Service-oriented architectures

ITMD_566

Project By:

Sneha Patil

Naga Satya Shilpa Annadevara

Our Research Topic: **Travel & Tourist API**

To begin with, what is an API? (Application Programming Interface)

1. An API is a software intermediary that allows two applications to talk to each other.

1. It functions like a waiter in a restaurant, taking a request from an application and sending it to a server.

1. To access an API, you must provide an **API key**, which is an authentic string of letters and numbers.

1. Popular Python frameworks for building APIs include Django, Flask, and Fast API.

1. This tutorial focuses on Fast API, a framework used in this context.

What is Fast API?

- Fast API is a modern web framework for building RESTful APIs in Python. **The key features are:**

a) **Fast to run:** With Pydantic and Starlette's help, extremely good performance that is comparable to NodeJS and Go. The quickest Python framework out there.
b) **Fast to code**
c) **Easy & Straightforward**: Designed to be easy to use and learn. Less time reading docs.
d) **OpenAPI based:** Fully compatible with Open API(Swagger) and JSON Schema.

- FastAPI is a good choice for building web services. It can replace Flask/Django.

Motivation:

Both of us like **taking trips & Travelling**. It is difficult to access and integrate full data since travel-related information is dispersed across several platforms and services.

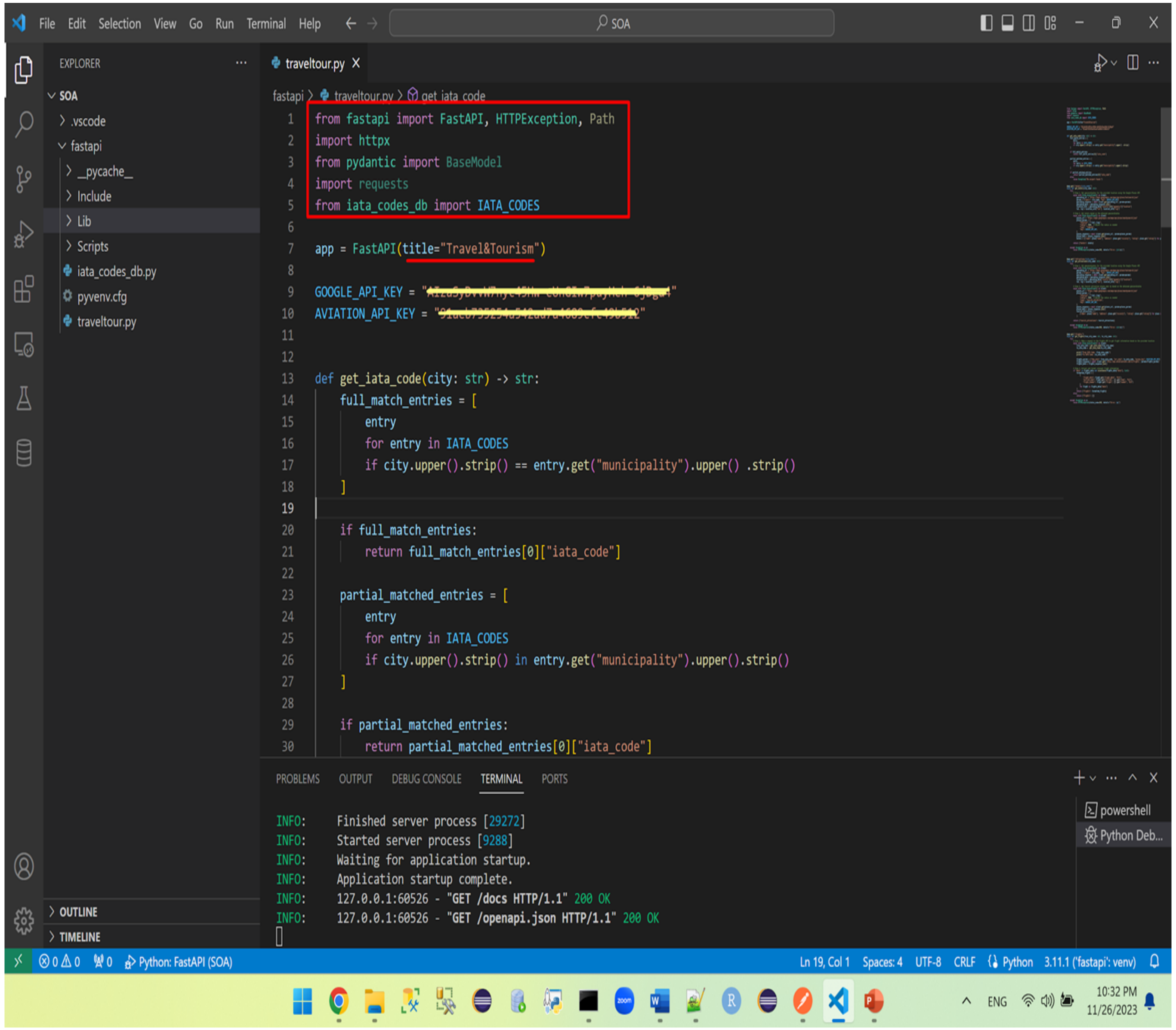
Accessing a variety of information through a single interface is made easier by this API, which acts as a central repository for travel data from hotels, airlines, tourism boards, and other sources.

Introduction of Travel & Tourism API:

- Definition: An API for travel and tourism is a collection of tools and protocols that let developers access and incorporate data on travel destinations, hotels, and services.
- Purpose and Scope: The purpose of the API is to provide a practical way to get a variety of travel-related information, such as airline tickets, lodging, and tourism destinations.

- Technical Aspects
  - ❑ **Programming Language**: We used **Python**.
  - ❑ **IDE:** We used **Visual Studio Code** for its user-friendly interface.
  - ❑ **Web Framework:** We made use of **Fast API's** features
  - ❑ **API Documentation:** We created interactive API documentation **Swagger**

A Closer Look at Our API Implementation

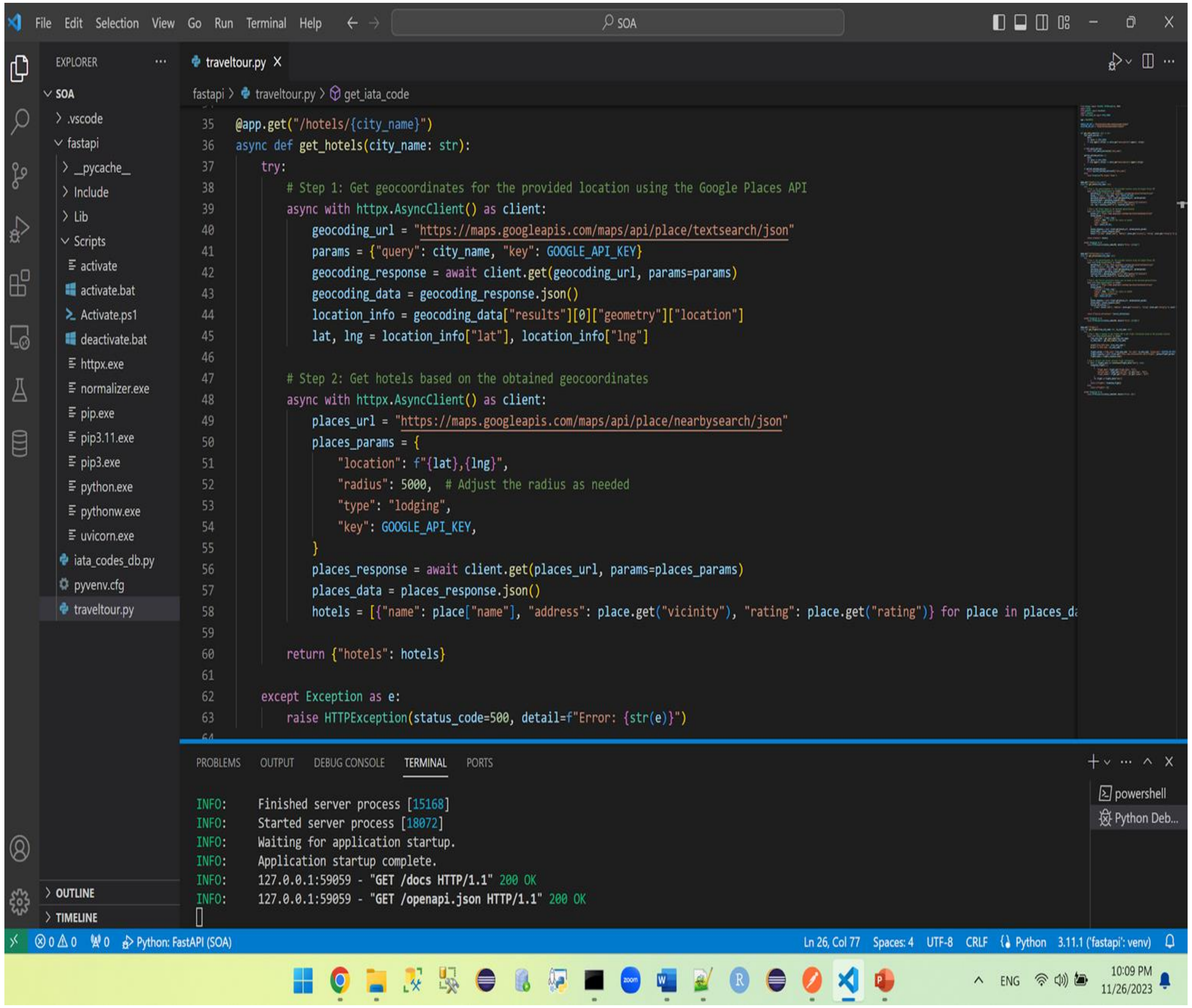

File Edit Selection View Go Run Terminal Help
SOA
EXPLORER
SOA
.vscode
fastapi
pycache
Include
Lib
Scripts
iata_codes_db.py
pyvenv.cfg
traveltour.py
traveltour.py
fastapi > traveltour.py > get_iata_code
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python Deb...
Python: FastAPI (SOA)
Ln 19, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.1 ('fastapi': venv)
ENG
10:32 PM
11/26/2023

```python
from fastapi import FastAPI, HTTPException, Path
import httpx
from pydantic import BaseModel
import requests
from iata_codes_db import IATA_CODES

app = FastAPI(title="Travel&Tourism")

GOOGLE_API_KEY = "[redacted]"
AVIATION_API_KEY = "[redacted]"


def get_iata_code(city: str) -> str:
    full_match_entries = [
        entry
        for entry in IATA_CODES
        if city.upper().strip() == entry.get("municipality").upper() .strip()
    ]

    if full_match_entries:
        return full_match_entries[0]["iata_code"]

    partial_matched_entries = [
        entry
        for entry in IATA_CODES
        if city.upper().strip() in entry.get("municipality").upper().strip()
    ]

    if partial_matched_entries:
        return partial_matched_entries[0]["iata_code"]
```

```
INFO:     Finished server process [29272]
INFO:     Started server process [9288]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:60526 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:60526 - "GET /openapi.json HTTP/1.1" 200 OK
```

- We installed Fast Api and all the necessary Libraries.
- We used 2 existing API's in our project:
  - **a) Google Places API**
  - **b) AviationStack API**
- API keys:
  - **1. GOOGLE_API_KEY**
  - **2. AVIATION_API_KEY**
- The IATA_CODES is a database of airport codes.
- 'get_iata_code' function.
  - Full match & Partial match entries.

A Closer Look at Our API Implementation ~ Hotels API



```python
@app.get("/hotels/{city_name}")
async def get_hotels(city_name: str):
    try:
        # Step 1: Get geocoordinates for the provided location using the Google Places API
        async with httpx.AsyncClient() as client:
            geocoding_url = "https://maps.googleapis.com/maps/api/place/textsearch/json"
            params = {"query": city_name, "key": GOOGLE_API_KEY}
            geocoding_response = await client.get(geocoding_url, params=params)
            geocoding_data = geocoding_response.json()
            location_info = geocoding_data["results"][0]["geometry"]["location"]
            lat, lng = location_info["lat"], location_info["lng"]

        # Step 2: Get hotels based on the obtained geocoordinates
        async with httpx.AsyncClient() as client:
            places_url = "https://maps.googleapis.com/maps/api/place/nearbysearch/json"
            places_params = {
                "location": f"{lat},{lng}",
                "radius": 5000,  # Adjust the radius as needed
                "type": "lodging",
                "key": GOOGLE_API_KEY,
            }
            places_response = await client.get(places_url, params=places_params)
            places_data = places_response.json()
            hotels = [{"name": place["name"], "address": place.get("vicinity"), "rating": place.get("rating")} for place in places_da

        return {"hotels": hotels}

    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Error: {str(e)}")
```

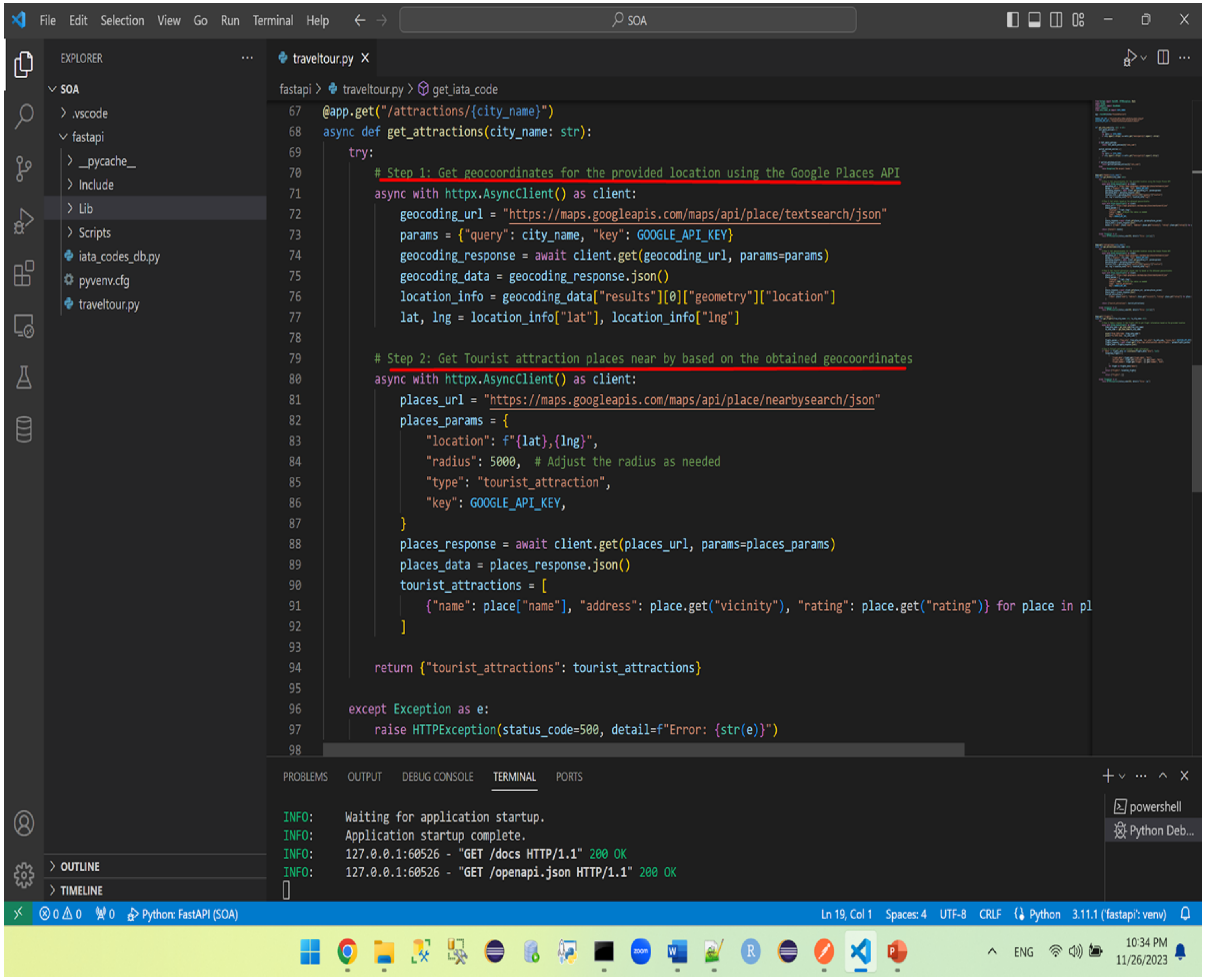
```
INFO:     Finished server process [15168]
INFO:     Started server process [18072]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:59059 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:59059 - "GET /openapi.json HTTP/1.1" 200 OK
```

- The endpoint (/hotels/{city_name})

- The parameters we used here are:
  - ✓ Location ~ lat & long.
  - ✓ Radius ~ 5000
  - ✓ Type ~ lodging

❑ The retrieved hotel data is:
  - ✓ Name.
  - ✓ Address.
  - ✓ Rating.

A Closer Look at Our API Implementation ~ Attractions API



```python
@app.get("/attractions/{city_name}")
async def get_attractions(city_name: str):
    try:
        # Step 1: Get geocoordinates for the provided location using the Google Places API
        async with httpx.AsyncClient() as client:
            geocoding_url = "https://maps.googleapis.com/maps/api/place/textsearch/json"
            params = {"query": city_name, "key": GOOGLE_API_KEY}
            geocoding_response = await client.get(geocoding_url, params=params)
            geocoding_data = geocoding_response.json()
            location_info = geocoding_data["results"][0]["geometry"]["location"]
            lat, lng = location_info["lat"], location_info["lng"]

        # Step 2: Get Tourist attraction places near by based on the obtained geocoordinates
        async with httpx.AsyncClient() as client:
            places_url = "https://maps.googleapis.com/maps/api/place/nearbysearch/json"
            places_params = {
                "location": f"{lat},{lng}",
                "radius": 5000,  # Adjust the radius as needed
                "type": "tourist_attraction",
                "key": GOOGLE_API_KEY,
            }
            places_response = await client.get(places_url, params=places_params)
            places_data = places_response.json()
            tourist_attractions = [
                {"name": place["name"], "address": place.get("vicinity"), "rating": place.get("rating")} for place in pl
            ]

        return {"tourist_attractions": tourist_attractions}

    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Error: {str(e)}")
```

```
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:60526 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:60526 - "GET /openapi.json HTTP/1.1" 200 OK
```

File Edit Selection View Go Run Terminal Help
SOA
EXPLORER
SOA
.vscode
fastapi
pycache
Include
Lib
Scripts
iata_codes_db.py
pyvenv.cfg
traveltour.py
fastapi > traveltour.py > get_iata_code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python Deb...
OUTLINE
TIMELINE
Python: FastAPI (SOA)
Ln 19, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.1 ('fastapi': venv)
ENG
10:34 PM 11/26/2023

- The end point here is (/attractions/{city_name}):
- The process is same.

❑ The parameters we used here are:

✓ Location ~ lat & long.

✓ Radius ~ 5000

✓ Type ~ lodging

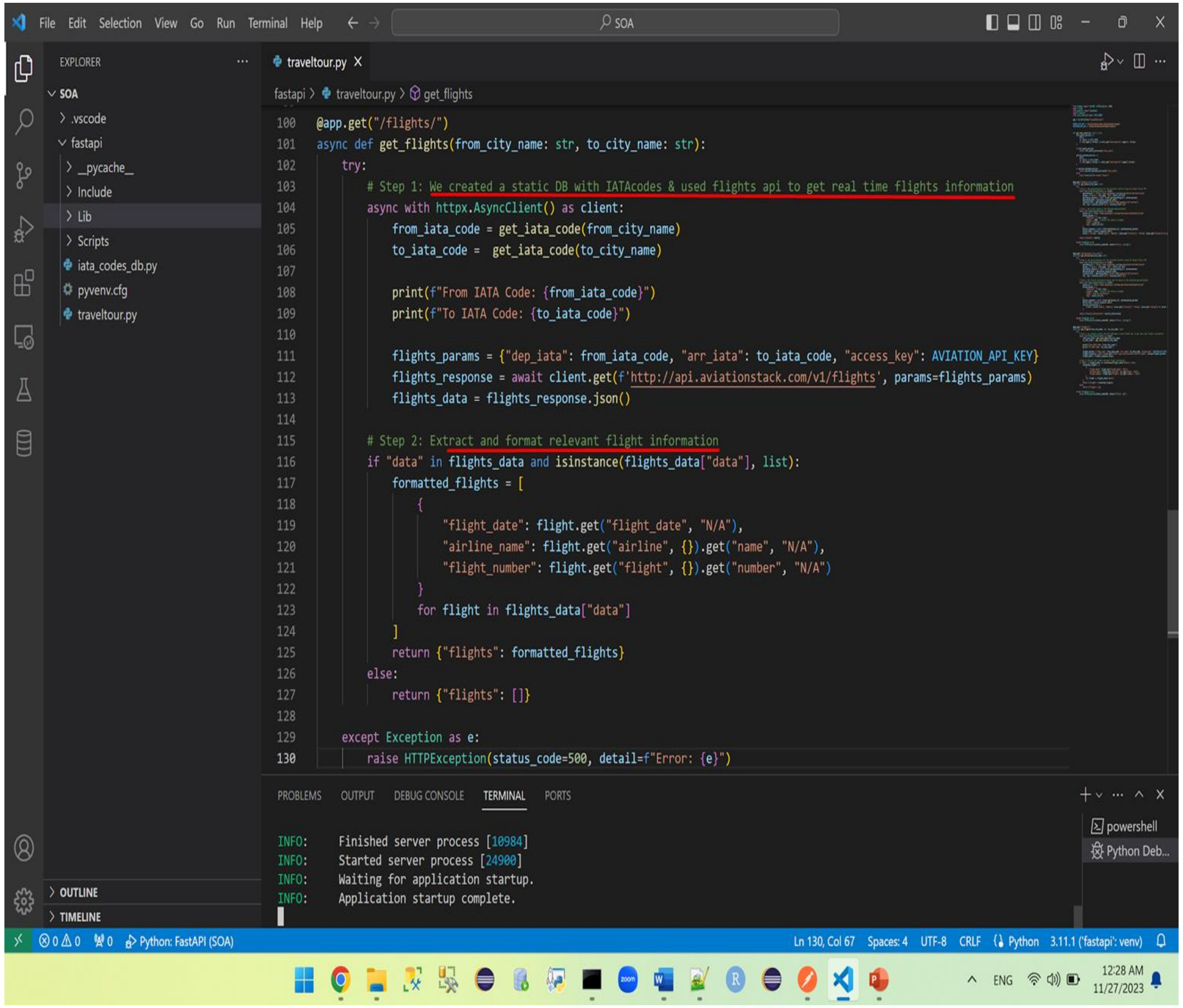
❑ The retrieved Tourist Attraction places data:

✓ Name.

✓ Address.

✓ Rating.

A Closer Look at Our API Implementation ~ Flights API


File Edit Selection View Go Run Terminal Help
SOA
EXPLORER
SOA
.vscode
fastapi
pycache
Include
Lib
Scripts
iata_codes_db.py
pyvenv.cfg
traveltour.py
OUTLINE
TIMELINE
traveltour.py
fastapi > traveltour.py > get_flights
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Python Deb...
Python: FastAPI (SOA)
Ln 130, Col 67 Spaces: 4 UTF-8 CRLF Python 3.11.1 ('fastapi': venv)
12:28 AM 11/27/2023

```python
@app.get("/flights/")
async def get_flights(from_city_name: str, to_city_name: str):
    try:
        # Step 1: We created a static DB with IATAcodes & used flights api to get real time flights information
        async with httpx.AsyncClient() as client:
            from_iata_code = get_iata_code(from_city_name)
            to_iata_code =  get_iata_code(to_city_name)

            print(f"From IATA Code: {from_iata_code}")
            print(f"To IATA Code: {to_iata_code}")

            flights_params = {"dep_iata": from_iata_code, "arr_iata": to_iata_code, "access_key": AVIATION_API_KEY}
            flights_response = await client.get(f'http://api.aviationstack.com/v1/flights', params=flights_params)
            flights_data = flights_response.json()

        # Step 2: Extract and format relevant flight information
        if "data" in flights_data and isinstance(flights_data["data"], list):
            formatted_flights = [
                {
                    "flight_date": flight.get("flight_date", "N/A"),
                    "airline_name": flight.get("airline", {}).get("name", "N/A"),
                    "flight_number": flight.get("flight", {}).get("number", "N/A")
                }
                for flight in flights_data["data"]
            ]
            return {"flights": formatted_flights}
        else:
            return {"flights": []}

    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Error: {e}")
```

```
INFO:     Finished server process [10984]
INFO:     Started server process [24900]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

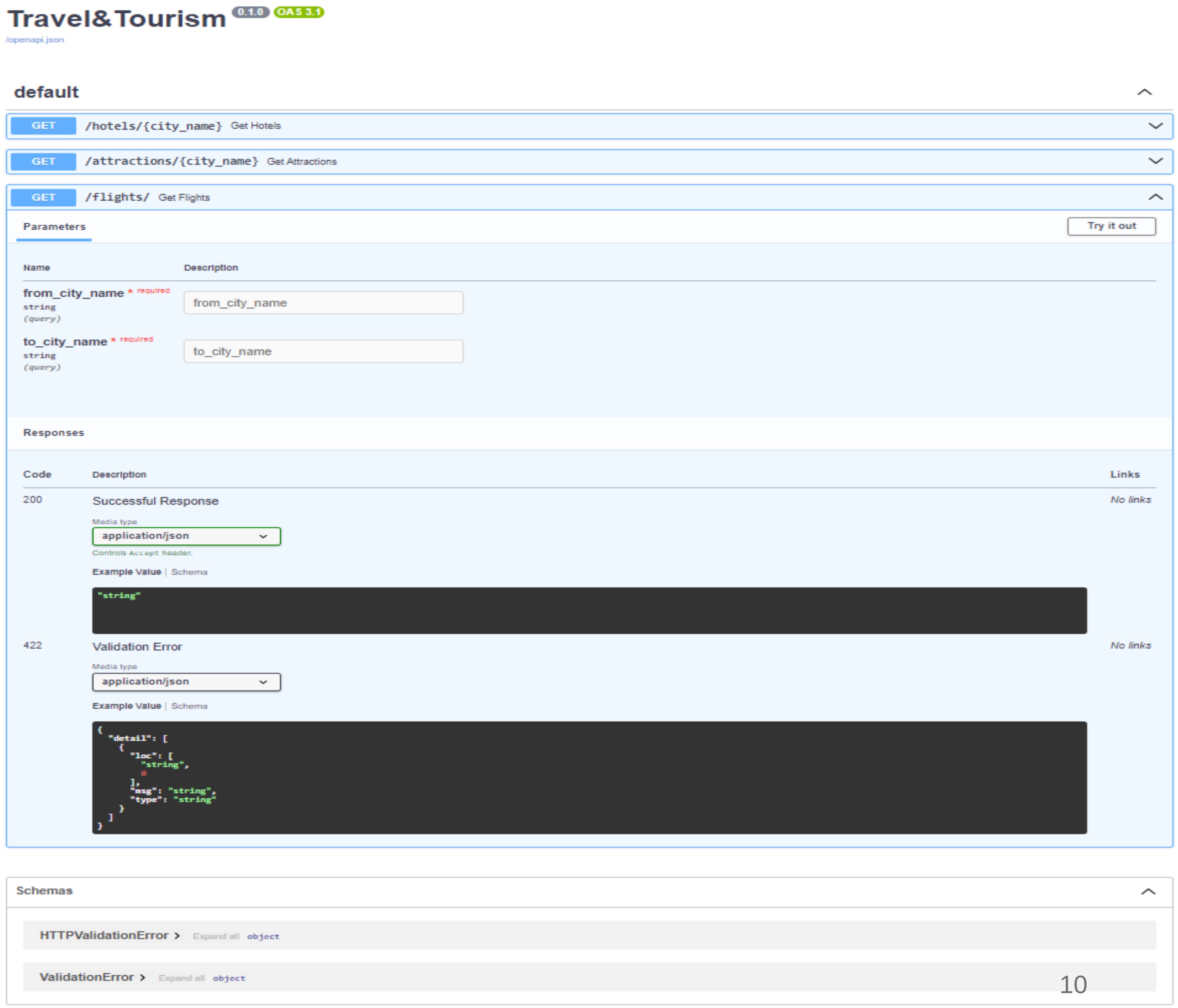
- The endpoint here Is (/flights/)
- It uses the get_iata_code function
- The parameters we used here are:
  - ✓ dep_iata
  - ✓ Arr_iata
- Then, it makes a request to the AviationStack API.
- The retrieved Flight Information:
  - ✓ flight_date
  - ✓ airline_name
  - ✓ flight_number

❖ **Limitations**:

- Fetches only direct flights
- to the biggest airports of the respective cities.

Our Rest API: Travel & Tourism API Interface

http://127.0.0.1:8000/docs#/



Travel&Tourism 0.1.0 OAS 3.1
/openapi.json

default

GET /hotels/{city_name} Get Hotels

GET /attractions/{city_name} Get Attractions

GET /flights/ Get Flights

Parameters | Try it out

Name	Description
from_city_name * required string (query)	from_city_name
to_city_name * required string (query)	to_city_name

Responses

Code	Description	Links
200	Successful Response	No links
422	Validation Error	No links

200 — Media type: application/json. Controls Accept header.
Example Value | Schema

```
"string"
```

422 — Media type: application/json
Example Value | Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

Schemas

HTTPValidationError > Expand all object

ValidationError > Expand all object

A Small Story

- My teammate and me are planning a small vacation during Christmas break due to ongoing deadlines and submission struggles this semester.

- We thoroughly considered various options to determine the most suitable destination.

•But,

- 1 City really grabbed our attention.
    - One is New York

- Now, Let’s explore NY with our API.

DEMO

Hope you like it!

Thank you.