El objetivo principal de este lab es mostrar cómo puede crear rápidamente puntos finales de API REST para sus aplicaciones LLM. Crearemos una aplicación Python simple (solo una función) que utiliza la API OpenAI GPT-4 y traduce texto del inglés al francés. Esta aplicación solo tiene una función que es `translate_text`:

```
from openai import OpenAI

import os

# Initialize OpenAI client with your API key

client = OpenAI()

def translate_text(input_str):

  completion = client.chat.completions.create(

    model="gpt-4-0125-preview",

    messages=[

      {

        "role": "system",

        "content": "You are an expert translator who translates text from english to french and only return translated text",

      },

      {"role": "user", "content": input_str},

    ],

  )

  return completion.choices[0].message.content
```

Esta función toma una entrada del usuario y simplemente la pasa al modelo OpenAI GPT-4. Tiene un mensaje del sistema para definir el contexto y el mensaje del usuario es el texto que obtenemos del usuario en `input_str`.

Puede probar esta función ejecutando lo siguiente. Deberías recuperar el texto en francés.

```
# test the function

translate_text("this  is a test string to translate")
```

Abierto AI

Si está considerando compartir esta aplicación con otras personas, simplemente enviar el archivo Python para que lo ejecuten en su línea de comandos podría funcionar para algunos usuarios, especialmente aquellos familiarizados con Python y las herramientas de línea de comandos.

Sin embargo, este enfoque tiene limitaciones. No todos los usuarios se sienten cómodos usando la línea de comandos, e incluso para aquellos que sí lo hacen, ejecutar un script de Python requiere un entorno Python configurado con todas las dependencias necesarias. Esto puede resultar engorroso y crear una barrera para que los usuarios accedan fácilmente a la funcionalidad de su aplicación de traducción.

Aquí es donde envolver su aplicación en una FastAPI para exponerla como un punto final de API REST se convierte en una solución poderosa. Al hacerlo, básicamente estás creando una interfaz accesible para tu aplicación a la que se puede acceder a través de Internet sin necesidad de que los usuarios ejecuten scripts de Python ellos mismos. Pueden interactuar con su aplicación a través de solicitudes web simples, que se pueden realizar desde una amplia variedad de plataformas y lenguajes de programación, no solo Python.

La creación de una API REST con FastAPI automatiza el proceso de recepción de solicitudes de traducción y envío de traducciones. Los usuarios pueden enviar texto para traducir a través de una interfaz web o una llamada API programática, lo que lo hace mucho más fácil y flexible para diferentes casos de uso.

Para convertir esta función simple en un punto final API usando FastAPI modificaremos nuestro código:

```
from fastapi import FastAPI, HTTPException

from pydantic import BaseModel

import os

# Initialize OpenAI client with your API key

client = OpenAI()

# Initialize FastAPI client

app = FastAPI()
```

```
# Create class with pydantic BaseModel

class TranslationRequest(BaseModel):

    input_str: str

def translate_text(input_str):

    completion = client.chat.completions.create(

        model="gpt-4-0125-preview",

        messages=[

            {

                "role": "system",

                "content": "You are an expert translator who translates text from english to french and only return translated text",

            },

            {"role": "user", "content": input_str},

        ],

    )

    return completion.choices[0].message.content

@app.post("/translate/")    # This line decorates 'translate' as a POST endpoint

async def translate(request: TranslationRequest):

    try:

        # Call your translation function

        translated_text = translate_text(request.input_str)
```

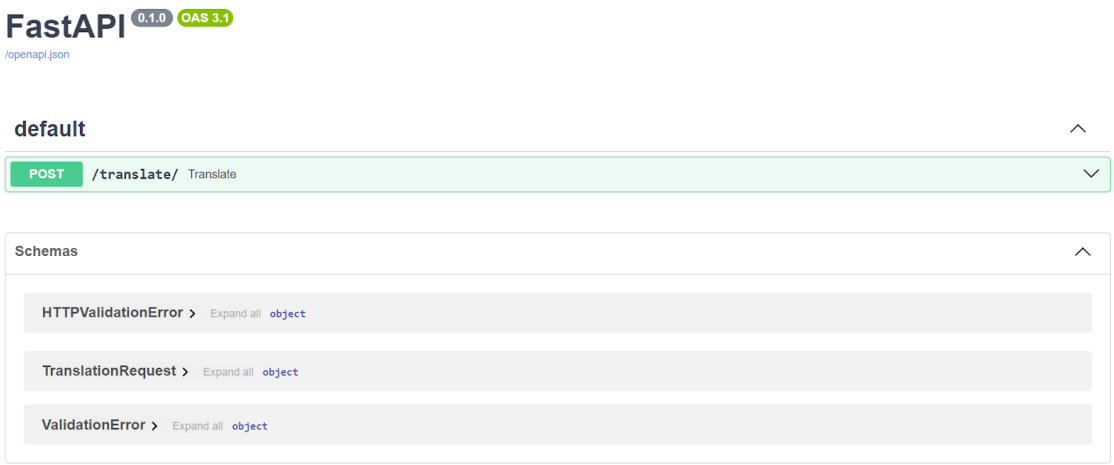
```
    return {"translated_text": translated_text}

except Exception as e:

    # Handle exceptions or errors during translation

    raise HTTPException(status_code=500, detail=str(e))
```

Para ejecutar y probar esta API localmente, simplemente ejecute el siguiente comando:

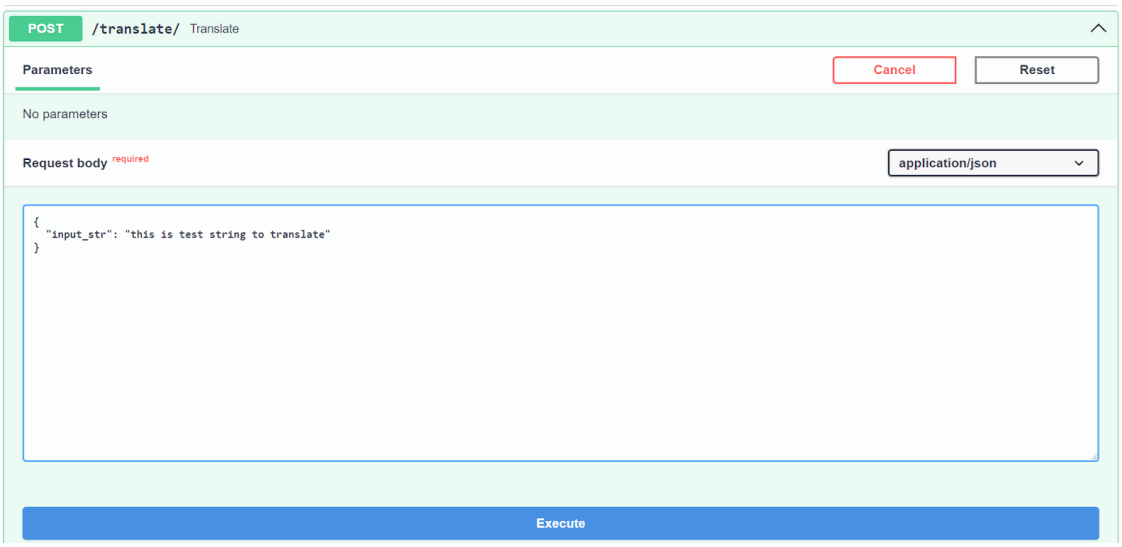
```
uvicorn main:app --reload
```

Aquí, main.py se refiere al archivo que contiene su código. El término aplicación apunta a la instancia de la aplicación FastAPI, que se identifica en su código `app = FastAPI()`.

Navegue hasta localhost:8000/docs para ver una interfaz de usuario que debería verse así:




Haga clic en PUBLICAR, luego haga clic en probarlo, coloque su cadena de texto y haga clic en Ejecutar.




Debería ver un comando curl que se envió a la API, la URL de solicitud que es su host local en este momento y la respuesta del servidor. El código 200 significa éxito; al lado puedes ver la traducción al francés.