



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 №1

Базова архітектура мікропроцесорів

Перевірив:

Родіонов А. М.

Виконав:

студент І курсу

групи ФБ-41мп

Сахній Н. Р.

Київ 2025

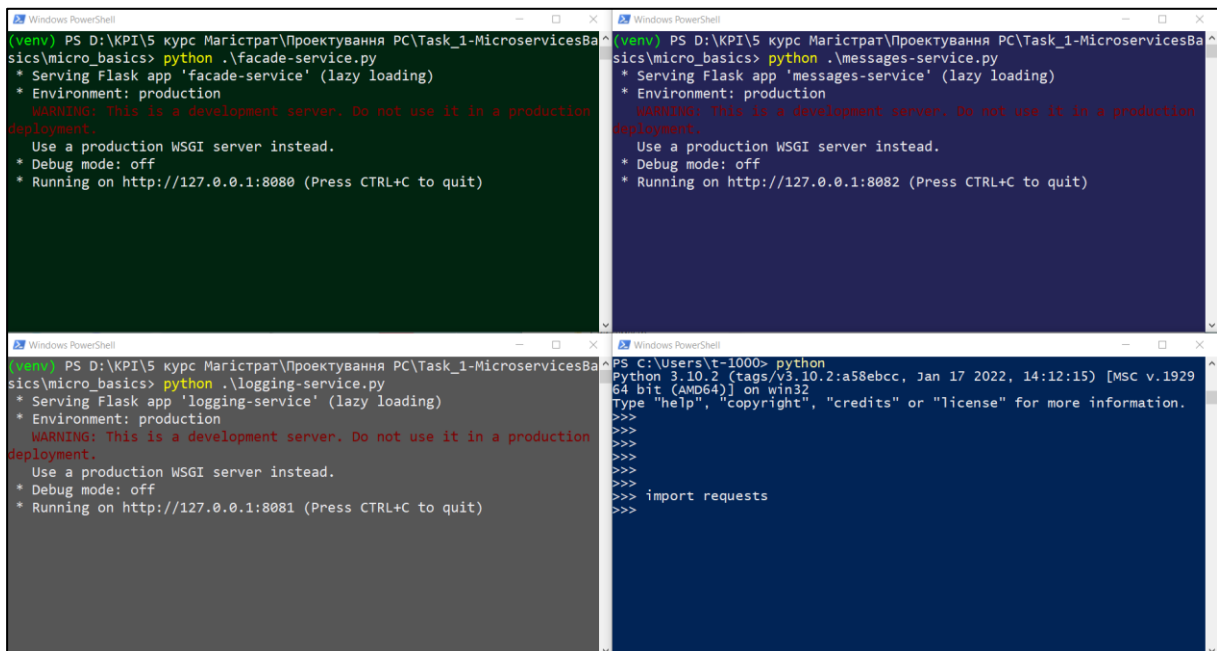
Мета роботи: Це завдання є основою для наступних завдань, у яких будуть додаватись додаткові програмні засоби та функціональність. Необхідно реалізувати три мікросервіси, взаємодія між якими відбувається на основі HTTP-протоколу, ґрунтуючись на базі парадигми REST API (або gRPC).

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.

Посилання на GitHub з проектом, що містить вихідні коди трьох мікросервісів:
[https://github.com/sazan24/KPI/blob/main/Master's%20degree/distributed systems design/micro basics/](https://github.com/sazan24/KPI/blob/main/Master's%20degree/distributed%20systems%20design/micro%20basics/)

Локальний запуск трьох мікросервісів на портах 8080, 8081 та 8082 :



```
(venv) PS D:\KPI\5 курс Марістрат\Проектування PC\Task_1-MicroservicesBasics\micro_basics> 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)
```

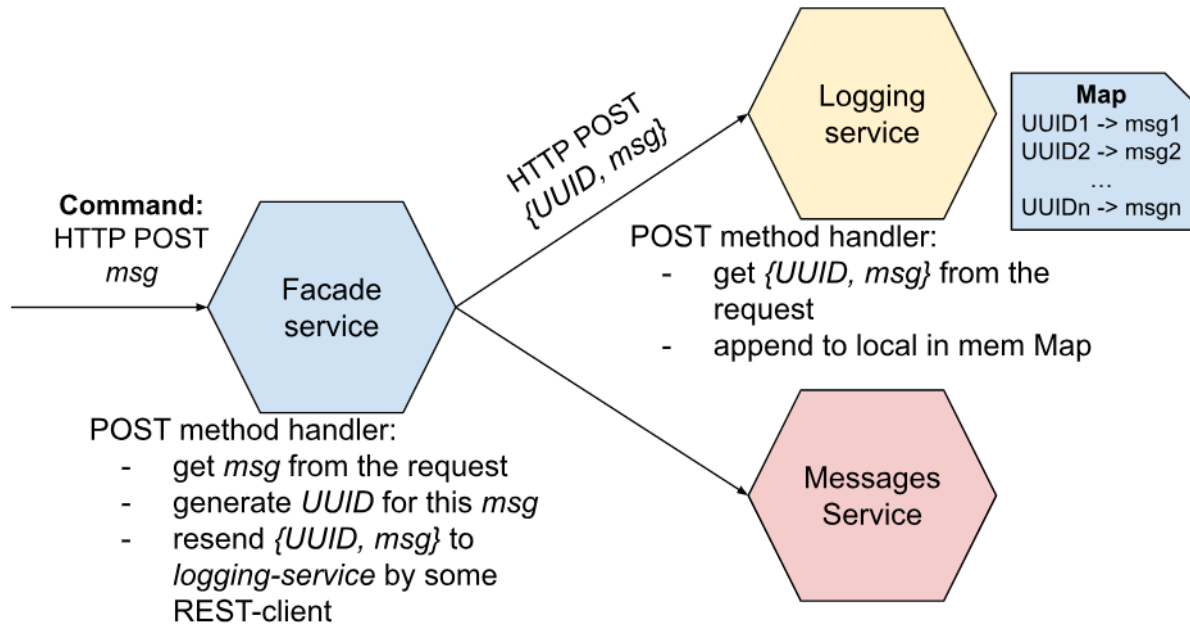
```
(venv) PS D:\KPI\5 курс Марістрат\Проектування PC\Task_1-MicroservicesBasics\micro_basics> python .\messages-service.py
* Serving Flask app 'messages-service' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production 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 Марістрат\Проектування PC\Task_1-MicroservicesBasics\micro_basics> python .\logging-service.py
* 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)
```

```
PS C:\Users\t-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
>>>
```

Частина 1. Базова функціональність системи

- Опис HTTP-POST Request Flow



Нижче наведено процес обробки POST-запитів у системі трьох мікросервісів:

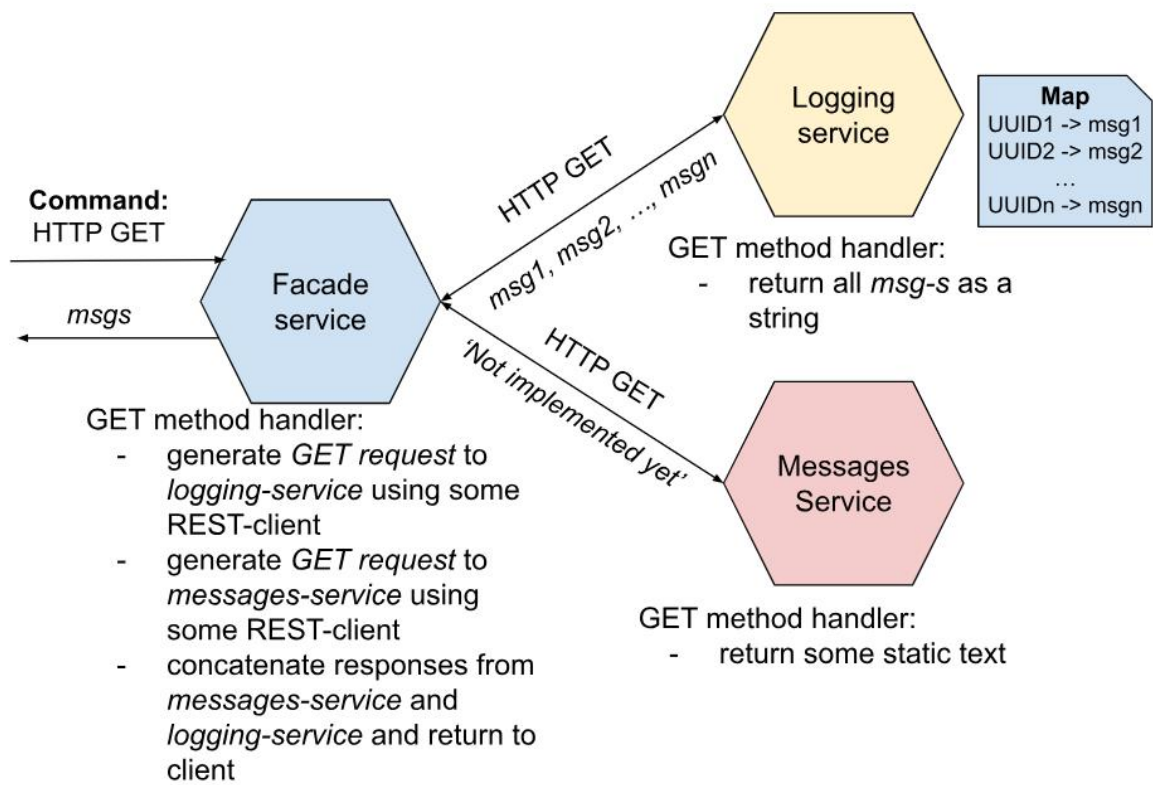
```
Windows PowerShell
* Running on http://127.0.0.1:8080 (Press CTRL+C to quit)
127.0.0.1 - - [01/Apr/2025 00:08:17] "POST /facade HTTP/1.1" 400 -
127.0.0.1 - - [01/Apr/2025 00:08:37] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [01/Apr/2025 00:09:13] "POST /facade HTTP/1.1" 201 -
127.0.0.1 - - [01/Apr/2025 00:10:52] "POST /facade HTTP/1.1" 201 -

Windows PowerShell
* Running on http://127.0.0.1:8082 (Press CTRL+C to quit)

Windows PowerShell
* Running on http://127.0.0.1:8081 (Press CTRL+C to quit)
- Message Was Logged: Hello 'Slack!'
127.0.0.1 - - [01/Apr/2025 00:08:37] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: Tweet...Tw331^!
127.0.0.1 - - [01/Apr/2025 00:09:13] "POST /logging HTTP/1.1" 201 -
- Message Was Logged: Icu and Else ^.^
127.0.0.1 - - [01/Apr/2025 00:10:52] "POST /logging HTTP/1.1" 201 -

Windows PowerShell
>>> res = requests.post("http://localhost:8080/facade", )
>>> print(f"--> {res.json()}t{res.status_code}")
--> {'error': 'No Message in Request'} 400
>>>
>>> res = requests.post("http://localhost:8080/facade", \
... data={"msg": "Hello 'Slack'!"})
>>> print(f"--> {res.json()}t{res.status_code}")
--> {'message': {'msg': 'Hello 'Slack'!', 'uuid': '523eed8c-9086-45ae-bece-d83d3d3539b6'}, 'response': {'success': 'Message Was Logged'}} 201
>>>
>>> res = requests.post("http://localhost:8080/facade", \
... data={"msg": "Tweet...Tw331^!"})
>>> print(f"--> {res.json()}t{res.status_code}")
--> {'message': {'msg': 'Tweet...Tw331^!', 'uuid': '74d2eea5-fd03-450e-953d-efc782235308'}, 'response': {'success': 'Message Was Logged'}} 201
>>>
>>> res = requests.post("http://localhost:8080/facade", \
... data={"msg": "Icu and Else ^.^"})
>>> print(f"--> {res.json()}t{res.status_code}")
--> {'message': {'msg': 'Icu and Else ^.^', 'uuid': '9ec43a5f-eea5-448d-bc0a-534e2c72562c'}, 'response': {'success': 'Message Was Logged'}} 201
>>>
```

- Опис HTTP-GET Request Flow



Нижче наведено процес обробки GET-запитів у системі трьох мікросервісів:

```

127.0.0.1 - - [01/Apr/2025 12:00:32] "GET /logging HTTP/1.1" 404 -
127.0.0.1 - - [01/Apr/2025 12:00:35] "GET /messages HTTP/1.1" 404 -
127.0.0.1 - - [01/Apr/2025 12:00:47] "GET /facade HTTP/1.1" 200 -

127.0.0.1 - - [01/Apr/2025 12:00:37] "GET /logging HTTP/1.1" 200 -
>>> res = requests.get("http://localhost:8080/logging")
>>> res = requests.get("http://localhost:8080/messages")
>>> res = requests.get("http://localhost:8080/facade")
>>> print(f"--> {res.json()}\t{res.status_code}")
--> ['logging-service: ["Hello \'Slack\'!', "Tweet...Tw33i^!", "IcU and ElsE ^_^"]\n', "messages-service: I'm not implemented yet ^_^"] 200
>>>

127.0.0.1 - - [01/Apr/2025 13:49:39] "GET /logging HTTP/1.1" 200 -
>>> res = requests.get("http://localhost:8081/logging")
>>> print(f"--> {res.json()}\t{res.status_code}")
--> ["Hello 'Slack'!", 'Tweet...Tw33i^!', 'IcU and ElsE ^_^'] 200
>>> res = requests.get("http://localhost:8082/messages")
>>> print(f"--> {res.text}\t{res.status_code}")
--> I'm not implemented yet ^_^ 200
>>>
  
```

Частина 2. Додатковий функціонал системи

❖ Наявність механізму retry та deduplication:

- При затримці зв'язку або відсутності відповіді від *logging-service* (або *messages-service*), буде застосовано механізм retry, який спробує повторити цю саму операцію (GET або POST) по передачі повідомлення:
 - POST** (`sending_with_retry`)

Щоб перевірити на працездатність механізм retry було призупинено *logging-service*, тим самим це дало змогу явно зафіксувати дві повторні спроби надсилання повідомлення після першої невдачі із часовими затримками:

```
W1 POST-request with {'uid': 'ccd0927d-1768-4cba-a42c-9f1d66c316fb', 'msg': 'Spirit MSG t0 LOG'} at 16:09:41
↑ RetryError: HTTPConnectionPool(host='localhost', port=8081): Max retries exceeded with url: /logging (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x000001AB2588DC60>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))

W2 POST-request with {'uid': 'ccd0927d-1768-4cba-a42c-9f1d66c316fb', 'msg': 'Spirit MSG t0 LOG'} at 16:09:47
↑ RetryError: HTTPConnectionPool(host='localhost', port=8081): Max retries exceeded with url: /logging (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x000001AB2588E440>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))

W3 POST-request with {'uid': 'ccd0927d-1768-4cba-a42c-9f1d66c316fb', 'msg': 'Spirit MSG t0 LOG'} at 16:09:54
↑ RetryError: HTTPConnectionPool(host='localhost', port=8081): Max retries exceeded with url: /logging (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x000001AB2588DC60>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))

127.0.0.1 - - [01/Apr/2025 16:09:55] "POST /facade HTTP/1.1" 500 -

PS D:\KPI\5 курс Марістрат\Проектування PC\Task_1-MicroservicesBasics\micro_basics>
PS D:\KPI\5 курс Марістрат\Проектування PC\Task_1-MicroservicesBasics\micro_basics>

>>> res = requests.post("http://localhost:8080/facade", \
... data={"msg": "Spirit MSG t0 LOG"})
>>>
```

Щоб змодельовати ситуацію, коли запит все ж буде надіслано після пари спроб, то було перезапущено *logging-service* в моменті повторних POST-запитів:

```
W1 POST-request with {'uid': 'c47c09da-03b5-41ec-a8d9-e44a5f660515', 'msg': 'SigmaN G0T Acc33$'} at 16:17:17
↑ RetryError: HTTPConnectionPool(host='localhost', port=8081): Max retries exceeded with url: /logging (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x000001AB2588F520>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))

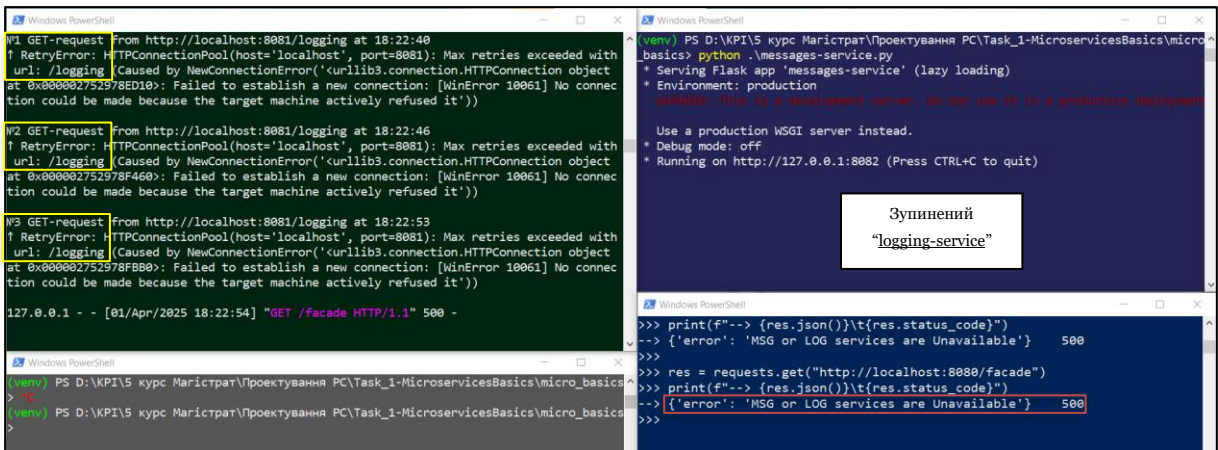
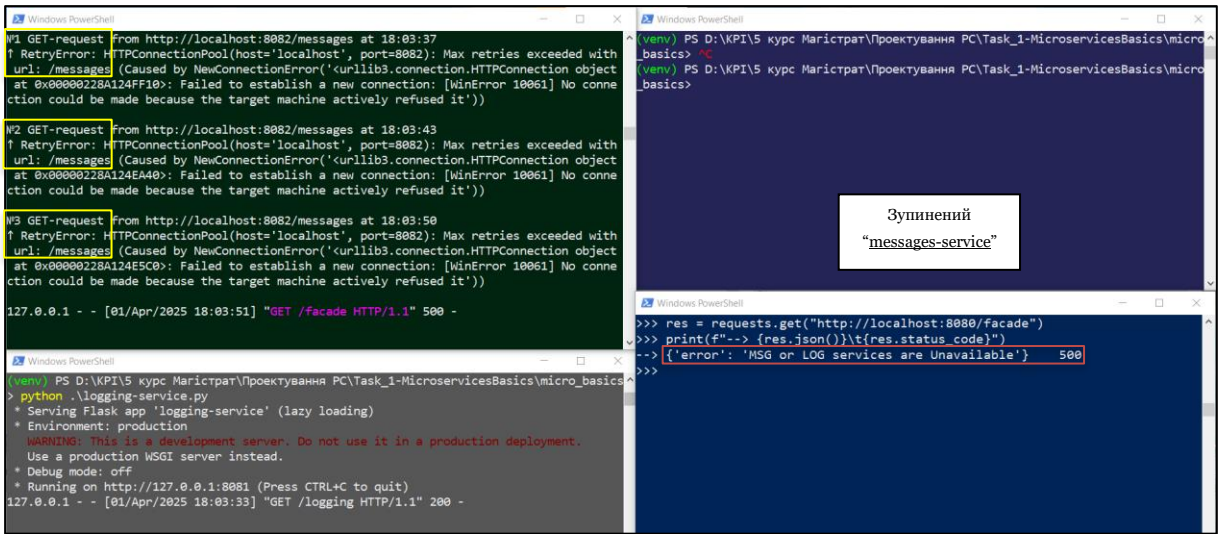
W2 POST-request with {'uid': 'c47c09da-03b5-41ec-a8d9-e44a5f660515', 'msg': 'SigmaN G0T Acc33$'} at 16:17:23
↑ RetryError: HTTPConnectionPool(host='localhost', port=8081): Max retries exceeded with url: /logging (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x000001AB2588FC70>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))

127.0.0.1 - - [01/Apr/2025 16:17:26] "POST /facade HTTP/1.1" 201 -

PS D:\KPI\5 курс Марістрат\Проектування PC\Task_1-MicroservicesBasics\micro_basics>
python .\logging-service.py
* 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)
Message Was Logged: SigmaN G0T Acc33$
127.0.0.1 - - [01/Apr/2025 16:17:26] "POST /logging HTTP/1.1" 201 -

>>> res = requests.post("http://localhost:8080/facade", \
... data={"msg": "Spirit MSG t0 LOG"})
>>>
>>> res = requests.post("http://localhost:8080/facade", \
... data={"msg": "SigmaN G0T Acc33$"})
>>>
```


- **GET**(receiving_with_retry)



- Водночас на *logging-service* для повідомлень застосовується механізм **deduplication** (*exactly once delivery*), з метою перевірки того, що одне й те саме повідомлення не повторилось декілька разів під час надсилання.

