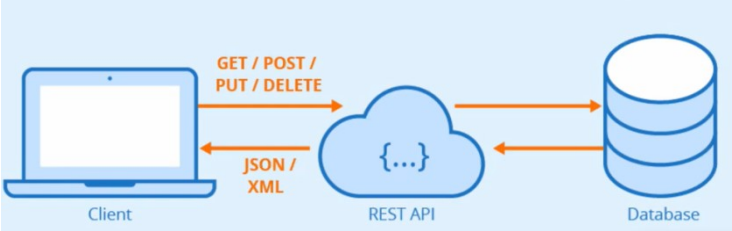
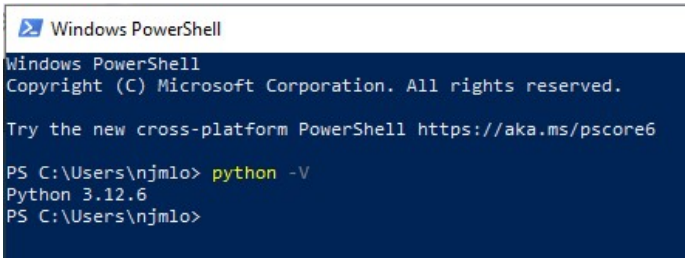
Test Automation using:	**Document highlight (main.py):**
• Python • PyCharm • FastAPI • Pydantic library • GitHub repository	• GET method • POST method • PUT method • PATCH method • DELETE method

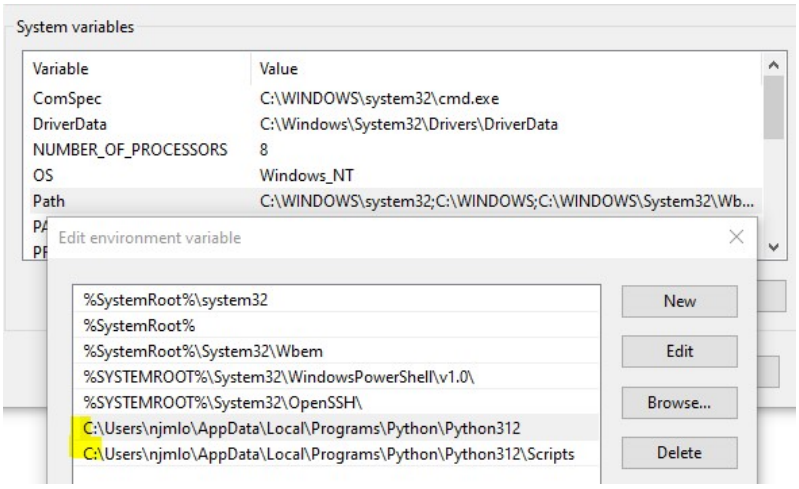
1. API Concept (interface between client and server (database), request and response)



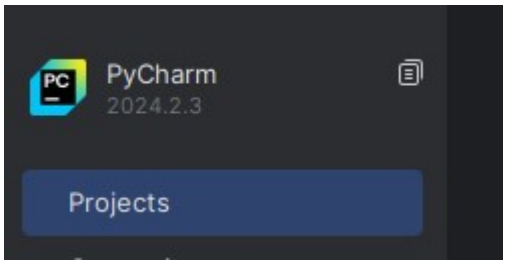
2. Python (version at the time of documentation)



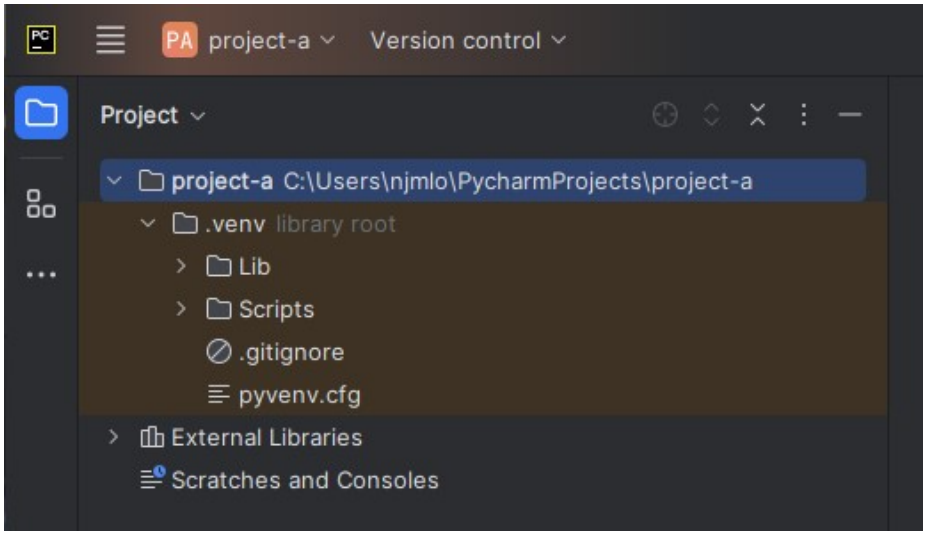
3. Added under System Variable Path



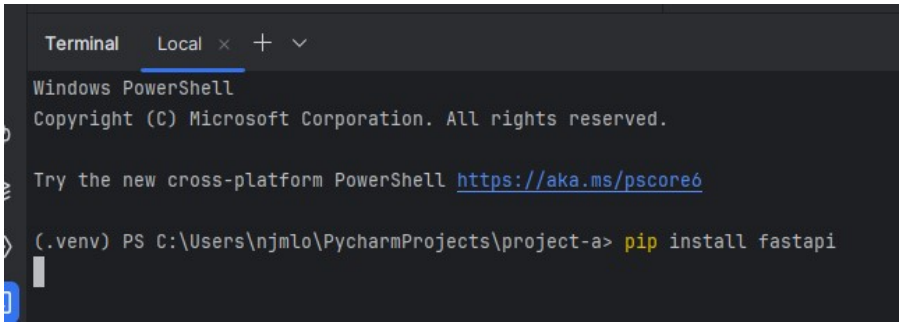
4. PyCharm Community Edition (IDE version at the time of documentation)

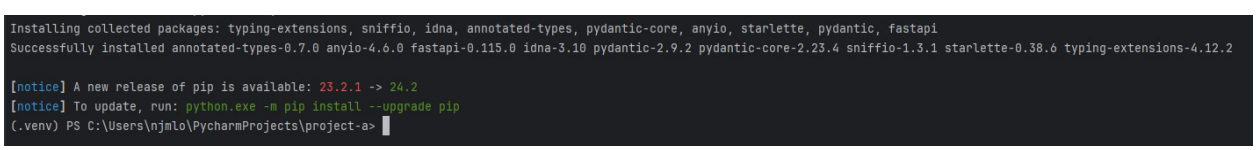


5. Create a New Project



6. Install FastAPI Web Framework



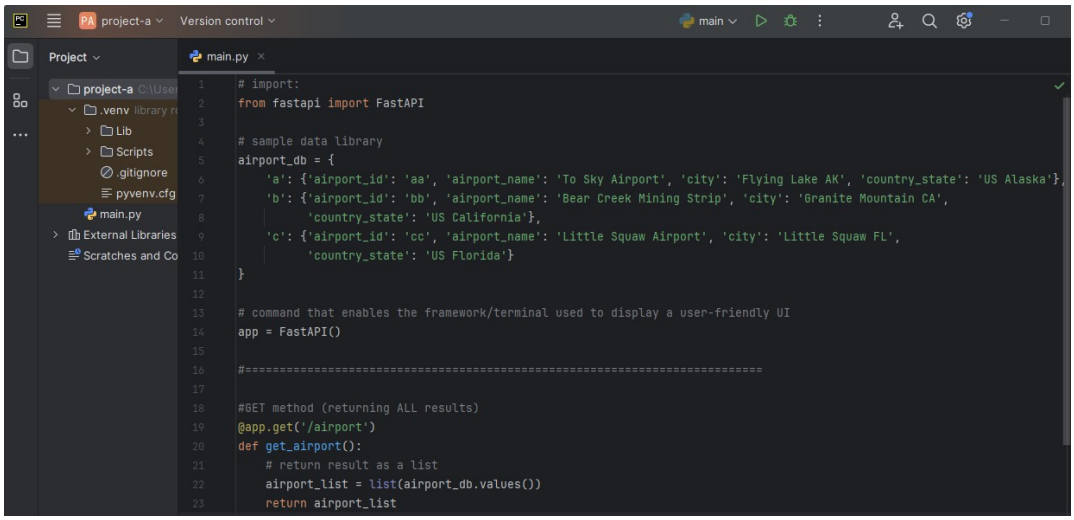


7. Install Uvicorn Web Server

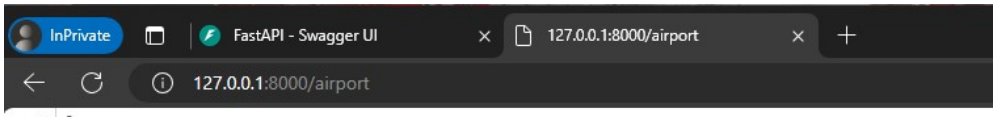
```
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> pip install "uvicorn [standard]"
```

```
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>
```

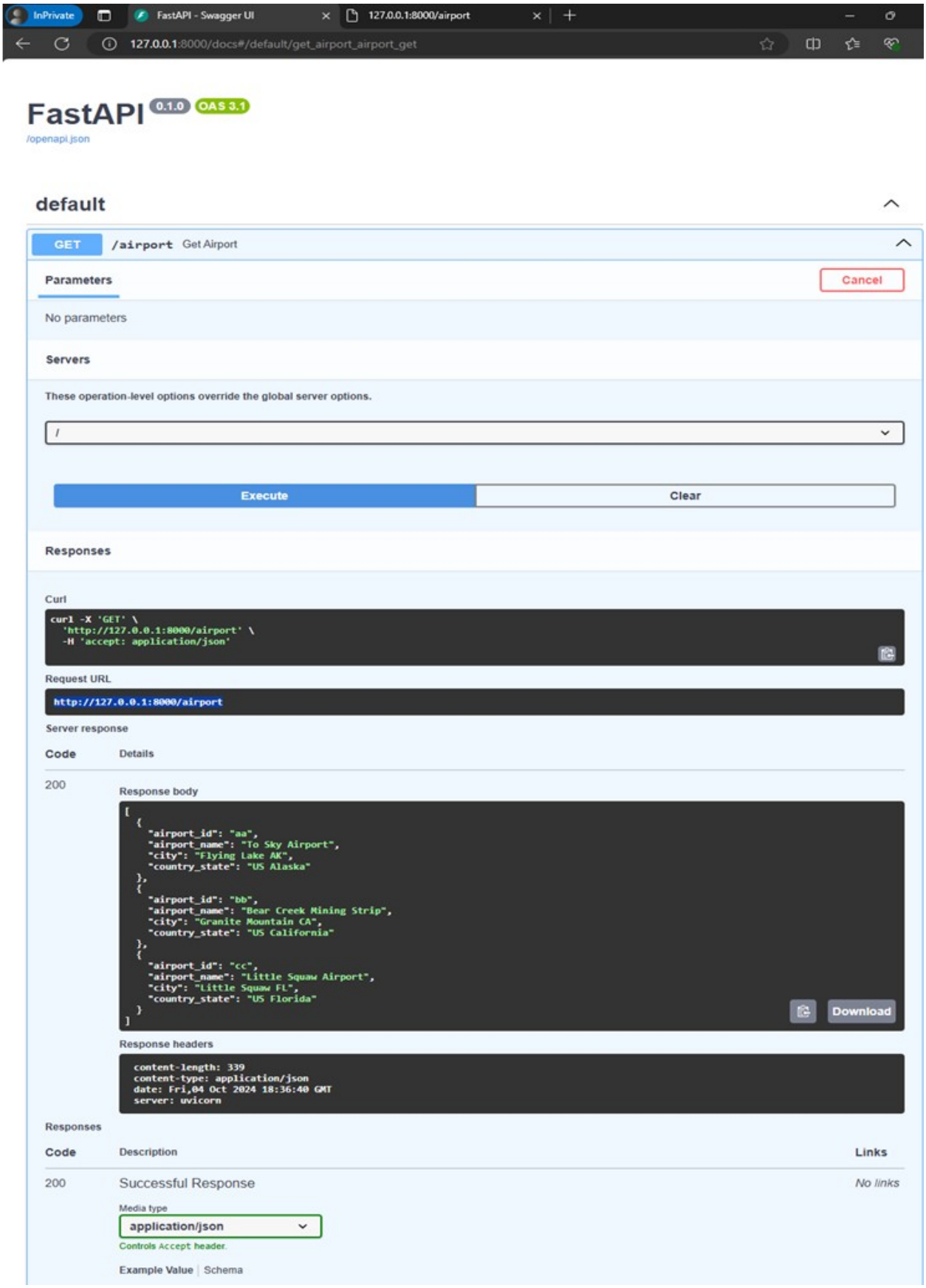
8. Sample GET (code in PyCharm, reload Uvicorn, and display ALL results with FastAPI)



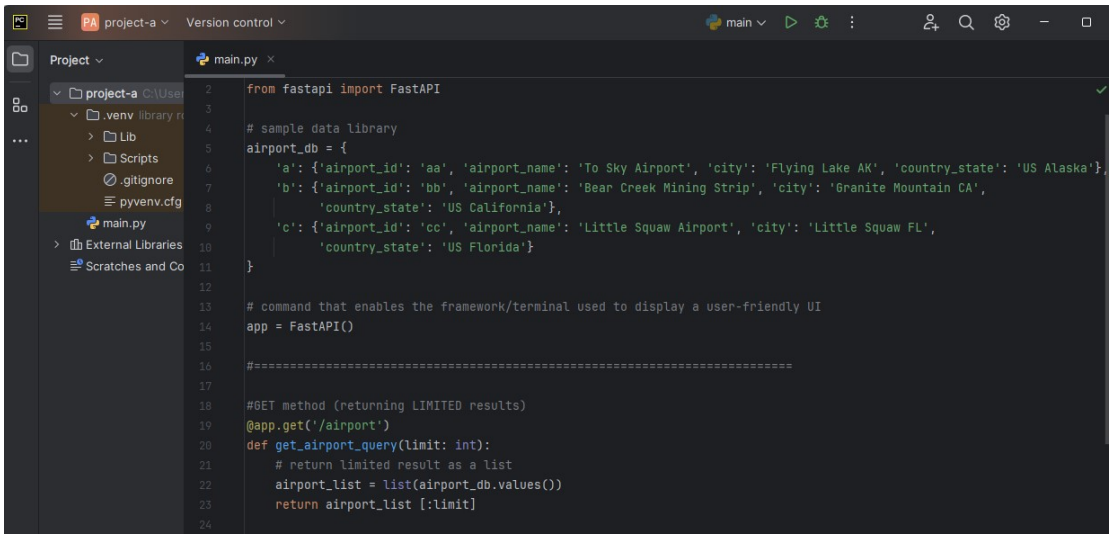

```
(.venv) PS C:\Users\njmlo\PycharmProjects\project-a> uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['C:\\Users\\njmlo\\PycharmProjects\\project-a']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [11212] using WatchFiles
INFO:     Started server process [19656]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```



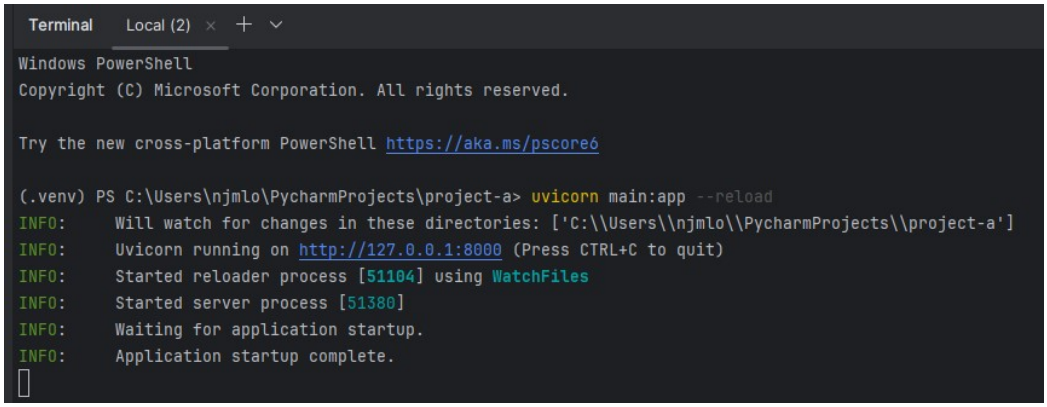

```
[
    {
        "airport_id": "aa",
        "airport_name": "To Sky Airport",
        "city": "Flying Lake AK",
        "country_state": "US Alaska"
    },
    {
        "airport_id": "bb",
        "airport_name": "Bear Creek Mining Strip",
        "city": "Granite Mountain CA",
        "country_state":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


InPrivate
FastAPI - Swagger UI
127.0.0.1:8000/airport
127.0.0.1:8000/docs#/default/get_airport_airport_get
FastAPI
0.1.0
OAS 3.1
/openapi.json
default
GET
/airport
Get Airport
Parameters
Cancel
No parameters
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'
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"
}
]
Download
Response headers
content-length: 339
content-type: application/json
date: Fri,04 Oct 2024 18:36:40 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.
Example Value
Schema

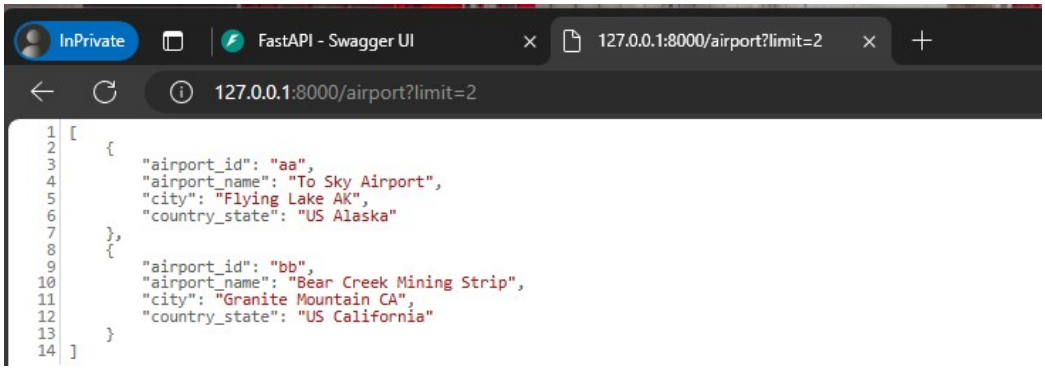
9. Sample GET (code in PyCharm, reload Uvicorn, and display LIMITED results with FastAPI)



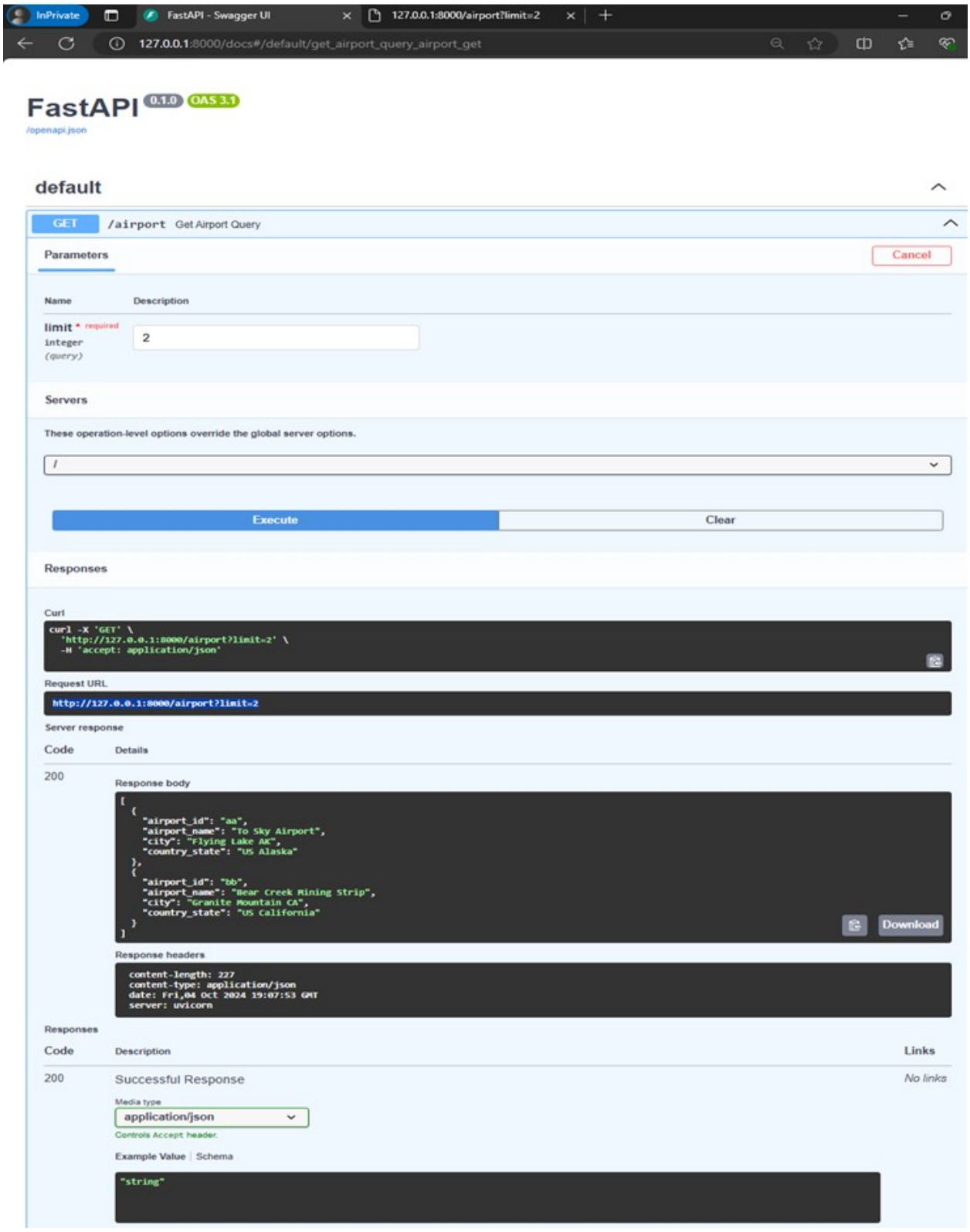
```
from fastapi import FastAPI

# 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]
```



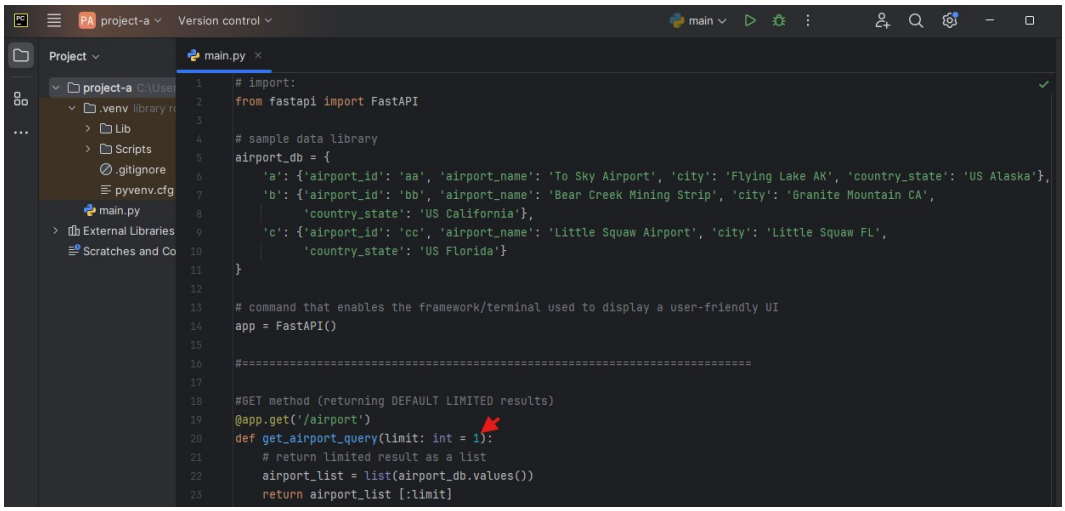
```
Terminal  Local (2)

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.
```

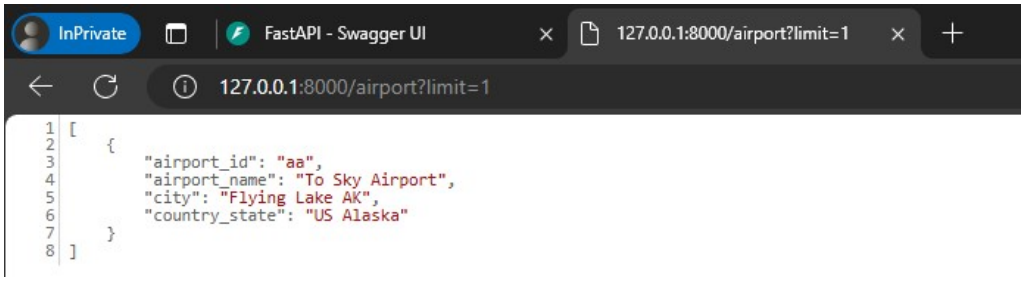


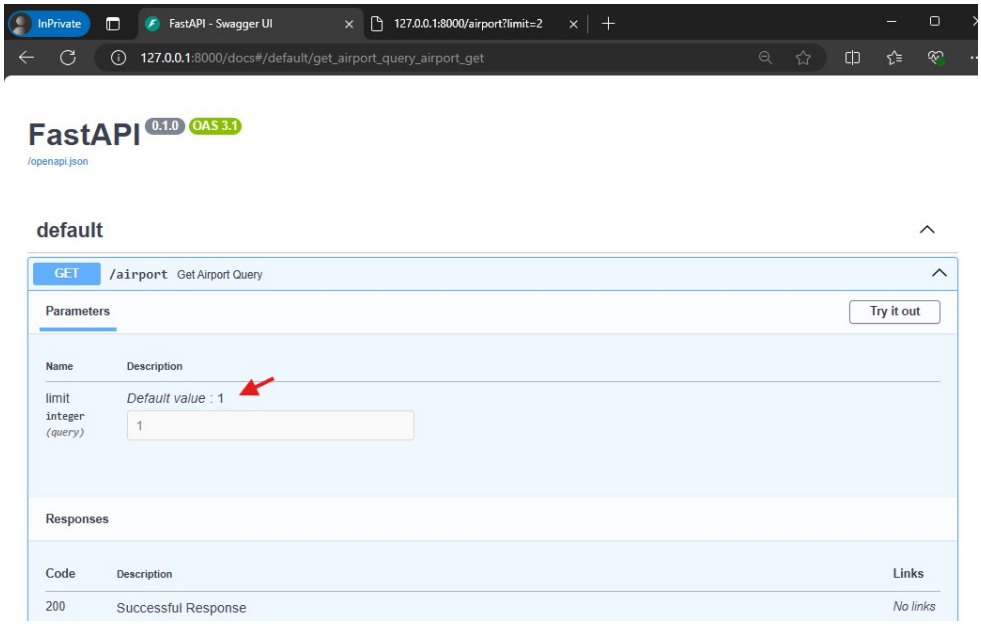
```
InPrivate  FastAPI - Swagger UI  127.0.0.1:8000/airport?limit=2

127.0.0.1:8000/airport?limit=2

[
    {
        "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"
    }
]
```

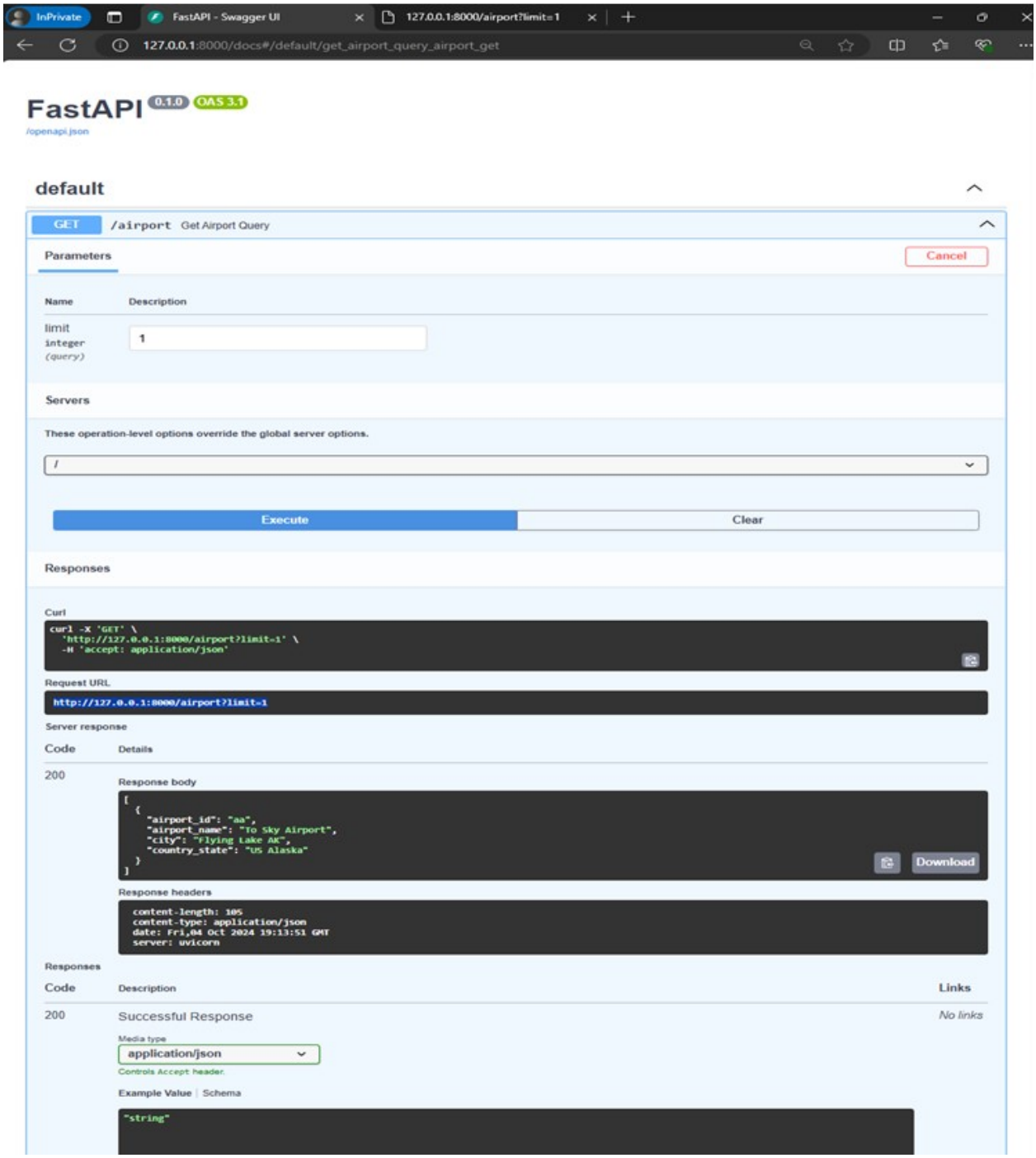

InPrivate
FastAPI - Swagger UI
127.0.0.1:8000/airport?limit=2
127.0.0.1:8000/docs#/default/get_airport_query_airport_get
FastAPI
0.1.0
OAS 3.1
/openapi.json
default
GET
/airport
Get Airport Query
Parameters
Cancel
Name
Description
limit * required
integer
(query)
2
Servers
These operation-level options 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
Response body
[
{
"airport_id": "aa",
"airport_name": "Yo 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"
}
]
Download
Response headers
content-length: 227
content-type: application/json
date: Fri,04 Oct 2024 19:07:53 GMT
server: uvicorn
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)

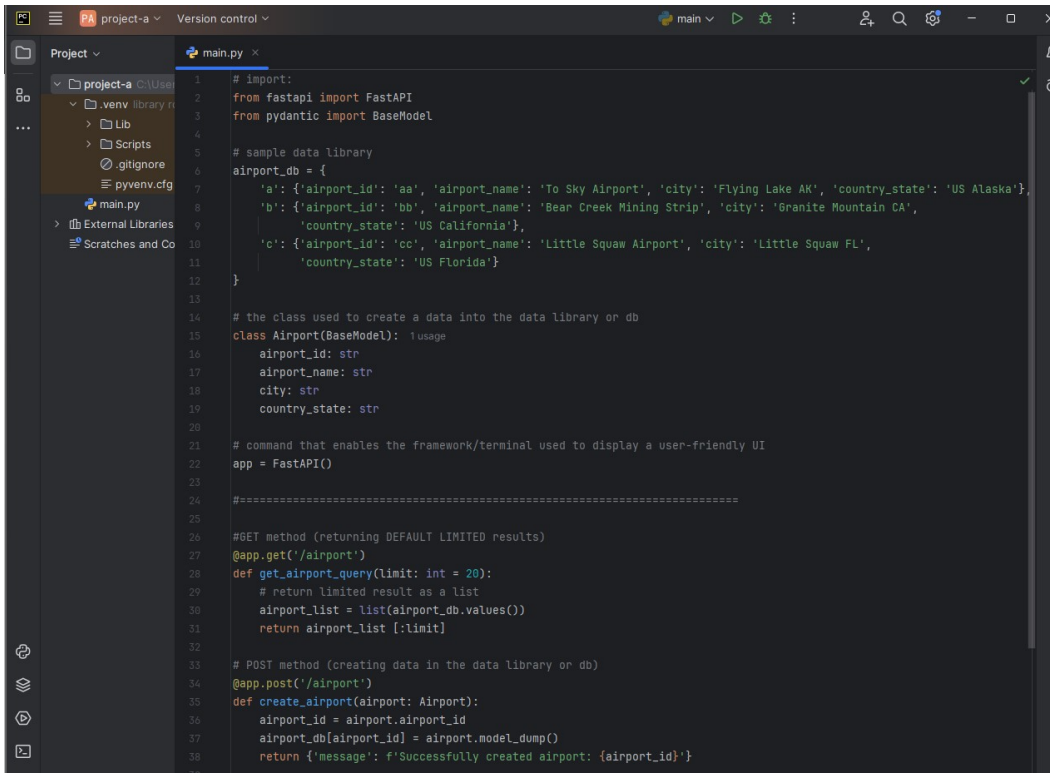




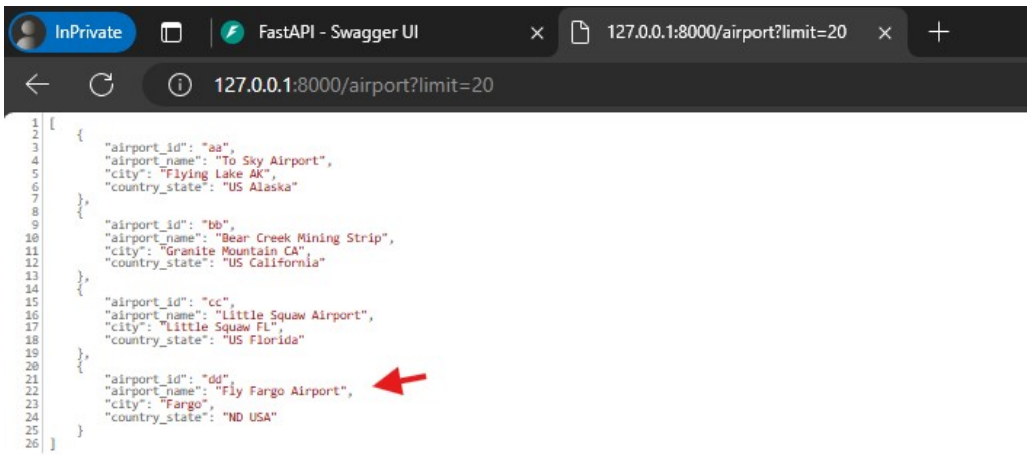



InPrivate
FastAPI - Swagger UI
127.0.0.1:8000/airport?limit=1
127.0.0.1:8000/docs#/default/get_airport_query_airport_get
FastAPI
0.1.0
OAS 3.1
/openapi.json
default
GET
/airport
Get Airport Query
Parameters
Cancel
Name
Description
limit
integer
(query)
1
Servers
These operation-level options override 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 Uvicorn, and display (added data) results with FastAPI (using the GET method))



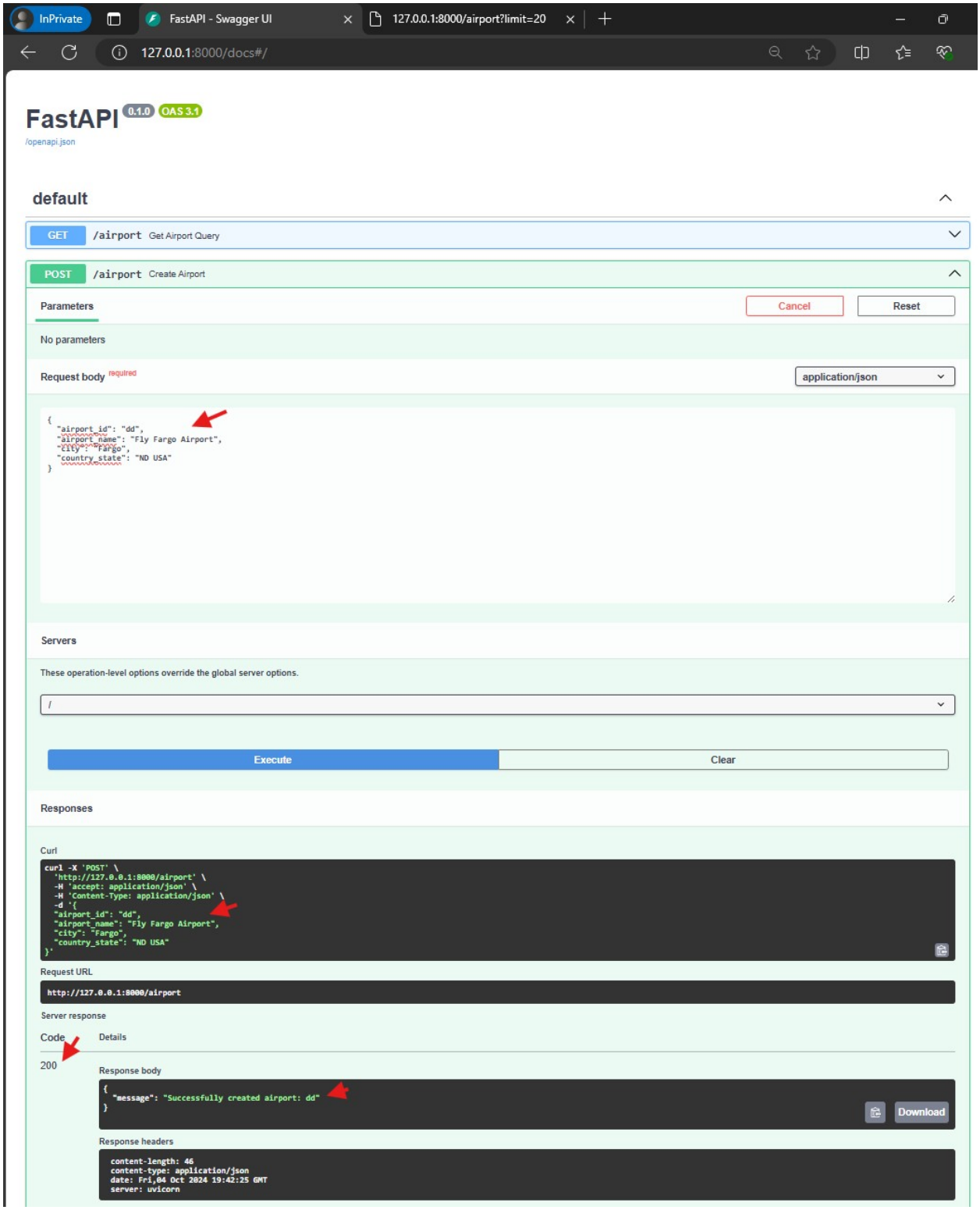
```python
# import:
from fastapi import FastAPI
from pydantic import BaseModel

# 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):  1 usage
    airport_id: str
    airport_name: str
    city: str
    country_state: str

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

#============================================================================

#GET method (returning DEFAULT LIMITED results)
@app.get('/airport')
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('/airport')
def create_airport(airport: Airport):
    airport_id = airport.airport_id
    airport_db[airport_id] = airport.model_dump()
    return {'message': f'Successfully created airport: {airport_id}'}
```



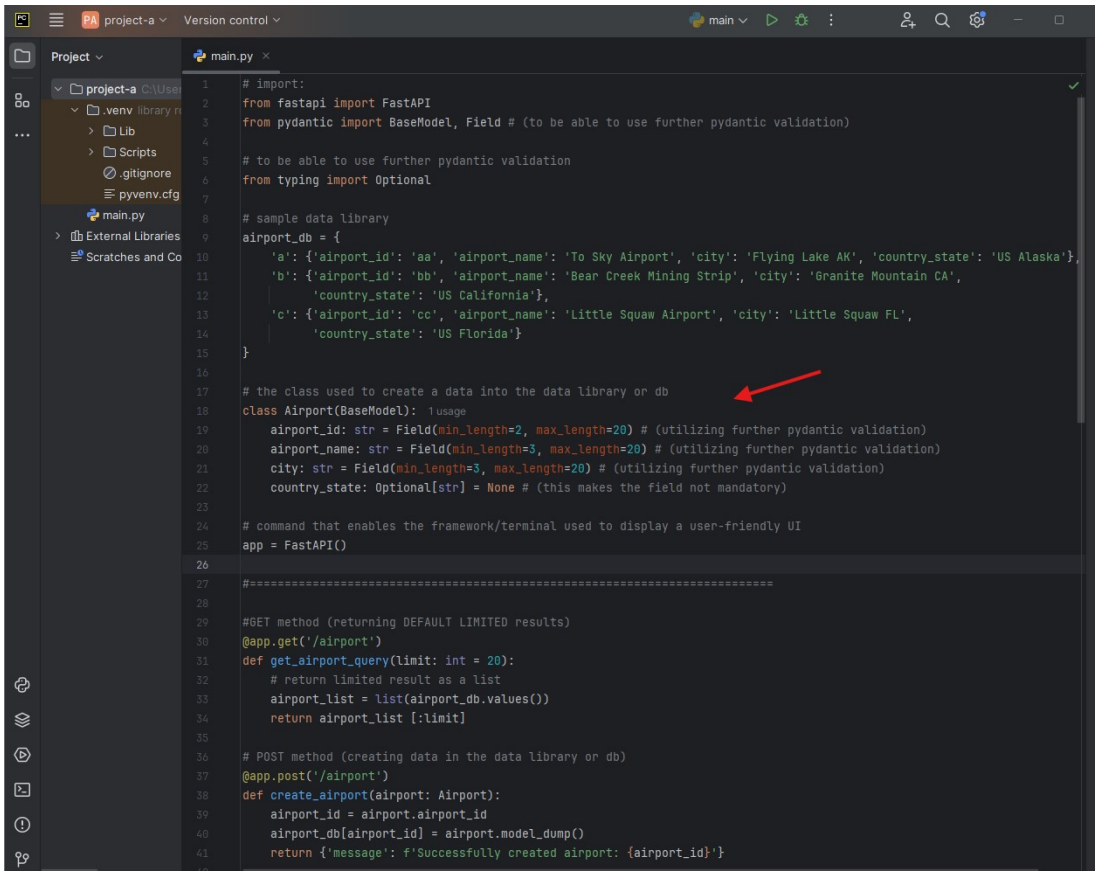
InPrivate | FastAPI - Swagger UI | 127.0.0.1:8000/airport?limit=20

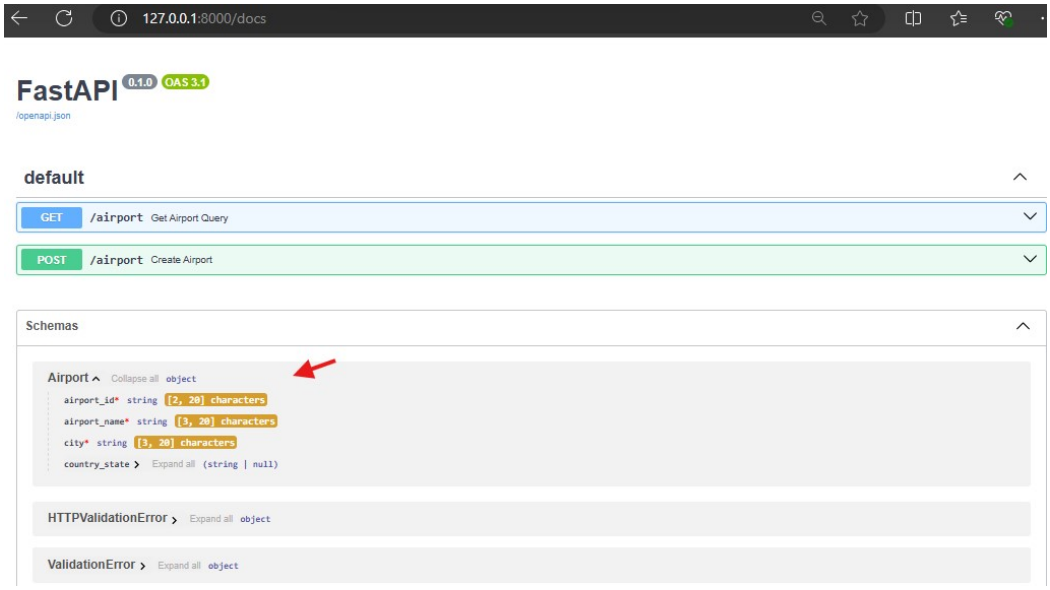
127.0.0.1:8000/airport?limit=20


```
[
    {
        "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"
    }
]
```

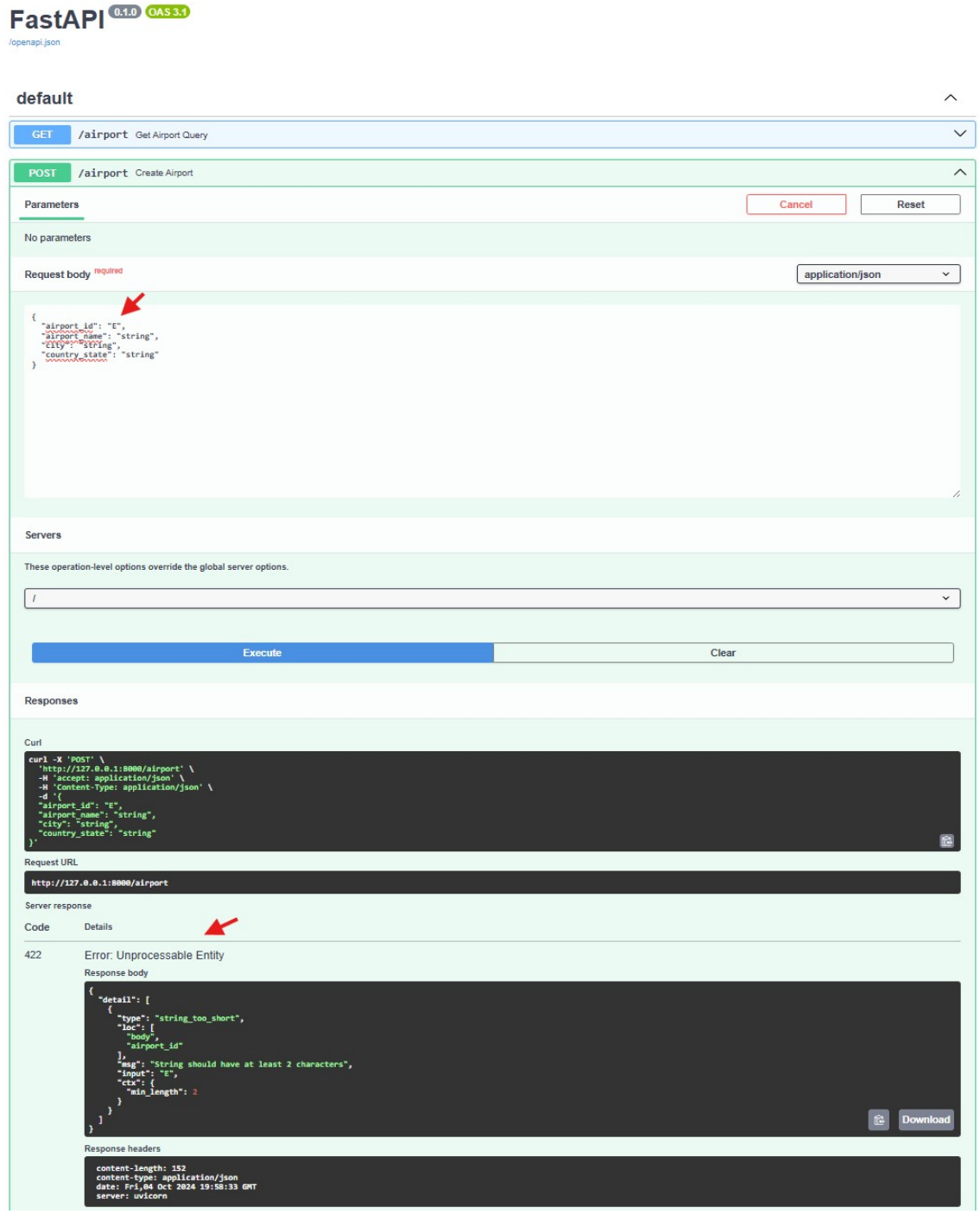

InPrivate
FastAPI - Swagger UI
127.0.0.1:8000/airport?limit=20
127.0.0.1:8000/docs#/
FastAPI
0.1.0
OAS 3.1
/openapi.json
default
GET /airport Get Airport Query
POST /airport Create Airport
Parameters
Cancel
Reset
No parameters
Request body required
application/json
{
"airport_id": "dd",
"airport_name": "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 'accept: application/json' \
-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 19:42:25 GMT
server: uvicorn

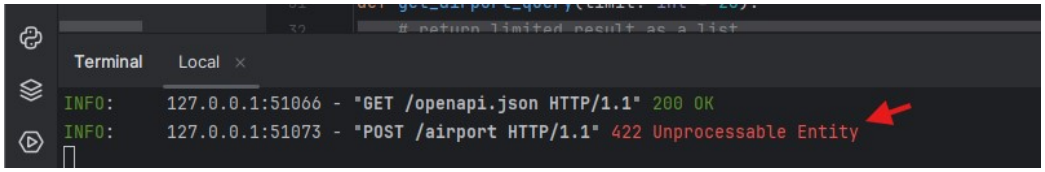
12. Sample POST (code in PyCharm (create data), reload Uvicorn, and display (added data) results with FastAPI (with enhanced pydantic validation added within the class))



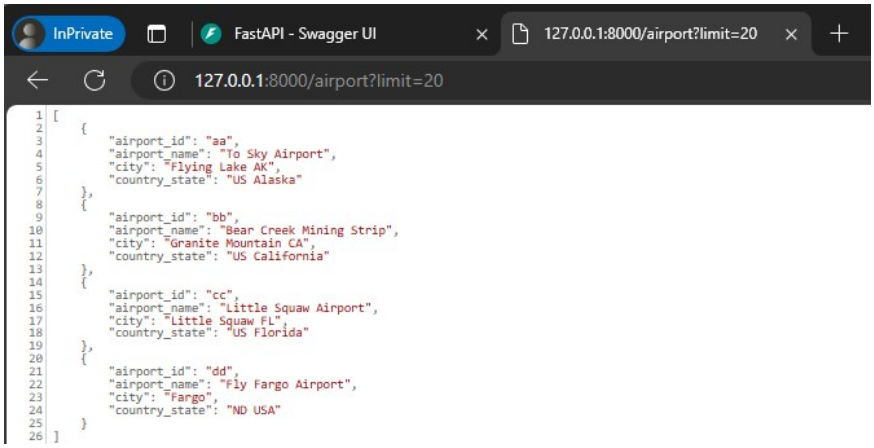


When executed to trigger the error validation

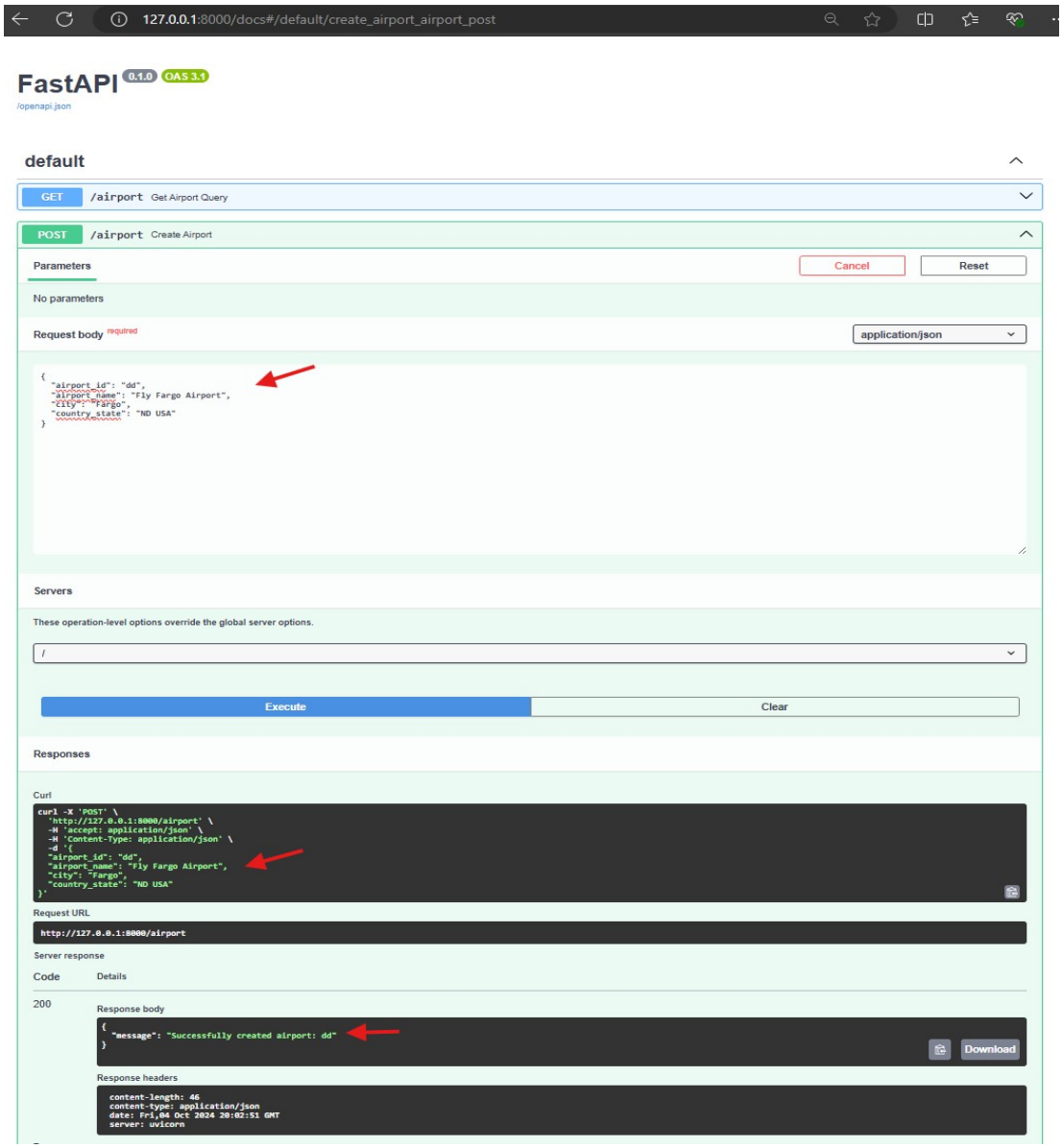




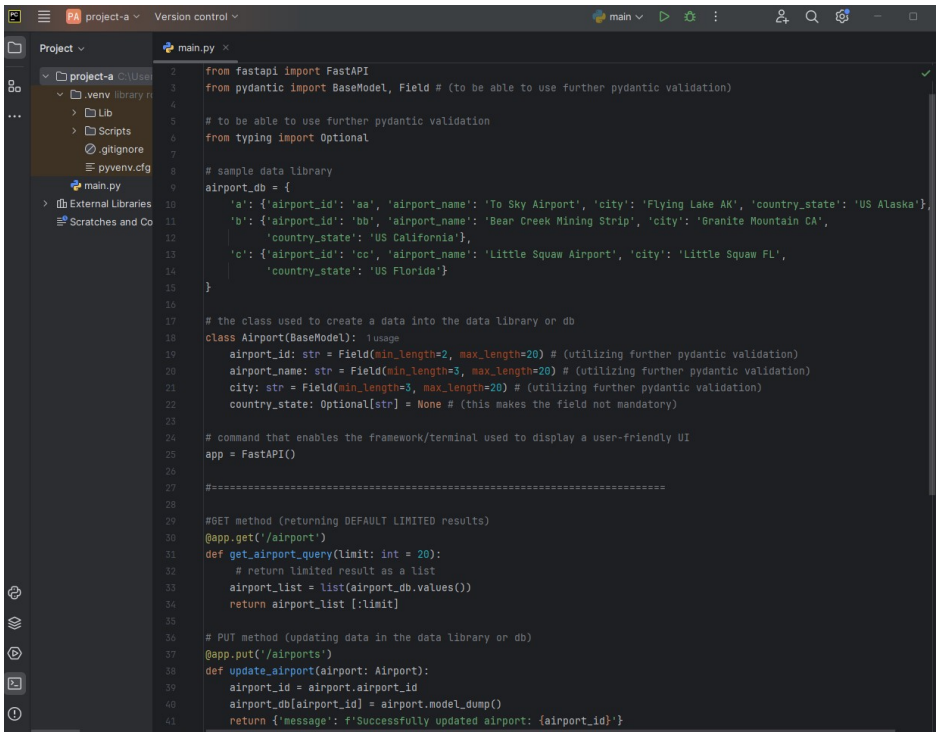
When executed without any error




```
[
    {
        "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"
    }
]
```

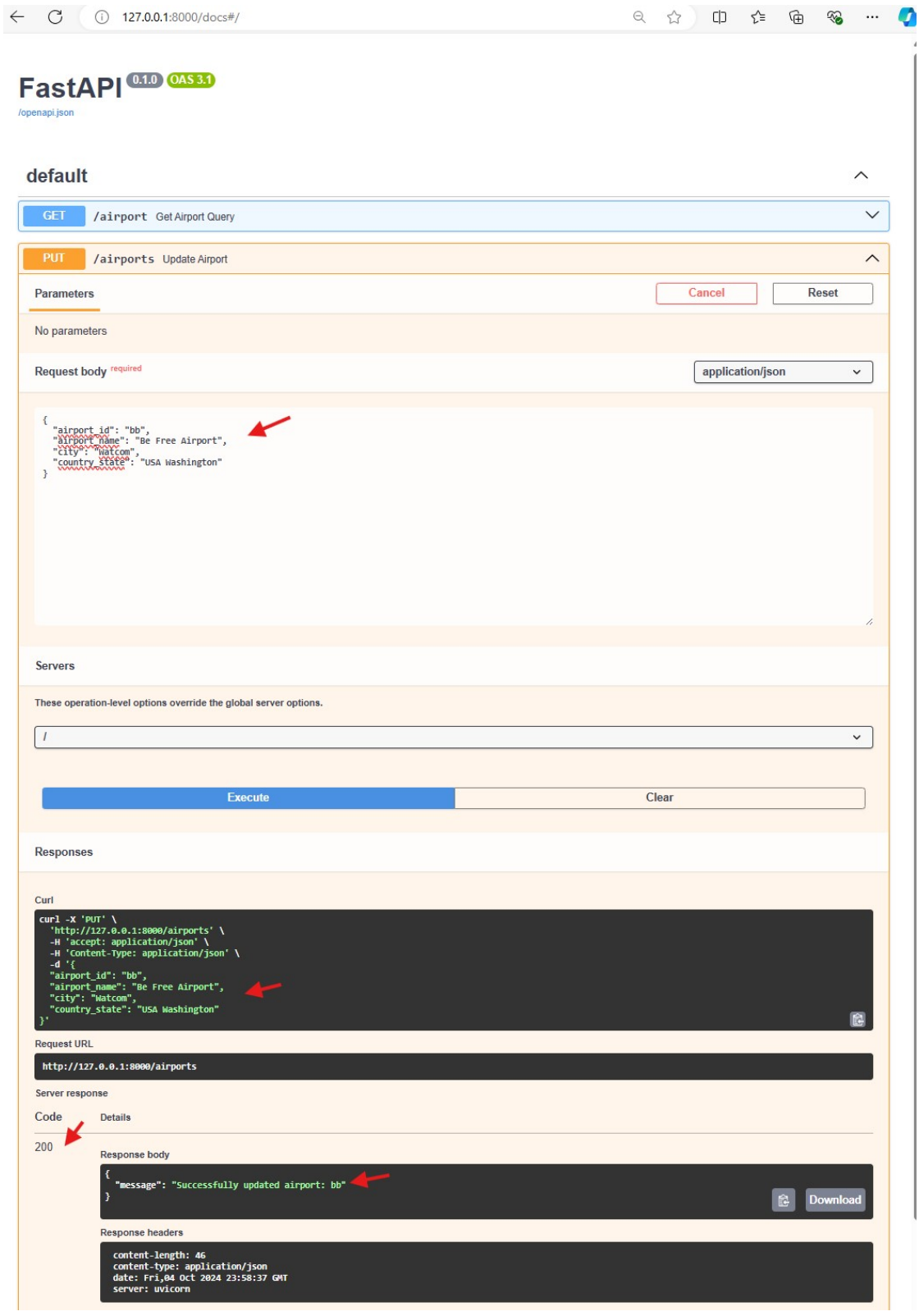



13. Sample PUT (code in PyCharm, reload Uvicorn, and display results (after update) with FastAPI)



```python
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):  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)

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

#=========================================================================

#GET method (returning DEFAULT LIMITED results)
@app.get('/airport')
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}'}
```

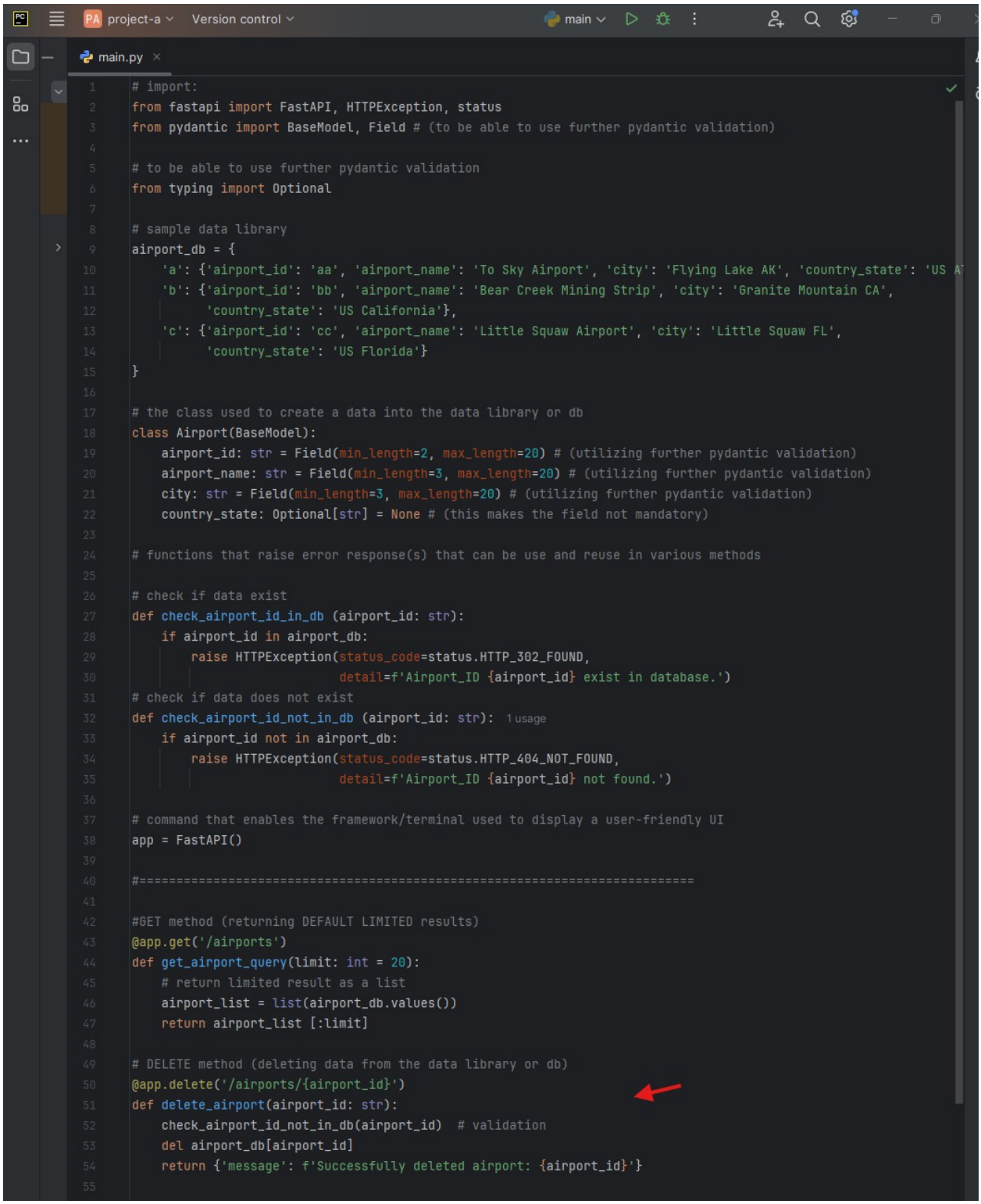
```
    },
    {
        "airport_id": "bb",
        "airport_name": "Be Free Airport",
        "city": "Watcom",
        "country_state": "USA Washington"
    }
]
```


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
Cancel
Reset
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

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

```python
# 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 A
    '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}'}
```

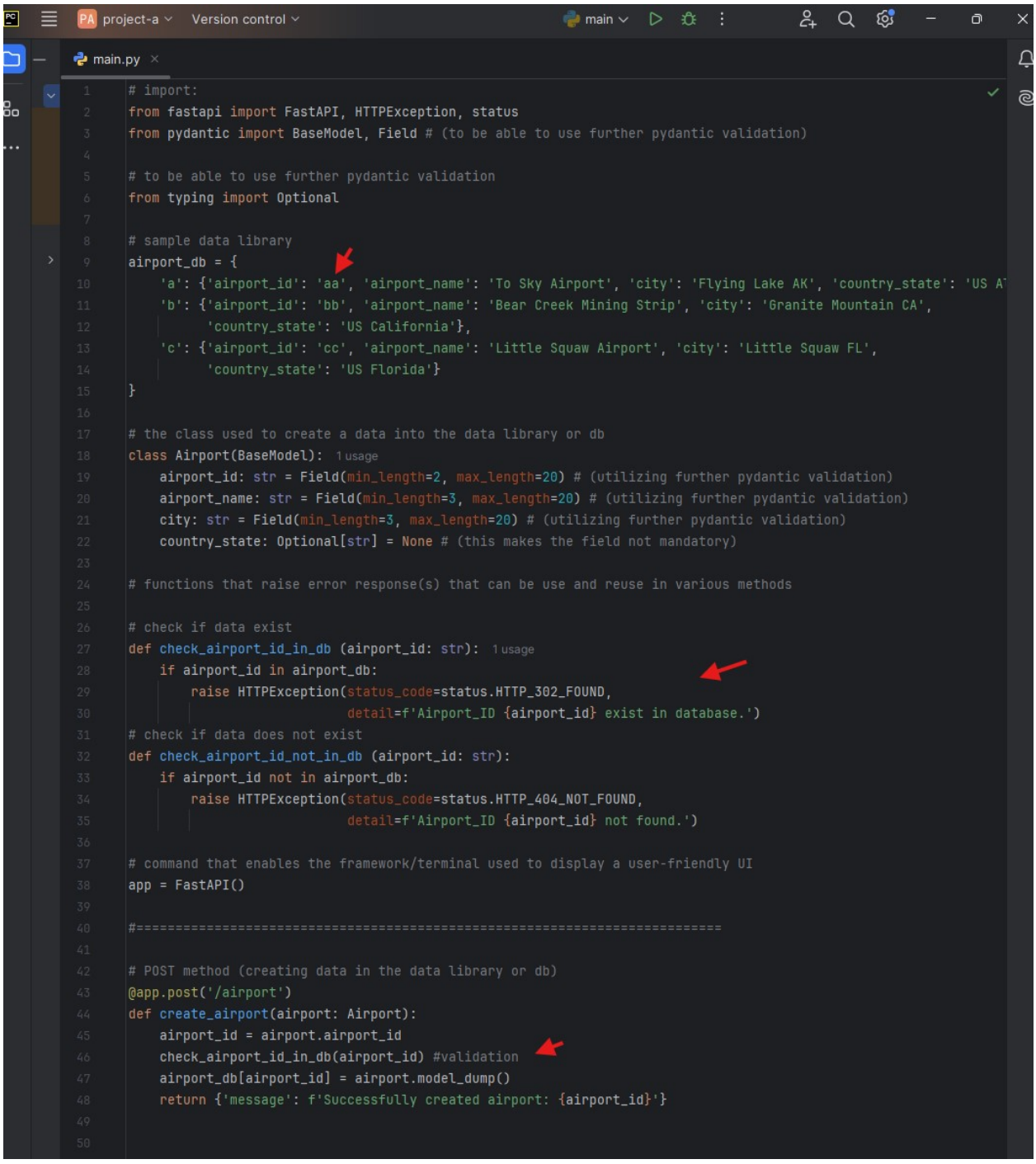
15. Sample DELETE (code in PyCharm), deleting specific data



```python
# 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 A
    '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



```python
# 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 A
    '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):  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):
    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}'}
```

127.0.0.1:8000/docs#/default/create_airport_airport_post

FastAPI 0.1.0 OAS 3.1

/openapi.json

default

POST `/airport` Create Airport

Parameters | Cancel | Reset

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 *Undocumented*	Error: Found

Response body

```
{
  "detail": "Airport_ID aa exist in database."
}
```

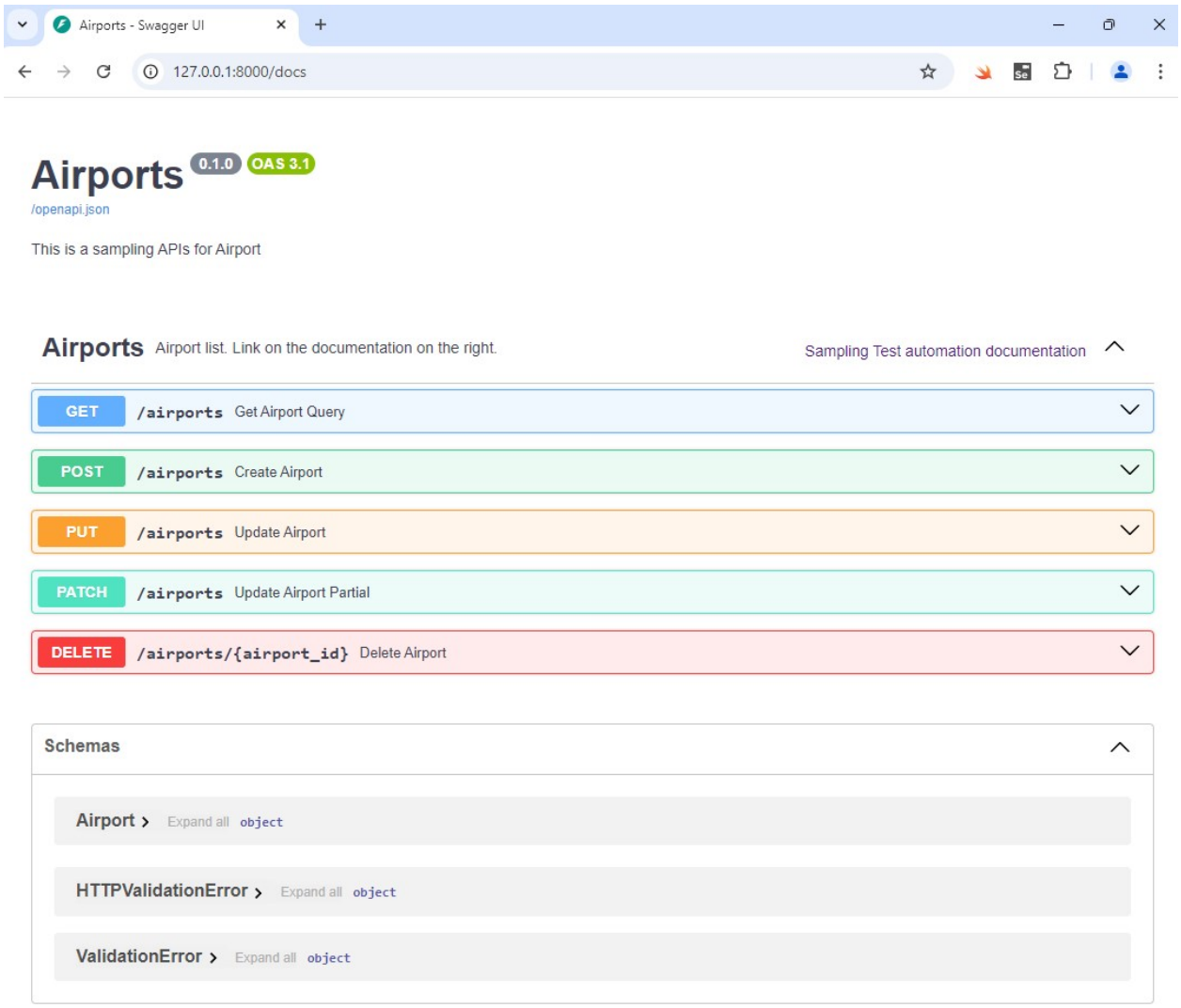
Download

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

```python
# 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(
# change the Swagger page name in FastAPI UI display
    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
    '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

main.py

```python
# 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(
# change the Swagger page name in FastAPI UI display
    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 A
    '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):  3 usages
    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
```

```python

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

#GET method (returning DEFAULT LIMITED results)
@app.get( path: '/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( path: '/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( path: '/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( path: '/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( path: '/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}'}
```

```
Terminal   Local ×  +  ⌄                                                    ⋮  —

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(
# change the Swagger page name in FastAPI UI display
    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
```

```python
# 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**********