

Politechnika Bydgoska im. J. J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki **Zakład Systemów Teleinformatycznych**



Przedmiot	Skryptowe języki programowania				
Prowadzący	mgr inż. Martyna Tarczewska				
Temat	FastAPI				
Student	Paweł Jońca				
Nr lab.	9, 3/4	Data wykonania	13.01.2025r		
Ocena		Data oddania spr.	13.01.2025r		

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

# Definicia 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] = []
```

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

```
# Endpoint: Aktualizuje element o podanum ID

compountains/(mountain_id)
@app.put("/mountains/(mountain_id)") new *

def update_mountain(mountain_id: int, mountain: Mountains) -> Mountains:

# Szukaj góry w bazie danuch

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

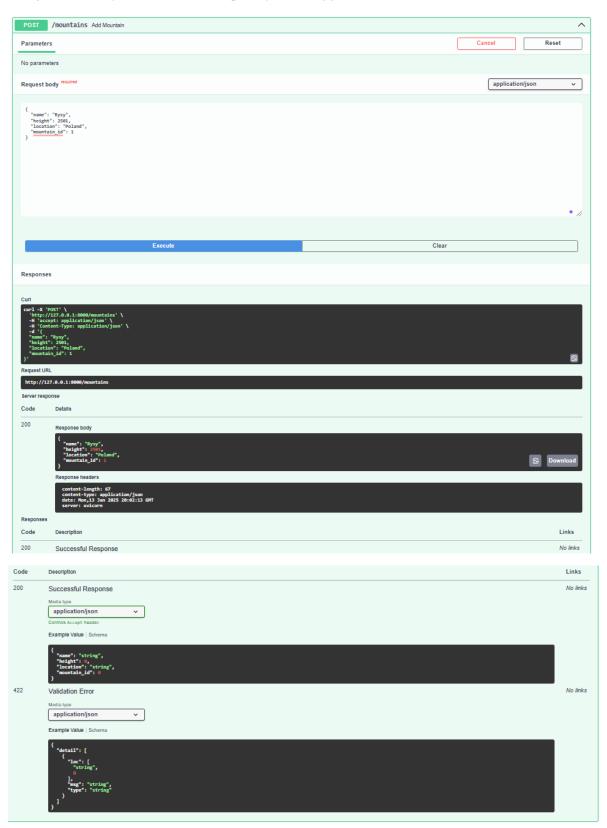
# Jeśli góra nie istnieje, zwróć bład 404

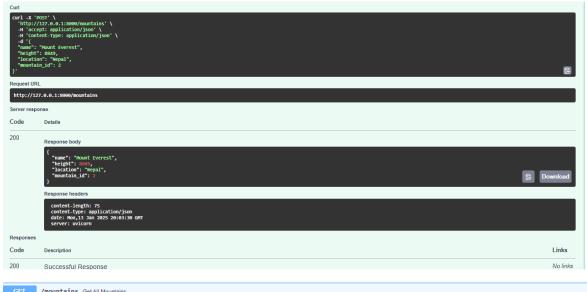
raise HTTPException(status_code=404, detail="Mountain not found")
```

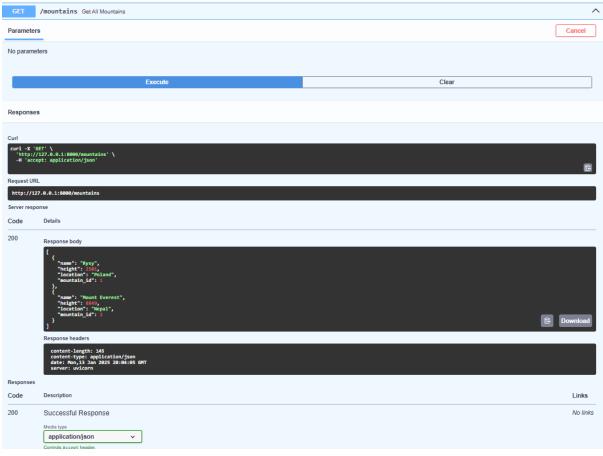
```
# Endpoint: Aktualizuje element o podanym ID
@app.put("/mountains/{mountain_id}")
def update_mountain(mountain_id: int, mountain: Mountains) -> Mountains:
    # Szukaj góry w bazie danych
    for i, existing_mountain in enumerate(database):
        if existing_mountain.mountain_id == mountain_id:
            # 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")
```

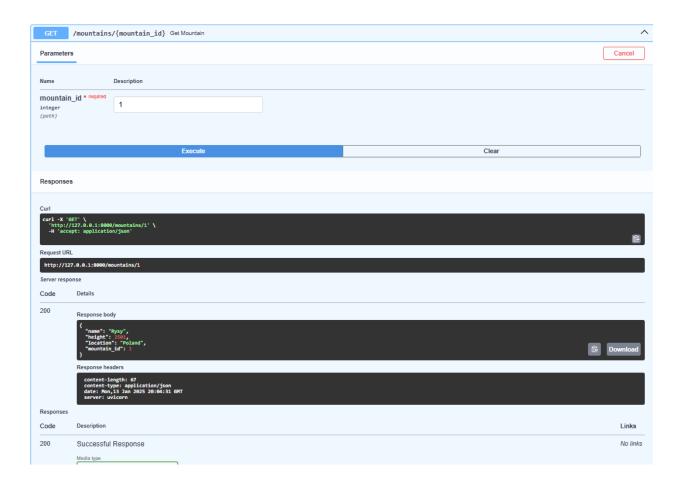
Zad 4

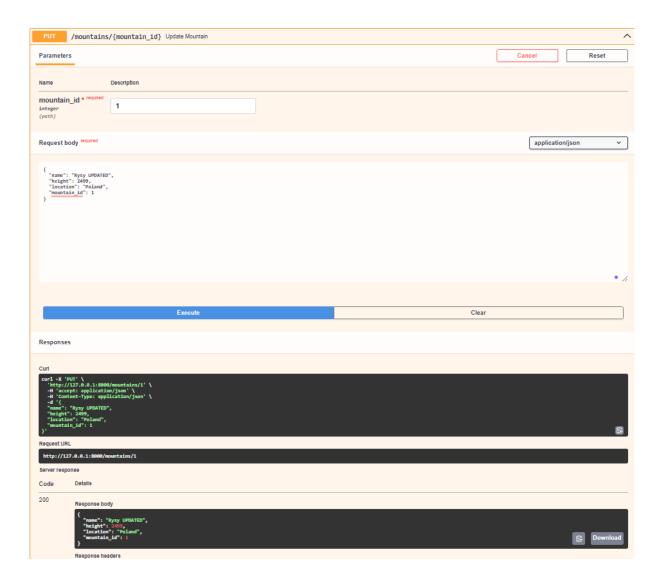
Zad 5 Podgląd na wszystkie dodane endpointy w Swagger UI.

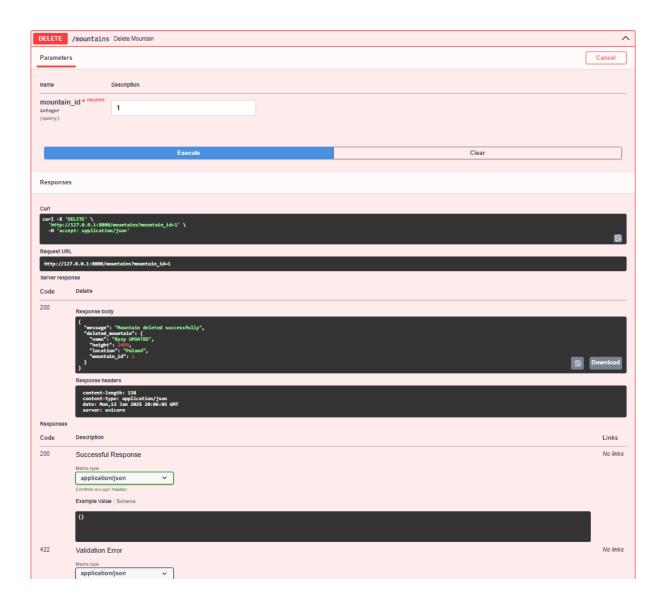


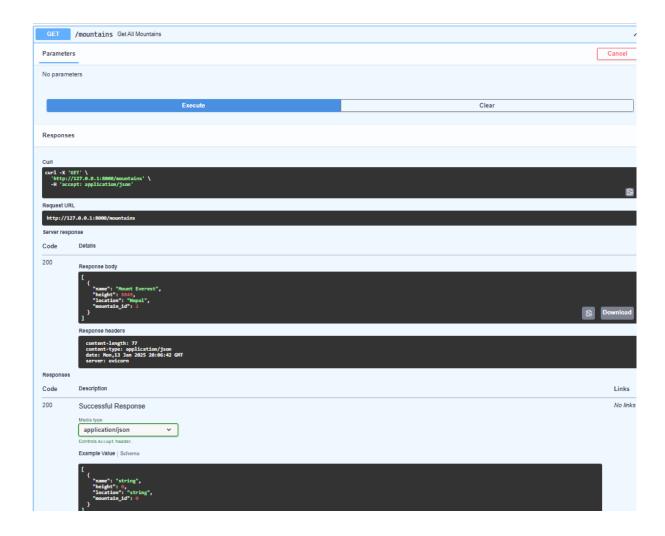












Zad 6.

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):
    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", de-
scription="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
app = FastAPI()
database: List[Mountains] = []
@app.get("/mountains")
@app.get("/mountains/{mountain id}")
            updated mountain = mountain.copy(update={"mountain id": moun-
    raise HTTPException(status code=404, detail="Mountain not found")
@app.delete("/mountains")
        if existing mountain.mountain id == mountain id:
leted mountain": deleted mountain}
```