

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Комплексна лабораторна робота**

з курсу: «Проєктування ПЗ технології ЦД»

Виконав  
аспірант групи КП-21ф  
Ільїн Максим Олександрович

Київ – 2024

**ЗМІСТ**

# 

[**ЗАВДАННЯ З КОМПЛЕКСНОЇ ЛАБОРАТОРНОЇ РОБОТИ 3**](#_yahqjdd6cbdi)

[**ТЕХНІЧНЕ ЗАВДАННЯ 4**](#_qi319xu07hp8)

[Тематика](#_5p64pz3lari) 4

[Мета 4](#_h7v0w8t0lj3n)

[Функціональні вимоги 4](#_vdfm96ln8b04)

[НЕфункціональні вимоги 5](#_dpwtvrc6irdq)

[Обмеження системи 5](#_t7ktanmcaco0)

[**ХІД РОБОТИ 6**](#_4fkoma81dtaj)

[Генератор даних 6](#_pacjowurgdvt)

[Azure Digital Twins 7](#_gy28f2aauwa6)

[Моделі двійників 9](#_pq2ad67fi7zz)

[Візуальна репрезентація двійників 13](#_oxbcy4trbanu)

[Моделювання взаємодії 14](#_d4qvo1pyy6b)

[Модуль аналітики 15](#_p9omh7d2jyjw)

[Результат роботи 16](#_a6v102ad823e)

[**ВИСНОВКИ 17**](#_8yihch0aryr)

# 

# 

# ЗАВДАННЯ З КОМПЛЕКСНОЇ ЛАБОРАТОРНОЇ РОБОТИ

1. Обрати об'єкт або процес – фізичний двійник (погодити з керівником роботи).
2. Розробити програмний генератор потокових даних або (за бажання) використати апаратне забезпечення.
3. Створити 3D модель фізичного двійника.
4. Обрати платформу для створення цифрового двійника.
5. Розробити цифровий двійник.
6. Протестувати цифровий двійник.
7. Підготувати демонстрацію та записати відео.
8. Підготувати звіт.
9. Завантажити на GitHub або аналогічний ресурс розроблений проєкт.
10. Захистити комплексну лабораторну роботу.

# ТЕХНІЧНЕ ЗАВДАННЯ

## Тематика

Система моніторингу якості повітря в містах.

## Мета

Створення комплексної системи для моніторингу та аналізу якості повітря в містах, що дозволить місцевим жителям та владі оцінювати стан навколишнього середовища та вживати відповідних заходів для його покращення.

## Функціональні вимоги

1. Цифровий двійник має інтегрувати дані з відкритих API для моніторингу різних показників якості повітря в місті, включно з концентрацією озону (O3), вуглекислого газу (CO2) та інших забруднюючих речовин.
2. Система повинна надавати актуалізовані дані про якість повітря в реальному часі для зазначених локацій у місті. Вона має мати здатність агрегувати дані з декількох джерел за допомогою API.
3. Має бути реалізована функція історичного перегляду показників якості повітря для оцінки динаміки змін у часі.
4. Система має включати простий та інтуїтивно зрозумілий інтерфейс для візуалізації даних.
5. Мають бути реалізовані системи сповіщення, які інформують користувачів про перевищення нормативних значень забруднювачів повітря.
6. Вхідні дані з API мають бути представлені у форматах JSON для забезпечення універсальності інтеграції з різними джерелами.
7. Вихідні дані для користувача мають візуалізуватися через веб-інтерфейс з використанням графіків (SVG або Canvas).
8. Backend: має бути реалізований на мові програмування Python (з використанням фреймворків Flask або Django для спрощення розробки веб-додатків).
9. Frontend: має бути реалізований із застосуванням HTML, CSS і JavaScript з бібліотеками для створення інтерактивного користувацького інтерфейсу.
10. Для зберігання даних: використати базу даних MongoDB.

## *НЕфункціональні вимоги*

1. Надійність: забезпечення стабільного збору та обробки даних з різних джерел в режимі 24/7.
2. Продуктивність: здатність обробляти великі об'єми інформації з мінімальним запізненням.
3. Масштабованість: можливість додавання нових джерел даних та підтримка зростаючої кількості користувачів без втрати продуктивності.
4. Користувацький інтерфейс: забезпечення легкої навігації та доступності інформації для всіх категорій користувачів.
5. Безпека: захист даних користувачів та забезпечення конфіденційності інформації, яка обробляється.

## *Обмеження системи*

1. Персоналізація: Користувачі мають обмежену можливість персоналізації інтерфейсу або налаштувань повідомлень, що включає обмеження у виборі конкретних показників забруднення, які відображаються, та в налаштуванні порогових значень для сповіщень.
2. Створення власних звітів та аналітики: Звіти користувачів обмежені графіками забруднення за конкретний період часу у вибраному регіоні.
3. Експорт даних: Присутня можливість експорту даних користувачами для подальшого аналізу чи інтеграції з іншими системами.

# ХІД РОБОТИ

## *Генератор даних*

*Дані:*

В якості джерела даних було використано [Air Quality API](https://rapidapi.com/weatherbit/api/air-quality):

* Даний API надає дані стосовно якості повітря, поточну якість повітря для будь-якої точки світу за її координатами або назвою міста та країни за період 72 годин.
* Дані оновлюються щогодинно.
* Безкоштовна версія API обмежена 25 викликами на день.

| Таблиця 1. Приклад результату виклику Air Quality API для міста Raleigh |
| --- |
| {  "city\_name": "Raleigh",  "country\_code": "US",  "data": [  {  "aqi": 52,  "co": 150.3,  "datetime": "2024-05-28:11",  "no2": 9,  "o3": 42.7,  "pm10": 19.2,  "pm25": 12.83,  "so2": 5.3,  "timestamp\_local": "2024-05-28T07:00:00",  "timestamp\_utc": "2024-05-28T11:00:00",  "ts": 1716894000  }  ]  } |

*Генератор:*

Генератор даних було реалізовано за допомогою Azure Functions:

* Тригером виклику функції є таймер, що спрацьовує раз на годину. Це пов’язано із частотою оновлення даних у API.
* Функція робить звернення до API, виконує збереження даних для історичного аналізу, перевіряє, чи не є перевищеними норми забруднення та оновлює дані у цифрових двійниках.
* Звернення виконується для 10 сенсорів – по одному на кожен район міста Київ.

*Тестування:*

В зв’язку із обмеженням API для отримання інформації про забруднення з датчиків у кількості 25 викликів на день, для тестування був розроблений програмний генератор даних, що виконує симуляцію реальних даних та оновлює показники забруднення кожні 10 секунд.

**

Рис. 1. Функція генерації симульованих даних

*Сховище даних:*

Для збереження історичних даних було використано MongoDB. Також було реалізовано CRUD модуль для роботи з зазначеною БД.

## Azure Digital Twins

Для реалізації ЦД було використано платформу Azure Digital Twins.

Реалізацію візуальної репрезентації було розділено на дві частини в зв’язку із обмеженнями платформи Azure Digital Twins:

* Azure Digital Twins 3D Scenes використовувалась для візуального моделювання взаємодії датчиків забруднення повітря із сервером міського додатку, який виконує сповіщення кінцевих користувачів про забруднення повітря.
* Flask server + HTML/CSS/JS використовувались для реалізації аналітичного модулю системи, що надає доступ до історичних даних у вигляді JSON-файлу, або графіків зміни відповідних забруднюючих факторів у часі.

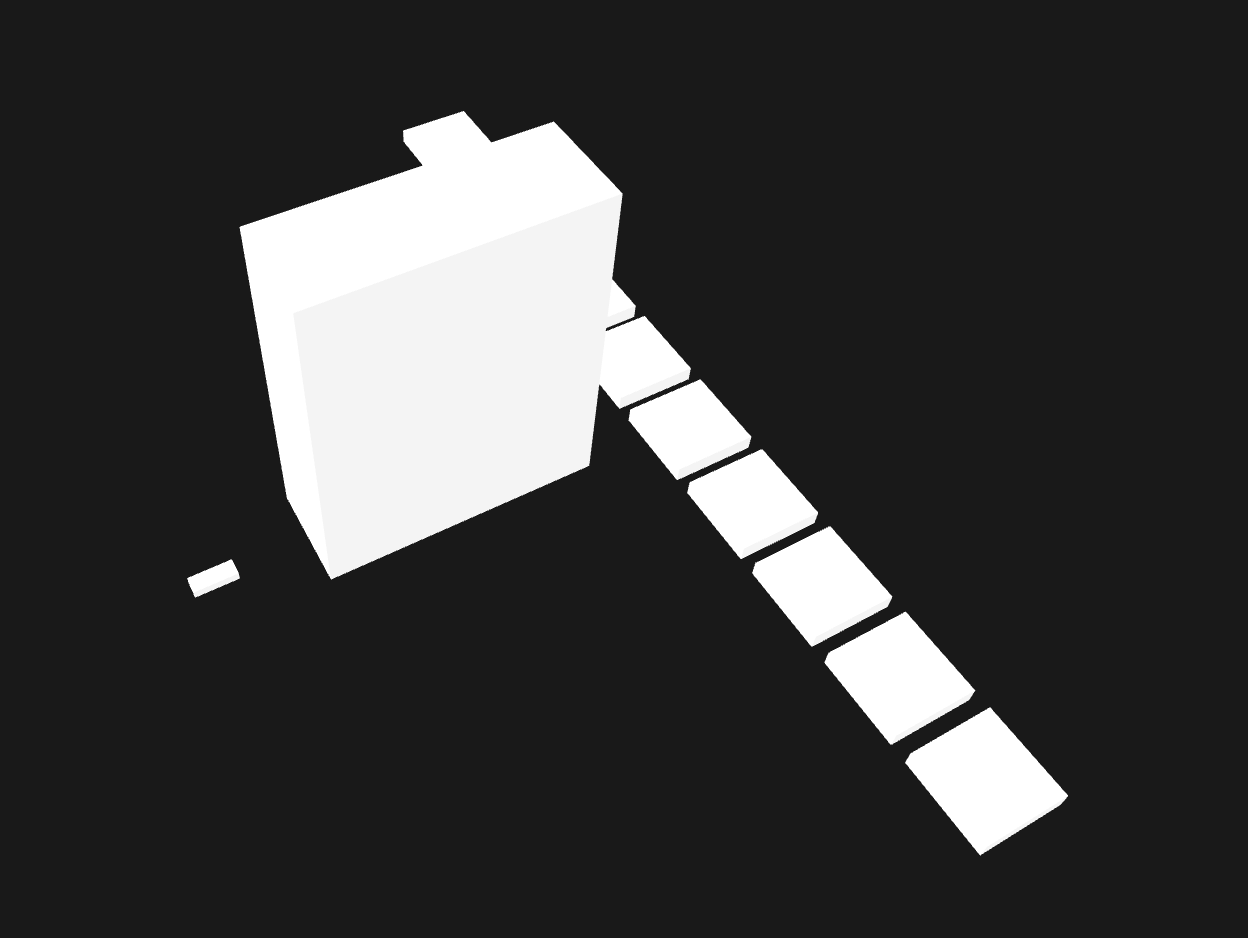


Рис. 2. 3D модель, що репрезентує 10 сенсорів забруднення, сервер додатку і телефон кінцевого користувача, створена у форматі .glb для відтворення у Azure Digital Twins 3D Scenes

## Моделі двійників

Було створено наступні моделі двійників, визначені відповідними JSON-файлами схем:

* DistrictSensor – цифровий двійник сенсору забруднення повітря, містить в схемі координати знаходження, назву району міста та показники забруднення.

| Таблиця 2. DistrictSensor.json |
| --- |
| {  "@id": "dtmi:environment:DistrictSensor;1",  "@type": "Interface",  "displayName": "District's Sensor",  "extends": "dtmi:environment:Coordinates;1",  "contents": [  {  "@type": "Property",  "name": "name",  "schema": "string",  "displayName": "District Name",  "description": "The name of the district."  },  {  "@type": "Property",  "name": "datetime",  "schema": "string",  "displayName": "Date Time",  "description": "The date and time of the measurement."  },  {  "@type": "Property",  "name": "aqi",  "schema": "integer",  "displayName": "Air Quality Index",  "description": "Air Quality Index value."  },  {  "@type": "Property",  "name": "co",  "schema": "double",  "displayName": "Carbon Monoxide",  "description": "Concentration of carbon monoxide (CO) in µg/m3."  },  {  "@type": "Property",  "name": "no2",  "schema": "double",  "displayName": "Nitrogen Dioxide",  "description": "Concentration of nitrogen dioxide (NO2) in µg/m3."  },  {  "@type": "Property",  "name": "o3",  "schema": "double",  "displayName": "Ozone",  "description": "Concentration of ozone (O3) in µg/m3."  },  {  "@type": "Property",  "name": "pm10",  "schema": "double",  "displayName": "PM10",  "description": "Concentration of particulate matter < 10 µm in µg/m3."  },  {  "@type": "Property",  "name": "pm25",  "schema": "double",  "displayName": "PM2.5",  "description": "Concentration of particulate matter < 2.5 µm in µg/m3."  },  {  "@type": "Property",  "name": "so2",  "schema": "double",  "displayName": "Sulfur Dioxide",  "description": "Concentration of sulfur dioxide (SO2) in µg/m3."  },  {  "@type": "Relationship",  "name": "alertsApplication",  "target": "dtmi:kyiv:CityApp;1",  "displayName": "Alerts City App Server",  "description": "The city application that receives alerts from this sensor."  }  ],  "@context": "dtmi:dtdl:context;2"  } |

* CoordinatesModel – модель координат географічної точки.

| Таблиця 3. CoordinatesModel.json |
| --- |
| {  "@id": "dtmi:environment:Coordinates;1",  "@type": "Interface",  "displayName": "Geographic Coordinates",  "contents": [  {  "@type": "Property",  "name": "latitude",  "schema": "double",  "displayName": "Latitude",  "description": "The latitude of the location."  },  {  "@type": "Property",  "name": "longitude",  "schema": "double",  "displayName": "Longitude",  "description": "The longitude of the location."  }  ],  "@context": "dtmi:dtdl:context;2"  } |

* CityApp – цифровий двійник застосунку Київ Цифровий.

| Таблиця 4. CityApp.json |
| --- |
| {  "@id": "dtmi:kyiv:CityApp;1",  "@type": "Interface",  "displayName": "City Application Server",  "contents": [  {  "@type": "Property",  "name": "lastNotification",  "schema": "string",  "displayName": "Last Notification",  "description": "Details of the last notification sent."  },  {  "@type": "Property",  "name": "alertLevel",  "schema": "integer",  "displayName": "Current Alert Level",  "description": "Current level of alert as determined by the number of critical notifications."  },  {  "@type": "Command",  "name": "sendNotification",  "displayName": "Send Notification to Users",  "commandType": "synchronous",  "request": {  "@type": "CommandPayload",  "name": "message",  "schema": "string",  "displayName": "Notification Message",  "description": "Notification message to be sent to city residents."  },  "response": {  "@type": "CommandPayload",  "name": "result",  "schema": "string",  "displayName": "Result",  "description": "Result of the notification command."  }  },  {  "@type": "Relationship",  "name": "alertsUser",  "target": "dtmi:app:UserPhone;1",  "displayName": "Alerts user",  "description": "Alert sent by server."  }  ],  "@context": "dtmi:dtdl:context;2"  } |

* UserPhone – цифровий двійник телефону користувача із встановленим застосунком Київ Цифровий.

| Таблиця 5. Phone.json |
| --- |
| {  "@id": "dtmi:app:UserPhone;1",  "@type": "Interface",  "displayName": "User Phone",  "contents": [  {  "@type": "Command",  "name": "showNotification",  "displayName": "Show Notification to Users",  "commandType": "synchronous",  "request": {  "@type": "CommandPayload",  "name": "message",  "schema": "string",  "displayName": "Notification Message",  "description": "Notification message sent to city residents."  },  "response": {  "@type": "CommandPayload",  "name": "result",  "schema": "string",  "displayName": "Result",  "description": "Result of the notification command."  }  },  {  "@type": "Relationship",  "name": "isNotifiedBy",  "target": "dtmi:kyiv:CityApp;1",  "displayName": "Is notified by",  "description": "The city application server that sends notifications."  }  ],  "@context": "dtmi:dtdl:context;2"  } |

## Візуальна репрезентація двійників

Цифрових двійників було створено за допомогою платформи Azure Digital Twins.

Моделі пов’язані наступним зв’язком:

* Дані на датчиках оновлюються із інтервалом раз на 10 секунд;
* При перевищенні допустимих значень забруднення, до серверу застосунку відправляється alert із вказаним рівнем небезпеки.
* Якщо рівень небезпеки становить 2 і більше, то на мобільному телефоні кінцевого користувача активується нотифікація із повідомленням про існуюче забруднення повітря.

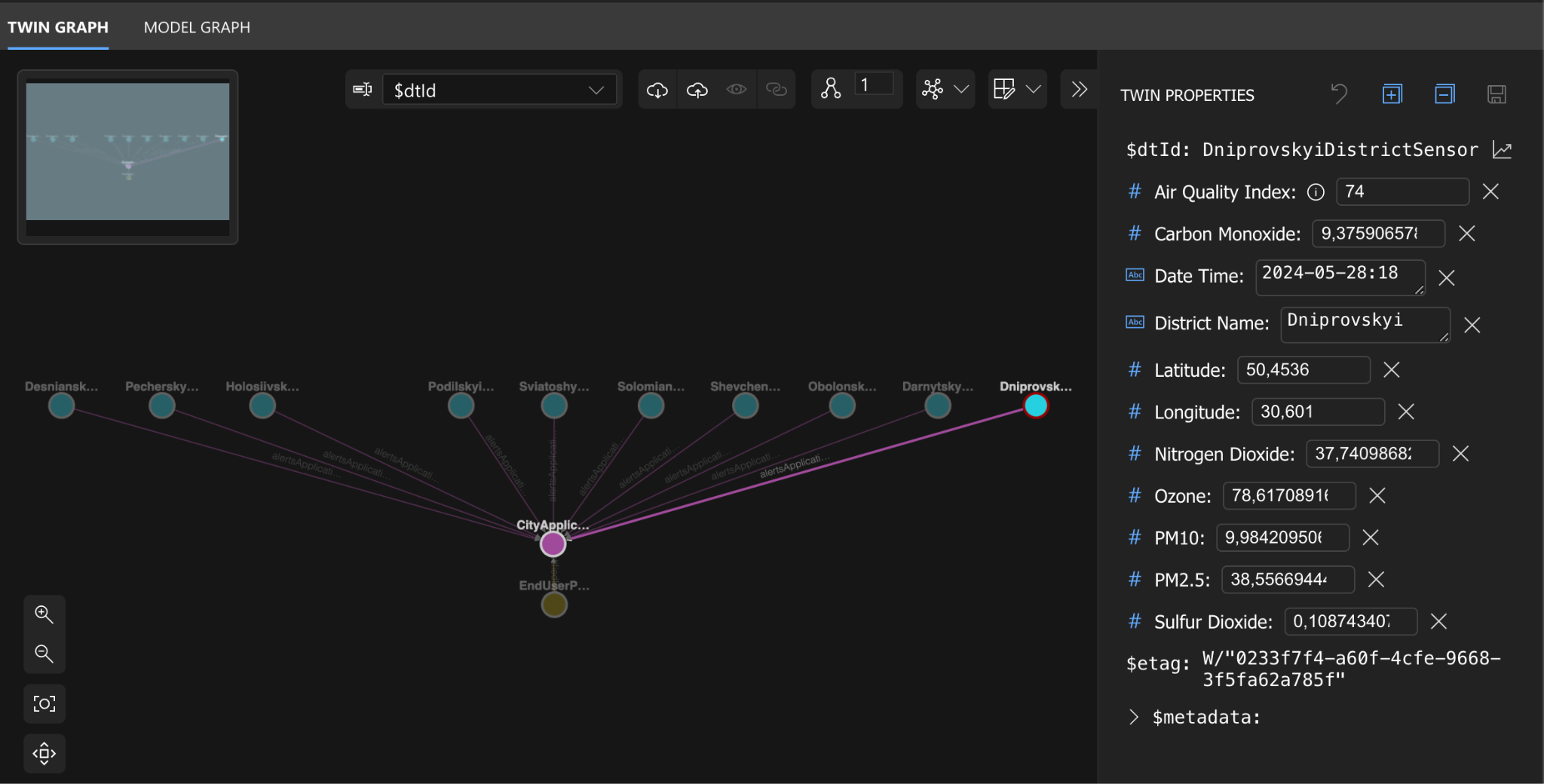


Рис. 3. Візуальна репрезентація двійників у Azure Digital Twins.

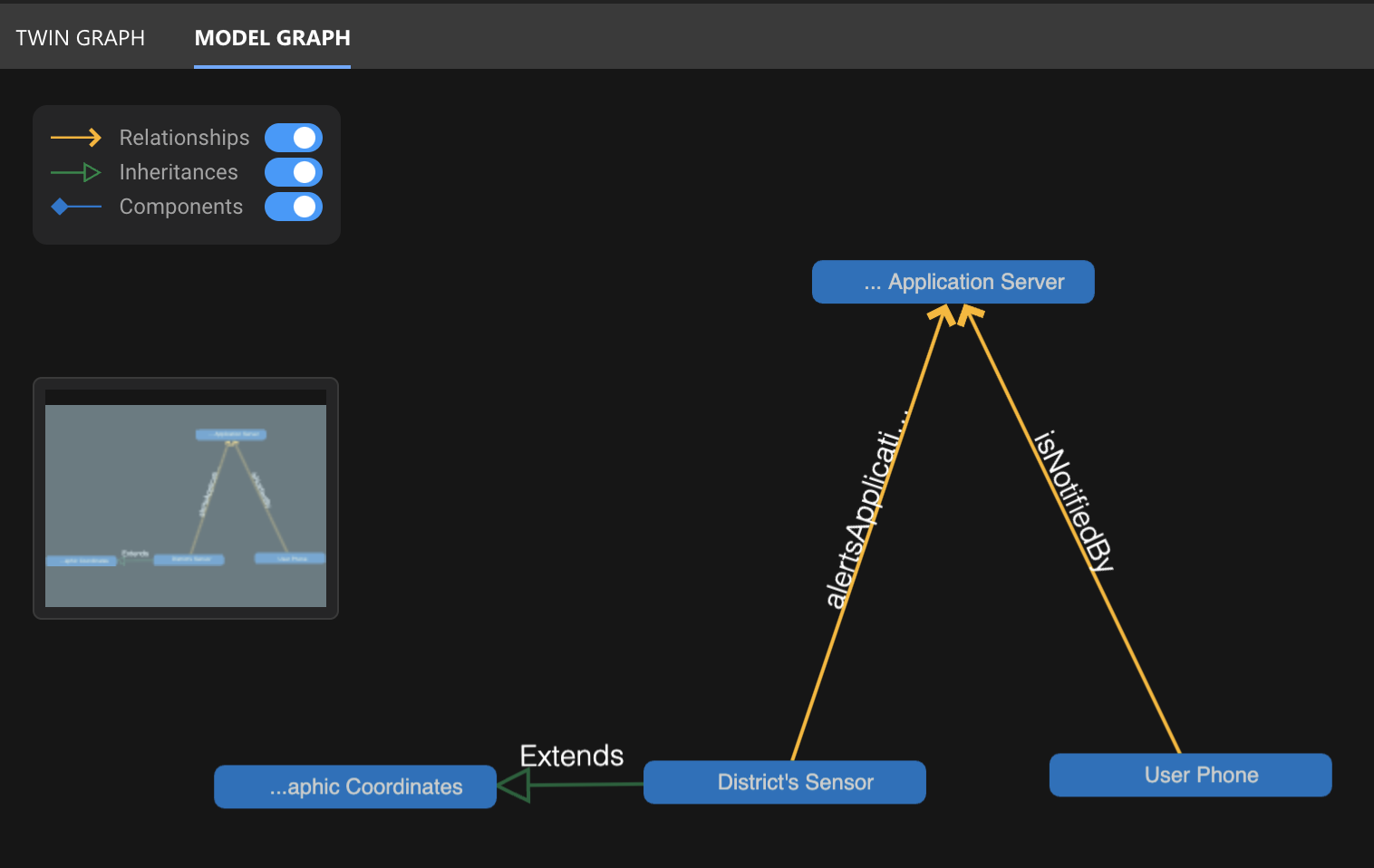


Рис. 4. Граф зв’язків моделей у Azure Digital Twins.

## Моделювання взаємодії

Візуальне моделювання взаємодії було реалізовано у Azure Digital Twins 3D Scenes.

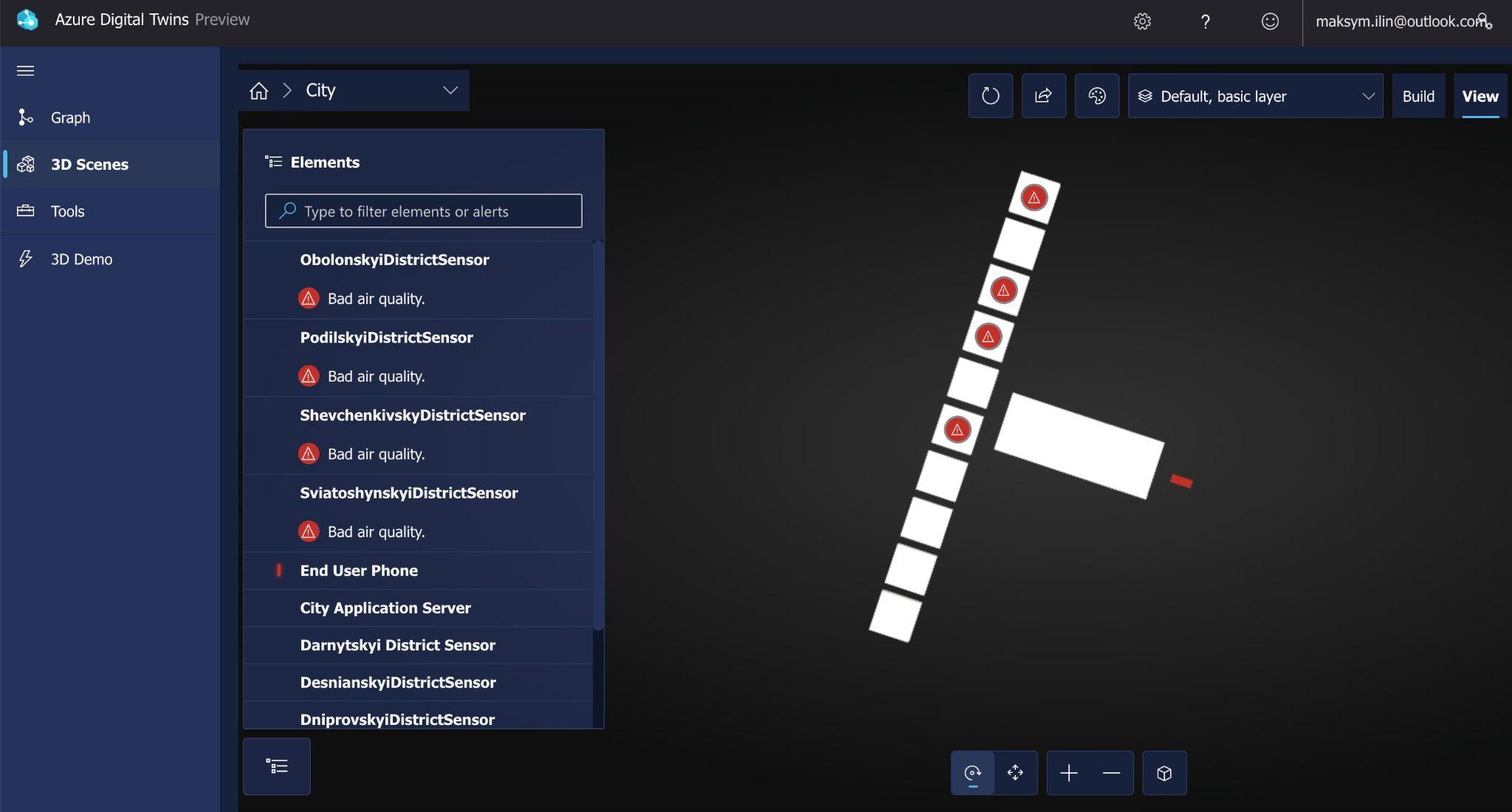


Рис. 5. Сцена, що відображає актуальний стан двійників у Azure Digital Twins 3D Scenes

## Модуль аналітики

Для реалізації модулю аналітики було використано сервіс Azure Maps.

Серверну частину було реалізовано за допомогою python-фреймворку Flask та інтерфейсу, реалізованому із використанням HTML/CSS/JS.

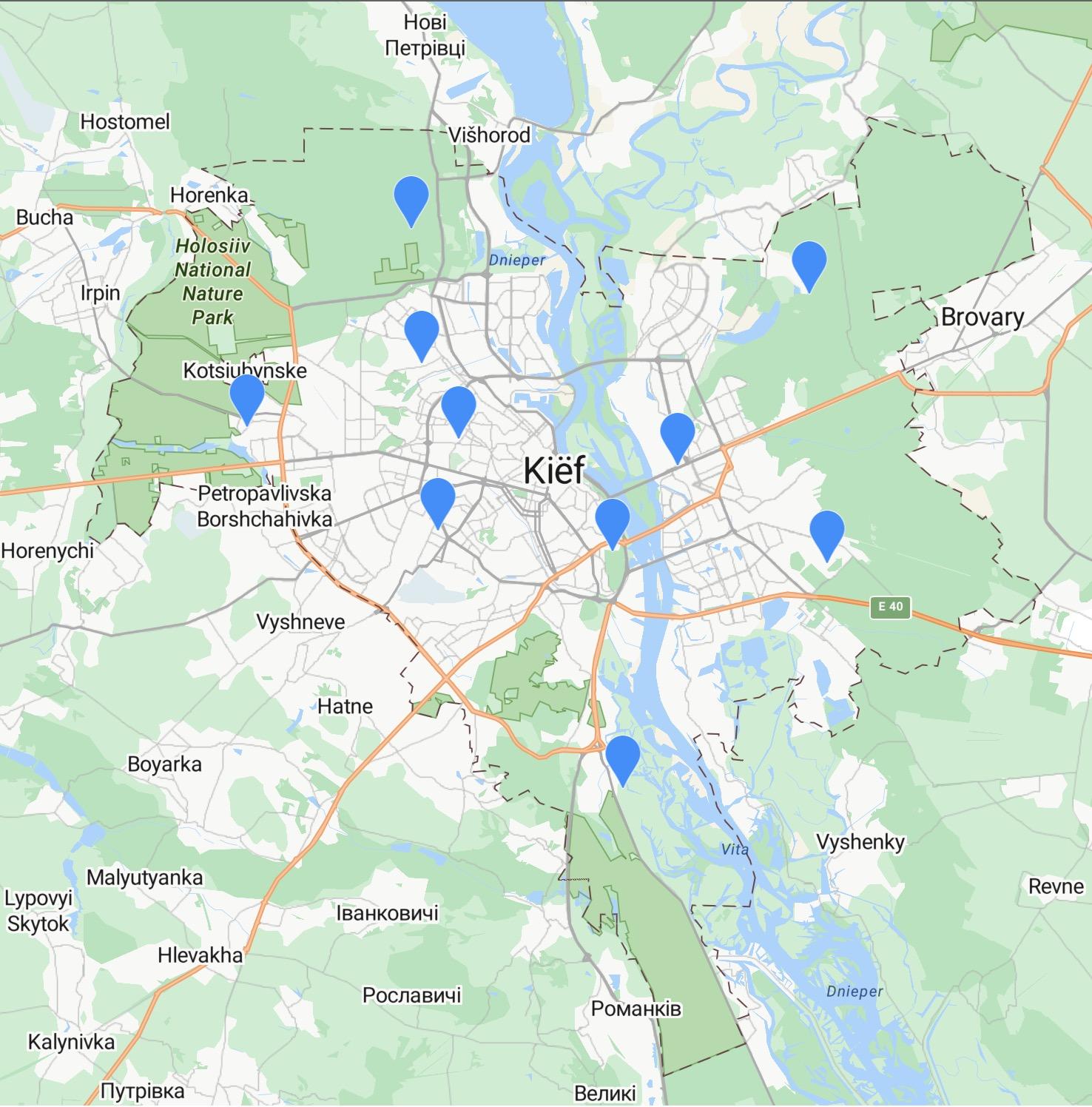


Рис. 6. Відображення наявних сенсорів на Azure Maps

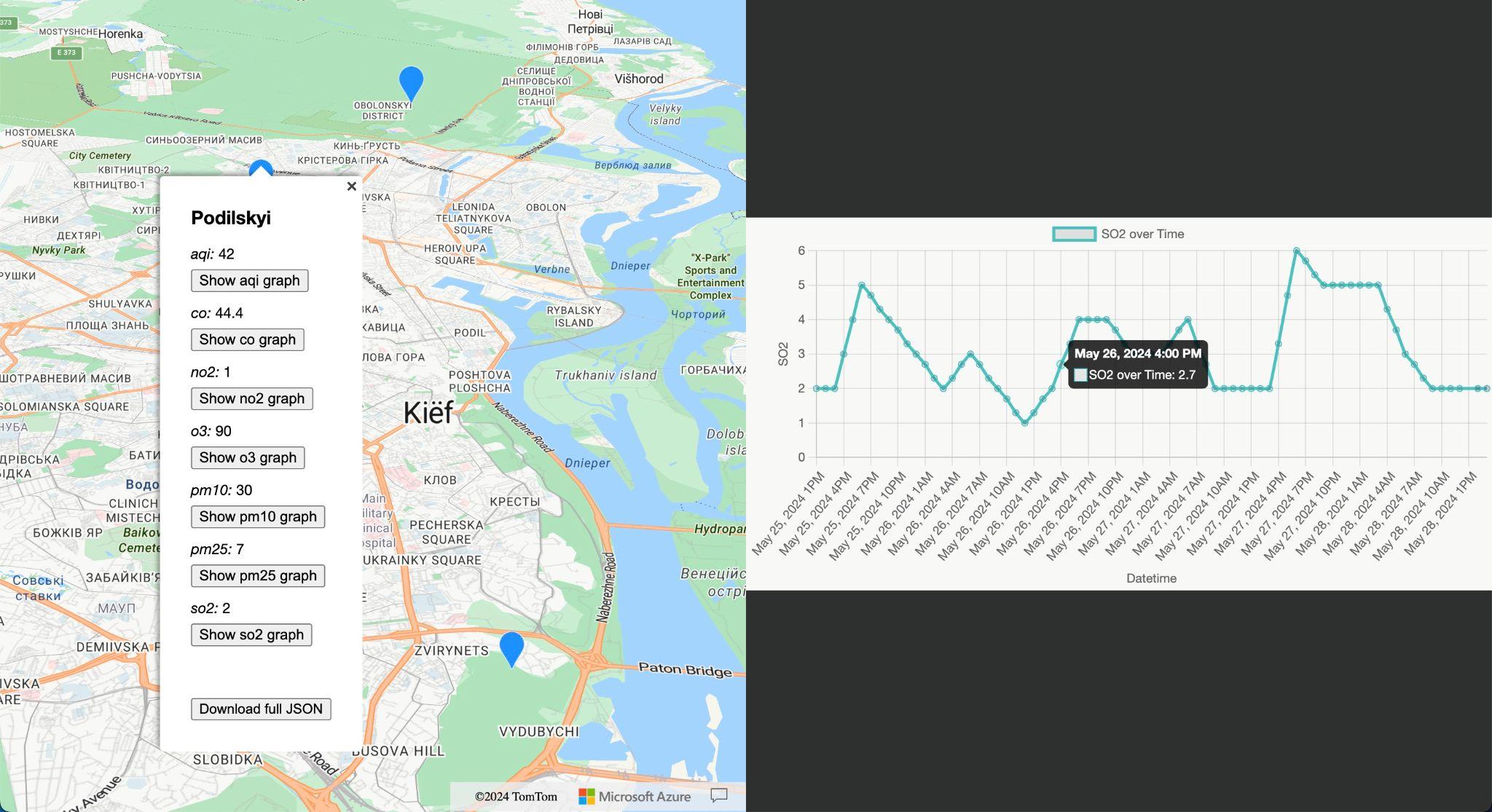


Рис. 7. Відображення актуальних значень показників забруднень та інтерактивного графіку історичних даних

## Результат роботи

Результат роботи, документація, відеопрезентація і всі супутні матеріали знаходяться [за посиланням](https://github.com/Miracle-Aligner/air_quality_dt).

# ВИСНОВКИ

В даній комплексній лабораторній роботі було створено Цифровий Двійник системи моніторингу якості повітря в містах. Для реалізації ЦД було використано платформу Azure Digital Twins. Реалізацію візуальної репрезентації було розділено на дві частини:

* Azure Digital Twins 3D Scenes;
* Flask + HTML/CSS/JS.

Розроблена система повністю відповідає технічному завданню і реалізовує відповідний функціонал із візуалізації цифрового двійника та аналітики даних, що отримуються в результаті його роботи.