Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

З лабораторної роботи № 3 з дисципліни
«Технології розробки вбудованих IoT системи»

„АДАПТЕРИ У PYTHON. СТВОРЕННЯ ТА ІНТЕГРАЦІЯ ВЛАСНОГО АДАПТЕРА”

Перевірив _____ _____
(прізвище, ім'я, по батькові)

Київ 2023

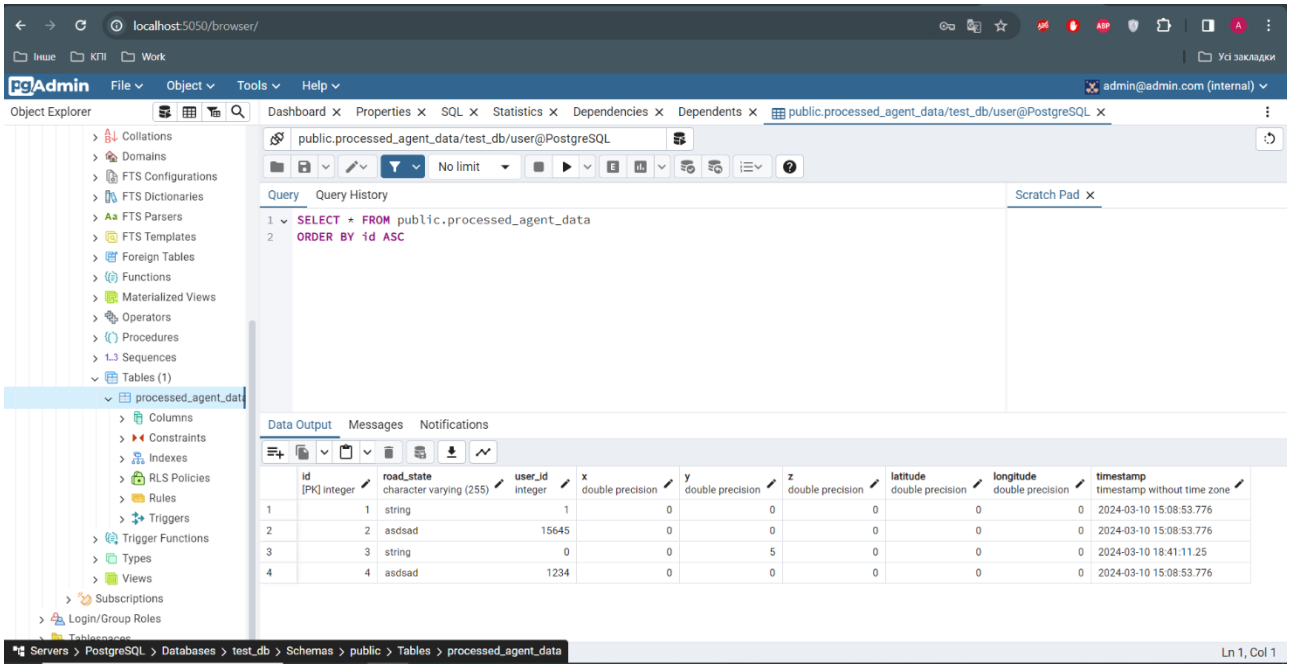
1) Посилання на git repo: https://github.com/faaant/iot

2) Реалізація класу адаптера та методу `save_data` адаптера:

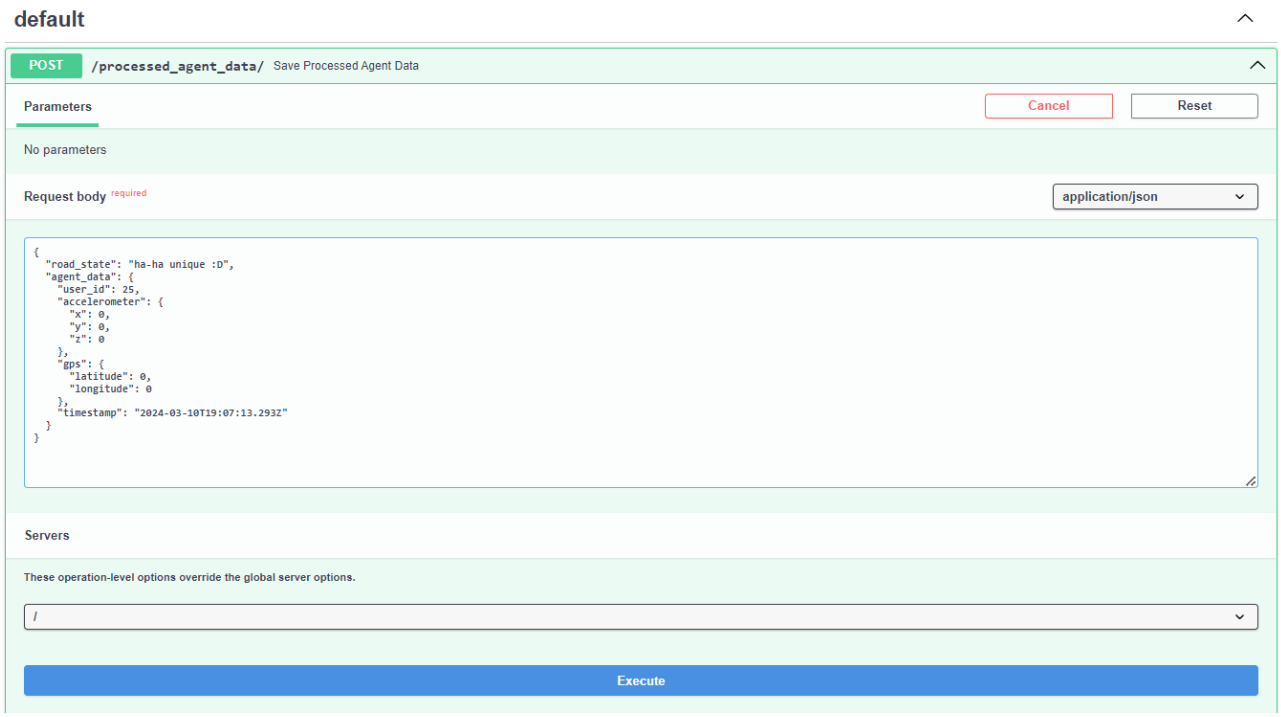
```
 import json
import logging
from typing import List

import pydantic_core
import requests

from app.entities.processed_agent_data import ProcessedAgentData
from app.interfaces.store_gateway import StoreGateway


class StoreApiAdapter(StoreGateway):
    def __init__(self, api_base_url):
        self.api_base_url = api_base_url

    def save_data(self, processed_agent_data_batch: List[ProcessedAgentData]):
        """
        Save the processed road data to the Store API.
        Parameters:
            processed_agent_data_batch (dict): Processed road data to be saved.
        Returns:
            bool: True if the data is successfully saved, False otherwise.
        """
        try:
            # Prepare the data as a list of dictionaries
            data = []
            for item in processed_agent_data_batch:
                # Convert datetime to ISO 8601 format string
                timestamp_isoformat = item.agent_data.timestamp.isoformat()
                # Create a dictionary with timestamp as string
                data_item = item.model_dump()
                data_item['agent_data']['timestamp'] = timestamp_isoformat
                data.append(data_item)

            # Make the POST request
            response = requests.post(f"{self.api_base_url}/processed_agent_data/",
json=data)

            # Check if the request was successful (status code 200)
            if response.status_code == 200 or response.ok :
                return True
            else:
                return False
        except Exception as e:
            logging.info(f"Error occured {e}")
            return False
```

3) Демонстрація результатів роботи в Swagger/MQTT та PgAdmin

pgAdmin початковий стан:



Swagger(hub):



Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:9000/processed_agent_data/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "road_state": "ha-ha unique :D",
  "agent_data": {
    "user_id": 25,
    "accelerometer": {
      "x": 0,
      "y": 0,
      "z": 0
    },
    "gps": {
      "latitude": 0,
      "longitude": 0
    },
    "timestamp": "2024-03-10T19:07:13.293Z"
  }
}'
```

Request URL

```
http://127.0.0.1:9000/processed_agent_data/
```

Server response

Code | Details

200

Response body

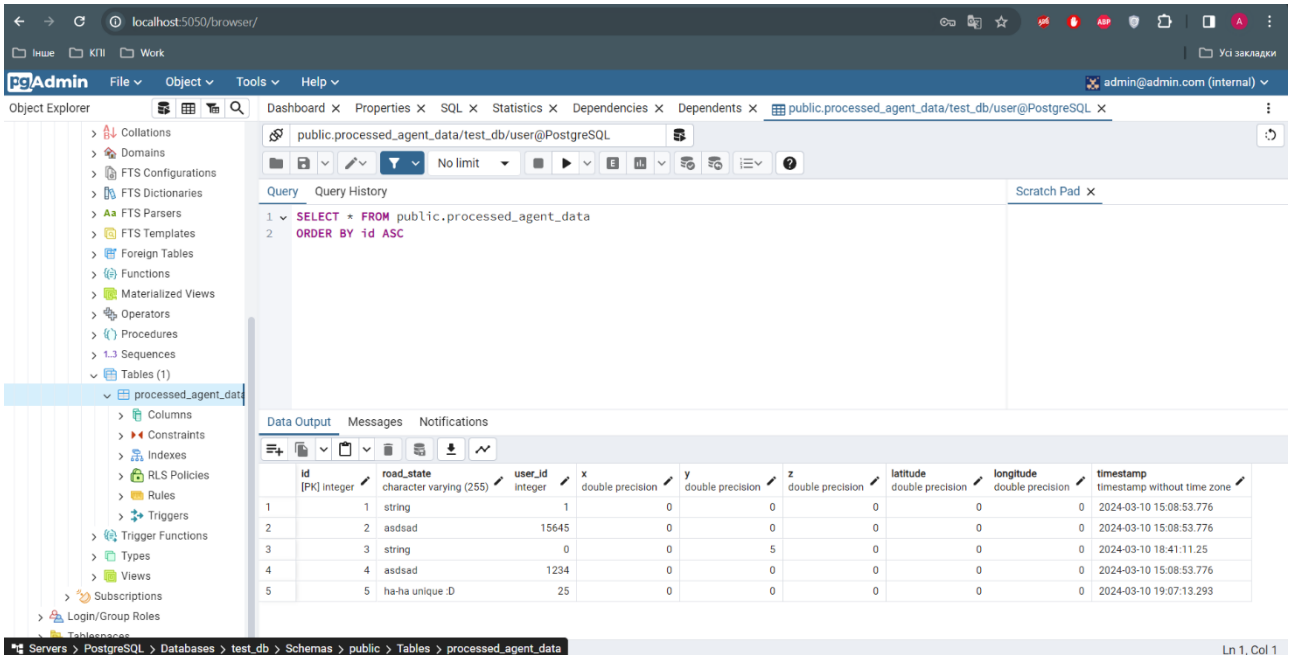
```
{
  "status": "ok"
}
```

Download

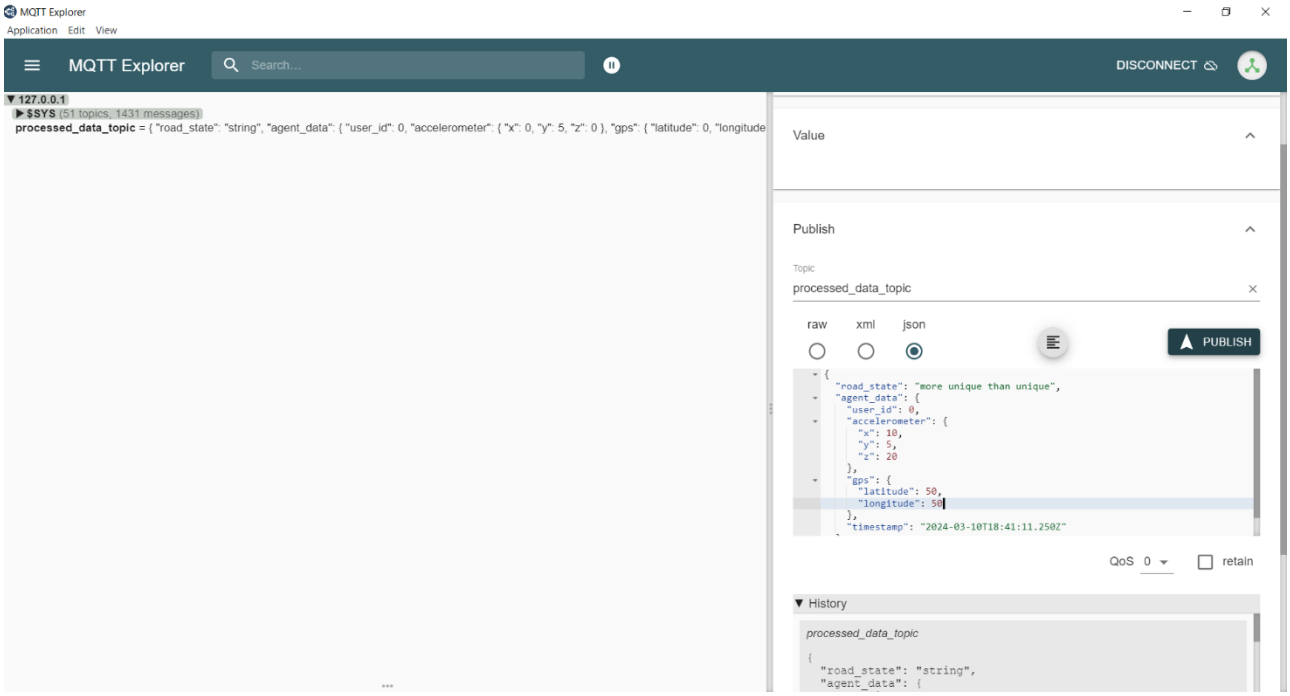
Response headers

```
content-length: 15
content-type: application/json
date: Sun,10 Mar 2024 19:15:19 GMT
server: uvicorn
```

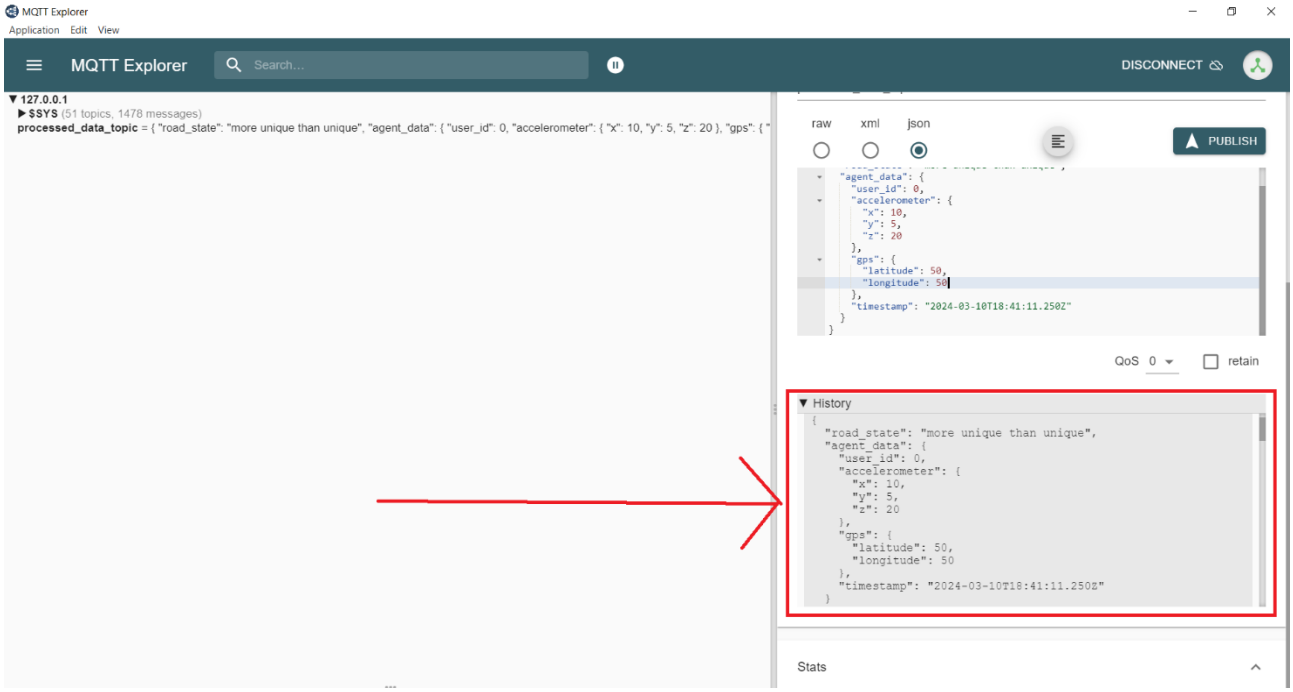
pgAdmin:



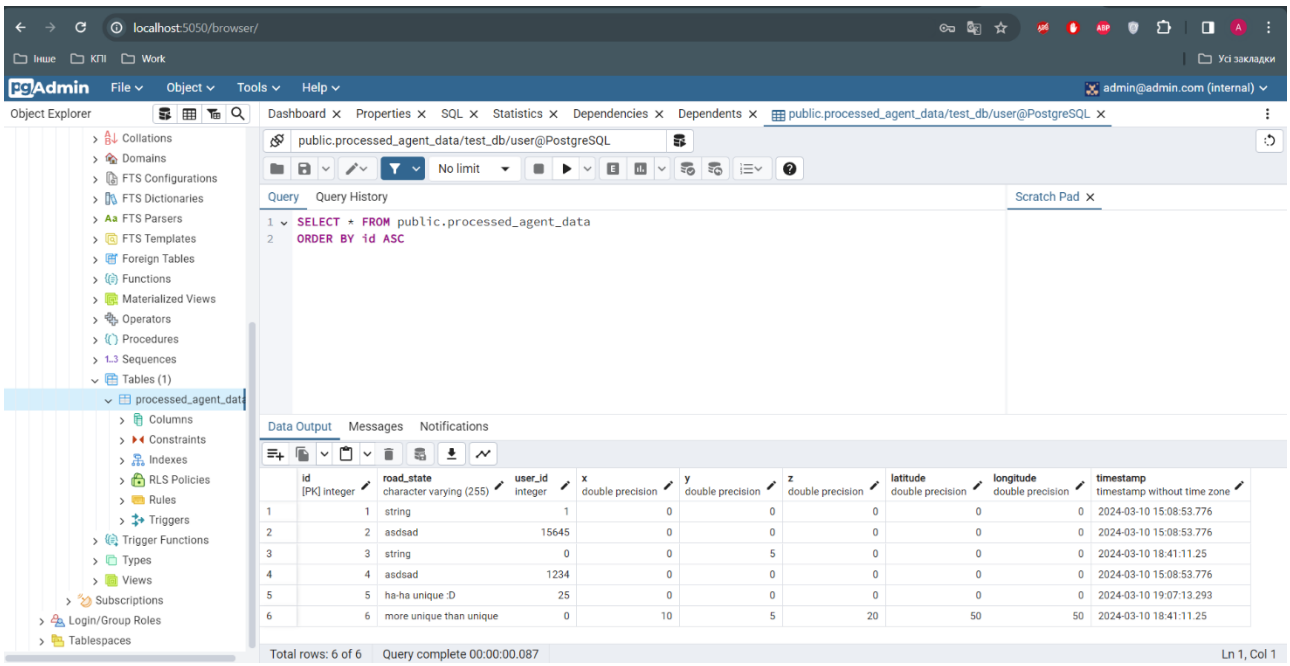
MQTT:



Бачимо оновлену історію



pgAdmin:



Висновок: під час виконання даної лабораторної роботи ми ознайомились з поняттям адаптерів у Python та їх застосуванням. З допомогою мови Python реалізували класс StoreApiAdapter, який наслідує StorageGateway(наданий в скелетоні проекту), та імплементує його метод `save_data` для збереження обробленої інформації.