Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Кафедра системного аналізу та теорії прийняття рішень

Звіт
з лабораторної роботи № 3

Реалізація простого RESTful API

Виконав
студентка групи К-23 Хілько В.К.

Прийняв Махно М.Ф.

Київ - 2023

Варіант №25
Основні завдання:

1 Розробка моделі даних:
Визначте, які дані будуть зберігатися (наприклад, статті блогу з такими полями: id, title, content, date_created). Дані зберігатись будуть в файл (але якщо вмієте працювати з БД, можна обрати будь-яку БД за бажанням).

2 Реалізація HTTP-методів:
- GET для отримання списку ресурсів або одного ресурсу за ID.
- POST для створення нового ресурсу.
- PUT або PATCH для оновлення ресурсу.
- DELETE для видалення ресурсу.

3 Обробка помилок:
Надання коректних HTTP-кодів відповіді (наприклад, 404 для "ресурс не знайдено" або 400 для "поганий запит").
Повернення інформативних повідомлень про помилки у відповіді.

4 Тестування API:
Використовуйте інструменти, такі як curl або спеціалізовані програми (наприклад, Postman), щоб перевірити роботу вашого API.

25 варіант -
екзамени:
o Поля: id, subject_id, date, location, examiner.

main

```
from flask import Flask
from flask_restful import Api, Resource, reqparse, abort
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
api = Api(app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///exams.db'
db = SQLAlchemy(app)


par = reqparse.RequestParser()
par.add_argument("subject_id", type=int, required=False)
par.add_argument("date", type=str)
```

```
par.add_argument("location", type=int, required=False)
par.add_argument("examiner", type=str, required=False)

class ExamModel(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    subject_id = db.Column(db.Integer)
    date = db.Column(db.String(255))
    location = db.Column(db.Integer)
    examiner = db.Column(db.String(255))

    def __init__(self, subject_id, date, location, examiner):
        self.subject_id = subject_id
        self.date = date
        self.location = location
        self.examiner = examiner

def abort_if_exam_doesnt_exist(exam_id):
    if not ExamModel.query.get(exam_id):
        abort(404, message=f"Exam {exam_id} doesn't exist")

class ExamResource(Resource):
    def get(self, exam_id=None):
        if exam_id :
            abort_if_exam_doesnt_exist(exam_id)
            exam = ExamModel.query.get(exam_id)
            return {
                "id": exam.id,
                "subject_id": exam.subject_id,
                "date": exam.date,
                "location": exam.location,
                "examiner": exam.examiner
            }
        else:
            exams = ExamModel.query.all()
            exam_data = [
                {
                    "id": exam.id,
                    "subject_id": exam.subject_id,
                    "date": exam.date,
                    "location": exam.location,
                    "examiner": exam.examiner
                }
                for exam in exams
            ]
            return '\n'.join(map(str, exam_data))

    def post(self):
        args = par.parse_args()
        new_exam = ExamModel(
            subject_id=args["subject_id"],
            date=args["date"],
            location=args["location"],
            examiner=args["examiner"],
        )
        db.session.add(new_exam)
        db.session.commit()
        return {
            "id": new_exam.id,
```

```
            "subject_id": new_exam.subject_id,
            "date": new_exam.date,
            "location": new_exam.location,
            "examiner": new_exam.examiner
        }, 201

    def delete(self, exam_id):
        abort_if_exam_doesnt_exist(exam_id)
        exam = ExamModel.query.get(exam_id)
        db.session.delete(exam)
        db.session.commit()
        return '', 204

    def put(self, exam_id):
        args = par.parse_args()
        exam = ExamModel.query.get(exam_id)
        if not exam:
            new_exam = ExamModel(
                subject_id=args["subject_id"],
                date=args["date"],
                location=args["location"],
                examiner=args["examiner"],
            )
            db.session.add(new_exam)
            db.session.commit()
            return {
                "id": new_exam.id,
                "subject_id": new_exam.subject_id,
                "date": new_exam.date,
                "location": new_exam.location,
                "examiner": new_exam.examiner
            }
        else:
            abort_if_exam_doesnt_exist(exam_id)
            exam.subject_id = args.get("subject_id", exam.subject_id)
            exam.date = args.get("date", exam.date)
            exam.location = args.get("location", exam.location)
            exam.examiner = args.get("examiner", exam.examiner)
            db.session.commit()
            return {
                "id": exam.id,
                "subject_id": exam.subject_id,
                "date": exam.date,
                "location": exam.location,
                "examiner": exam.examiner
            }, 201


api.add_resource(ExamResource, "/api/exams",
"/api/exams/<int:exam_id>")


if __name__ == "__main__":
    with app.app_context():
        db.drop_all()
        db.create_all()
        first_db = {
```

```
            1: {"subject_id": 23, "date": "2023-12-04 9:00", "location":
205, "examiner": "Ivanov S.M."},
            2: {"subject_id": 311, "date": "2023-12-05 9:00",
"location": 232, "examiner": "Onotsky V.V"},
            3: {"subject_id": 24, "date": "2023-12-14 9:00", "location":
1, "examiner": "Korobova M.V."},
            4: {"subject_id": 14, "date": "2023-12-18 9:00", "location":
1, "examiner": "Ivokhin E.V."},
        }

        for exam_id, exam_info in first_db.items():
            new_exam = ExamModel(
                subject_id=exam_info["subject_id"],
                date=exam_info["date"],
                location=exam_info["location"],
                examiner=exam_info["examiner"]
            )
            db.session.add(new_exam)

        db.session.commit()
    app.run(debug=True, port=1103, host="127.0.0.1")
```

local

```
import json
import requests

BASE_URL = "http://127.0.0.1:1103/api/exams"
def liner():
    print('~'*40)
def print_error(e):
    print(f"An error occurred while making the request: {e}")

def get_exam(exam_id=None):
    try:
        if not exam_id :
            url = f"{BASE_URL}"
            res = requests.get(url)
            res.raise_for_status()
            print('Get exams:')
            print(res.json())
            liner()
        else:
            url = f"{BASE_URL}/{exam_id}"
            res = requests.get(url)
            if res.status_code == 404:
                print(f'Get exam {exam_id}')
                print(f"Exam {exam_id} not found")
                liner()
            else:
                res.raise_for_status()
                print('Get exam')
                print(res.json())
                liner()
    except requests.exceptions.RequestException as e:
        print_error(e)

def post_exam(subject_id, date, location, examiner):
    try:
```

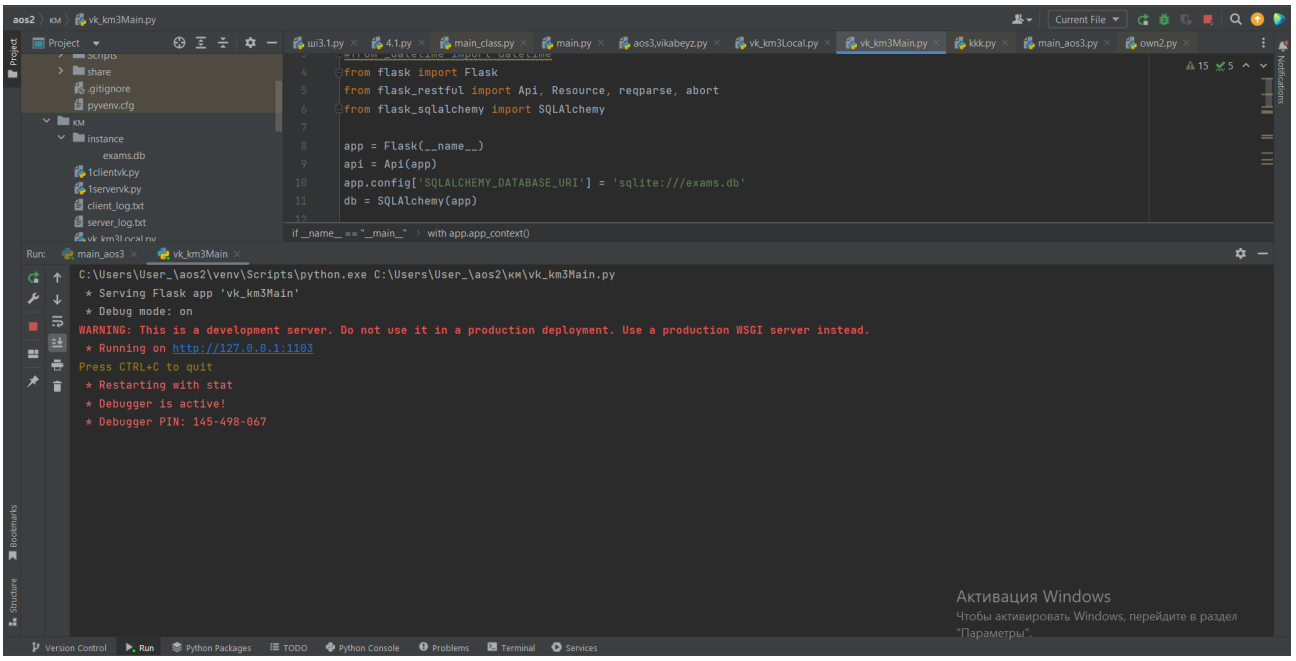
```
        data = {
            "subject_id": subject_id,
            "date": date,
            "location": location,
            "examiner": examiner
        }
        headers = {
            "Content-Type": "application/json"
        }
        res = requests.post(BASE_URL, data=json.dumps(data),
headers=headers)
        res.raise_for_status()
        print('Post exam')
        print(res.json())
        liner()
    except requests.exceptions.RequestException as e:
        print('Post exam')
        print_error(e)
        liner()

def put_exam(exam_id, subject_id=None, date=None, location=None,
examiner=None):
    try:
        url = f"{BASE_URL}/{exam_id}"
        data = {}
        if subject_id:
            data["subject_id"] = subject_id
        if date:
            data["date"] = date
        if location:
            data["location"] = location
        if examiner:
            data["examiner"] = examiner
        headers = {
            "Content-Type": "application/json"
        }
        res = requests.put(url, data=json.dumps(data), headers=headers)
        if res.status_code == 404:
            print(f"Exam {exam_id} not found")
        elif res.status_code == 400:
            print(res.json()["message"])
            liner()
        else:
            res.raise_for_status()
            print(f'Put exam {exam_id}')
            print(res.json())
            liner()
    except requests.exceptions.RequestException as e:
        print(f'Put exam {exam_id}')
        print_error(e)
        liner()


def delete_exam(exam_id):
    try:
        url = f"{BASE_URL}/{exam_id}"
        res = requests.delete(url)
```

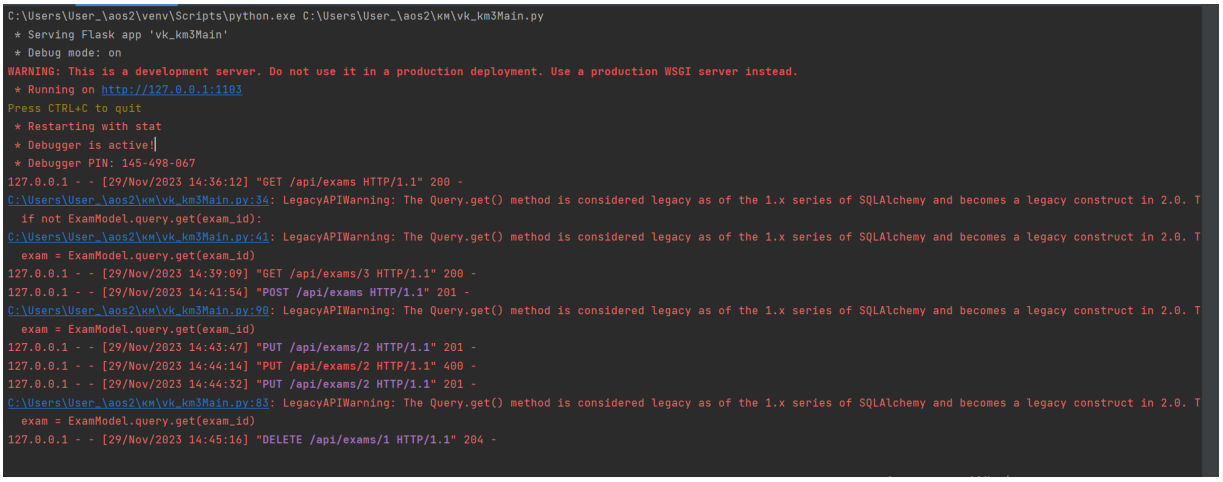
```
        if res.status_code == 404:
            print('Delete exam')
            print(f"Exam {exam_id} not found")
        else:
            res.raise_for_status()
            print(f'Delete exam {exam_id}')
            print(f"Exam {exam_id} deleted")

    except requests.exceptions.RequestException as e:
        print('Delete exam')
        print_error(e)

    liner()


def print_menu():
    print("1. Get exams")
    print("2. Get exam by ID")
    print("3. Post exam")
    print("4. Put exam")
    print("5. Delete exam")
    print("0. Exit")

def main():
    while True:
        print_menu()
        choice = input("Enter your choice (0-6): ")

        if choice == "0":
            print("Exiting the program. Goodbye!")
            break
        elif choice == "1":
            get_exam()
        elif choice == "2":
            exam_id = input("Enter exam ID: ")
            get_exam(exam_id)
        elif choice == "3":
            subject_id = input("Enter subject ID: ")
            date = input("Enter date: ")
            location = input("Enter location: ")
            examiner = input("Enter examiner: ")
            post_exam(subject_id, date, location, examiner)
        elif choice == "4":
            exam_id = input("Enter exam ID: ")
            subject_id = input("Enter new subject ID(press enter if you
don't want to change) : ")
            date = input("Enter new date(press enter if you don't want
to change) : ")
            location = input("Enter new location (press enter if you
don't want to change): ")
            examiner = input("Enter new examiner(press enter if you
don't want to change): ")
            put_exam(exam_id, subject_id=subject_id or None,
                                date=date or None, location=location
or None, examiner=examiner or None)
        elif choice == "5":
            exam_id = input("Enter exam ID: ")
            delete_exam(exam_id)
```

```
        else:
            print("Invalid choice. Please enter a number between 0 and
6.")

main()
```

Консоль

main





local

```
1. Get exams
2. Get exam by ID
3. Post exam
4. Put exam
5. Delete exam
0. Exit
Enter your choice (0-6): 2
Enter exam ID: 3
Get exam
{'id': 3, 'subject_id': 24, 'date': '2023-12-14 9:00', 'location': 1, 'examiner': 'Korobova M.V.'}
```

```
0. Exit
Enter your choice (0-6): 1
Get exams:
{'id': 1, 'subject_id': 23, 'date': '2023-12-04 9:00', 'location': 205, 'examiner': 'Ivanov S.M.'}
{'id': 2, 'subject_id': 311, 'date': '2023-12-05 9:00', 'location': 232, 'examiner': 'Onotsky V.V'}
{'id': 3, 'subject_id': 24, 'date': '2023-12-14 9:00', 'location': 1, 'examiner': 'Korobova M.V.'}
{'id': 4, 'subject_id': 14, 'date': '2023-12-18 9:00', 'location': 1, 'examiner': 'Ivokhin E.V.'}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1. Get exams
```

```
Enter your choice (0-6): 3
Enter subject ID: 4
Enter date: 12:09:2022 9:00
Enter location: 12
Enter examiner: Who
Post exam
{'id': 5, 'subject_id': 4, 'date': '12:09:2022 9:00', 'location': 12, 'examiner': 'Who'}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
Enter your choice (0-6): 4
Enter exam ID: 2
Enter new subject ID(press enter if you don't want to change) :
Enter new date(press enter if you don't want to change) :
Enter new location (press enter if you don't want to change): 1254
Enter new examiner(press enter if you don't want to change):
Put exam 2
{'id': 2, 'subject_id': None, 'date': None, 'location': 1254, 'examiner': None}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
0. Exit
Enter your choice (0-6): 5
Enter exam ID: 1
Delete exam 1
Exam 1 deleted
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

База даних

до

Table: exam_model Page: 0 Jump << < 1-1 > >> Refresh

id	subject_id	date	location	examiner
1	23	2023-12-04 ...	205	Ivanov S.M.
2	311	2023-12-05 ...	232	Onotsky V.V
3	24	2023-12-14 ...	1	Korobova ...
4	14	2023-12-18 ...	1	Ivokhin E.V.

після

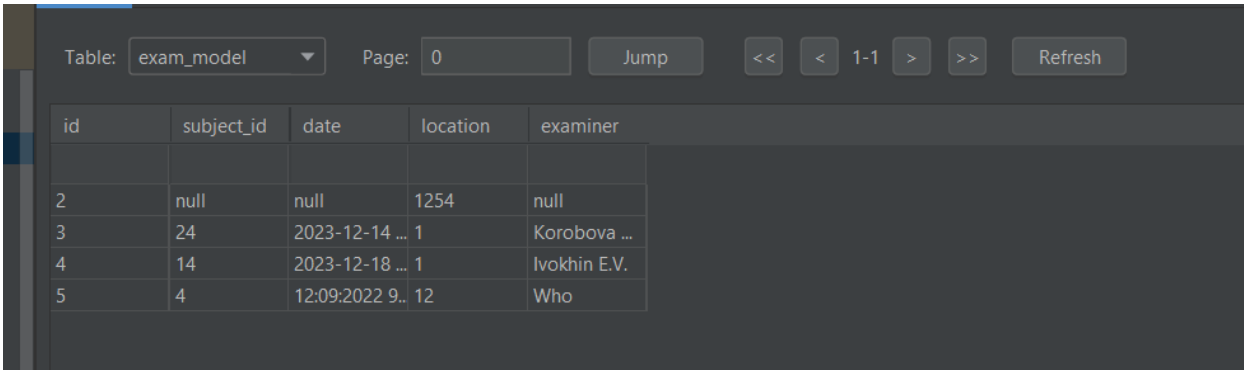


Table: exam_model
Page: 0
Jump
<<
<
1-1
>
>>
Refresh
id
subject_id
date
location
examiner
2
null
null
1254
null
3
24
2023-12-14 ...
1
Korobova ...
4
14
2023-12-18 ...
1
Ivokhin E.V.
5
4
12:09:2022 9..
12
Who