Task1 - Базова архітектура мікросервісів

Сотнікова Поліна ФБ-41мп

Посилання на Гітхаб репозиторій для 3 мікросервісів: *micro_basics*

1) facade-service.py:

```
from flask import Flask, request, jsonify
import requests
import uuid

app = Flask(__name__)

LOG_SERVICE_URL = "http://localhost:5001/logs"
MSG_SERVICE_URL = "http://localhost:5002/messages"

def send_to_logging_service(msg_id, content):
    payload = {"id": msg_id, "content": content}
    response = requests.post(LOG_SERVICE_URL, json=payload)
    return response

def fetch_service_data():
    logs = requests.get(LOG_SERVICE_URL).json()
    messages = requests.get(MSG_SERVICE_URL).json()
    return logs, messages

@app.route("/", methods=["POST", "GET"])
def process_request():
    if request.method == "POST":
        content = request.json.get("message")
        if not content:
            return jsonify({"error": "No message received"}), 400

        message_id = str(uuid.uuid4())
        log_response = send_to_logging_service(message_id, content)

        if log_response.status_code not in range(200, 300):
            return jsonify({"error": "Failed to log message"}), 500

        return jsonify({
            "status": "Message logged",
            "message_id": message_id,
            "content": content
        }), 201

    elif request.method == "GET":
        logs, messages = fetch_service_data()
        return jsonify({"logs": logs, "messages": messages}), 200

if __name__ == "__main__":
    app.run(port=5000)
```

2) logging-service:

```
from flask import Flask, request, jsonify

app = Flask(__name__)
log_storage = {}

@app.route("/logs", methods=["POST", "GET"])
def handle_logs():
    if request.method == "POST":
        data = request.json
```

```
        log_id = data.get("id")
        content = data.get("content")

        if not log_id or not content:
            return jsonify({"error": "Missing log ID or content"}), 400

        log_storage[log_id] = content
        print(f"[LOGGED] {content}")
        return jsonify({"message": "Log entry saved"}), 201

    elif request.method == "GET":
        return jsonify(log_storage), 200

if __name__ == "__main__":
    app.run(port=5001)
```

3) message-service.py:

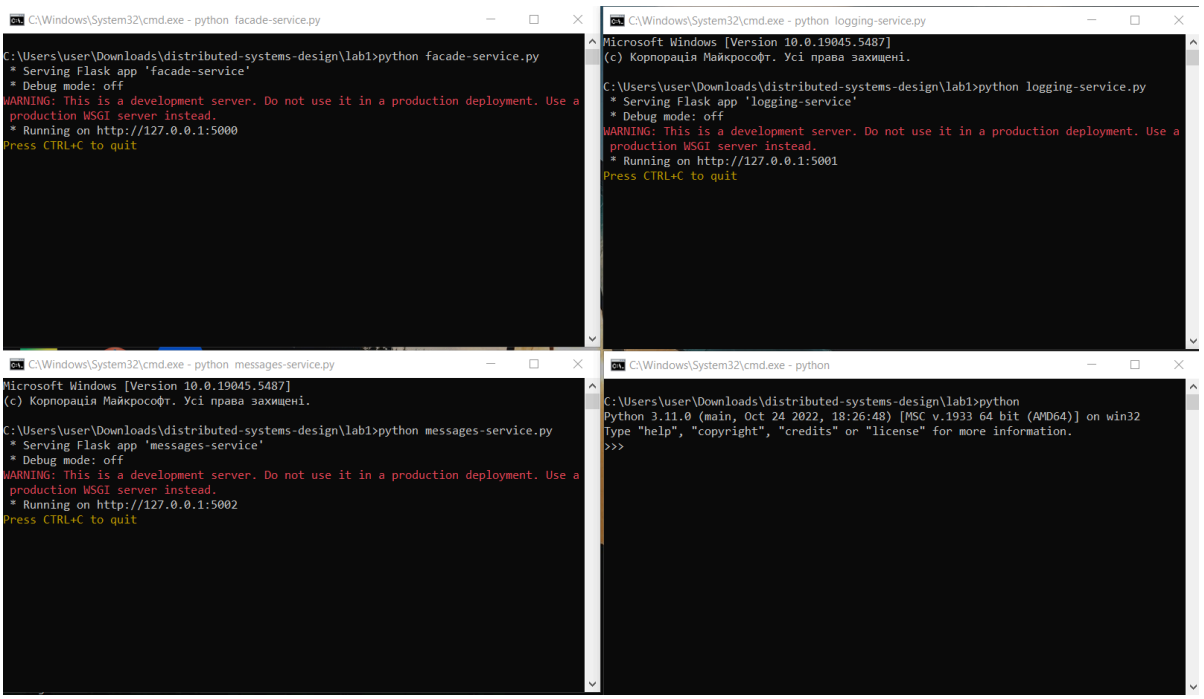
```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route("/messages", methods=["GET"])
def fetch_messages():
    return jsonify({"message": "Feature not implemented yet"}), 200

if __name__ == "__main__":
    app.run(port=5002)
```

Відкриваємо консоль, запускаємо мікросервіси:



POST-запити:

```
C:\Windows\System32\cmd.exe - python facade-service.py

C:\Users\user\Downloads\distributed-systems-design\lab1>python facade-service.py
 * Serving Flask app 'facade-service'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [02/Mar/2025 23:35:05] "POST / HTTP/1.1" 201 -
127.0.0.1 - - [02/Mar/2025 23:35:10] "POST / HTTP/1.1" 201 -
127.0.0.1 - - [02/Mar/2025 23:35:16] "POST / HTTP/1.1" 201 -
```

```
C:\Windows\System32\cmd.exe - python logging-service.py

C:\Users\user\Downloads\distributed-systems-design\lab1>python logging-service.py
 * Serving Flask app 'logging-service'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI serve
r instead.
 * Running on http://127.0.0.1:5001
Press CTRL+C to quit
[LOGGED] test message 1
127.0.0.1 - - [02/Mar/2025 23:35:05] "POST /logs HTTP/1.1" 201 -
[LOGGED] test message 2
127.0.0.1 - - [02/Mar/2025 23:35:10] "POST /logs HTTP/1.1" 201 -
[LOGGED] test message 3
127.0.0.1 - - [02/Mar/2025 23:35:16] "POST /logs HTTP/1.1" 201 -
```

```
C:\Windows\System32\cmd.exe - python messages-service.py

C:\Users\user\Downloads\distributed-systems-design\lab1>python messages-service.py
 * Serving Flask app 'messages-service'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
 * Running on http://127.0.0.1:5002
Press CTRL+C to quit
```

```
C:\Windows\System32\cmd.exe - python

C:\Users\user\Downloads\distributed-systems-design\lab1>python
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> import json
>>> response = requests.post('http://localhost:5000', json={'message': 'test message 1'})
>>> print(response.text)
{"content":"test message 1","message_id":"b0948a70-9d8b-4a9b-b395-72150de51d86","status":"Message logged"}

>>> response = requests.post('http://localhost:5000', json={'message': 'test message 2'})
>>> print(response.text)
{"content":"test message 2","message_id":"46ecd6c3-5987-4cf6-956a-7971f86ebba3","status":"Message logged"}

>>> response = requests.post('http://localhost:5000', json={'message': 'test message 3'})
>>> print(response.text)
{"content":"test message 3","message_id":"2b413660-502c-495e-afca-1f727df2786a","status":"Message logged"}

>>>
```

GET-запити:

```
C:\Windows\System32\cmd.exe - python facade-service.py

C:\Users\user\Downloads\distributed-systems-design\lab1>python facade-service.py
 * Serving Flask app 'facade-service'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [02/Mar/2025 23:35:05] "POST / HTTP/1.1" 201 -
127.0.0.1 - - [02/Mar/2025 23:35:10] "POST / HTTP/1.1" 201 -
127.0.0.1 - - [02/Mar/2025 23:35:16] "POST / HTTP/1.1" 201 -
127.0.0.1 - - [02/Mar/2025 23:35:46] "GET / HTTP/1.1" 200 -
```

```
C:\Windows\System32\cmd.exe - python logging-service.py

C:\Users\user\Downloads\distributed-systems-design\lab1>python logging-service.py
 * Serving Flask app 'logging-service'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI serve
r instead.
 * Running on http://127.0.0.1:5001
Press CTRL+C to quit
[LOGGED] test message 1
127.0.0.1 - - [02/Mar/2025 23:35:05] "POST /logs HTTP/1.1" 201 -
[LOGGED] test message 2
127.0.0.1 - - [02/Mar/2025 23:35:10] "POST /logs HTTP/1.1" 201 -
[LOGGED] test message 3
127.0.0.1 - - [02/Mar/2025 23:35:16] "POST /logs HTTP/1.1" 201 -
127.0.0.1 - - [02/Mar/2025 23:35:46] "GET /logs HTTP/1.1" 200 -
```

```
C:\Windows\System32\cmd.exe - python messages-service.py

C:\Users\user\Downloads\distributed-systems-design\lab1>python messages-service.py
 * Serving Flask app 'messages-service'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
 * Running on http://127.0.0.1:5002
Press CTRL+C to quit
127.0.0.1 - - [02/Mar/2025 23:35:46] "GET /messages HTTP/1.1" 200 -
```

```
C:\Windows\System32\cmd.exe - python

C:\Users\user\Downloads\distributed-systems-design\lab1>python
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> import json
>>> response = requests.post('http://localhost:5000', json={'message': 'test message 1'})
>>> print(response.text)
{"content":"test message 1","message_id":"b0948a70-9d8b-4a9b-b395-72150de51d86","status":"Message logged"}

>>> response = requests.post('http://localhost:5000', json={'message': 'test message 2'})
>>> print(response.text)
{"content":"test message 2","message_id":"46ecd6c3-5987-4cf6-956a-7971f86ebba3","status":"Message logged"}

>>> response = requests.post('http://localhost:5000', json={'message': 'test message 3'})
>>> print(response.text)
{"content":"test message 3","message_id":"2b413660-502c-495e-afca-1f727df2786a","status":"Message logged"}

>>> response = requests.get('http://localhost:5000/')
>>> print(response.json())
{'logs': {'2b413660-502c-495e-afca-1f727df2786a': 'test message 3', '46ecd6c3-5987-4cf6-956a-7971f86ebba3':
'test message 2', 'b0948a70-9d8b-4a9b-b395-72150de51d86': 'test message 1'}, 'messages': {'message': 'Featur
e not implemented yet'}}
>>>
```

Додаткове завдання:

Змінений код для **retry mechanism** + **deduplication**:

1) facade-service.py:

```
from flask import Flask, request, jsonify
import requests
import uuid
import time

app = Flask(__name__)

LOGGING_SERVICE_URL = "http://localhost:5001/log"

def send_with_retry(msg_id, content, retries=3, delay=2):
    """Відправка POST-запиту з повторенням у разі невдачі та логуванням у консоль"""
    for attempt in range(retries):
        try:
            print(f"[{attempt + 1}/{retries}] Sending log {msg_id} to
{LOGGING_SERVICE_URL}...")
            response = requests.post(LOGGING_SERVICE_URL, json={"id": msg_id, "content":
content}, timeout=5)

            if response.status_code in [200, 201]:  # 200 (OK), 201 (Created)
                print(f"[SUCCESS] Log {msg_id} saved successfully.")
                return response.json()
            else:
                print(f"[ERROR] Log {msg_id} failed with status {response.status_code}:
{response.text}")

        except requests.exceptions.RequestException as e:
            print(f"[{attempt + 1}/{retries}] Failed to send log {msg_id}: {e}")

        time.sleep(delay)  # Затримка перед наступною спробою

    print(f"[FAILED] Log {msg_id} could not be saved after {retries} retries.")
    return {"error": "Failed to log message after retries"}

@app.route("/", methods=["POST", "GET"])
def process_request():
    if request.method == "POST":
        data = request.get_json()
        if not data or "message" not in data:
            return jsonify({"error": "No message received"}), 400

        message_id = str(uuid.uuid4())
        log_status = send_with_retry(message_id, data["message"])

        return jsonify({
            "status": log_status,
            "message_id": message_id,
            "content": data["message"]
        }), 201

    elif request.method == "GET":
        try:
            response = requests.get("http://localhost:5001/logs", timeout=5)
            return jsonify(response.json()), response.status_code
        except requests.exceptions.RequestException as e:
            return jsonify({"error": f"Failed to fetch logs - {e}"}), 500

if __name__ == "__main__":
    app.run(port=5000)
```

2) logging-service.py:

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

app = Flask(__name__)
log_storage = {}  # {log_id: content}

@app.route("/log", methods=["POST"])
def log_message():
    data = request.get_json()
    log_id = data.get("id")
    content = data.get("content")

    if not log_id or not content:
        return jsonify({"error": "Invalid log entry"}), 400

    if log_id in log_storage:  # Перевіряємо, чи такий log_id вже існує
        return jsonify({"message": "Duplicate log entry ignored"}), 409

    log_storage[log_id] = content
    print(f"[LOGGED] {content}")
    return jsonify({"message": "Log entry saved"}), 201

@app.route("/logs", methods=["GET"])
def get_logs():
    return jsonify(log_storage), 200

if __name__ == "__main__":
    app.run(port=5001)
```

3) message-service.py: без змін

Механізм retry:

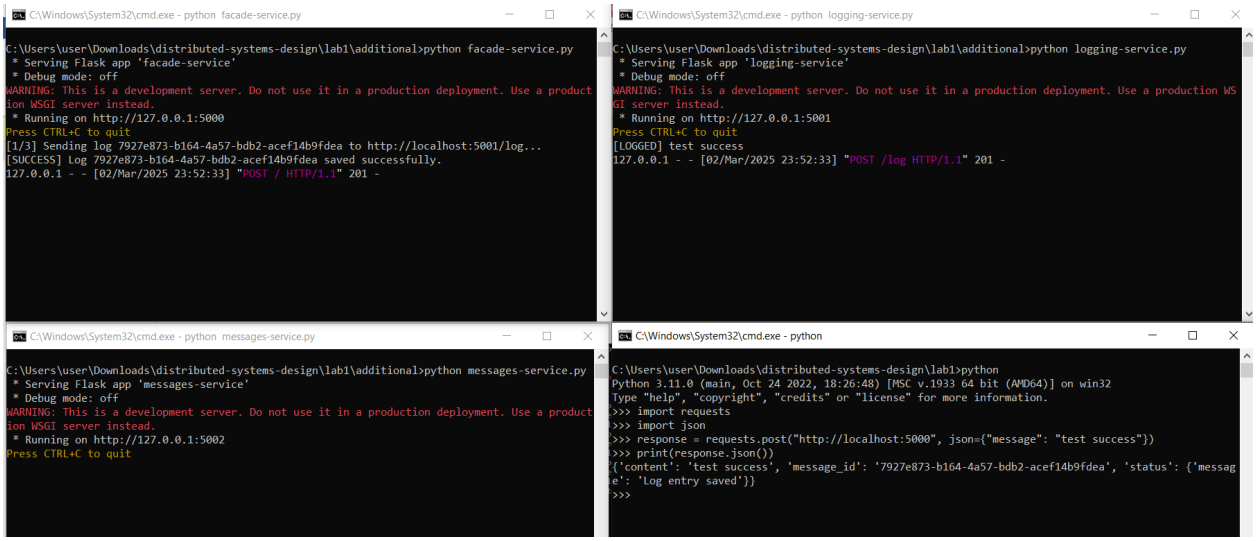
Відсутність зв’язку з logging-service:

```
C:\Windows\System32\cmd.exe - python facade-service.py
C:\Users\user\Downloads\distributed-systems-design\lab1\additional>python facade-service.py
 * Serving Flask app 'facade-service'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
[1/3] Sending log 0f9cf2e0-ec34-46e1-b447-8a172ab46e1a to http://localhost:5001/log...
[1/3] Failed to send log 0f9cf2e0-ec34-46e1-b447-8a172ab46e1a: HTTPConnectionPool(host='localhost', port=5001): Max retries exceeded with url: /log (Caused by NewConnectionError('<urllib3.c
onnection.HTTPConnection object at 0x0000020460D9F110>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))
[2/3] Sending log 0f9cf2e0-ec34-46e1-b447-8a172ab46e1a to http://localhost:5001/log...
[2/3] Failed to send log 0f9cf2e0-ec34-46e1-b447-8a172ab46e1a: HTTPConnectionPool(host='localhost', port=5001): Max retries exceeded with url: /log (Caused by NewConnectionError('<urllib3.c
onnection.HTTPConnection object at 0x0000020461A37450>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))
[3/3] Sending log 0f9cf2e0-ec34-46e1-b447-8a172ab46e1a to http://localhost:5001/log...
[3/3] Failed to send log 0f9cf2e0-ec34-46e1-b447-8a172ab46e1a: HTTPConnectionPool(host='localhost', port=5001): Max retries exceeded with url: /log (Caused by NewConnectionError('<urllib3.c
onnection.HTTPConnection object at 0x0000020461A41C50>: Failed to establish a new connection: [WinError 10061] No connection could be made because the target machine actively refused it'))
[FAILED] Log 0f9cf2e0-ec34-46e1-b447-8a172ab46e1a could not be saved after 3 retries.
127.0.0.1 - - [02/Mar/2025 23:50:36] "POST / HTTP/1.1" 201 -
```

```
C:\Windows\System32\cmd.exe - python

C:\Users\user\Downloads\distributed-systems-design\lab1>python
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> response = requests.post("http://localhost:5000", json={"message": "test retry"})
>>> print(response.json())
{'content': 'test retry', 'message_id': '0f9cf2e0-ec34-46e1-b447-8a172ab46e1a', 'status': {'error': 'Failed to log message after retries'}}
>>>
```

Запускаємо logging-service:



<u>Deduplication для повідомлень:</u>