



МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»  
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ  
КАФЕДРА ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

## **Проектування розподілених систем**

### **Лабораторна робота №3**

#### **Мікросервиси з використанням Hazelcast** **Distributed Map**

Перевірив:  
Родіонов А. М.

Виконав:  
студент I курсу  
групи ФБ-41мп  
Сахній Н. Р.

Київ 2025

**Мета роботи:** Це завдання базується на основі попередніх і є їх розвитком.

Таким чином, щоби вдосконалити роботу **logging-service**, необхідно додати:

- підтримку “**Hazelcast Distributed Map**” у якості сховища повідомлень.
- можливість запускати одночасно декілька копій **logging-service**.
- механізм, за яким **facade-service** випадковим чином зможе обирати до якої копії **logging-service** звертатись для запису та читання повідомлень.

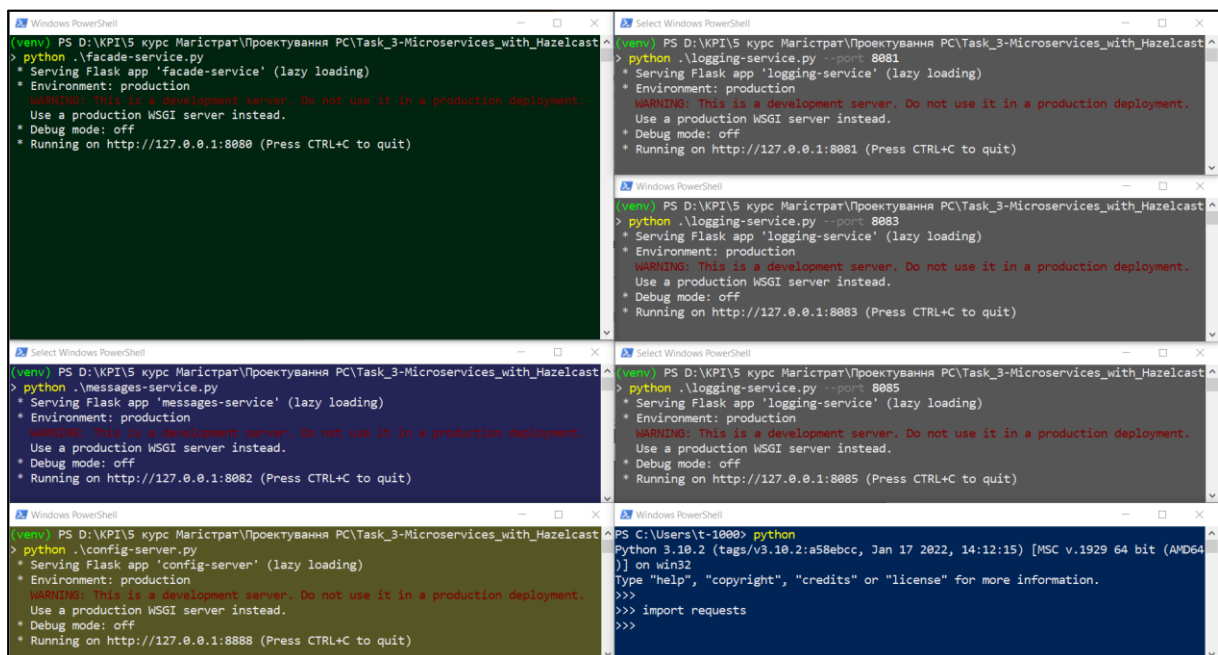
**Архітектура** складається з трьох мікросервісів, реалізованих на мові **Python**:

- **facade-service** – обробляє POST/GET-запити надіслані клієнтом.
- **logging-service** – зберігає у своїй пам’яті всі повідомлення із їхніми унікальними ідентифікаторами та надає до них доступ для їх перегляду.
- **messages-service** – поки заглушка, що повертає статичне повідомлення.
- **config-server** – статично надає інформацію про конфігурації системи.

Посилання на GitHub з проектом, що містить вихідні коди трьох мікросервісів:

[https://github.com/sazan24/KPI/tree/main/Master's%20degree/Distributed%20Systems%20Design/Task 3-Microservices with Hazelcast/](https://github.com/sazan24/KPI/tree/main/Master's%20degree/Distributed%20Systems%20Design/Task%203-Microservices%20with%20Hazelcast/)

**Запуск сервісів на портах: фасад x1 (8080), меседж x1 (8082), логгін x3 (8081 8083, 8085) та менеджер конфігураційних налаштувань (8888):**



```
(venv) PS D:\KPI\5 курс Марієттар\Проектування PC\Task_3-Microservices_with_Hazelcast> python .\facade-service.py
* Serving Flask app 'facade-service' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8080 (Press CTRL+C to quit)

(venv) PS D:\KPI\5 курс Марієттар\Проектування PC\Task_3-Microservices_with_Hazelcast> python .\logging-service.py --port 8081
* Serving Flask app 'logging-service' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8081 (Press CTRL+C to quit)

(venv) PS D:\KPI\5 курс Марієттар\Проектування PC\Task_3-Microservices_with_Hazelcast> python .\logging-service.py --port 8083
* Serving Flask app 'logging-service' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8083 (Press CTRL+C to quit)

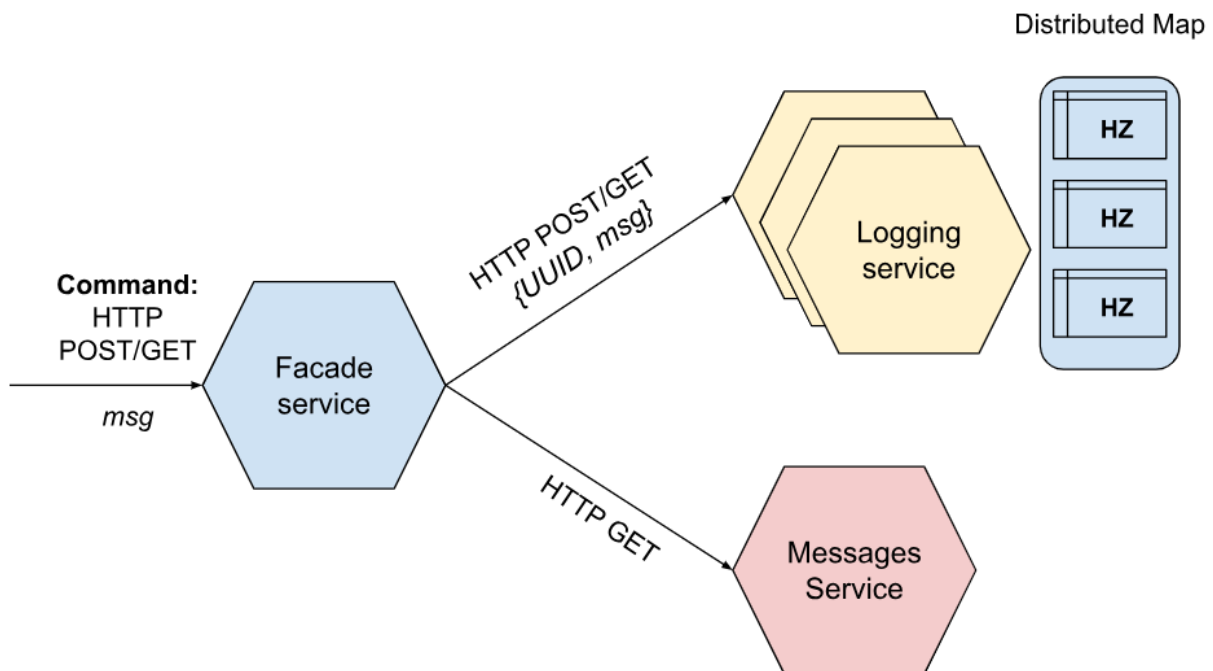
(venv) PS D:\KPI\5 курс Марієттар\Проектування PC\Task_3-Microservices_with_Hazelcast> python .\messages-service.py
* Serving Flask app 'messages-service' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8082 (Press CTRL+C to quit)

(venv) PS D:\KPI\5 курс Марієттар\Проектування PC\Task_3-Microservices_with_Hazelcast> python .\config-server.py
* Serving Flask app 'config-server' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8888 (Press CTRL+C to quit)

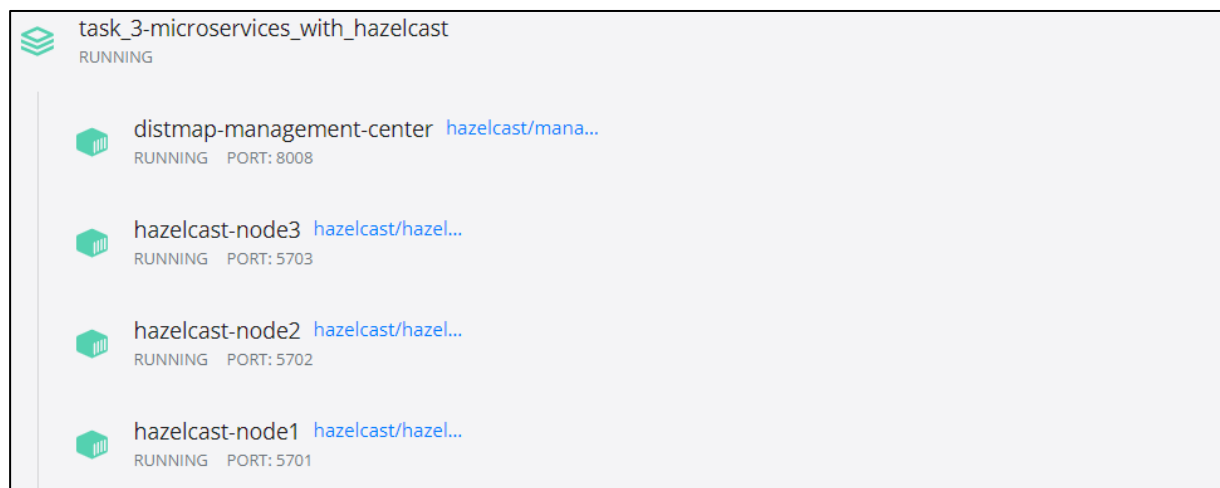
PS C:\Users\t-1900> python
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import requests
>>>
```

## Частина 1. Базова функціональність системи

- Опис HTTP POST/GET Request Flow



Нижче наведено перелік докер-контейнерів, на основі яких розгорнений 3-ох нодовий Hazelcast-кластер, що використовується в якості сховища повідомлень:



```
t-1000@DESKTOP-DRI0PBB MINGW64 ~  
$ docker ps  
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS  
34631f4dcf4f   hazelcast/management-center:5.4.0   "bash ./bin/mc-start..." 5 hours ago   Up 20 minutes 8081/tcp, 8443/tcp, 0.0.0.0:8008->8080/tcp  
566a5818f5de   hazelcast/hazelcast:5.4.0           "hz start"               5 hours ago   Up 20 minutes 0.0.0.0:5703->5701/tcp  
8be04c3cb310   hazelcast/hazelcast:5.4.0           "hz start"               5 hours ago   Up 20 minutes 0.0.0.0:5702->5701/tcp  
20853aaab99    hazelcast/hazelcast:5.4.0           "hz start"               5 hours ago   Up 20 minutes 0.0.0.0:5701->5701/tcp
```

## Завдання до виконання:

1. Запустити три екземпляра **logging-service** (якщо локально, то на різних портах), і відповідно мають запуститись також три екземпляра **Hazelcast**.

```
Select Windows PowerShell
>>> res = requests.get("http://localhost:8081/logging")
>>> print(res.json())
[]
>>> res = requests.get("http://localhost:8083/logging")
>>> print(res.json())
[]
>>> res = requests.get("http://localhost:8085/logging")
>>> print(res.json())
```

```
Select Windows PowerShell
(venv) PS D:\KPI\5 курс Магістрат\Проектування PC\Task_3-Microservices_with_Hazelcast
> python .\logging-service.py --port 8081
* Serving Flask app 'logging-service' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8081 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 16:04:35] "GET /logging HTTP/1.1" 200 -
```

```
Select Windows PowerShell
(venv) PS D:\KPI\5 курс Магістрат\Проектування PC\Task_3-Microservices_with_Hazelcast
> python .\logging-service.py --port 8083
* Serving Flask app 'logging-service' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8083 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 16:05:33] "GET /logging HTTP/1.1" 200 -
```

```
Select Windows PowerShell
(venv) PS D:\KPI\5 курс Магістрат\Проектування PC\Task_3-Microservices_with_Hazelcast
> python .\logging-service.py --port 8085
* Serving Flask app 'logging-service' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8085 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 16:05:40] "GET /logging HTTP/1.1" 200 -
```

Members										
Search			Default View							
Member		Script...	Console	Streaming		Hazel...		OS Com...	OS ...	
192.168.88.232:5701	△	Disabled	Disabled	Enabled	No	5.4.0	90	11.74 GB	4.48 %	
192.168.88.232:5702	△	Disabled	Disabled	Enabled	No	5.4.0	91	11.75 GB	1.99 %	
192.168.88.232:5703	△	Disabled	Disabled	Enabled	No	5.4.0	90	11.74 GB	4.93 %	

## 2. Через HTTP-POST записати 10 повідомлень msg1-msg10 ч/з *facade-service*.

```
Windows PowerShell
* Running on http://127.0.0.1:8080 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [07/Apr/2025 16:32:56] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [07/Apr/2025 16:32:56] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [07/Apr/2025 16:32:56] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [07/Apr/2025 16:32:56] "POST /facade HTTP/1.1" 201 -

Windows PowerShell
>>> for i in range(1, 11):
...     msg = "msg" + str(i)
...     res = requests.post("http://localhost:8080/facade", data={"msg": msg})
...     print(res.json())
...
{'message': {'msg': 'msg1', 'uuid': '8821c6bb-efe6-4cec-86df-76fa828e33e3'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'msg2', 'uuid': 'd47e2df5-3a71-4716-971f-45e638599a7c'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'msg3', 'uuid': '2db9a838-786d-40f9-ba7d-5ec49921b695'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'msg4', 'uuid': '9e72337f-08a9-45f6-b170-fe70396df334'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'msg5', 'uuid': 'd3301c2d-49d1-4a12-a568-4f39292d5f06'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'msg6', 'uuid': '8fe30f6a-677b-4cd9-9a40-d5f96479f178'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'msg7', 'uuid': '88438544-b767-4715-a610-6dfb037fe988'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'msg8', 'uuid': '0b30ea74-7416-465d-9dee-f108d458b12c'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'msg9', 'uuid': '13155255-89ad-4274-82e5-7cae5e0817e5'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'msg10', 'uuid': '2ef687be-f9f7-4e90-9aef-cf58c80d3fb'}, 'response': {'success': 'Message Was Logged'}}
>>>
```

Map Statistics (In-Memory Format: BINARY)

RESET TIME 1 minute ago → now

Default View

Member ^	Entries	Gets	Puts	Rem...	Sets	Entry Memory	Eve...	Hits
192.168.88.232:5701	2	0	2	0	0	328.00 B	0	0
192.168.88.232:5702	3	0	3	0	0	492.00 B	0	0
192.168.88.232:5703	5	0	5	0	0	821.00 B	0	0
TOTAL	10	0	10	0	0	1.60 kB	0	0

## 3. Показати, які повідомлення отримав кожен з екземплярів *logging-service*.

```
Windows PowerShell
* Running on http://127.0.0.1:8081 (Press CTRL+C to quit)
- Message Was Logged: msg4
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg5
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg6
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg10
127.0.0.1 - - [07/Apr/2025 16:32:56] "POST /logging HTTP/1.1" 201 -

Windows PowerShell
* Running on http://127.0.0.1:8083 (Press CTRL+C to quit)
- Message Was Logged: msg2
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg3
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg8
127.0.0.1 - - [07/Apr/2025 16:32:56] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg9
127.0.0.1 - - [07/Apr/2025 16:32:56] "POST /logging HTTP/1.1" 201 -

Windows PowerShell
* Running on http://127.0.0.1:8085 (Press CTRL+C to quit)
- Message Was Logged: msg1
127.0.0.1 - - [07/Apr/2025 16:32:55] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg7
127.0.0.1 - - [07/Apr/2025 16:32:56] "POST /logging HTTP/1.1" 201 -
```

#### 4. Через HTTP-GET з *facade-service* прочитати всі повідомлення.

```
Windows PowerShell
* Running on http://127.0.0.1:8080 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 16:49:54] "GET /facade HTTP/1.1" 200 -

Windows PowerShell
* Running on http://127.0.0.1:8081 (Press CTRL+C to quit)

Windows PowerShell
* Running on http://127.0.0.1:8082 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 16:49:54] "GET /messages HTTP/1.1" 200 -

Windows PowerShell
* Running on http://127.0.0.1:8083 (Press CTRL+C to quit)







Windows PowerShell
* Running on http://127.0.0.1:8085 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 16:49:54] "GET /logging HTTP/1.1" 200 -

Windows PowerShell
* Running on http://127.0.0.1:8888 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 16:49:54] "GET /services/logging-service HTTP/1.1" 200 -
127.0.0.1 - - [07/Apr/2025 16:49:54] "GET /services/messages-service HTTP/1.1" 200 -

Windows PowerShell
>>> print(res.json())
[{"logging-service": ["msg6", "msg7", "msg3", "msg8", "msg4", "msg2", "msg9", "msg1", "msg10", "msg5"]\n, "messages-service: I'm not implemented yet ^_^"}
>>>
```

#### 5. Вимкнути один/два екземпляри *logging-service* (разом з ними будуть вимикатись й ноди Hazelcast) та перевірити чи вичитується повідомлення.

```
t-1000@DESKTOP-DRIOPBB MINGW64 ~
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
34631f4dcf4f   hazelcast/management-center:5.4.0   "bash ./bin/mc-start..." 6 hours ago    Up 40 minutes 8081/tcp, 8443/tcp, 0.0.0.0:8008->8080/tcp
20853aaab99    hazelcast/hazelcast:5.4.0           "hz start"                6 hours ago    Up 40 minutes 0.0.0.0:5701->5701/tcp
```

Members 									
Search  Default View    									
Member ^	Script...	Console ^	Streaming ^	Hazel...	OS Com...	OS ...			
192.168.50.48:5701	No In	Disabled	Disabled	Enabled	No	5.4.0	271	11.75 GB	1.38 %

```
Windows PowerShell
* Running on http://127.0.0.1:8080 (Press CTRL+C to quit)
W! GET-request from '{api-endpoint}' at 20:32:58
! RetryError: HTTPConnectionPool(host='localhost', port=8081): Max retries exceeded with url: /logging (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x00000283090CA140>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))
127.0.0.1 - - [07/Apr/2025 20:33:00] "GET /facade HTTP/1.1" 200 -

Windows PowerShell
* Running on http://127.0.0.1:8081 (Press CTRL+C to quit)
+++++[STOP-NODE]+++++
hazelcast-node2
PS D:\KPI\5 курс Марістрат\Проектування PC\Task_3-Microservices_with_Hazelcast>




Windows PowerShell
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8083 (Press CTRL+C to quit)
+++++[STOP-NODE]+++++
hazelcast-node3
PS D:\KPI\5 курс Марістрат\Проектування PC\Task_3-Microservices_with_Hazelcast>

Windows PowerShell
* Running on http://127.0.0.1:8082 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 20:33:00] "GET /messages HTTP/1.1" 200 -

Windows PowerShell
* Running on http://127.0.0.1:8085 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 20:33:00] "GET /logging HTTP/1.1" 200 -

Windows PowerShell
* Running on http://127.0.0.1:8888 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 20:33:54] "GET /services/logging-service HTTP/1.1" 200 -
127.0.0.1 - - [07/Apr/2025 20:33:00] "GET /services/logging-service HTTP/1.1" 200 -
127.0.0.1 - - [07/Apr/2025 20:33:00] "GET /services/messages-service HTTP/1.1" 200 -

Windows PowerShell
>>> res = requests.get("http://localhost:8080/facade")
>>> print(res.json())
[{"logging-service": ["msg3", "msg9", "msg7", "msg10", "msg6", "msg4", "msg1", "msg2", "msg5", "msg8"]\n, "messages-service: I'm not implemented yet ^_^"}
>>>
```

Map Statistics (In-Memory Format: BINARY) <span>RESET TIME</span> <span>1 minute ago</span> <span>now</span> <span>Default View</span>   									
Member ^	Entries	Gets	Puts	Remov...	Sets	Entry Me...	Ev...	Hits	
192.168.50.48:5701	10	12	2	0	0	1.60 kB	0	12	
TOTAL	10	12	2	0	0	1.60 kB	0	12	

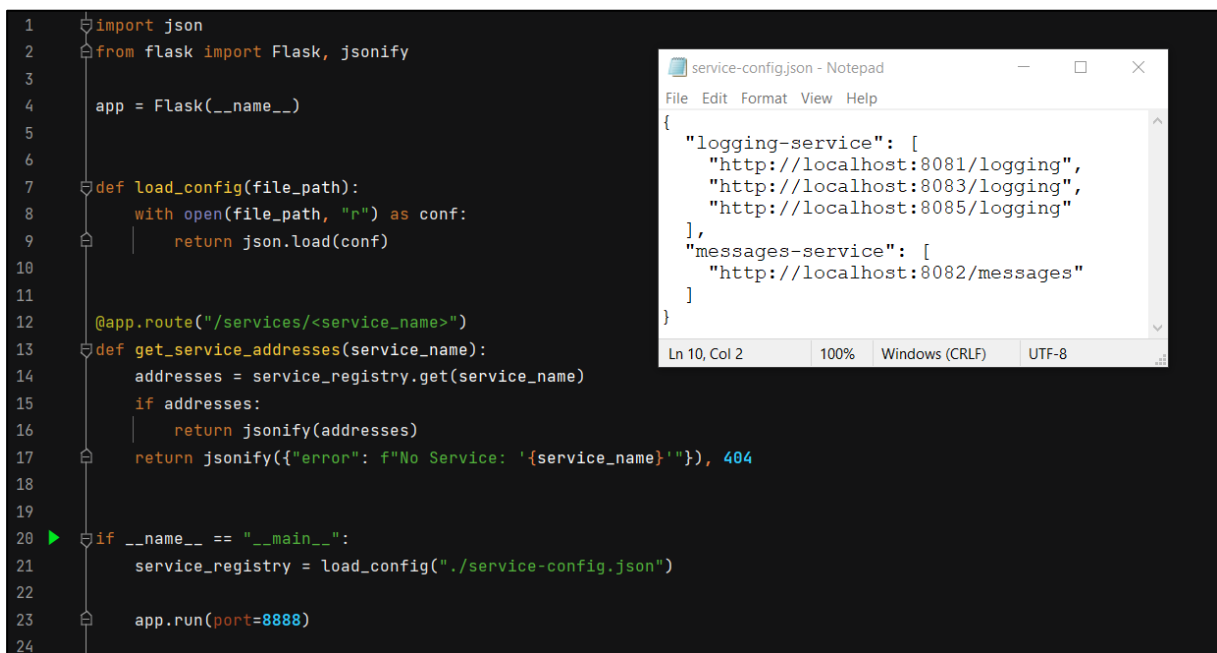


## Частина 2. Додатковий функціонал системи

Через те, що тепер може бути декілька запущених екземплярів **logging-service**, то **facade-service** має знати про їх IP-адреси, для доступу до них.

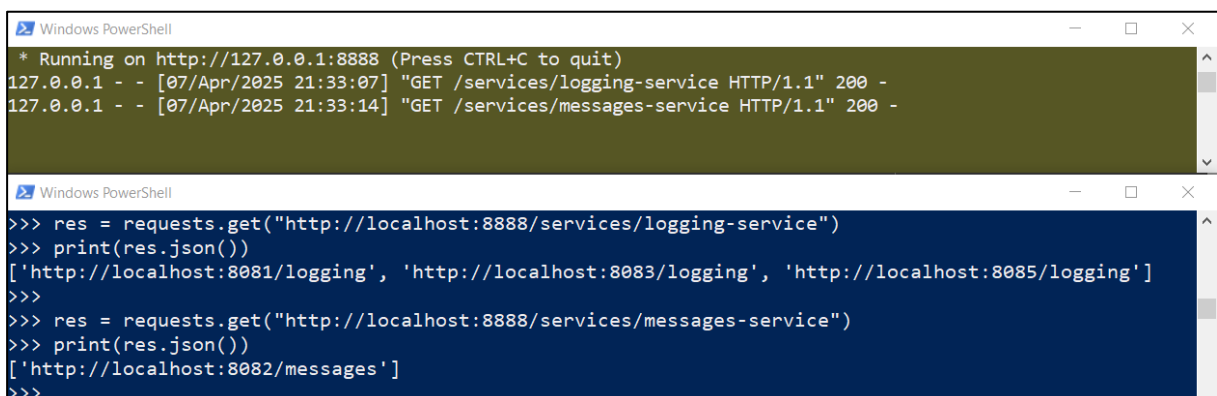
Перелік IP-адрес, може міститись у коді самого **facade-service**, чи передаватись у нього при старті, проте такий підхід не є гнучким, у разі динамічного призначення чи зміни цих адрес. Тому, винесемо інформацію про IP-адреси інших мікросервісів у окремий реєстр, за який буде відповідати **config-server** (пізніше він буде замінений на “Service registry and discovery”).

Отже, надалі перед зверненням до **logging-service** та **messages-service**, **facade-service** буде робити запит до **config-server**, де по імені сервіса йому повертатиметься перелік всіх IP-адрес екземплярів даного сервіса. Перелік IP-адрес мікросервісів на **config-server**, братиметься з конфігураційного файлу:



```
1 import json
2 from flask import Flask, jsonify
3
4 app = Flask(__name__)
5
6
7 def load_config(file_path):
8     with open(file_path, "r") as conf:
9         return json.load(conf)
10
11
12 @app.route("/services/<service_name>")
13 def get_service_addresses(service_name):
14     addresses = service_registry.get(service_name)
15     if addresses:
16         return jsonify(addresses)
17     return jsonify({"error": f"No Service: '{service_name}'"}), 404
18
19
20 if __name__ == "__main__":
21     service_registry = load_config("./service-config.json")
22
23     app.run(port=8888)
24
```

```
{
  "logging-service": [
    "http://localhost:8081/logging",
    "http://localhost:8083/logging",
    "http://localhost:8085/logging"
  ],
  "messages-service": [
    "http://localhost:8082/messages"
  ]
}
```



```
* Running on http://127.0.0.1:8888 (Press CTRL+C to quit)
127.0.0.1 - - [07/Apr/2025 21:33:07] "GET /services/logging-service HTTP/1.1" 200 -
127.0.0.1 - - [07/Apr/2025 21:33:14] "GET /services/messages-service HTTP/1.1" 200 -
```

```
>>> res = requests.get("http://localhost:8888/services/logging-service")
>>> print(res.json())
['http://localhost:8081/logging', 'http://localhost:8083/logging', 'http://localhost:8085/logging']
>>>
>>> res = requests.get("http://localhost:8888/services/messages-service")
>>> print(res.json())
['http://localhost:8082/messages']
>>>
```