|  |  |  |  |
| --- | --- | --- | --- |
| Obraz zawierający symbol, design  Opis wygenerowany automatycznie | Politechnika Bydgoska im. J. J. Śniadeckich  **Wydział Telekomunikacji,**  **Informatyki i Elektrotechniki** | | Obraz zawierający symbol, logo, Grafika, Czcionka  Opis wygenerowany automatycznie |
| **Przedmiot** | Skryptowe języki programowania | | |
| **Prowadzący** | mgr inż. Martyna Tarczewska | | |
| **Temat** | *FastAPI* | | |
| **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 == 4]  
 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**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Ikona komputerowa

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Strona internetowa

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, numer

Opis wygenerowany automatycznie**

**Zadanie 6**

Artykuł wyjaśnia, że stosowanie funkcji async w FastAPI pozwala na obsługę wielu równoległych żądań bez konieczności blokowania wątków. To umożliwia serwerowi obsługęwiększej liczby klientów jednocześnie, co z kolei zwiększa wydajność i skalowalność aplikacji.Async/await w Pythonie pozwalają na wydajne zarządzanie współbieżnością, co jest szczególnieprzydatne w aplikacjach sieciowych, takich jak serwisy webowe obsługiwane przez FastAPI.

**Wnioski**

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