Лабораторна робота №4

Золотов Іван ФБ-31мп

Посилання на GitHub

https://github.com/SideOctopus/Lab_4

Переписую код з минулих робіт для Messaging queue

Клієнт:

```python
import requests

def send_post(message):
    # Функція для відправки POST-запиту з повідомленням на сервер
    response = requests.post('http://localhost:5000/data', data=message.encode('utf-8'))
    print(f'Response from server: {response.text}') # Виведення відповіді сервера

def send_get():
    # Функція для відправки GET-запиту для отримання повідомлень від сервера
    response = requests.get('http://localhost:5000/data')
    print(f'Response from server: {response.text}') # Виведення відповіді сервера


def main_menu():
    while True:
        print("\n*** Client Menu ***")
        print("1. Send POST request with a message")
        print("2. Send GET request to retrieve messages")
        print("3. Exit")
        choice = input("Enter your choice (1-3): ")

        if choice == '1':
            message = input("Enter the message to send: ")
            send_post(message)
        elif choice == '2':
            send_get()
        elif choice == '3':
            print("Exiting the client...")
            break
        else:
            print("Invalid choice. Please choose from 1 to 3.")

if __name__ == '__main__':
    main_menu()
```

facadeService:

```python
from flask import Flask, request, jsonify
import random
import requests
import hazelcast

app = Flask(__name__)

# Підключення до кластера Hazelcast
hz = hazelcast.HazelcastClient(cluster_name="dev", cluster_members=["127.0.0.1:5701"])
# Отримання черги з повідомленнями з Hazelcast
messages_queue = hz.get_queue("queue").blocking()

@app.route('/data', methods=['GET', 'POST'])
def handle_data():
    # Випадковий вибір портів для логування та повідомлень
    logging_port = random.randint(5001, 5003)
    message_port = random.randint(5005, 5006)

    if request.method == 'GET':
        # Відправка GET-запитів до сервісів логування та повідомлень
        response_logging = requests.get(f'http://127.0.0.1:{logging_port}/data', timeout=5)
        response_logging.raise_for_status() # Перевірка статусу відповіді
        response_message = requests.get(f'http://127.0.0.1:{message_port}/data').text
        return jsonify({'Message data': response_message, 'Log data': response_logging.text})
    elif request.method == 'POST':
        # Обробка POST-запиту з отриманим повідомленням
        message = request.get_data().decode('utf-8')
        messages_queue.offer(message) # Додавання повідомлення до черги в Hazelcast
        log_data = {"msg": message}
        print(f'Sending log to logging-service on port {logging_port}')
        response = requests.post(f'http://127.0.0.1:{logging_port}/data', json=log_data)
        response.raise_for_status() # Перевірка статусу відповіді
        return f'Message sent to queue and logged: {message}'

if __name__ == '__main__':
    print('Starting facade service on port 5000')
    app.run(debug=True, port=5000)
```

loggingService:

```python
from flask import Flask, request, jsonify
import argparse

app = Flask(__name__)
logs = []

@app.route('/data', methods=['GET', 'POST'])
def data():
    if request.method == 'GET':
        # Обробка GET-запиту: повертає усі логи у форматі JSON
        return jsonify(logs)
    elif request.method == 'POST':
        # Обробка POST-запиту: отримання логу з тіла запиту, запис до списку логів і виведення у консоль
        log = request.json['msg']
        logs.append(log)
        print(f'Received message via POST: {log}')  # Вивід отриманого повідомлення у консоль
        return f'Log recorded: {log}', 200  # Підтвердження запису логу з кодом статусу 200

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("--port", type=int, required=True)
    args = parser.parse_args()

    print(f'Starting logging service on port {args.port}')  # Виведення інформації про запуск сервісу логування
    app.run(debug=True, port=args.port)

```

messageService

```python
from flask import Flask, jsonify, request
import hazelcast
import argparse
import threading

app = Flask(__name__)
message_list = []

def queue_event():
    while True:
        item = messages_queue.take()
        message_list.append(item)
        print("Отримано повідомлення з черги: ", str(item))

@app.route('/data', methods=['GET', 'POST'])
def data():
    if request.method == 'GET':
        # Обробка GET-запиту: повертає всі отримані повідомлення у вигляді рядка через новий рядок
        return '\n'.join(message_list)
    elif request.method == 'POST':
        # Обробка POST-запиту: отримання повідомлення з JSON тіла запиту, додавання його до списку повідомлень і виведення у консоль підтвердження отримання
        msg = request.json['msg']
        message_list.append(msg)
        print(f'Отримано повідомлення через POST: {msg}')
        return f'Повідомлення отримано: {msg}', 200

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("--port", type=int, required=True)
    args = parser.parse_args()

    # Підключення до кластера Hazelcast та отримання блокуючої черги повідомлень
    hz = hazelcast.HazelcastClient(cluster_name="dev", cluster_members=["127.0.0.1:5701"])
    messages_queue = hz.get_queue("queue").blocking()

    # Запуск окремого потоку для обробки подій черги
    event_thread = threading.Thread(target=queue_event)
    event_thread.start()

    print(f'Запуск сервісу повідомлень на порту {args.port}')
    app.run(debug=True, port=args.port)

```

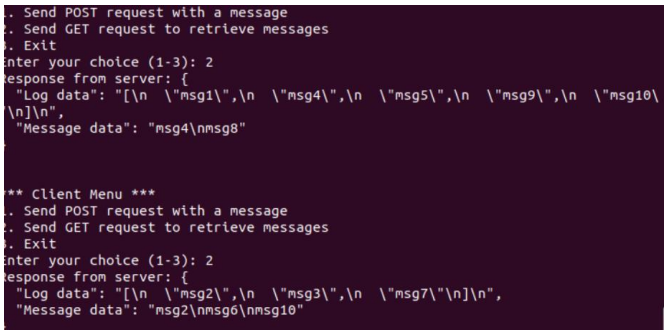
Далі я запустив 2 messageService, facadeService, 3 loggingService та один клієнт, записав 10 повідомлень та перевірив розподіл у логах та messageService

```
*** Client Menu ***
1. Send POST request with a message
2. Send GET request to retrieve messages
3. Exit
Enter your choice (1-3): 1
Enter the message to send: msg1
Response from server: Message sent to queue: msg1

*** Client Menu ***
1. Send POST request with a message
2. Send GET request to retrieve messages
3. Exit
Enter your choice (1-3): 1
Enter the message to send: msg2
Response from server: Message sent to queue: msg2

*** Client Menu ***
1. Send POST request with a message
2. Send GET request to retrieve messages
3. Exit
Enter your choice (1-3): 1
Enter the message to send: msg3
```

Після запуску, виконав декілька запитів get та отримав відмінності в messagingService:



https://github.com/SideOctopus/Lab_4