FUNDAMENTOS DE FASTAPI

Requisitos necesarios para trabajar:

1. Visual Studio Code
2. Python 3.10+
3. Pip: Instalador de librerías de Python

Python es un lenguaje que tiene su potencia en que se comunica con todo.

Su gran incremento de demanda se debe a Hacking y Big Data.

Tenemos múltiples ramas dentro de Python:

* Hacking
* Big Data: Gráficos y análisis de datos
* Jupyter: Big data
* Programas sencillos
* Django: Aplicaciones Web
* FastApi: Creación de Apis con lenguaje Python

A todos estos elementos se les llama Frameworks. Un Framework es una arquitectura basada en un lenguaje y que genera una aplicación.

Un Api es un servicio que permite representar datos y también nos permite interactuar con dichos datos, ya sea para modificar, insertar o recuperar datos.

Un servicio Api es tan popular porque representa datos en su mínima expresión, es decir, devuelve texto.

El texto que devuelve es un estándar, es decir, un texto representativo con un estándar que puede ser tanto JSON como XML.

JSON es una representación textual de un objeto (más popular)

XML es una representación textual con etiquetas de un objeto. (España)

Un servicio Api está formado por diversas características:

* Controller: Son los routers que permiten acceder a unos determinados Endpoints.
* EndPoints: Son los accesos URLS para las peticiones a cada elemento de un Controller.
* Models: Son las clases que puedo recibir/enviar entre las peticiones a cada Controller.
* Responses: Las respuestas pueden ser simples o pueden ser algo más complejas, depende de nuestro servicio.
* Services: Son los elementos de dónde recuperamos los datos a utilizar en el Api y sus controladores.

Además de estas características un servicio puede tener una serie de métodos dentro de los controladores:

* GET: Este método se utiliza cuando recuperamos datos de cualquier Api
* PUT: Este método se utiliza cuando queremos modificar algún elemento dentro del Api.
* DELETE: Este método se utiliza cuando queremos eliminar algo del Api.
* POST: Este método tiene dos vertientes:
  + Insertar información en el Api.
  + Enviar objetos y hacer con ellos lo que deseemos, por ejemplo, en la seguridad de los Apis se utiliza información POST.

<https://apicruddepartamentoscore.azurewebsites.net/index.html>

Departamentos es nuestro **Controller**

GET: Api/Departamentos es un **EndPoint**

Lo que estamos visualizando es un Api con documentación **Swagger**

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Configuración de nuestro entorno de trabajo.

En algún momento crearemos un FastApi en la carpeta en la que tengo el documento de Word.

Documentación y proyecto de FastApi

<https://github.com/serraguti/FASTAPI>

Para consumir un Api no necesitamos casi nada, simplemente código de Python, librería **request** y un entorno de trabajo de Python

Un Environment es un entorno de trabajo aislado dentro de Python y que se utiliza para agrupar todas las librerías y características de un Framework.

En muchas ocasiones, crear un entorno de trabajo es opcional, pero en algunos Frameworks como, FastApi, Django o Jupyter es imprescindible.

En el Environment es dónde podremos agregar las librerías.

Creamos una nueva carpeta para nuestras clases Python llamada **consumoapispython**

Dentro de VS Code instalamos la extensión de Python de Microsoft

Texto

El contenido generado por IA puede ser incorrecto.

Abrimos la carpeta con VS Code

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Tendremos un entorno creado y podremos ejecutar tanto Frameworks como simples programas de Python

Seleccionamos una ventana de terminal de tipo command prompt

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

También podemos crearlo por comandos el entorno virtual desde el terminal

Crear un entorno nuevo

python -m venv venv

Activar el entorno

venv\Scripts\activate.bat

Verificar que tenemos el entorno activado

echo $VIRTUAL\_ENV

El primer paso para utilizar Apis es consumir dichos Apis.

Lo bueno de un Api es que se puede consumir con cualquier lenguaje:

* JavaScript, JQuery, React, Angular, Java, C#, Python

Lo primero que vamos a realizar es crear una carpeta y consumir el Api de departamentos con un GET.

Necesitamos la librería **requests** para consumir cualquier api desde Python

Mediante **pip** podemos instalar cualquier librería en nuestro entorno.

Un entorno está aislado, las librerías no se comparten con otros entornos.

Escribimos en la línea de comandos la siguiente instrucción:

**pip install requests**

Texto

El contenido generado por IA puede ser incorrecto.

El api que vamos a leer será leer un Empleado (por ejemplo)

Necesitamos una librería que ya está integrada dentro del entorno de Python llamada **json**, ya que los datos que vienen del servicio vienen en dicho formato.

Creamos un nuevo programa de Python llamado **python01getempleado.py**

Imagen que contiene Aplicación

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Necesitamos el Request de acceso.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

**PYTHON01GETEMPLEADO.PY**

**import requests**

**print("Leyendo un empleado")**

**apiUrl = "https://apiejemplos.azurewebsites.net/api/Empleados/7839"**

**response = requests.get(apiUrl)**

**#ESTO NOS DEVUELVE UNA RESPUESTA**

**#QUEREMOS EL JSON**

**jsonObject = response.json()**

**#ESTO ES UN OBJETO QUE PODEMOS ACCEDER A SUS PROPIEDADES**

**print("Apellido: " + jsonObject["apellido"])**

**print(jsonObject)**

**print("Fin de programa")**

El siguiente ejemplo será leer un conjunto de datos.

Vamos a leer todos los departamentos, creamos un nuevo programa llamado **python02alldepartamentos.py**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

**PYTHON02ALLDEPARTAMENTOS.PY**

**import requests**

**print("Leyendo muchos departamentos")**

**apiUrl = "https://apiejemplos.azurewebsites.net/api/Departamentos"**

**response = requests.get(apiUrl)**

**departamentos = response.json()**

**#RECORREMOS TODOS LOS DEPARTAMENTOS**

**for row in departamentos:**

**numero = row["idDepartamento"]**

**nombre = row["nombre"]**

**localidad = row["localidad"]**

**print(*f*"Numero: {numero}, Nombre: {nombre}, Localidad {localidad}")**

**print("Fin de programa")**

El siguiente paso es centrarnos en FASTAPI

Requisitos necesarios:

1. Environment
2. Instalar con pip FastApi
3. Instalar uvicorn o cualquier server para ejecutar el Api
4. Podemos instalar más librerías, pero con estas por ahora nos bastarán

**Nota:** Si instalamos fastapi standard ya instala también una serie de características para utilizar FastApi como, por ejemplo: **uvicorn**

Qué características vamos a encontrar:

1. **Tipado de datos**: FastApi utiliza en el lenguaje Python el tipado de datos, elemento que no es necesario para nada en Python Standard



1. Librerías opcionales de parámetros: En los métodos de los Api, como hemos visto, tenemos la posibilidad de enviar información y recibirla. Para poder enviar información, además de indicar el tipo de información, debemos indicar si dicha información será obligatoria o no para nuestro método.

FastApi ya viene con la documentación generada basada en OpenApi:

1. **Swagger:** Documentación estándar que utilizan multitud de servicios en la nube: Java, Net Core (8.0), FastApi
2. **Redoc:** Otro tipo de documentación con otro diseño.

Necesitamos una clase principal que, normalmente, se denomina **main.py**.

Dentro de dicha clase es dónde estarán nuestros métodos EndPoints.

Por defecto, solamente tendremos un Controller llamado **Default**, aunque por supuesto, crearemos nuestros propios controllers.

Existirá una variable llamada como lo deseemos (normalmente **app**) que será de tipo **FastApi** y la que contendrá los métodos GET, POST, PUT o DELETE.

Dicha variable generará también los ENDPOINTS para cada método.

Comenzamos creando una nueva carpeta llamada **fastapi**

Abrimos la carpeta raíz con VS Code.