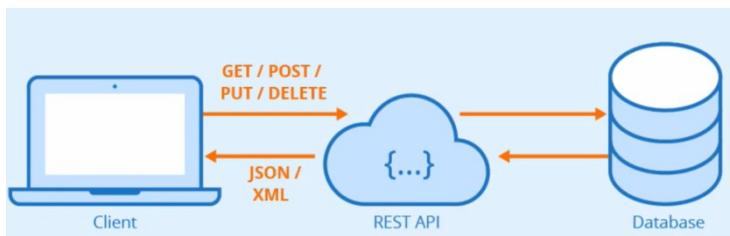


**Test Automation using:**

- Python
- PyCharm
- FastAPI
- Pydantic library
- GitHub repository

**Document highlight (main.py):**

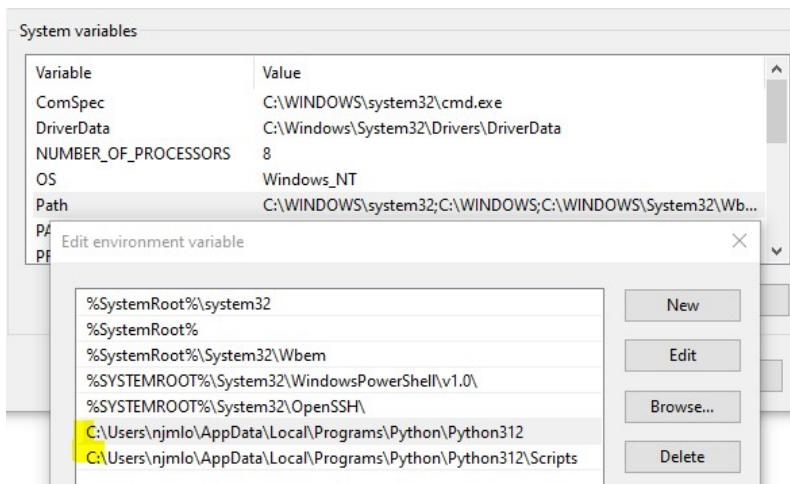
- GET method
- POST method
- PUT method
- PATCH method
- DELETE method

**1. API Concept (interface between client and server (database), request and response)****2. Python (version at the time of documentation)**

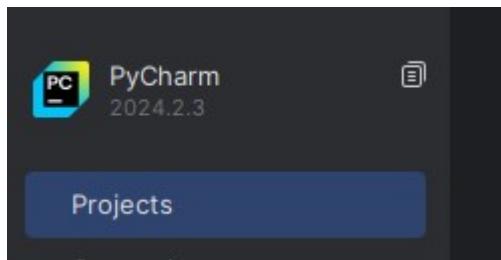
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

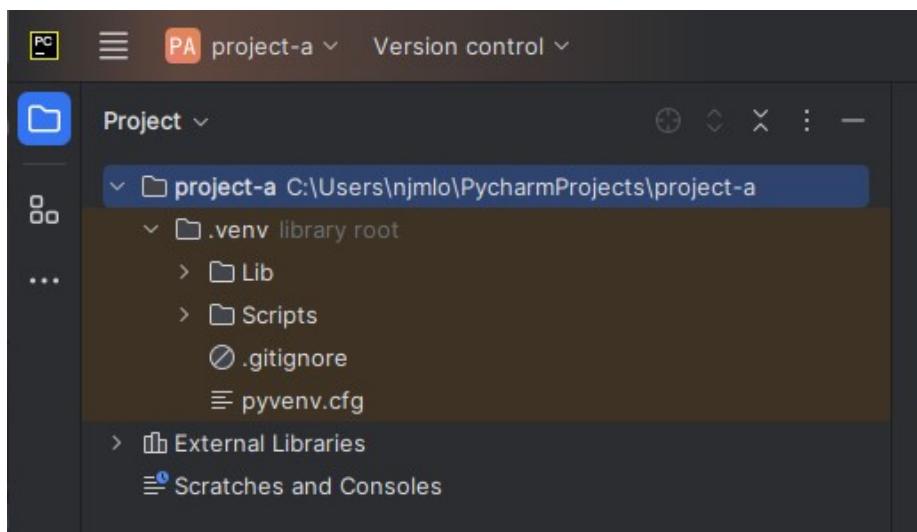
PS C:\Users\njmlo> python -V
Python 3.12.6
PS C:\Users\njmlo>
```

**3. Added under System Variable Path**

#### 4. PyCharm Community Edition (IDE version at the time of documentation)



#### 5. Create a New Project



#### 6. Install FastAPI Web Framework

```
Terminal Local × + ▾
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

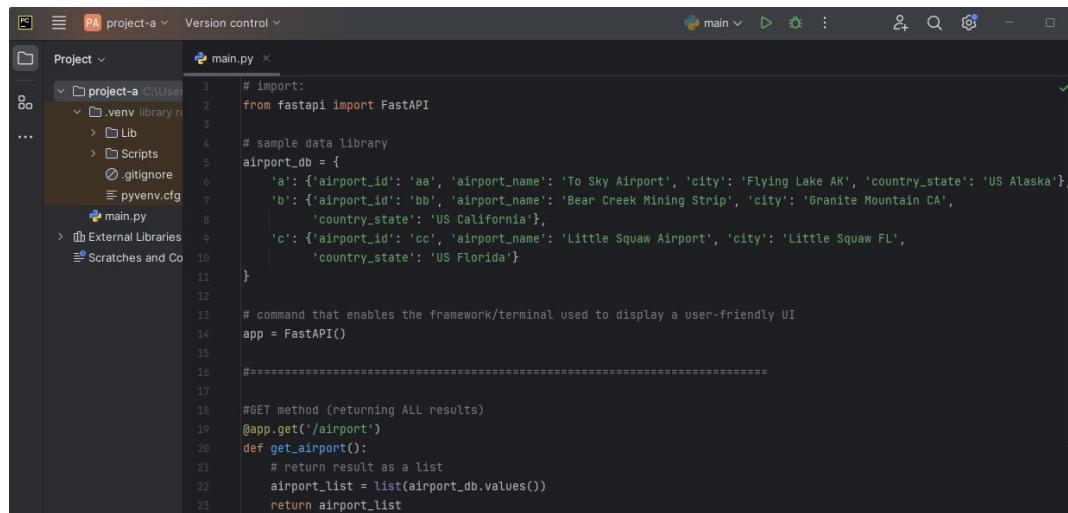
> (.venv) PS C:\Users\njmlo\PycharmProjects\project-a> pip install fastapi
```

```
Installing collected packages: typing-extensions, sniffio, idna, annotated-types, pydantic-core, anyio, starlette, pydantic, fastapi
Successfully installed annotated-types-0.7.0 anyio-4.0.0 fastapi-0.115.0 idna-3.10 pydantic-2.9.2 pydantic-core-2.23.4 sniffio-1.3.1 starlette-0.38.6 typing-extensions-4.12.2
[notice] A new release of pip is available: 23.2.1 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a>
```

## 7. Install Uvicorn Web Server

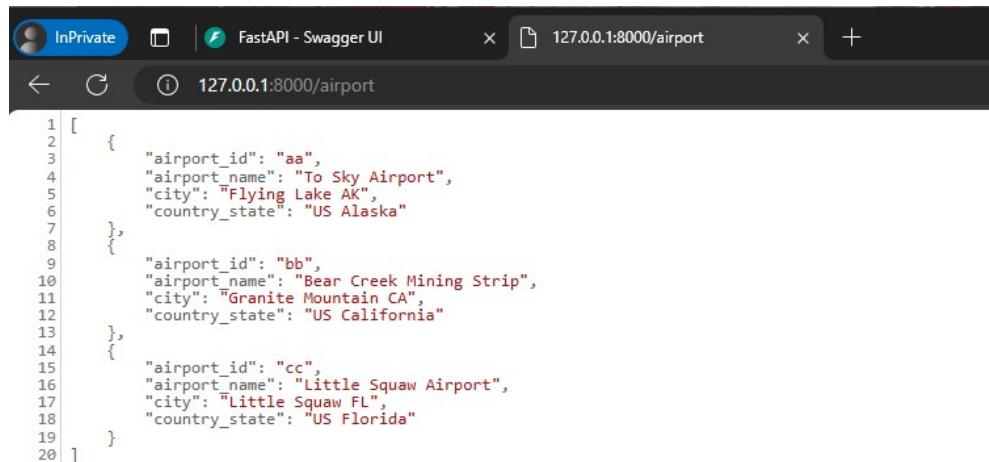
```
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> pip install "uvicorn [standard]"  
|  
Installing collected packages: websockets, pyyaml, python-dotenv, httptools, h11, colorama, watchfiles, click, uvicorn  
Successfully installed click-8.1.7 colorama-0.4.6 h11-0.14.0 httptools-0.6.1 python-dotenv-1.0.1 pyyaml-6.0.2 uvicorn-0.30.6 watchfiles-0.24.0 websockets-13.1  
[notice] A new release of pip is available: 23.2.1 -> 24.2  
[notice] To update, run: python.exe -m pip install --upgrade pip  
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a>
```

## 8. Sample GET (code in PyCharm, reload Uvicorn, and display ALL results with FastAPI)

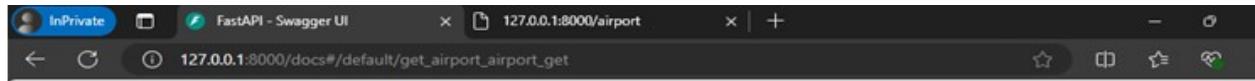


```
Project: project-a  
File: main.py  
1 # import:  
2     from fastapi import FastAPI  
3  
4 # sample data library  
5 airport_db = {  
6     'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US Alaska'},  
7     'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA',  
8           'country_state': 'US California'},  
9     'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL',  
10           'country_state': 'US Florida'}  
11 }  
12  
13 # command that enables the framework/terminal used to display a user-friendly UI  
14 app = FastAPI()  
15  
#=====  
16 #GET method (returning ALL results)  
17 @app.get('/airport')  
18 def get_airport():  
19     # return result as a list  
20     airport_list = list(airport_db.values())  
21     return airport_list
```

```
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> uvicorn main:app --reload  
INFO: Will watch for changes in these directories: ['C:\\\\Users\\\\njmlo\\\\PycharmProjects\\\\project-a']  
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)  
INFO: Started reloader process [11212] using WatchFiles  
INFO: Started server process [19650]  
INFO: Waiting for application startup.  
INFO: Application startup complete.
```



```
[  
1  {  
2      "airport_id": "aa",  
3      "airport_name": "To Sky Airport",  
4      "city": "Flying Lake AK",  
5      "country_state": "US Alaska"  
6  },  
7  {  
8      "airport_id": "bb",  
9      "airport_name": "Bear Creek Mining Strip",  
10     "city": "Granite Mountain CA",  
11     "country_state": "US California"  
12  },  
13  {  
14      "airport_id": "cc",  
15      "airport_name": "Little Squaw Airport",  
16      "city": "Little Squaw FL",  
17      "country_state": "US Florida"  
18  }  
20 ]
```



# FastAPI 0.1.0 OAS 3.1

/openapi.json

## default

GET /airport Get Airport

### Parameters

No parameters

[Cancel](#)

### Servers

These operation-level options override the global server options.

/

[Execute](#)

[Clear](#)

### Responses

#### Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/airport' \
  -H 'accept: application/json'
```

[Copy](#)

#### Request URL

<http://127.0.0.1:8000/airport>

#### Server response

Code Details

200

#### Response body

```
[  
  {  
    "airport_id": "aa",  
    "airport_name": "To Sky Airport",  
    "city": "Flying Lake AK",  
    "country_state": "US Alaska"  
  },  
  {  
    "airport_id": "bb",  
    "airport_name": "Bear Creek Mining Strip",  
    "city": "Granite Mountain CA",  
    "country_state": "US California"  
  },  
  {  
    "airport_id": "cc",  
    "airport_name": "Little Squaw Airport",  
    "city": "Little Squaw FL",  
    "country_state": "US Florida"  
  }]
```

[Copy](#)

[Download](#)

#### Response headers

```
content-length: 339  
content-type: application/json  
date: Fri, 04 Oct 2024 18:36:40 GMT  
server: unicorn
```

### Responses

Code Description

Links

200

Successful Response

No links

#### Media type

[application/json](#)

[▼](#)

Controls Accept header.

[Example Value](#) | [Schema](#)

## 9. Sample GET (code in PyCharm, reload Uvicorn, and display LIMITED results with FastAPI)

The screenshot shows the PyCharm IDE interface with the project 'project-a' open. The main.py file is selected and displayed in the editor. The code implements a FastAPI application that returns a limited list of airports from a database.

```
from fastapi import FastAPI
# sample data library
airport_db = {
    'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US Alaska'},
    'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA',
          'country_state': 'US California'},
    'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL',
          'country_state': 'US Florida'}
}
# command that enables the framework/terminal used to display a user-friendly UI
app = FastAPI()
#=====
#GET method (returning LIMITED results)
@app.get('/airport')
def get_airport_query(limit: int):
    # return limited result as a list
    airport_list = list(airport_db.values())
    return airport_list [:limit]
```

The screenshot shows a Windows PowerShell terminal window. It displays the command `uvicorn main:app --reload` being run, followed by the application's startup logs. The logs indicate that Uvicorn is running on port 8000 and a reloader process has started.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\\\Users\\\\njmlo\\\\PycharmProjects\\\\project-a']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [51104] using WatchFiles
INFO:     Started server process [51380]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

The screenshot shows a Microsoft Edge browser window titled 'FastAPI - Swagger UI'. The URL is `127.0.0.1:8000/airport?limit=2`. The response is a JSON array containing two airport entries, which are the first two entries from the database defined in main.py.

```
[{"airport_id": "aa", "airport_name": "To Sky Airport", "city": "Flying Lake AK", "country_state": "US Alaska"}, {"airport_id": "bb", "airport_name": "Bear Creek Mining Strip", "city": "Granite Mountain CA", "country_state": "US California"}]
```

InPrivate FastAPI - Swagger UI 127.0.0.1:8000/airport?limit=2 - O

127.0.0.1:8000/docs#/default/get\_airport\_query\_airport\_get

FastAPI 0.1.0 OAS 3.1  
[/openapi.json](#)

## default

**GET** /airport Get Airport Query

**Parameters**

Name Description

limit \* required integer (query) 2

**Servers**

These operation-level options override the global server options.

/

**Execute** **Clear**

**Responses**

**Curl**

```
curl -X 'GET' \
  'http://127.0.0.1:8000/airport?limit=2' \
  -H 'Accept: application/json'
```

**Request URL**

<http://127.0.0.1:8000/airport?limit=2>

**Server response**

Code	Details
200	<p><b>Response body</b></p> <pre>[{"airport_id": "aa", "airport_name": "To Sky Airport", "city": "Flying Lake AK", "country_state": "US Alaska"}, {"airport_id": "bb", "airport_name": "Bear Creek Mining Strip", "city": "Granite Mountain CA", "country_state": "US California"}]</pre> <p><b>Download</b></p> <p><b>Response headers</b></p> <pre>content-length: 227 content-type: application/json date: Fri, 04 Oct 2024 19:07:53 GMT server: uvicorn</pre>

**Responses**

Code	Description	Links
200	Successful Response	No links

Media type

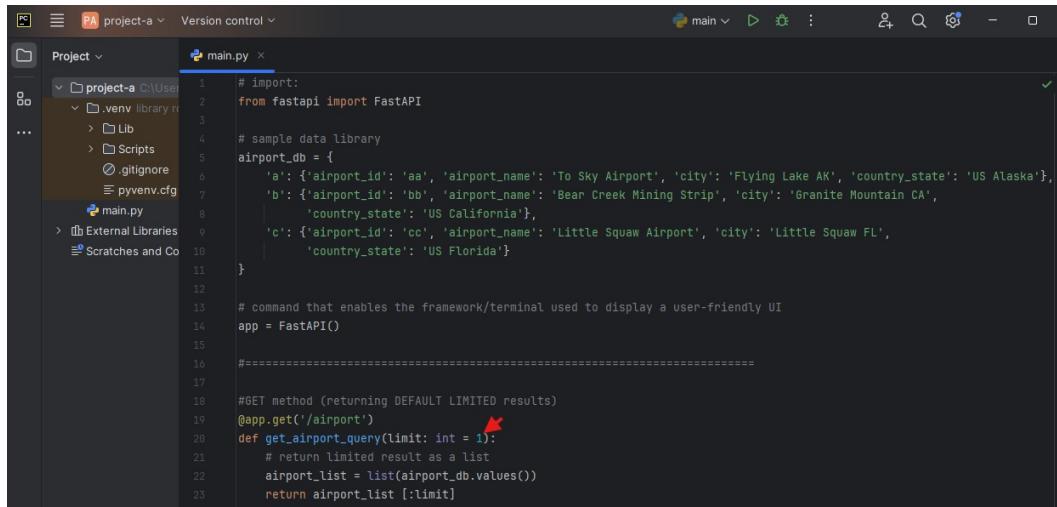
application/json

Controls Accept header.

Example Value | Schema

"string"

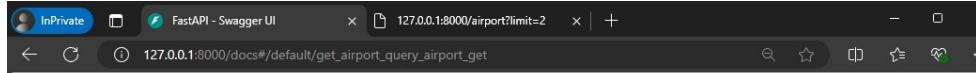
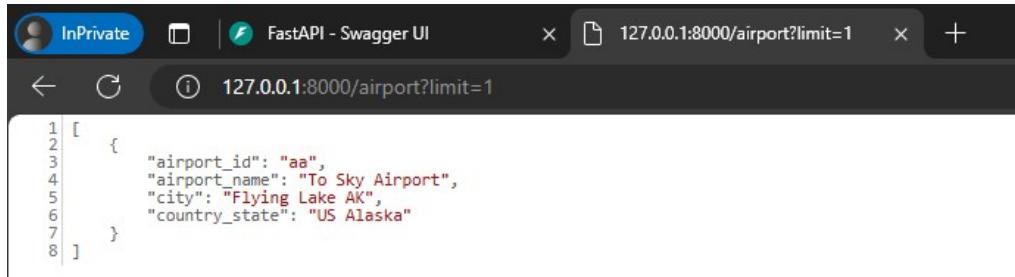
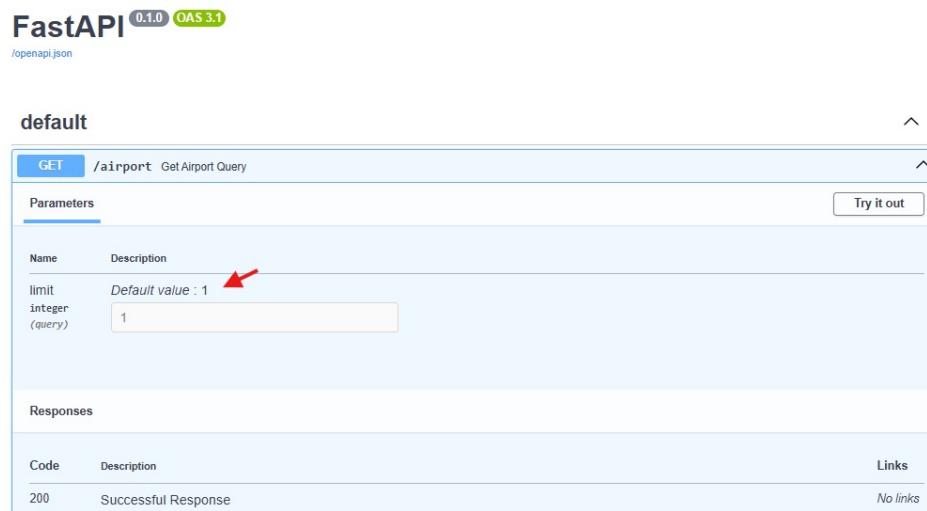
**10. Sample GET (code in PyCharm (with default parm value), reload Uvicorn, and display LIMITED results with FastAPI)**



```

1 # import:
2 from fastapi import FastAPI
3
4 # sample data library
5 airport_db = {
6     'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US Alaska'},
7     'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA',
8           'country_state': 'US California'},
9     'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL',
10        'country_state': 'US Florida'}
11 }
12
13 # command that enables the framework/terminal used to display a user-friendly UI
14 app = FastAPI()
15
16 #####
17
18 #GET method (returning DEFAULT LIMITED results)
19 @app.get('/airport')
20 def get_airport_query(limit: int = 1):
21     # return limited result as a list
22     airport_list = list(airport_db.values())
23     return airport_list [:limit]

```

FastAPI 0.1.0 OAS 3.1

/openapi.json

### default

GET /airport Get Airport Query

Parameters	
Name	Description
limit (query)	Default value : 1

Responses

Code	Description	Links
200	Successful Response	No links

InPrivate FastAPI - Swagger UI 127.0.0.1:8000/airport?limit=1

127.0.0.1:8000/docs#/default/get\_airport\_query\_airport\_get

# FastAPI 0.1.0 OAS 3.1

/openapi.json

## default

GET /airport Get Airport Query

Parameters

Name	Description
limit	integer (query)

Servers

These operation-level options override the global server options.

/

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://127.0.0.1:8000/airport?limit=1' \
-H 'accept: application/json'
```

Request URL

http://127.0.0.1:8000/airport?limit=1

Server response

Code Details

200 Response body

```
[{"airport_id": "aa", "airport_name": "To Sky Airport", "city": "Flying Lake AK", "country_state": "US Alaska"}]
```

Download

Response headers

```
content-length: 105
content-type: application/json
date: Fri, 04 Oct 2024 19:13:51 GMT
server: uvicorn
```

Responses

Code Description Links

200 Successful Response No links

Media type

application/json

Controls Accept header

Example Value | Schema

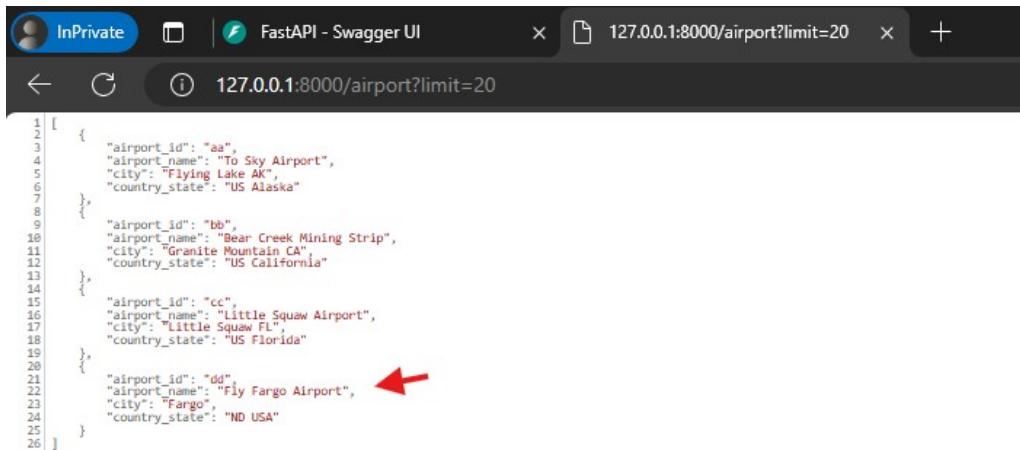
```
"string"
```

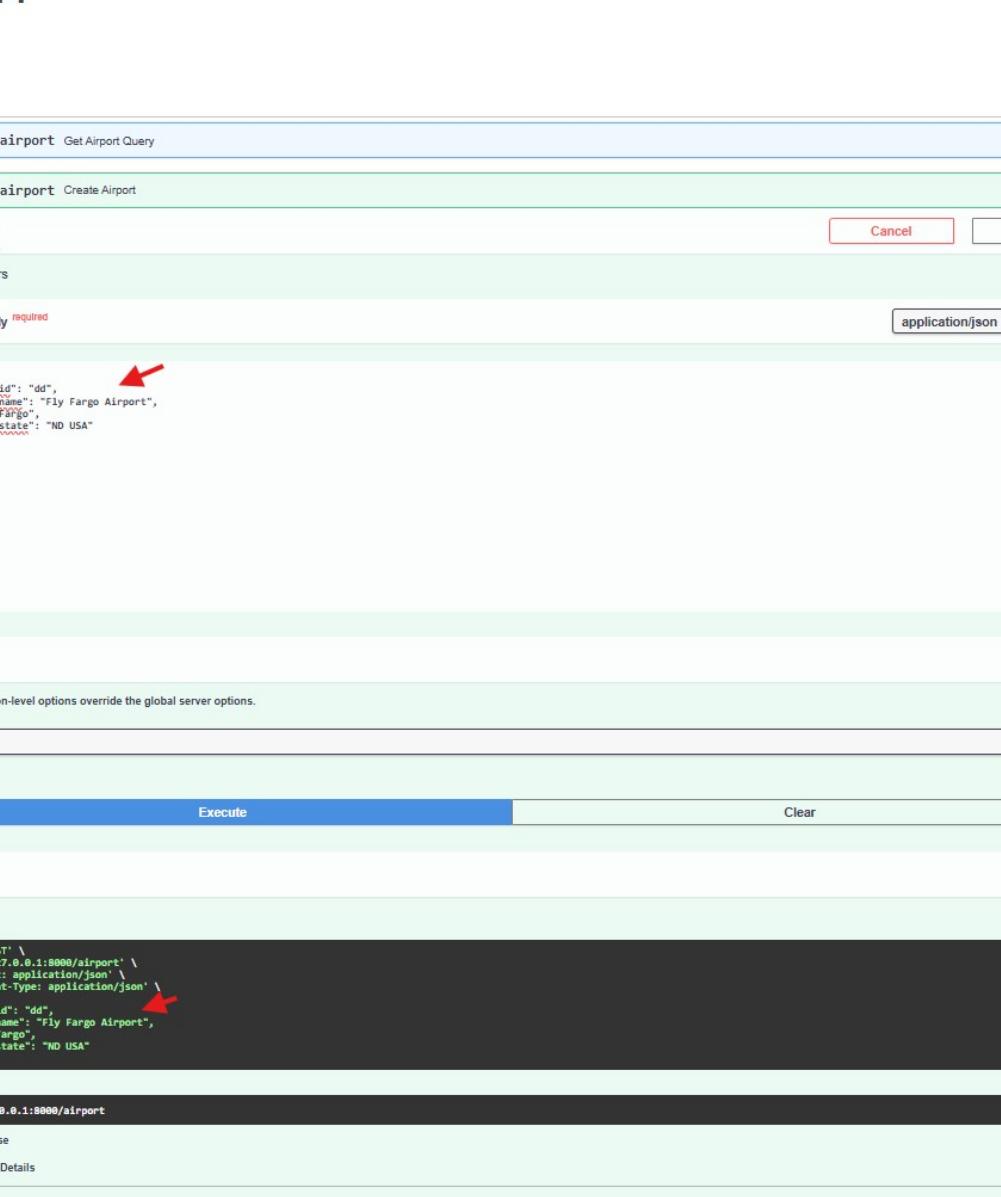
11. Sample POST (code in PyCharm (create data), reload Unicorn, and display (added data) results with FastAPI (using the GET method))

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- Project Explorer (left sidebar):** Shows a project named "project-a" containing files like "main.py", "pyenv.cfg", and "pyenv.local". It also lists "External Libraries" and "Scratches and Co".
- Code Editor (center):** The file "main.py" is open, displaying Python code for a FastAPI application. The code includes imports for FastAPI and Pydantic, defines a database of airports, creates a model class for airports, and sets up a FastAPI application with GET and POST routes.
- Status Bar (top right):** Shows the current branch as "main", along with other status icons.

```
1 # import:
2 from fastapi import FastAPI
3 from pydantic import BaseModel
4
5 # sample data library
6 airport_db = {
7     'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US Alaska'},
8     'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA',
9           'country_state': 'US California'},
10    'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL',
11          'country_state': 'US Florida'}
12 }
13
14 # the class used to create a data into the data library or db
15 class Airport(BaseModel):
16     airport_id: str
17     airport_name: str
18     city: str
19     country_state: str
20
21 # command that enables the framework/terminal used to display a user-friendly UI
22 app = FastAPI()
23
24 =====
25
26 #GET method (returning DEFAULT LIMITED results)
27 @app.get('/airport')
28 def get_airport_query(limit: int = 20):
29     # return limited result as a list
30     airport_list = list(airport_db.values())
31     return airport_list [:limit]
32
33 # POST method (creating data in the data library or db)
34 @app.post('/airport')
35 def create_airport(airport: Airport):
36     airport_id = airport.airport_id
37     airport_db[airport_id] = airport.model_dump()
38     return {'message': f'Successfully created airport: {airport_id}'}
```





The screenshot shows the FastAPI - Swagger UI interface at [127.0.0.1:8000/docs#](http://127.0.0.1:8000/docs#/). The main page displays the **default** API documentation. Under the **POST /airport** endpoint, there is a **Request body** section marked as required, containing the following JSON schema:

```
{  
    "airport_id": "dd",  
    "airport_name": "Fly Fargo Airport",  
    "city": "Fargo",  
    "country_state": "ND USA"  
}
```

A red arrow points to the JSON code in this section. Below the request body, there is a **Servers** section with a note about overriding global server options, a **Curl** command, and a **Request URL** input field set to <http://127.0.0.1:8000/airport>. The **Responses** section shows a successful **Code 200** response with a **Response body** containing the message: "Successfully created airport: dd". A red arrow points to this message. The **Response headers** section lists standard HTTP headers.

## 12. Sample POST (code in PyCharm (create data), reload Uvicorn, and display (added data) results with FastAPI (with enhanced pydantic validation added within the class))

```

1 # import:
2 from fastapi import FastAPI
3 from pydantic import BaseModel, Field # (to be able to use further pydantic validation)
4
5 # to be able to use further pydantic validation
6 from typing import Optional
7
8 # sample data library
9 airport_db = {
10     'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US Alaska'},
11     'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA',
12           'country_state': 'US California'},
13     'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL',
14           'country_state': 'US Florida'}
15 }
16
17 # the class used to create a data into the data library or db
18 class Airport(BaseModel): 	usage
19     airport_id: str = Field(min_length=2, max_length=20) # (utilizing further pydantic validation)
20     airport_name: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
21     city: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
22     country_state: Optional[str] = None # (this makes the field not mandatory)
23
24 # command that enables the framework/terminal used to display a user-friendly UI
25 app = FastAPI()
26
27 #=====
28
29 #GET method (returning DEFAULT LIMITED results)
30 @app.get('/airport')
31 def get_airport_query(limit: int = 20):
32     # return limited result as a list
33     airport_list = list(airport_db.values())
34     return airport_list [:limit]
35
36 # POST method (creating data in the data library or db)
37 @app.post('/airport')
38 def create_airport(airport: Airport):
39     airport_id = airport.airport_id
40     airport_db[airport_id] = airport.model_dump()
41     return {'message': f'Successfully created airport: {airport_id}'}

```

← ⌂ ⓘ 127.0.0.1:8000/docs

FastAPI 0.1.0 OAS 3.1

/openapi.json

default

**GET** /airport Get Airport Query

**POST** /airport Create Airport

Schemas

Airport ▾ Collapse all object

```

airport_id* string [2, 20] characters
airport_name* string [3, 20] characters
city* string [3, 20] characters
country_state > Expand all (string | null)

```

HTTPValidationError > Expand all object

ValidationError > Expand all object

## When executed to trigger the error validation

FastAPI 0.1.0 OAS 3.1  
[/openapi.json](#)

default

GET /airport Get Airport Query

POST /airport Create Airport

Parameters

No parameters

Request body required

```
{ "airport_id": "E", "airport_name": "string", "city": "string", "country_state": "string" }
```

Cancel Reset

Servers

These operation-level options override the global server options.

/

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/airport' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "airport_id": "E",
    "airport_name": "string",
    "city": "string",
    "country_state": "string"
  }'
```

Request URL

http://127.0.0.1:8000/airport

Server response

Code Details ↗

422 Error: Unprocessable Entity

Response body

```
{
  "detail": [
    {
      "type": "string_too_short",
      "loc": [
        "body",
        "airport_id"
      ],
      "msg": "String should have at least 2 characters",
      "param": "E",
      "ctx": {
        "min_length": 2
      }
    }
  ]
}
```

Download

Response headers

```
content-length: 152
content-type: application/json
date: Fri, 04 Oct 2024 19:58:33 GMT
server: uvicorn
```

Terminal Local

```
git: get_airport_query: command not found
# return limited result as a list
INFO: 127.0.0.1:51066 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:51073 - "POST /airport HTTP/1.1" 422 Unprocessable Entity ↗
```

## When executed without any error

FastAPI - Swagger UI | 127.0.0.1:8000/airport?limit=20

```
[  
  {  
    "airport_id": "aa",  
    "airport_name": "To Sky Airport",  
    "city": "Flying Lake AK",  
    "country_state": "US Alaska"  
  },  
  {  
    "airport_id": "bb",  
    "airport_name": "Bear Creek Mining Strip",  
    "city": "Granite Mountain CA",  
    "country_state": "US California"  
  },  
  {  
    "airport_id": "cc",  
    "airport_name": "Little Squaw Airport",  
    "city": "Little Squaw FL",  
    "country_state": "US Florida"  
  },  
  {  
    "airport_id": "dd",  
    "airport_name": "Fly Fargo Airport",  
    "city": "Fargo",  
    "country_state": "ND USA"  
  }  
]
```

127.0.0.1:8000/docs#/default/create\_airport\_airport\_post

FastAPI 0.1.0 OAS 3.1

default

GET /airport Get Airport Query

POST /airport Create Airport

Parameters

No parameters

Request body required

```
{  
  "airport_id": "dd",  
  "airport_name": "Fly Fargo Airport",  
  "city": "Fargo",  
  "country_state": "ND USA"  
}
```

Servers

These operation-level options override the global server options.

/

Execute Clear

Responses

Curl

```
curl -X 'POST' \  
  'http://127.0.0.1:8000/airport' \  
  -H 'Content-Type: application/json' \  
  -d '{  
    "airport_id": "dd",  
    "airport_name": "Fly Fargo Airport",  
    "city": "Fargo",  
    "country_state": "ND USA"  
  }'
```

Request URL

http://127.0.0.1:8000/airport

Server response

Code Details

200 Response body

```
{  
  "message": "Successfully created airport: dd"  
}
```

Download

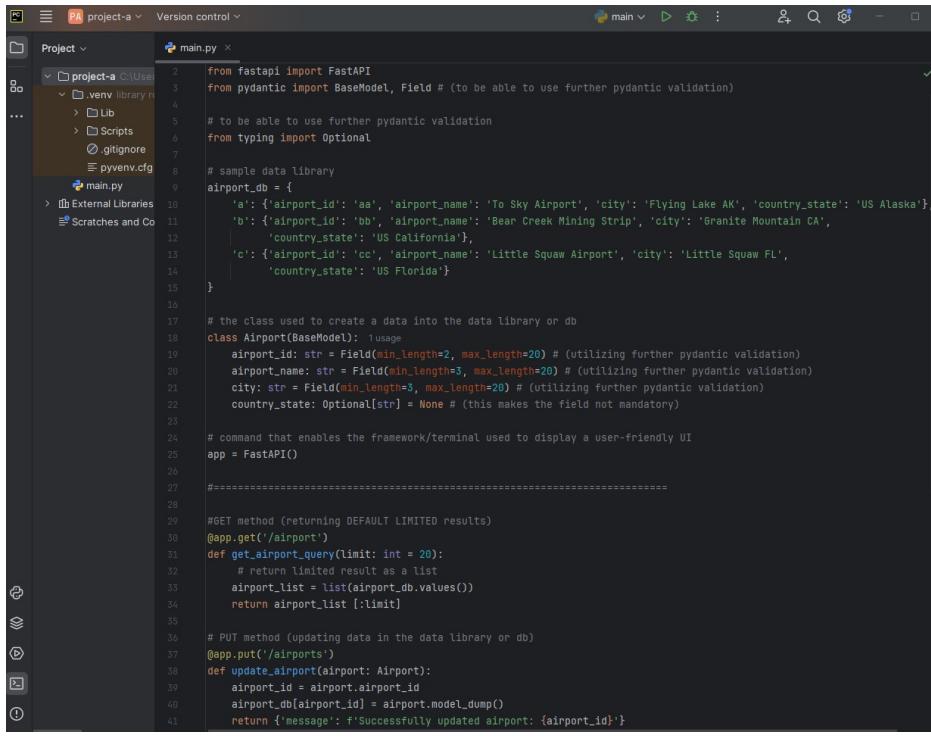
Response headers

```
content-length: 46  
content-type: application/json  
date: Fri, 04 Oct 2024 20:02:51 GMT  
server: uvicorn
```

^

Cancel Reset application/json

### 13. Sample PUT (code in PyCharm, reload Uvicorn, and display results (after update) with FastAPI)



```
from fastapi import FastAPI
from pydantic import BaseModel, Field # (to be able to use further pydantic validation)
# to be able to use further pydantic validation
from typing import Optional
# sample data library
airport_db = {
    'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US Alaska'},
    'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA',
          'country_state': 'US California'},
    'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL',
          'country_state': 'US Florida'}
}
# the class used to create a data into the data library or db
class Airport(BaseModel):
    usage
        airport_id: str = Field(min_length=2, max_length=20) # (utilizing further pydantic validation)
        airport_name: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
        city: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
        country_state: Optional[str] = None # (this makes the field not mandatory)
# command that enables the framework/terminal used to display a user-friendly UI
app = FastAPI()
#=====
#GET method (returning DEFAULT LIMITED results)
@app.get('/airports')
def get_airport_query(limit: int = 20):
    # return limited result as a list
    airport_list = list(airport_db.values())
    return airport_list [:limit]
# PUT method (updating data in the data library or db)
@app.put('/airports')
def update_airport(airport: Airport):
    airport_id = airport.airport_id
    airport_db[airport_id] = airport.model_dump()
    return {'message': f'Successfully updated airport: {airport_id}'}
```

```
19 },
20 {
21     "airport_id": "bb",
22     "airport_name": "Be Free Airport",
23     "city": "Watcom",
24     "country_state": "USA Washington"
25 }
26 ]
```

← ⏪ ⓘ 127.0.0.1:8000/docs#/ ⏴ ⏵ ⏷ ⏸ ⏹ ⏺ ⏻ ⏻ ⏻

# FastAPI 0.1.0 OAS 3.1

/openapi.json

## default

GET /airport Get Airport Query

PUT /airports Update Airport

Parameters

No parameters

Request body required

application/json

```
{ "airport_id": "bb", "airport_name": "Be Free Airport", "city": "Watcom", "country_state": "USA Washington" }
```

Servers

These operation-level options override the global server options.

/

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
  'http://127.0.0.1:8000/airports' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "airport_id": "bb",
    "airport_name": "Be Free Airport",
    "city": "Watcom",
    "country_state": "USA Washington"
}'
```

Request URL

http://127.0.0.1:8000/airports

Server response

Code Details  
200

Response body

```
{ "message": "Successfully updated airport: bb" }
```

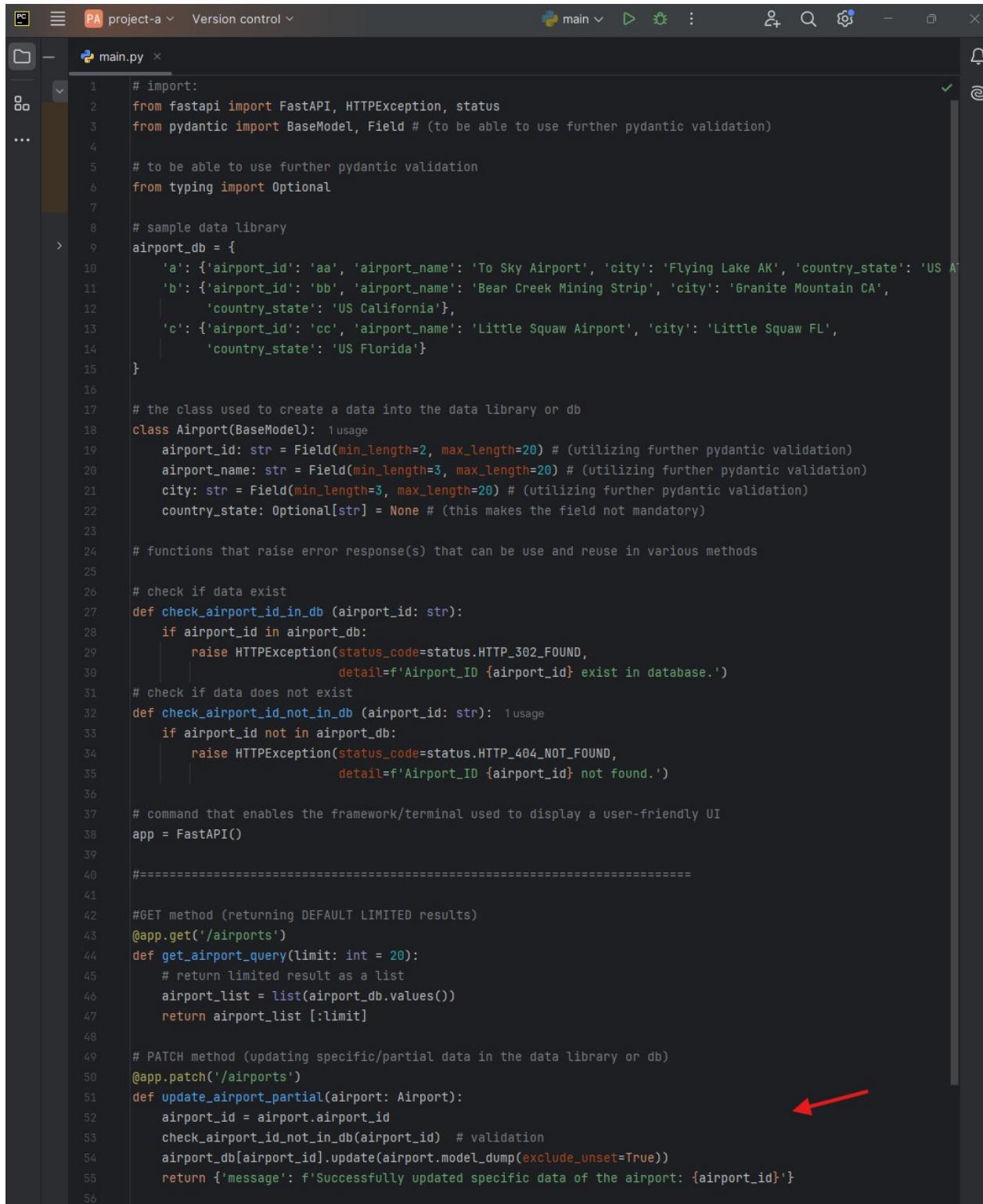
Download

Response headers

```
content-length: 46
content-type: application/json
date: Fri, 04 Oct 2024 23:58:37 GMT
server: uvicorn
```

The screenshot shows the FastAPI documentation for the PUT /airports endpoint. It includes a JSON request body with an 'airport\_id' field. A red arrow points to the 'airport\_id' field. Another red arrow points to the 'airport\_name' field in the curl command. A third red arrow points to the 'message' field in the JSON response body.

#### 14. Sample PATCH (code in PyCharm, updating specific data element of the data)



```
# import:
from fastapi import FastAPI, HTTPException, status
from pydantic import BaseModel, Field # (to be able to use further pydantic validation)
# to be able to use further pydantic validation
from typing import Optional

# sample data library
airport_db = {
    'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US AK'},
    'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA',
          'country_state': 'US California'},
    'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL',
          'country_state': 'US Florida'}
}

# the class used to create a data into the data library or db
class Airport(BaseModel): 1 usage
    airport_id: str = Field(min_length=2, max_length=20) # (utilizing further pydantic validation)
    airport_name: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
    city: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
    country_state: Optional[str] = None # (this makes the field not mandatory)

# functions that raise error response(s) that can be use and reuse in various methods

# check if data exist
def check_airport_id_in_db (airport_id: str):
    if airport_id in airport_db:
        raise HTTPException(status_code=status.HTTP_302_FOUND,
                            detail=f'Airport_ID {airport_id} exist in database.')
# check if data does not exist
def check_airport_id_not_in_db (airport_id: str): 1 usage
    if airport_id not in airport_db:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
                            detail=f'Airport_ID {airport_id} not found.')

# command that enables the framework/terminal used to display a user-friendly UI
app = FastAPI()

#####
#GET method (returning DEFAULT LIMITED results)
@app.get('/airports')
def get_airport_query(limit: int = 20):
    # return limited result as a list
    airport_list = list(airport_db.values())
    return airport_list [:limit]

# PATCH method (updating specific/partial data in the data library or db)
@app.patch('/airports')
def update_airport_partial(airport: Airport):
    airport_id = airport.airport_id
    check_airport_id_not_in_db(airport_id) # validation
    airport_db[airport_id].update(airport.model_dump(exclude_unset=True))
    return {'message': f'Successfully updated specific data of the airport: {airport_id}'}
```

## 15. Sample DELETE (code in PyCharm), deleting specific data

```
# import:
from fastapi import FastAPI, HTTPException, status
from pydantic import BaseModel, Field # (to be able to use further pydantic validation)

# to be able to use further pydantic validation
from typing import Optional

# sample data library
airport_db = {
    'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US Alaska'},
    'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA', 'country_state': 'US California'},
    'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL', 'country_state': 'US Florida'}
}

# the class used to create a data into the data library or db
class Airport(BaseModel):
    airport_id: str = Field(min_length=2, max_length=20) # (utilizing further pydantic validation)
    airport_name: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
    city: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
    country_state: Optional[str] = None # (this makes the field not mandatory)

# functions that raise error response(s) that can be use and reuse in various methods

# check if data exist
def check_airport_id_in_db(airport_id: str):
    if airport_id in airport_db:
        raise HTTPException(status_code=status.HTTP_302_FOUND,
                            detail=f'Airport_ID {airport_id} exist in database.')
    # check if data does not exist
def check_airport_id_not_in_db(airport_id: str): 1 usage
    if airport_id not in airport_db:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
                            detail=f'Airport_ID {airport_id} not found.')

# command that enables the framework/terminal used to display a user-friendly UI
app = FastAPI()

#=====
#GET method (returning DEFAULT LIMITED results)
@app.get('/airports')
def get_airport_query(limit: int = 20):
    # return limited result as a list
    airport_list = list(airport_db.values())
    return airport_list [:limit]

# DELETE method (deleting data from the data library or db)
@app.delete('/airports/{airport_id}')
def delete_airport(airport_id: str): ←
    check_airport_id_not_in_db(airport_id) # validation
    del airport_db[airport_id]
    return {'message': f'Successfully deleted airport: {airport_id}'}
```

## 16. Sample POST method that captures error responses

A sampling when a data exist

```
# import:
from fastapi import FastAPI, HTTPException, status
from pydantic import BaseModel, Field # (to be able to use further pydantic validation)

# to be able to use further pydantic validation
from typing import Optional

# sample data library
airport_db = {
    'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US AK'},
    'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA',
          'country_state': 'US California'},
    'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL',
          'country_state': 'US Florida'}
}

# the class used to create a data into the data library or db
class Airport(BaseModel):
    airport_id: str = Field(min_length=2, max_length=20) # (utilizing further pydantic validation)
    airport_name: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
    city: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
    country_state: Optional[str] = None # (this makes the field not mandatory)

# functions that raise error response(s) that can be use and reuse in various methods

# check if data exist
def check_airport_id_in_db(airport_id: str):
    if airport_id in airport_db:
        raise HTTPException(status_code=status.HTTP_302_FOUND,
                            detail=f'Airport_ID {airport_id} exist in database.')
    # check if data does not exist
def check_airport_id_not_in_db(airport_id: str):
    if airport_id not in airport_db:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
                            detail=f'Airport_ID {airport_id} not found.')

# command that enables the framework/terminal used to display a user-friendly UI
app = FastAPI()

#=====
# POST method (creating data in the data library or db)
@app.post('/airport')
def create_airport(airport: Airport):
    airport_id = airport.airport_id
    check_airport_id_in_db(airport_id) #validation
    airport_db[airport_id] = airport.model_dump()
    return {'message': f'Successfully created airport: {airport_id}'}
```

# FastAPI 0.1.0 OAS 3.1

/openapi.json

## default

**POST /airport Create Airport**

**Parameters**

No parameters

**Request body required**

application/json

```
{  
    "airport_id": "aa",  
    "airport_name": "Test",  
    "city": "string",  
    "country_state": "string"  
}
```

**Servers**

These operation-level options override the global server options.

/

**Execute** **Clear**

**Responses**

**Curl**

```
curl -X 'POST' \  
      'http://127.0.0.1:8000/airport' \  
      -H 'accept: application/json' \  
      -H 'Content-Type: application/json' \  
      -d '{  
        "airport_id": "aa",  
        "airport_name": "test",  
        "city": "string",  
        "country_state": "string"  
      }'
```

**Request URL**

http://127.0.0.1:8000/airport

**Server response**

**Code** **Details**

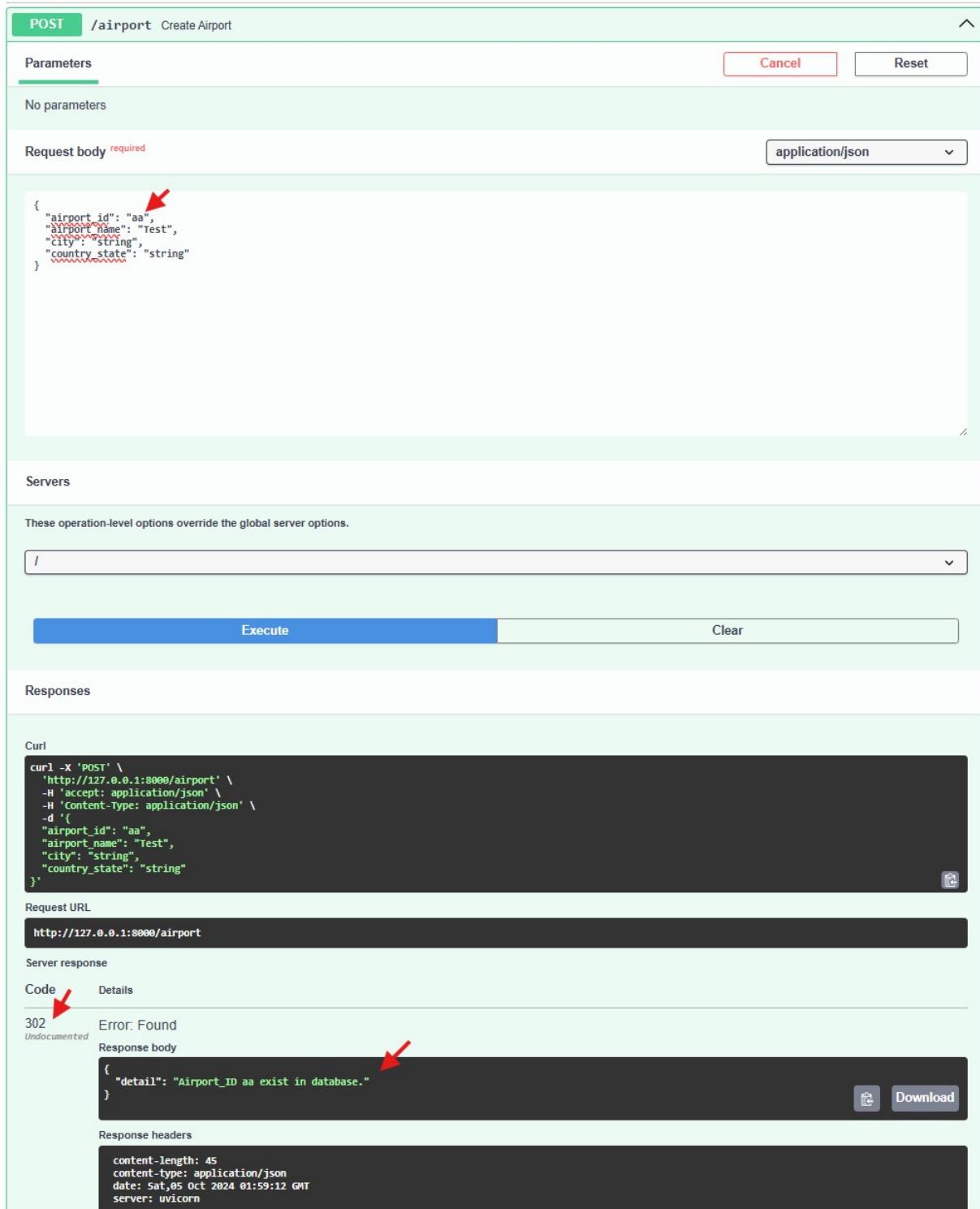
302 Error: Found  
*Undocumented*

{  
 "detail": "Airport\_ID aa exist in database."  
}

**Download**

**Response headers**

```
content-length: 45  
content-type: application/json  
date: Sat, 05 Oct 2024 01:59:12 GMT  
server: uvicorn
```



**17. In summary, the sampling methods used in this publication, with an enhanced page display section tags**

```
# import:
from fastapi import FastAPI, HTTPException, status
from pydantic import BaseModel, Field # (to be able to use further pydantic validation)

# to be able to use further pydantic validation
from typing import Optional

# command that enables the framework/terminal used to display a user-friendly UI
app = FastAPI(
    title='Airports',
    description='This is a sampling APIs for Airport'
)
# text values for the Swagger method group name/tags in FastAPI UI display
tags_metadata = [
    {
        "name": "Airports",
        "description": "Airport list. Link on the documentation on the right.", ←
        "externalDocs": {
            "description": "Sampling Test automation documentation",
            "url": "https://fastapi.tiangolo.com/",
        },
    },
]
# sample data library
airport_db = {
    'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US Alaska'},
    'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA', 'country_state': 'US California'},
    'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL', 'country_state': 'US Florida'}
}
# the class used to create a data into the data library or db
class Airport(BaseModel):
    airport_id: str = Field(min_length=2, max_length=20) # (utilizing further pydantic validation)
    airport_name: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
    city: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
    country_state: Optional[str] = None # (this makes the field not mandatory)
# functions that raise error response(s) that can be use and reuse in various methods
# check if data exist
def check_airport_id_in_db (airport_id: str):
    if airport_id in airport_db:
        raise HTTPException(status_code=status.HTTP_302_FOUND,
                            detail=f'Airport_ID {airport_id} exist in database.')
# check if data does not exist
def check_airport_id_not_in_db (airport_id: str):
    if airport_id not in airport_db:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
                            detail=f'Airport_ID {airport_id} not found.')
# change the Swagger method group name/tags in FastAPI UI display ←
app.openapi_tags=tags_metadata
```

Airports - Swagger UI

127.0.0.1:8000/docs

# Airports 0.1.0 OAS 3.1

/openapi.json

This is a sampling APIs for Airport

**Airports** Airport list. Link on the documentation on the right.

Sampling Test automation documentation ^

**GET** /airports Get Airport Query

**POST** /airports Create Airport

**PUT** /airports Update Airport

**PATCH** /airports Update Airport Partial

**DELETE** /airports/{airport\_id} Delete Airport

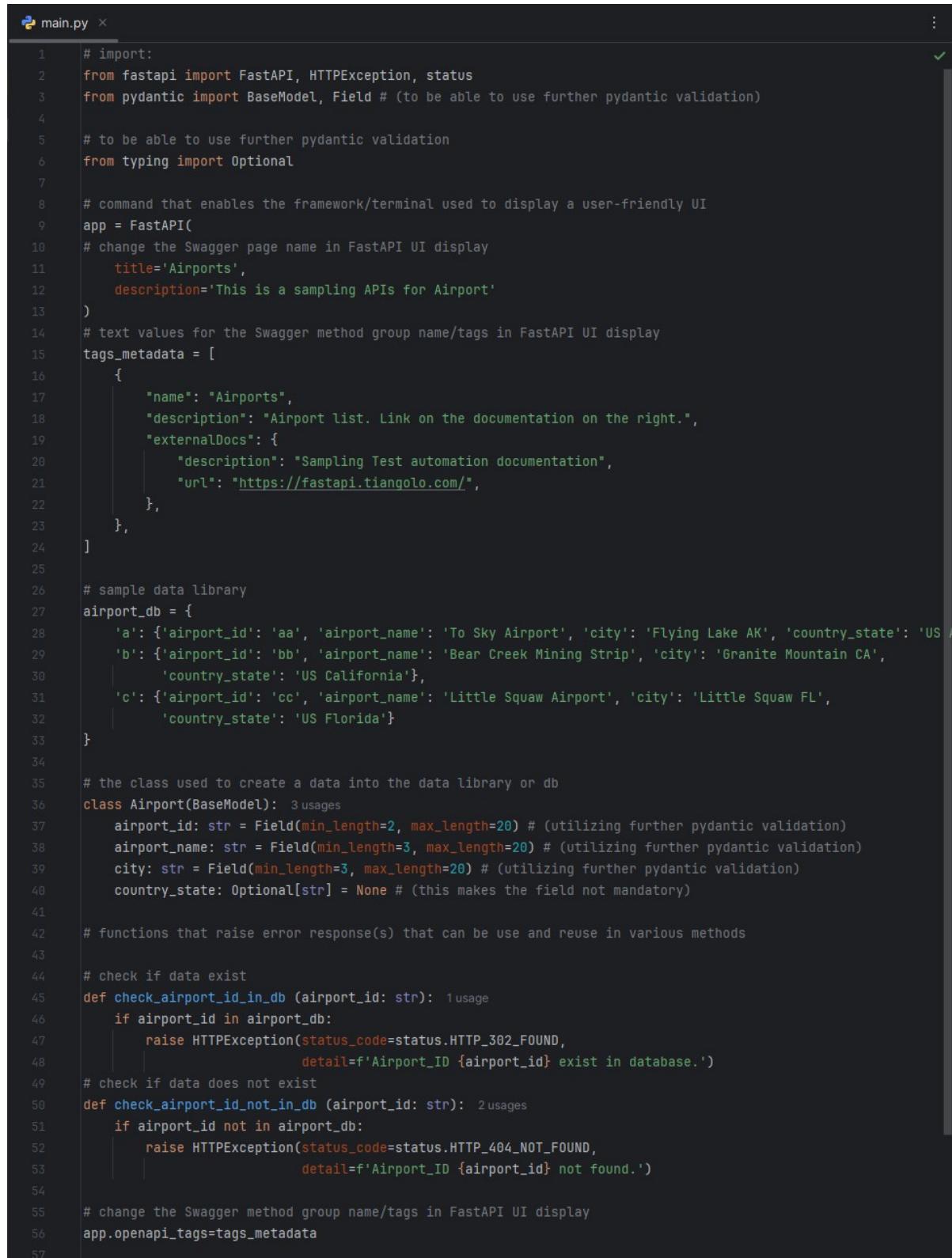
## Schemas

**Airport** > Expand all object

**HTTPValidationError** > Expand all object

**ValidationError** > Expand all object

## 18. Here is a clean end to end snapshot of the Python code in PyCharm



The screenshot shows the PyCharm IDE interface with the file 'main.py' open. The code is a FastAPI application for managing airports. It includes imports for FastAPI, HTTPException, status, Pydantic's BaseModel and Field, and typing's Optional. The app is initialized with a title of 'Airports' and a description of 'This is a sampling APIs for Airport'. A list of tags metadata is defined, each entry containing a name, description, and external documentation URL. A sample database 'airport\_db' is provided with entries for airports 'aa', 'bb', and 'cc'. The class 'Airport' is defined as a BaseModel with fields for airport\_id, airport\_name, city, and country\_state. Two helper functions, 'check\_airport\_id\_in\_db' and 'check\_airport\_id\_not\_in\_db', are defined to handle error responses based on the existence of an airport ID in the database. Finally, the app is configured to use the tags\_metadata defined earlier.

```
# import:
from fastapi import FastAPI, HTTPException, status
from pydantic import BaseModel, Field # (to be able to use further pydantic validation)

# to be able to use further pydantic validation
from typing import Optional

# command that enables the framework/terminal used to display a user-friendly UI
app = FastAPI(
    title='Airports',
    description='This is a sampling APIs for Airport'
)
# text values for the Swagger method group name/tags in FastAPI UI display
tags_metadata = [
    {
        "name": "Airports",
        "description": "Airport list. Link on the documentation on the right.",
        "externalDocs": {
            "description": "Sampling Test automation documentation",
            "url": "https://fastapi.tiangolo.com/"
        },
    },
]
# sample data library
airport_db = {
    'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city': 'Flying Lake AK', 'country_state': 'US Alaska'},
    'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip', 'city': 'Granite Mountain CA', 'country_state': 'US California'},
    'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city': 'Little Squaw FL', 'country_state': 'US Florida'}
}
# the class used to create a data into the data library or db
class Airport(BaseModel):
    airport_id: str = Field(min_length=2, max_length=20) # (utilizing further pydantic validation)
    airport_name: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
    city: str = Field(min_length=3, max_length=20) # (utilizing further pydantic validation)
    country_state: Optional[str] = None # (this makes the field not mandatory)

# functions that raise error response(s) that can be use and reuse in various methods
# check if data exist
def check_airport_id_in_db (airport_id: str): 1 usage
    if airport_id in airport_db:
        raise HTTPException(status_code=status.HTTP_302_FOUND,
                            detail=f'Airport_ID {airport_id} exist in database.')
# check if data does not exist
def check_airport_id_not_in_db (airport_id: str): 2 usages
    if airport_id not in airport_db:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
                            detail=f'Airport_ID {airport_id} not found.')
# change the Swagger method group name/tags in FastAPI UI display
app.openapi_tags=tags_metadata
```

```

57
58     #=====Method calls below=====
59
60     #GET method (returning DEFAULT LIMITED results)
61     @app.get( path: '/airports', tags=["Airports"])
62     def get_airport_query(limit: int = 20):
63         # return limited result as a list
64         airport_list = list(airport_db.values())
65         return airport_list [:limit]
66
67     # POST method (creating data in the data library or db)
68     @app.post( path: '/airports', tags=["Airports"])
69     def create_airport(airport: Airport):
70         airport_id = airport.airport_id
71         check_airport_id_in_db(airport_id) #validation
72         airport_db[airport_id] = airport.model_dump()
73         return {'message': f'Successfully created airport: {airport_id}'}
74
75     # PUT method (updating data in the data library or db)
76     @app.put( path: '/airports', tags=["Airports"])
77     def update_airport(airport: Airport):
78         airport_id = airport.airport_id
79         airport_db[airport_id] = airport.model_dump()
80         return {'message': f'Successfully updated airport: {airport_id}'}
81
82     # PATCH method (updating specific/partial data in the data library or db)
83     @app.patch( path: '/airports', tags=["Airports"])
84     def update_airport_partial(airport: Airport):
85         airport_id = airport.airport_id
86         check_airport_id_not_in_db(airport_id) # validation
87         airport_db[airport_id].update(airport.model_dump(exclude_unset=True))
88         return {'message': f'Successfully updated specific data of the airport: {airport_id}'}
89
90     # DELETE method (deleting data from the data library or db)
91     @app.delete( path: '/airports/{airport_id}', tags=["Airports"])
92     def delete_airport(airport_id: str):
93         check_airport_id_not_in_db(airport_id) # validation
94         del airport_db[airport_id]
95         return {'message': f'Successfully deleted airport: {airport_id}'}

```

The screenshot shows a Windows PowerShell window titled "Terminal Local x +". The command `uvicorn main:app --reload` is run in the directory `C:\Users\njmlo\PycharmProjects\project-a>`. The output shows the application starting up, including logs about watching for changes, running on port 8000, and the server process starting.

```

Terminal Local x +
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\\\Users\\\\njmlo\\\\PycharmProjects\\\\project-a']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [4592] using WatchFiles
INFO:     Started server process [632]
INFO:     Waiting for application startup.
INFO:     Application startup complete.

```

## 19. Here is the code in text (main.py file)

```
# import:
from fastapi import FastAPI, HTTPException, status
from pydantic import BaseModel, Field # (to be able to use further pydantic
validation)

# to be able to use further pydantic validation
from typing import Optional

# command that enables the framework/terminal used to display a user-friendly
UI
app = FastAPI(
    title='Airports',
    description='This is a sampling APIs for Airport'
)
# text values for the Swagger method group name/tags in FastAPI UI display
tags_metadata = [
    {
        "name": "Airports",
        "description": "Airport list. Link on the documentation on the
right.",
        "externalDocs": {
            "description": "Sampling Test automation documentation",
            "url": "https://fastapi.tiangolo.com/",
        },
    },
]
# sample data library
airport_db = {
    'a': {'airport_id': 'aa', 'airport_name': 'To Sky Airport', 'city':
'Flying Lake AK', 'country_state': 'US Alaska'},
    'b': {'airport_id': 'bb', 'airport_name': 'Bear Creek Mining Strip',
'city': 'Granite Mountain CA',
        'country_state': 'US California'},
    'c': {'airport_id': 'cc', 'airport_name': 'Little Squaw Airport', 'city':
'Little Squaw FL',
        'country_state': 'US Florida'}
}

# the class used to create a data into the data library or db
class Airport(BaseModel):
    airport_id: str = Field(min_length=2, max_length=20) # (utilizing further
pydantic validation)
    airport_name: str = Field(min_length=3, max_length=20) # (utilizing
further pydantic validation)
    city: str = Field(min_length=3, max_length=20) # (utilizing further
pydantic validation)
    country_state: Optional[str] = None # (this makes the field not
mandatory)

# functions that raise error response(s) that can be use and reuse in various
methods
```

```

# check if data exist
def check_airport_id_in_db (airport_id: str):
    if airport_id in airport_db:
        raise HTTPException(status_code=status.HTTP_302_FOUND,
                            detail=f'Airport_ID {airport_id} exist in
database.')
# check if data does not exist
def check_airport_id_not_in_db (airport_id: str):
    if airport_id not in airport_db:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND,
                            detail=f'Airport_ID {airport_id} not found.')

# change the Swagger method group name/tags in FastAPI UI display
app.openapi_tags=tags_metadata

#=====Method calls below=====

#GET method (returning DEFAULT LIMITED results)
@app.get('/airports', tags=["Airports"])
def get_airport_query(limit: int = 20):
    # return limited result as a list
    airport_list = list(airport_db.values())
    return airport_list [:limit]

# POST method (creating data in the data library or db)
@app.post('/airports', tags=["Airports"])
def create_airport(airport: Airport):
    airport_id = airport.airport_id
    check_airport_id_in_db(airport_id) #validation
    airport_db[airport_id] = airport.model_dump()
    return {'message': f'Successfully created airport: {airport_id}'}

# PUT method (updating data in the data library or db)
@app.put('/airports', tags=["Airports"])
def update_airport(airport: Airport):
    airport_id = airport.airport_id
    airport_db[airport_id] = airport.model_dump()
    return {'message': f'Successfully updated airport: {airport_id}'}

# PATCH method (updating specific/partial data in the data library or db)
@app.patch('/airports', tags=["Airports"])
def update_airport_partial(airport: Airport):
    airport_id = airport.airport_id
    check_airport_id_not_in_db(airport_id) # validation
    airport_db[airport_id].update(airport.model_dump(exclude_unset=True))
    return {'message': f'Successfully updated specific data of the airport:
{airport_id}'}

# DELETE method (deleting data from the data library or db)
@app.delete('/airports/{airport_id}', tags=["Airports"])
def delete_airport(airport_id: str):
    check_airport_id_not_in_db(airport_id) # validation
    del airport_db[airport_id]
    return {'message': f'Successfully deleted airport: {airport_id}'}
*****END*****

```