



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 №4

Мікросервиси з використанням Messaging Queue

Перевірив:

Родіонов А. М.

Виконав:

студент І 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 *messages-service*, необхідно додати:

- підтримку “**Message Queue**” (Hazelcast чи Kafka) у якості каналу доставки будь-яких повідомлень між *facade-service* та *messages-service*.
- можливість запускати одночасно декілька копій *messages-service*.
- механізм, з яким *facade-service* довільним чином зможе обирати до якої копії *messages-service* звертатись для запису та читання повідомлень.

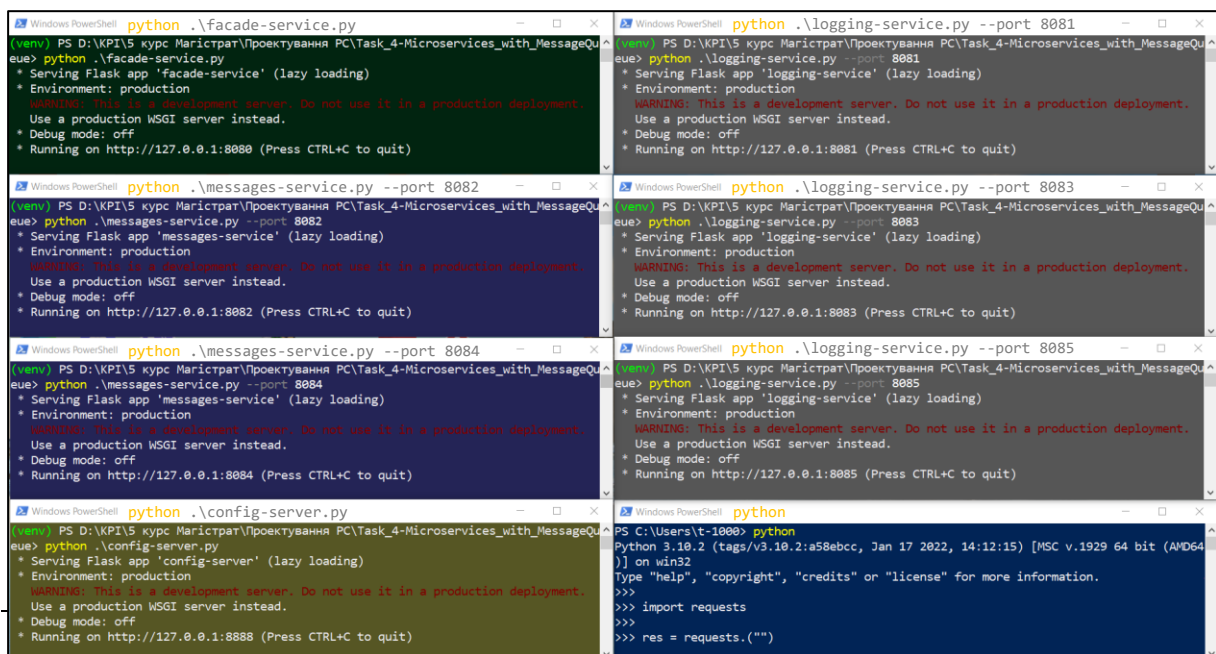
Архітектура складається з трьох мікросервісів, реалізованих на мові **Python**:

- *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* – отримує та зберігає повідомлення із черги.
- *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 4-Microservices with MessageQueue/](https://github.com/sazan24/KPI/tree/main/Master's%20degree/Distributed%20Systems%20Design/Task%204-Microservices%20with%20MessageQueue/)

Запуск сервісів на портах: фасад x1 (8080), меседж x2 (8082, 8084), логгін x3 (8081, 8083, 8085) та сервер конфігураційних записів (8888):



```
python .\facade-service.py
PS D:\KPI\5 курс Марієтпар\Проектування PC\Task_4-Microservices_with_MessageQueue> 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)

python .\logging-service.py --port 8081
PS D:\KPI\5 курс Марієтпар\Проектування PC\Task_4-Microservices_with_MessageQueue> 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)

python .\messages-service.py --port 8082
PS D:\KPI\5 курс Марієтпар\Проектування PC\Task_4-Microservices_with_MessageQueue> python .\messages-service.py --port 8082
* 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)

python .\logging-service.py --port 8083
PS D:\KPI\5 курс Марієтпар\Проектування PC\Task_4-Microservices_with_MessageQueue> 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)

python .\messages-service.py --port 8084
PS D:\KPI\5 курс Марієтпар\Проектування PC\Task_4-Microservices_with_MessageQueue> python .\messages-service.py --port 8084
* 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://127.0.0.1:8084 (Press CTRL+C to quit)

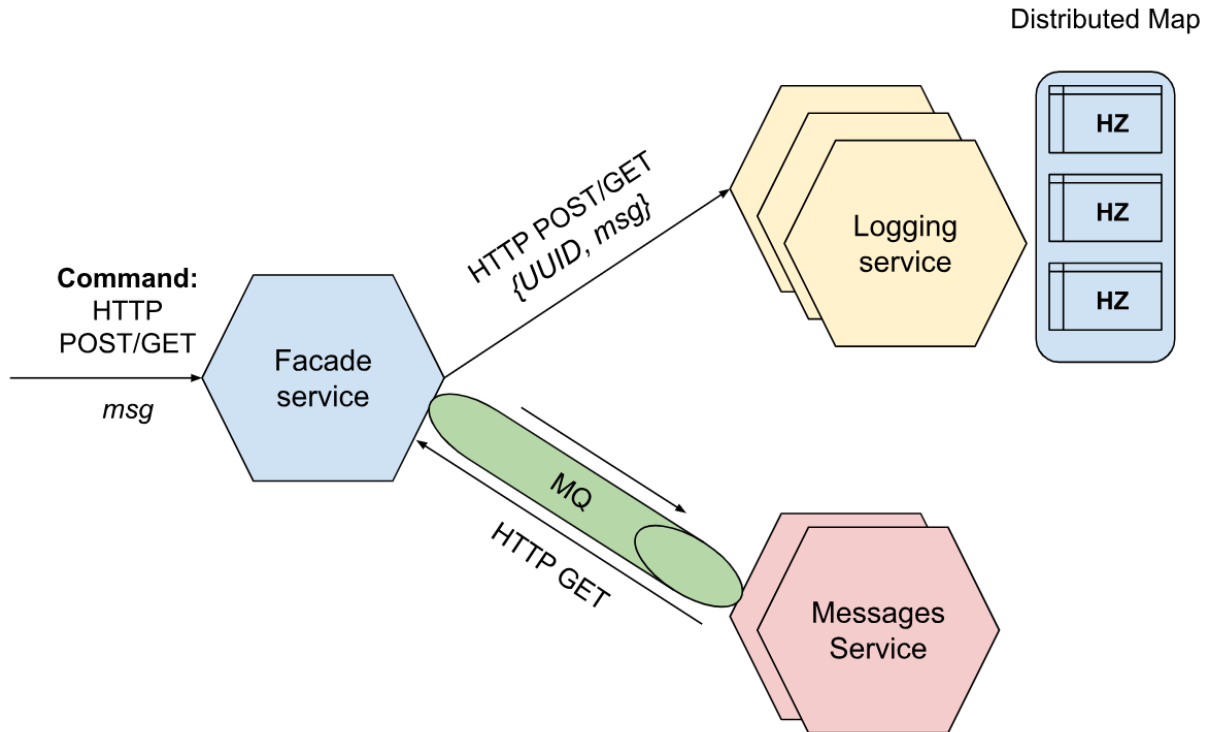
python .\logging-service.py --port 8085
PS D:\KPI\5 курс Марієтпар\Проектування PC\Task_4-Microservices_with_MessageQueue> python .\logging-service.py --port 8085
* Serving Flask app 'logging-service' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:8085 (Press CTRL+C to quit)

python .\config-server.py
PS D:\KPI\5 курс Марієтпар\Проектування PC\Task_4-Microservices_with_MessageQueue> 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)


python
PS C:\Users\it-1000> 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
>>> res = requests.get("http://127.0.0.1:8888")
>>>
```


Частина 1. Демонстрація функціональності системи

- Опис HTTP POST/GET Request Flow




Нижче наведено докер-контейнери, на основі яких розгорнений 2-ох нодовий Hazelcast-кластер, що використовується в якості сховища та черги повідомлень:


 task_4-microservices_with_messagequeue
RUNNING



hazelcast-node1 [hazelcast/hazel...](#)
RUNNING PORT: 5701



hazelcast-node2 [hazelcast/hazel...](#)
RUNNING PORT: 5702



distque-managment-center [hazelcast/mana...](#)
RUNNING PORT: 8008

```
t-1000@DESKTOP-DRI0PBB MINGW64 /d/KPI/5 курс Марістрат/Проектування PC/Task_4-Microservices_with_MessageQueue
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
7767f3e6747b   hazelcast/management-center:5.4.0   "bash ./bin/mc-start..." 3 minutes ago  Up 3 minutes  8081/tcp, 8443/tcp, 0.0.0.0:8008->8080/tcp
b782296efe34   hazelcast/hazelcast:5.4.0           "hz start"               3 minutes ago  Up 3 minutes  0.0.0.0:5701->5701/tcp
65e7b3e15c7a   hazelcast/hazelcast:5.4.0           "hz start"               3 minutes ago  Up 3 minutes  0.0.0.0:5702->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 python
>>> 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 python .\logging-service.py --port 8081
(venv) PS D:\KPI\5 курс Магістрат\Проектування PC\Task_4-Microservices_with_MessageQueues> 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 - - [08/Apr/2025 14:15:07] "GET /logging HTTP/1.1" 200 -
```

```
Select Windows PowerShell python .\logging-service.py --port 8083
(venv) PS D:\KPI\5 курс Магістрат\Проектування PC\Task_4-Microservices_with_MessageQueues> 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 - - [08/Apr/2025 14:15:43] "GET /logging HTTP/1.1" 200 -
```

```
Select Windows PowerShell python .\logging-service.py --port 8085
(venv) PS D:\KPI\5 курс Магістрат\Проектування PC\Task_4-Microservices_with_MessageQueues> 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 - - [08/Apr/2025 14:15:49] "GET /logging HTTP/1.1" 200 -
```

Members									
Search									
Default View									
Member	Script...	Con...	Stream...	H...	Owned...	OS Commit...	OS...		
10.246.8.9:5701	△ Disabled	Disabled	Enabled	No	5.4.0	136	11.74 GB	3.44 %	
10.246.8.9:5702	△ Disabled	Disabled	Enabled	No	5.4.0	135	11.74 GB	3.84 %	

```
>>> 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",  
        "uid": "9a82a795-0ece-4b92-a3d4-156730edf2e5",  
        "response": {  
            "success": "Message Was Logged"  
        }  
    },  
    "message": {  
        "msg": "msg2",  
        "uid": "4dc45ff9-2bc3-4c70-8f85-fc5691ca639",  
        "response": {  
            "success": "Message Was Logged"  
        }  
    },  
    "message": {  
        "msg": "msg3",  
        "uid": "9a32d3cb-e439-4c34-b157-332851ad99bd",  
        "response": {  
            "success": "Message Was Logged"  
        }  
    },  
    "message": {  
        "msg": "msg4",  
        "uid": "f5ba56e6-2f56-4c3e-8959-364af75bb6ffe",  
        "response": {  
            "success": "Message Was Logged"  
        }  
    },  
    "message": {  
        "msg": "msg5",  
        "uid": "1bc8bf9d-a950-4017-9c01-4fc4cb997581",  
        "response": {  
            "success": "Message Was Logged"  
        }  
    },  
    "message": {  
        "msg": "msg6",  
        "uid": "fd1a9f5c-b120-41b1-b42c-9e41647e7302",  
        "response": {  
            "success": "Message Was Logged"  
        }  
    },  
    "message": {  
        "msg": "msg7",  
        "uid": "54ae663ea-5e37-496a-e109-74646cae839d",  
        "response": {  
            "success": "Message Was Logged"  
        }  
    },  
    "message": {  
        "msg": "msg8",  
        "uid": "98f53a14-8fe3-43e4-82b3-a88c86b7335f",  
        "response": {  
            "success": "Message Was Logged"  
        }  
    },  
    "message": {  
        "msg": "msg9",  
        "uid": "bd2e4421-9763-4126-bb1e-7d64d86795cf",  
        "response": {  
            "success": "Message Was Logged"  
        }  
    },  
    "message": {  
        "msg": "msg10",  
        "uid": "938405a9-52c0-4b3e-aad8-512b0a1a8d7",  
        "response": {  
            "success": "Message Was Logged"  
        }  
    }  
}
```


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

```
Windows PowerShell python .\logging-service.py --port 8081
- Message Was Logged: msg6
127.0.0.1 - - [08/Apr/2025 14:54:07] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg7
127.0.0.1 - - [08/Apr/2025 14:54:07] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg9
127.0.0.1 - - [08/Apr/2025 14:54:07] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg10
127.0.0.1 - - [08/Apr/2025 14:54:07] "POST /logging HTTP/1.1" 201 -

Windows PowerShell python .\logging-service.py --port 8083
- Message Was Logged: msg1
127.0.0.1 - - [08/Apr/2025 14:54:06] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg2
127.0.0.1 - - [08/Apr/2025 14:54:07] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg3
127.0.0.1 - - [08/Apr/2025 14:54:07] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg5
127.0.0.1 - - [08/Apr/2025 14:54:07] "POST /logging HTTP/1.1" 201 -

Windows PowerShell python .\logging-service.py --port 8085
- Message Was Logged: msg4
127.0.0.1 - - [08/Apr/2025 14:54:07] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: msg8
127.0.0.1 - - [08/Apr/2025 14:54:07] "POST /logging HTTP/1.1" 201 -
```

Map Statistics (In-Memory Format: BINARY) RESET TIME 1 minute ago → now

Default View 🗑️ 📄 +

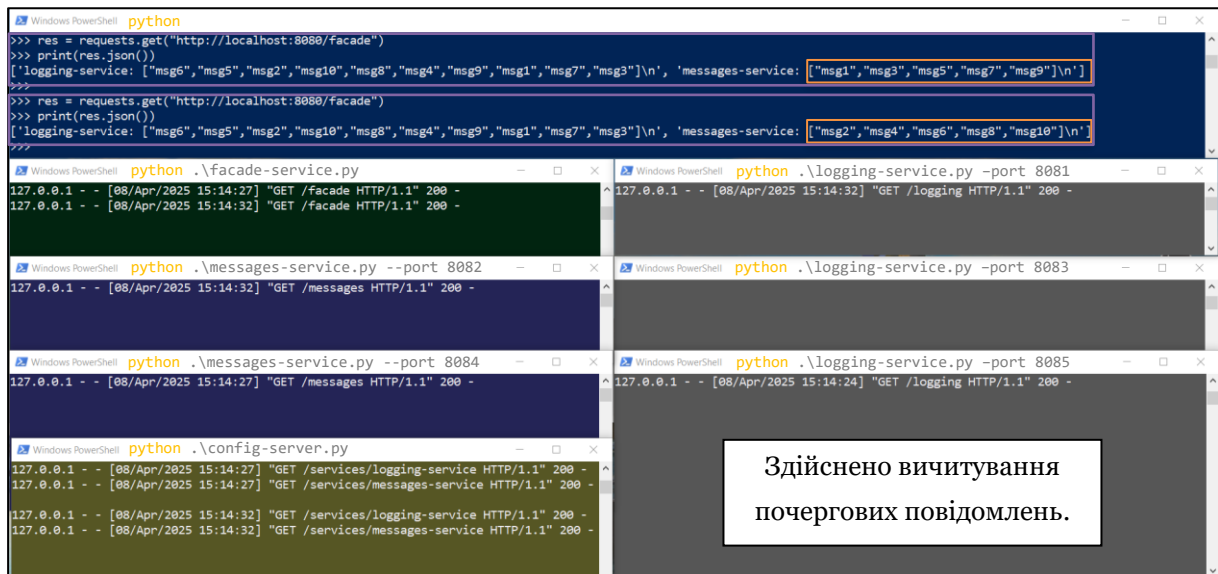
Member ^	Entries	Gets	Puts	Remov...	S...	Entry Memory	Ev...	Hits
10.246.8.9:5701	4	0	4	0	0	656.00 B	0	0
10.246.8.9:5702	6	0	6	0	0	985.00 B	0	0
TOTAL	10	0	10	0	0	1.60 kB	0	0

5. Показати, які повідомлення отримав кожен з екземплярів *messages-service*

```
Windows PowerShell python .\messages-service.py --port 8082
- Message Was Received: 'msg2'
- Message Was Received: 'msg4'
- Message Was Received: 'msg6'
- Message Was Received: 'msg8'
- Message Was Received: 'msg10'

Windows PowerShell python .\messages-service.py --port 8084
- Message Was Received: 'msg1'
- Message Was Received: 'msg3'
- Message Was Received: 'msg5'
- Message Was Received: 'msg7'
- Message Was Received: 'msg9'
```

6. Декілька разів викликати HTTP-GET на **facade-service** та отримати об'єднані дві множини повідомлень з **logging-service** та **messages-service**.



```
python
>>> res = requests.get("http://localhost:8080/facade")
>>> print(res.json())
[{"logging-service": ["msg6", "msg5", "msg2", "msg10", "msg8", "msg4", "msg9", "msg1", "msg7", "msg3"], "messages-service": ["msg1", "msg3", "msg5", "msg7", "msg9"]}]

>>> res = requests.get("http://localhost:8080/facade")
>>> print(res.json())
[{"logging-service": ["msg6", "msg5", "msg2", "msg10", "msg8", "msg4", "msg9", "msg1", "msg7", "msg3"], "messages-service": ["msg2", "msg4", "msg6", "msg8", "msg10"]}]

python .\facade-service.py
127.0.0.1 - - [08/Apr/2025 15:14:27] "GET /facade HTTP/1.1" 200 -
127.0.0.1 - - [08/Apr/2025 15:14:32] "GET /facade HTTP/1.1" 200 -

python .\messages-service.py --port 8082
127.0.0.1 - - [08/Apr/2025 15:14:32] "GET /messages HTTP/1.1" 200 -

python .\messages-service.py --port 8084
127.0.0.1 - - [08/Apr/2025 15:14:27] "GET /messages HTTP/1.1" 200 -

python .\config-server.py
127.0.0.1 - - [08/Apr/2025 15:14:27] "GET /services/logging-service HTTP/1.1" 200 -
127.0.0.1 - - [08/Apr/2025 15:14:27] "GET /services/messages-service HTTP/1.1" 200 -
127.0.0.1 - - [08/Apr/2025 15:14:32] "GET /services/logging-service HTTP/1.1" 200 -
127.0.0.1 - - [08/Apr/2025 15:14:32] "GET /services/messages-service HTTP/1.1" 200 -

python .\logging-service.py -port 8081
127.0.0.1 - - [08/Apr/2025 15:14:32] "GET /logging HTTP/1.1" 200 -

python .\logging-service.py -port 8083
127.0.0.1 - - [08/Apr/2025 15:14:24] "GET /logging HTTP/1.1" 200 -

python .\logging-service.py -port 8085
127.0.0.1 - - [08/Apr/2025 15:14:24] "GET /logging HTTP/1.1" 200 -

Здійснено вичитування
почергових повідомлень.
```

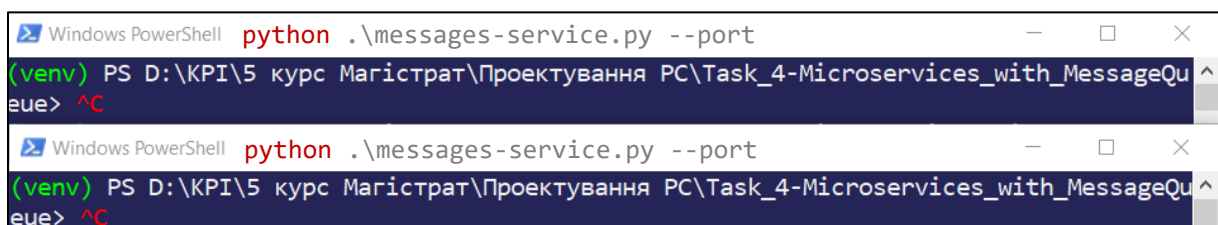
Частина 2. Перевірка відмовостійкості черги повідомлень

Класична черга повідомлень зберігає додані в неї повідомлення на своїй стороні, тому якщо її сервер раптово “впаде”, то повідомлення які містяться у черзі стануть недоступними та можуть взагалі втратитись.

Для запобігання цього, рекомендується налаштовувати кластер для “Message Queue”, тобто запускати декілька серверів та налаштовувати між ними реплікацію (чи backup для Hazelcast). Тепер якщо один сервер з кластеру “Message Queue” впаде, то повідомлення будуть доступні з його репліки (копії)

Для перевірки відмовостійкості вам необхідно буде перевірити, що не буде втрат повідомлень у випадку зупинки одного з серверів “Message Queue”.

- не запускайте (вимкніть) два екземпляри **messages-service**, для того щоб повідомлення тимчасово не вичитувались та зберігались у “Message Queue”



```
python .\messages-service.py --port
(venv) PS D:\KPI\5 курс Марістрат\Проектування PC\Task_4-Microservices_with_MessageQueue> ^C

python .\messages-service.py --port
(venv) PS D:\KPI\5 курс Марістрат\Проектування PC\Task_4-Microservices_with_MessageQueue> ^C
```

– відправте 10 (можна і 100) повідомлень через *facade-service*

```
python
>>> for i in range(1, 11):
...     msg = "flt" + str(i)
...     res = requests.post("http://localhost:8080/facade", data={"msg": msg})
...     print(res.json())
...
{'message': {'msg': 'flt1', 'uuid': 'd968322b-c47a-4ad4-8f25-7152f167fbfe'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'flt2', 'uuid': 'e07990a0-ccb7-45bc-a8f7-815ad45a3251'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'flt3', 'uuid': '9bde791b-8464-47d9-bd7a-e5a44b53a645'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'flt4', 'uuid': 'a88929ea-acc6-4c1a-9b3f-031ebc76cd45'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'flt5', 'uuid': '8e4a7fc5-21e4-47c2-8ae7-d23cf0d4790d'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'flt6', 'uuid': 'c86561e7-49e3-4b33-afe2-6ce4d4779897'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'flt7', 'uuid': 'ae3ead37-30ab-4c0d-a08d-6882b228e9fc'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'flt8', 'uuid': 'b753aa85-6f45-405e-8509-ce89040af63c'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'flt9', 'uuid': '873942d5-58a7-4706-bc13-ed3793d4d09e'}, 'response': {'success': 'Message Was Logged'}}
{'message': {'msg': 'flt10', 'uuid': '25055a2c-a531-4181-a837-5cb5a98f527d'}, 'response': {'success': 'Message Was Logged'}}
>>>

python .\facade-service.py
- Message Was Queued: 'flt1'
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /facade HTTP/1.1" 201 -
- Message Was Queued: 'flt2'
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /facade HTTP/1.1" 201 -
- Message Was Queued: 'flt3'
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /facade HTTP/1.1" 201 -
- Message Was Queued: 'flt4'
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /facade HTTP/1.1" 201 -
- Message Was Queued: 'flt5'
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /facade HTTP/1.1" 201 -
- Message Was Queued: 'flt6'
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /facade HTTP/1.1" 201 -
- Message Was Queued: 'flt7'
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /facade HTTP/1.1" 201 -
- Message Was Queued: 'flt8'
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /facade HTTP/1.1" 201 -
- Message Was Queued: 'flt9'
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /facade HTTP/1.1" 201 -
- Message Was Queued: 'flt10'
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /facade HTTP/1.1" 201 -

python .\config-server.py
127.0.0.1 - - [08/Apr/2025 15:55:22] "GET /services/logging-service HTTP/1.1" 200 -

python .\logging-service.py --port 8081
- Message Was Logged: flt10
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /logging HTTP/1.1" 201 -

python .\logging-service.py --port 8083
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: flt6
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: flt7
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: flt9
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /logging HTTP/1.1" 201 -

python .\logging-service.py --port 8085
127.0.0.1 - - [08/Apr/2025 15:14:24] "GET /logging HTTP/1.1" 200 -
- Message Was Logged: flt2
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: flt3
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: flt4
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: flt8
127.0.0.1 - - [08/Apr/2025 15:55:22] "POST /logging HTTP/1.1" 201 -
```

Queue Statistics

RESET METRICS

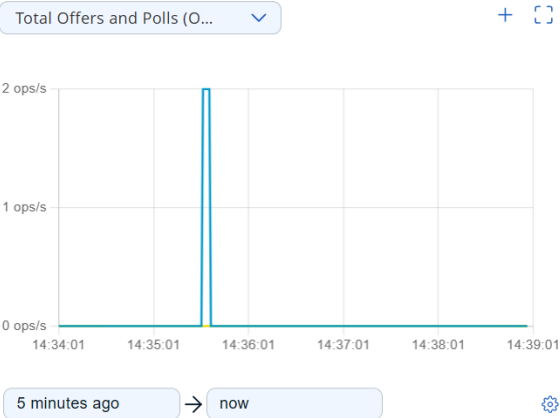
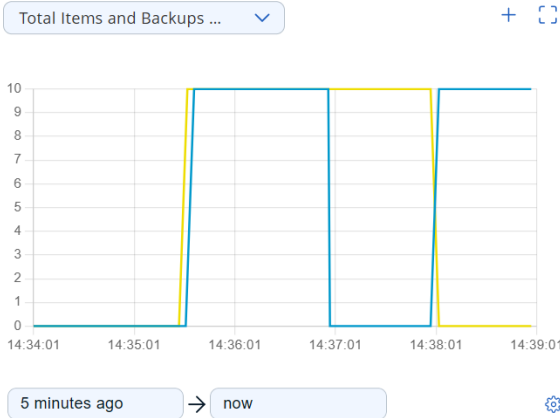
Now

Member ^	Items ^	Backups ^	Max Age ^	Min Age ^	Average Age ^
10.246.8.9:5701	10	0	2ms	1ms	1ms
10.246.8.9:5702	0	10	1ms	1ms	1ms

– вимкніть той із серверів (“Message Queue”), який містить елементи

```
t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Магістрат/Проектування PC/Task_4-Microservices_with_MessageQueue
$ docker stop b782296efe34
b782296efe34

t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Магістрат/Проектування PC/Task_4-Microservices_with_MessageQueue
$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
7767f3e6747b   hazelcast/management-center:5.4.0   "bash ./bin/mc-start..." 2 hours ago   Up 2 hours   8081/tcp, 8443/tcp, 0.0.0.0:8008->8080/tcp
65e7b3e15c7a   hazelcast/hazelcast:5.4.0           "hz start"               2 hours ago   Up About an hour   0.0.0.0:5702->5701/tcp
```



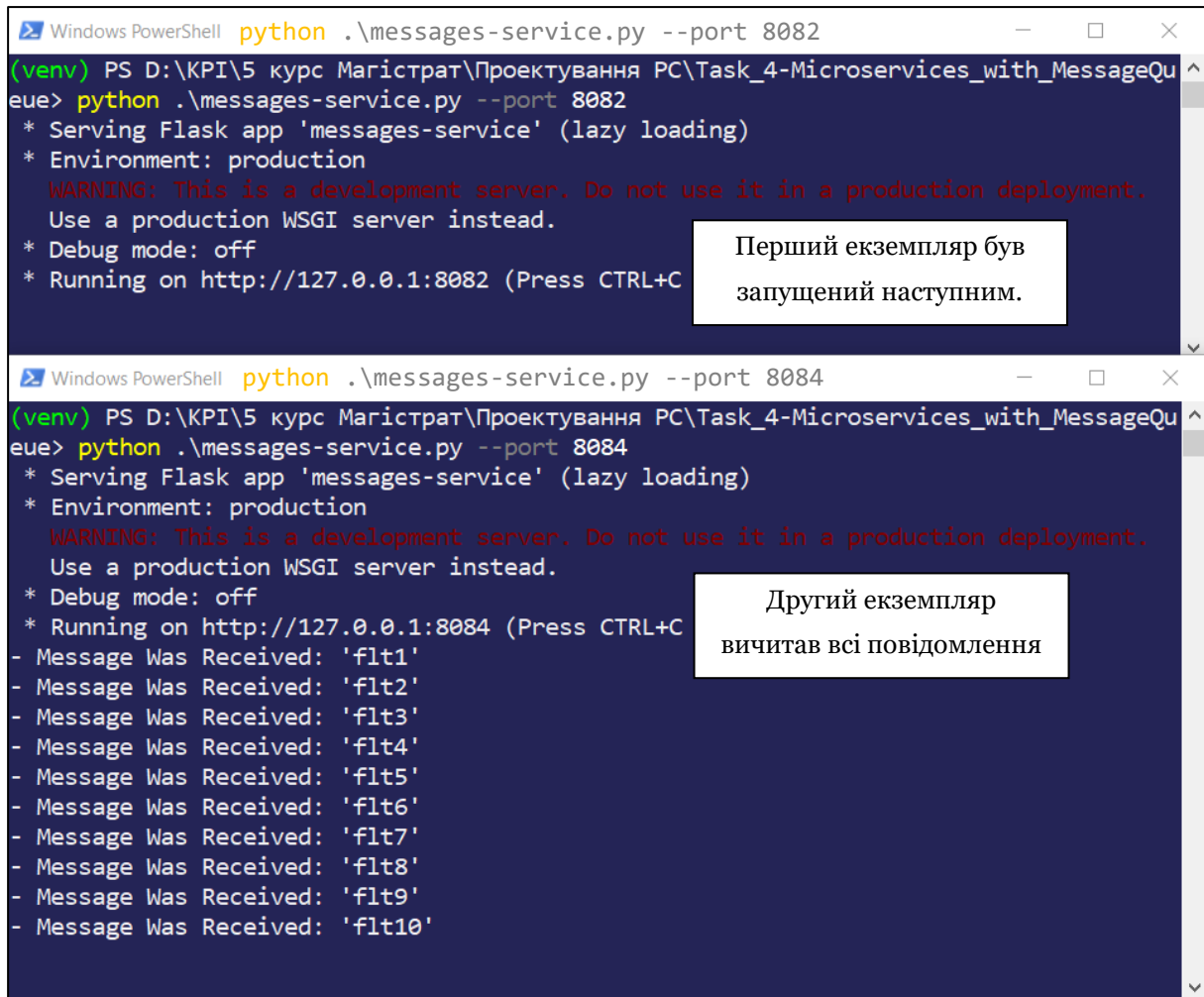
Queue Statistics

RESET METRICS

Now

Member ^	Items ^	Backups ^	Max Age ^	Min Age ^	Average Age ^
10.246.8.9:5702	10	0	1ms	1ms	1ms

- запустити два екземпляри **messages-service** і перевірити, що вони отримали всі повідомлення з черги (можуть бути запущені асинхронно)



```
Windows PowerShell python .\messages-service.py --port 8082
(venv) PS D:\KPI\5 курс Марістрат\Проектування PC\Task_4-Microservices_with_MessageQueue> python .\messages-service.py --port 8082
* 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)
```

Перший екземпляр був запущений наступним.

```
Windows PowerShell python .\messages-service.py --port 8084
(venv) PS D:\KPI\5 курс Марістрат\Проектування PC\Task_4-Microservices_with_MessageQueue> python .\messages-service.py --port 8084
* 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://127.0.0.1:8084 (Press CTRL+C to quit)
- Message Was Received: 'flt1'
- Message Was Received: 'flt2'
- Message Was Received: 'flt3'
- Message Was Received: 'flt4'
- Message Was Received: 'flt5'
- Message Was Received: 'flt6'
- Message Was Received: 'flt7'
- Message Was Received: 'flt8'
- Message Was Received: 'flt9'
- Message Was Received: 'flt10'
```

Другий екземпляр вичитав всі повідомлення