Trabajando con servicios FastAPI

El objetivo de este documento es mostrar paso a paso cómo crear un servicio web básico utilizando **FastAPI**, un moderno framework web para Python.
A lo largo del documento, se mostrará:

- Cómo preparar el entorno de desarrollo en **Windows** y **Linux**.
- Cómo crear un proyecto en **Visual Studio Code**.
- Cómo escribir un servicio web que recibe una palabra y la devuelve al reves como respuesta.
- Cómo ejecutar y probar el servicio

¿Qué es FastAPI?

FastAPI es un framework web moderno, rápido y de alto rendimiento para construir APIs con Python, basado en las anotaciones de tipo estándar de Python.

FastAPI destaca por su velocidad de ejecución, ofreciendo un rendimiento comparable con frameworks desarrollados en **NodeJS** y **Go**. Esta velocidad excepcional lo posiciona como uno de los frameworks de Python más rápidos disponibles en el mercado, gracias a su arquitectura basada en Starlette y la optimización que proporciona Pydantic para el manejo de datos.

Una de las características más destacadas de FastAPI es su capacidad para acelerar significativamente el proceso de desarrollo. Los desarrolladores pueden incrementar su velocidad de desarrollo entre 200% y 300%, lo que se traduce en una entrega más rápida de funcionalidades. Además, el framework ayuda a reducir aproximadamente 40% de los errores inducidos por desarrolladores, minimizando la duplicación de código y permitiendo obtener múltiples características a partir de una sola declaración de parámetros.

FastAPI ha sido diseñado con la experiencia del desarrollador como prioridad. Ofrece un soporte excelente en editores de código con autocompletado inteligente en todas partes, lo que hace que trabajar con el framework sea intuitivo y natural. Su curva de aprendizaje es suave, requiriendo menos tiempo para dominar sus conceptos y menos tiempo dedicado a leer documentación o depurar código.

El framework genera código listo para producción desde el primer momento, incluyendo documentación interactiva automática que facilita el mantenimiento y la colaboración en equipos. FastAPI implementa validación automática de datos y manejo de errores, lo que contribuye a crear aplicaciones más estables y confiables. Está completamente basado en estándares abiertos como OpenAPI (anteriormente conocido como Swagger) y JSON Schema, garantizando compatibilidad y interoperabilidad.

Una característica revolucionaria de FastAPI es su capacidad para generar documentación interactiva automáticamente, sin requerir configuración adicional. Esta documentación incluye una

interfaz web que permite probar los endpoints directamente desde el navegador, facilitando tanto el desarrollo como las pruebas de la API.

fuente - https://fastapi.tiangolo.com/

Preparando el entorno de desarrollo

Antes de comenzar a crear nuestro servicio con FastAPI, es necesario asegurarse de que el entorno de desarrollo esté correctamente configurado.
Lo primero que hay que tener instalado es **Python 3.8 o superior** instalado. También algún IDE como **Visual Studio Code**.

Lo primero que recomendamos hacer es crear un entorno virtual. Un entorno virtual te permite instalar paquetes de Python solo dentro de tu proyecto, evitando conflictos con otros proyectos. Si tenés instalado Anaconda o Conda, podés crear tu entorno virtual con esas herramientas.

En Windows:
abrir la terminal y escribir
`python -m venv entornoFastApi`
Para activar el entorno, ejecuta:
`entornoFastApi\Scripts\activate`
	En Linux
	abrir la terminal y escribir
	`python3 -m venv entornoFastApi`
	Para activar el entorno, ejecuta:
	`source entornoFastApi/bin/activate`

Luego, con el entorno activo, instala las dependencias necesarias, **FastAPI y Uvicorn**:

```
pip install fastapi uvicorn
```

Crear el proyecto

Ahora que tenemos nuestro entorno listo y las dependencias instaladas, es momento de comenzar a construir nuestro servicio web.

Antes de escribir código, vamos a crear una carpeta para organizar nuestro proyecto. Podés llamarla, por ejemplo, `invertir_texto`. Esta carpeta será el lugar donde guardaremos todos los archivos relacionados con nuestro servicio web.

Una vez que tengas creada la carpeta, abrí Visual Studio Code y cargá esa carpeta como espacio de trabajo. Luego, dentro de ella, creá un archivo llamado `main.py`, que será el archivo principal donde escribiremos el código de nuestro servicio.

```
from fastapi import FastAPI

app = FastAPI()
```

```python
@app.get("/invertir_texto/")

def invertir_texto(texto: str):

    texto_invertido = texto[::-1]

    return {"respuesta": texto_invertido}
```

Ejecutar el servicio

Para ejecutar el servicio web que acabamos de crear, primero asegúrate de que tu entorno virtual `entornoFastApi` esté activado. Luego, en la terminal de Visual Studio Code, dentro de la carpeta del proyecto, ejecuta el siguiente comando:
`uvicorn main:app --reload`

Este comando inicia el servidor Uvicorn que servirá tu aplicación FastAPI. La opción `--reload` hace que el servidor se reinicie automáticamente cada vez que guardes cambios en el código, facilitando el desarrollo.

Si todo está correcto, verás un mensaje similar a este:

`Uvicorn running on http://127.0.0.1:8000`

Ahora, tu servicio está activo y escuchando solicitudes en la dirección http://127.0.0.1:8000.

Probando el servicio

Una vez que tu servicio esté en funcionamiento, es importante verificar que realmente responda como esperás. Existen varias formas de hacerlo, desde el navegador hasta herramientas más avanzadas. A continuación, te explico tres maneras recomendadas para probar tu servicio FastAPI.

1. Usando el navegador web

La forma más simple de probar tu servicio es directamente desde el navegador. Solo tenés que escribir en la barra de direcciones la URL del servicio incluyendo el parámetro `texto`. Por ejemplo:

http://127.0.0.1:8000/invertir_texto/?texto=hola

El navegador te mostrará una respuesta en formato JSON como esta:

{

  "respuesta": "aloh"

}

Esta opción es útil para pruebas rápidas con parámetros simples.

2. Utilizando la documentación interactiva de FastAPI

FastAPI genera automáticamente dos interfaces de documentación para tu servicio web, que podés abrir desde el navegador:

- **Swagger UI:** http://127.0.0.1:8000/docs
- **ReDoc:** http://127.0.0.1:8000/redoc

Ambas interfaces muestran una descripción automática de todas las rutas disponibles en tu API, los parámetros que acepta y cómo es la respuesta esperada.

Swagger UI es una interfaz muy interactiva. Te permite **probar el servicio directamente desde la página web**:

1. Ingresá a http://127.0.0.1:8000/docs
2. Buscá el endpoint `/invertir_texto/`
3. Hacé clic en **"Try it out"**
4. Ingresá una palabra en el campo de texto
5. Hacé clic en **"Execute"**
6. Verás la respuesta JSON justo debajo, junto con detalles de la solicitud enviada.

ReDoc es una interfaz más enfocada en la **documentación estructurada**, ideal para leer y entender cómo está definida tu API. Muestra todos los detalles de los endpoints, los parámetros, los tipos de datos, las respuestas posibles, etc.

A diferencia de Swagger, **ReDoc no permite probar los endpoints directamente desde la interfaz**. Es solo para consulta y documentación.

Trabajo para Técnicas de NLP y AI aplicadas a textos semi-estructurados para identificar similitud y extraer información

Modalidad: Grupos de **2 personas**
Desarrollar un **servicio web utilizando fast API** que gestione una funcionalidad a elección del grupo

1- passive_voice

Permite detectar cuando se utiliza voz pasiva

Español

- Herramienta que lo implementa: Spacy, Postagging. Se recomienda utilizar el modelo trf debido a que es el modelo de mayor precisión.
- Escenarios de prueba:
    - El perro ha caído por el barranco
    - Las peras fueron comidas por Pedro
    - Los cuadros fueron pintados por Juan

Inglés

- Herramienta que lo implementa: idem para español. También existe una herramienta llamada PassivePy que detecta cuando una oración está en pasiva, en inglés.
- Escenarios de prueba. The cat was chased by the dog.
    - Cookies were baked for her friends by Sarah.
    - Brightly will shine the sun in the sky.
    - This text had been preprocesed and translated by Julia.
    - A letter was sent to Mary
    - Mary was sent a letter

2- adverbs

Detección de adverbios.
Español e Inglés: En los dos lenguajes se puede utilizar Spacy.
Es muy sencillo, solo hay que chequear que token.pos_ == "ADV".
Ejemplos en Español:

- Rápidamente salió corriendo, pero luego volvió tranquilamente.
- Hoy comimos muy temprano.
- Casi nunca llega tarde.
- Allí estaba esperando con paciencia.
- Realmente aprecio tu ayuda.

Ejemplos en inglés:

- She speaks fluently
- They arrived early to the meeting.
- He runs quickly when he's nervous.
- I usually eat breakfast at 8 a.m.
- The teacher explained the topic clearly.

3- past_verbs

Español

- Herramienta que lo implementa: Spacy, Postagging.
- Método que lo resuelve: Lo anterior funciona bien, pero hay que tener en cuenta que en casos donde participe un infinitivo puede no detectar correctamente el pasado. Se pueden usar de Spacy cosas como el token.morph y preguntar por tense o part.
- Escenarios de prueba: Es un ejemplo trivial.
    - El perro ha caído por el barranco
    - El perro se cayó por el barranco
    - El dueño tiró a su perro por el barranco
    - Juan de niño jugaba al fútbol.

Inglés

- Herramienta que lo implementa: idem para español.
- Método que lo resuelve: idem para español.
- Escenarios de prueba. Son ejemplos triviales de voz pasiva. Completar con otros tiempos verbales en pasado.
    - The cat was chased by the dog.
    - Cookies were baked for her friends by Sarah.
    - Brightly will shine the sun in the sky.
    - She ate sushi yesterday.
    - I was studying when you called.
    - They were playing soccer at 6 PM.
    - The project was being discussed in the meeting.
    - Juan had finished his homework before dinner.
    - By 2020, Melina had written three novels.
    - The letter had been sent before the storm started.
    - The keys had been lost for weeks.
    - She had been working at the company since 2015.
    - I used to drink coffee every morning.
    - When I was young, I would climb trees.
    - The message might have been deleted.
    - She would have traveled if she had saved money.
    - My phone got stolen at the concert. (Este es un caso especial)
    - She was given a gift. / A gift was given to her. (Dos formas de decir lo mismo)

4- long_sentences

Long sentences (default: more than 25 words).

Para esto no se necesita ninguna tecnología de NLP o IA. Simplemente con split, te quedas con todas las palabras en un arreglo sin tomar ningún espacio. Y luego haces un size o len para obtener la cantidad de elementos del arreglo. Para el caso en ingles podría llegar a dar problema esto con este tipo de palabras Object-oriented. Pero se soluciona al sacar los caracteres como -.
Ejemplo en español:

- A pesar de que ayer prometí levantarme temprano para terminar el informe, esta mañana me quedé dormido porque el despertador no sonó, así que ahora tengo que trabajar a toda prisa antes de la reunión de la tarde.

Ejemplo en Inglés:

- Understanding Object-Oriented design principles—such as inheritance, polymorphism, and abstraction—is essential for developers aiming to write flexible, modular code that efficiently handles evolving project requirements across diverse programming languages.