
	Politechnika Bydgoska im. J. J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki <b>Zakład Systemów Teleinformatycznych</b>		
<b>Przedmiot</b>	Skryptowe języki programowania		
<b>Prowadzący</b>	mgr inż. Martyna Tarczewska		
<b>Temat</b>	FastAPI		
<b>Student</b>	Paweł Jońca		
<b>Nr lab.</b>	9, 3/4	<b>Data wykonania</b>	13.01.2025r
<b>Ocena</b>		<b>Data oddania spr.</b>	13.01.2025r

```

from typing import List, Union
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import uvicorn

# /
app = FastAPI()

# Definicja klasy Mountains
class Mountains(BaseModel): 5 usages new *
    name: str
    height: float # Wysokość w metrach
    location: str # Lokalizacja
    mountain_id: int | None = None

# Prowizoryczna "baza danych" dla obiektów Mountains
database: List[Mountains] = []

```

```

app = FastAPI()

# Definicja klasy Mountains
class Mountains(BaseModel):
    name: str
    height: float # Wysokość w metrach
    location: str # Lokalizacja
    mountain_id: int | None = None

# Prowizoryczna "baza danych" dla obiektów Mountains
database: List[Mountains] = []

```

## Zad 1

```
# Endpoint: Zwraca wszystkie elementy z bazy danych
@get("/mountains")
@app.get("/mountains") new *
def get_all_mountains() -> List[Mountains]:
    return database

# Endpoint: Zwraca element o wybranym ID
@get("/mountains/{mountain_id}")
@app.get("/mountains/{mountain_id}") new *
def get_mountain(mountain_id: int) -> Mountains:
    mountains = [x for x in database if x.mountain_id == mountain_id]
    if not mountains:
        raise HTTPException(status_code=404, detail="Mountain not found")
    return mountains[0]
```

```
# Endpoint: Zwraca wszystkie elementy z bazy danych
@app.get("/mountains")
def get_all_mountains() -> List[Mountains]:
    return database

# Endpoint: Zwraca element o wybranym ID
@app.get("/mountains/{mountain_id}")
def get_mountain(mountain_id: int) -> Mountains:
    mountains = [x for x in database if x.mountain_id == mountain_id]
    if not mountains:
        raise HTTPException(status_code=404, detail="Mountain not found")
    return mountains[0]
```

## Zad 2

```
# Dodaje element do bazy danych
@post("/mountains/{mountain_id}")
@app.post("/mountains/{mountain_id}") new *
def add_mountain(mountain_id: int, mountain: Mountains) -> Mountains:
    mountain.mountain_id = mountain_id
    database.append(mountain)
    return mountain
```

```
# Dodaje element do bazy danych
@app.post("/mountains/{mountain_id}")
def add_mountain(mountain_id: int, mountain: Mountains) -> Mountains:
    mountain.mountain_id = mountain_id
    database.append(mountain)
    return mountain
```

### Zad 3

```
# Endpoint: Aktualizuje element o podanym ID
@router.put("/mountains/{mountain_id}")
@app.put("/mountains/{mountain_id}") new *
def update_mountain(mountain_id: int, mountain: Mountains) -> Mountains:
    # Szukaj góry w bazie danych
    for i, existing_mountain in enumerate(database):
        if existing_mountain.mountain_id == mountain_id:
            # Zaktualizuj dane góry
            updated_mountain = mountain.copy(update={"mountain_id": mountain_id})
            database[i] = updated_mountain
            return updated_mountain
    # Jeśli góra nie istnieje, zwróć błąd 404
    raise HTTPException(status_code=404, detail="Mountain not found")
```

```
# Endpoint: Aktualizuje element o podanym ID
@app.put("/mountains/{mountain_id}")
def update_mountain(mountain_id: int, mountain: Mountains) -> Mountains:
    # Szukaj góry w bazie danych
    for i, existing_mountain in enumerate(database):
        if existing_mountain.mountain_id == mountain_id:
            # Zaktualizuj dane góry
            updated_mountain = mountain.copy(update={"mountain_id": mountain_id})
            database[i] = updated_mountain
            return updated_mountain
    # Jeśli góra nie istnieje, zwróć błąd 404
    raise HTTPException(status_code=404, detail="Mountain not found")
```

### Zad 4

```
# Endpoint: Usuwa element o podanym ID
@router.delete("/mountains/{mountain_id}")
@app.delete("/mountains/{mountain_id}") new *
def delete_mountain(mountain_id: int) -> dict:
    for i, existing_mountain in enumerate(database):
        if existing_mountain.mountain_id == mountain_id:
            deleted_mountain = database.pop(i)
            return {"message": "Mountain deleted successfully", "deleted_mountain": deleted_mountain}
    raise HTTPException(status_code=404, detail="Mountain not found")
```

```
# Endpoint: Usuwa element o podanym ID
@app.delete("/mountains/{mountain_id}")
def delete_mountain(mountain_id: int) -> dict:
    for i, existing_mountain in enumerate(database):
        if existing_mountain.mountain_id == mountain_id:
            deleted_mountain = database.pop(i)
            return {"message": "Mountain deleted successfully", "deleted_mountain": deleted_mountain}
    raise HTTPException(status_code=404, detail="Mountain not found")
```

## Zad 5

Podgląd na wszystkie dodane endpointy w Swagger UI.

POST /mountains Add Mountain

Parameters

No parameters

Request body required

application/json

```
{  "name": "Rysy",  "height": 2501,  "location": "Poland",  "mountain_id": 1}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \  'http://127.0.0.1:8080/mountains' \  -H 'accept: application/json' \  -H 'Content-Type: application/json' \  -d '{  "name": "Rysy",  "height": 2501,  "location": "Poland",  "mountain_id": 1}'
```

Request URL

http://127.0.0.1:8080/mountains

Server response

Code	Details
200	<div>Response body</div> <div><pre>{  "name": "Rysy",  "height": 2501,  "location": "Poland",  "mountain_id": 1}</pre></div> <div>Response headers</div> <div><pre>content-length: 67  content-type: application/json  date: Mon, 13 Jan 2025 20:02:13 GMT  server: uvicorn</pre></div>

Responses

Code	Description	Links
200	Successful Response	No links

Code	Description	Links
200	<div>Successful Response</div> <div>Media type</div> <div>application/json</div> <div>Controls Accept header</div> <div>Example Value   Schema</div> <div><pre>{  "name": "string",  "height": 0,  "location": "string",  "mountain_id": 0}</pre></div>	No links
422	<div>Validation Error</div> <div>Media type</div> <div>application/json</div> <div>Example Value   Schema</div> <div><pre>{  "detail": [    {      "loc": [        "string",        0      ],      "msg": "string",      "type": "string"    }  ]}</pre></div>	No links

Curl

```
curl -X 'POST' \
'http://127.0.0.1:8000/mountains' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
  "name": "Mount Everest",
  "height": 8849,
  "location": "Nepal",
  "mountain_id": 2
}'
```

Request URL

http://127.0.0.1:8000/mountains

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{   "name": "Mount Everest",   "height": 8849,   "location": "Nepal",   "mountain_id": 2 }</pre></div><div><div>Download</div></div></div> <div><div>Response headers</div><div><pre>content-length: 75 content-type: application/json date: Mon, 13 Jan 2025 20:03:30 GMT server: uvicorn</pre></div></div>

Responses

Code	Description	Links
200	Successful Response	No links

GET

/mountains

Get All Mountains

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

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

Request URL

http://127.0.0.1:8000/mountains

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{   {     "name": "Rysy",     "height": 2563,     "location": "Poland",     "mountain_id": 1   },   {     "name": "Mount Everest",     "height": 8849,     "location": "Nepal",     "mountain_id": 2   } }</pre></div><div><div>Download</div></div></div> <div><div>Response headers</div><div><pre>content-length: 145 content-type: application/json date: Mon, 13 Jan 2025 20:04:05 GMT server: uvicorn</pre></div></div>

Responses

Code	Description	Links
200	Successful Response	No links

Media type

application/json

Controls Accept header.

GET

/mountains/{mountain\_id} Get Mountain

⌵

Parameters

Cancel

Name	Description
mountain_id * required	
integer	
(path)	

Execute

Clear

Responses

Curl

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

Request URL

```
http://127.0.0.1:8000/mountains/1
```

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{   "name": "Bysy",   "height": 2500,   "location": "Poland",   "mountain_id": 1 }</pre></div><div><div>Download</div></div></div> <div><div>Response headers</div><div><pre>content-length: 67 content-type: application/json date: Mon, 13 Jan 2025 20:04:31 GMT server: uvicorn</pre></div></div>

Responses

Code	Description	Links
200	Successful Response	No links

Media type

PUT

/mountains/{mountain\_id} Update Mountain

⌵

Parameters

Cancel

Reset

Name	Description
mountain_id <span>required</span>	
integer	
(path)	1

Request body required

application/json

```
{
  "name": "Rysy UPDATED",
  "height": 2499,
  "location": "Poland",
  "mountain_id": 1
}
```

Execute

Clear

Responses

Curl

```
curl -X 'PUT' \
  'http://127.0.0.1:8000/mountains/1' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Rysy UPDATED",
    "height": 2499,
    "location": "Poland",
    "mountain_id": 1
  }'
```

Request URL

http://127.0.0.1:8000/mountains/1

Server response

Code	Details
200	<div><div>Response body</div><pre>{   "name": "Rysy UPDATED",   "height": 2499,   "location": "Poland",   "mountain_id": 1 }</pre><div><div>Download</div></div></div> <div><div>Response headers</div></div>

DELETE

/mountains Delete Mountain

Parameters

Cancel

Name	Description
mountain_id <span>*</span> required	
integer (query)	<input type="text" value="1"/>

Execute

Clear

Responses

Curl

```
curl -X 'DELETE' \
'http://127.0.0.1:8080/mountains?mountain_id=1' \
-H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8080/mountains?mountain_id=1
```

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{   "message": "Mountain deleted successfully",   "deleted_mountain": {     "name": "Bysy UPDATED",     "height": 2499,     "location": "Poland",     "mountain_id": 1   } }</pre></div><div>Download</div></div> <div><div>Response headers</div><div><pre>content-length: 338 content-type: application/json date: Mon, 13 Jan 2025 20:06:03 GMT server: uvicorn</pre></div></div>

Responses

Code	Description	Links
200	<div>Successful Response</div> <div>Media type application/json</div> <div>Controls Accept header.</div> <div>Example Value   Schema</div> <div><pre>{}</pre></div>	No links
422	<div>Validation Error</div> <div>Media type application/json</div>	No links



The screenshot shows a REST client interface with the following sections:

- GET /mountains** Get All Mountains
- Parameters**: No parameters. A **Cancel** button is visible.
- Execute** and **Clear** buttons.
- Responses**:
  - Curl**:

```
curl -X 'GET' \
'http://127.0.0.1:8000/mountains' \
-H 'accept: application/json'
```
  - Request URL**: `http://127.0.0.1:8000/mountains`
  - Server response**:

Code	Details
200	<p><b>Response body</b></p> <pre>{   "name": "Mount Everest",   "height": 8849,   "location": "Nepal",   "mountain_id": 2 }</pre> <p><b>Response headers</b></p> <pre>content-length: 77 content-type: application/json date: Mon, 13 Jan 2025 20:06:42 GMT server: uvicorn</pre>
- Responses** table:

Code	Description	Links
200	Successful Response	No links

Media type: **application/json**

Controls Accept header: **Example Value** | **Schema**

```
{
  "name": "string",
  "height": 0,
  "location": "string",
  "mountain_id": 0
}
```

Zad 6.

Stosowanie funkcji asynchronicznych w FastAPI jest dobrą praktyką

- Zmniejsza zużycie zasobów.
- Poprawia skalowalność i wydajność.
- Umożliwia bardziej responsywne działanie aplikacji.
- Wspiera nowoczesne podejście do programowania backendowego.

Dzięki tym zaletom, funkcje async są szczególnie zalecane w aplikacjach wymagających obsługi dużej liczby zapytań lub operacji I/O.

## Wnioski

W trakcie ćwiczenia utworzyłem obiekt przy użyciu biblioteki Pydantic, co ułatwiło zarządzanie danymi w aplikacji zbudowanej w FastAPI. Dodałem różne endpointy (GET, POST, PUT, DELETE), co pozwoliło na realizację operacji CRUD i efektywną pracę z listą danych. Dzięki zastosowaniu funkcji asynchronicznych aplikacja działa wydajniej, szczególnie podczas obsługi wielu zapytań jednocześnie.

## \*Zadanie dodatkowe\*

```
from typing import Optional
from pydantic import BaseModel, Field

class Item(BaseModel):
    """usage: new"""
    name: str = Field(..., title="Name of the item", description="The name of the item being created", max_length=50)
    description: Optional[str] = Field(None, title="Description of the item", description="A detailed description of the item", max_length=300)
    price: float = Field(..., title="Price of the item", description="The price must be a positive value", gt=0)
    tax: Optional[float] = Field(None, title="Tax applied to the item", description="Optional tax percentage for the item", ge=0)

# Przykładowe użycie w FastAPI
from fastapi import FastAPI

app = FastAPI()

@app.post("/items/")
async def create_item(item: Item):
    return item
```

```
from typing import Optional
from pydantic import BaseModel, Field

class Item(BaseModel):
    name: str = Field(..., title="Name of the item", description="The name of the item being created", max_length=50)
    description: Optional[str] = Field(None, title="Description of the item", description="A detailed description of the item", max_length=300)
    price: float = Field(..., title="Price of the item", description="The price must be a positive value", gt=0)
    tax: Optional[float] = Field(None, title="Tax applied to the item", description="Optional tax percentage for the item", ge=0)

# Przykładowe użycie w FastAPI
from fastapi import FastAPI

app = FastAPI()

@app.post("/items/")
async def create_item(item: Item):
    return item
```

## Wyjaśnienie pól:

1. name

- **Typ:** str
- **Opis:** Pole wymagane (... oznacza brak wartości domyślnej), które reprezentuje nazwę tworzonego przedmiotu.
- **Dodatkowe informacje:**
  - title: "Name of the item" – krótki tytuł widoczny w dokumentacji.
  - description: "The name of the item being created" – szczegółowy opis pola.
  - max\_length: 50 – maksymalna długość tekstu.

## 2. description

- **Typ:** Optional[str] (pole opcjonalne, może być None).
- **Opis:** Szczegółowy opis przedmiotu, który użytkownik może dodać.
- **Dodatkowe informacje:**
  - title: "Description of the item" – krótki tytuł w dokumentacji.
  - description: "A detailed description of the item" – szczegółowy opis pola.
  - max\_length: 300 – maksymalna długość tekstu.

## 3. price

- **Typ:** float
- **Opis:** Pole wymagane, które reprezentuje cenę przedmiotu. Musi być wartością dodatnią.
- **Dodatkowe informacje:**
  - title: "Price of the item" – krótki tytuł w dokumentacji.
  - description: "The price must be a positive value" – szczegółowy opis pola.
  - gt: 0 – wartość musi być większa niż 0.

## 4. tax

- **Typ:** Optional[float] (pole opcjonalne, może być None).
- **Opis:** Wartość podatku, która może być zastosowana do przedmiotu. Jeśli nie podano, domyślnie None.
- **Dodatkowe informacje:**
  - title: "Tax applied to the item" – krótki tytuł w dokumentacji.
  - description: "Optional tax percentage for the item" – szczegółowy opis pola.
  - ge: 0 – wartość musi być większa lub równa 0.

```

from typing import List, Optional
from fastapi import FastAPI, HTTPException, Query
from pydantic import BaseModel
import uvicorn

app = FastAPI()

# Definicja klasy Mountains
class Mountains(BaseModel):
    name: str
    height: float # Wysokość w metrach
    location: str # Lokalizacja
    mountain_id: Optional[int] = None

# Prowizoryczna "baza danych" dla obiektów Mountains
database: List[Mountains] = []

# Endpoint: Zwraca wszystkie elementy z bazy danych
@app.get("/mountains")
def get_all_mountains() -> List[Mountains]:
    return database

# Endpoint: Zwraca element o wybranym ID
@app.get("/mountains/{mountain_id}")
def get_mountain(mountain_id: int) -> Mountains:
    mountains = [x for x in database if x.mountain_id == mountain_id]
    if not mountains:
        raise HTTPException(status_code=404, detail="Mountain not found")
    return mountains[0]

# Dodaje element do bazy danych
@app.post("/mountains")
def add_mountain(mountain: Mountains) -> Mountains:
    new_id = len(database) + 1 # Proste generowanie ID
    mountain.mountain_id = new_id
    database.append(mountain)
    return mountain

# Endpoint: Aktualizuje element o podanym ID
@app.put("/mountains/{mountain_id}")
def update_mountain(mountain_id: int, mountain: Mountains) -> Mountains:
    for i, existing_mountain in enumerate(database):
        if existing_mountain.mountain_id == mountain_id:
            updated_mountain = mountain.copy(update={"mountain_id": mountain_id})
            database[i] = updated_mountain
            return updated_mountain
    raise HTTPException(status_code=404, detail="Mountain not found")

# Endpoint: Usuwa element o podanym ID
@app.delete("/mountains/{mountain_id}")
def delete_mountain(mountain_id: int) -> dict:
    for i, existing_mountain in enumerate(database):
        if existing_mountain.mountain_id == mountain_id:
            deleted_mountain = database.pop(i)
            return {"message": "Mountain deleted successfully", "deleted_mountain": deleted_mountain}
    raise HTTPException(status_code=404, detail="Mountain not found")

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

```

