Проектування розподілених систем

Task 1 - Базова архітектура мікросервісів

Мартиненко Денис ФБ-42мп

facade-service

У випадку POST-запиту facade сервіс отримує текст повідомлення, генерує id та надсилає POST-запит з повідомленням logging сервісу.

У випадку GET-запиту facade сервіс виконує два GET-запити до logging та messages сервісів.

Також цей код спробує зробити POST-запит до logging-service 3 рази з інтервалом 2 секунди, якщо виникне помилка.

```
import requests, uuid
from flask import Flask, request, jsonify
from tenacity import retry, stop_after_attempt, wait_fixed

app = Flask(__name__)
logging_service = 'http://localhost:8881/logging'
messages_service = 'http://localhost:8882/messages'

@retry(stop=stop_after_attempt(3), wait=wait_fixed(2))
def send_with_retry(url, data):
    response = requests.post(url, data=data)
    if response.status_code != 201:
        raise Exception('Logging service did not accept message.')
    return response

@app.route('/', methods=['POST', 'GET'])
def home():
    if request.method == 'POST':
        txt = request.form.get('txt')
        if not txt:
            return jsonify('No message'), 400
        id = str(uuid.uuid4())
        msg = {'id': id, 'txt': txt}
        try:
            send_with_retry(logging_service, msg)
            return jsonify(msg), 200
        except Exception as e:
            return jsonify({'error': str(e)}), 500

    elif request.method == 'GET':
        try:
            logging_response = requests.get(logging_service)
            messages_response = requests.get(messages_service)
```

```
            combined_response = [
                f'logging-service: {logging_response.text}',
                f'message-service: {messages_response.text}'
            ]
            return jsonify(combined_response), 200
        except Exception as e:
            return jsonify({'error': str(e)}), 500

if __name__ == '__main__':
    app.run(port=8880)
```

logging-service

У випадку POST-запиту logging сервіс отримує id та текст повідомлення і зберігає його.

У випадку GET-запиту logging сервіс повертає текст всіх збережених повідомлень.

Також цей код запобігає повторному логуванню повідомлень із тим самим UUID.

```
from flask import Flask, request, jsonify

app = Flask(__name__)
msg = {}

@app.route('/logging', methods=['POST', 'GET'])
def home():
    if request.method == 'POST':
        id, txt = request.form.get('id'), request.form.get('txt')
        if not (id and txt):
            return jsonify('No id or text'), 400

        if id in msg:
            print(f'Duplicate message detected: {id}')
            return jsonify('Duplicate detected'), 200

        msg[id] = txt
        print('Message:', txt)
        return jsonify('Message logged'), 201

    elif request.method == 'GET':
        return jsonify(list(msg.values()))

if __name__ == '__main__':
    app.run(port=8881)
```

messages-service

У випадку GET-запиту messages сервіс просто повертає текст 'Message-service not implemented yet'.

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/messages', methods=['GET'])

def home():
    if request.method == 'GET':
        return 'Message-service not implemented yet'
    else:
        return jsonify('Method not allowed'), 405

if __name__ == '__main__':
    app.run(port=8882)
```

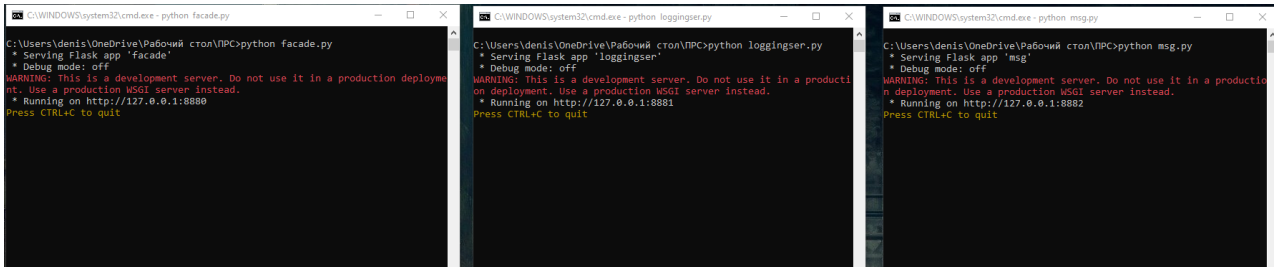
logging-service через gRPC:

```
import grpc
from concurrent import futures
import logging_pb2, logging_pb2_grpc

msg = {}

class LoggingService(logging_pb2_grpc.LoggingServiceServicer):
    def LogMessage(self, request, context):
        if request.id in msg:
            return logging_pb2.LogReply(status='Duplicate')
        msg[request.id] = request.txt
        print('Message:', request.txt)
        return logging_pb2.LogReply(status='Logged')

    def GetMessages(self, request, context):
        return logging_pb2.MessagesReply(messages=list(msg.values()))

def serve():
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    logging_pb2_grpc.add_LoggingServiceServicer_to_server(LoggingService(),
server)
    server.add_insecure_port('[::]:50051')
    server.start()
    server.wait_for_termination()

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

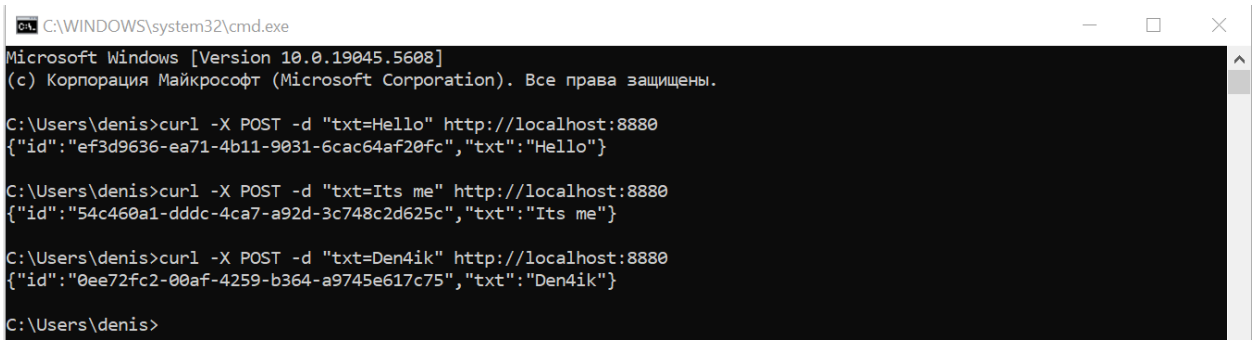
facade-service із gRPC-клієнтом:

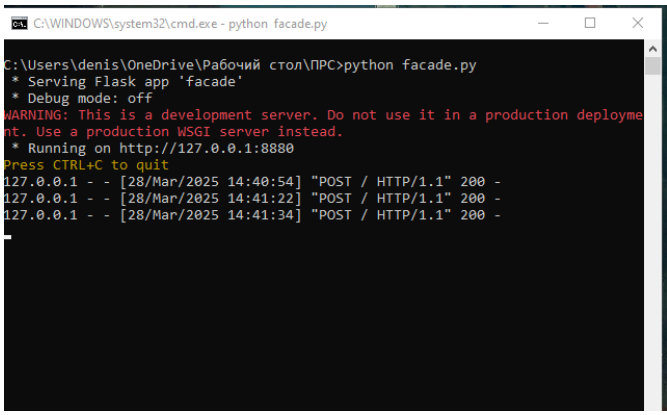
```python
import grpc, uuid
from flask import Flask, request, jsonify
import logging_pb2, logging_pb2_grpc

app = Flask(__name__)
channel = grpc.insecure_channel('localhost:50051')
client = logging_pb2_grpc.LoggingServiceStub(channel)

@app.route('/', methods=['POST', 'GET'])
def home():
    if request.method == 'POST':
        txt = request.form.get('txt')
        if not txt:
            return jsonify('No message'), 400
        id = str(uuid.uuid4())
        response = client.LogMessage(logging_pb2.LogRequest(id=id, txt=txt))
        return jsonify({'id': id, 'status': response.status}), 200

    elif request.method == 'GET':
        response = client.GetMessages(logging_pb2.Empty())
        return jsonify(list(response.messages)), 200

if __name__ == '__main__':
    app.run(port=8880)
```

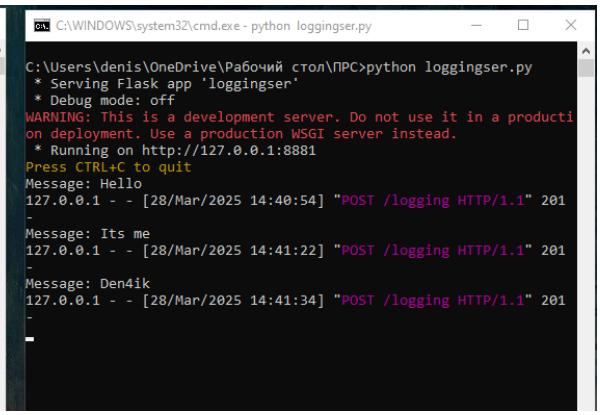
Запуск:

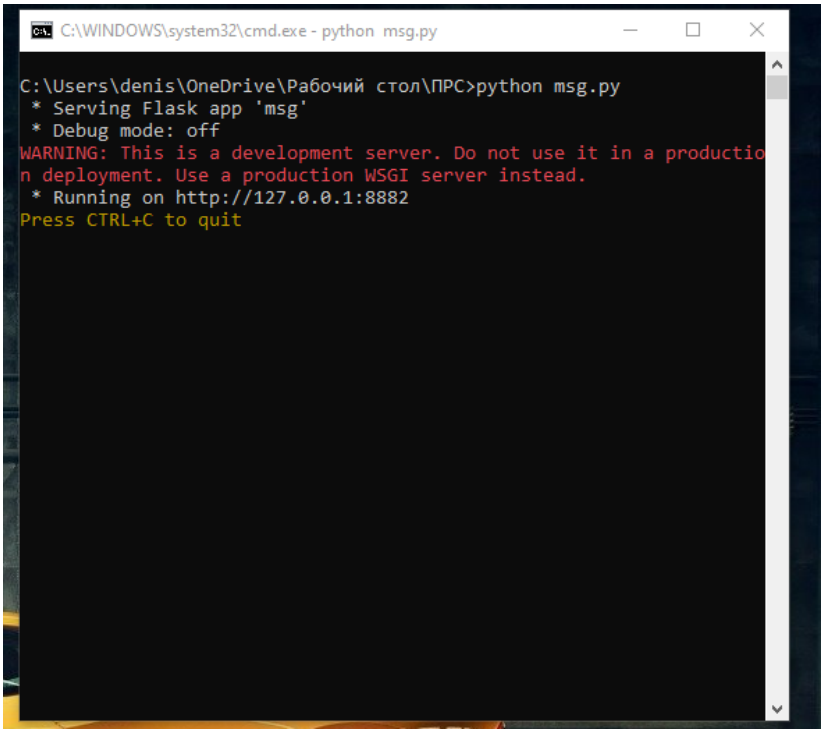


POST request:

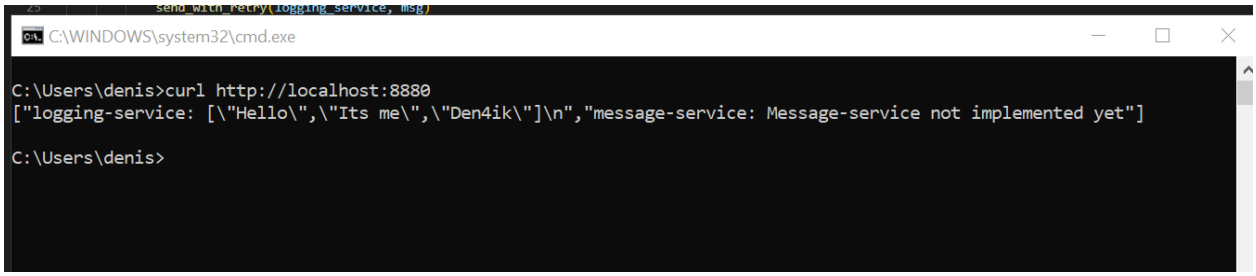


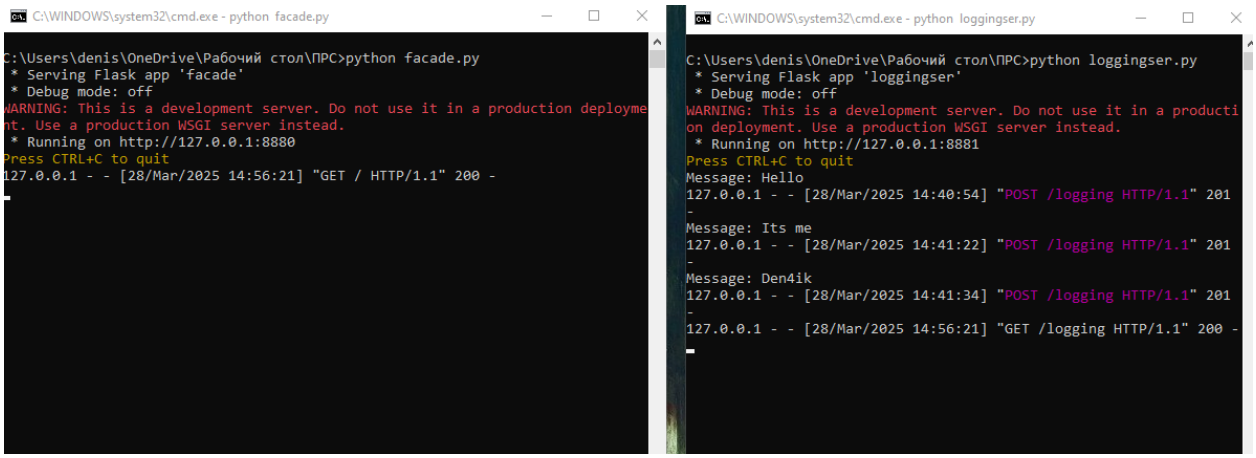


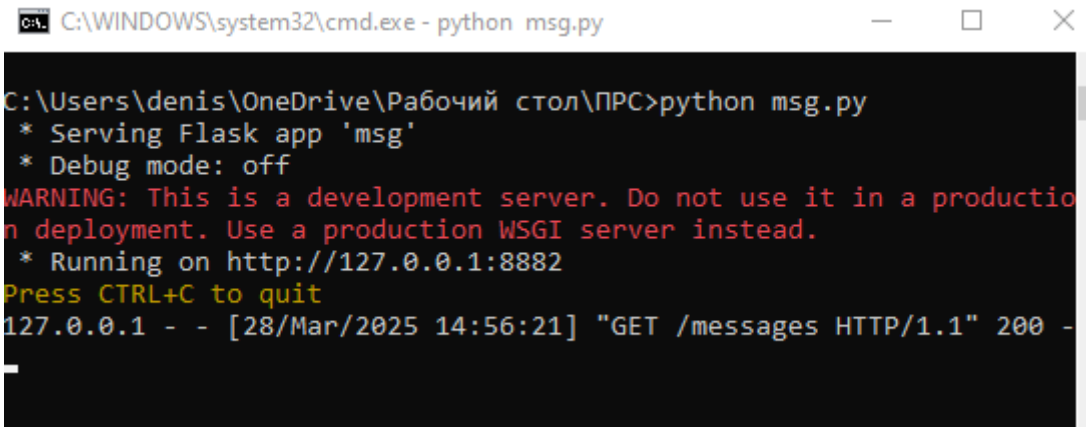




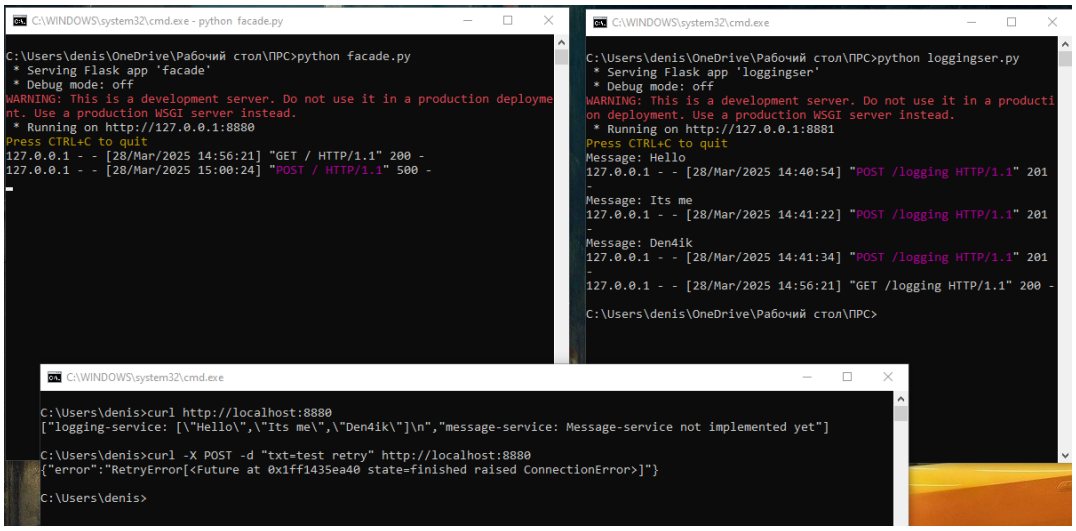
GET request:



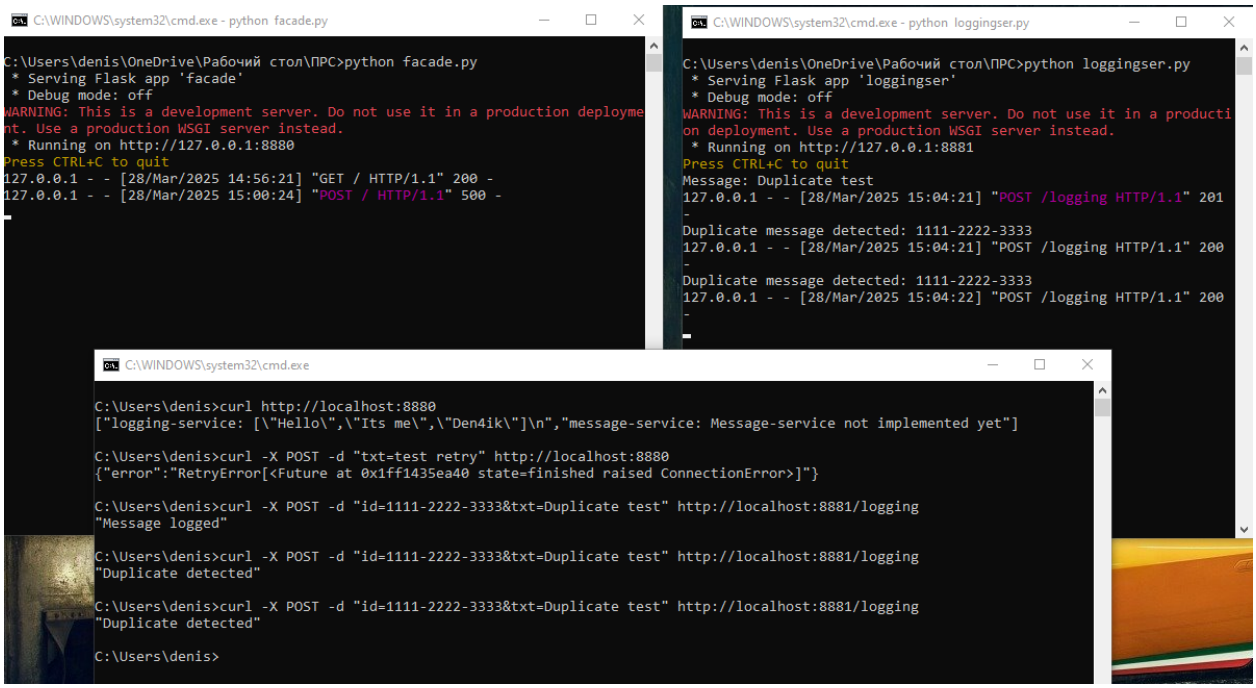




Перевірка retry:



Перевірка dublicate:



Через grpc: