REST_API

Завдання 1_Develop a REST API

Функціональність

*Розробити REST API для керування студентами на Python за допомогою бібліотеки Flask під назвою **app.py** із збереженням даних у файлі **students.csv**. API має підтримувати запити GET, POST, PUT, PATCH і DELETE. Кожен студент повинен мати поля для ідентифікатора, імені, прізвища та віку.*

HTTP запити

GET

Отримати інформацію про конкретного студента (відобразити всю доступну інформацію):

- за своїм ідентифікатором; якщо наданий ідентифікатор не знайдено у файлі CSV, повертає помилку.
- За своїм прізвищем; якщо є кілька студентів з однаковими прізвищами, відобразити їх усіх; якщо вказане прізвище не знайдено у файлі CSV, повертає помилку.
- Отримати список усіх студентів (відобразити всю доступну інформацію).

POST

Створити нового студента

- Поле ідентифікатора має генеруватися автоматично з кроком +1.
- Тіло запиту POST має містити ім'я, прізвище та вік.
- Якщо в тілі POST передано неіснуюче поле або якщо поля не передано взагалі, поверніть помилку.
- Якщо будь-яке з полів відсутнє в тілі POST, поверніть повідомлення про помилку без запису у файл CSV.
- У разі успішного запиту повернути інформацію про доданого студента.

PUT

Оновити інформацію про студента по його ID

- Має бути можливість оновлювати поля імені, прізвища та віку.
- Якщо наданий ідентифікатор не знайдено у файлі CSV, поверніть помилку.
- Тіло запиту PUT має містити ім'я, прізвище та вік відповідно.
- Якщо в тілі PUT передано неіснуюче поле або якщо поля не передано взагалі, поверніть помилку.
- У разі успішного запиту повернути оновлену інформацію про студента.

PATCH

Оновити вік про студента по його ID

- Має бути можливість оновити вікове поле.
- Якщо наданий ідентифікатор не знайдено у файлі CSV, поверніть помилку.
- Тіло запиту PATCH має містити вік.
- Якщо в тілі PATCH передано неіснуюче поле або якщо поля не передано взагалі, поверніть помилку.
- У разі успішного запиту повернути оновлену інформацію про студента.

DELETE

Видалити студента по його ID із CSV файлу

- Якщо наданий ідентифікатор не знайдено у файлі CSV, поверніть помилку.
- У разі успішного запиту поверніть повідомлення з підтвердженням успішного видалення студента.

Створимо робочу директорію в якому буде два файли – ***app.py*** та ***students.csv***
Після чого створимо віртуальне середовище

```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ python3 -m venv venv
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ source venv/bin/activate
(venv) dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$
```

Встановимо Flask

```
(venv) dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ pip install Flask
```

Код

Імпорт потрібних модулів та ініціалізація Flask

```python
from flask import Flask, request, jsonify, abort
import csv
import os

app = Flask(__name__)
CSV_FILE = 'students.csv'

```

Створимо допоміжні функції для читання та запису даних у ***students.csv***

```python
# Функція для зчитування даних студентів з CSV файлу
def read_students():
    students = []
    if os.path.exists(CSV_FILE):
        with open(CSV_FILE, mode='r') as file:
            reader = csv.DictReader(file)
            students = list(reader)
    return students

# Функція для запису оновлених даних студентів до CSV файлу
def write_students(students):
    with open(CSV_FILE, mode='w', newline='') as file:
        fieldnames = ['id', 'first_name', 'last_name', 'age']
        writer = csv.DictWriter(file, fieldnames=fieldnames)
        writer.writeheader()
        writer.writerows(students)

```

GET метод. Отримання інформації про студентів за ID, прізвищем або отримання списку всіх студентів

```python
# Маршрут для отримання інформації про студентів за ID, прізвищем або отримання списку всіх студентів
@app.route('/students', methods=['GET'])
def get_students():
    id = request.args.get('id')
    last_name = request.args.get('last_name')
    students = read_students()

    # Пошук студента за ID
    if id:
        student = next((s for s in students if int(s['id']) == int(id)), None)
        if student:
            return jsonify(student)
        else:
            return abort(404, description="Student not found")
    # Пошук студентів за прізвищем
    elif last_name:
        filtered_students = [s for s in students if s['last_name'] == last_name]
        if filtered_students:
            return jsonify(filtered_students)
        else:
            return abort(404, description="No students found with that last name")
    # Повернення списку всіх студентів
    else:
        return jsonify(students)

# Маршрут для отримання списку всіх студентів (альтернативний маршрут)
@app.route('/students/all', methods=['GET'])
def get_all_students():
    students = read_students()
    return jsonify(students)

```

POST метод. Додавання нового студента

```python
# Маршрут для додавання нового студента
@app.route('/students', methods=['POST'])
def add_student():
    new_student = request.get_json()
    required_fields = ['first_name', 'last_name', 'age']

    # Перевірка наявності необхідних полів у запиті
    if not all(field in new_student for field in required_fields):
        return abort(400, description="Missing fields in request")

    # Перевірка на відсутність зайвих полів у запиті
    if any(field not in required_fields for field in new_student.keys()):
        return abort(400, description="Extra fields in request")

    # Генерація нового ID для студента
    students = read_students()
    new_id = max([int(s['id']) for s in students], default=0) + 1
    new_student['id'] = new_id

    # Додавання нового студента до списку і запис у CSV файл
    students.append(new_student)
    write_students(students)
    return jsonify(new_student), 201

```

PUT метод. Оновлення інформації про студента по його ID

```python
# Маршрут для оновлення інформації про студента за його ID
@app.route('/students/<int:id>', methods=['PUT'])
def update_student(id):
    updated_student = request.get_json()
    required_fields = ['first_name', 'last_name', 'age']

    # Перевірка наявності всіх необхідних полів у запиті
    if not all(field in updated_student for field in required_fields):
        return abort(400, description="Missing fields in request")

    # Перевірка на відсутність зайвих полів у запиті
    if any(field not in required_fields for field in updated_student.keys()):
        return abort(400, description="Extra fields in request")

    # Оновлення інформації про студента
    students = read_students()
    for student in students:
        if int(student['id']) == id:
            student.update(updated_student)
            write_students(students)
            return jsonify(student)

    return abort(404, description="Student not found")

```

PATCH метод. Оновлення віку студента по його ID

```python
# Маршрут для оновлення віку студента за його ID
@app.route('/students/<int:id>/age', methods=['PATCH'])
def update_student_age(id):
    updated_age = request.get_json()

    # Перевірка наявності поля віку в запиті
    if 'age' not in updated_age:
        return abort(400, description="Missing age field in request")

    # Перевірка на відсутність зайвих полів у запиті
    if len(updated_age.keys()) > 1:
        return abort(400, description="Extra fields in request")

    # Оновлення віку студента
    students = read_students()
    for student in students:
        if int(student['id']) == id:
            student['age'] = updated_age['age']
            write_students(students)
            return jsonify(student)

    return abort(404, description="Student not found")

```

DELETE метод. Видалення студента за його ID

```python
# Маршрут для видалення студента за його ID
@app.route('/students/<int:id>', methods=['DELETE'])
def delete_student(id):
    students = read_students()
    student_to_delete = next((s for s in students if int(s['id']) == id), None)

    # Видалення студента зі списку і запис у CSV файл
    if student_to_delete:
        students.remove(student_to_delete)
        write_students(students)
        return jsonify({"message": "Student deleted successfully"})
    else:
        return abort(404, description="Student not found")

```

Налаштування запуску та роботи Flask

```python
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Результати виконання

Відкриємо файл students.csv та заповнимо трохи даними

```
id,first_name,last_name,age
1,John,Doe,20
2,Jane,Smith,22
3,Emily,Jones,19
4,Michael,Brown,21
5,Sarah,Johnson,23
6,David,Williams,22
7,Linda,Davis,20
8,Robert,Garcia,19
9,Karen,Miller,21
10,James,Martinez,22
```

Запустимо Flask та відкриємо нове термінальне вікно для тестування.

```
(venv) dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.0.109:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 742-438-295
```

GET

Отримання всіх студентів

```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ curl -X GET http://127.0.0.1:5000/students/all
[
  {
    "age": "20",
    "first_name": "John",
    "id": "1",
    "last_name": "Doe"
  },
  {
    "age": "22",
    "first_name": "Jane",
    "id": "2",
    "last_name": "Smith"
  },
  {
    "age": "19",
    "first_name": "Emily",
    "id": "3",
    "last_name": "Jones"
  },
  {
    "age": "21",
    "first_name": "Michael",
    "id": "4",
    "last_name": "Brown"
  },
  {
    "age": "23",
    "first_name": "Sarah",
    "id": "5",
    "last_name": "Johnson"
  },
  {
    "age": "22",
    "first_name": "David",
    "id": "6",
    "last_name": "Williams"
  },
  {
    "age": "20",
    "first_name": "Linda",
    "id": "7",
    "last_name": "Davis"
  },
  {
    "age": "19",
    "first_name": "Robert",
    "id": "8",
    "last_name": "Garcia"
  },
  {
    "age": "21",
    "first_name": "Karen",
    "id": "9",
    "last_name": "Miller"
  },
  {
    "age": "22",
    "first_name": "James",
    "id": "10",
    "last_name": "Martinez"
  }
]
```

Отримання студентів за ID

```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ curl -X GET http://127.0.0.1:5000/students?id=1
{
  "age": "20",
  "first_name": "John",
  "id": "1",
  "last_name": "Doe"
}
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$
```

Отримання студентів за прізвищем

```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ curl -X GET http://127.0.0.1:5000/students?last_name=Doe
[
  {
    "age": "20",
    "first_name": "John",
    "id": "1",
    "last_name": "Doe"
  }
]
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$
```

POST

Додавання нового студента

```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ curl -X POST http://127.0.0.1:5000/students -H "Content-Type: application/json" -d '{"first_name": "Anna", "last_name": "White", "age": 25}'
{
  "age": 25,
  "first_name": "Anna",
  "id": 11,
  "last_name": "White"
}
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$
```

PUT

Оновлення інформації про студента

```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ curl -X PUT http://127.0.0.1:5000/students/1 -H "Content-Type: application/json" -d '{"first_name": "John", "last_name": "Doe", "age": 21}'
{
  "age": 21,
  "first_name": "John",
  "id": "1",
  "last_name": "Doe"
}
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$
```

PATCH

Оновлення віку студента

```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ curl -X PATCH http://127.0.0.1:5000/students/1/age -H "Content-Type: application/json" -d '{"age": 22}'
{
  "age": 22,
  "first_name": "John",
  "id": "1",
  "last_name": "Doe"
}
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$
```

DELETE

Видалення студента

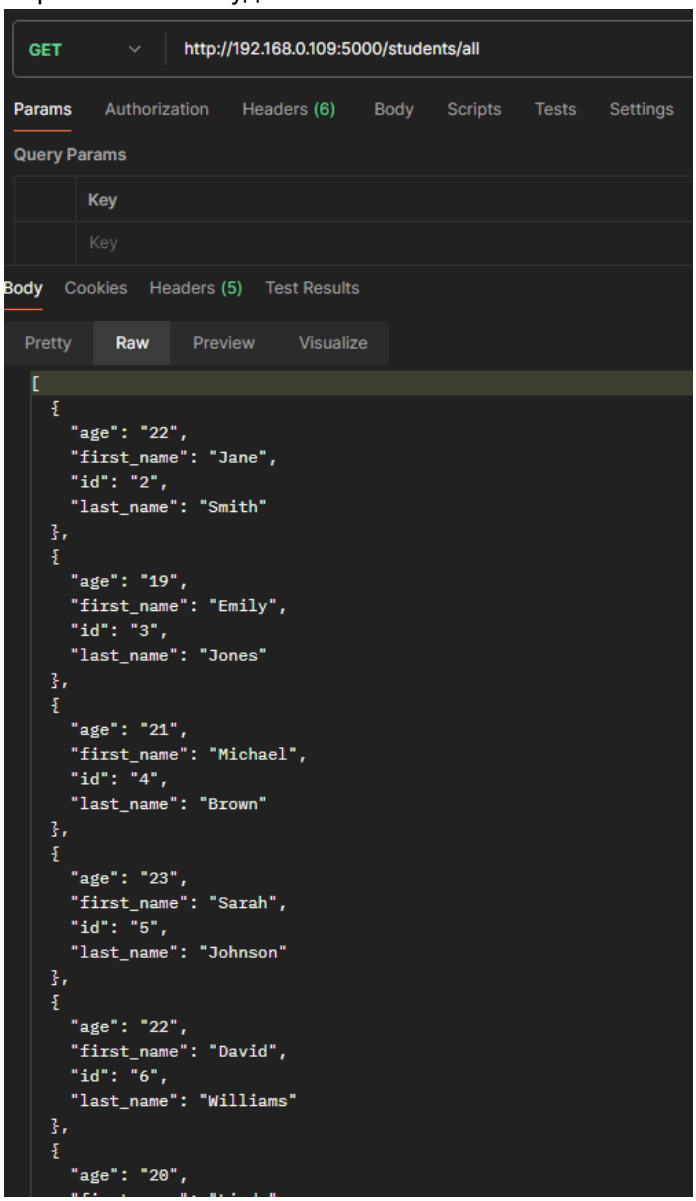
```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$ curl -X DELETE http://127.0.0.1:5000/students/1
{
  "message": "Student deleted successfully"
}
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_12_REST_API$
```

```
127.0.0.1 - - [31/Aug/2024 08:57:49] "GET /students/all HTTP/1.1" 200 -
127.0.0.1 - - [31/Aug/2024 09:06:01] "GET /students?id=1 HTTP/1.1" 200 -
127.0.0.1 - - [31/Aug/2024 09:09:04] "GET /students?last_name=Doe HTTP/1.1" 200 -
127.0.0.1 - - [31/Aug/2024 09:11:19] "POST /students HTTP/1.1" 201 -
127.0.0.1 - - [31/Aug/2024 09:13:25] "PUT /students/1 HTTP/1.1" 200 -
127.0.0.1 - - [31/Aug/2024 09:14:54] "PATCH /students/1/age HTTP/1.1" 200 -
127.0.0.1 - - [31/Aug/2024 09:16:45] "DELETE /students/1 HTTP/1.1" 200 -
```

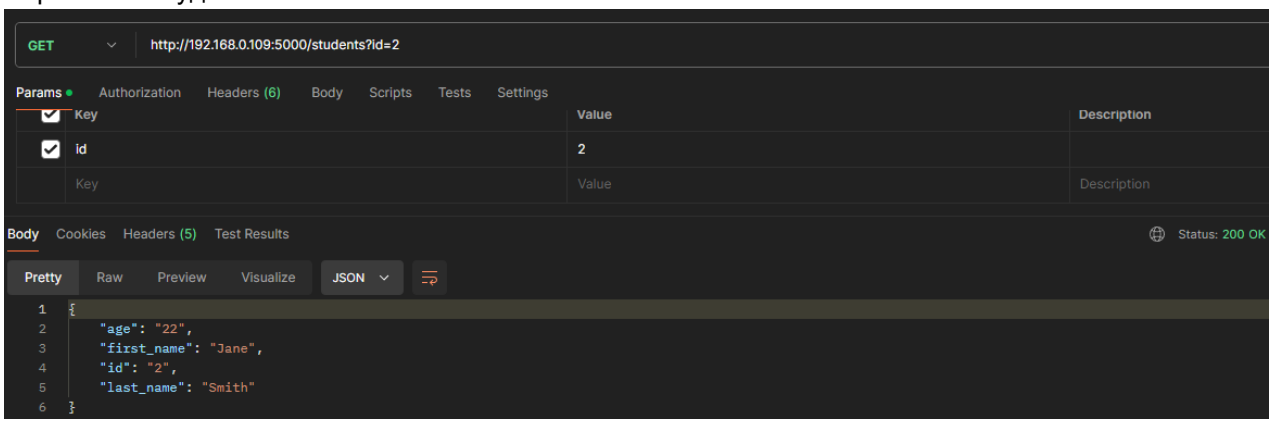
Тестування через Postman

GET

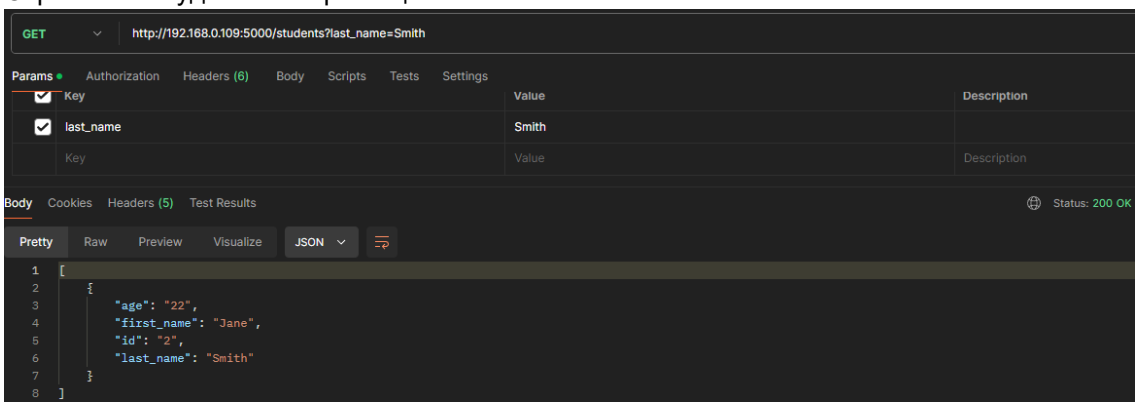
Отримання всіх студентів



Отримання студентів за ID

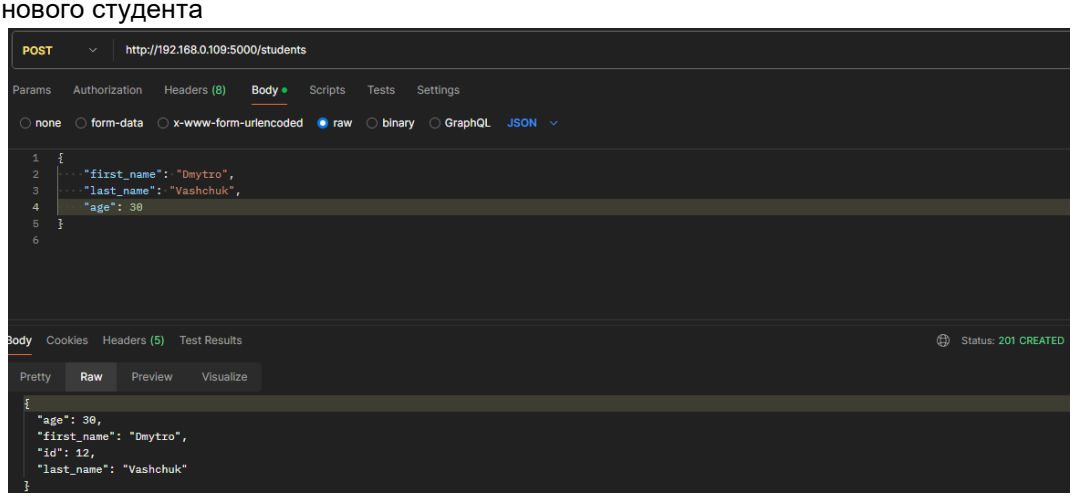


Отримання студентів за прізвищем



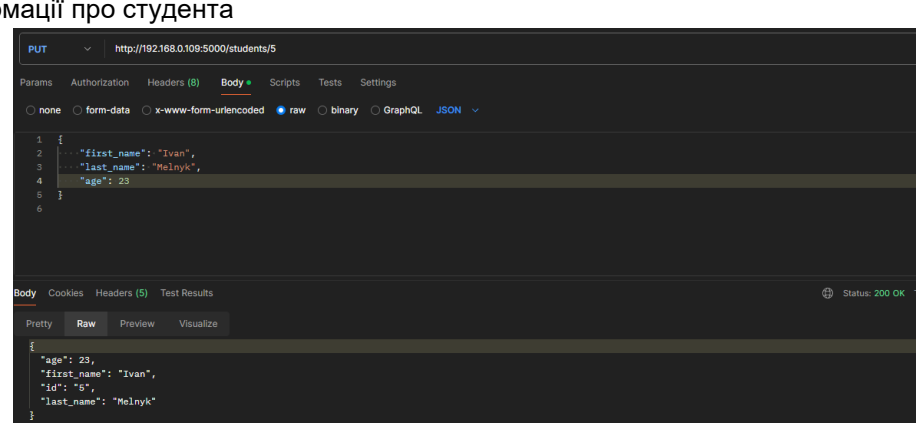
POST

Додавання нового студента



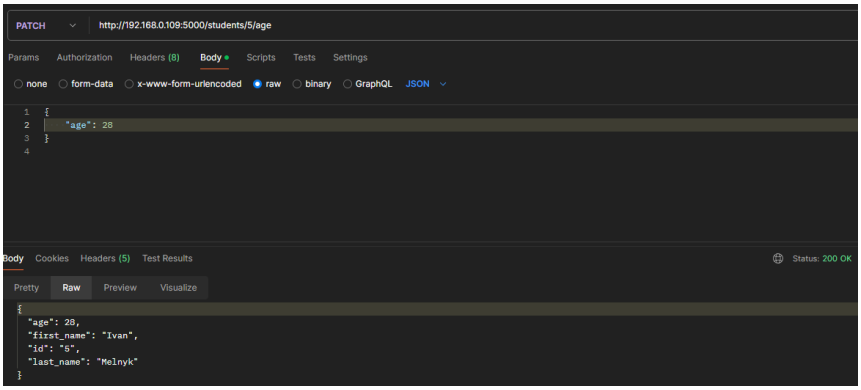
PUT

Оновлення інформації про студента



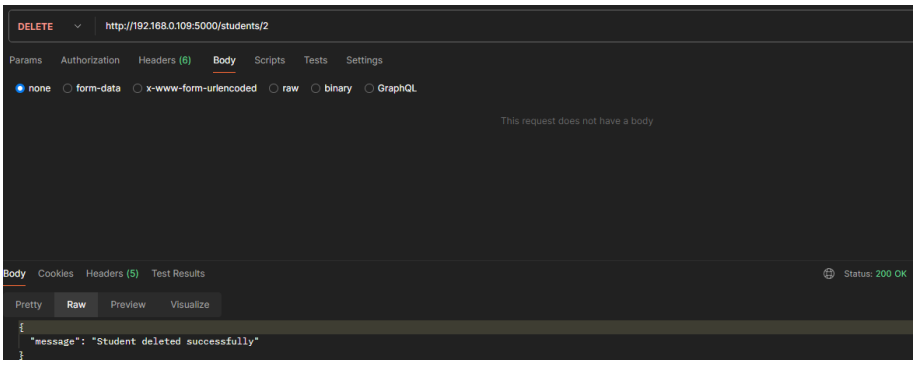
PATCH

Оновлення віку студента



DELETE

Видалення студента




```
192.168.0.107 - - [31/Aug/2024 09:19:09] "GET /students/all HTTP/1.1" 200 -
192.168.0.107 - - [31/Aug/2024 09:21:34] "GET /students?id=2 HTTP/1.1" 200 -
192.168.0.107 - - [31/Aug/2024 09:23:22] "GET /students?last_name=Smith HTTP/1.1" 200 -
192.168.0.107 - - [31/Aug/2024 09:26:50] "POST /students HTTP/1.1" 201 -
192.168.0.107 - - [31/Aug/2024 09:29:34] "PUT /students/5 HTTP/1.1" 200 -
192.168.0.107 - - [31/Aug/2024 09:31:28] "PATCH /students/5/age HTTP/1.1" 200 -
192.168.0.107 - - [31/Aug/2024 09:33:21] "DELETE /students/2 HTTP/1.1" 200 -
```

Завдання 2_ Create test_requests.py

Функціональність

*Створіть файл **test_requests.py**, щоб перевірити створений REST API. У цьому файлі продемонструйте функціональність усіх методів із використанням бібліотеки запитів у такій послідовності:.*

- Отримати всіх існуючих студентів (GET).
- Створіть трьох студентів (POST).
- Отримати інформацію про всіх студентів (GET).
- Оновити вік другого студента (PATCH).
- Отримати інформацію про другого студента (GET).
- Оновіть ім’я, прізвище та вік третього учня (PUT).
- Отримати інформацію про третього студента (GET).
- Отримати всіх існуючих студентів (GET).
- Видалити першого користувача (DELETE).
- Отримати всіх існуючих студентів (GET).
- Відобразіть результати виконання в консолі та запишіть їх у файл results.txt.

Код

Створимо файл ***test_requests.py*** в корені проекту

test_requests.py

```python
import requests
import json

# URL вашого API
BASE_URL = "http://127.0.0.1:5000/students"
RESULTS_FILE = "results.txt"

def log_and_print(message):
    print(message)
    with open(RESULTS_FILE, "a") as file:
        file.write(message + "\n")

def main():
    # Очистити файл перед початком тестів
    open(RESULTS_FILE, 'w').close()

    # 1. Отримати всіх існуючих студентів (GET)
    response = requests.get(BASE_URL)
    log_and_print("GET all students:\n" + response.text)

    # 2. Створити трьох студентів (POST)
    students = [
        {"first_name": "Dmytro", "last_name": "Vashchuk", "age": 20},
        {"first_name": "Roma", "last_name": "TOR", "age": 22},
        {"first_name": "Mykola", "last_name": "Melnyl", "age": 24}
    ]
    for student in students:
        response = requests.post(BASE_URL, json=student)
        log_and_print(f"POST create student {student['first_name']}:\n" + response.text)
```

```python

    # 3. Отримати інформацію про всіх студентів (GET)
    response = requests.get(BASE_URL)
    log_and_print("GET all students after adding new ones:\n" + response.text)

    # 4. Оновити вік другого учня (PATCH)
    student_id = 2  # Ідентифікатор другого студента
    updated_age = {"age": 23}
    response = requests.patch(f"{BASE_URL}/{student_id}/age", json=updated_age)
    log_and_print(f"PATCH update age of student {student_id}:\n" + response.text)

    # 5. Отримати інформацію про другого студента (GET)
    response = requests.get(f"{BASE_URL}?id={student_id}")
    log_and_print(f"GET student {student_id} info:\n" + response.text)

    # 6. Оновити ім'я, прізвище та вік третього студента (PUT)
    student_id = 3  # Ідентифікатор третього студента
    updated_info = {"first_name": "James", "last_name": "Bond", "age": 30}
    response = requests.put(f"{BASE_URL}/{student_id}", json=updated_info)
    log_and_print(f"PUT update student {student_id} info:\n" + response.text)

    # 7. Отримати інформацію про третього студента (GET)
    response = requests.get(f"{BASE_URL}?id={student_id}")
    log_and_print(f"GET student {student_id} info:\n" + response.text)

    # 8. Отримати всіх існуючих студентів (GET)
    response = requests.get(BASE_URL)
    log_and_print("GET all students after updates:\n" + response.text)

    # 9. Видалити першого користувача (DELETE)
    student_id = 1  # Ідентифікатор першого студента
    response = requests.delete(f"{BASE_URL}/{student_id}")
    log_and_print(f"DELETE student {student_id}:\n" + response.text)

    # 10. Отримати всіх існуючих студентів (GET)
    response = requests.get(BASE_URL)
    log_and_print("GET all students after deletion:\n" + response.text)

if __name__ == "__main__":
    main()
```

Після запуску файлу ***test_requests.py*** всі результати виконання запитів зберігаються у файлі **results.txt**