Creado por:

Isabel Maniega

Ejercicio

Realizar una Api Rest con FastAPI.

Para ello necesitamos crear una carpeta nueva de nombre testAPI, creamos el entorno virtual e instalamos las librerías necesarias.

A continuación creamos un script de nombre main.py donde pondremos:

In [ ]:

```
# Crear una Api rest
from fastapi import FastAPI
from models import User

app = FastAPI()

# Crear un listado:
database = [{"id": 1, "name": "Juan Perez", "age": 25, "profesion": "Ingeniero"},
            {"id": 2, "name": "Susana Ruiz", "age": 45, "profesion": "Profesora"}]

# TODO: Mostrar el listado: GET

# TODO: Mostrar un dato en concreto: GET

# TODO:Insertar un dato en es listado: POST

# TODO:Actualizaréis un dato del listado: PUT

# TODO:Eliminareis un dato: DELETE

# TODO: Eliminar todos los datos: DELETE
```

Crearemos otro script de nombre models.py, donde crearemos el modelo de usuarios:

In [ ]:

```
from pydantic import BaseModel

# TODO: insertar el modelo de datos
class User(BaseModel):
    pass
```

Realizamos los distintos métodos GET, POST, PUT y DELETE.

Será necesario realizarlo en Visual Studio Code

Solución:

Crear una carpeta donde crearemos la proyecto y creamos el entorno virtual:

- virtualenv < nombre del entorno >

Una vez creado el entorno activaremos el entorno virtual con la siguiente instrucción:

- LINUX / MAC: source env/bin/activate
- WINDOWS: .\env\Scripts\activate

In [1]:

```
# (env) isabel@isabel-SVE1512E1EW:~/FEI_projects/testAPI$

# Aparece el entorno virtual activado entre paréntesis
```

Procedemos a la instalación de las librerias:

- pip install uvicorn[standard] fastapi

Por ultimo ejecutamos la app:

- uvicorn main:app --reload

Importamos las librerías:

In [ ]:

```
from fastapi import FastAPI, status, Response
from models import User
```

Creamos una variable con los datos de las etiquetas:

In [ ]:

```
tags_metadata=[
    {
        "name": "TEST",
        "description": "Bienvenida",
    },
    {
        "name": "Users",
        "description": "Muestra los gestión de los usuarios",
    },
]
```

Declaramos la aplicación y añadimos el titulo de la API-Rest, las etiquetas (variable anterior), la persona de contacto y la version de la API

In [ ]:

```
app = FastAPI(title="DataScience Course",
              openapi_tags=tags_metadata,
              contact={"name": "Isabel Maniega",
                       "url": "https://es.linkedin.com/in/isabel-maniega-cuadrado-4
                       "email": "isabelmaniega@hotmail.com",
                 },
              openapi_url="/api/v0.1/openapi.json")
```

AVISO: para el contacto por email nos pedirá que instalemos:

- pip install email-validator

Listado de partida con la información agregar:

In [ ]:

```
database = [{"id": 1, "name": "Juan Perez", "age": 25, "profesion": "Ingeniero"},
            {"id": 2, "name": "Susana Ruiz", "age": 45, "profesion": "Profesora"}]
```

Declarar el primer método GET para información de la WEB y tenga en cuenta la etiqueta Test:

*Veremos en todos los casos un atributo **description** corresponde a la descripción de la función.*

In [ ]:

```
@app.get("/", tags=["TEST"], description="Mostrar la información de la WEB")
async def info():
    return {"msg": "Bienvenido a nuestra Api Rest"}
```

2) Mostrar la información de todos los datos a la url "/getData/" si es correcto responde con un 200 y tenga en cuenta la etiqueta Users:

In [ ]:

```
@app.get("/getData/", status_code=status.HTTP_200_OK, tags=["Users"],
         description="Mostrar todos los usuarios")
async def show():
    return database
```

3) Mostrar la información de un dato a la url "/getData/{id}" si es correcto responde con un 200 y tenga en cuenta la etiqueta Users:

In [ ]:

```
@app.get("/getData/{item_id}", status_code=status.HTTP_200_OK, tags=["Users"],
         description="Mostrar un usuario")
async def show(id: int, response: Response):
    for i in range(0,len(database)):
        if database[i]["id"] == id:
            response.status_code = status.HTTP_200_OK
            return database[i]
    response.status_code = status.HTTP_404_NOT_FOUND
    return {"id": id, "msg":"User Not Found"}
```

4) Insertar la información de un dato a la url "/postData/" si es correcto responde con un 201 y tenga en cuenta la etiqueta Users:

In [ ]:

```
@app.post("/postData/", status_code=status.HTTP_201_CREATED, tags=["Users"],
          description="Insertar un usuario")
async def insert(item: User):
    # apendizar el diccionario en la lista
    database.append(item.dict())
    return item
```

5) Actualizar la información de un dato a la url "/putData/{id}" si es correcto responde con un 200 y tenga en cuenta la etiqueta Users:

In [ ]:

```
@app.put("/putData/{id}", status_code=status.HTTP_200_OK, tags=["Users"],
         description="Actualizar un usuario")
async def update(id: int, item: User, response: Response):
    for i in range(0,len(database)):
        if database[i]["id"] == id:
            database[i] = item.dict()
            response.status_code = status.HTTP_200_OK
            return item
    response.status_code = status.HTTP_404_NOT_FOUND
    return {"id": id, "msg":"User Not Found"}
```

6) Eliminar la información de un dato a la url "/deleteData/{id}" si es correcto responde con un 204 y tenga en cuenta la etiqueta Users:

In [ ]:

```
@app.delete("/deleteData/{id}", tags=["Users"],
            description="Eliminar un usuario")
async def delete(id: int, response: Response):
    for value in database:
        if value["id"] == id:
            database.remove(value)
            response.status_code = status.HTTP_204_NO_CONTENT
            return {"item_id": id, "msg": "Eliminado"}
    response.status_code = status.HTTP_404_NOT_FOUND
    return {"id": id, "msg":"User Not Found"}
```

7) Eliminar la información todos los datos a la url "/deleteData/" si es correcto responde con un 200 y tenga en cuenta la etiqueta Users:

In [ ]:

```
@app.delete("/deleteData/", tags=["Users"],
            description="Eliminar todos usuario")
async def delete(response: Response):
    database.clear()
    response.status_code = status.HTTP_200_OK
    return {"msg": []}
```

Creado por:

Isabel Maniega