

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

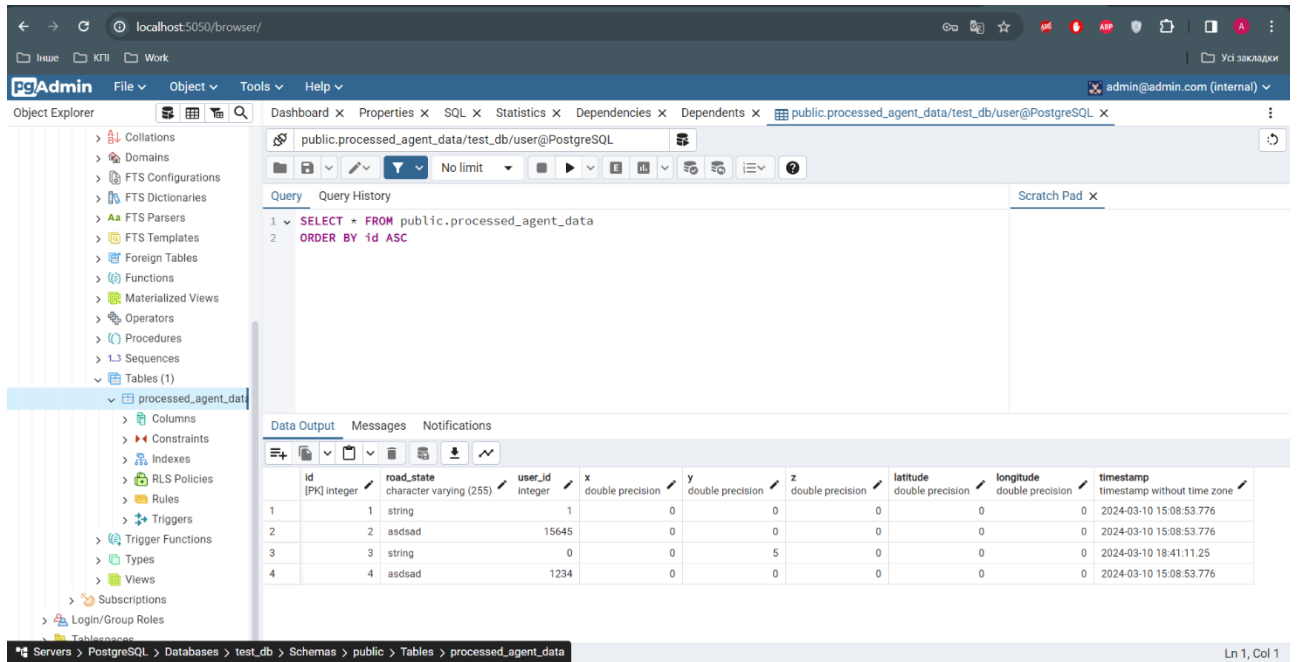
    def save_data(self, processed_agent_data_batch: List[ProcessedAgentData]):
        """
        Save the processed road data to the Store API.
        Parameters:
            processed_agent_data_batch (dict): Processed road data to be saved.
        Returns:
            bool: True if the data is successfully 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)

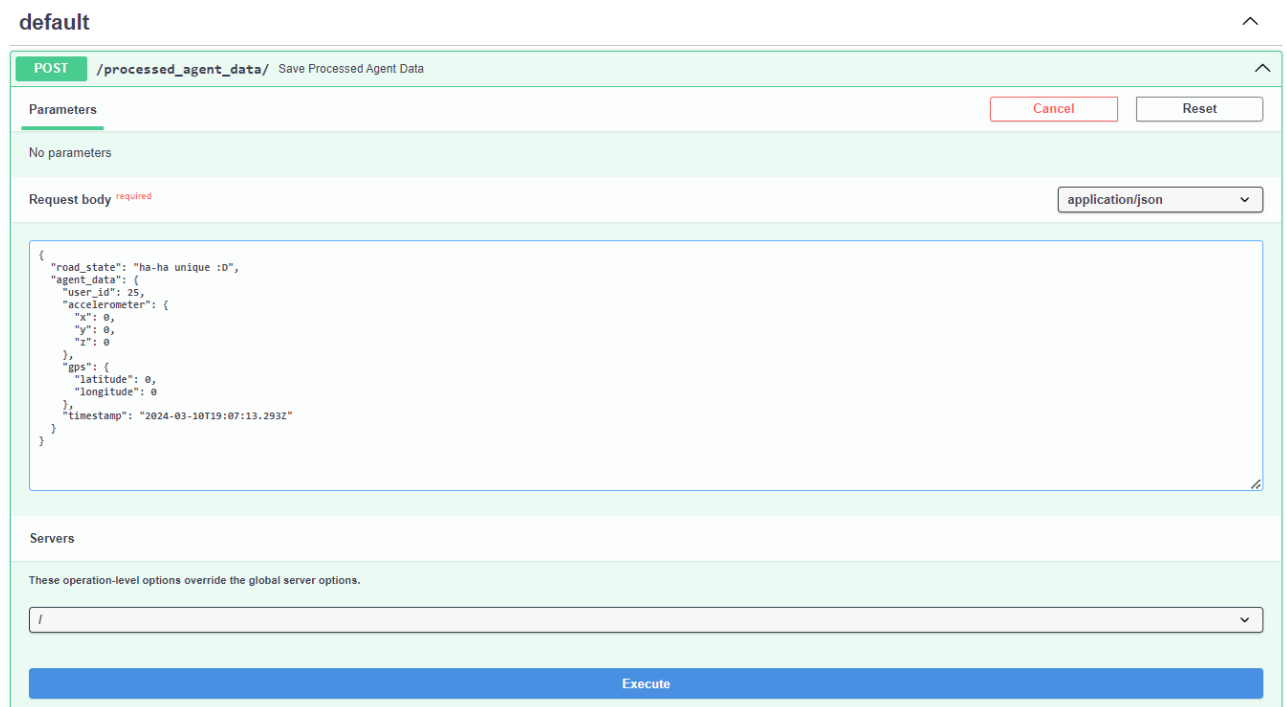
            # Check if the request was successful (status code 200)
            if response.status_code == 200 or response.ok:
                return True
            else:
                return False
        except Exception as e:
            logging.info(f"Error occurred {e}")
            return False
```

### 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"
}
```

Response headers

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

## pgAdmin:

localhost:5050/browser/

pgAdmin

Object Explorer

- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (1)
  - processed\_agent\_data
    - Columns
    - Constraints
    - Indexes
    - RLS Policies
    - Rules
    - Triggers
    - Trigger Functions
    - Types
    - Views
    - Subscriptions
- Login/Group Roles

Query

```
1 SELECT * FROM public.processed_agent_data
2 ORDER BY id ASC
```

Data Output

	id [PK] integer	road_state character varying (255)	user_id integer	x double precision	y double precision	z double precision	latitude double precision	longitude double precision	timestamp timestamp without time zone
1	1	string	1	0	0	0	0	0	2024-03-10 15:08:53.776
2	2	asdsad	15645	0	0	0	0	0	2024-03-10 15:08:53.776
3	3	string	0	0	5	0	0	0	2024-03-10 18:41:11.25
4	4	asdsad	1234	0	0	0	0	0	2024-03-10 15:08:53.776
5	5	ha-ha unique :D	25	0	0	0	0	0	2024-03-10 19:07:13.293

Servers > PostgreSQL > Databases > test\_db > Schemas > public > Tables > processed\_agent\_data

Ln 1, Col 1

## MQTT:

MQTT Explorer

Application Edit View

MQTT Explorer Search...

DISCONNECT

▼ 127.0.0.1

► SSYS (51 topics, 1431 messages)

processed\_data\_topic = { "road\_state": "string", "agent\_data": { "user\_id": 0, "accelerometer": { "x": 0, "y": 5, "z": 0 }, "gps": { "latitude": 0, "longitude": 0 } } }

Value

Publish

Topic

processed\_data\_topic

raw xml json

PUBLISH

```
{
  "road_state": "more unique than unique",
  "agent_data": {
    "user_id": 0,
    "accelerometer": {
      "x": 10,
      "y": 5,
      "z": 20
    },
    "gps": {
      "latitude": 50,
      "longitude": 50
    }
  },
  "timestamp": "2024-03-10T18:41:11.250Z"
}
```

QoS 0 retain

▼ History

processed\_data\_topic

```
{
  "road_state": "string",
  "agent_data": {
    "user_id": 0,
    "accelerometer": {
      "x": 0,
      "y": 5,
      "z": 0
    },
    "gps": {
      "latitude": 0,
      "longitude": 0
    }
  },
  "timestamp": "2024-03-10T18:41:11.250Z"
}
```

Бачимо оновлену історію

MQTT Explorer

Application Edit View

MQTT Explorer Search...

DISCONNECT

▼ 127.0.0.1

► SSYS (51 topics, 1478 messages)

processed\_data\_topic = { "road\_state": "more unique than unique", "agent\_data": { "user\_id": 0, "accelerometer": { "x": 10, "y": 5, "z": 20 }, "gps": { "latitude": 0, "longitude": 0 } } }

raw xml json

PUBLISH

```
{
  "agent_data": {
    "user_id": 0,
    "accelerometer": {
      "x": 10,
      "y": 5,
      "z": 20
    },
    "gps": {
      "latitude": 50,
      "longitude": 50
    }
  },
  "timestamp": "2024-03-10T18:41:11.250Z"
}
```

QoS 0 retain

▼ History

```
{
  "road_state": "more unique than unique",
  "agent_data": {
    "user_id": 0,
    "accelerometer": {
      "x": 10,
      "y": 5,
      "z": 20
    },
    "gps": {
      "latitude": 50,
      "longitude": 50
    }
  },
  "timestamp": "2024-03-10T18:41:11.250Z"
}
```

Stats

pgAdmin:

The screenshot shows the PgAdmin web interface. On the left is the Object Explorer with a tree view of database objects. The main pane displays a SQL query: `SELECT * FROM public.processed_agent_data ORDER BY id ASC`. Below the query editor is the 'Data Output' tab, which shows a table with 6 rows and 10 columns. The columns are: id (integer), road\_state (character varying), user\_id (integer), x (double precision), y (double precision), z (double precision), latitude (double precision), longitude (double precision), and timestamp (timestamp without time zone). The data rows contain various values, including strings like 'asdsad', integers like 15645 and 1234, and timestamps.

id	road_state	user_id	x	y	z	latitude	longitude	timestamp
1	string	1	0	0	0	0	0	2024-03-10 15:08:53.776
2	asdsad	15645	0	0	0	0	0	2024-03-10 15:08:53.776
3	string	0	0	5	0	0	0	2024-03-10 18:41:11.25
4	asdsad	1234	0	0	0	0	0	2024-03-10 15:08:53.776
5	ha-ha unique :D	25	0	0	0	0	0	2024-03-10 19:07:13.293
6	more unique than unique	0	10	5	20	50	50	2024-03-10 18:41:11.25

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