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 № 2 з дисципліни

«Технології розробки вбудованих IoT системи»

„CRUD ОПЕРАЦІЇ. РОЗРОБКА CRUD ОПЕРАЦІЙ.”

**Перевірив**

(прізвище, ім'я, по батькові)

Київ 2023

1. Посилання на git repo: <https://github.com/faaant/iot>
2. Реалізація CRUD операцій для processed\_agent\_data

# FastAPI CRUDL endpoints

@app.post("/processed\_agent\_data/")

async def create\_processed\_agent\_data(data: List[ProcessedAgentData]):

    query\_values = []

    for item in data:

        value = {

            "road\_state": item.road\_state,

            "x": item.agent\_data.accelerometer.x,

            "y": item.agent\_data.accelerometer.y,

            "z": item.agent\_data.accelerometer.z,

            "latitude": item.agent\_data.gps.latitude,

            "longitude": item.agent\_data.gps.longitude,

            "timestamp": item.agent\_data.timestamp,

            "user\_id": item.agent\_data.user\_id,

        }

        query\_values.append(value)

    query = processed\_agent\_data.insert().values(query\_values)

    # Execute the query within a transaction

    with engine.begin() as conn:

        conn.execute(query)

    # Send data to subscribers

    for item in data:

        await send\_data\_to\_subscribers(item.agent\_data.user\_id, item.model\_dump())

@app.get(

    "/processed\_agent\_data/{processed\_agent\_data\_id}",

    response\_model=ProcessedAgentDataInDB,

)

def read\_processed\_agent\_data(processed\_agent\_data\_id: int):

    # Construct a query to select data from the processed\_agent\_data table based on the provided ID

    query = select(processed\_agent\_data).where(

        processed\_agent\_data.c.id == processed\_agent\_data\_id

    )

    # Execute the query and retrieve the data

    with engine.connect() as conn:

        result = conn.execute(query)

        data = result.fetchone()

        if data is None:

            raise HTTPException(status\_code=404, detail="Data not found")

    return data

@app.get("/processed\_agent\_data/", response\_model=list[ProcessedAgentDataInDB])

def list\_processed\_agent\_data():

    # Construct a query to select data from the processed\_agent\_data table

    query = select(processed\_agent\_data)

    # Execute the query and retrieve the data

    with engine.connect() as conn:

        result = conn.execute(query)

        data\_list = result.fetchall()

    return data\_list

@app.put(

    "/processed\_agent\_data/{processed\_agent\_data\_id}",

    response\_model=ProcessedAgentDataInDB,

)

def update\_processed\_agent\_data(processed\_agent\_data\_id: int, data: ProcessedAgentData):

    # Construct the update query

    query = processed\_agent\_data.update().where(

        processed\_agent\_data.c.id == processed\_agent\_data\_id

    ).values(

        road\_state=data.road\_state,

        x=data.agent\_data.accelerometer.x,

        y=data.agent\_data.accelerometer.y,

        z=data.agent\_data.accelerometer.z,

        latitude=data.agent\_data.gps.latitude,

        longitude=data.agent\_data.gps.longitude,

        timestamp=data.agent\_data.timestamp,

        user\_id=data.agent\_data.user\_id,

    )

    # Execute the query within a transaction

    with engine.begin() as conn:

        result = conn.execute(query)

        # Check if any rows were affected (updated)

        if result.rowcount == 0:

            raise HTTPException(status\_code=404, detail="Data not found")

    return read\_processed\_agent\_data(processed\_agent\_data\_id)

@app.delete(

    "/processed\_agent\_data/{processed\_agent\_data\_id}",

    response\_model=ProcessedAgentDataInDB,

)

def delete\_processed\_agent\_data(processed\_agent\_data\_id: int):

    processed\_agent\_data\_to\_be\_removed = read\_processed\_agent\_data(processed\_agent\_data\_id);

    # Construct the delete query

    query = processed\_agent\_data.delete().where(

        processed\_agent\_data.c.id == processed\_agent\_data\_id

    )

    # Execute the query within a transaction

    with engine.begin() as conn:

        result = conn.execute(query)

        # Check if any rows were affected (deleted)

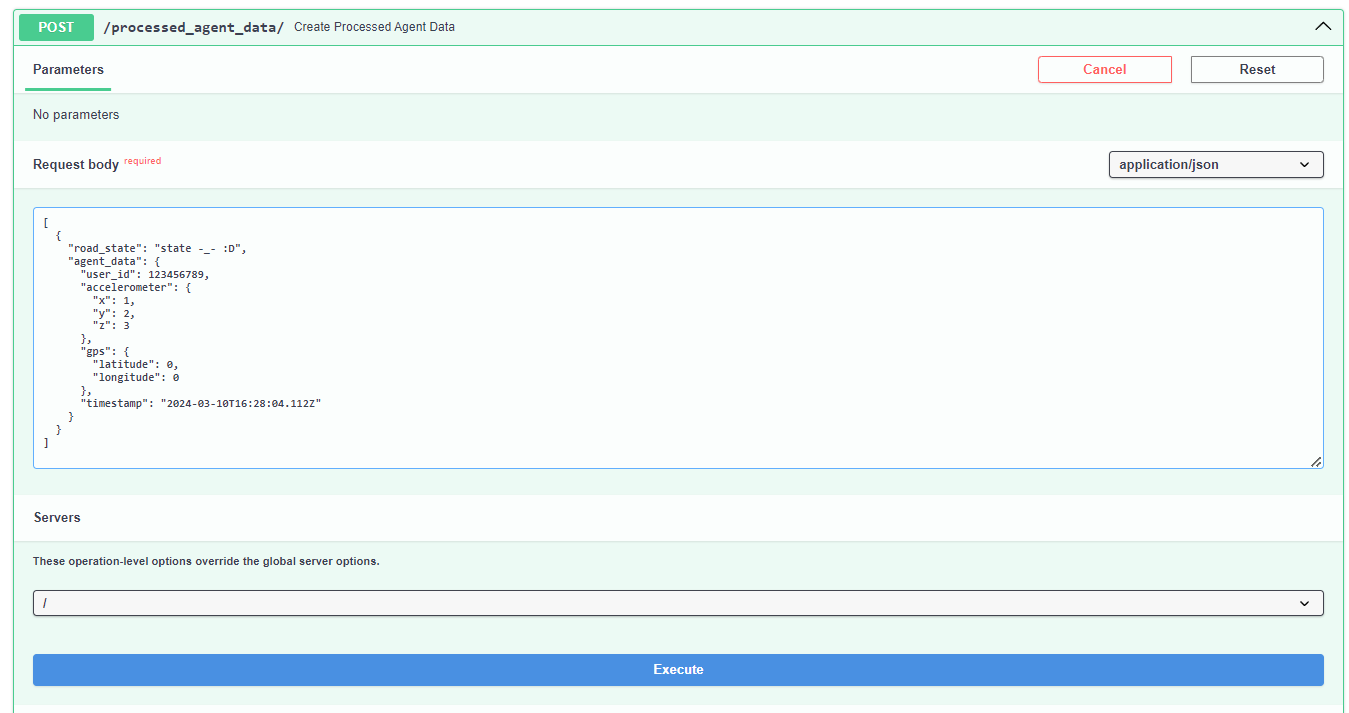
        if result.rowcount == 0:

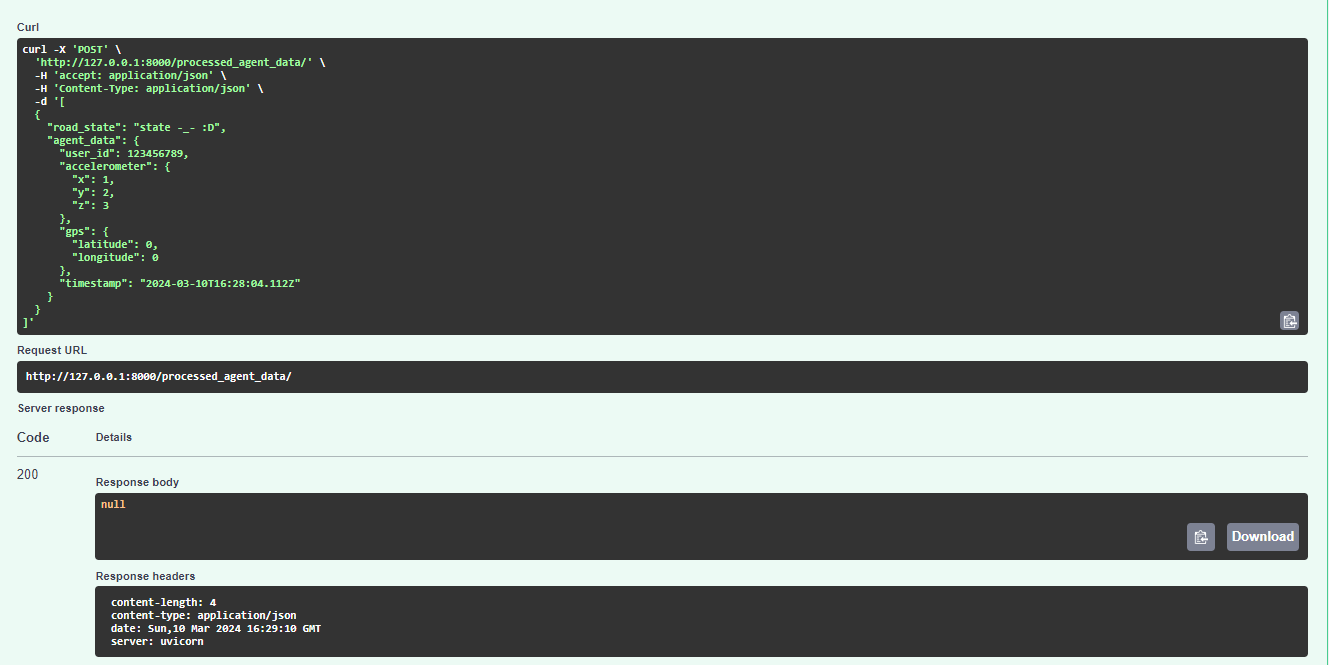
            raise HTTPException(status\_code=404, detail="Data not found")

    return processed\_agent\_data\_to\_be\_removed

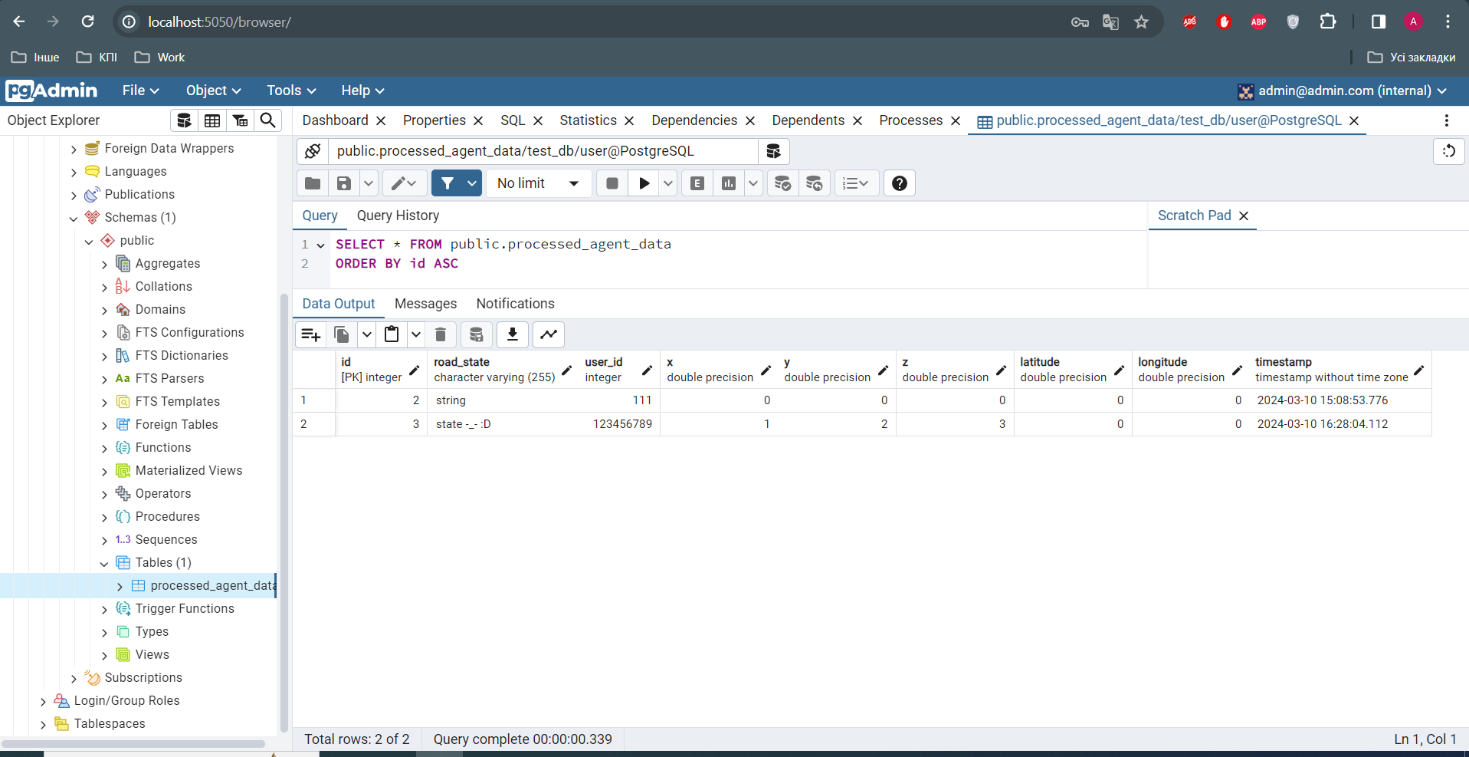
1. Показ результатів роботи в Swagger та PgAdmin

**POST:**

Swagger:  


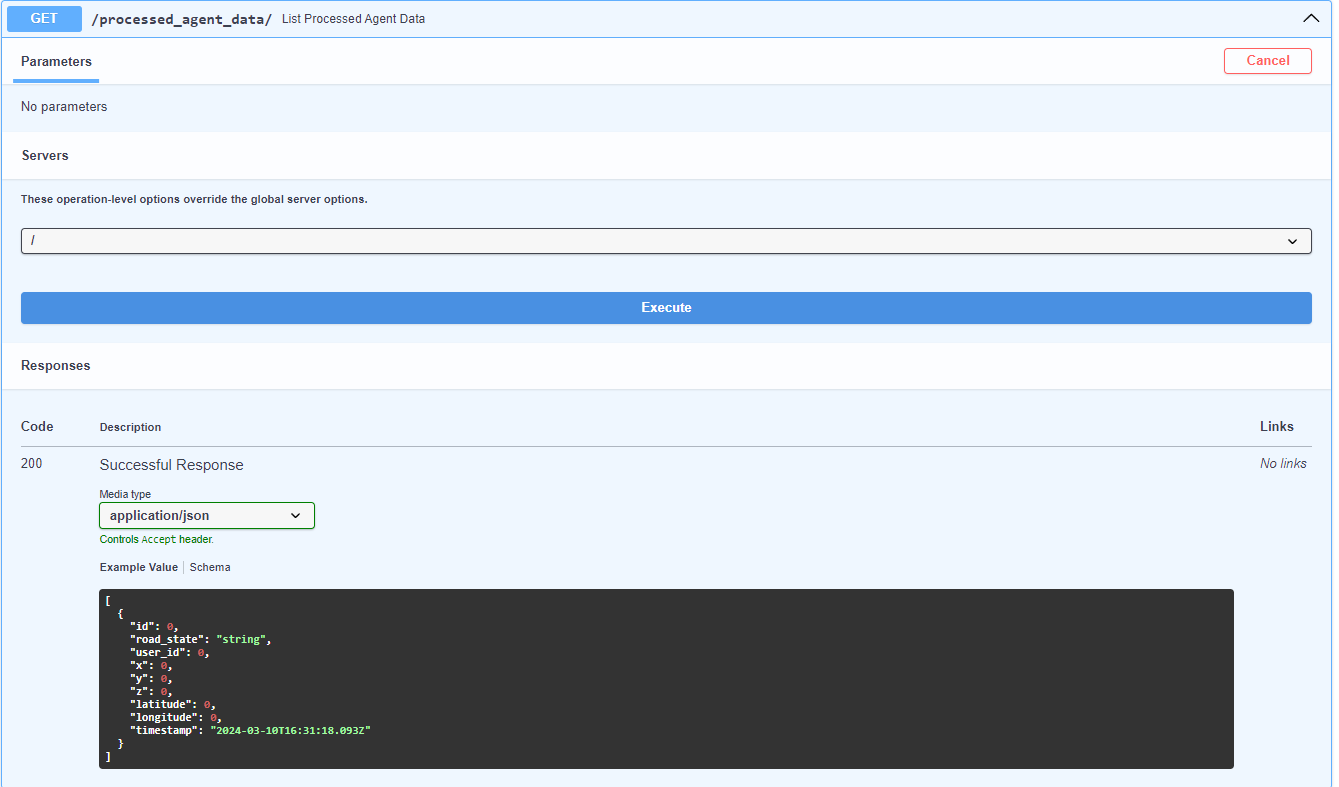


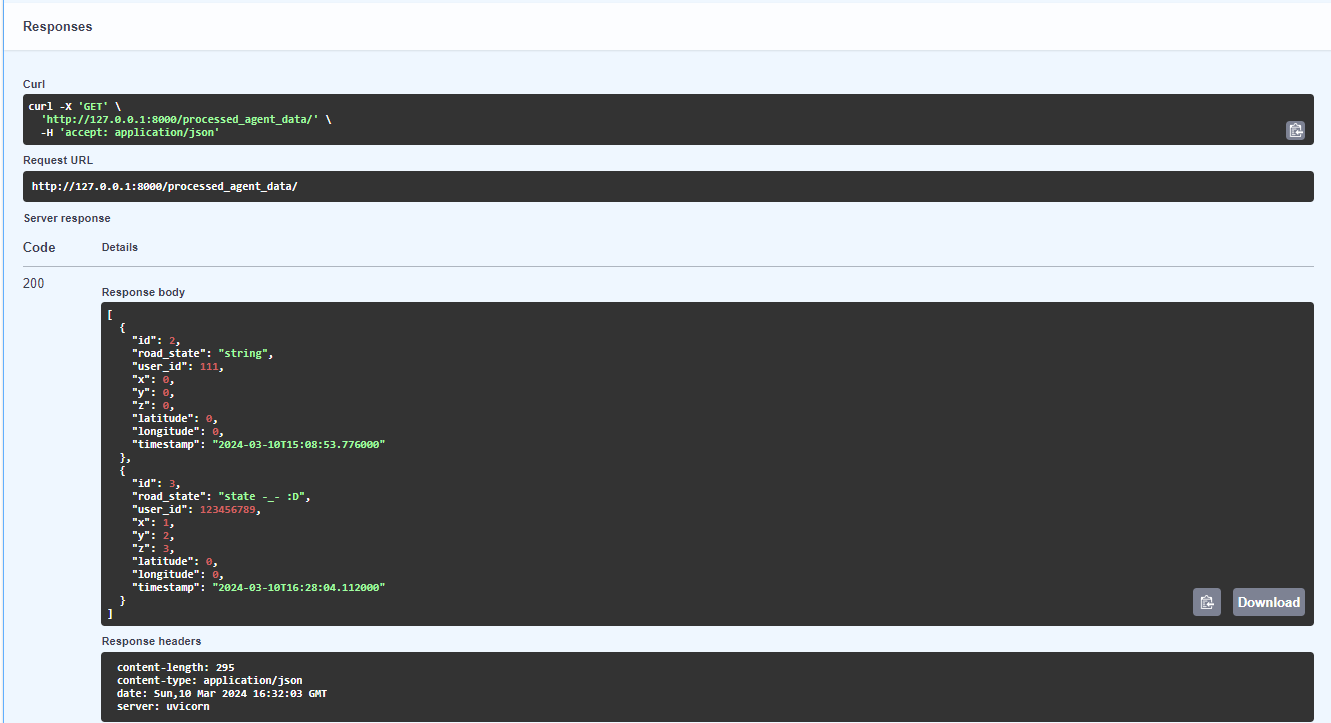
Результат у pgAdmin:



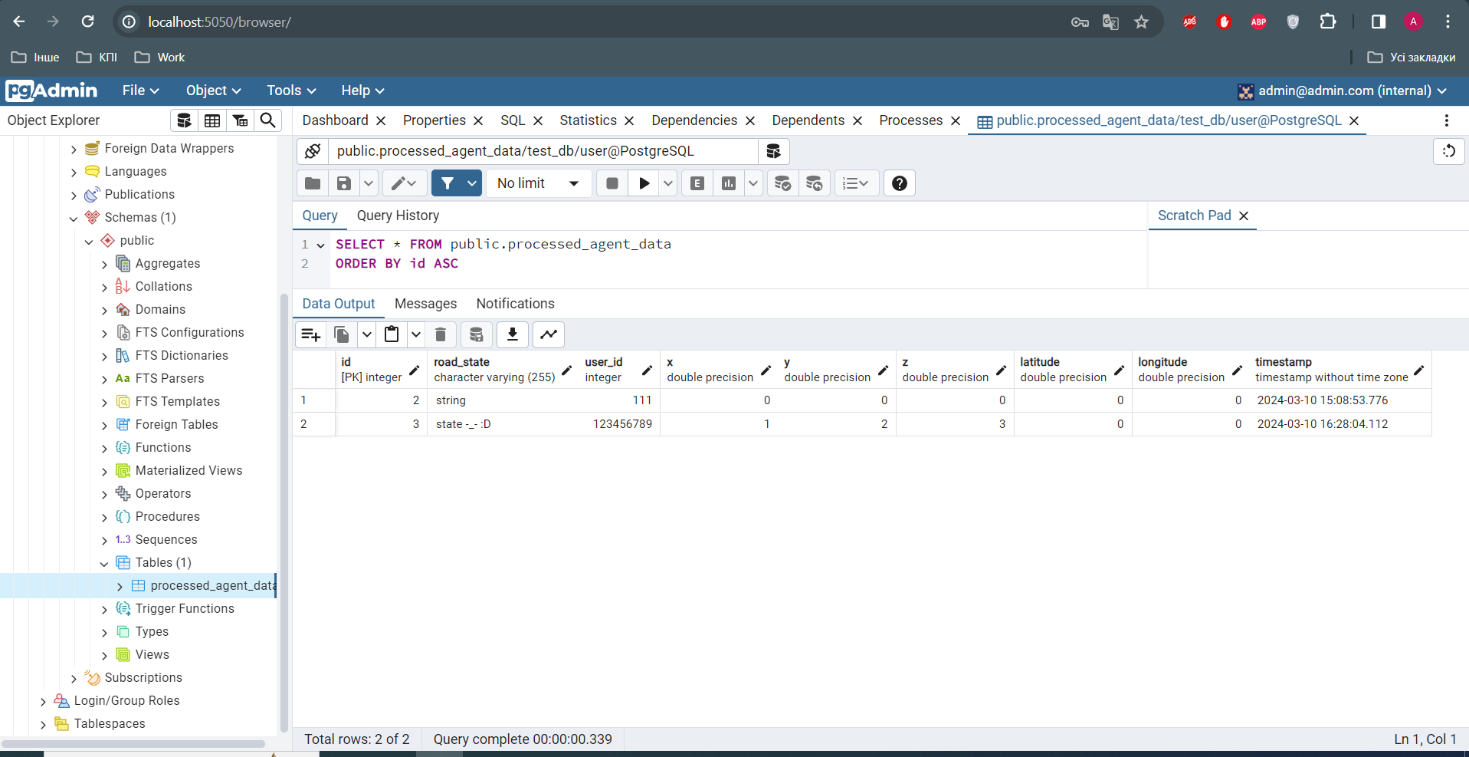
**GET:**

Swagger:



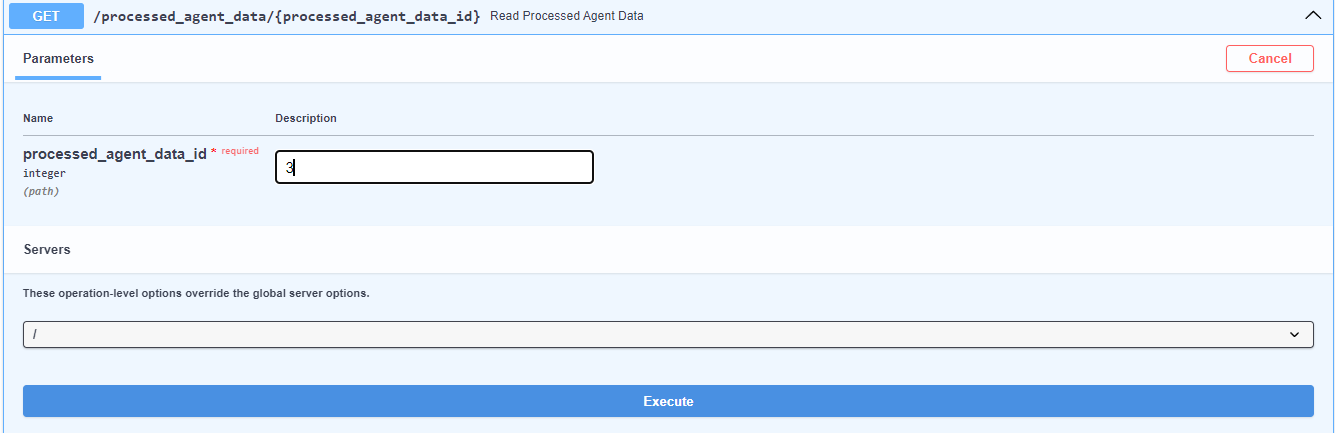


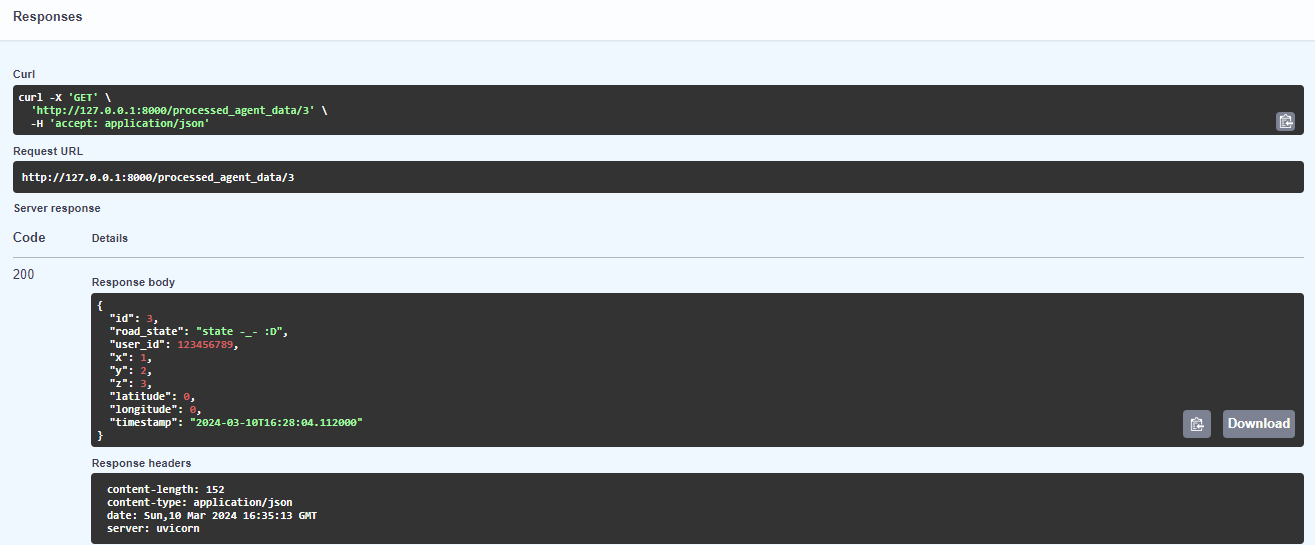
PgAdmin:



**GET:ID**

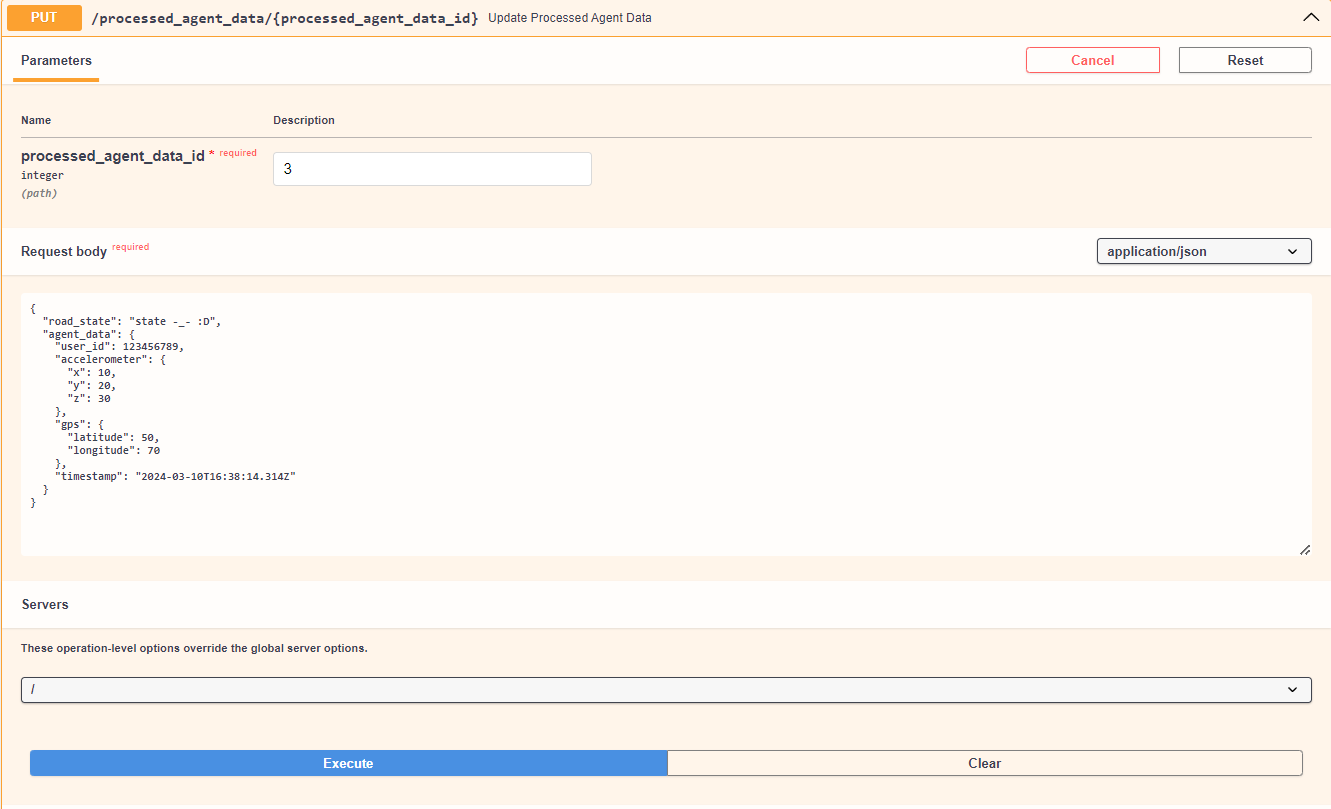
Swagger:

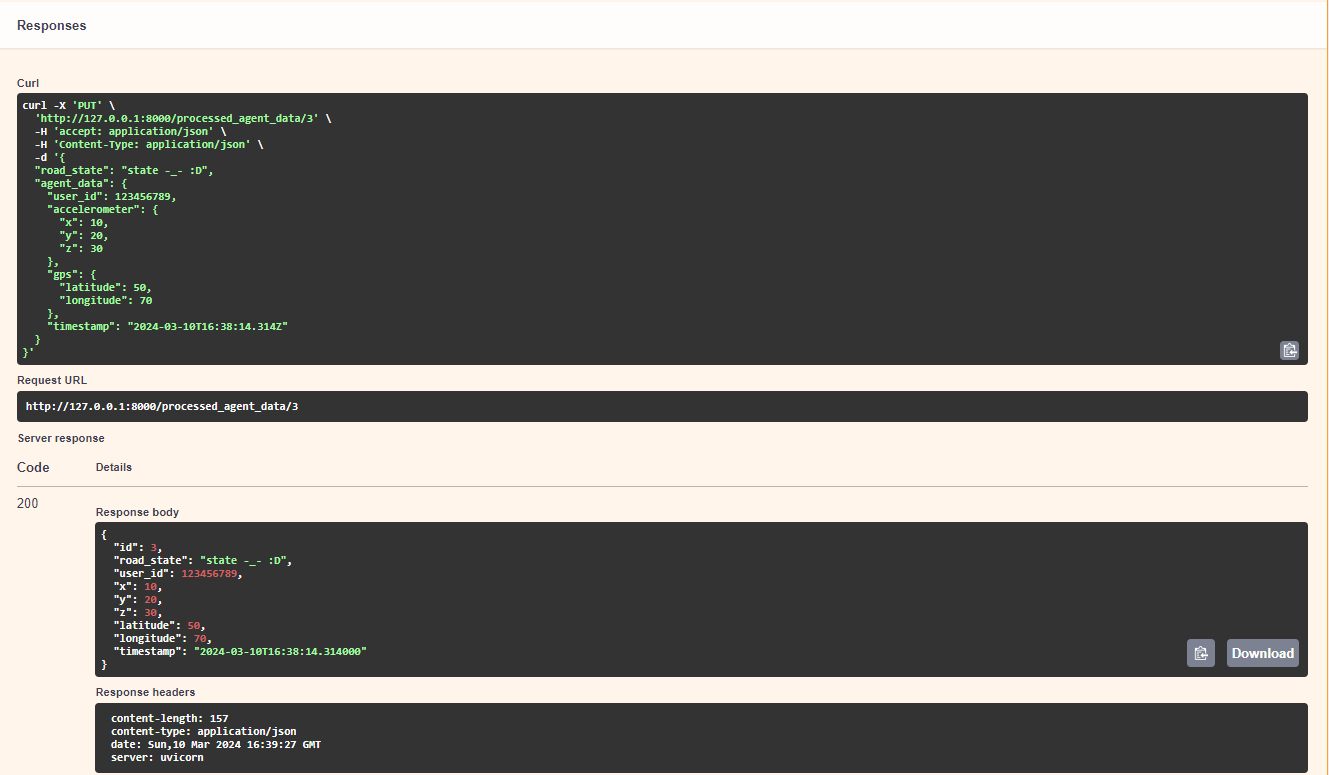




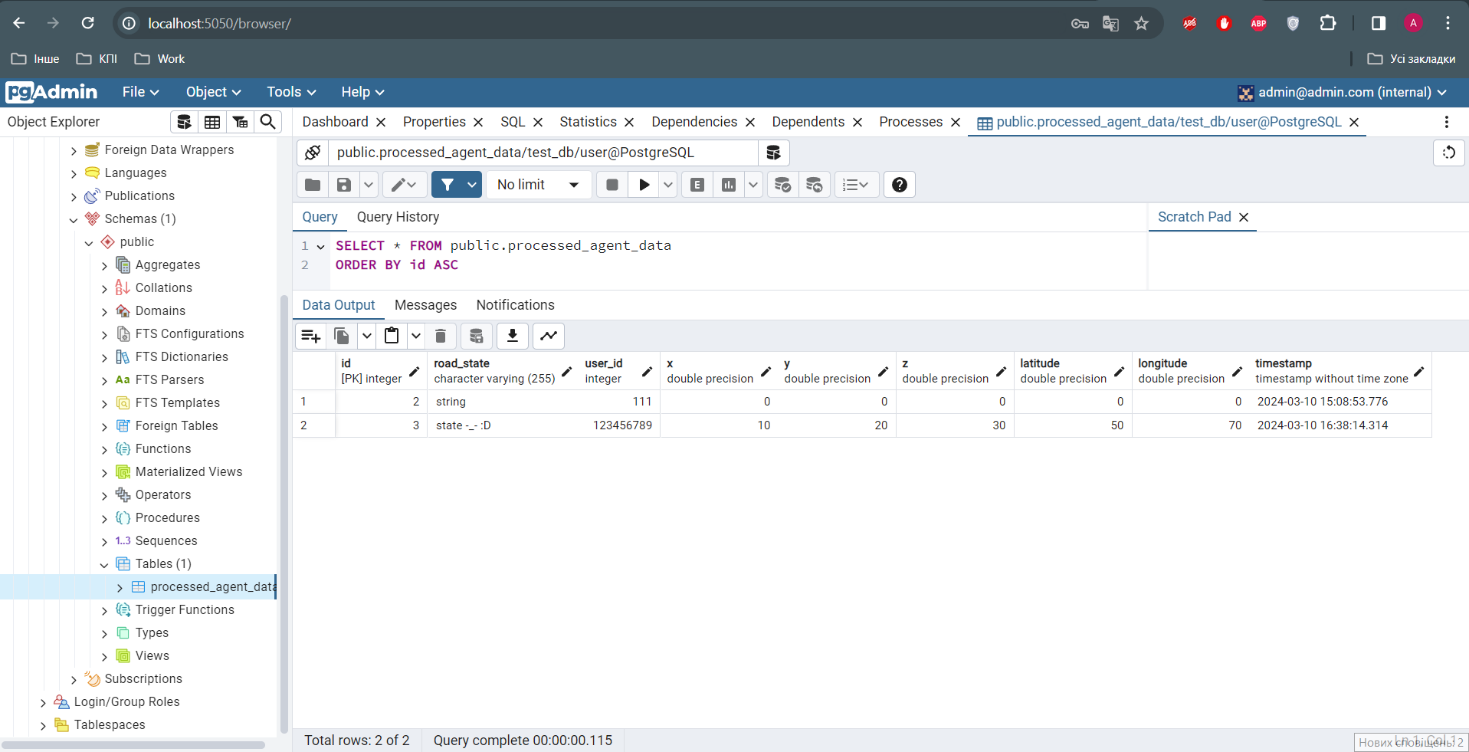
**UPDATE:**

Swagger:



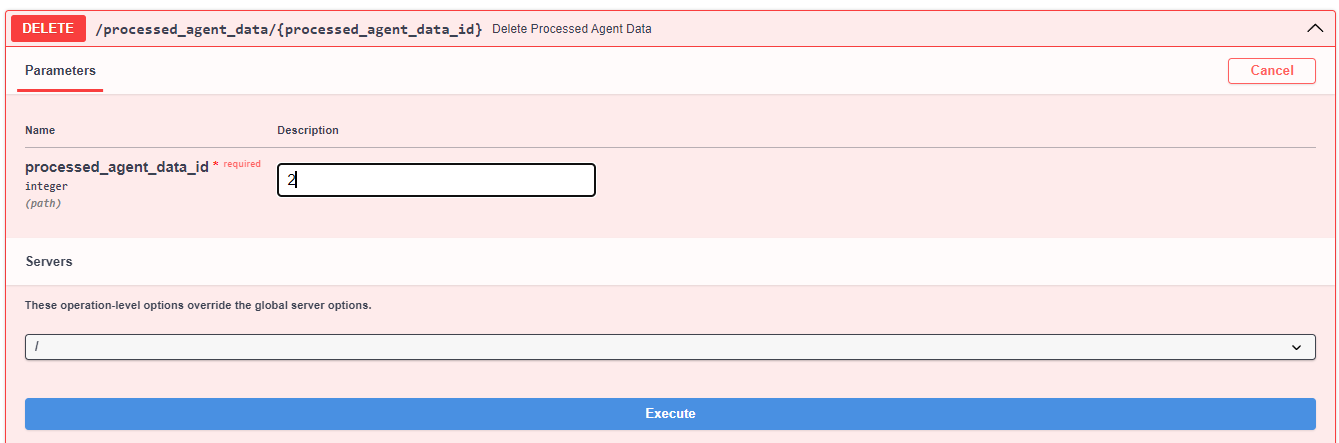


pgAdmin:



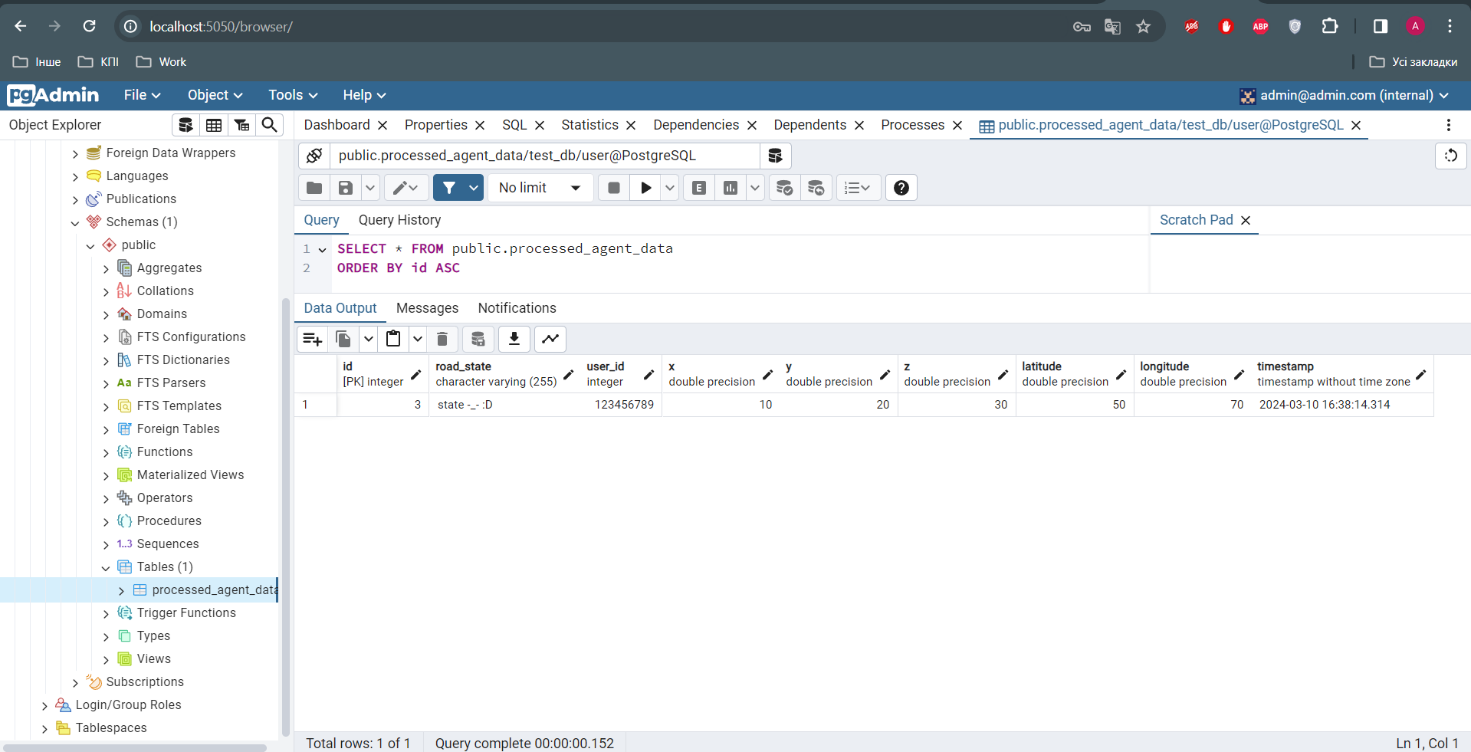
**DELETE:**

Swagger:





pgAdmin:



**Висновок:** під час виконання даної лабораторної роботи ми ознайомились з поняттям CRUD та їх застосуваннями. З допомогою мови Python реалізували CRUD операції для endpoint-a processed\_agent\_data.