
	Politechnika Bydgoska im. J. J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki		
Przedmiot	Skryptowe języki programowania		
Prowadzący	mgr inż. Martyna Tarczewska		
Temat	<i>FastAPI</i>		
Student	Marcin Ogórkiewicz		
Nr ćw.	14	Data wykonania	22.01.2024
Ocena		Data oddania spr.	22.01.2024

Zadania 1 – 4

```

from typing import List, Optional

import uvicorn
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel

app = FastAPI()

class MyItem(BaseModel):
    id: int
    name: str
    description: Optional[str] = None
    price: float

# Przykładowa baza danych
database: List[MyItem] = [
    MyItem(id=1, name="Item1", description="Description1", price=10.0),
    MyItem(id=2, name="Item2", description="Description2", price=20.0),
    MyItem(id=3, name="Item3", description="Description3", price=30.0)
]

# Zadanie 1: Endpointy GET
@app.get("/items", response_model=List[MyItem])
def get_all_items():
    return database

@app.get("/items/{item_id}", response_model=MyItem)
def get_item(item_id: int):

```

```

    item = next((item for item in database if item_id == item.id), None)
    if item is None:
        raise HTTPException(status_code=404, detail="Item not found")
    return item

# Zadanie 2: Endpoint POST
@app.post("/items", response_model=MyItem)
def create_item(item: MyItem):
    database.append(item)
    return item

# Zadanie 3: Endpoint PUT
@app.put("/items/{item_id}", response_model=MyItem)
def update_item(item_id: int, updated_item: MyItem):
    index = next((index for index, item in enumerate(database) if item_id
== item.id), None)
    if index is None:
        raise HTTPException(status_code=404, detail="Item not found")

    database[index] = updated_item
    return updated_item

# Zadanie 4: Endpointy DELETE
@app.delete("/items/{item_id}")
def delete_item(item_id: int):
    database[:] = [item for item in database if item.id != item_id]
    return {"message": "Item deleted"}

@app.delete("/items")
def delete_all_items():
    database.clear()
    return {"message": "All items deleted"}

# Zadanie 5: Automatyczna dokumentacja
# Swagger UI: http://127.0.0.1:8000/docs
# ReDoc: http://127.0.0.1:8000/redoc

if __name__ == '__main__':
    uvicorn.run(app, port=8000, host='127.0.0.1')

```

Zadanie 5

GET

/items

Get All Items

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

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

Request URL

```
http://127.0.0.1:8000/items
```

Server response

Code

Details

200

Response body

```
[
  {
    "name": "Item1",
    "description": "Description1",
    "price": 10
  },
  {
    "name": "Item2",
    "description": "Description2",
    "price": 20
  },
  {
    "name": "Item3",
    "description": "Description3",
    "price": 30
  }
]
```

Download

Response headers

```
content-length: 178
content-type: application/json
date: Mon, 22 Jan 2024 10:16:07 GMT
server: uvicorn
```

POST

/items

Create Item

Parameters

Cancel

Reset

No parameters

Request body

required

application/json

{

"id": 4,

"name": "Item4",

"description": "Description4",

"price": 40.0

}

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/items' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 4,
    "name": "Item4",
    "description": "Description4",
    "price": 40.0
  }'
```

Request URL

http://127.0.0.1:8000/items

Server response

Code

Details

200

Response body

{

"id": 4,

"name": "Item4",

"description": "Description4",

"price": 40

}

Download

Response headers

content-length: 65

content-type: application/json

date: Mon, 22 Jan 2024 18:31:09 GMT

server: uvicorn

GET

/items/{item_id} Get Item

Parameters

Cancel

Name	Description
item_id * required integer (path)	<div>1</div>

Execute

Clear

Responses

Curl

curl -X 'GET' \n'http://127.0.0.1:8000/items/1' \n-H 'accept: application/json'

Request URL

http://127.0.0.1:8000/items/1

Server response

Code	Details
200	<div><div>Response body</div><div>{\n "id": 1,\n "name": "Item",\n "description": "Description1",\n "price": 10\n}</div><div>Download</div></div> <div><div>Response headers</div><div>content-length: 65\ncontent-type: application/json\ndate: Mon, 22 Jan 2024 10:31:54 GMT\nserver: uvicorn</div></div>

PUT

/items/{item_id} Update Item

Parameters

Cancel

Reset

Name	Description
item_id <small>* required</small> integer (path)	<input type="text" value="1"/>

Request body * required

application/json

```
{
  "id": 1,
  "name": "ItemUpdated",
  "description": "DescriptionUpdated",
  "price": 11.0
}
```

Execute

Clear

Responses

Curl

```
curl -X 'PUT' \
  'http://127.0.0.1:8000/items/1' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 1,
    "name": "ItemUpdated",
    "description": "DescriptionUpdated",
    "price": 11.0
  }'
```

Request URL

http://127.0.0.1:8000/items/1

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "id": 1, "name": "ItemUpdated", "description": "DescriptionUpdated", "price": 11 }</pre></div><div>Download</div></div> <div><div>Response headers</div><div><pre>content-length: 79 content-type: application/json date: Mon, 22 Jan 2024 10:33:20 GMT server: uvicorn</pre></div></div>

Responses

DELETE

/items/{item_id} Delete Item

Cancel

Parameters

Name	Description
item_id * required	
integer	1
(path)	

ExecuteClear

Responses

Curl

```
curl -X 'DELETE' \
'http://127.0.0.1:8000/items/1' \
-H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/items/1
```

Server response

Code	Details
200	<div><div>Response body</div><pre>{ "message": "Item deleted" }</pre><div>Download</div></div> <div><div>Response headers</div><pre>content-length: 26 content-type: application/json date: Mon, 22 Jan 2024 10:34:03 GMT server: uvicorn</pre></div>

Responses

Code	Description	Links
------	-------------	-------

DELETE

/items Delete All Items

Cancel

Parameters

No parameters

ExecuteClear

Responses

Curl

```
curl -X 'DELETE' \
'http://127.0.0.1:8000/items' \
-H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/items
```

Server response

Code	Details
200	<div><div>Response body</div><pre>{ "message": "All items deleted" }</pre><div>Download</div></div> <div><div>Response headers</div><pre>content-length: 31 content-type: application/json date: Mon, 22 Jan 2024 10:34:44 GMT server: uvicorn</pre></div>

Responses

Code	Description	Links
------	-------------	-------

Zadanie 6

Artykuł wyjaśnia, że stosowanie funkcji async w FastAPI pozwala na obsługę wielu równoległych żądań bez konieczności blokowania wątków. To umożliwia serwerowi obsługę większej liczby klientów jednocześnie, co z kolei zwiększa wydajność i skalowalność aplikacji. Async/await w Pythonie pozwalają

na wydajne zarządzanie współbieżnością, co jest szczególnie przydatne w aplikacjach sieciowych, takich jak serwisy webowe obsługiwane przez FastAPI.

Wnioski

Ćwiczenie pozwoliło mi na poznanie nowej dla mnie technologii, jaką jest framework FastAPI.