REST_API

Завдання 1_Dockerfile

*Створити Dockerfile для REST_API із ДЗ_12. Потрібно вибрати **gunicorn** як веб-сервер для запуску API та налаштувати його для запуску на 0.0.0.0:8000.*

Створюємо в робочому каталозі файл Docker file із наступним вмістом

```
# Базовий образ Python 3.9 на основі Slim (мінімальна версія)
FROM python:3.9-slim

# Встановлюємо робочу директорію всередині контейнера
WORKDIR /app

# Копіюємо файл requirements.txt у контейнер
COPY requirements.txt /app/requirements.txt

# Встановлюємо залежності з requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Копіюємо всі інші файли в робочу директорію контейнера
COPY . /app

# Вказуємо порт, який буде використовуватись додатком
EXPOSE 8000

# Команда для запуску додатку за допомогою Gunicorn
CMD ["gunicorn", "--bind", "0.0.0.0:8000", "app:app"]
```

Збираємо Docker-образ

```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_13_Docker/src$ docker build -t home
work12-api .
[+] Building 1.9s (10/10) FINISHED                                docker:default
 => [internal] load build definition from Dockerfile                        0.1s
 => => transferring dockerfile: 877B                                        0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim          1.2s
 => [internal] load .dockerignore                                           0.1s
 => => transferring context: 2B                                             0.0s
 => [1/5] FROM docker.io/library/python:3.9-slim@sha256:2851c06da1fdc3c45178  0.0s
 => [internal] load build context                                           0.1s
 => => transferring context: 19.11kB                                        0.0s
 => CACHED [2/5] WORKDIR /app                                               0.0s
 => CACHED [3/5] COPY requirements.txt /app/requirements.txt                0.0s
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt         0.0s
 => [5/5] COPY . /app                                                       0.1s
 => exporting to image                                                      0.2s
 => => exporting layers                                                     0.1s
 => => writing image sha256:d38fecf96aa0f4e485d2ba4da5811d914f28b639f9df1d72  0.0s
 => => naming to docker.io/library/homework12-api                           0.0s
```

Переконаймося, що образ був успішно створений

```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_13_Docker/src$ docker images
REPOSITORY       TAG       IMAGE ID       CREATED          SIZE
homework12-api   latest    d38fecf96aa0   12 minutes ago   141MB
<none>           <none>    be9b6adc722d   2 days ago       163MB
nginx            latest    39286ab8a5e1   5 weeks ago      188MB
hello-world      latest    d2c94e258dcb   16 months ago    13.3kB
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_13_Docker/src$
```

Запускаємо контейнер після успішної збірки

```
dmytro@ubuntuserver:~/dan_it_homeworks/Homework_13_Docker/src$ docker run -p 8000:8
000 homework12-api
[2024-09-20 07:13:04 +0000] [1] [INFO] Starting gunicorn 20.1.0
[2024-09-20 07:13:04 +0000] [1] [INFO] Listening at: http://0.0.0.0:8000 (1)
[2024-09-20 07:13:04 +0000] [1] [INFO] Using worker: sync
[2024-09-20 07:13:04 +0000] [7] [INFO] Booting worker with pid: 7
```

Завдання 2_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

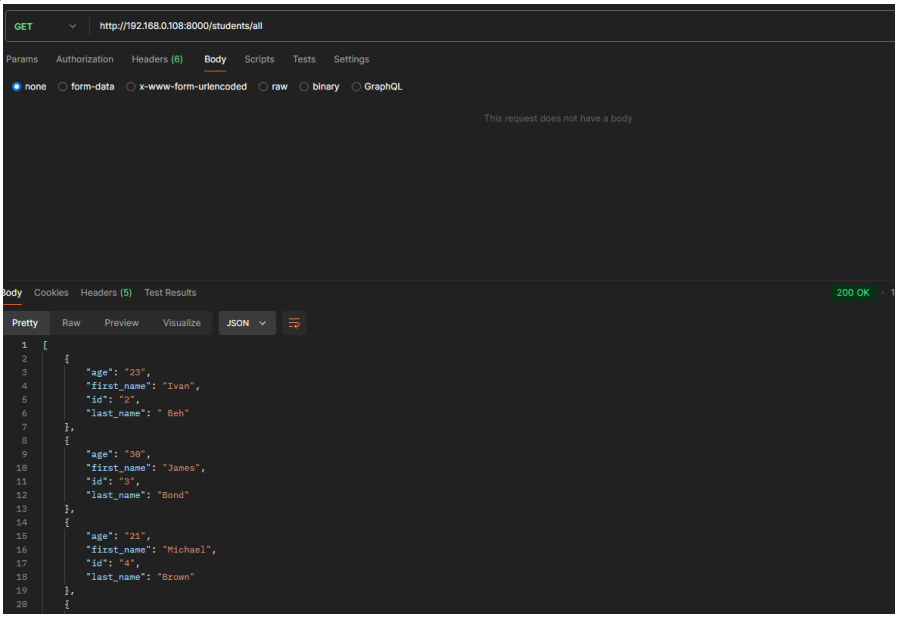
Видалити студента по його ID із CSV файлу

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

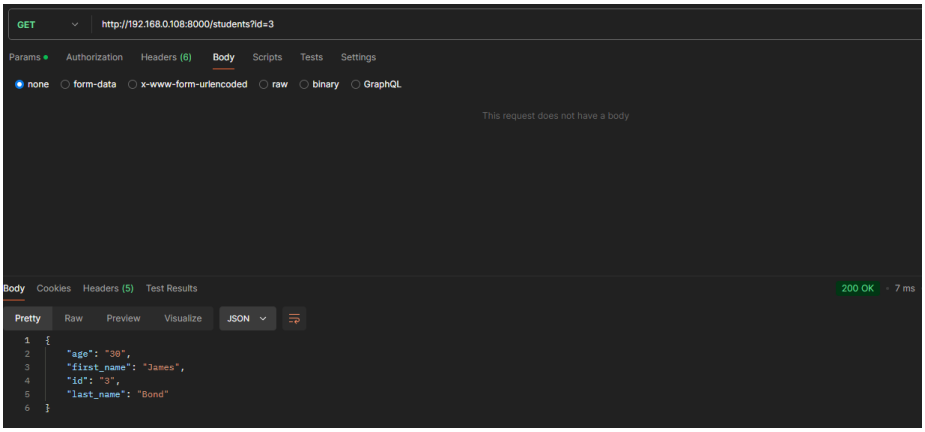
Тестування через Postman

GET

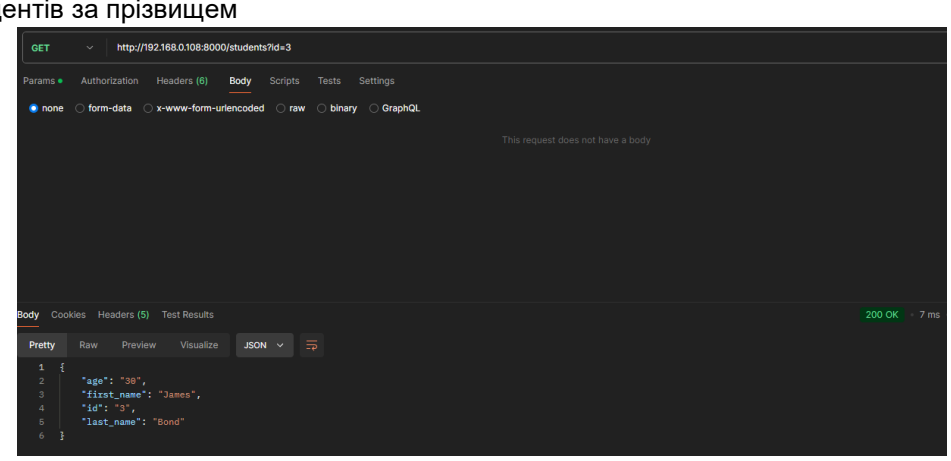
Отримання всіх студентів



Отримання студентів за ID

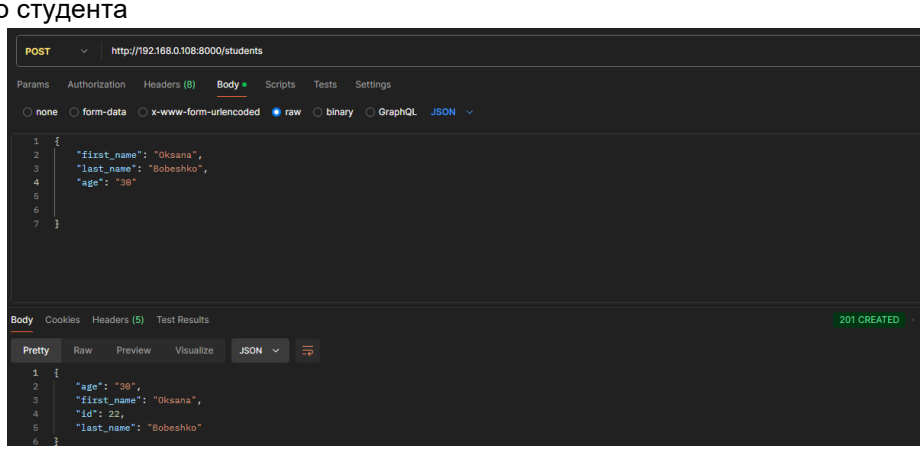


Отримання студентів за прізвищем



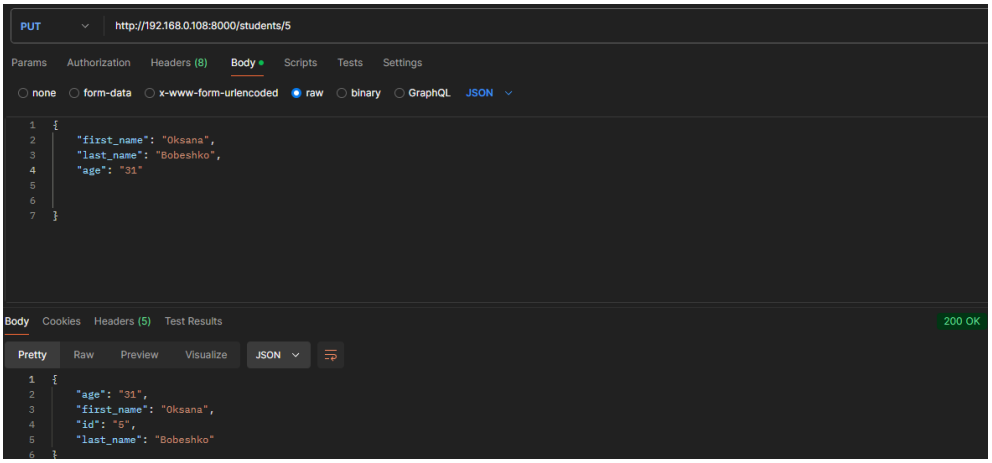
POST

Додавання нового студента



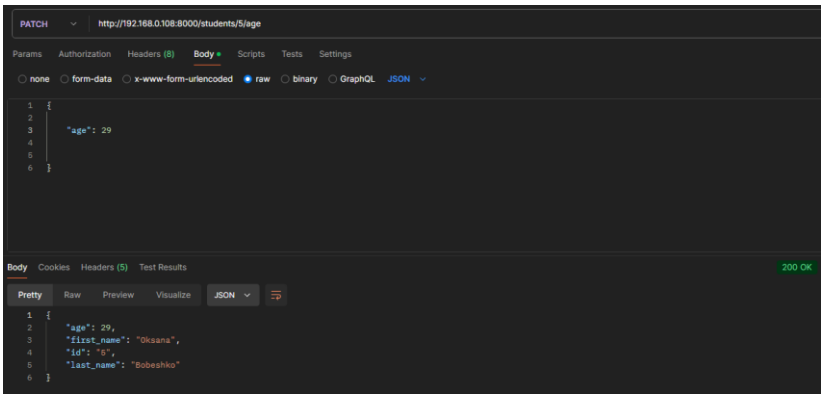
PUT

Оновлення інформації про студента



PATCH

Оновлення віку студента



DELETE

Видалення студента