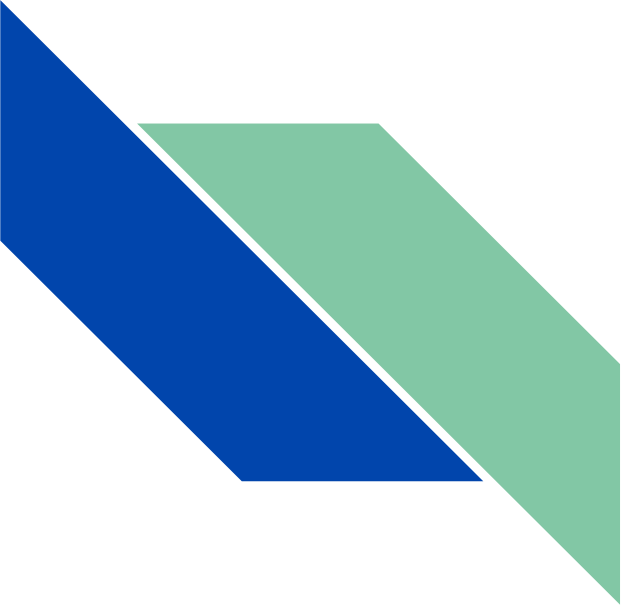


Curso Intensivo de Python - Tema 9

Agustín Arenas

Curso Intensivo de Python - Tema 9

1) FastAPI
2) API
3) HTTP
4) GET
5) POST
6) PUT
7) DELETE
8) Trabajo final

FastAPI

- Para empezar a crear una API hay que instalar FastAPI y Uvicorn, para hacerlo pegamos las siguientes líneas en la consola
  - **pip install fastapi**
  - **pip install "uvicorn[standard]"**

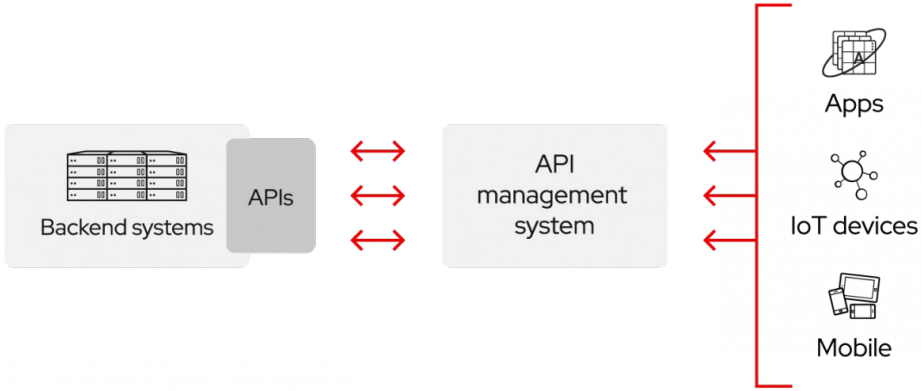
FastAPI

- Para ejecutar el proyecto se debe ejecutar **uvicorn** de la siguiente manera en tu terminal:
  - **uvicorn main:app --reload**

- Para hacer test podemos entrar al link http://127.0.0.1:8000/docs o http://localhost:8000/docs

- Para más información pueden consultar la documentación oficial de FastAPI https://fastapi.tiangolo.com/

API



- Una API es un intermediario entre dos sistemas, que permite que una aplicación se comunique con otra y pida datos o acciones específicas

- Las API son un medio simplificado para conectar su propia infraestructura a través del desarrollo de aplicaciones nativas de la nube, pero también le permiten compartir sus datos con clientes y otros usuarios externos

HTTP

- Define un conjunto de **métodos de petición** para indicar la acción que se desea realizar para un recurso determinado

- Permiten comunicar al servidor lo que se quiere realizar con un resource bajo una URL

- Los métodos más importantes de **HTTP** son **POST, GET, PUT y DELETE**

GET

- El método **GET** solicita una representación de un recurso específico
- Las peticiones que usan el método GET sólo deben recuperar datos

GET

```python
from fastapi import FastAPI

app = FastAPI()

lista_de_productos = ['Producto 1', 'Producto 2', 'Producto 3']

@app.get("/")
async def root():
    '''
    Método que se ejecuta en la raiz del sitio
    @param Ninguno
    @return mensaje de bienvenida
    '''
    return {"message": "Hello World"}

@app.get("/items/{item_id}")
async def read_item(item_id: int):
    '''
    Método que se ejecuta en /items/{item_id}
    @param item_id: int
    @return dict con el producto solicitado
    '''
    return {"item_id": item_id, "item": lista_de_productos[item_id]}
```

POST

- Aunque se puedan enviar datos a través del método GET, es recomendable utilizar **POST**
- El método **POST** se utiliza para enviar una entidad a un recurso en específico
- Causa un cambio en el estado o efectos secundarios en el servidor

POST

```python
from fastapi import FastAPI

app = FastAPI()

lista_de_productos = [{'nombre': 'Producto 1', 'precio': 100},
                      {'nombre': 'Producto 2', 'precio': 200},
                      {'nombre': 'Producto 3', 'precio': 300}]

@app.post("/items")
async def create_item(item: Item):
    '''
    Método que se ejecuta en /items
    @param item: Item
    @return lista con todos los elemetos
    '''
    lista_de_productos.append(item)
    return {"list": lista_de_productos}
```

PUT

- El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición

- Utilizado normalmente para actualizar contenidos, pero también pueden crearlos

PUT

```python
from fastapi import FastAPI

app = FastAPI()

lista_de_productos = [{'nombre': 'Producto 1', 'precio': 100},
                      {'nombre': 'Producto 2', 'precio': 200},
                      {'nombre': 'Producto 3', 'precio': 300}]

@app.put("/items/{item_id}")
async def update_item(item_id: int, item: Item):
    '''
    Método que se ejecuta en /items/{item_id}
    @param item_id: int
    @param item: Item
    @return item actualizado
    '''
    lista_de_productos[item_id] = item
    return {"item": lista_de_productos[item_id]}
```

DELETE

- El método DELETE borra un recurso en específico
- Simplemente elimina un resource identificado en la URI

DELETE

```python
from fastapi import FastAPI

app = FastAPI()

lista_de_productos = [{'nombre': 'Producto 1', 'precio': 100},
                      {'nombre': 'Producto 2', 'precio': 200},
                      {'nombre': 'Producto 3', 'precio': 300}]

@app.delete("/items/{item_id}")
async def update_item(item_id: int):
    '''
    Método que se ejecuta en /items/{item_id}
    @param item_id: int
    @return el producto eliminado
    '''
    return {"item": lista_de_productos.pop(item_id)}
```

Trabajo final

- Walmart tiene muchos productos y el área de ventas necesita hacer una API de control de los productos que están a la venta. Con esta API se debe poder ver todos los productos, crear nuevos individualmente y crear varios productos a la vez, modificar productos existentes, eliminar productos, poder buscar por Identificador y por nombre (busquedas por separado). También añadir una ruta para que se guarden los cambios realizados.

- Cada producto debe tener un Id, Nombre, Precio por unidad, Precio por mayor (a partir de 3 unidades), Stock del producto, Descripción, Calificación (de 1 a 5 estrellas) y Pasillo donde encontrar el producto. Se tiene que guardar toda la información en un archivo para poder modificarla.

- Cuando se cargue un producto nuevo la API debe devolver la nueva lista de productos, al igual que si se elimina o modifica algún producto, cuando se busca solo se tiene que devolver el producto que coincide con la búsqueda.

¿Preguntas?