

Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Лабораторна робота № 4

Тема: «Створення Application programming interface для хмарного додатку»

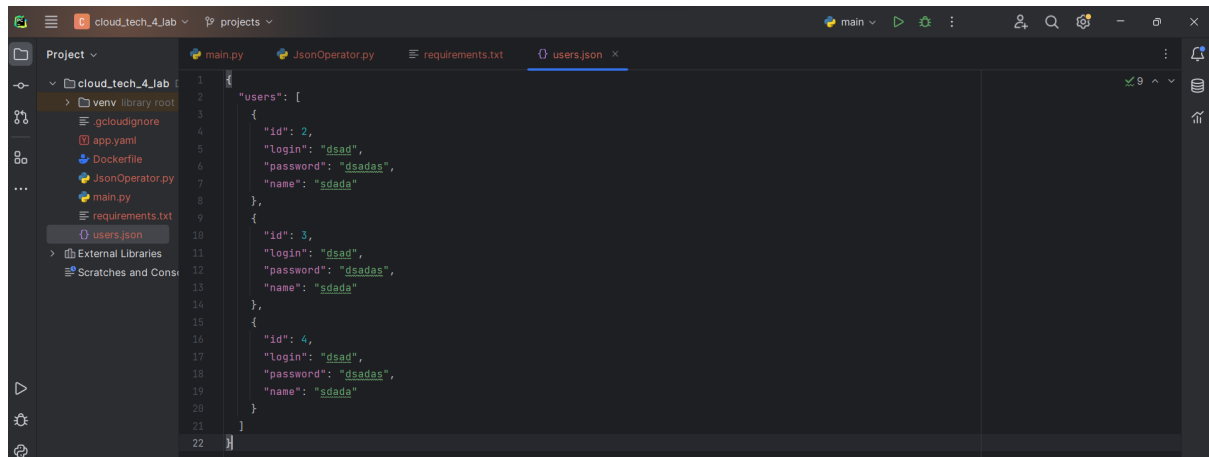
Дисципліна «Хмарні технології»

Підготував:

студент гр. ПЗ-43(1)

Мішак Максим

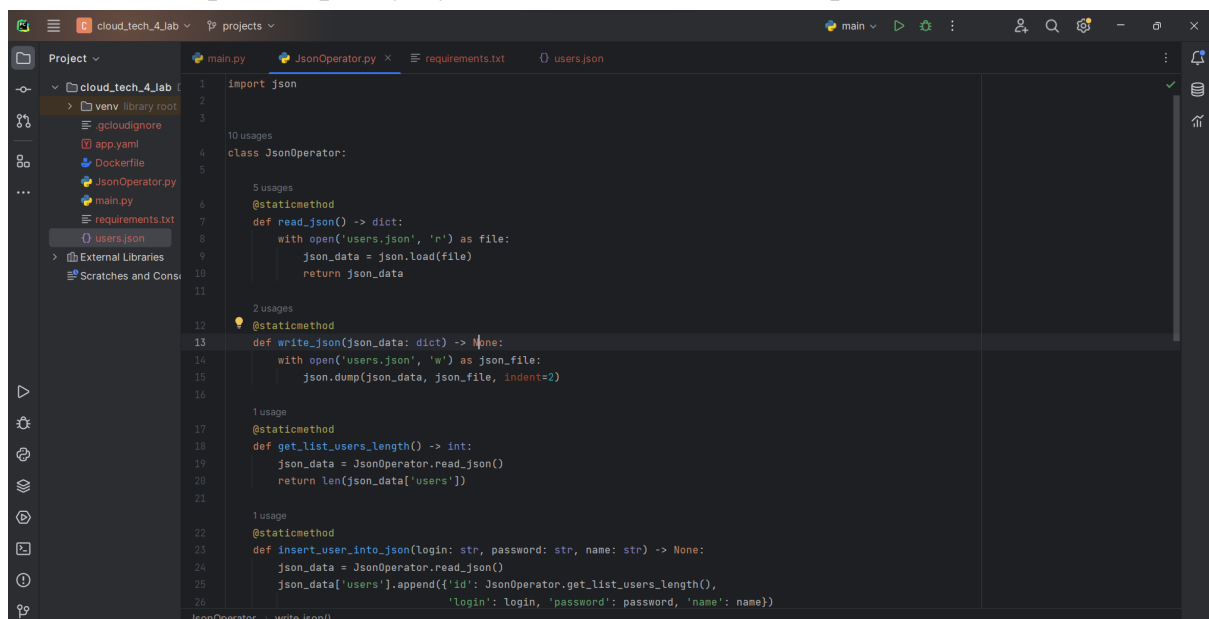
API - це програмний інтерфейс, який призначений для комунікації з логікою програми. Перевагою його використання є те, що API, як окрема структура, дозволяє реалізовувати доступ з різних клієнтів, при тому використовувати один той самий сервер для всіх джерел запитів. Для написання серверу я використовував фреймворк flask для мови програмування python. Для зберігання даних, та подальшої демонстрацію роботи додатку буду використовувати JSON файл, який буде виступа структурованим сховищем.

A screenshot of a code editor window titled 'cloud\_tech\_4\_lab'. The left sidebar shows a project tree with files like 'main.py', 'JsonOperator.py', 'requirements.txt', and 'users.json'. The main editor area displays the content of 'users.json', which is a JSON array of user objects. Each object has 'id', 'login', 'password', and 'name' fields. The data is as follows:

```
{
  "users": [
    {
      "id": 2,
      "login": "dsad",
      "password": "dsadas",
      "name": "sdada"
    },
    {
      "id": 3,
      "login": "dsad",
      "password": "dsadas",
      "name": "sdada"
    },
    {
      "id": 4,
      "login": "dsad",
      "password": "dsadas",
      "name": "sdada"
    }
  ]
}
```

Структура файлу виглядає так: по ключу “users” доступний список з користувачами. Список в собі містить множину словників, в яких зберігається інформація про користувачів: ідентифікатор, логін, пароль, ім'я.

Для обробки файлу було написано клас JsonOperator.

A screenshot of a code editor window titled 'cloud\_tech\_4\_lab'. The left sidebar shows the same project tree as the previous image. The main editor area displays the code for the 'JsonOperator' class in 'JsonOperator.py'. The code includes imports for 'json' and 'os', and defines several static methods: 'read\_json()', 'write\_json()', 'get\_list\_users\_length()', and 'insert\_user\_into\_json()'. The code is as follows:

```
import json
import os

class JsonOperator:

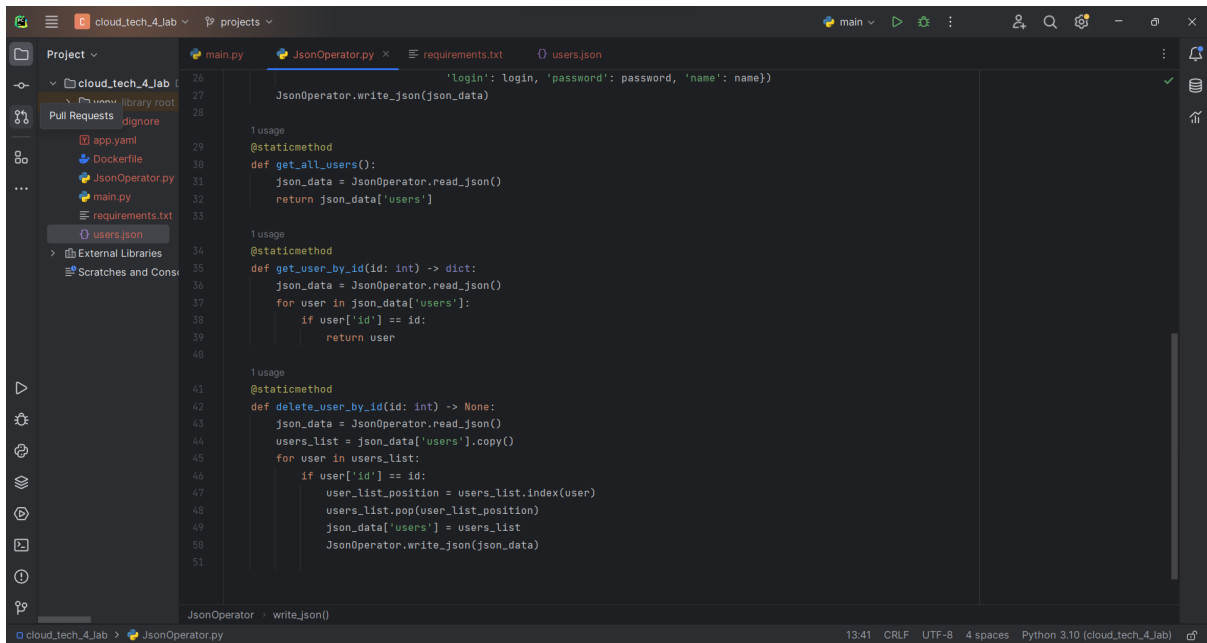
    @staticmethod
    def read_json() -> dict:
        with open('users.json', 'r') as file:
            json_data = json.load(file)
            return json_data

    @staticmethod
    def write_json(json_data: dict) -> None:
        with open('users.json', 'w') as json_file:
            json.dump(json_data, json_file, indent=2)

    @staticmethod
    def get_list_users_length() -> int:
        json_data = JsonOperator.read_json()
        return len(json_data['users'])

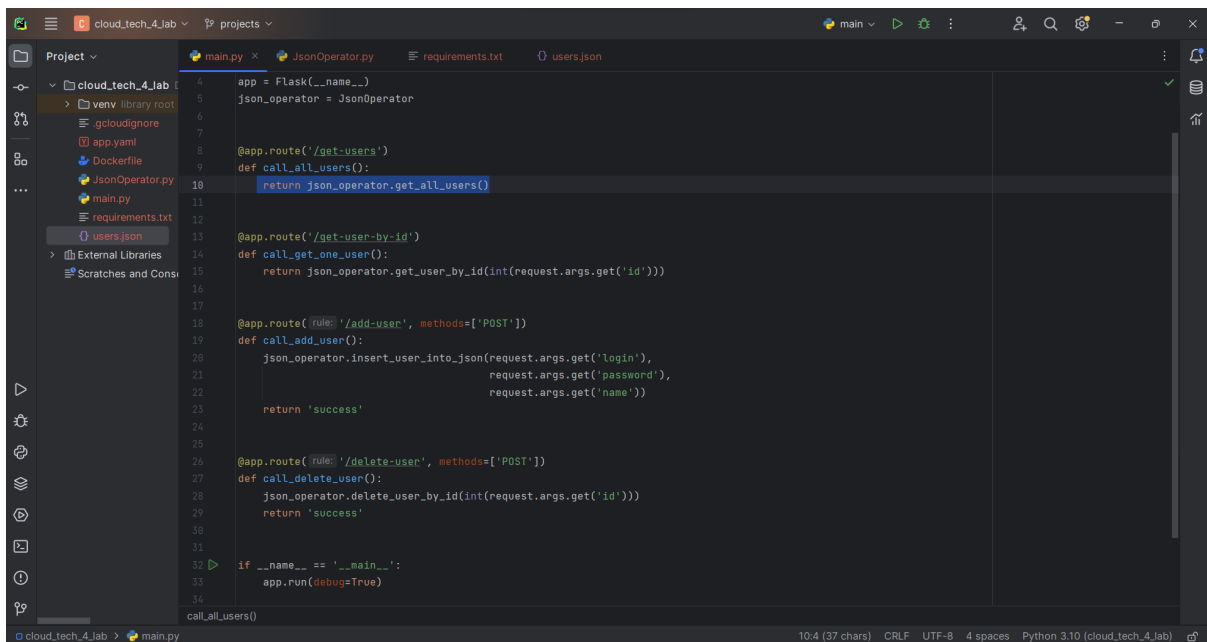
    @staticmethod
    def insert_user_into_json(login: str, password: str, name: str) -> None:
        json_data = JsonOperator.read_json()
        json_data['users'].append({'id': JsonOperator.get_list_users_length(),
                                   'login': login, 'password': password, 'name': name})

JsonOperator.write_json()
```



```
26         'login': login, 'password': password, 'name': name})
27     JsonOperator.write_json(json_data)
28
29     1 usage
30     @staticmethod
31     def get_all_users():
32         json_data = JsonOperator.read_json()
33         return json_data['users']
34
35     1 usage
36     @staticmethod
37     def get_user_by_id(id: int) -> dict:
38         json_data = JsonOperator.read_json()
39         for user in json_data['users']:
40             if user['id'] == id:
41                 return user
42
43     1 usage
44     @staticmethod
45     def delete_user_by_id(id: int) -> None:
46         json_data = JsonOperator.read_json()
47         users_list = json_data['users'].copy()
48         for user in users_list:
49             if user['id'] == id:
50                 user_list_position = users_list.index(user)
51                 users_list.pop(user_list_position)
52                 json_data['users'] = users_list
53                 JsonOperator.write_json(json_data)
```

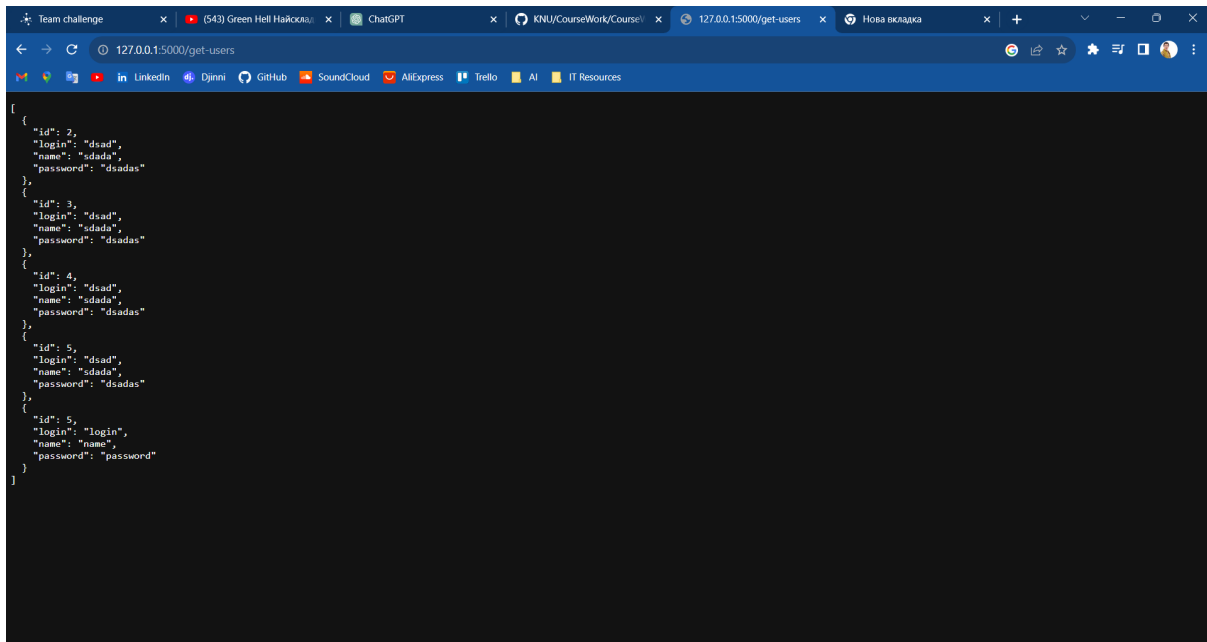
Цей клас реалізує доступ до файлу. Він дозволяє зчитувати записи з файлу, записувати у файл, також отримувати вміст файлу у вигляді словника, додавати записи, видаляти записи. Для того, щоб організувати систему роутингу на сервері, та реалізацію доступів до методів за допомогою HTTP запитів використовуються `app.route` декоратори та проксі методи для виклику методів комунікації з файлом.



```
4 app = Flask(__name__)
5 json_operator = JsonOperator
6
7
8 @app.route('/get-users')
9 def call_all_users():
10     return json_operator.get_all_users()
11
12
13 @app.route('/get-user-by-id')
14 def call_get_one_user():
15     return json_operator.get_user_by_id(int(request.args.get('id')))
16
17
18 @app.route(rule='/add-user', methods=['POST'])
19 def call_add_user():
20     json_operator.insert_user_into_json(request.args.get('login'),
21                                         request.args.get('password'),
22                                         request.args.get('name'))
23     return 'success'
24
25
26 @app.route(rule='/delete-user', methods=['POST'])
27 def call_delete_user():
28     json_operator.delete_user_by_id(int(request.args.get('id')))
29     return 'success'
30
31
32 if __name__ == '__main__':
33     app.run(debug=True)
34
35 call_all_users()
```

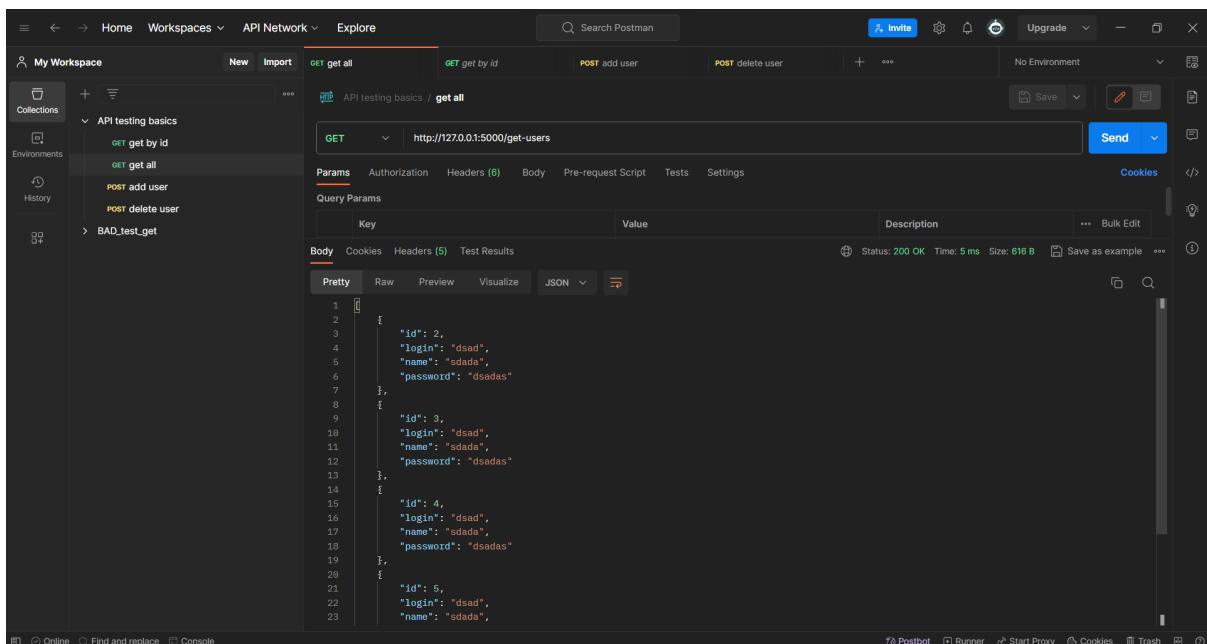
На цьому фрагменті коду видно, по якому саме принципу реалізується доступ до методів, та які ендпоінти обробляються. Після перевірки роботи класу-коннектора до JSON файлу потрібно запустити сервер та перевірити його роботу. Для цього буду використовувати влаштований у flask проксі-сервер `nginx` та програму для тестування

Postman. Спершу, перейшовши з браузера по відповідній адресі отримав список всіх користувачів.

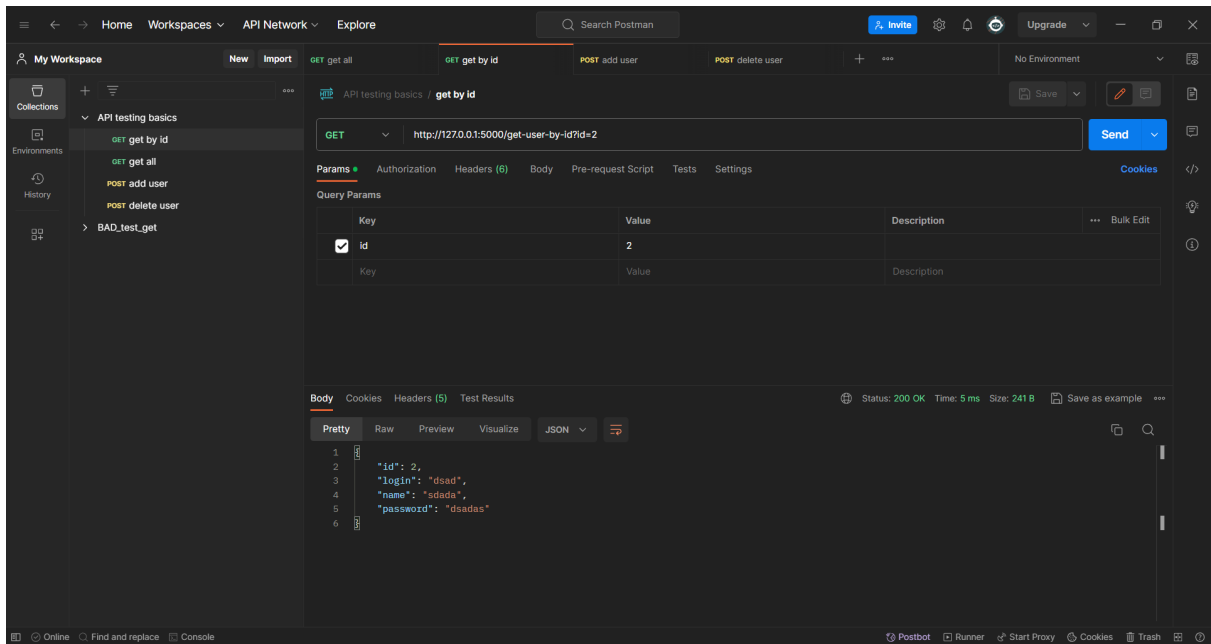


```
{
  "id": 2,
  "login": "dsad",
  "name": "sdada",
  "password": "dsadas"
},
{
  "id": 3,
  "login": "dsad",
  "name": "sdada",
  "password": "dsadas"
},
{
  "id": 4,
  "login": "dsad",
  "name": "sdada",
  "password": "dsadas"
},
{
  "id": 5,
  "login": "dsad",
  "name": "sdada",
  "password": "dsadas"
},
{
  "id": 5,
  "login": "login",
  "name": "name",
  "password": "password"
}
}
```

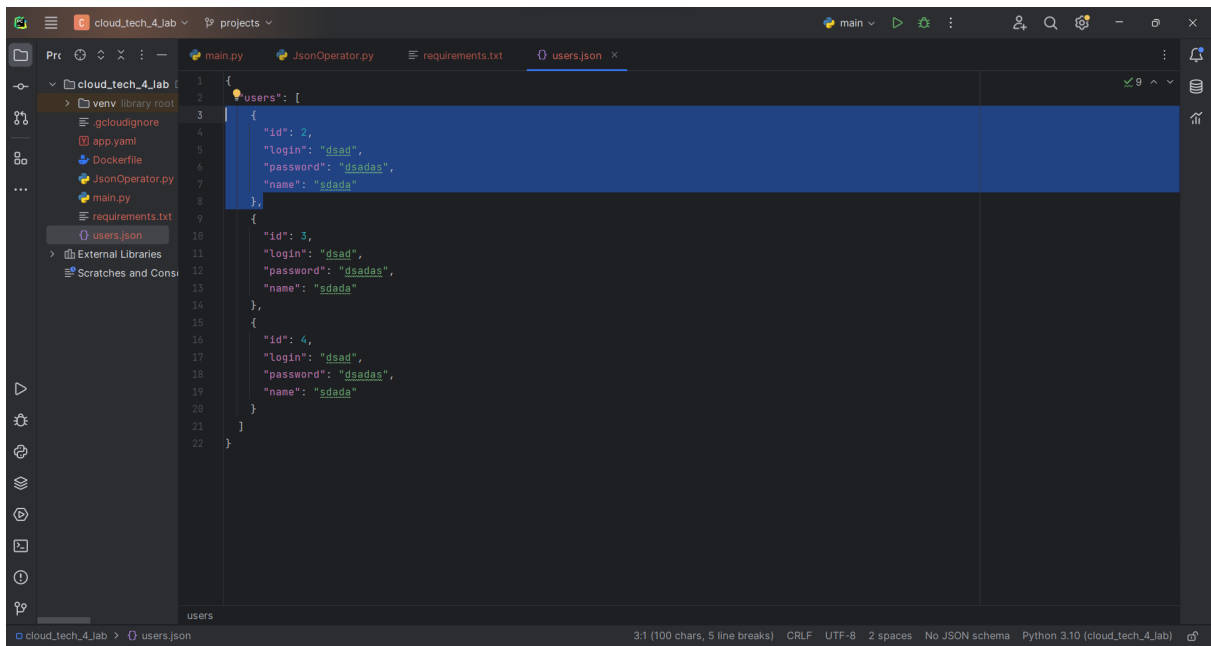
Після цього, потрібно протестувати решту напрямків. Це буде відбуватись з Postman, тому що там є зручний редактор заголовків та вмісту запитів.



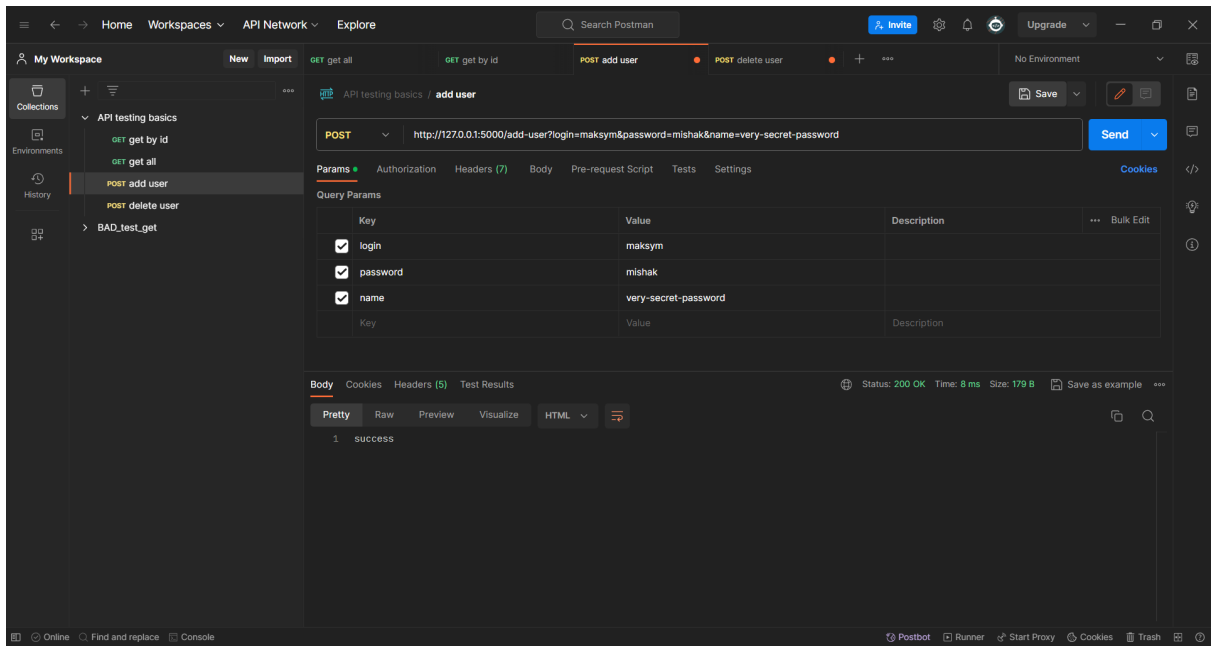
Перевірка /get-users



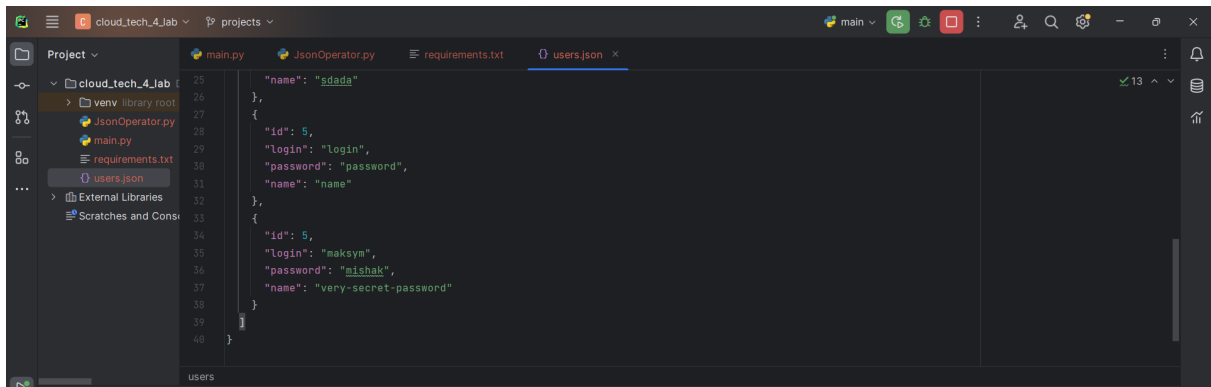
## Перевірка /get-user-by-id



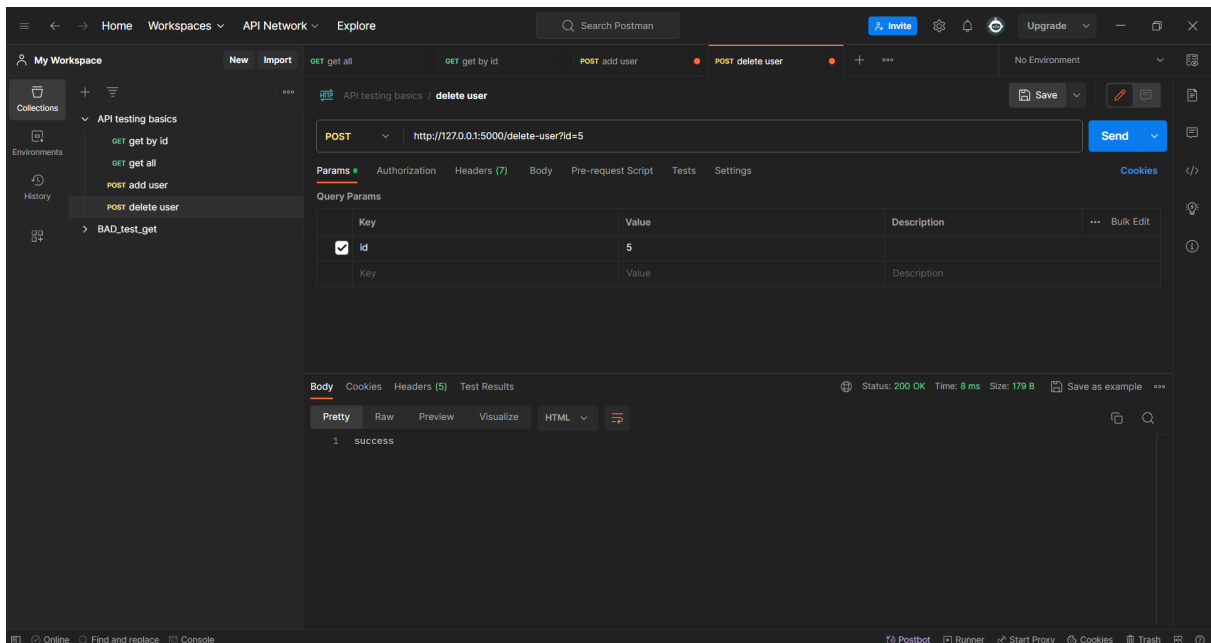
## Відповідний запис у JSON файлі



## Перевірка роботи /add-user



## Відповідний доданий файл



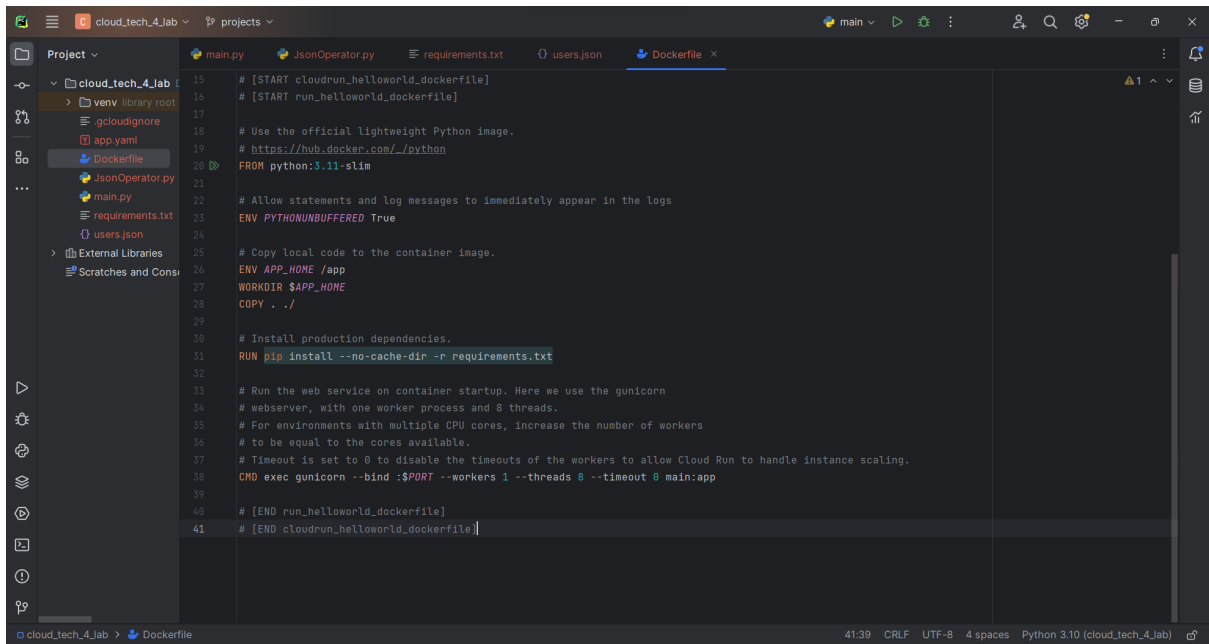
## Перевірка /delete-user

```
1 {
2   "users": [
3     {
4       "id": 2,
5       "login": "dsad",
6       "password": "dsadas",
7       "name": "sdada"
8     },
9     {
10      "id": 3,
11      "login": "dsad",
12      "password": "dsadas",
13      "name": "sdada"
14    },
15    {
16      "id": 4,
17      "login": "dsad",
18      "password": "dsadas",
19      "name": "sdada"
20    }
21  ]
22 }
```

```
127.0.0.1 - - [21/Nov/2023 23:50:28] "GET /get-users HTTP/1.1" 200 -
127.0.0.1 - - [21/Nov/2023 23:50:40] "GET /get-users HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2023 00:15:09] "GET /get-users HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2023 00:15:21] "GET /get-user-by-id?id=2 HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2023 00:16:45] "POST /add-user?login=maKsym&password=mishak&name=very-secret-password HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2023 00:20:28] "POST /delete-user?id=5 HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2023 00:20:49] "POST /delete-user?id=5 HTTP/1.1" 200 -
```

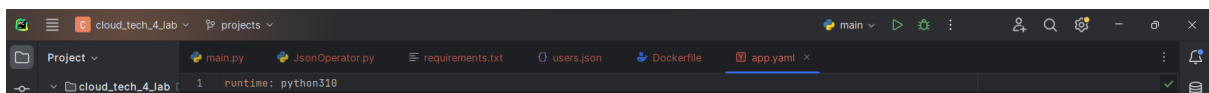
Вміст файлу після виконання запиту.

Після перевірки роботи програми на локальній машині можна приступати до розгортання цього додатку на хмарний хостинг. Для цього потрібно написати відповідну архітектуру для хмари, завдяки якій проект можна буде запуснути та використовувати. Для запуску проекту flask на базі app engine потрібно декілька складових: встановлений фреймворк та всі необхідні для нього доповнення, встановлені решта фреймворків, які використовувались в проекті, зазначення версії мови програмування та сторонніх фреймворків, налаштування Docker контейнера для використання проксі-сервера gunicorn. Для цього було використані відповідні розширення конфігураційних файлів.



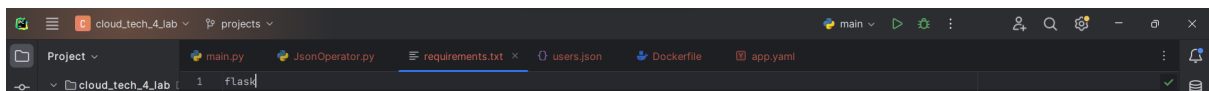
```
15 # [START cloudrun_helloworld_dockerfile]
16 # [START_run_helloworld_dockerfile]
17
18 # Use the official lightweight Python image.
19 # https://hub.docker.com/_/python
20 FROM python:3.11-slim
21
22 # Allow statements and log messages to immediately appear in the logs
23 ENV PYTHONUNBUFFERED True
24
25 # Copy local code to the container image.
26 ENV APP_HOME /app
27 WORKDIR $APP_HOME
28 COPY . ./
29
30 # Install production dependencies.
31 RUN pip install --no-cache-dir -r requirements.txt
32
33 # Run the web service on container startup. Here we use the gunicorn
34 # webserver, with one worker process and 8 threads.
35 # For environments with multiple CPU cores, increase the number of workers
36 # to be equal to the cores available.
37 # Timeout is set to 0 to disable the timeouts of the workers to allow Cloud Run to handle instance scaling.
38 CMD exec gunicorn --bind :$PORT --workers 1 --threads 8 --timeout 0 main:app
39
40 # [END run_helloworld_dockerfile]
41 # [END cloudrun_helloworld_dockerfile]
```

Docker файл для конфігурації контейнера для використання gunicorn.



```
1 runtime: python310
```

YAML файл для вказівки параметра runtime, який відповідає за те, яку версію мови програмування буде використовувати середовище

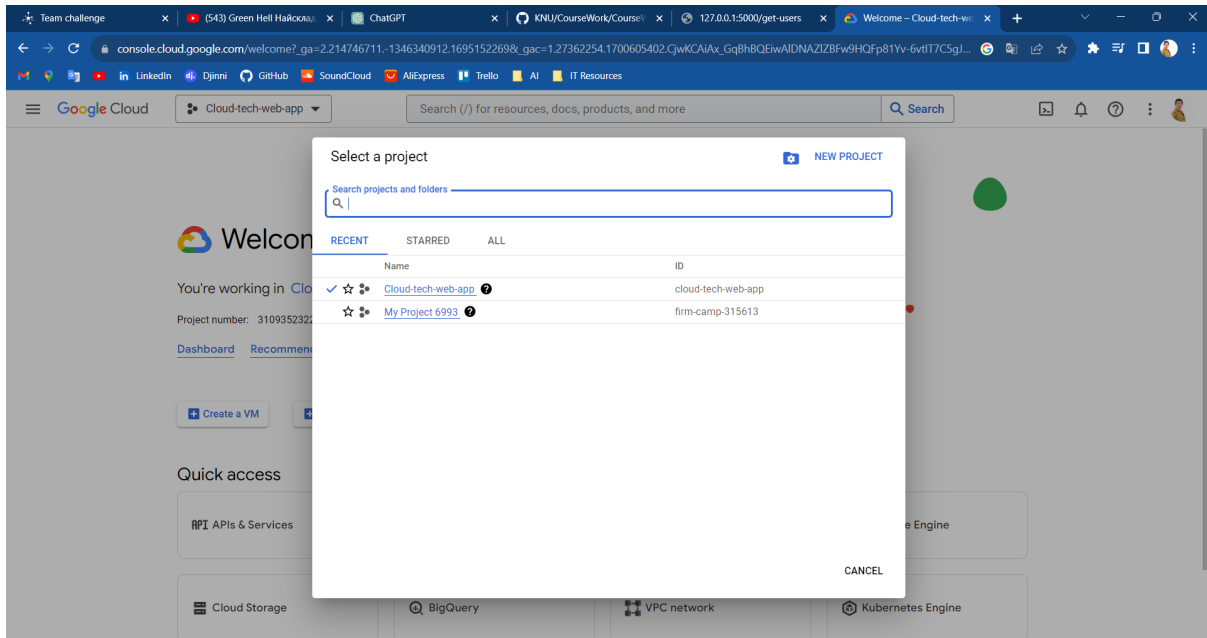


```
1 flask
```

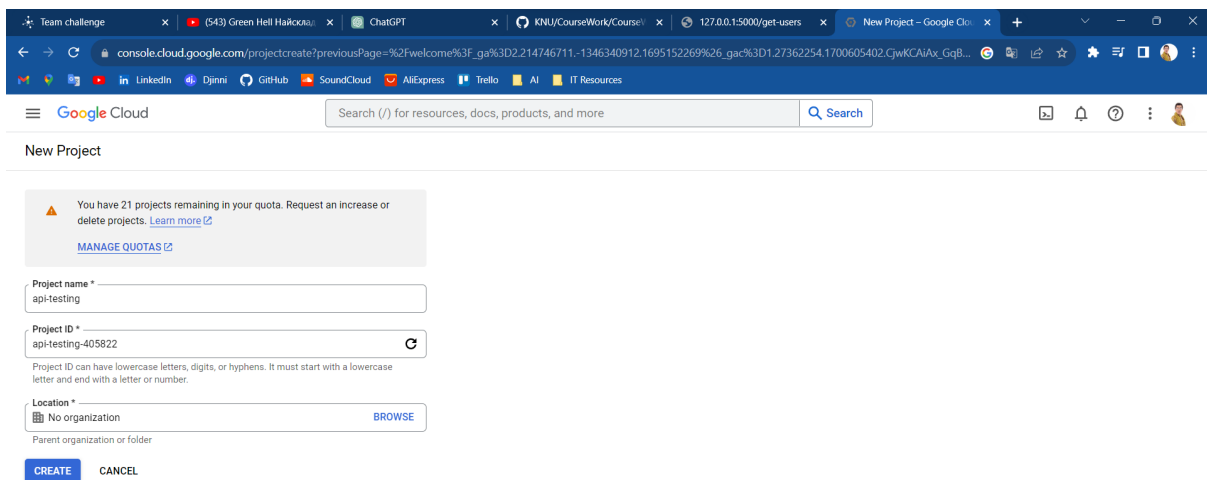
requirements.txt файл, який використовується для вказання бібліотек та сторонніх імпортованих засобів



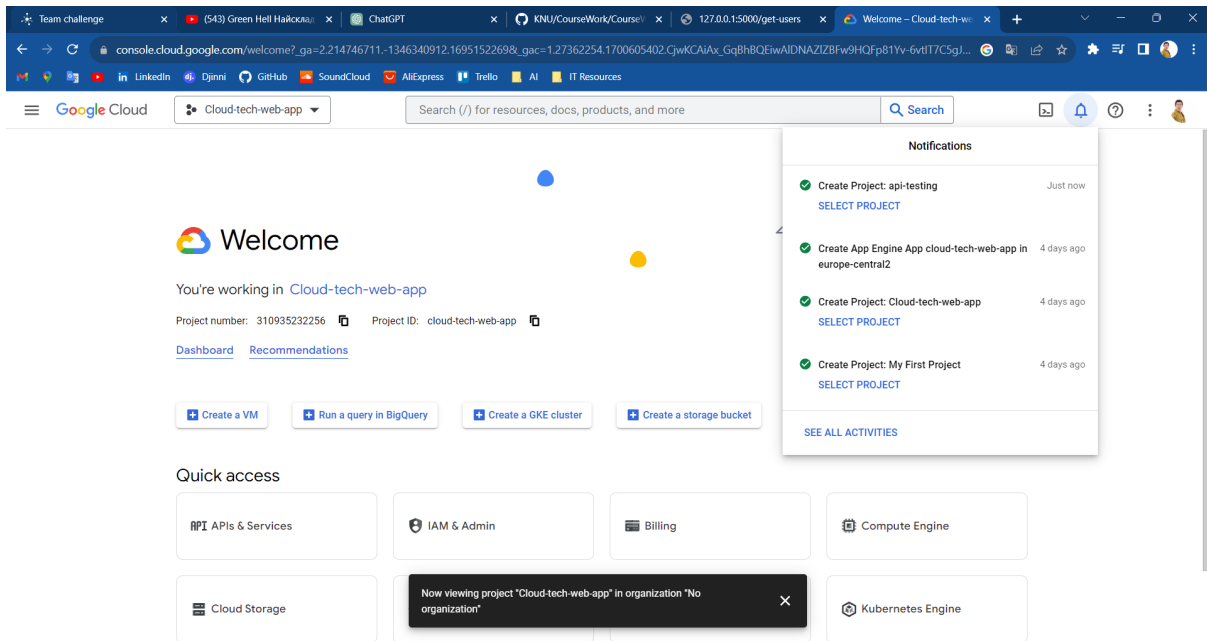
Після написання інфраструктури потрібно створити Google Cloud проект, який буде містити елемент App Engine та на якому буде розміщено додаток. Для цього потрібно перейти на Google Cloud та натиснути кнопку New Project



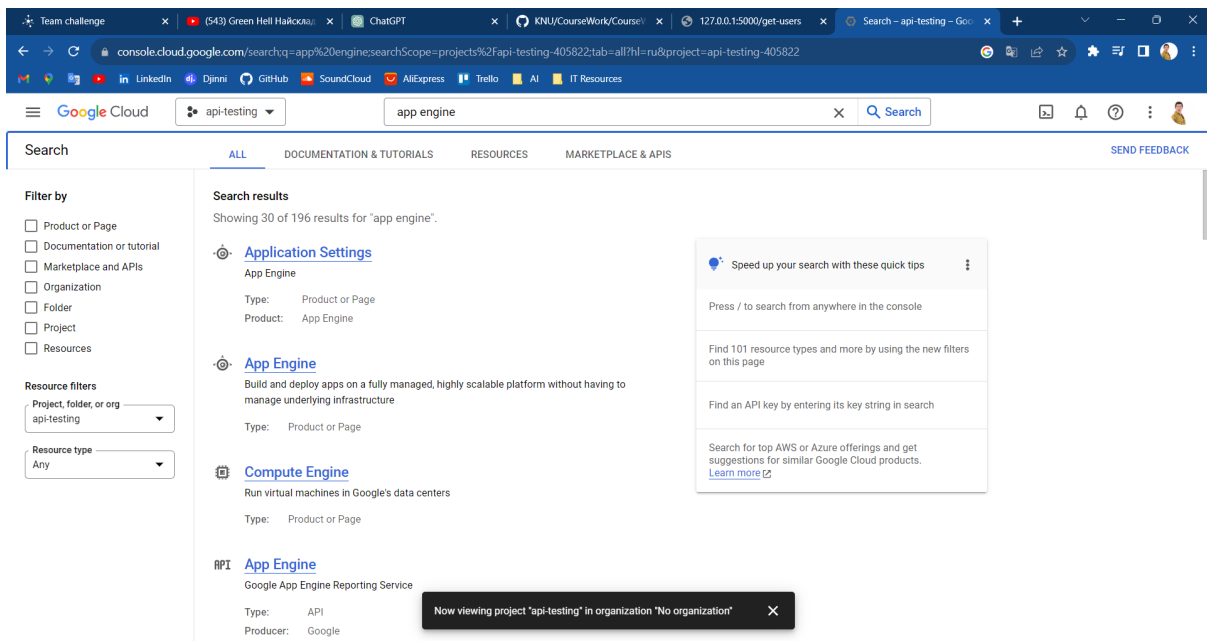
Після чого вказати назву проекту



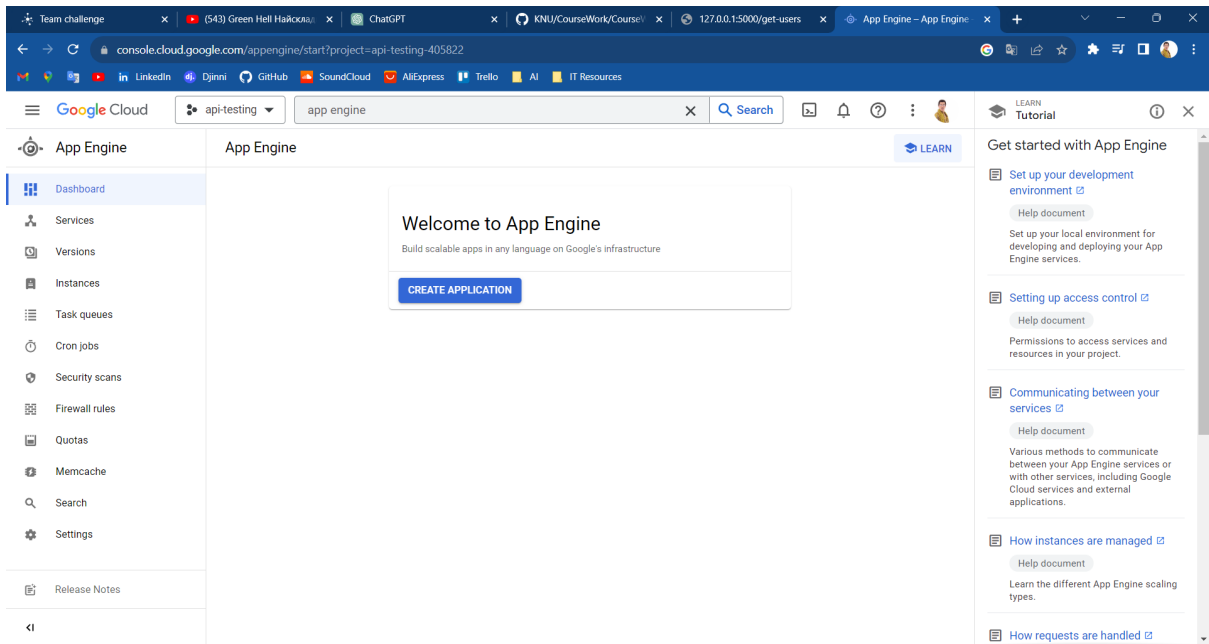
Тепер можна створювати проект та чекати розгортання



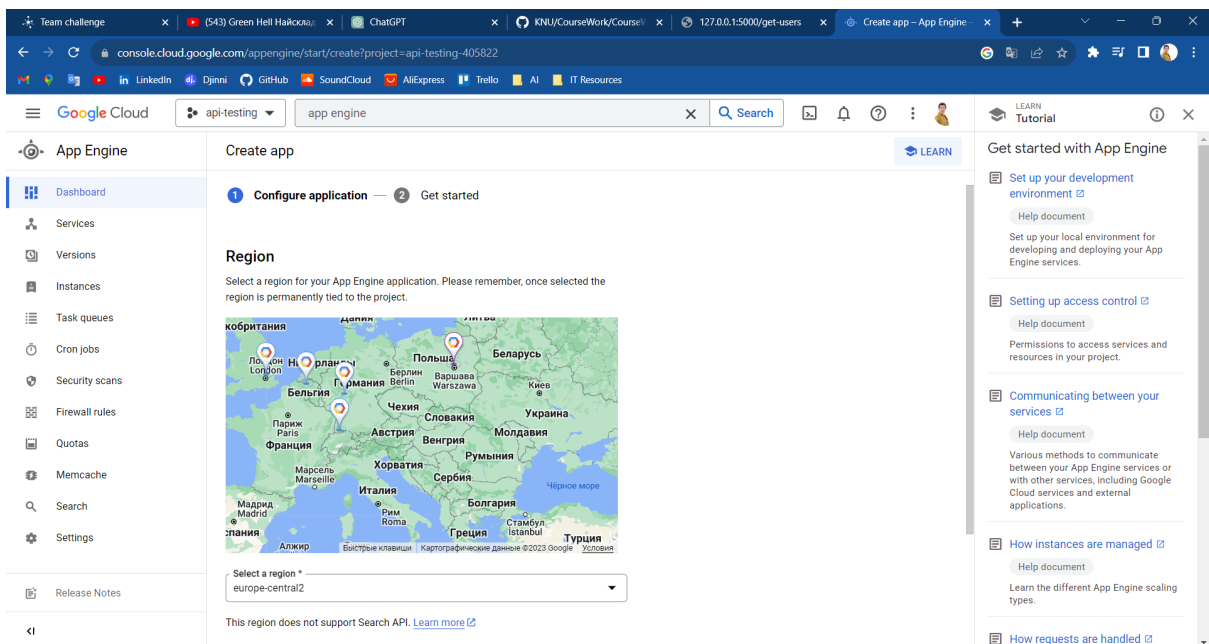
Коли проект буде розгорнуто, потрібно додати елемент App Engine до його елементів, для цього потрібно в пошуку вписати App Engine



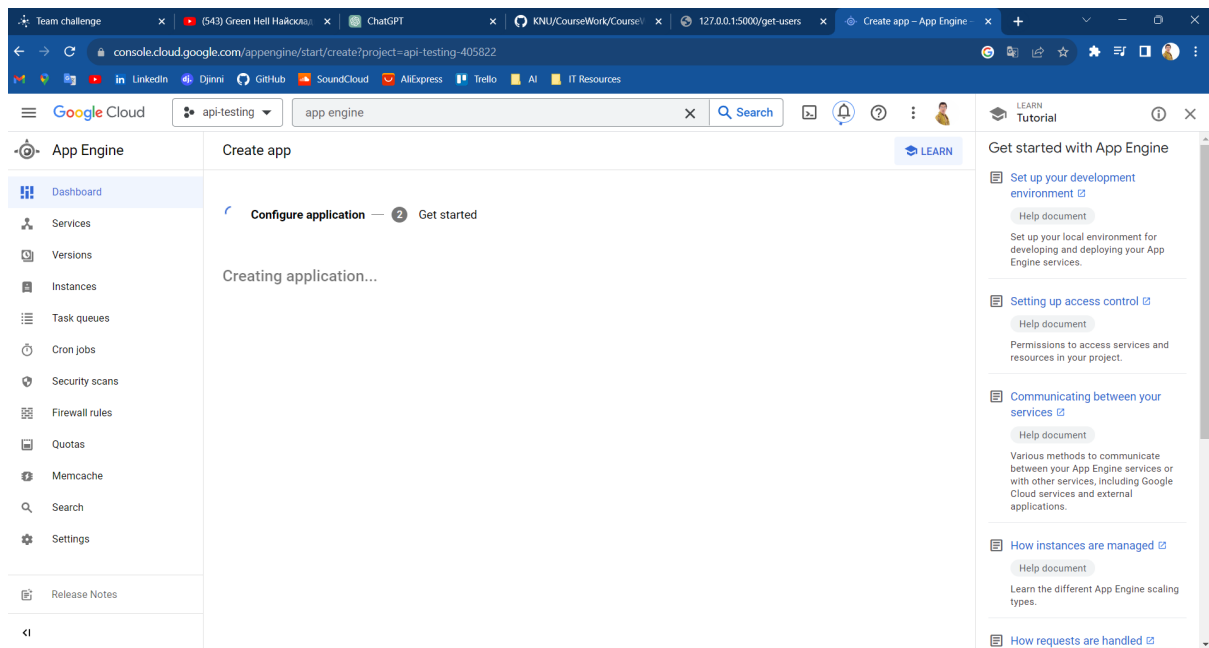
Вибираємо відповідний елемент, та переходимо по його посиланню



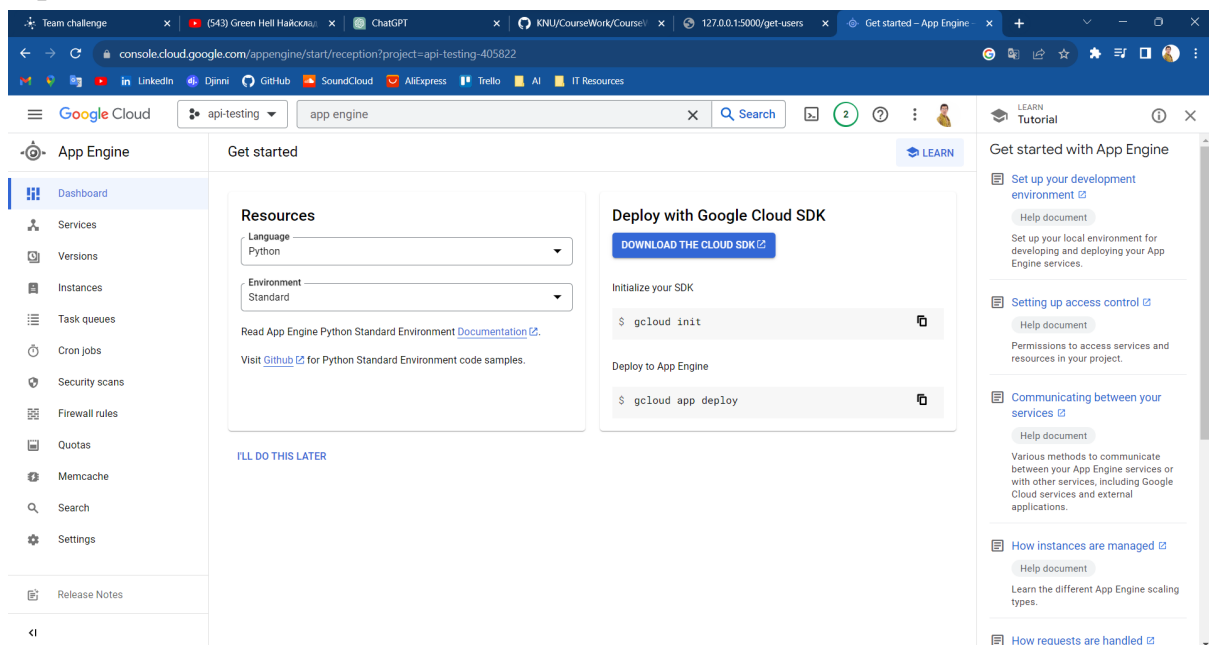
## Після чого натиснути Create application



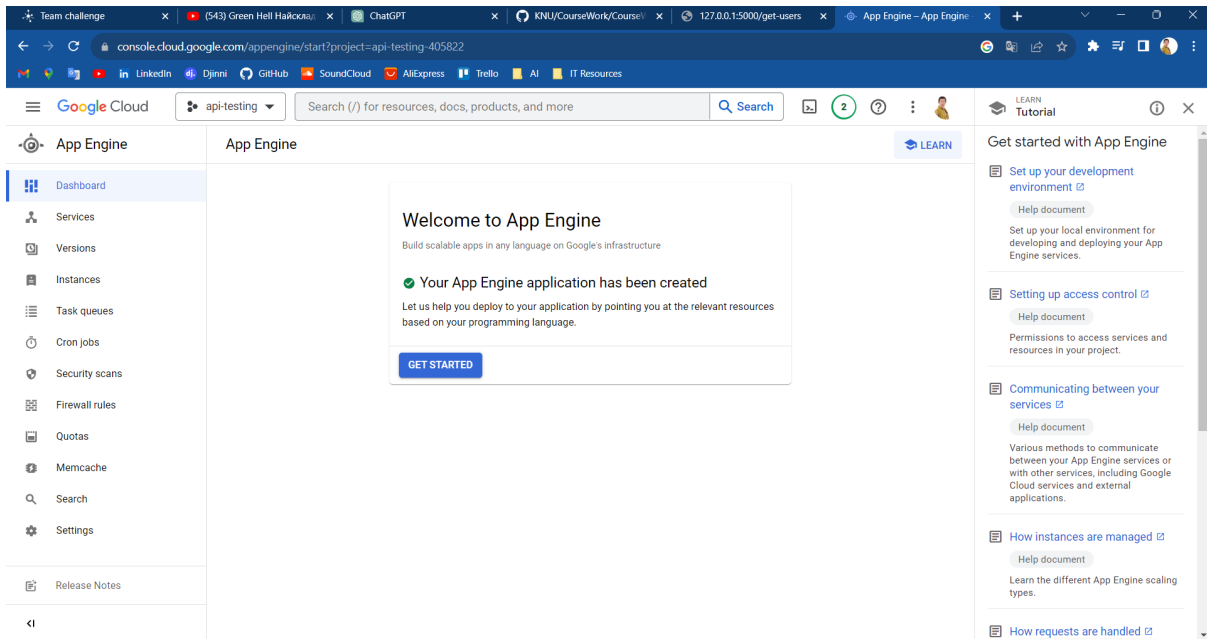
Вибираємо фізичне розташування дата центра, для свого проекту та натискаємо кнопку Next



Чекаємо застосування налаштувань та переходимо до налаштування середовища



В налаштуваннях середовища потрібно вказати мову програмування, яка буде використовуватись, та тип середовища. Після застосування налаштувань буде відображатись такий екран:



Тепер потрібно завантажити проект на хостинг. Для цього буду використовувати `gcloud` cli. Переходимо в робочу директорію проекту та відкриваємо термінал

```
(venv) PS D:\projects\KNU\cloud_tech\cloud_tech_4_lab> gcloud projects list
PROJECT_ID      NAME            PROJECT_NUMBER
api-testing-405822  api-testing      11007307403
cloud-tech-web-app  Cloud-tech-web-app  318935232256
firm-camp-315613    My Project 6993    754111572351
fleet-parser-405519  My First Project  677964875281
pricepitstop        pricepitstop      823253240876
(venv) PS D:\projects\KNU\cloud_tech\cloud_tech_4_lab>
```

Для вибору проекту потрібно знати його `project-id`, для цього можна вивести список всіх проектів на акаунті. Після вибору проекту потрібно скопіювати його ідентифікатор, та задати його для змінної `project` в інструменті Google Cloud.

```
(venv) PS D:\projects\KNU\cloud_tech\cloud_tech_4_lab> gcloud config set project api-testing-405822
Updated property [core/project].
(venv) PS D:\projects\KNU\cloud_tech\cloud_tech_4_lab>
```

Після цього потрібно ініціалізувати робочу директорію для `gcloud`

```
(venv) PS D:\projects\KNU\cloud_tech\cloud_tech_4_lab> gcloud init
Welcome! This command will take you through the configuration of gcloud.

Settings from your current configuration [flask-app] are:
accessibility:
  screen_reader: 'False'
core:
  account: maxteroidyandex@gmail.com
  disable_usage_reporting: 'True'
  project: api-testing-405822

Pick configuration to use:
[1] Re-initialize this configuration [flask-app] with new settings
[2] Create a new configuration
[3] Switch to and re-initialize existing configuration: [default]
Please enter your numeric choice: 
```

Потрібно зазначити, в якому саме режимі буде працювати проект, в мене є налаштований попередньо профіль параметрів, я заставляю його. Хоча за потреби можна зазначити новий. Після цього потрібно пройти процес аутентифікації, якщо цього не було зроблено заздалегідь, або використати вже аутентифікований обліковий запис.

```
(venv) PS D:\projects\KNU\cloud_tech\cloud_tech_4_lab> gcloud init
Welcome! This command will take you through the configuration of gcloud.
```

Settings from your current configuration [flask-app] are:

```
accessibility:
  screen_reader: 'False'
core:
  account: maxteroidyandex@gmail.com
  disable_usage_reporting: 'True'
  project: api-testing-405822
```

Pick configuration to use:

- [1] Re-initialize this configuration [flask-app] with new settings
- [2] Create a new configuration
- [3] Switch to and re-initialize existing configuration: [default]

Please enter your numeric choice: 1

Your current configuration has been set to: [flask-app]

You can skip diagnostics next time by using the following flag:  
gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.  
Checking network connection...done.  
Reachability Check passed.  
Network diagnostic passed (1/1 checks passed).

Network diagnostic passed (1/1 checks passed).

Choose the account you would like to use to perform operations for this configuration:

- [1] maxteroidyandex@gmail.com
- [2] Log in with a new account

Please enter your numeric choice: 1

You are logged in as: [maxteroidyandex@gmail.com].

Pick cloud project to use:

- [1] api-testing-405822
- [2] cloud-tech-web-app
- [3] firm-camp-315613
- [4] fleet-parser-405519
- [5] pricepitstop

able to do this for you the next time you run it, make sure the  
Compute Engine API is enabled for your project on the  
<https://console.developers.google.com/apis> page.

Your Google Cloud SDK is configured and ready to use!

\* Commands that require authentication will use maxteroidyandex@gmail.com by default  
\* Commands will reference project 'api-testing-405822' by default  
Run 'gcloud help config' to learn how to change individual settings

This gcloud configuration is called [flask-app]. You can create additional configurations if you work with multiple accounts and/or projects.  
Run 'gcloud topic configurations' to learn more.

## Тепер безпосередньо завантажити проект на хостинг

```
(venv) PS D:\projects\KNU\cloud_tech\cloud_tech_4_lab> gcloud app deploy
Services to deploy:

descriptor:      [D:\projects\KNU\cloud_tech\cloud_tech_4_lab\app.yaml]
source:          [D:\projects\KNU\cloud_tech\cloud_tech_4_lab]
target project:  [api-testing-405822]
target service:  [default]
target version:  [20231122t003559]
target url:      [https://api-testing-405822.lm.r.appspot.com]
target service account: [api-testing-405822@appspot.gserviceaccount.com]
```

Do you want to continue (Y/n)? **Y**

Beginning deployment of service [default]...  
Created .gcloudignore file. See 'gcloud topic gcloudignore' for details.

```
== Uploading 918 files to Google Cloud Storage ==
```

File upload done.  
Updating service [default]...done.  
Setting traffic split for service [default]...done.  
Deployed service [default] to [<https://api-testing-405822.lm.r.appspot.com>]

You can stream logs from the command line by running:  
\$ gcloud app logs tail -s default

To view your application in the web browser run:  
\$ gcloud app browse

```
(venv) PS D:\projects\KNU\cloud_tech\cloud_tech_4_lab>
```

Після завершення процесу завантаження отримаємо посилання, за яким можна перейти на розгорнутий проект. При переході на відповідну кінцеву точку бачимо результат:

