Міністерство освіти і науки України
Криворізький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерних наук і технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти – бакалавр

за освітньо-професійною програмою

«Комп'ютерні науки»

зі спеціальності

122 – Комп'ютерні науки

тема роботи:

«Розробка веб-сервісу для візуалізації результатів екологічного моніторингу джерел викидів для умов ПАТ «АрселорМіттал Кривий Ріг»»

Виконала студентка гр. КН-21 _____ Кожемятова К. К.

Керівник _____ Рубан С. А.

Нормоконтроль _____ Маринич І. А.

Завідувач кафедри _____ Рубан С. А.

Кривий Ріг – 2025

КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет: *інформаційних технологій*

Кафедра: *автоматизації, комп'ютерних наук і технологій*

Ступінь вищої освіти: *Бакалавр*

Спеціальність: *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Зав. кафедрою: *к.т.н. Рубан С.А.*

_____

« *29* » *січня* 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

студентці групи КН-21 *Кожемятовій Катерині Костянтинівні*

1. Тема кваліфікаційної роботи: *«Розробка веб-сервісу для візуалізації результатів екологічного моніторингу джерел викидів для умов ПАТ «АрселорМіттал Кривий Ріг»»*

затверджено наказом по університету № *254с* від *13.05.2025 р.*

2. Термін здачі кваліфікаційної роботи: *05.06.2025 р.*

3. Склад кваліфікаційної роботи: *Пояснювальна записка обсягом 68 с., додатки, презентація у Microsoft PowerPoint (13 слайдів) в електронному та друкованому вигляді*

4. Консультанти кваліфікаційної роботи:

Розділ 1-2 доц. Рубан С. А.

Нормоконтроль доц. Маринич І. А.

5. Календарний план:

№	Етапи роботи	Термін виконання
1	*Вступ*	*01.04.25*
2	*Розділ 1*	*05.04.25*
3	*Розділ 2*	*01.05.25*
4	*Висновки*	*25.05.25*
5	*Оформлення кваліфікаційної роботи*	*28.05.25*
6	*Підготовка презентації та графічного матеріалу*	*20.05.25*
7	*Підготовка доповіді до захисту*	*05.06.25*

6. Дата видачі завдання: *29.01.2025р.*

Керівник _____ / Рубан С. А./

7. Запевнення: *Я, Кожемятова Катерина Костянтинівна, запевняю, що ця кваліфікаційна робота виконана самостійно, не містить академічного плагіату, фабрикації, фальсифікації. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

Із чинним Положенням про академічну доброчесність Криворізького національного університету ознайомлена.

Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі умисних порушень робота не допускається до захисту або оцінюється незадовільно.

Студент _____ / Кожемятова К. К./

АНОТАЦІЯ

Кожемятова К. К. Розробка веб-сервісу для візуалізації результатів екологічного моніторингу джерел викидів для умов ПАТ «АрселорМіттал Кривий Ріг» : кваліфікаційна робота бакалавра : 122 – Комп’ютерні науки. Кривий Ріг. Криворізький національний університет, 2025. 68 с.

Насьогодні моніторинг стану навколишнього середовища та потенційних джерел викидів є важливою складовою екологічної політики, сприяє обізнаності населення щодо актуального стану довкілля, дозволяє місцевій владі формувати політику стимулювання впровадження екологічно сталих рішень, а підприємствам – відчувати відповідальність за дотримання норм, правил та регламентів технологічних процесів.

Метою роботи є розробка прототипу веб-сервісу, що дозволяє зберігати та відображати у вигляді графіків контрольовані показники екологічного моніторингу джерел викидів на межі захисної зони металургійного підприємства.

У вступі наведено обґрунтування актуальності теми роботи, сформульована мета та визначені основні завдання дослідження.

Перший розділ присвячено аналізу наявних на ринку рішень та можливих способів реалізації веб-сервісі, сформульовані функціональні вимоги.

Другий розділ присвячений розробці архітектури системи, проєктуванню бази даних, реалізації основного функціоналу додатку, а також апробації запропонованих рішень та опису методики використання додатку.

Практична цінність результатів роботи полягає у створення прототипу веб-сервісу, що легко масштабується і є чудовим каркасом для подальшого розвитку та удосконалення, а у подальшому і для впровадження в умовах ПАТ «АрселорМіттал Кривий Ріг».

Ключові слова:

ВЕБ-СЕРВІС, ЕКОЛОГІЧНИЙ МОНІТОРИНГ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ФРЕЙМВОРК, ASP.NET CORE, ПРОТОКОЛ MQTT

ЗМІСТ

ВСТУП

У сучасних умовах промислова діяльність значною мірою впливає на стан довкілля, що обумовлює необхідність створення ефективних інструментів екологічного контролю. Особливої актуальності це набуває в металургійній галузі, де технологічні процеси супроводжуються викидами шкідливих речовин, що можуть мати суттєве вплив на атмосферний повітря, ґрунти та водні ресурси. Забезпечення належного екологічного нагляду вимагає впровадження сучасних інформаційних систем, які дозволяють здійснювати автоматизований збір, обробку, збереження та візуалізацію екологічних даних в режимі, наближеному до сьогодення.

Зростання доступності сенсорного обладнання, поширення IoT-рішень, розвиток відкритих протоколів обміну даними (зокрема MQTT) та потужних серверних платформ на кшталт ASP.NET Core створюють передумови для побудови гнучких та масштабованих систем екологічного моніторингу. При цьому особливе значення мають такі вимоги до системи, як зручність адміністрування об’єктів спостереження, ефективне зберігання та обробка телеметричних даних, наочне та інтуїтивно зрозуміле відображення інформації, а також можливість інтеграції з аналітичними модулями для подальшої обробки результатів моніторингу.

Метою цієї роботи є проектування та реалізація інформаційної системи візуалізації даних екологічного моніторингу для металургійного підприємства.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

– виконати огляд та аналіз предметної області, зокрема існуючих рішень у галузі систем екологічного моніторингу;

– проаналізувати методологічні підходи та перспективні технології для реалізації системи;

– розробити адміністративну панель для управління параметрами,

агрегатами та підрозділами;

– реалізувати сервіс обробки екологічних даних через протокол MQTT та інтерфейси візуалізації історичних значень.

Практична цінність полягає у створенні адаптованої до промислових умов системи, яка дозволяє підприємствам підвищити рівень екологічної відповідальності, автоматизувати облік викидів та підтримувати прийняття рішень на основі актуальних даних.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ МОЖЛИВИХ ПІДХОДІВ ДО ВИРІШЕННЯ ЗАВДАННЯ РОЗРОБКИ ВЕБ-СЕРВІСУ ВІЗУАЛІЗАЦІЇ РЕЗУЛЬТАТІВ ЕКОЛОГІЧНОГО МОНІТОРИНГУ ДЛЯ УМОВ МЕТАЛУРГІЙНОГО ВИРОБНИЦТВА

1.1 Аналіз предметної галузі та існуючих рішень з візуалізації даних екологічного моніторингу

На сучасному етапі розвитку цифрових технологій для реалізації систем екологічного моніторингу доступна низка готових рішень. Такі системи дозволяють конфігурувати параметри спостереження, вказувати джерела даних, встановлювати періодичність збору показників, здійснювати їх прив’язку до технологічних об’єктів і надавати кінцевим користувачам можливість переглядати динаміку зміни екологічних параметрів у часі.

Серед найбільш поширених підходів можна виділити такі:

1. Комерційні екологічні платформи (зокрема AQMIS, Enablon, Ecometrica, Sphera) орієнтовані на великі промислові підприємства та забезпечують облік викидів, контроль за дотриманням стандартів (наприклад, ISO 14001), формування звітності тощо [1-5]. Основними перевагами є комплексність функцій і підтримка регуляторних вимог, а недоліками — висока вартість та складність адаптації.

2. SCADA-системи з екологічними модулями (наприклад, Siemens WinCC, GE iFIX, Ignition) надають змогу моніторити параметри в режимі реального часу з безпосереднім підключенням до сенсорів. Такі рішення ефективні на підприємствах, де вже впроваджено системи автоматизації, однак потребують спеціалізованого обслуговування.

3. Рішення на базі відкритих інструментів, таких як Grafana, InfluxDB, Telegraf або Node-RED забезпечують високу гнучкість і масштабованість. Вони дозволяють збирати дані з сенсорів або API, зберігати їх у часових базах і будувати інтерактивні дашборди. Цей підхід є особливо доцільним для проєктів

із обмеженим бюджетом, хоча вимагає ручного налаштування та самостійної реалізації адміністративного інтерфейсу. Такий підхід, зокрема, використовується для збору та візуалізації інформації на КП Кривбасводоканал.

4. Ще одним підходом є використання стандарту SensorThings API від OGC, зокрема реалізації FROST Server [6-13]. Він дозволяє моделювати зв’язок між спостереженнями, сенсорами, об’єктами та просторово-часовими параметрами. Система забезпечує API-доступ до даних, однак потребує створення окремого візуального інтерфейсу для користувачів.

5. Нарешті, можлива розробка власної веб-системи, яка повністю адаптується до потреб підприємства. Така система дозволяє реалізувати індивідуальну модель параметрів, джерел і звітності, проте потребує значних ресурсів на етапі розробки, впровадження та супроводу.

Далі розглянемо основні наведені рішення більш детально.

Система AQMIS (Air Quality Management Information System) є однією з провідних комерційних платформ, призначених для управління даними про якість повітря та екологічний мониторинг на рівні підприємств, регіонів або держав [1, 2].

Функціональні можливості AQMIS охоплюють повний цикл управління екологічною інформацією [1]. Система дозволяє:

– приймати дані з автоматизованих станцій моніторингу, сенсорів та лабораторних вимірювань;

– здійснювати перевірку, нормалізацію та валідацію даних;

– зберігати інформацію у централізованій базі з підтримкою історичних архівів;

– виконувати аналіз відповідності фактичних показників нормативним значенням;

– формувати регулярну звітність відповідно до вимог місцевого чи міжнародного екологічного законодавства;

– будувати інтерактивні графіки, карти загрязнення, тренди та прогнози;

– налаштовувати сповіщення про перевищення гранично допустимих

концентрацій.

Система орієнтована на використання державними органами, муніципалітетами, великими промисловими підприємствами та консалтинговими структурами, що займаються питаннями екології та охорони навколишнього середовища. Так, на рис. 1.1 наведено скріншот панелі відображення (дашборда) показників якості повітря (рівень CO, NO_2, O_3), створений з використанням хмарної версії AQMIS Cloud [2].

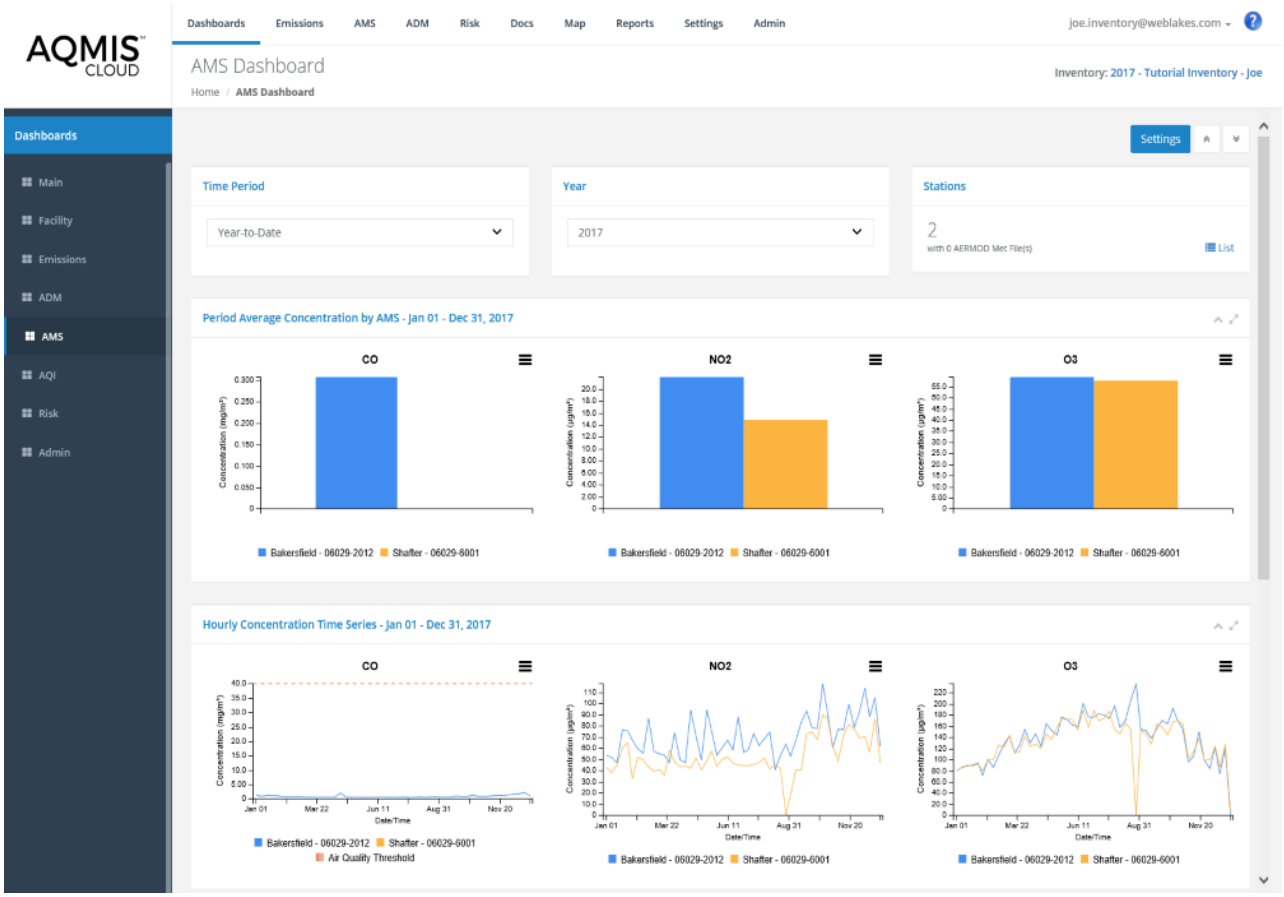


Рисунок 1.1 – Приклад інтерфейсу сторінки панелі відображення показників якості повітря з використанням AQMIS Cloud

До основних переваг AQMIS можна віднести [1]:

– високу надійність і масштабованість , що дозволяє впроваджувати її на рівні міст чи цілих регіонів;

– підтримку нормативних вимог , зокрема сумісність із ISO 14000, EPA та іншими міжнародними стандартами;

– наявність гнучкого механізму конфігурації джерел даних та підтримки різних протоколів збирання інформації;

– розвинені інструменти візуалізації та звітності, що сприяють прозорості екологічного моніторингу;

– інтерфейс користувача, орієнтований на нетехнічних фахівців, що забезпечує зручність у роботі з даними.

Водночас, система має певні обмеження, які варто враховувати при виборі інструменту для конкретного проекту. Основні недоліки AQMIS полягають у:

– високій вартості ліцензування та впровадження, що може бути критичним фактором для невеликих підприємств;

– складності адаптації до специфічних внутрішніх процесів замовника без залучення сторонніх консультантів чи розробників;

– обмеженій кастомізації інтерфейсів та алгоритмів обробки даних в межах готового рішення;

– залежності від постачальника при масштабуванні чи оновленні системи.

Таким чином, AQMIS є потужною та перевіреною у практиці платформою, яка найкраще підходить для впровадження на об’єктах із підвищеними вимогами до якості та відповідності екологічним стандартам. У контексті автоматизації екологічного моніторингу металургійного підприємства її використання доцільно розглядати у випадках, коли передбачено централізовану звітність, зовнішній аудит або інтеграцію з регіональними системами екологічного моніторингу.

Enablon – це комплексна програмна платформа для управління ризиками, охороною навколишнього середовища, охороною праці та соціальною відповідальністю (EHS&S), яка активно використовується у великих промислових підприємствах у всьому світі [3]. Однією з ключових функціональних можливостей Enablon є підтримка екологічного моніторингу, включно зі збором, обробкою та візуалізацією екологічних показників у відповідності до національних і міжнародних екологічних стандартів.

Платформа дозволяє інтегрувати дані з різноманітних джерел – датчиків, лабораторних вимірювань, внутрішніх систем контролю або зовнішніх провайдерів – із можливістю їх централізованого зберігання та автоматичної перевірки на відповідність встановленим нормативам. Однією з важливих функцій є графічне подання динаміки змін екологічних параметрів у реальному або ретроспективному режимі. Завдяки гнучкому механізму побудови інформаційних панелей (дашбордів), користувачі можуть оперативно виявляти відхилення, формувати звіти та приймати управлінські рішення на основі актуальних даних (рис. 1.2).

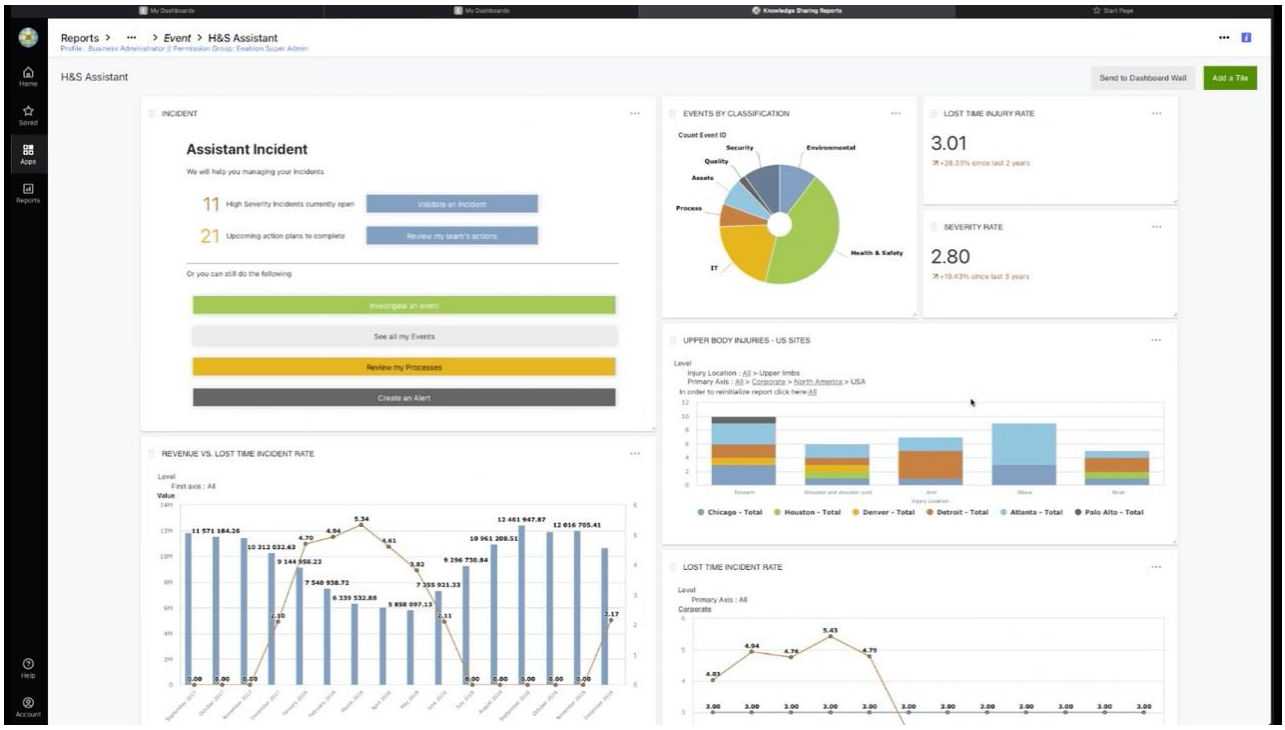


Рисунок 1.2 – Приклад інтерфейсу сторінки панелі відображення екологічних показників з використанням ПЗ Enablon

У контексті промислового підприємства, такого як металургійний комбінат, Enablon може використовуватися для моніторингу рівня викидів, стану атмосферного повітря, водовідведення, відходів виробництва, а також для відстеження виконання природоохоронних зобов’язань. Система підтримує налаштування допустимих рівнів для кожного контрольованого параметра та автоматично сигналізує про порушення.

Серед переваг Enablon – висока масштабованість, підтримка інтеграції з ERP-системами, широкі аналітичні можливості, а також відповідність міжнародним нормам у сфері екологічного управління (наприклад, ISO 14001). До недоліків можна віднести високу вартість ліцензування, складність початкового налаштування та потребу у спеціалізованому навчанні персоналу.

Таким чином, Enablon є ефективним інструментом для побудови централізованих систем екологічного моніторингу з функціями візуалізації, аналізу та контролю екологічної ситуації на підприємстві.

Ecometrica Platform – це сучасна екологічна інформаційна система, призначена для збирання, обробки, аналізу та візуалізації даних, пов’язаних із впливом діяльності підприємства на навколишнє середовище. Вона широко використовується компаніями та організаціями, які прагнуть забезпечити стале розвиток, покращити прозорість екологічної звітності та відповідати міжнародним стандартам у сфері сталого управління [4].

Платформа орієнтована насамперед на оцінку викидів парникових газів, впливів на клімат, управління лісовими ресурсами, просторовий моніторинг та аналіз показників сталого розвитку. Вона підтримує різноманітні підходи до збору даних, включаючи інтеграцію з IT-системами підприємства, імпорт з таблиць Excel, використання геопросторової інформації та супутникових даних. Однією з ключових особливостей є використання картографічної візуалізації для аналізу впливу на довкілля у реальному чи наближеному до реального часі.

Основні функціональні можливості Ecometrica включають [5]:

– створення інвентаризацій викидів парникових газів на основі стандартизованих методологій (наприклад, GHG Protocol);

– побудову інтерактивних звітів та дашбордів за екологічними показниками;

– просторовий аналіз даних з використанням супутникових знімків та геоінформаційних систем (ГІС);

– підтримку користувацьких методик обчислення впливів;

– автоматизоване формування звітності відповідно до стандартів CDP,

TCFD, GRI тощо;

– надання доступу до аналітичних результатів для різних категорій користувачів (менеджмент, аудитори, громадськість).

Серед ключових переваг системи Ecometrica можна виділити:

– високу гнучкість конфігурації для різних типів організацій та секторів економіки;

– підтримку геопросторової аналітики, що дає змогу враховувати локаційний контекст впливів на довкілля;

– відповідність провідним міжнародним екологічним та сталим стандартам;

– інтуїтивно зрозумілий інтерфейс та наявність модулів візуалізації, які спрощують аналітику для користувачів без технічного досвіду;

– модульну структуру, яка дозволяє розширювати функціональність в межах однієї платформи.

Незважаючи на зазначені переваги, платформа має і певні обмеження. До недоліків Ecometrica слід зарахувати:

– значну вартість ліцензії та підписки, що робить її менш доступною для малих та середніх підприємств;

– залежність від хмарної інфраструктури, що в окремих випадках може бути критичним фактором для підприємств зі специфічними вимогами до зберігання даних;

– орієнтацію насамперед на західні екологічні стандарти, що потребує адаптації при впровадженні в регіонах із локальними нормативами;

– обмеження на глибоку кастомізацію користувацького інтерфейсу без залучення команди розробників.

Загалом Ecometrica є ефективним інструментом для підприємств і організацій, які прагнуть здійснювати екологічний моніторинг на стратегічному рівні, інтегруючи фінансові, кліматичні та просторові аспекти оцінки сталості.

У межах розробки відкритих систем екологічного моніторингу особливу увагу привертає стандарт SensorThings API [6-9], розроблений Open Geospatial

Consortium (OGC). Він призначений для уніфікованого представлення, обміну та інтеграції даних, отриманих від сенсорів і пристроїв Інтернету речей (IoT). Цей стандарт забезпечує інтероперабельність між різними джерелами даних і підтримує просторово-часовий контекст спостережень.

Так, на рис. 1.3 наведено приклад застосування SensorThings API для візуалізації даних моніторингу у рамках досліджень, що проводяться організацією British Geological Survey [10].

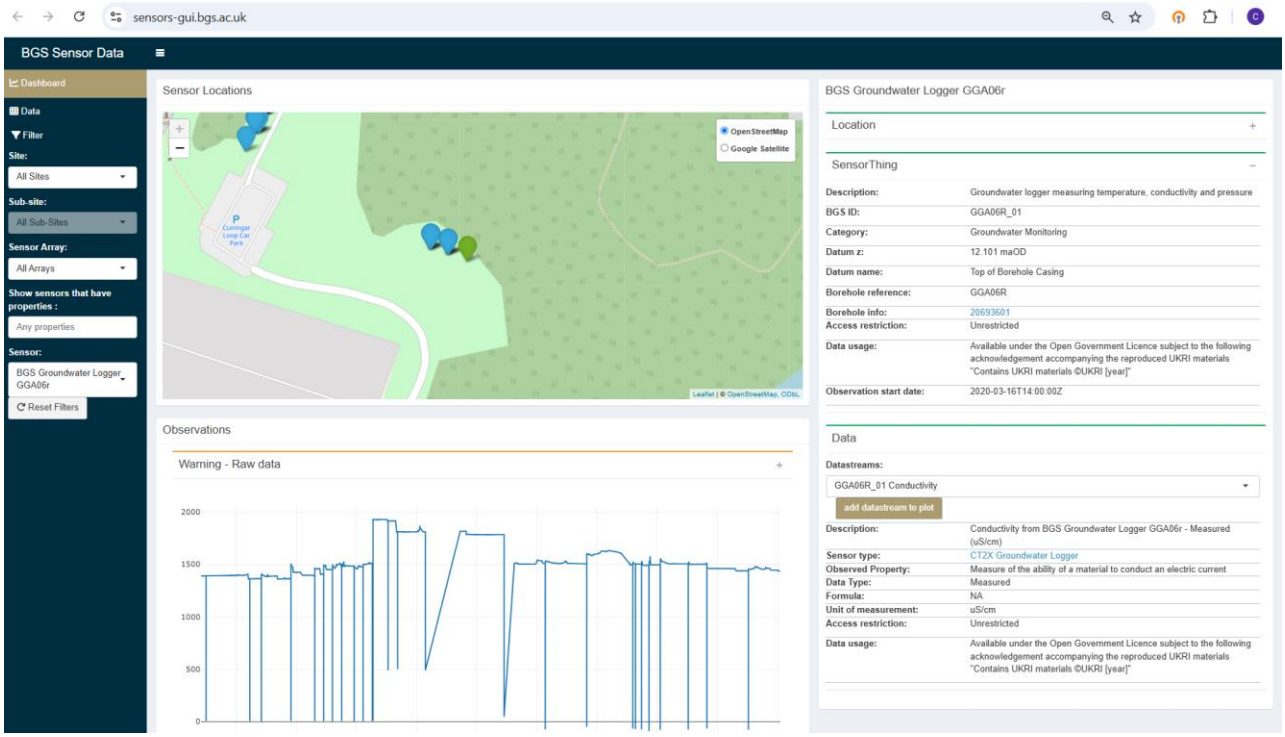


Рисунок 1.3 – Приклад інтерфейсу сторінки візуалізації даних моніторингу з використанням SensorThings API

Однією з найпоширеніших реалізацій цього стандарту є FROST-Server (Flexible Real-time Open Source Things Server) – відкрите серверне програмне забезпечення, що підтримує як REST API, так і MQTT-протокол для обміну повідомленнями в режимі реального часу [11]. Платформа реалізує повну модель даних SensorThings API, включаючи сутності: Thing (об’єкт), Location, Datastream, Sensor, ObservedProperty, Observation та FeatureOfInterest (рис. 1.4).

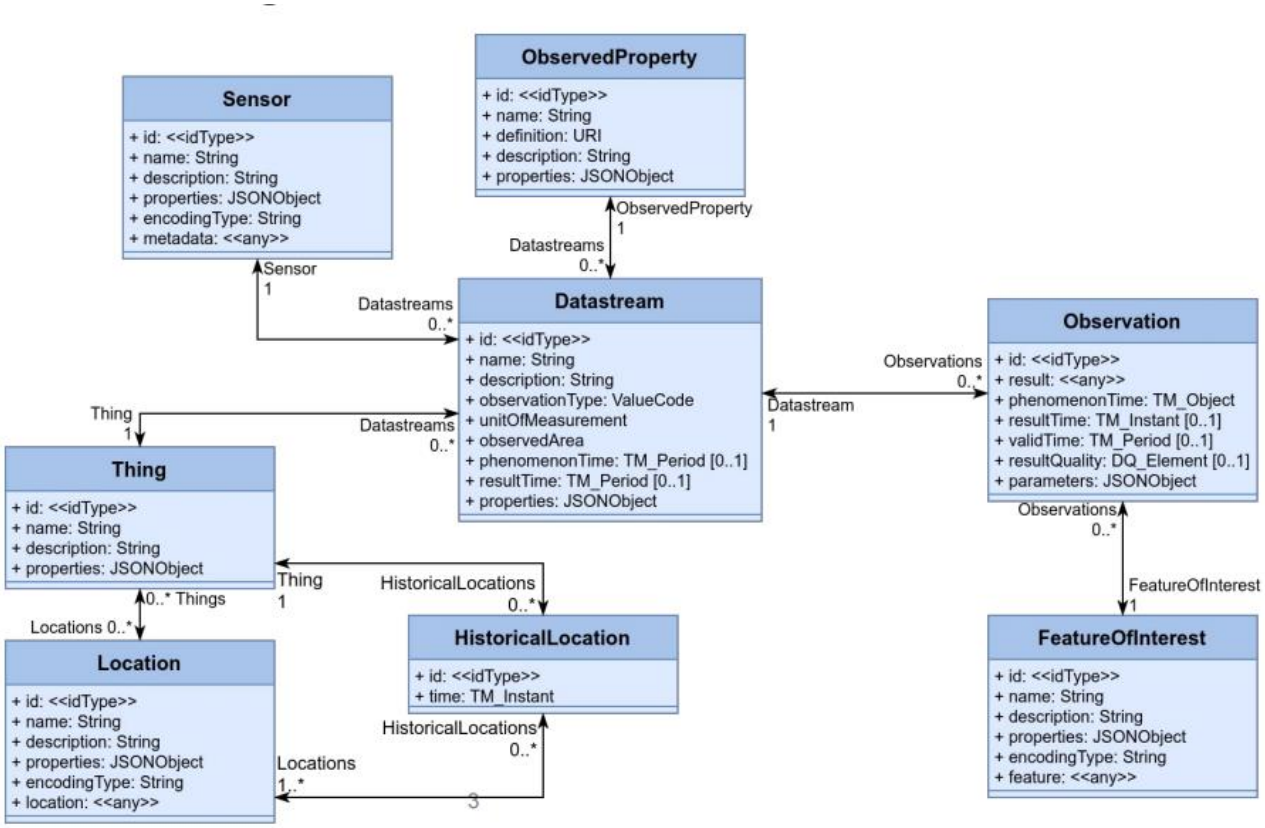


Рисунок 1.4 – Модель даних SensorThings API

Архітектурно FROST-Server побудований за модульним принципом, що дозволяє масштабувати систему відповідно до обсягів даних [12, 13]. Основними компонентами є: інтерфейс API для клієнтського доступу, механізм обробки повідомлень у реальному часі (MQTT-брокер), ORM-рівень для взаємодії з базою даних (PostgreSQL) та компоненти для фільтрації й агрегації даних (рис. 1.5).

Системи на базі FROST-Server знаходять застосування у міських екологічних ініціативах, системах Smart City, моніторингу водних об’єктів та атмосферного повітря. Наприклад, у Нідерландах реалізовано проєкт Smart Emission, в якому FROST використовується для збору даних про якість повітря з великої кількості міських сенсорів. У країнах ЄС FROST-Server використовується для інтеграції метеорологічних, гідрологічних і супутникових даних у межах регіональних платформ відкритих даних.

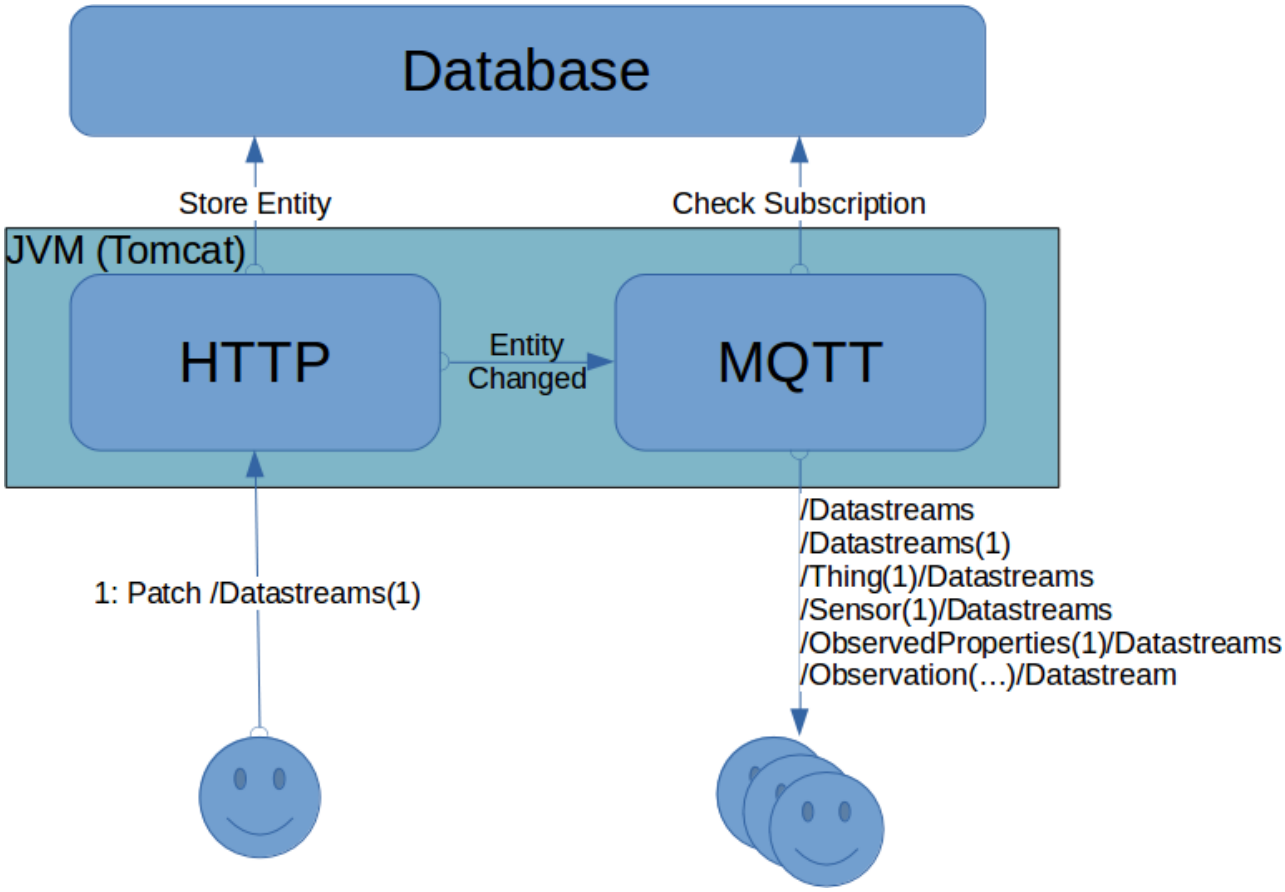


Рисунок 1.5 – Узагальнена архітектура FROST-серверу

Перевагами платформи є її відкритість і відповідність міжнародним стандартам, можливість гнучкого моделювання об’єктів, підтримка як історичних, так і потокових даних, а також легка інтеграція з інструментами для візуалізації та аналізу. Завдяки підтримці просторової прив’язки, система ефективно працює у поєднанні з геоінформаційними платформами (наприклад, QGIS, ArcGIS, Grafana).

Попри численні переваги, система має і певні недоліки. Зокрема, відсутність готового інтерфейсу користувача вимагає розробки або налаштування клієнтських рішень. Первинне розгортання, конфігурація API, безпека та підтримка авторизації також потребують високої технічної кваліфікації. Крім того, SensorThings API орієнтований насамперед на структуру даних, а не на реалізацію бізнес-логіки, що обумовлює потребу у зовнішніх компонентах для звітності, порівняння з нормативами чи обчислень.

Таким чином, використання FROST-Server у системах екологічного моніторингу є доцільним у випадках, коли пріоритетом є відкритість, масштабованість, інтеграція з геоданими та підтримка високочастотного збору показників. Його реалізація є ефективним кроком у напрямку створення гнучких, сучасних та інтероперабельних платформ моніторингу навколишнього середовища.

Серед існуючих на ринку рішень доцільно розглянути також українські розробки та сервіси. Одним з найвідоміших рішень у галузі онлайн-моніторингу екологічних показників є сервіс SaveEcoBot [20], що уявляє собою українську онлайн-платформу, орієнтовану на моніторинг та публічний доступ до екологічної інформації, зокрема показників якості атмосферного повітря (рис 1.6).

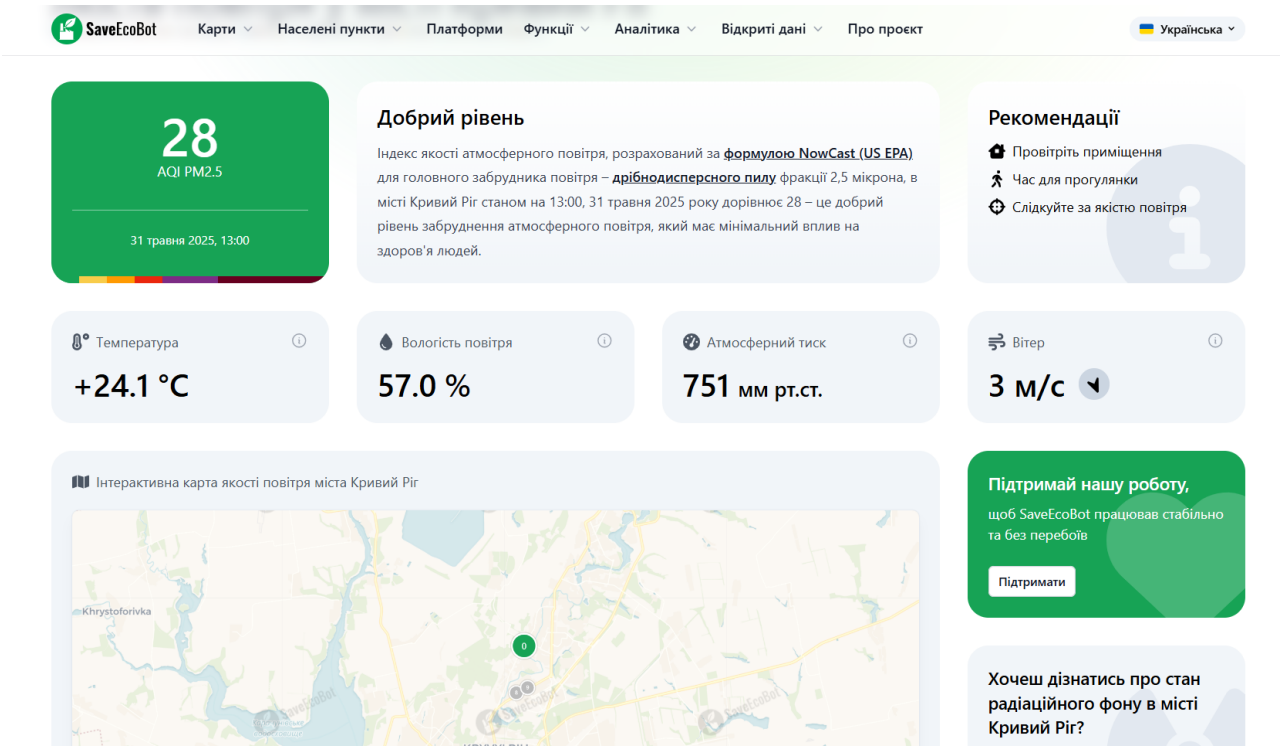


Рисунок 1.6 – Сторінка відображення показників якості повітря у м. Кривий Ріг на платформі SaveEcoBot

Платформа була створена для підвищення прозорості екологічних даних та залучення громадськості до питань охорони навколишнього середовища.

Вона об’єднує дані з чисельних станцій моніторингу, як офіційних (державних та муніципальних), так і громадських.

SaveEcoBot має відкритий веб-інтерфейс, який надає користувачам доступ до картографічної візуалізації даних у реальному часі. На інтерактивній карті відображаються пости моніторингу з актуальними показниками забруднення атмосферного повітря, такими як концентрації пилу PM2.5 та PM10, вміст оксиду вуглецю, діоксиду азоту тощо. Це дозволяє відслідковувати просторово-годинну динаміку забруднення, що є важливою функцією для оцінки екологічної ситуації у промислових регіонах. Так, на рис. 1.7 наведено графік змінення середнього індексу якості повітря у м. Кривий Ріг за інтервал часу у 21 день. Також існує можливість завантажити архів з даними у форматі CSV.

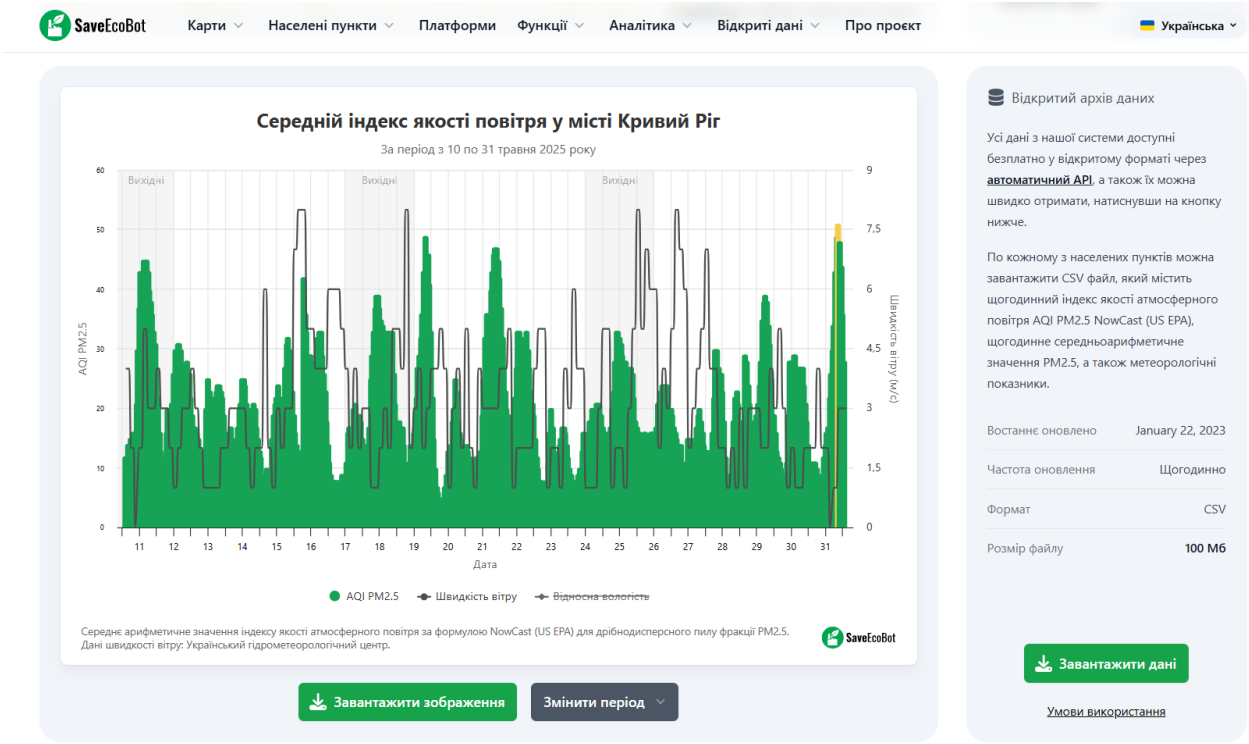


Рисунок 1.7 – Приклад графіку відображення середнього індексу якості повітря на платформі SaveEcoBot

З точки зору реалізації системи візуалізації екологічних показників в межах металургійного виробництва, SaveEcoBot демонструє приклад інтеграції даних з різних джерел, відображення результатів на карті та у вигляді часових

графіків. Але, з іншого боку, громадський сегмент не має специфічних функцій для промислових підприємств (наприклад, розмежування прав доступу, адміністрування агрегатів та нормативів), які є критично важливими для корпоративних систем екологічного моніторингу.

Перевагою SaveEcoBot є відкритий доступ до даних через публічні API [20], що дає змогу використовувати його як джерело зовнішньої інформації при розробці власних систем візуалізації екологічних даних. Проте для реалізації комплексної системи моніторингу в умовах металургійного виробництва необхідне створення більш гнучкої архітектури, яка враховує особливості підприємства: багаторівневий доступ, налаштування контрольованих параметрів та інтеграцію з промисловими датчиками через MQTT або OPC-UA протоколи.

В цілому, SaveEcoBot є важливим українським інструментом для підвищення освіченості населення щодо екологічної ситуації, а також може бути прикладом для розробки корпоративних платформ, що відображають та аналізують дані про стан довкілля в межах конкретного підприємства.

Ще одним проєктом у галузі екологічного моніторингу є EcoCity – це українська громадська платформа моніторингу якості атмосферного повітря, яка об’єднує мережу автоматизованих станцій, розміщених у різних регіонах України [21]. Проект реалізується неприбутковою громадською організацією «Фрі Ардуіно» та спрямований на підвищення екологічної освіченості населення шляхом надання відкритого доступу до актуальної інформації про стан повітря.

Платформа надає інтерактивну карту, де в реальному часі відображаються дані про концентрацію забруднюючих речовин, таких як дрібнодисперсна пилка (PM1, PM2.5, PM10), а також метеорологічні параметри – температура, вологість, атмосферний тиск (рис. 1.8). Користувачі можуть отримувати сповіщення про перевищення допустимих норм забруднення та переглядати історичні дані вимірювань.

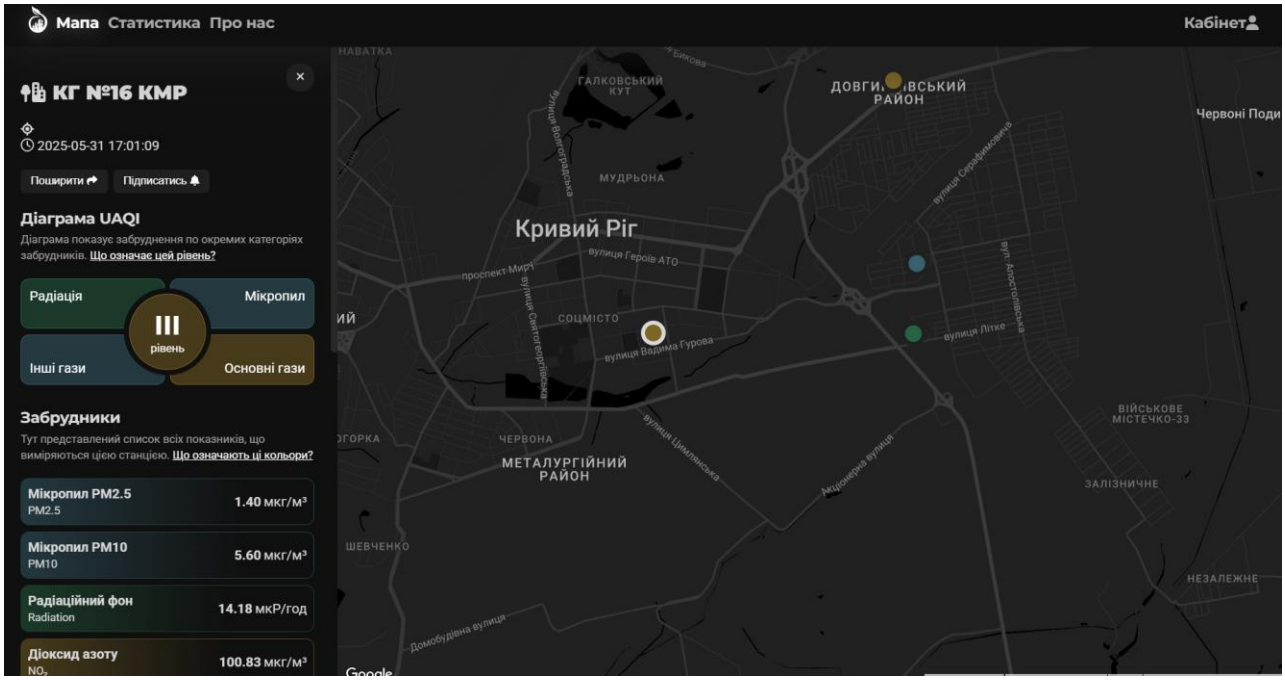


Рисунок 1.8 – Сторінка відображення показників якості повітря для одного з постів моніторингу у м. Кривий Ріг на платформі EcoCity

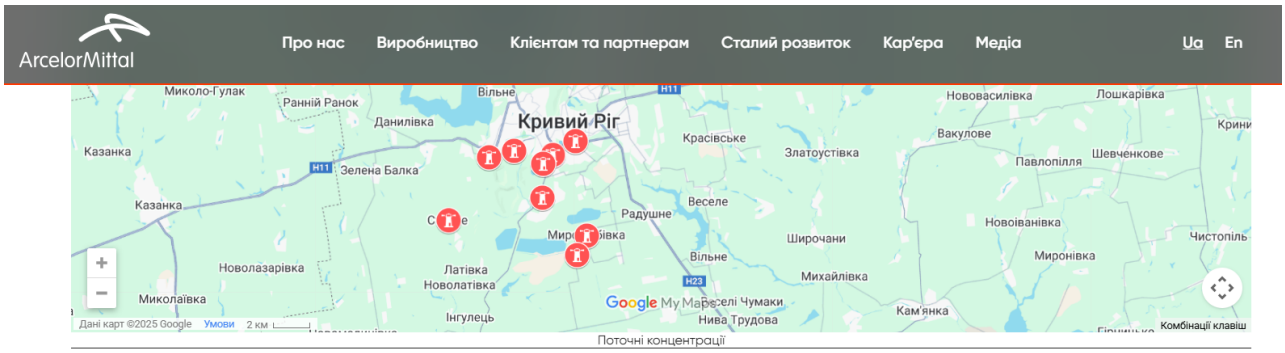
Важливою особливістю EcoCity є її відкритість: дані, зібрані станціями, доступні для громадськості, що сприяє прозорості та прилученню громадян до питань екологічного моніторингу. Крім того, платформа підтримує інтеграцію з іншими сервісами та дозволяє користувачам встановлювати власні станції моніторингу, що забезпечує гнучкість та розширюваність системи.

У контексті металургійного виробництва, де контроль за викидами є критично важливим, EcoCity може бути прикладом для розробки внутрішніх систем моніторингу. Зокрема, використання подібної архітектури дозволяє підприємствам впроваджувати власні мережі сенсорів для збору даних про забруднення з подальшою візуалізацією та аналізом інформації. Це сприяє своєчасному виявленню перевищень нормативів та прийняттю відповідних управлінських рішень.

Таким чином, EcoCity є ефективним інструментом для громадського моніторингу якості повітря та може бути основою для створення корпоративних систем візуалізації екологічних показників, особливо у галузях з підвищеним рівнем викидів, таких як металургія.

Також варто відзначити наявні регіональні ініціативи підприємств Криворіжжя. Управлінням екології виконавчого комітету Криворізької міської ради було розроблено систему екологічного моніторингу, доступ до якої здійснювався через міський геопортал. Але під час військового стану з міркувань безпеки доступ користувачів до порталу тимчасово заблоковано.

ПАТ «АрселорМіттал Кривий Ріг» також поступово впроваджує власну автоматизовану систему моніторингу атмосферного повітря на межі санітарно-захисної зони підприємства [22]. Розробка системи почалася з 2014 року, коли з'явились перші 3 автоматизовані пости контролю якості повітря і результати контролю почали викладати у вільний доступ. Протягом 2022-2023 років було введено в експлуатацію ще 6 постів, виконано інтеграцію даних в міську систему екологічного моніторингу. Так, на рис. 1.9 наведено відображення виміряних показників екологічного моніторингу на сайті ПАТ «АрселорМіттал Кривий Ріг».



Поточні концентрації

Дані щодо вмісту пилу на АПС №3 відсутні з технічних причин.
Метеопоказники на АПС №2 відсутні з технічних причин.

	Дата отримання показників	Діоксид азоту (NO2) мг/м3	Оксид азоту (NO) мг/м3	Діоксид сірки (SO2) мг/м3	Оксид вуглецю (CO) мг/м3	Аміак (NH3) мг/м3	Сірководень (H2S) мг/м3	Середня швидкість вітру(м/с)	Напрямок вітру(°)	Температура(°C)	Відносна вологість (%)	Атм. тиск (мм. рт. ст.)	Опади (мм)	Пил (мг/м3)	Пил PM 2.5 (мг/м3)	Пил PM 10 (мг/м3)
АПН 1	2025-05-31 14:00:00	0.007	0.001	0.001	0.476	–	–	X	X	X	X	X	X	0.017	0.002	0.003
АПН 2	2025-05-31 13:40:00	0.006	0.005	0.001	0.722	0.001	0.001	X	X	X	X	X	X	0.005	0.001	0.002
АПН 3	2025-05-31 14:00:00	0.009	0.002	0.026	0.688	–	–	2.656	261	27.2	43.2	753.2	0	X	0.001	0.001
АПК 4	2025-05-31 14:00:00	–	–	–	–	–	–	1.307	292	24.4	49.3	751.2	0.4	–	0.001	0.001
АПК 5	2025-05-31 14:00:00	–	–	–	–	–	–	2.143	328	26.7	44.4	756.1	0	–	0.001	0.001
АПК 6	2025-05-31 14:00:00	–	–	–	–	–	–	4.367	286	26	48.1	752.7	0	–	0.001	0.001
АПК 7	2025-05-31 14:00:00	–	–	–	–	–	–	2.212	344	26	48.3	751.7	0	–	0.001	0.001
АПК 8	2025-05-31 14:00:00	–	–	–	–	–	–	4.544	174	25.4	45.6	749.8	0	–	0.001	0.001
АПК 9	2025-05-31 14:00:00	0.008	–	–	–	–	–	5.135	206	25.2	43.8	748.4	4.4	–	0	0

Рисунок 1.9 – Сторінка відображення основних екологічним показників, що контролюються постами моніторингу на межі санітарно-захисної зони сайті ПАТ «АрселорМіттал Кривий Ріг»

Аналіз розглянутих існуючих систем екологічного моніторингу, зокрема Enablon, Ecometrica, SaveEcoBot та EcoCity, дозволяє визначити ключові функціональні можливості, які мають бути реалізовані в сучасних програмних рішеннях для збирання, обробки та візуалізації екологічних даних. Всі ці системи демонструють важливість централізованого зберігання інформації, здатності інтегрувати дані з різнорідних джерел та подання результатів у зручному графічному форматі.

Досвід цих рішень свідчить про те, що для промислових підприємств критично важливо забезпечувати не лише технічну реалізацію збору та відображення даних, а й організаційні механізми контролю доступу, гнучке адміністрування та відповідність екологічним стандартам. Ці аспекти повинні бути враховані при створенні власних корпоративних систем екологічного моніторингу, зокрема в металургійній промисловості, де контроль за рівнем викидів має пріоритетне значення.

1.2 Розробка технічних вимог до системи візуалізації даних екологічного моніторингу

Аналіз існуючих систем, виконаний у попередньому розділі, показав, що основними функціональними можливостями, що виділяються в системах екологічного моніторингу, є:

– автоматизований збір даних з використанням сенсорних пристроїв або зовнішніх джерел;

– збереження історичних даних та забезпечення їх доступності для аналізу;

– гнучка візуалізація інформації у вигляді графіків, карт та звітів, що дозволяє оперативно оцінювати ситуацію та виявляти відхилення від нормативів;

– система сповіщень про перевищення допустимих показників для запобігання екологічним інцидентам;

– інтероперабельність та відкриті API для розширення функціоналу та інтеграції з іншими корпоративними чи громадськими платформами.

У контексті металургійного виробництва, зокрема на основі інформації, що наведена на сторінці відображення результатів вимірювання екологічних показників, видно, що параметри контролю прив’язані до певних технологічних агрегатів, які в свою чергу до виробництва (структурного підрозділу).

Для формалізації та візуалізації наведених вимог доцільно розробити діаграму варіантів використання системи (рис. 1.10). Діаграма варіантів використання відображає функціональні можливості системи візуалізації даних екологічного моніторингу для металургійного підприємства та моделює взаємодію основних ролей користувачів із системою. Передбачено два типи користувачів: Користувач і Адміністратор . При цьому адміністратор має розширені повноваження та успадковує функціональність звичайного користувача. Такий підхід реалізовано за допомогою механізму узагальнення (generalization), що дозволяє не дублювати спільні функції та спрощує підтримку моделі.

Користувач має доступ до функцій перегляду архівних екологічних даних, отриманих із технологічних агрегатів підприємства. Реалізовано можливість фільтрації цих даних за датою, структурним підрозділом чи агрегатом, що забезпечує гнучкий підхід до аналізу інформації. Крім того, користувач може візуалізувати результати у вигляді графіків, що полегшує оцінку динаміки показників та виявлення відхилень.

Адміністратор, окрім можливостей, доступних звичайному користувачеві, має повний набір засобів для управління основними довідковими об'єктами системи. Зокрема, йому надається можливість створювати, переглядати, редагувати та видаляти структурні підрозділи, технологічні агрегати, а також параметри, що підлягають екологічному контролю.

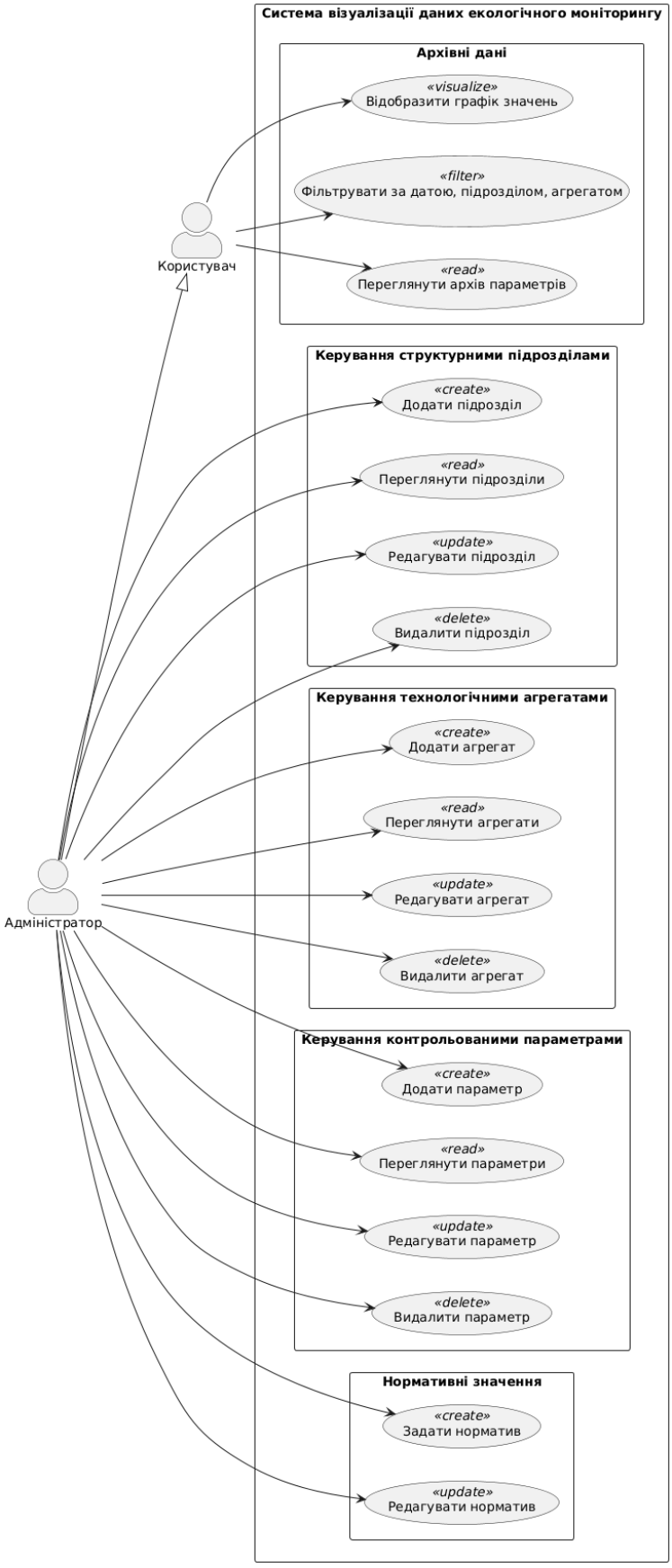


Рисунок 1.10 – Діаграма варіантів використання UML для веб-сервісу візуалізації даних екологічного моніторингу

Крім цього, адміністратор може встановлювати та змінювати нормативні значення для кожного з параметрів, що дозволяє здійснювати перевірку відповідності фактичних показників установленим межам допустимих викидів.

Усі варіанти використання супроводжуються відповідними стереотипами UML, які позначають тип операції: «create» – створення, «read» – перегляд, «update» – редагування, «delete» – видалення, а також «filter» та «visualize» – для дій фільтрації та візуалізації відповідно. Стереотипи оформлюються в подвійних кутових дужках (наприклад, <create>), що є частиною стандартної UML-нотації.

В цілому, побудована діаграма дозволяє комплексно описати функціональні можливості системи, формалізувати рольову модель доступу та забезпечити основу для подальшого проектування користувацького інтерфейсу, логіки доступу та програмної реалізації. Вона також є ефективним інструментом для узгодження вимог між розробниками, аналітиками та кінцевими користувачами.

Також доцільно додати ряд функціональних вимог, що не були відображені на діаграмі з точки зору лаконічності, але доцільні при реалізації веб-сервісу. Так, програмне забезпечення, що розробляється, повинно:

1. Забезпечити можливість перегляду:

– звіту щодо параметрів по належності до виробництва, структурного підрозділу;

– звіту щодо параметрів по належності до об’єкту спостереження, технологічного агрегату;

– можливість перегляду всіх параметрів, що обробляються в ПЗ.

2. Забезпечити можливість конфігурації/додавання нових або інших існуючих параметрів в систему.

3. Забезпечити інтерфейс вводу та зберігання інформації про відсутність кожного із параметрів, що на даний момент обробляються ПЗ.

4. Забезпечити автоматичне отримання даних про статуси працездатності

АСЕКМ та технологічного обладнання для демонстрації/передачі їх у бази даних та на зовнішньому сайті.

5. Забезпечити можливість конфігурації/додавання нових або інших існуючих параметрів до таблиць у форматі HTML.

6. Забезпечити автоматичне видалення старих даних в системі, також організувати можливість зміни налаштування періоду зберігання.

7. Забезпечити ведення журналів ПЗ, з нотуванням дій при роботі операторів/адміністраторів системи.

8. Забезпечити можливість коригування внесених нормативних значень згідно ДНВ або значень для порівняння у випадку зміни режимів роботи технологічних агрегатів, режимних карт для установок очищення газу після виконання ПНР, реконструкції, модернізації або змін нормативів, після отримання нових ДНВ.

Висновки до розділу:

У першому розділі роботи виконано аналіз існуючих систем та рішень, що дозволяють здійснювати візуалізацію даних екологічного моніторингу, зокрема для умов металургійного виробництва, визначено їх основні функціональні можливості, переваги і недоліки, що дозволило сформулювати функціональні на нефункціональні вимоги до веб-сервісу, що створюється. Відзначено важливість централізованого зберігання інформації, здатності інтегрувати дані з різнорідних джерел та забезпечувати подання результатів у зручному графічному форматі. На основі проведеного аналізу було формалізовано функціональні вимоги у вигляді діаграми варіантів використання UML, розроблено технічне завдання на створення системи.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ВІЗУАЛІЗАЦІЇ ДАНИХ ЕКОЛОГІЧНОГО МОНІТОРИНГУ

2.1 Обґрунтування технологій реалізації системи візуалізації даних екологічного моніторингу

У процесі розробки інформаційної системи візуалізації даних екологічного моніторингу для металургійного підприємства ключовим етапом є вибір відповідних технологій, що дозволяють забезпечити надійність, масштабованість, розширюваність та відповідність вимогам до обробки даних у реальному часі.

З урахуванням архітектурних та функціональних вимог проєкту було прийнято рішення реалізувати серверну частину системи з використанням платформи ASP.NET Core 8. Це сучасна кросплатформна веб-платформа з відкритим кодом, яка надає потужні засоби для створення високопродуктивних вебсервісів, REST API та фонового оброблення даних [26].

Серед переваг ASP.NET Core 8 слід виокремити:

– високу продуктивність і підтримку асинхронного програмування;

– розвинену інфраструктуру для DI-контейнерів, журналювання, конфігурації;

– підтримку фонового виконання служб (IHostedService);

– можливість розгортання на будь-якій операційній системі (Windows, Linux, macOS);

– активну спільноту та довгострокову підтримку від Microsoft.

Запропоноване рішення передбачає поділ системи на два основні логічні компоненти: адміністративний веб-інтерфейс і фоновий сервіс для обробки екологічних показників.

Адміністративна панель створюється з використанням архітектурного стилю MVC та синтаксису Razor і забезпечує інтерфейс для виконання CRUD-

операцій над доменними сутностями, пов’язаними із завданнями екологічного моніторингу в умовах промислового виробництва – структурними підрозділами, технологічними агрегатами та контрольованими параметрами. Цей компонент дозволяє адміністраторам системи конфігурувати логіку прив’язки параметрів до агрегатів, встановлювати нормативні значення та здійснювати базове адміністрування системи.

Другий компонент – сервіс фонової обробки даних – реалізується у вигляді класу, що наслідує інтерфейс IHostedService. Цей сервіс працює паралельно з веб-застосунком та відповідає за підключення до MQTT-брокера, отримання повідомлень із черг, парсинг даних та їх збереження до бази даних. Для обробки MQTT-повідомлень обрано бібліотеку MqttNet, яка забезпечує гнучкий та стабільний механізм публікації й підписки на теми, обробки повідомлень, підтримки QoS-рівнів та безпечного з’єднання.

IHostedService – це інтерфейс, що надається платформою ASP.NET Core і використовується для реалізації фонових служб , які виконуються паралельно з основним веб-додатком [19]. Служба починає виконання при запуску програми (StartAsync) і завершується при її зупинці (StopAsync).

В рамках проекту цей механізм використовується для створення довготривалого процесу, який:

– підключається до MQTT-брокера;

– очікує на вхідні повідомлення від сенсорів;

– використовується для парсингу вхідних даних (за потреби);

– зберігає їх у базі даних для подальшого аналізу та візуалізації.

Це дозволяє ізолювати логіку обробки телеметрії від веб-інтерфейсу, що покращує масштабованість та стабільність системи.

Використана у проєкті бібліотека MqttNet – це відкрита .NET-бібліотека для реалізації MQTT-клієнтів та серверів [14-16]. Вона підтримує всі основні функції протоколу MQTT, зокрема:

– підключення до брокера;

– підписка на одну або декілька тем (topics);

– обробка отриманих повідомлень через події (ApplicationMessageReceivedHandler);

– підтримку QoS (Quality of Service);

– шифрування з’єднання (TLS), аутентифікацію.

У проекті MqttNet використовується для створення MQTT-клієнта в межах IHostedService, який підписується на топіки типу asekm/monitoring/+та обробляє повідомлення, у тому числі і у форматі JSON. Отримані значення в залежності від теми записуються до бази даних з використанням EF Core.

Архітектура системи представлена на рис. 2.1. Ця діаграма ілюструє розділення системи на адміністративну частину та окремий фоновий сервіс, який підключається до MQTT-брокера, обробляє екологічні дані через бібліотеку MqttNet та зберігає результати у БД.

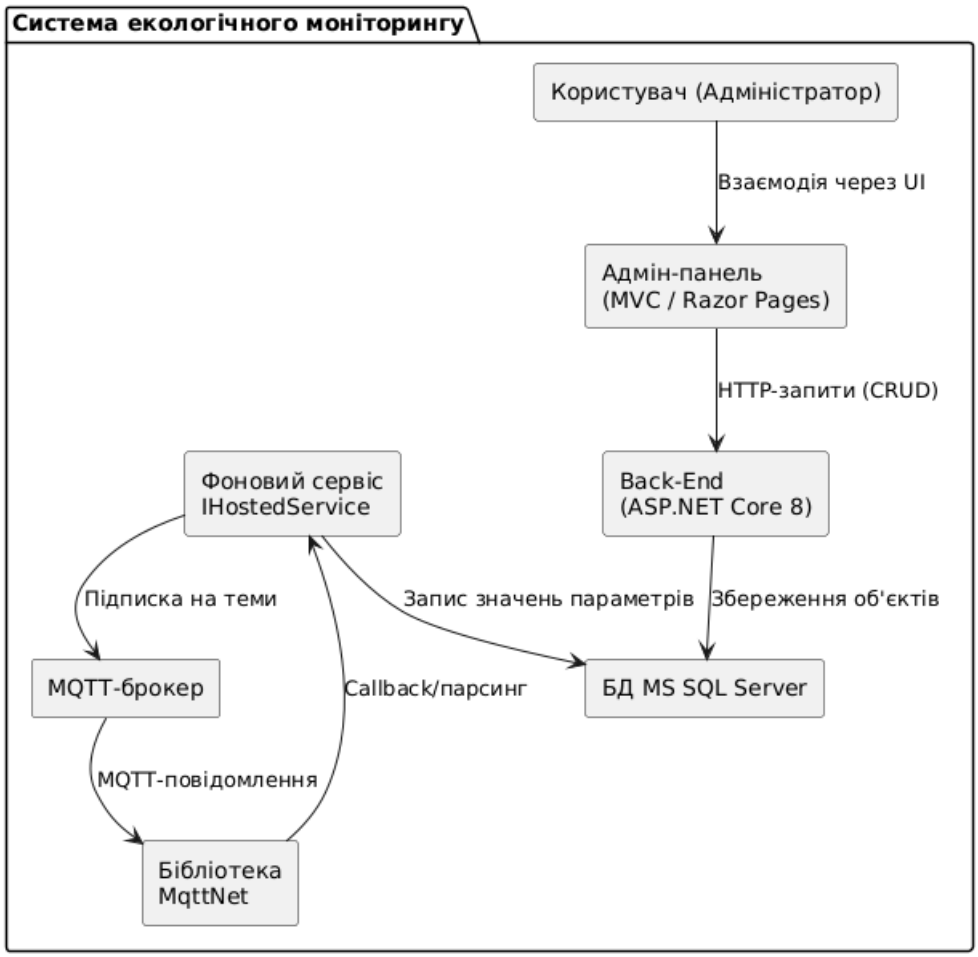


Рисунок 2.1 – Узагальнена архітектура системи візуалізації даних екологічного моніторингу

Поєднання платформи ASP.NET Core 8 із бібліотекою MqttNet дозволяє реалізувати стійку до збоїв, продуктивну й масштабовану систему, здатну обробляти як адміністративні дії користувачів, так і вхідні потоки екологічних даних у реальному часі [16, 28].

Для розробки системи візуалізації даних екологічного моніторингу було також вирішено використати ряд бібліотек, що поставляються у вигляді пакетів Nuget, зокрема:

1. AutoMapper – це бібліотека для автоматичного відображення (мапінгу) об’єктів одного типу на інший [17, 18]. У проекті вона використовується для перетворення сутностей бази даних у DTO-об’єкти, які передаються у вигляді моделей представлення (ViewModel) до користувацького інтерфейсу. Це значно спрощує обробку даних та зменшує кількість повторюваного коду.

2. Microsoft.AspNetCore.Components.QuickGrid.EntityFrameworkAdapter – ця бібліотека забезпечує адаптацію компонента QuickGrid для роботи з джерелами даних на базі Entity Framework Core. Вона дозволяє реалізувати ефективне відображення великих наборів даних з автоматичною підтримкою пагінації, сортування та фільтрації без необхідності створювати додаткову логіку вручну.

3. Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore – бібліотека використовується для діагностики помилок, що виникають під час роботи з базою даних через Entity Framework Core. Вона дозволяє автоматично виводити інформацію про виключення, пов’язані з міграціями або запитами до бази даних, у середовищі розробки.

4. Microsoft.AspNetCore.Identity.EntityFrameworkCore – цей пакет надає реалізацію системи аутентифікації та авторизації користувачів з використанням Entity Framework Core [27]. У проекті він використовується для управління обліковими записами адміністраторів, ролями та правами доступу до функціональних частин адміністративної панелі.

5. Microsoft.EntityFrameworkCore.Design – бібліотека потрібна для генерації вихідного коду моделей, контексту бази даних та міграцій у середовищі розробки. Вона дозволяє використовувати команди типу dotnet ef migrations для створення та оновлення структури бази даних.

6. Microsoft.EntityFrameworkCore.SqlServer – бібліотека-провайдер бази даних, який забезпечує роботу Entity Framework Core із СУБД Microsoft SQL Server. Він використовується у проекті для взаємодії з базою даних, що містить інформацію про підрозділи, агрегати, параметри, нормативи та екологічні показники.

7. Microsoft.EntityFrameworkCore.Tools – інструментальна бібліотека, яка дозволяє виконувати міграції, оновлення бази даних та тестування запитів з консолі або з середовища розробки Visual Studio. Вона є необхідною частиною інфраструктури розробника під час роботи з EF Core.

8. Microsoft.VisualStudio.Web.CodeGeneration.Design –цей пакет забезпечує можливість автоматичної генерації коду (scaffolding) для сторінок, контролерів та форм введення даних. У проекті він спрощує створення CRUD-інтерфейсів для сутностей, що зберігаються в базі даних, прискорюючи етап розробки інтерфейсної частини.

На рис. 2.2. зображені бібліотеки, що були використані у проєкті системи візуалізації даних екологічного моніторингу нугет-пакети.

Таким чином, обрані інструменти є оптимальними для реалізації завдань системи моніторингу, враховуючи потребу в обробці великого обсягу даних із датчиків, а також наявність адміністративного інтерфейсу з можливістю налаштування конфігурації об’єктів моніторингу. ASP.NET Core 8 забезпечує як надійну основу для веб-інтерфейсу, так і ефективну реалізацію сервісних механізмів для обробки екологічних показників, що надходять у режимі реального часу.

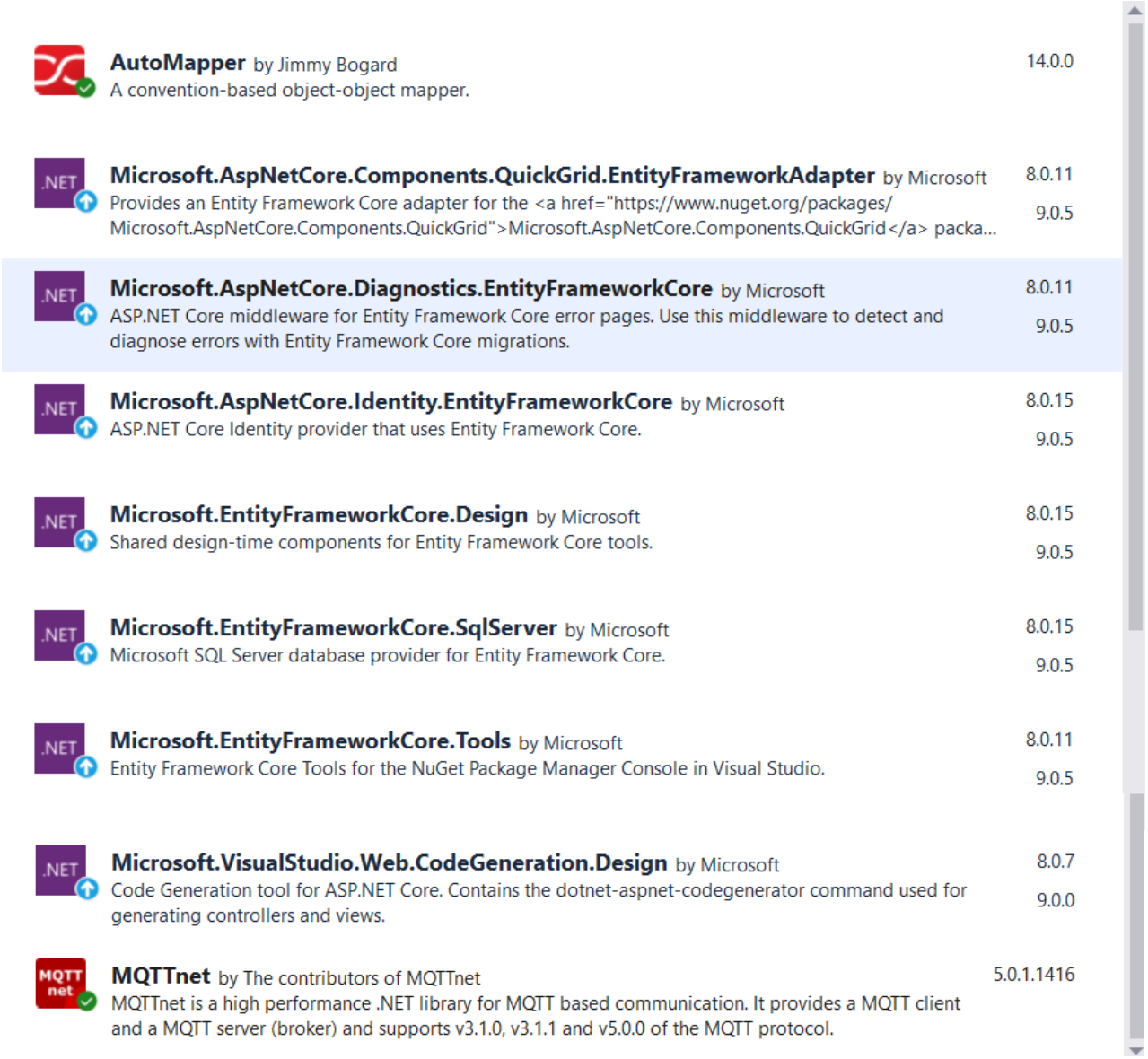


Рисунок 2.2 – Перелік бібліотек, використаних під час розробки інформаційної системи візуалізації даних екологічного моніторингу

2.2 Проєктування структури бази даних веб-сервісу для візуалізації даних екологічного моніторингу

Основними доменними сутностями веб-сервісу є структурні підрозділи, технологічні агрегати, параметри та дані вимірювань.

Для інформаційного моделювання предметної області, визначення основних сутностей та зв’язків між ними було використано інструменти уніфікованої мови моделювання UML. На рис. 2.3 наведено UML діаграму

класів [24], що описує основні доменні сутності предметної області та взаємозв’язки між ними. До структурного підрозділу може належати декілька технологічних агрегатів, тож між ними реалізовано зв’язок «один до багатьох». Так само технологічний агрегат характеризується зв’язком один до багатьох з екологічними показниками (параметрами), що вимірюються. У свою чергу, кожен показник може мати багато значень, виміряних в різні моменти часу (також зв’язок «один до багатьох»).

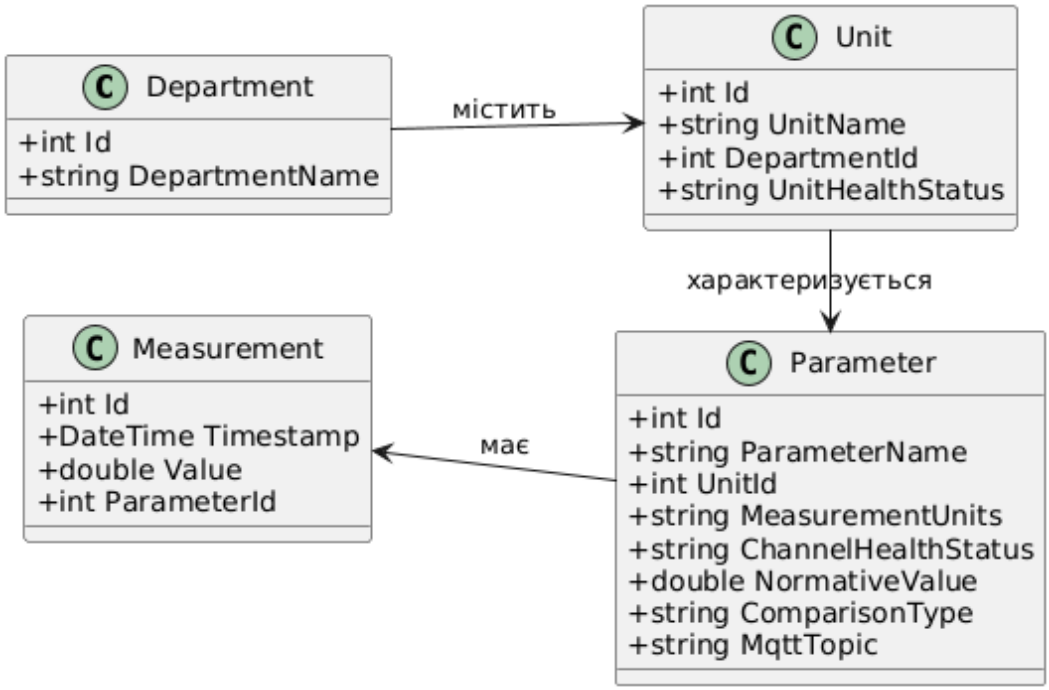


Рисунок 2.3 – Діаграма класів UML основних сутностей інформаційної системи аналізу результатів дослідницької діяльності НПП університету

Для збереження відповідної інформації у базі даних були розроблені таблиці Departments (структурні підрозділи), Units (агрегати), Parameters (вимірювані параметри) та Measurements (дані вимірювань) (рис. 2.3). Також на рис. 2.4 наведено таблицю EFMigrationsHistory, яка зберігає інформацію про історію міграцій і дозволяє керувати життєвим циклом БД з використанням інкрементального підходу [25].

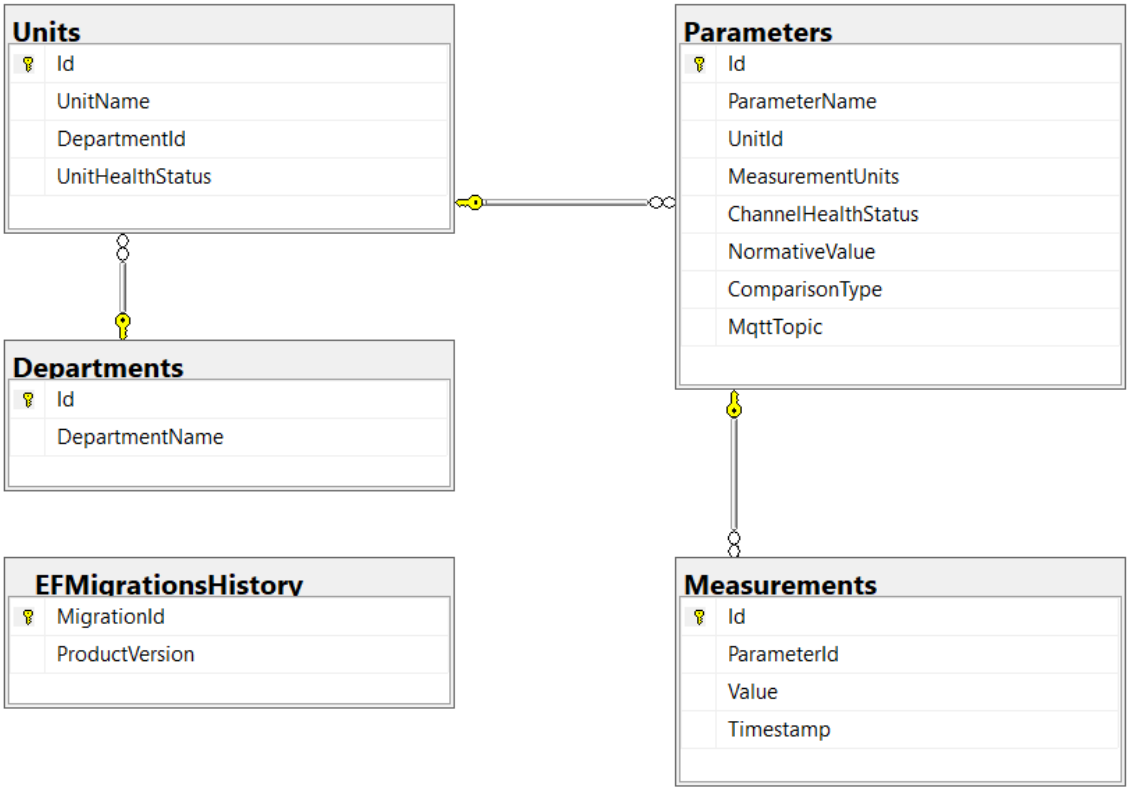


Рисунок 2.4 –Основні таблиці та зв’язки, що відповідають доменним сутностям проєкту системи візуалізації даних екологічного моніторингу

Основні характеристики таблиць БД, що відповідають суті предметної галузі, наведено у табл. 2.1.

Таблиця 2.1 – Структура таблиць бази даних «ScienceEvaluatorDb»

Назва таблиці	Назва поля	Тип даних	Атрибут
Departments	Id	int	Первинний ключ
	DepartmentName	nvarchar (max), not null	
Units	Id	int	Первинний ключ
	UnitName	nvarchar (max), not null	
	DepartmentId	int	Зовнішній ключ
	UnitHealthStatus	nvarchar (max), not null	

Продовження табл. 2.1

Назва таблиці	Назва поля	Тип даних	Атрибут
Parameters	Id	int	Первинний ключ
	ParameterName	nvarchar (max), not null	
	UnitId	int	Зовнішній ключ
	MeasurementUnits	nvarchar (max), not null	
	ChannelHealthStatus	nvarchar (max), not null	
	NormativeValue	float, not null	
	ComparisonType	nvarchar (max), not null	
	MqttTopic	nvarchar (max), not null	
Measurements	Id	int	Первинний ключ
	ParameterId	int	Зовнішній ключ
	Value	float, not null	
	Timestamp	datetime2(7), not null	

У проєкті використані наступні таблиці інфраструктури Microsoft Identity [27]:

1. AspNetUsers – центральна таблиця, яка містить облікові записи користувачів системи. Зберігає унікальний ідентифікатор користувача, ім’я входу (UserName), хеш пароля, електронну адресу, статус підтвердження акаунта, параметри безпеки (наприклад, використання двофакторної автентифікації), а також службову інформацію (дати створення, блокування тощо). У контексті проєкту таблиця містить записи як для звичайних користувачів, так і для адміністраторів системи.

2. AspNetRoles – таблиця ролей, в якій зберігаються всі типи доступу, які можуть бути присвоєні користувачам. Наприклад, у системі екологічного моніторингу може бути реалізовано розмежування між «Адміністратором»,

«Оператором», «Аналітиком» тощо. Це дозволяє реалізувати політику доступу на основі ролей (Role-Based Access Control, RBAC).

3. AspNetUserRoles – проміжна таблиця, яка забезпечує зв’язок багато до багатьох між користувачами та ролями. Вона фіксує, які ролі присвоєні кожному користувачеві, дозволяючи, наприклад, одному користувачеві мати одночасно права адміністратора та оператора.

4. AspNetUserClaims – містить додаткові атрибути (claims), які можна прив’язати до конкретного користувача. Claims використовують для гнучкого керування авторизацією – наприклад, для визначення доступу до перегляду даних конкретного підрозділу або типу екологічного параметра.

5. AspNetRoleClaims – аналогічно до AspNetUserClaims, ця таблиця зберігає claims, пов’язані з конкретною ролью. Це дозволяє задавати політику доступу не на рівні окремого користувача, а на рівні ролі, що спрощує адміністрування системи.

6. AspNetUserLogins – забезпечує підтримку зовнішніх провайдерів автентифікації, наприклад Google або Microsoft Account. У проєкті таблиця може використовуватися для подальшого розширення функціоналу, пов’язаного з інтеграцією з зовнішніми сервісами аутентифікації.

7. AspNetUserTokens – зберігає токені доступу, які можуть бути використані для реалізації двофакторної аутентифікації, функцій скидання пароля або підтвердження реєстрації.

Структура таблиць та зв’язків між ними, що реалізують інфраструктуру Microsoft Identity у розробленому проєкті системи візуалізації результатів екологічного моніторингу наведено на рис. 2.5.

У сукупності ці таблиці формують надійну модель управління доступом, яка забезпечує гнучке адміністрування прав користувачів, захист персональних даних та можливість масштабування в міру розвитку інформаційної системи. У контексті системи візуалізації екологічних показників це дозволяє реалізувати контрольовану і безпечну взаємодію з даними залежно від ролі та функціональних обов’язків користувача.

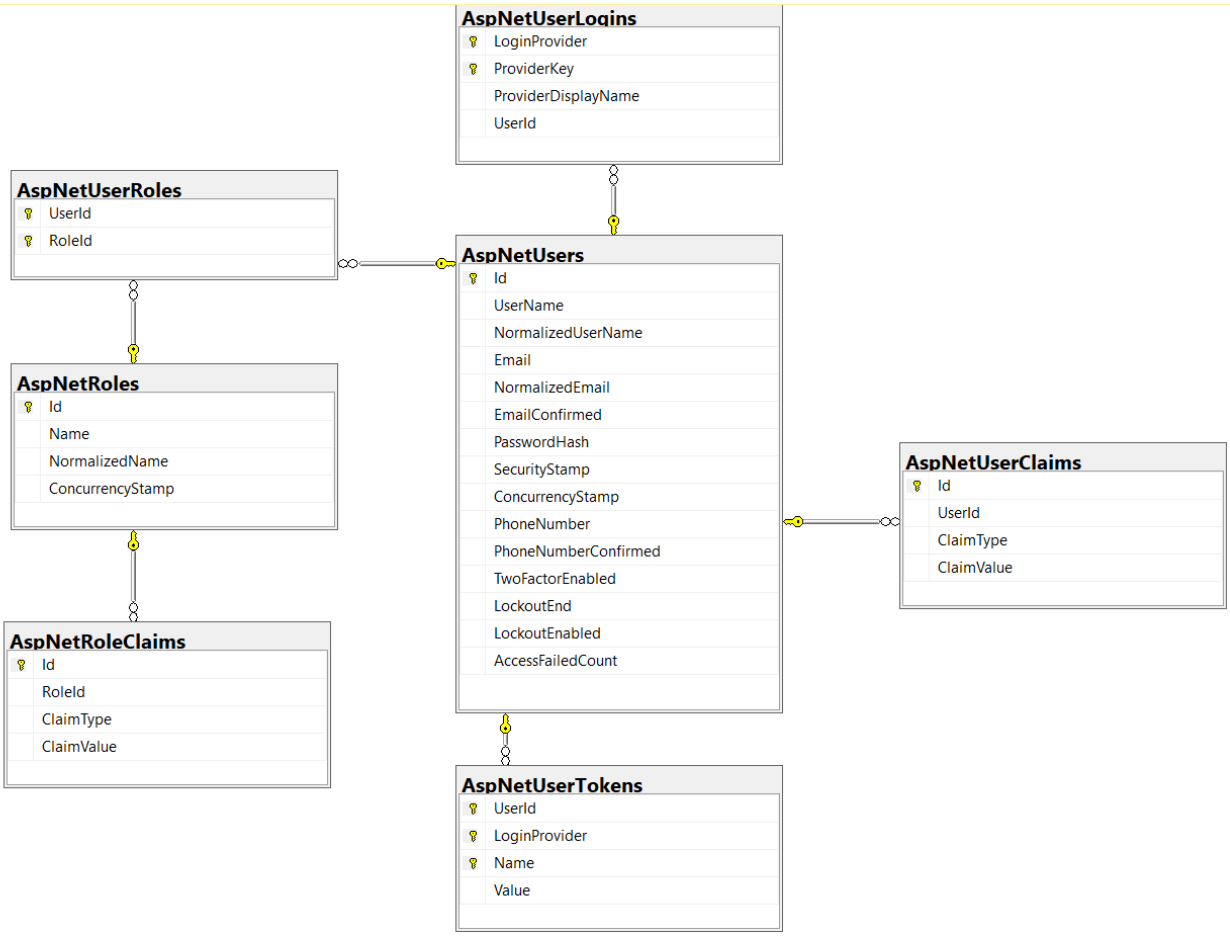


Рисунок 2.5 –Схема таблиць та зв’язків для реалізації аутентифікації та авторизації у системі візуалізації даних екологічного моніторингу

На рис. 2.6 наведено перелік таблиць веб-сервісу для візуалізації даних екологічного моніторингу після розгортання системи.

У проєктах ASP.NET Core для об’єктно-реляційного відображення між таблицями БД і основними класами використовується бібліотека EntityFramework Core. Для забезпечення зручного доступу до даних у таблицях і маніпулювання ними як елементами колекцій створюється так званий клас контексту даних, який містить набір властивостей типу DbSet<T>, де T – типи сутностей, яким відповідають таблиці бази даних [29]. Оскільки для аутентифікації та авторизації використовується бібліотека Microsoft.AspNetCore.Identity, то типовим рішенням є успадкування від класу IdentityDbContext<TUser>, де TUser – клас-нащадок класу TIdentityUser<TKey>.

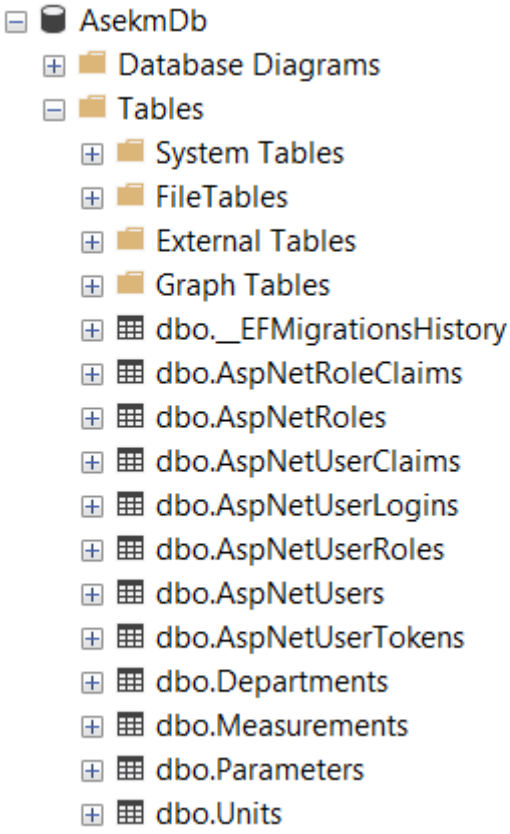


Рисунок 2.5 – База даних веб-сервісу візуалізації даних екологічного моніторингу

Для розроблюваного веб-сервісу візуалізації даних екологічного моніторингу клас контексту буде містити властивості-колекції типу DbSet для доступу до відомостей про структурні підрозділи (Departments), технологічні агрегати (Units), вимірювані параметри (Parameters) та виміряні значення (Measurements). На рис. 2.6 наведено код класу EcoDbContext. Як видно з коду, клас містить захищений (protected) метод OnConfiguring, який напряму визначає рядок підключення до БД. Цей метод включений тому, що клас контексту винесено у окрему бібліотеку, і на етапі запуску міграцій та початкового розгортання БД використовується саме ця конфігурація.

```
namespace AsekmDomainClassLibrary.Data
{
    18 references
    public class EcoDbContext : IdentityDbContext<IdentityUser>
    {
        0 references
        public EcoDbContext(DbContextOptions<EcoDbContext> options) : base(options) { }


        12 references
        public DbSet<Department> Departments { get; set; }

        12 references
        public DbSet<Unit> Units  { get; set; }

        13 references
        public DbSet<Parameter> Parameters { get; set; }

        9 references
        public DbSet<Measurement> Measurements { get; set; }

        0 references
        public EcoDbContext() { }
        //protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        //{
        //    string connStr = "Server=(localdb)\\MSSQLLocalDb;Database=AsekmDb;Trusted_Connectio
        //    optionsBuilder.UseSqlServer(connStr);
        //}
    }
}
```

Рисунок 2.6 – Код класу контексту БД для веб-сервісу візуалізації даних екологічного моніторингу

2.3 Проєктування каркасу веб-сервісу з використанням фреймворку ASP.NET Core

Структура проєкту системи візуалізації даних екологічного моніторингу побудована з урахуванням принципів модульності, розділення обов'язків та масштабованості. Система реалізована на основі платформи ASP.NET Core 8 та складається з двох основних проектів, кожен з яких виконує окрему роль у загальній архітектурі (рис. 2.7).

Перший проєкт – це бібліотека класів AsekmDomainClassLibrary, яка визначає доменну модель системи. Він містить такі основні компоненти, Department (структурні підрозділи підприємства), Unit (технологічні агрегати, що належать певному підрозділу), Parameter (екологічні показники, що контролюються на агрегатах), Measurement (конкретні значення параметрів).

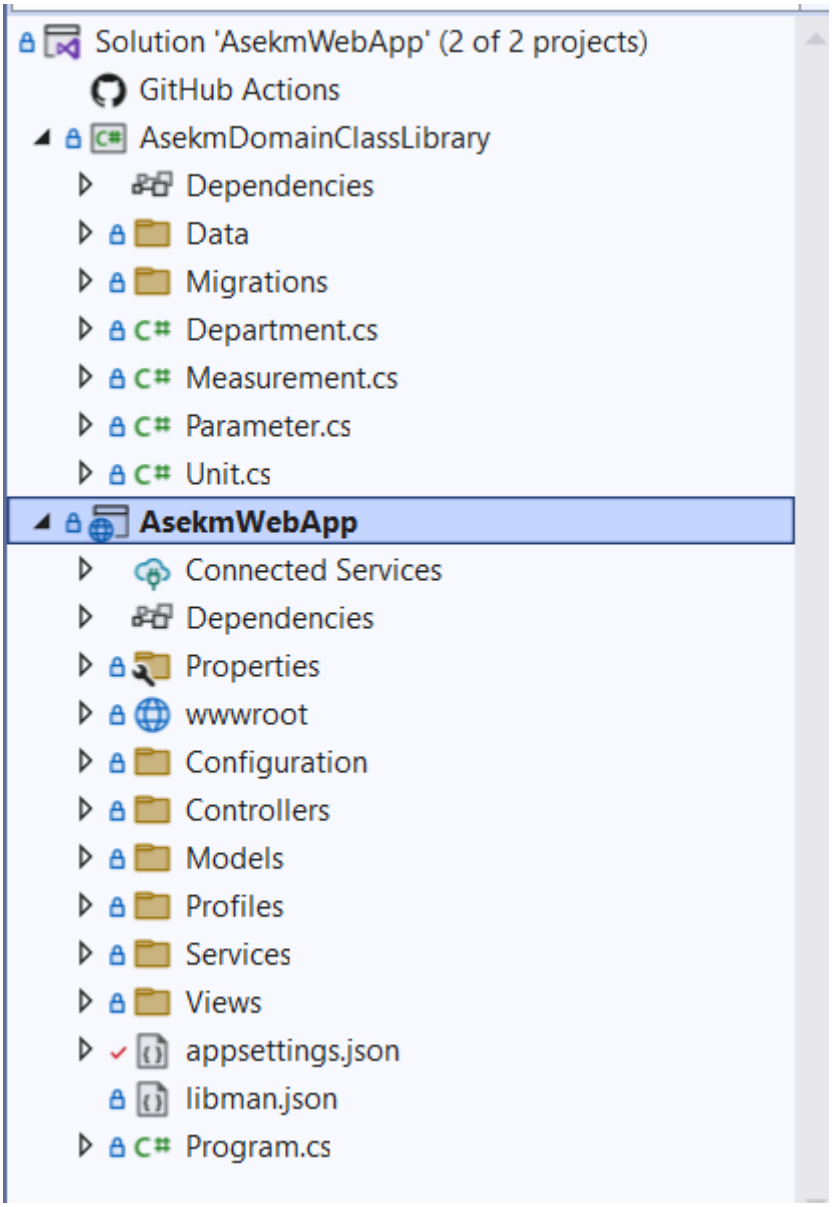


Рисунок 2.7 – Структура веб-сервісу візуалізації даних екологічного моніторингу

Папка Data містить клас EcoDbContext, що реалізує контекст даних для взаємодії з реляційною базою даних за допомогою Entity Framework Core. Контекст забезпечує конфігурацію зв’язків між сутностями відповідно до концепції «один до багатьох», що відповідає моделі предметної області.

Папка Migrations містить міграції, які дозволяють автоматично створювати чи оновлювати схему бази даних. Це спрощує розгортання проекту та адаптацію його до змін у моделях доменної області.

Інший проект є веб-застосунком на ASP.NET Core 8, що реалізує всі інтерфейсні та сервісні функції системи. Його структура включає наступні папки [26, 29]:

– wwwroot – містити статичні ресурси веб-застосунку, зокрема стилі, скрипти та зображення, які використовуються для візуалізації даних та підвищення зручності користувацького інтерфейсу.

– Configuration – містить класи для проєкції конфігурації системи в об’єкти під час впровадження залежностей. Це дозволяє централізовано налаштовувати ключові параметри взаємодії з базою даних, брокером MQTT та іншими сервісами.

– Controllers – містить контролери, які обробляють HTTP-запити користувачів та відповідають за обробку логіки веб-інтерфейсу. Контролери реалізують шаблон MVC (Model-View-Controller) та керують потоками даних між моделями та представленнями.

– Models – включає моделі представлення та класи DTO (Data Transfer Object). Моделі представлення відповідають за структуру даних, що відображаються у вебінтерфейсі, тоді як DTO спрощують обмін даними між рівнями застосування.

– Profiles – містить профілі бібліотеки AutoMapper. Ці профілі визначають правила картенгу між доменними моделями та DTO або ViewModel, що зменшує кількість шаблонного коду та забезпечує коректну трансформацію даних.

– Services – включає сервіси, що реалізують бізнес-логіку. Особливе місце тут займає реалізація інтерфейсу IHostedService, що відповідає за фоновий збір даних через бібліотеку MqttNet. Цей сервіс підключається до MQTT-брокера, отримує значення параметрів та записує їх до бази даних у реальному часі.

– Views – містить представлення Razor (Razor Views), які реалізують візуалізацію даних екологічного моніторингу у вигляді вебінтерфейсу. Вони забезпечують зручне відображення історичних даних, графіків та дозволяють адміністраторам виконувати CRUD-операції з об’єктами моніторингу.

Діаграма компонентів UML для додатку наведена на рис. 2.8.

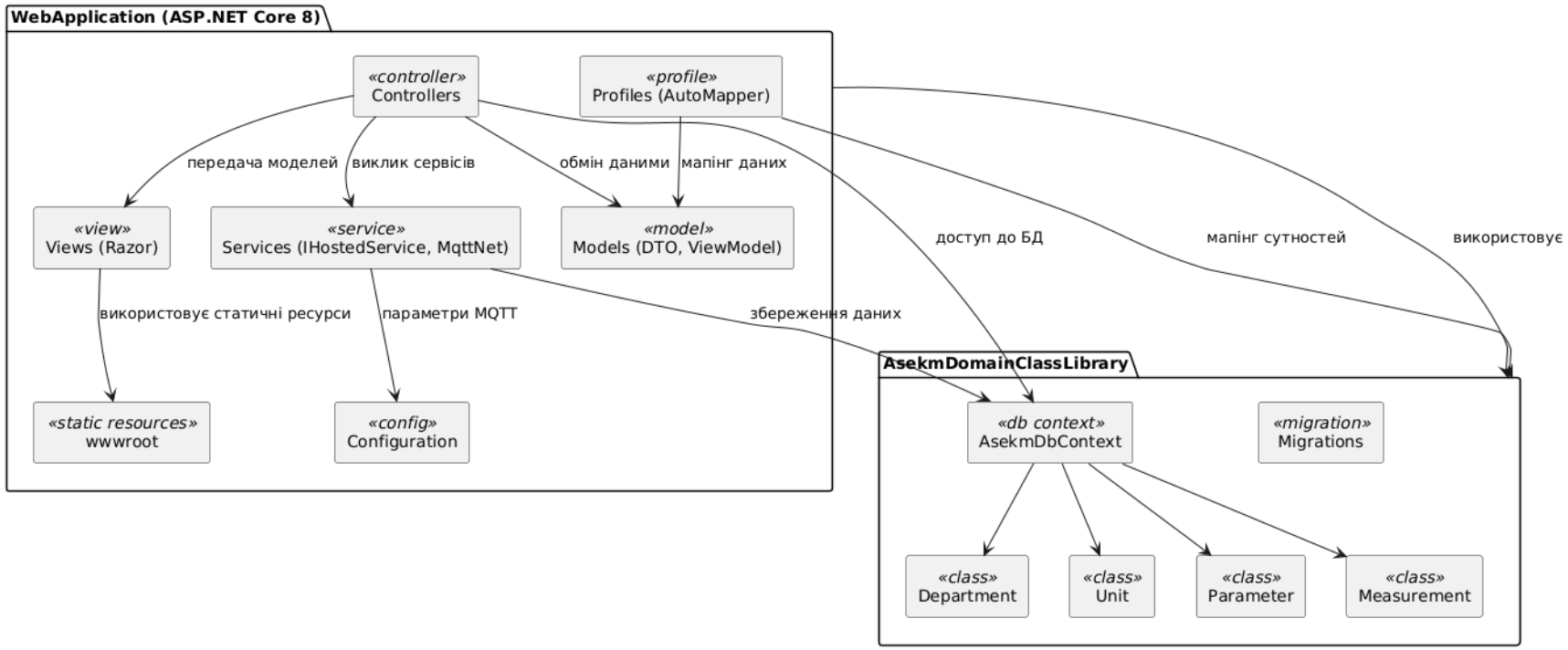


Рисунок 2.8 – Діаграма компонентів для веб-сервісу візуалізації даних екологічного моніторингу

2.4 Реалізація основного функціоналу веб-сервісу з використанням фреймворку ASP.NET Core

Оскільки реалізація функціоналу веб-сервісу візуалізації даних екологічного моніторингу здійснювалась на базі фреймворку ASP.NET Core з використанням архітектурного патерну MVC, у проєті було створено основні контролери, що реалізують керування життєвим циклом основних доменних сутностей предметної галузі, а також контролери, що відповідають за аутентифікацію та авторизацію користувачів.

У таблиці 2.2 наведено короткий опис розроблених контролерів та їх функціональне призначення у проєкті. Для дотримання вимог написання чистого коду та відповідності принципам проєктування класів SOLID під час написання коду контролерів використовувались принципи інверсії керування і додаткові сервіси впроваджувалися у код контролерів через конструктор.

Методи контролера, як правило, реалізовані у вигляді асинхронних із застосуванням конструкцій async/await, що дозволяє ефективно обробляти запити без блокування потоків та підвищує продуктивність веб-застосунку. Повернення результатів таких методів здійснюється через тип Task, параметризований класом, що реалізує інтерфейс IActionResult. Це є загальноприйнятим підходом у веб-застосунках на базі ASP.NET Core для забезпечення уніфікованого підходу до обробки результатів дій.

Для відображення даних у системі візуалізації екологічних показників використовуються спеціальні службові класи – так звані моделі представлення, що забезпечують передачу необхідних даних між контролером та поданням, а також спрощують валідацію введеної користувачем інформації.

Важливу роль у роботі контролерів відіграє система маршрутизації ASP.NET Core , яка відповідає за відображення HTTP- запитів на відповідні методи дії.

Таблиця 2.2 – Перелік контролерів проєкту інформаційної системи аналізу результатів дослідницької діяльності НПП університету

№ з/п	Назва контролеру	Призначення
1	AccountController	Відповідає за операції реєстрації, аутентифікацію та авторизацію користувачів
2	UserControllers	Відповідає за керування обліковими записами користувачів
3	RolesController	Відповідає за керування ролями користувачів
4	DepartmentsController	Відповідає за створення, перегляд, редагування та видалення інформації про структурні підрозділи
5	UnitsController	Відповідає за створення, перегляд, редагування та видалення інформації про технологічні агрегати
6	ParametersController	Відповідає за реалізацію CRUD-операцій над відомостями про вимірювані параметри екологічного моніторингу
7	MeasurementsController	Відповідає за реалізацію CRUD-операцій над результатами вимірювань параметрів екологічного моніторингу
8	HomeController	Відповідає за відображення сторінок візуалізації параметрів екологічного моніторингу

Таким чином, архітектура веб-застосунку базується на чіткому розмежуванні обов’язків між контролерами, моделями представлення та представленнями Razor, що разом забезпечує надійність, гнучкість та високу швидкодію системи.

Так, для прикладу, розглянемо процес додавання нового технологічного параметра в системі візуалізації даних екологічного моніторингу (рис. 2.9). У відповідності до принципів архітектурного шаблону MVC дану операцію реалізовано за допомогою двох методів контролера, що відповідають за обробку HTTP-запитів GET та POST.

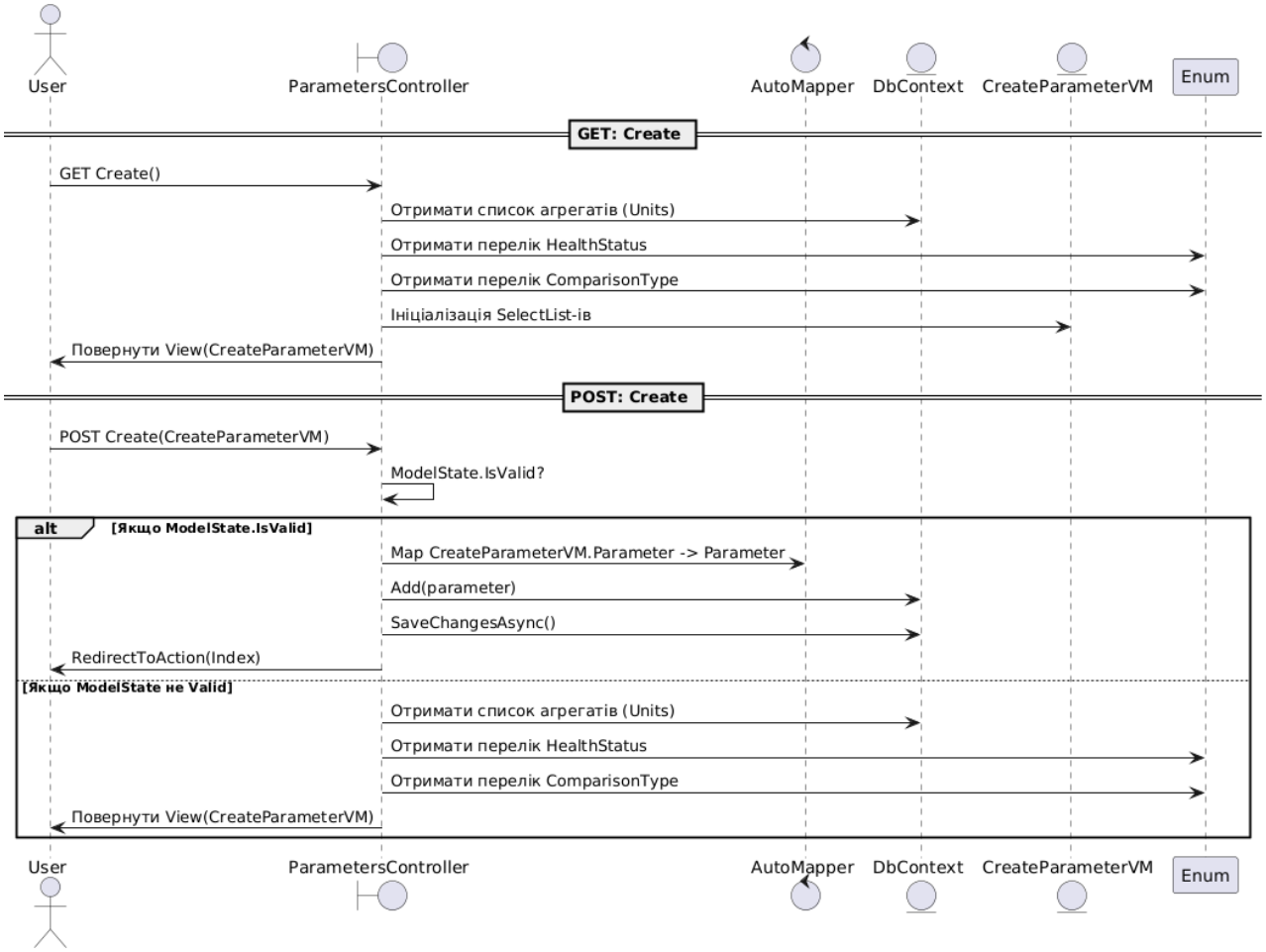


Рисунок 2.9 – UML- діаграма послідовності для методу Create контролеру ParametersController, що реалізують додавання технологічного параметра

У методі GET Create() реалізується ініціалізація моделі представлення CreateParameterVM, яка використовується для формування інтерфейсу додавання нового параметра у представленні Create. На цьому етапі здійснюється отримання необхідних даних з бази даних (списку доступних технологічних агрегатів) та переліку значень перерахувань (HealthStatus і ComparisonType) для формування випадаючих списків (SelectList) у веб-

інтерфейсі. Після цього контролер передає підготовлену модель представлення у відповідне подання (View), що дозволяє користувачеві ввести необхідні дані для створення нового технологічного параметра. Код реалізації методів Create контролера ParametersController наведено на рис. 2.10, код реалізації форми для додавання технологічного параметру наведено на рис. 2.11.

```
57      public IActionResult Create()
58      {
59          CreateParameterVM vM = new CreateParameterVM
60          {
61              Units = new SelectList(
62                  _context.Units, "Id", nameof(Unit.UnitName)),
63              HealthStatuses = new SelectList(
64                  Enum.GetNames(typeof(HealthStatus))),
65              ComparisonTypes = new SelectList(
66                  Enum.GetNames(typeof(ComparisonType)))
67          };
68          return View(vM);
69      }

71      // POST: Parameters/Create
72      
73      [HttpPost]
74      [ValidateAntiForgeryToken]
        0 references
75      public async Task<IActionResult> Create(CreateParameterVM vM)
76      {
77          if (ModelState.IsValid)
78          {
79              Parameter parameter = mapper.Map<Parameter>(vM.Parameter);
80              _context.Add(parameter);
81              await _context.SaveChangesAsync();
82              return RedirectToAction(nameof(Index));
83          }
84          vM.Units = new SelectList(
85              _context.Units, "Id",
86              nameof(Unit.UnitName),
87              vM.Parameter.UnitId);
88          vM.HealthStatuses = new SelectList(
89              Enum.GetNames(typeof(HealthStatus)),
90              vM.Parameter.ChannelHealthStatus);
91          vM.ComparisonTypes = new SelectList(
92              Enum.GetNames(typeof(ComparisonType)),
93              vM.Parameter.ComparisonType);
94          return View(vM);
95      }
```

Рисунок 2.10 – Лістинг методів Create(GET та POST версії) для додавання технологічного параметра

```
@model AsekmWebApp.Models.ViewModels.CreateParameterVM
@{
    ViewData["Title"] = "Create";
}
<h1>Додавання</h1>
<h4>Параметр</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="mb-3">
                <label asp-for="Parameter.ParameterName" class="control-label"></label>
                <input asp-for="Parameter.ParameterName" class="form-control" />
                <span asp-validation-for="Parameter.ParameterName" class="text-danger"></span>
            </div>
            <div class="mb-3">
                <label asp-for="Parameter.UnitId" class="control-label"></label>
                <select asp-for="Parameter.UnitId" class="form-select"
                asp-items="Model.Units"></select>
            </div>
            <div class="mb-3">
                <label asp-for="Parameter.MeasurementUnits" class="control-label"></label>
                <input asp-for="Parameter.MeasurementUnits" class="form-control" />
                <span asp-validation-for="Parameter.MeasurementUnits" class="text-danger"></span>
            </div>
            <div class="mb-3">
                <label asp-for="Parameter.ChannelHealthStatus" class="control-label"></label>
                <select asp-for="Parameter.ChannelHealthStatus" class="form-select"
                asp-items="Model.HealthStatuses"
                ></select>
                <span asp-validation-for="Parameter.ChannelHealthStatus" class="text-danger"></span>
            </div>
            <div class="mb-3">
                <label asp-for="Parameter.NormativeValue" class="control-label"></label>
                <input asp-for="Parameter.NormativeValue" class="form-control" />
                <span asp-validation-for="Parameter.NormativeValue" class="text-danger"></span>
            </div>
            <div class="mb-3">
                <label asp-for="Parameter.ComparisonType" class="control-label"></label>
                <select asp-for="Parameter.ComparisonType" class="form-select"
                 asp-items="Model.ComparisonTypes"></select>
                <span asp-validation-for="Parameter.ComparisonType" class="text-danger"></span>
            </div>

            <div class="mb-3">
                <label asp-for="Parameter.MqttTopic" class="control-label"></label>
                <input asp-for="Parameter.MqttTopic" class="form-control" />
                <span asp-validation-for="Parameter.MqttTopic" class="text-danger"></span>
            </div>
            <div class="mb-3">
                <input type="submit" value="Додати" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>
<div>
    <a asp-action="Index">Повернутися до списку</a>
</div>
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Рисунок 2.11 – Лістинг представлення Create.cshtml

Метод POST Create() обробляє запит типу POST, що надходить від користувача після заповнення форми. У разі, якщо стан моделі (ModelState) є валідним, за допомогою бібліотеки AutoMapper здійснюється мапінг даних із об’єкту передачі даних ParameterDTO у доменну сутність Parameter . Отриманий об’єкт зберігається у базі даних за допомогою Entity Framework Core , після чого виконується редірект користувача до головної сторінки контролера ParametersController, що відображає повний список параметрів.

У разі, якщо валідація моделі не пройдена, відбувається повторне формування випадаючих списків із бази даних та переліків перерахувань, що забезпечує коректне відображення помилок та дає змогу користувачеві виправити введені дані. Після цього контролер повертає оновлене представлення, яке містить дані, введені користувачем, а також повідомлення про помилки.

Таким чином, наведений опис процесу демонструє поетапне виконання операцій створення нового технологічного параметра з використанням сучасних технологій ASP.NET Core (DTO, моделі представлення, AutoMapper) та чітко розділеною відповідальністю між логікою контролера, обробкою даних та представленням у веб-інтерфейсі.

2.5 Реалізація читання виміряних значень екологічних показників з використанням протоколу MQTT

Як було вказано вище, для читання виміряних значень контрольованих екологічних показників до сервісів проєкту додано клас-реалізацію інтерфейсу IHostedService з назвою MqttClientHostedService. Екземпляр даного класу створюється при запуску проєкту ASP.NET Core на хості, після чого відбувається виклик методу StartAsync(), який запускає обробку подій, пов’язаних із отриманням повідомлень про зміну значень вимірюваних показників [19]. На рис. 2.12 наведено діаграму класів UML, що була розроблена для класу MqttClientHostedService

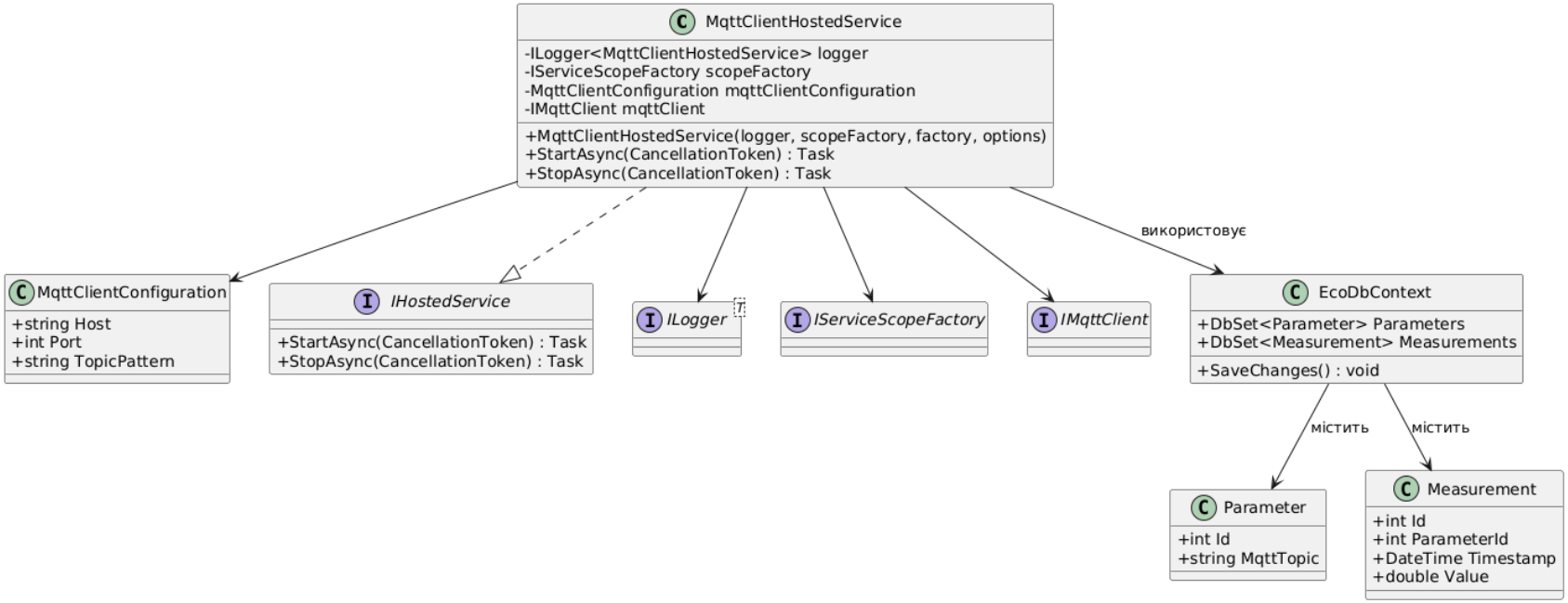


Рисунок 2.12 – Діаграма класів UML для сервісу MqttClientHostedService

Діаграма класів ілюструє основні компоненти, задіяні в реалізації сервісу MqttClientHostedService, який забезпечує обробку даних із MQTT-брокера. Вона відображає ключові залежності, такі як інтерфейси ILogger, IServiceScopeFactory, IMqttClientта IHostedService, а також класи EcoDbContext, Parameter та Measurement. Зв’язки між компонентами демонструють, як дані надходять від брокера MQTT, обробляються через створення scope та зберігаються у базі даних за допомогою контексту EF Core. Це підкреслює чітку архітектуру та модульність рішення.

Для відображення послідовності взаємодії допоміжних класів при реалізації методу StartAsync на рис. 2.13 наведено діаграму послідовності UML. Метод StartAsync у класі MqttClientHostedService реалізує ключову функціональність фонової служби, яка відповідає за зчитування екологічних даних із MQTT-брокера та збереження їх у базі даних. Цей процес починається з отримання налаштувань підключення (хост, порт, шаблон топіків) з конфігураційного об’єкта MqttClientConfiguration. Далі виконується підключення до брокера за допомогою клієнта IMqttClient та підписка на визначені теми. Це забезпечує постійний канал отримання даних у реальному часі.

Особливістю реалізації є обробка події отримання повідомлень (ApplicationMessageReceivedAsync) [14, 16]. Для повідомлення створюється scope через IServiceScopeFactory, що дозволяє отримати ізольований примірник EcoDbContext. Далі виконується пошук параметра (Parameter) у базі даних відповідно до теми повідомлення. Якщо такий параметр знайдено, створюється новий об’єкт результатів вимірювання (Measurement), який зберігає ідентифікатор параметра, значення та годину отримання. Після цього запис фіксується в базі даних, що гарантує збереження інформації про кожний екологічний показник.

Діаграма послідовності UML, створена для цього методу, детально ілюструє всі кроки взаємодії: від моменту ініціалізації з’єднання з брокером до обробки отриманих повідомлень.

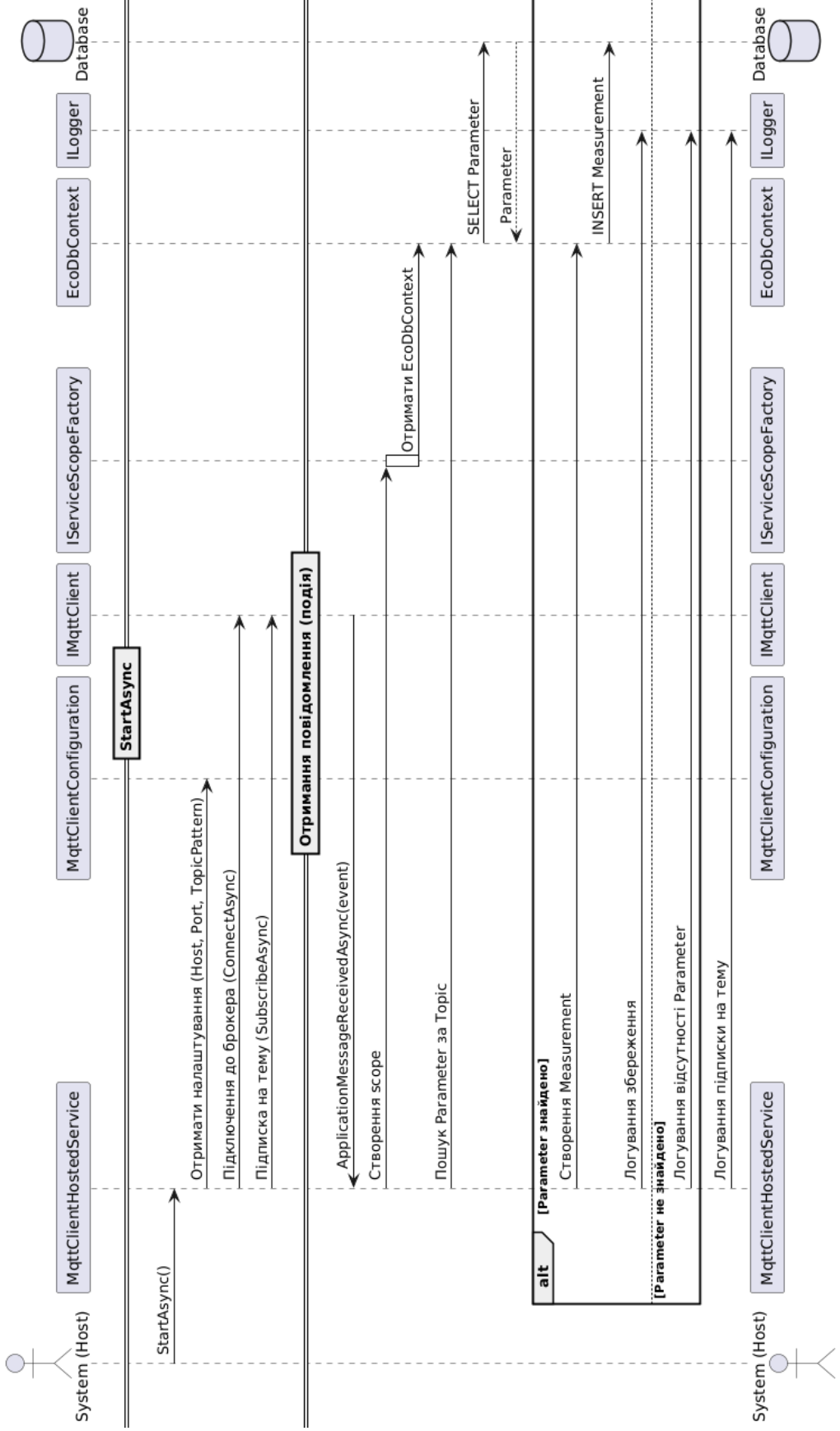


Рисунок 2.13 – Діаграма послідовності UML для сервісу MqttClientHostedService

Вона відображає роль кожного компоненту: ініціатора (системи), сервісу, клієнта MQTT, фабрики Scope та контексту БД. Це дозволяє візуально проаналізувати, як відбувається обмін повідомленнями, створення об’єктів та збереження даних у сховищі. Такий підхід до проектування не тільки підвищує прозорість та зрозумілість процесу, а й сприяє подальшому розширенню та модифікації системи екологічного моніторингу відповідно до вимог металургійного підприємства.

Код реалізації методу StartAsync сервісу MqttClientHostedService наведено на рис. 2.14.

```
public async Task StartAsync(CancellationToken cancellationToken)
{
    var mqttClientOptions = new MqttClientOptionsBuilder()
        .WithTcpServer(mqttClientConfiguration.Host,
        mqttClientConfiguration.Port)
        .Build();
    mqttClient.ApplicationMessageReceivedAsync +=  e =>
    {
        using var scope = scopeFactory.CreateScope();
        using EcoDbContext context = scope.ServiceProvider
        .GetRequiredService<EcoDbContext>();
        logger.LogInformation("New message received!");
        logger.LogInformation(e.ApplicationMessage.ConvertPayloadToString());
        logger.LogInformation(e.ApplicationMessage.Topic);
        Parameter? parameter = context.Parameters
        .FirstOrDefault(t => t.MqttTopic == e.ApplicationMessage.Topic);
        if (parameter is not null)
        {
            Measurement measurement = new Measurement
            {
                ParameterId = parameter.Id,
                Timestamp = DateTime.Now,
                Value = double.Parse(e.ApplicationMessage.ConvertPayloadToString())
            };
            context.Measurements.Add(measurement);
            context.SaveChanges();
        };
        return Task.CompletedTask;
    };
    await mqttClient.ConnectAsync(mqttClientOptions, cancellationToken);
    await mqttClient.SubscribeAsync(mqttClientConfiguration.TopicPattern,
        cancellationToken: cancellationToken);
    logger.LogInformation("MQTT client subscribed to topic.");
}
```

Рисунок 2.14 – Лістинг методу StartAsync сервісу MqttClientHostedService

2.6 Практична апробація та опис методики використання розробленого веб-сервісу

Для використання системи необхідно у адмін-панелі створити опис екологічних показників, інформація про поточні значення яких буде надходити від датчиків вузлів контролю. Оскільки вимірювані показники прив’язані до технологічних агрегатів, які, у свою чергу, прив’язані до структурних підрозділів підприємства, то на першому кроці потрібно створити відповідні структурні підрозділи.

На рис. 2.15 наведено сторінку, які відкривається при виборі пункту «Підрозділи» головного меню, а також доступну за адресою /departments. Дана сторінка відображає перелік структурних підрозділів підприємства. На даній сторінці також існує можливість перейти на сторінку додавання нового структурного підрозділу.

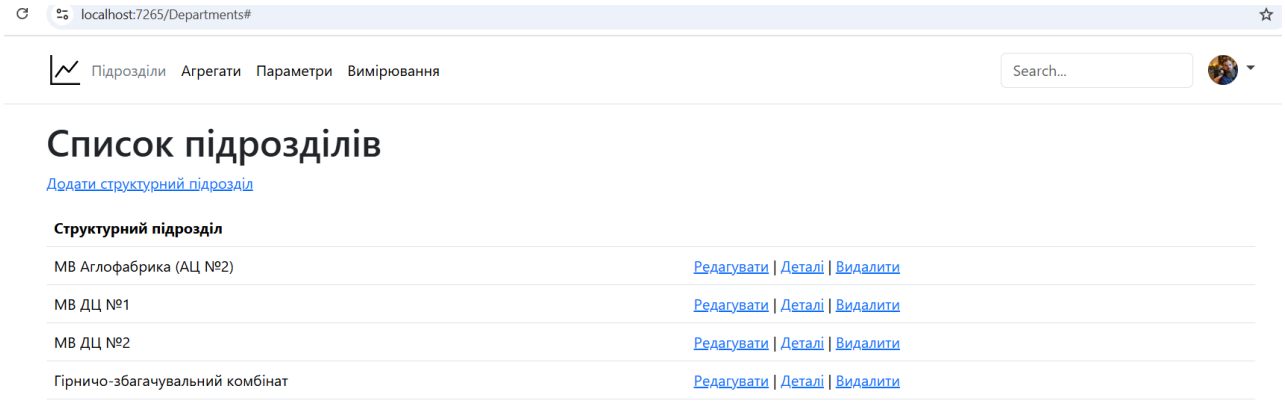


Рисунок 2.15 – Сторінка відображення списку структурних підрозділів підприємства

Для додавання нового структурного підрозділу необхідно ввести його назву та натиснути «Додати» (рис. 2.16), після чого відбудеться валідація даних, і якщо не виникне помилок валідації, інформація про новий структурний підрозділ буде додана, і виконано перенаправлення (редірект) на сторінку зі списком структурних підрозділів.

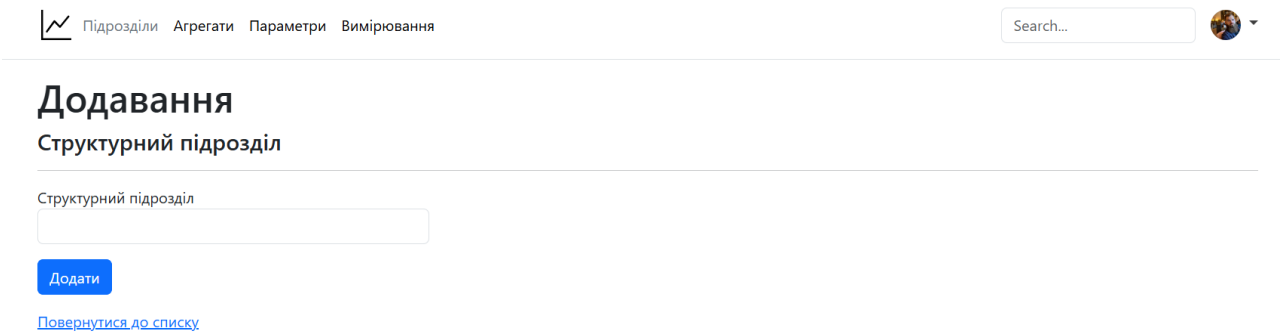


Рисунок 2.16 – Сторінка додавання структурного підрозділу підприємства

Також зі сторінки з відображенням списку підрозділів доступні інші CRUD-операції над підрозділами – редагування, перегляд деталей, видалення. Так, для прикладу, на рис. 2.17 наведено сторінку редагування структурного підрозділу. Аналогічний функціонал реалізовано для усіх доменних сутностей предметної області – технологічних агрегатів, вимірюваних параметрів, а також їх значень.

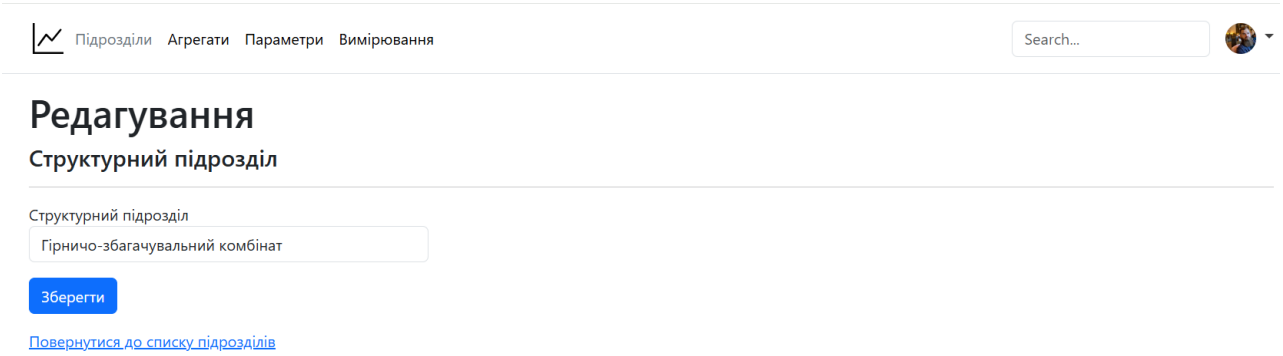


Рисунок 2.17 – Сторінка редагування структурного підрозділу підприємства

На наступному кроці необхідно додати технологічний агрегат-джерело викидів. Для цього необхідно обрати пункт «Агрегати» головного меню. Відкриється сторінка зі списком всіх раніше доданих технологічних агрегатів (рис. 2.18), на якій відображається основна інформація – назва агрегату, його приналежність до структурного підрозділу, статус працездатності, а також посилання на виконання CRUD-операцій над даною сутністю доменної області.

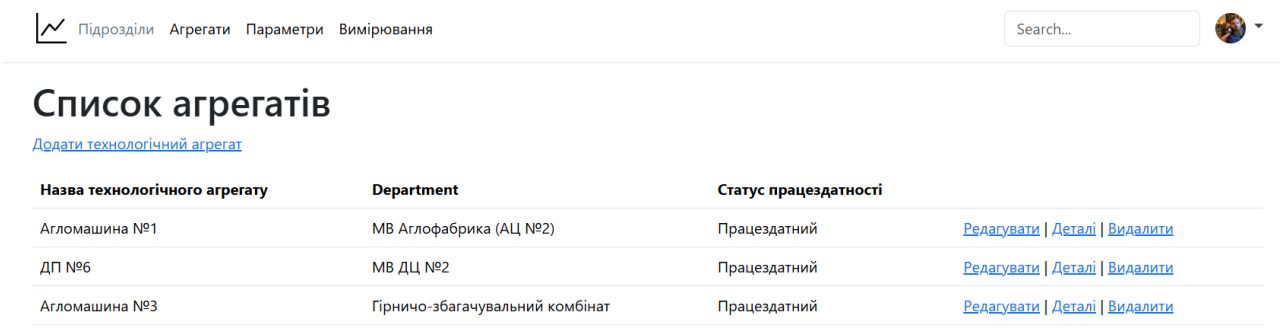


Рисунок 2.18 – Сторінка відображення списку агрегатів

Для додавання технологічного агрегату необхідно перейти за посиланням «Додати технологічний агрегат», після чого відкриється форма для додавання інформації про агрегат (рис. 2.19). На цій формі необхідно ввести назву технологічного агрегату, обрати приналежність до структурного підрозділу зі списку раніше доданих підрозділів, а також вказати статус працездатності.

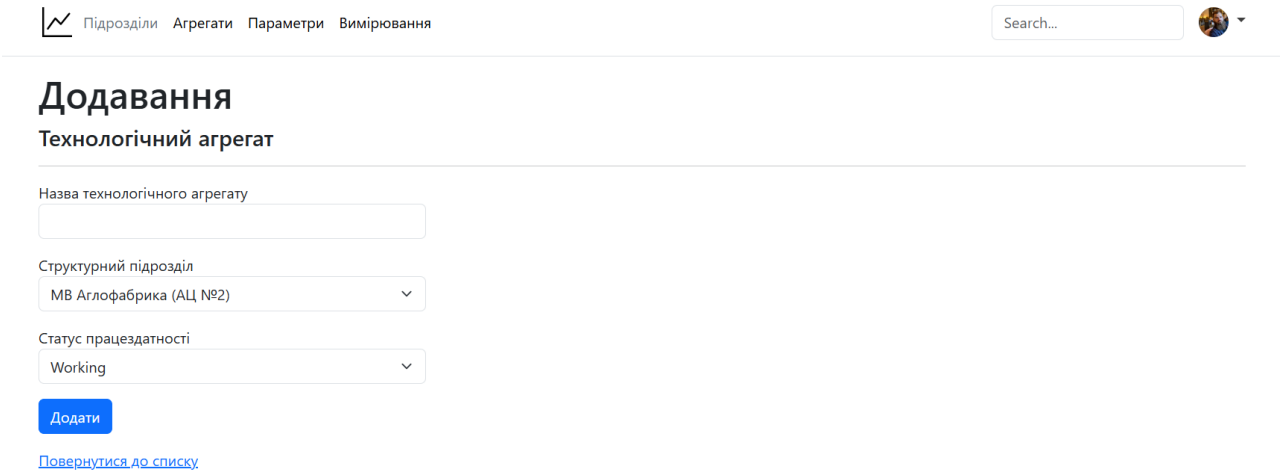


Рисунок 2.19 – Форма для додавання технологічного агрегату

Для редагування інформації про раніше доданий технологічний агрегат необхідно на сторінці з відображенням списку агрегатів перейти за посиланням «Редагувати». Форма для редагування відкривається із заповненою поточними значеннями полями (рис 2.20). Після оновлення інформації необхідно натиснути кнопку «Зберегти» для збереження внесених змін.

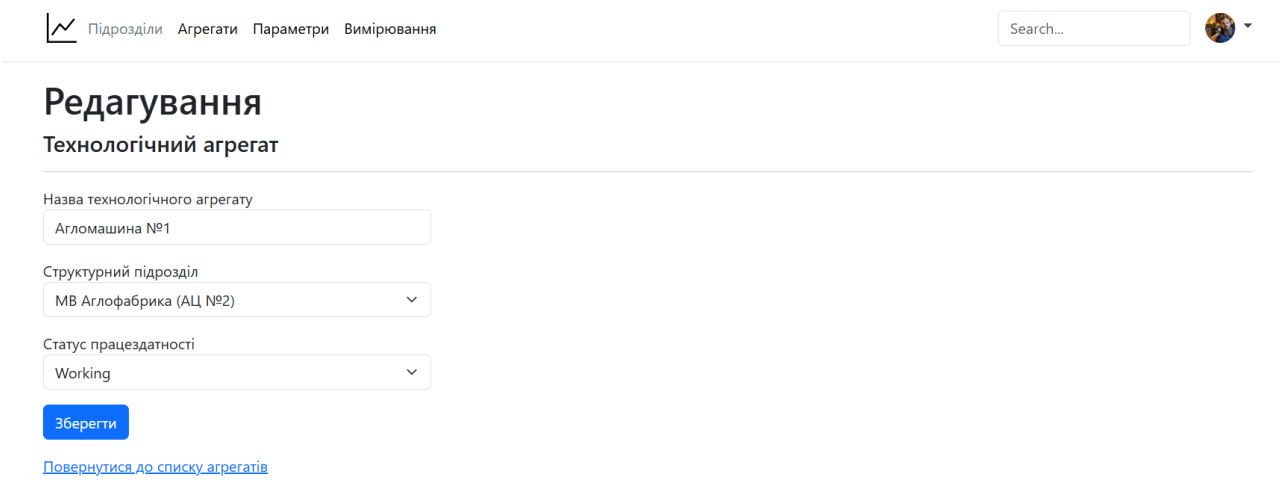


Рисунок 2.20 – Форма для редагування інформації про технологічний агрегат

При спробі видалення технологічного агрегату відкривається сторінка підтвердження, на якій можна або остаточно видалити інформацію, або повернутися на сторінку зі списком агрегатів (рис. 2.21).

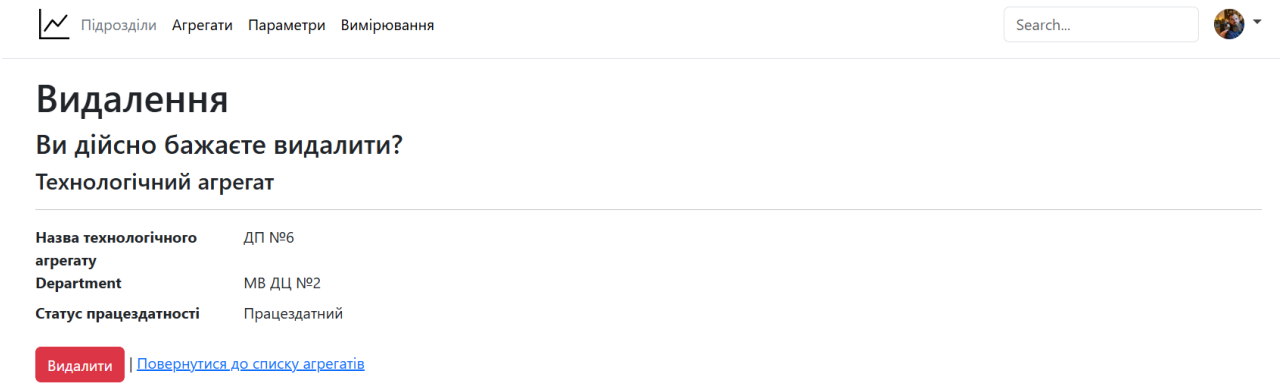


Рисунок 2.21 – Сторінка підтвердження видалення інформації про технологічний агрегат

Для додавання інформації про вимірювані показники екомоніторингу необхідно обрати пункт «Параметри» головного меню. На сторінці відображається таблиця з доданими раніше параметрами, а також посиланням для додавання нового параметра. По кожному параметру у таблиці відображається назва, технологічний апарат, до якого він належить, одиниці вимірювання, статус працездатності, нормативне значення, тип порівняння

(має бути менше, більше, дорівнювати), а також тема (топік) у брокері MQTT, на яку необхідно підписатися для читання значень (рис. 2.22).

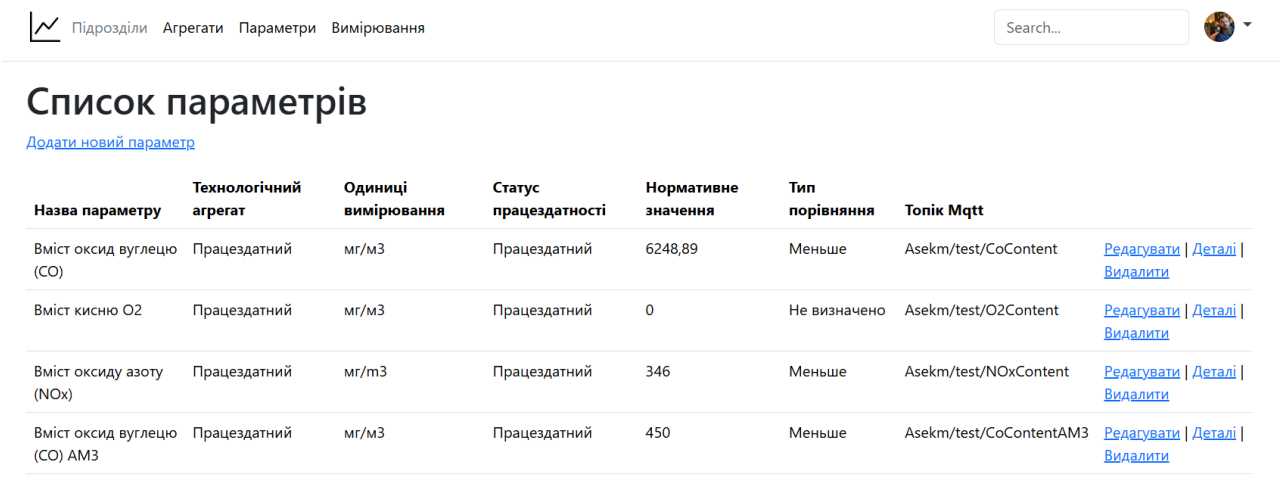


Рисунок 2.22 – Сторінка відображення списку контрольованих параметрів

Для додавання нового параметра необхідно перейти за посиланням «Додати новий параметр» на сторінці відображення списку параметрів, після чого буде здійснено перехід на форму додавання (рис. 2.23).

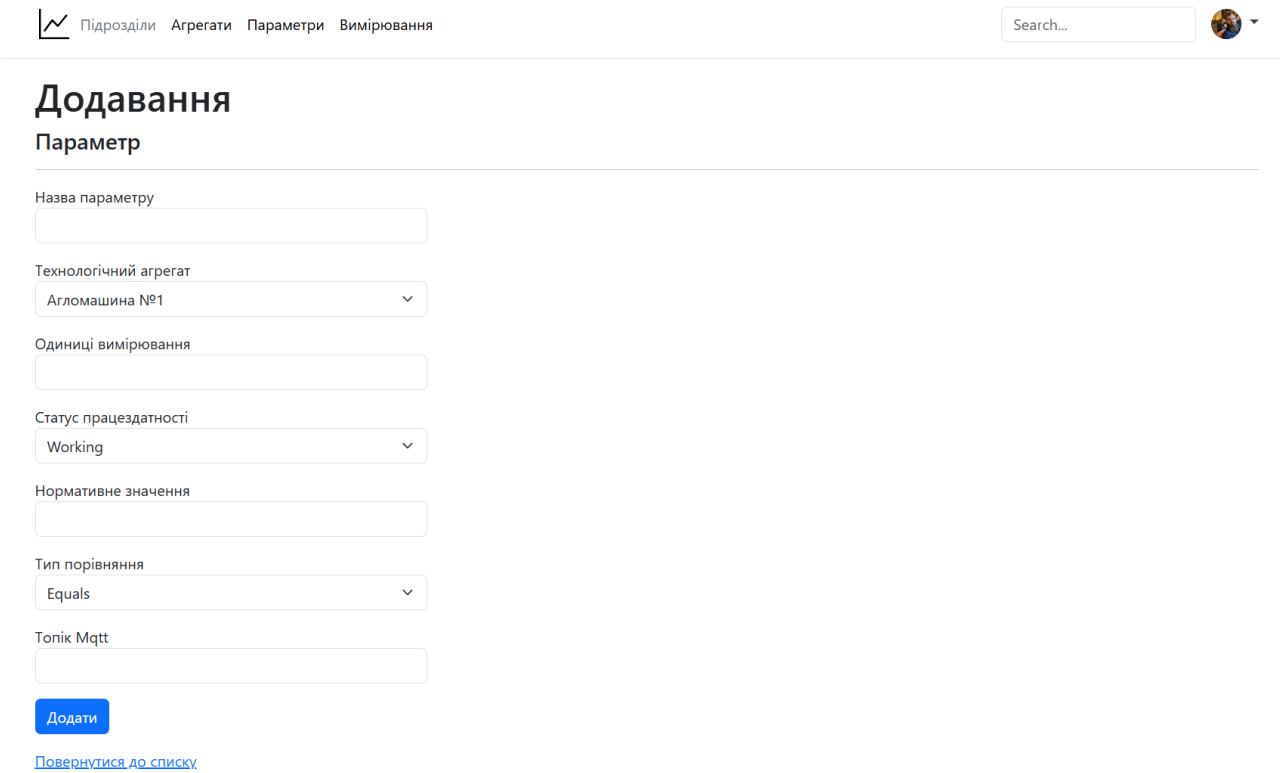


Рисунок 2.23 – Форма для додавання контрольованого параметру

Для додавання параметру необхідно вказати його назву, обрати зі списку технологічний агрегат, до якого належить цей контрольований параметр, вказати назву одиниць вимірювання, обрати зі списку тип порівняння та статус працездатності, ввести назву теми (топіку) на MQTT-брокері.

Також існує можливість редагування (рис. 2.24) інформації про контрольовані параметри.

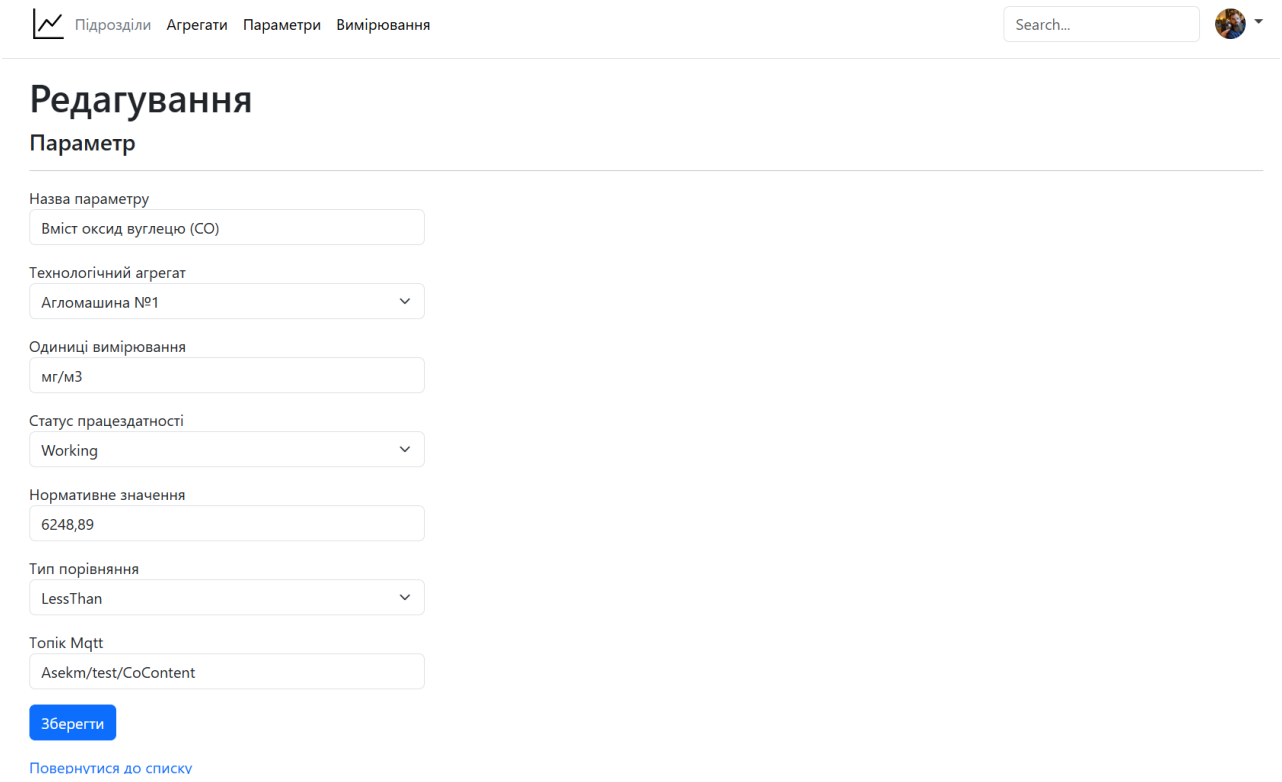


Рисунок 2.24– Форма для редагування інформації про контрольований параметр

При спробі видалення контрольованого параметру відкривається сторінка підтвердження, на якій можна або остаточно видалити інформацію, або повернутися на сторінку зі списком технологічних параметрів (рис. 2.25). Варто відзначити, що на етапі проєктування додатку було вирішено внести зміни у структуру БД і реалізувати «м’яке видалення», тож до таблиці Parameters з використанням механізму міграцій було додано властивість IsDeleted, яка встановлюється у значення true при підтвердженні видалення користувачем. Ця зміна викликана необхідністю зберігати історію змін значень

параметру при його видаленні.

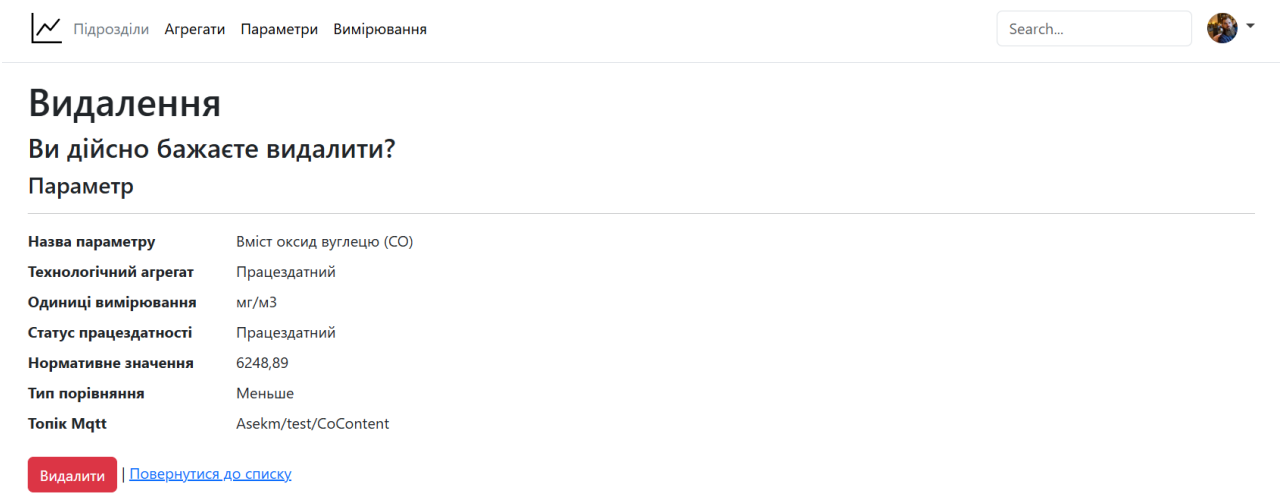


Рисунок 2.25 – Сторінка підтвердження видалення інформації про контрольований параметр

Для перегляду графіків змінення контрольованих параметрів необхідно перейти на головну сторінку веб-сервісу. На даній сторінці представлено три випадаючі списки, за допомогою яких необхідно спочатку обрати структурний підрозділ, потім один із технологічних агрегатів обраного підрозділу, і на останньому етапі – потрібний параметр (рис. 2.26).

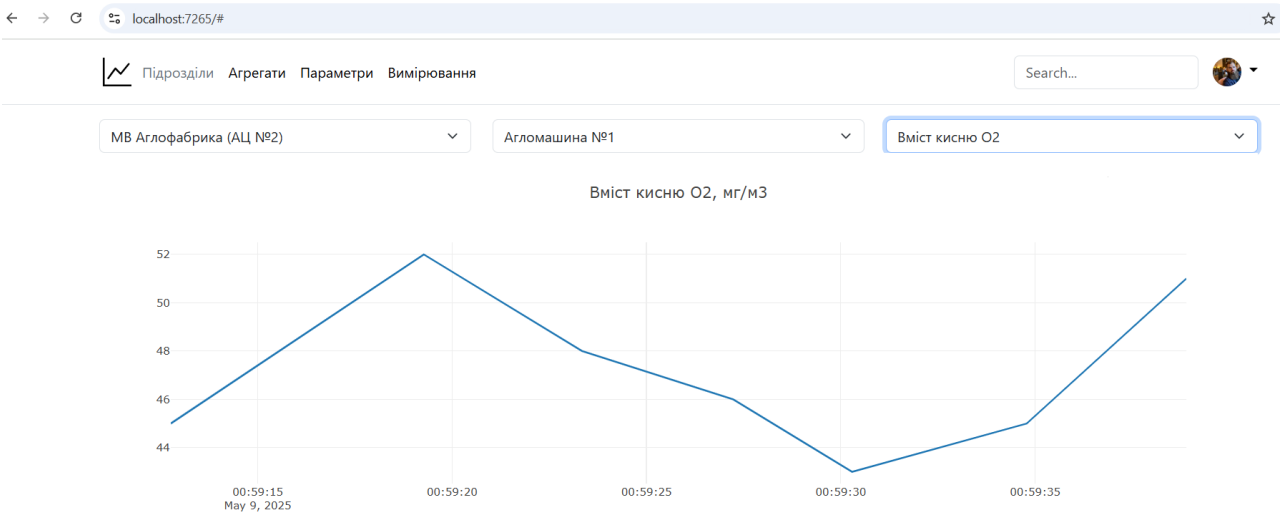


Рисунок 2.26 – Відображення графіку зміни контрольованого параметра – вмісту кисню біля агломашини №1

Дані у випадаючі списки підвантажуються динамічно при виборі однієї з опцій користувачем. Даний функціонал реалізовано завдяки використанню технології Ajax та наявності методів контролера Home, що повертають відповідні дані у форматі JSON. Так, на рис. 2.26 наведено результат вибору іншого контрольованого показника моніторингу.

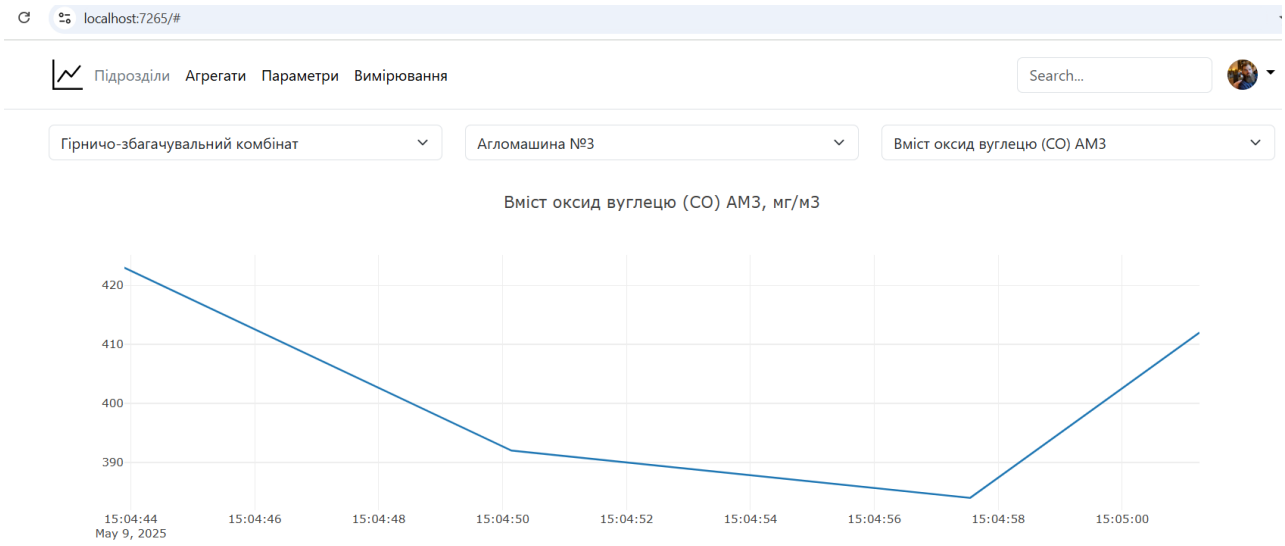


Рисунок 2.27 – Відображення графіку зміни контрольованого параметра – вмісту оксиду вуглецю біля агломашини №1

Отже, виконана апробація продемонструвала, що розроблений проєкт веб-сервісу для візуалізації даних екологічного моніторингу повністю відповідає встановленим вимогам, дозволяє керування основними сутностями предметної області та забезпечує отримання даних від MQTT-брокера, їх збереження у БД та відображення історії змін у вигляді графіків з прив’язкою до технологічних агрегатів.

Висновки до розділу:

У другому розділі виконано проєктування архітектури веб-сервісу для візуалізації показників екологічного моніторингу. Обгрунтовано доцільність реалізації додатку з використанням фреймворку ASP.NET Core 8, реалізацію

адмін-панелі для керування основними доменними сутностями на базі архітектурного шаблону MVC, отримання інформації про зміни значень контрольованих показників та запис її у БД – з використанням виділеної фонової задачі, що виконується хостом, а також реалізація відображення даних – з використанням WebAPI та технології AJAX.

Розроблено структуру бази даних для реалізації веб-сервісу, виконано проєктування відповідних класів, що забезпечило можливість використання ORM-фреймворку Entity Framework Core для об’єктно-реляційного співставлення екземплярів класів та таблиць при реалізації основних операцій по керуванню життєвим циклом інформації про основні сутності предметної області.

З використанням діаграм UML виконано опис розробленого додатку та особливостей його реалізації.

Виконано апробацію розробленої системи та наведено опис методики її використання. Проведені дослідження показали, що розроблена система виконує усі визначені функції та може бути рекомендована до подальшого розвитку та впровадження в умовах виробництва.

ВИСНОВКИ

У кваліфікаційній роботі була розроблена інформаційна система візуалізації даних екологічного моніторингу джерел викидів, що призначена для застосування в умовах металургійного підприємства. дозволяють здійснювати не лише контроль фактичного рівня забруднень, а й своєчасно приймати рішення для зменшення негативного впливу на довкілля.

Виконана робота охопила кілька ключових етапів. Спочатку було проведено огляд та аналіз існуючих систем екологічного моніторингу, зокрема таких рішень, як Enablon, Ecometrica, SaveEcoBot та EcoCity. Це дозволило визначити основні функціональні можливості, які мають бути реалізовані в сучасній інформаційній системі: збір даних від сенсорних пристроїв, централізоване зберігання, візуалізація в зручній для користувача формі, інтеграція з іншими джерелами даних та гнучке налаштування прав доступу.

На наступному кроці було здійснено проєктування та обґрунтування вибору інструментів реалізації системи. Основною платформою для розробки обрано ASP.NET Core 8 завдяки її відкритості, високій продуктивності та підтримці сучасних веб-технологій. Також було використано бібліотеку AutoMapper для спрощення перетворень між моделями представлення та доменними об’єктами, а для взаємодії з брокером MQTT – бібліотеку MqttNet, що забезпечує надійне з’єднання з сенсорними пристроями.

Архітектура системи передбачає розділення на дві основні складові: адміністративну панель для управління об’єктами моніторингу та фоновий сервіс, реалізований через інтерфейс IHostedService , який відповідає за обробку екологічних даних у реальному часі. Було розроблено структуру бази даних, що моделює підрозділи підприємства, технологічні агрегати, параметри та виміряні значення, що забезпечує цілісність та зручність обробки великих масивів даних.

У результаті розроблено прототип інформаційної системи, що дає змогу адміністраторам конфігурувати підрозділи, агрегати та контрольовані параметри, а також автоматично зберігати отримані дані в базі даних для

подальшої аналітики та прийняття управлінських рішень. Такий підхід дозволяє підвищити рівень екологічної безпеки на підприємстві, автоматизувати контроль за станом довкілля та забезпечити відповідність вимогам екологічного законодавства.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1 AQMIS – Lakes Software. Lakes Software – Leader in Environmental Software Solutions. URL: https://www.weblakes.com/software/emissions-management/aqmis/ (дата звернення: 31.05.2025).

2 Air Quality and Emissions Resources | California Air Resources Board. Homepage | California Air Resources Board. URL: https://ww2.arb.ca.gov/air-quality-and-emissions-resources (дата звернення: 31.05.2025).

3 Enablon, a Leader in EHS. Wolters Kluwer - Combining Domain Expertise With Advanced Technology | Wolters Kluwer. URL: https://www.wolterskluwer.com/en/solutions/enablon (дата звернення: 31.05.2025).

4 ESG Software | Sustainability Management Software | EcoOnline. EcoOnline. URL: https://www.ecoonline.com/esg-software/ (дата звернення: 31.05.2025).

5 Comprehensive compliance for the EU CBAM | Sphera. Sphera. URL: https://sphera.com/solutions/supply-chain-transparency/supply-chain-sustainability-solution/comprehensive-compliance-for-the-eu-cbam/ (дата звернення: 31.05.2025).

6 OGC SensorThings API Standard | OGC Publications. Open Geospatial Consortium. URL: https://www.ogc.org/standards/sensorthings/ (дата звернення: 31.05.2025).

7 OGC SensorThings API. OGC API. URL: https://ogcapi.ogc.org/sensorthings/ (дата звернення: 31.05.2025).

8 OGC SensorThings API Documentation | SensorUp OGC SensorThings API Developer Centre. Home | SensorUp OGC SensorThings API Developer Centre. URL: https://developers.sensorup.com/docs/#introduction (дата звернення: 31.05.2025).

9 Frequently Asked Questions. OGC SensorThings API. URL: https://ogc-iot.github.io/ogc-iot-api/faq.html (дата звернення: 31.05.2025).

10 BGS Sensor Data. British Geological Survey. URL: https://sensors-gui.bgs.ac.uk/ (дата звернення: 31.05.2025).

11 FROST®-Server - An open source implementation of OGC SensorThings API - Fraunhofer IOSB. Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB. URL: https://www.iosb.fraunhofer.de/en/projects-and-products/frost-server.html (дата звернення: 31.05.2025).

12 FROST Deployment Architecture. FROST-Server Documentation. URL: https://fraunhoferiosb.github.io/FROST-Server/deployment/architecture-packages.html (дата звернення: 31.05.2025).

13 GitHub - FraunhoferIOSB/FROST-Server: A Complete Server implementation of the OGC SensorThings API. GitHub. URL: https://github.com/FraunhoferIOSB/FROST-Server (дата звернення: 31.05.2025).

14 How to Use MQTT in C# with MQTTnet. DEV Community. URL: https://dev.to/emqx/how-to-use-mqtt-in-c-with-mqttnet-1j97 (дата звернення: 31.05.2025).

15 MQTTnet. .NET Foundation. URL: https://old.dotnetfoundation.org/projects/mqttnet (дата звернення: 31.05.2025).

16 Develop distributed application workloads with MQTTnet - Azure IoT Operations. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/azure/iot-operations/create-edge-apps/howto-develop-mqttnet-apps (дата звернення: 31.05.2025).

17 AutoMapper. AutoMapper. URL: https://automapper.org/ (дата звернення: 31.05.2025).

18 AutoMapper – AutoMapper documentation. URL: https://docs.automapper.org/en/stable/ (дата звернення: 31.05.2025).

19 Implement background tasks in microservices with IHostedService and the BackgroundService class. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/ru-ru/dotnet/architecture/microservices/multi-container-microservice-net-applications/background-tasks-with-ihostedservice (дата звернення: 31.05.2025).

20 Якість повітря у місті Кривий Ріг, Дніпропетровська область онлайн: карта якості атмосферного повітря України - SaveEcoBot. Єдина в Україні екологічна система - SaveEcoBot. URL: https://www.saveecobot.com/maps/kryvyi-rih (дата звернення: 31.05.2025).

21 Всеукраїнська мережа громадського моніторингу якості повітря. EcoCity: Reborn. URL: https://reborn.eco-city.org.ua/?station=1981 (дата звернення: 31.05.2025).

22 Екомоніторинг | АрселорМіттал Кривий Ріг. Головна | АрселорМіттал Кривий Ріг. URL: https://ukraine.arcelormittal.com/corporate-responsibility/ecology/ecomonitoring (дата звернення: 31.05.2025).

23 Training Session on data interoperability: FROST open source server solution for interoperable dynamic data. EDIAQI Project. URL: https://ediaqi.eu/resources/training-sessions/training-session-data-interoperability-frost-open-source-server (дата звернення: 02.06.2025).

24 Plant UML. url: https://plantuml.com// (дата звернення – 21.05.23).

25 Booth J. Database Design Succinctly. Morrisville, NC : Syncfusion Inc, 2022. URL: https://www.syncfusion.com/succinctly-free-ebooks/downloads/database-design-succinctly.pdf (дата звернення: 01.06.2025).

26 Freeman A. Pro ASP.NET Core 6. Berkeley, CA : Apress, 2022. URL: https://doi.org/10.1007/978-1-4842-7957-1 (дата звернення: 01.06.2025).

27 Introduction to Identity on ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity (дата звернення 10.05.2023).

28 Classon I. Migrating ASP.NET Microservices to ASP.NET Core 8. Berkeley, CA : Apress, 2024. URL: https://doi.org/10.1007/979-8-8688-1026-8 (дата звернення: 01.06.2025).

29 Giretti A. Coding Clean, Reliable, and Safe REST APIs with ASP.NET Core 8. Berkeley, CA : Apress, 2023. URL: https://doi.org/10.1007/978-1-4842-9979-1 (дата звернення: 01.06.2025).

30 Моркун Н. В., Завсегдашня І. В. Методичні вказівки до виконання

кваліфікаційної роботи бакалавру для студентів спеціальності 122 “Комп’ютерні науки”. Кривий Ріг : Видавничий центр КНУ, 2021. 50 с.

31 ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлення. Київ, ДП «УкрННЦ», 2015. 26с. (Інформація та документація).

32 ДСТУ 8302:2015. Бібліографічне посилання. Загальні вимоги та правила складання Київ, ДП «УкрННЦ», 2016. 16 с. (Інформація та документація).

33 ДСТУ 3582:2013. Бібліографічний опис. Скорочення слів і словосполучень в українській мові. Загальні вимоги та правила. Київ, ДП «УкрННЦ», 2013. 23 с. (Інформація та документація).