el archivo no se encuentra, se devuelve un mensaje de error con un código de estado 404 (No encontrado).

Descargar el archivo en el formato especificado:

```
if formato.lower() == 'csv':
    response = Response(content_type='text/csv')
    response.headers['Content-Disposition'] = 'attachment; filename="data.csv"'
    df.to_csv(response, index=False)
    return response
if formato.lower() == 'json':
    response = Response(content_type='application/json')
    response.headers['Content-Disposition'] = 'attachment; filename="data.json"'
    df.to_json(response, orient='records')
    return response
```

Si el formato es 'csv', se crea una instancia de Response con el tipo de contenido 'text/csv' y se establece el encabezado 'Content-Disposition' para indicar el nombre del archivo a descargar. Luego, se utiliza el método to_csv() de DataFrame para escribir los datos en la respuesta. Si el formato es 'json', se realiza un proceso similar pero utilizando 'application/json' como tipo de contenido y se utiliza to_json() para convertir el DataFrame en JSON.

Iniciar la aplicación Flask:

```
if __name__ == '__main__':
    app.run()
```

Se verifica si el script se ejecuta directamente (es decir, no se importa como módulo) y luego se llama al método run() del objeto de la aplicación Flask para iniciar el servidor web y comenzar a manejar las solicitudes entrantes.