НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ Ігор БОЛБОТ	**_____ Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Програмне забезпечення вебсервісу для аналізу екологічних показників та візуалізації даних моніторингу**»**

Спеціальність «121 Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення»

Гарант освітньої програми
к.т.н., доцент _____ **Ганна ВАЙГАНГ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи
науковий ступінь та вчене звання _____ **Дмитро НІКОЛАЄНКО**
(підпис)

Виконав _____ **Віталій МАСЛИГАН**
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук
к.т.н., доцент _____Белла ГОЛУБ
(підпис)

«___» _____2025 р.

ЗАВДАННЯ

ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Маслигану Віталію Романовичу

(прізвище, ім'я, по батькові)

Спеціальність «121 Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення»
Тема бакалаврської кваліфікаційної роботи «Програмне забезпечення вебсервісу для аналізу екологічних показників та візуалізації даних моніторингу» затверджена наказом від «19» грудня 2025 р. № 3204«С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи:

Вимоги до розробки програмного забезпечення вебсервісу для аналізу екологічних показників та візуалізації даних моніторингу

Перелік питань, що підлягають дослідженню:

1. Системний аналіз предметної області для аналізу екологічних показників.

2. Аналіз існуючух рішень у сфері екологічного моніторингу

3. Результати тестування та аналіз екологічних показників

Перелік графічних документів (за необхідності).

Дата видачі завдання «19» грудня 2025 р.

Керівник бакалаврської кваліфікаційної робота _____ **Дмитро НІКОЛАЄНКО**
(підпис)

Завдання взяв до виконання _____ **Віталій МАСЛИГАН**
(підпис)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

API - прикладний програмний інтерфейс (Application Programming Interface)

БД - база даних

AQI - індекс якості повітря (Air Quality Index)

CSV - формат табличного обміну даними (Comma-Separated Values)

DTO - об’єкт передавання даних між програмними шарами (Data Transfer Object)

DWH - аналітичне сховище даних (Data Warehouse)

ER-діаграма - діаграма «сутність-зв’язок» (Entity-Relationship Diagram)

GeoJSON - формат подання просторових об’єктів у JSON-структурі

GIS - геоінформаційна система (Geographic Information System)

GUI - графічний інтерфейс користувача (Graphical User Interface)

HTTP - протокол передавання гіпертексту (HyperText Transfer Protocol)

HTTPS - захищений протокол обміну даними у вебсередовищі

IoT - інтернет речей, мережа сенсорних пристроїв (Internet of Things)

JSON - текстовий формат обміну структурованими даними (JavaScript Object Notation)

KPI - ключовий показник ефективності (Key Performance Indicator)

MQTT - легковаговий протокол обміну повідомленнями для IoT-пристроїв

PM2.5 - дрібнодисперсний пил із діаметром частинок до 2,5 мкм

PM10 - завислі частинки з діаметром до 10 мкм

PostGIS - розширення PostgreSQL для просторових даних

PostgreSQL - об’єктно-реляційна система керування базами даних

RBAC - керування доступом на основі ролей (Role-Based Access Control)

REST - архітектурний стиль побудови веб API

SQL - мова структурованих запитів до баз даних (Structured Query Language)

TLS - криптографічний протокол захисту мережевого обміну

UML - уніфікована мова моделювання (Unified Modeling Language)

UUID - універсальний унікальний ідентифікатор (Universally Unique Identifier)

ІС - інформаційна система

ПЗ - програмне забезпечення

ВСТУП

Актуальність роботи зумовлена тим, що екологічні дані надходять із різнорідних джерел: автоматизованих постів спостереження, лабораторних вимірювань, відкритих API та геоінформаційних сервісів. Без єдиного вебсервісу такі відомості складно зіставляти, перевіряти та використовувати для оперативного аналізу стану довкілля.

Розроблення вебсервісу для аналізу екологічних показників має практичне значення, оскільки дає змогу централізувати збір вимірювань, контролювати перевищення нормативів, будувати інтерактивні карти та формувати звіти для фахівців, органів управління й користувачів, зацікавлених у якості довкілля.

Метою дослідження є проєктування та розроблення програмного забезпечення вебсервісу, який забезпечує приймання, збереження, оброблення, аналіз і візуалізацію екологічних показників у зручному вебінтерфейсі.

Для досягнення поставленої мети сформульовано такі основні **завдання:**

– проаналізувати предметну область екологічного моніторингу та визначити вимоги до вебсервісу;

– обґрунтувати архітектуру програмного забезпечення з урахуванням потокового надходження вимірювань;

– спроєктувати логічну та фізичну модель даних для збереження показників довкілля;

– реалізувати модулі приймання даних від сенсорів, лабораторних джерел та зовнішніх API;

– розробити механізми очищення, агрегації, порівняння з нормативами та візуалізації показників;

– забезпечити просторове відображення результатів моніторингу на інтерактивній карті;

– реалізувати підсистему звітності, сповіщень і журналювання критичних перевищень;

– провести тестування продуктивності, коректності оброблення даних та стабільності роботи вебсервісу;

– оцінити придатність розробленого рішення для підтримки екологічного аналізу та прийняття управлінських рішень.

Об’єктом дослідження є процес інформаційної підтримки екологічного моніторингу територій із використанням вебтехнологій, баз даних та засобів просторової візуалізації.

Предметом дослідження є моделі даних, алгоритми оброблення та програмні компоненти вебсервісу, призначеного для аналізу показників повітря, води, ґрунту та метеорологічних параметрів.

Методологічною основою роботи є системний аналіз, об’єктно-орієнтоване моделювання, ER-моделювання, принципи REST API, методи попередньої обробки часових рядів, просторового аналізу та візуального подання даних.

Практична цінність роботи полягає у формуванні програмної основи, яку можна використати для створення інформаційно-аналітичного вебсервісу екологічного моніторингу з підтримкою карт, графіків, фільтрів, журналу подій і експорту звітів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Екологічний моніторинг охоплює систематичне спостереження за станом повітря, водних ресурсів, ґрунтів і супутніх метеорологічних умов. У межах цифрової системи такі спостереження подаються як часові ряди вимірювань, пов’язані з пунктами контролю, сенсорами, параметрами та географічними координатами (представлено на рис.1.1).

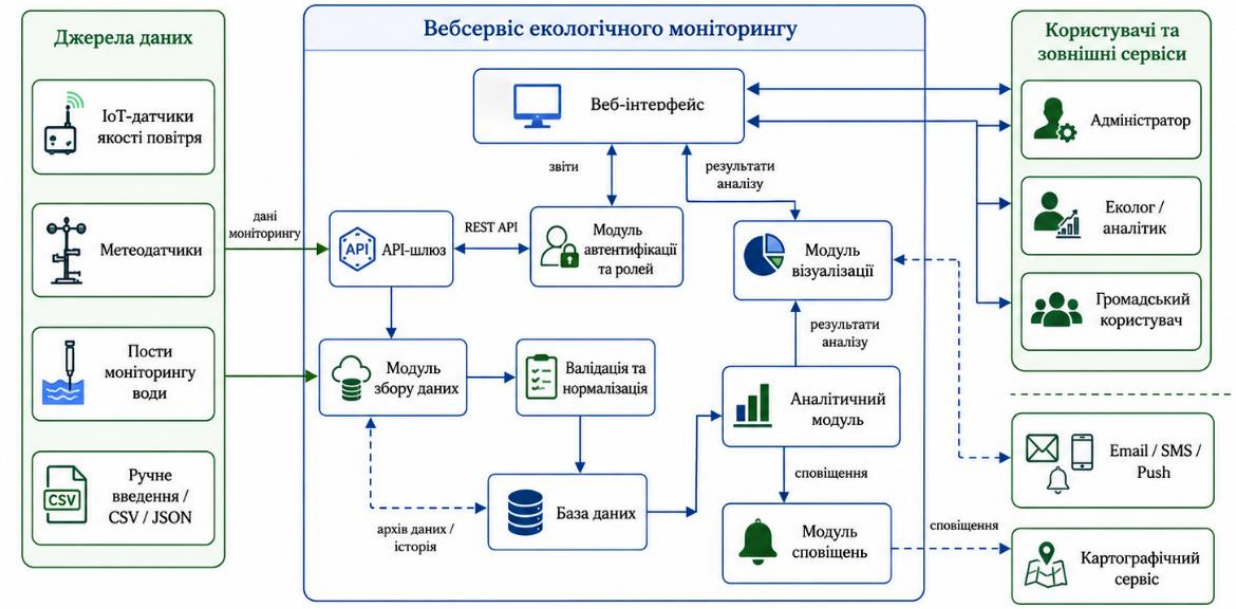

Рис. 1.1. Загальна архітектура потоків даних та компонентів вебсервісу екологічного моніторингу

Джерелами вхідних даних є стаціонарні пости, мобільні сенсорні вузли, лабораторні протоколи, відкриті набори даних, API зовнішніх платформ та файли імпорту у форматах CSV або JSON. Кожне джерело має власну частоту оновлення, формат і набір контрольованих показників.

У підсистемі приймання даних виконується перевірка структури повідомлень, узгодження одиниць вимірювання, прив’язка до пункту моніторингу та нормалізація часових міток. Це необхідно для коректного порівняння показників між різними джерелами та періодами.

Подальший аналіз передбачає агрегацію вимірювань за годинами, добами або вибраними періодами, розрахунок середніх і максимальних значень, виявлення перевищень порогів та підготовку даних для графіків, карт і звітів.

Важливою складовою предметної області є блок правил, який порівнює поточні значення з допустимими межами. У разі перевищення концентрації забруднювача або різкої зміни показника система створює подію та передає її до журналу сповіщень.

Довідниковий рівень системи зберігає інформацію про параметри моніторингу, типи сенсорів, одиниці вимірювання, територіальні зони, користувачів, ролі й нормативні межі. Узгодження цих даних забезпечує цілісність аналітичної моделі.

Для такого вебсервісу важливими є масштабованість, надійність збереження часових рядів, захищений доступ до API, швидке оновлення інформаційних панелей і можливість просторового аналізу даних у розрізі територій.

Предметна область поєднує задачі збору даних, їх очищення, збереження, аналітичної обробки та візуалізації. Саме тому програмне забезпечення має бути побудоване як модульний вебсервіс із чітким розмежуванням функцій між клієнтською, серверною та аналітичною частинами.

Таблиця 1.1

Основні характеристики предметної області та вимоги до вебсервісу

Компонент / Характеристика	Опис
Джерела даних	IoT-сенсори, пости моніторингу, лабораторні файли, OpenAQ/Copernicus API
Типи задач	Збір, очищення, нормалізація, просторовий аналіз, графіки, сповіщення
Обмеження	Частота вимірювань, якість сенсорів, затримки API, обсяг часових рядів
Вимоги до швидкодії	Оновлення дашборду та карти з мінімальною затримкою
Вимоги до точності	Коректна інтерпретація одиниць, нормалізація часу та перевірка якості даних
Інтеграція	Відкриті екологічні API, GIS-сервіси, BI-панелі, сервіси сповіщень

Екологічний моніторинг охоплює систематичне спостереження за станом повітря, водних ресурсів, ґрунтів і супутніх метеорологічних умов. У межах цифрової системи такі спостереження подаються як часові ряди вимірювань, пов’язані з пунктами контролю, сенсорами, параметрами та географічними координатами.

1.2 Аналіз існуючих рішень у сфері екологічного моніторингу

Існуючі рішення у сфері екологічного моніторингу можна умовно поділити на відкриті платформи екологічних даних, міські сервіси контролю якості повітря, міжнародні API, геоінформаційні системи та спеціалізовані аналітичні панелі. Такі системи дають змогу збирати, переглядати та аналізувати показники стану довкілля, однак їхні можливості часто залежать від конкретного джерела даних, закритої архітектури або обмежених сценаріїв використання.

Одним із прикладів є платформа EcoCity (на рис.1.2), яка орієнтована на збір і відображення показників якості повітря з мережі громадських та автоматизованих станцій моніторингу. Система дає змогу переглядати поточні екологічні показники у зручному візуальному форматі та швидко оцінювати ситуацію в окремих населених пунктах. Водночас така платформа переважно спрямована на відображення вже наявних даних і не завжди забезпечує гнучке налаштування власних алгоритмів аналізу, правил контролю та структури звітності.

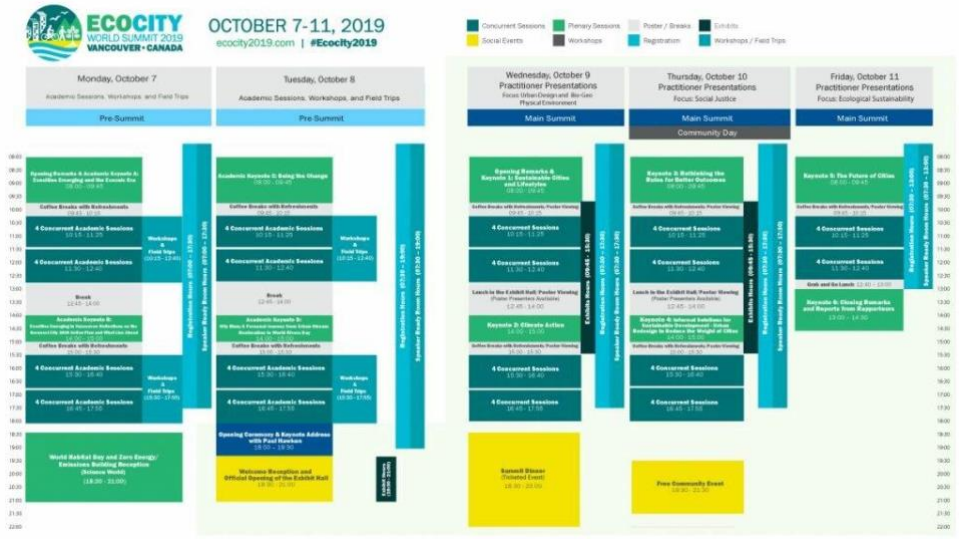

Рис. 1.2. Інтерфейс аналітичної панелі платформи EcoCity

Сервіс SaveEcoBot (рис.1.3) поєднує картографічне відображення екологічної інформації, доступ до показників якості повітря та зручне подання даних для широкого кола користувачів. Його перевагою є відкритість і простота використання, оскільки користувач може швидко отримати інформацію про стан повітря у вибраному регіоні. Разом з тим для дослідницьких або навчально-прикладних задач, де необхідно використовувати власні джерела даних, правила перевірки, додаткові показники та спеціалізовані звіти, потрібна окрема програмна реалізація.

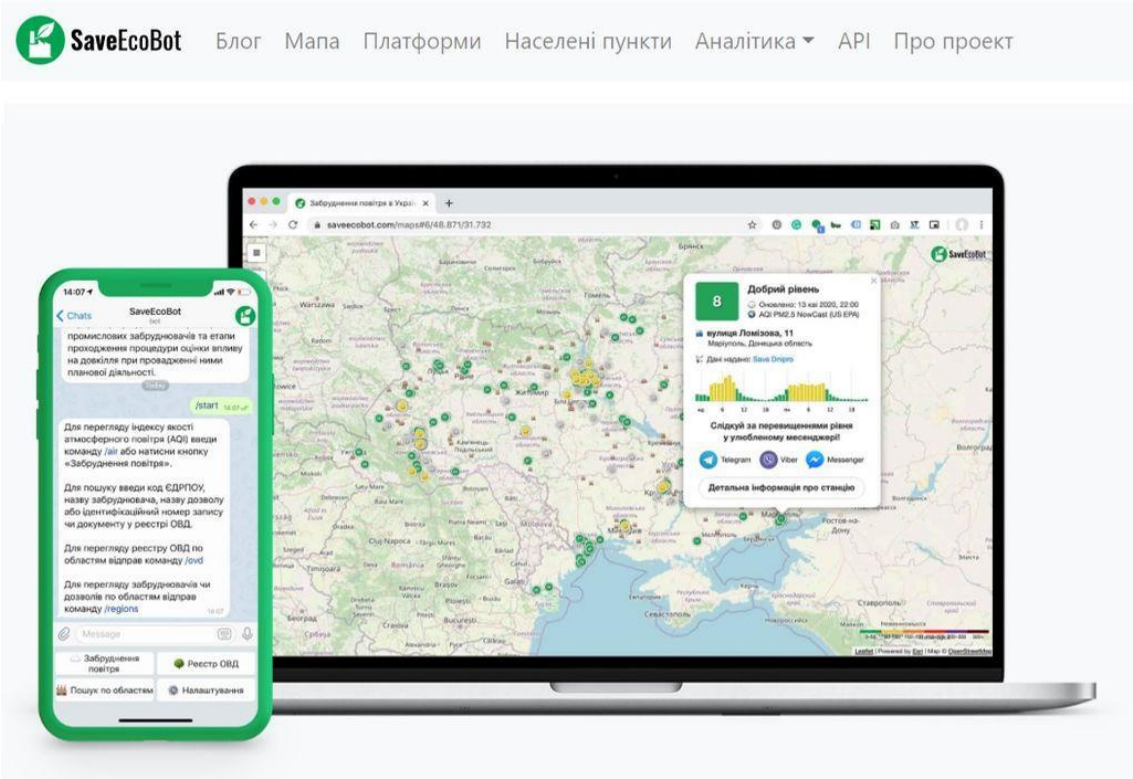

Рис. 1.3. Інтерфейс карти та графіків у сервісі SaveEcoBot

OpenAQ є міжнародною платформою відкритих даних про якість повітря, яка надає доступ до вимірювань через API (інтерфейс представлений на рис.1.4). Таке рішення є корисним для інтеграції глобальних і регіональних екологічних даних у сторонні інформаційні системи. Проте OpenAQ не є повноцінним локальним вебсервісом моніторингу, оскільки не охоплює всі задачі збереження власних вимірювань, налаштування нормативних порогів, побудови внутрішніх звітів і керування користувачами.

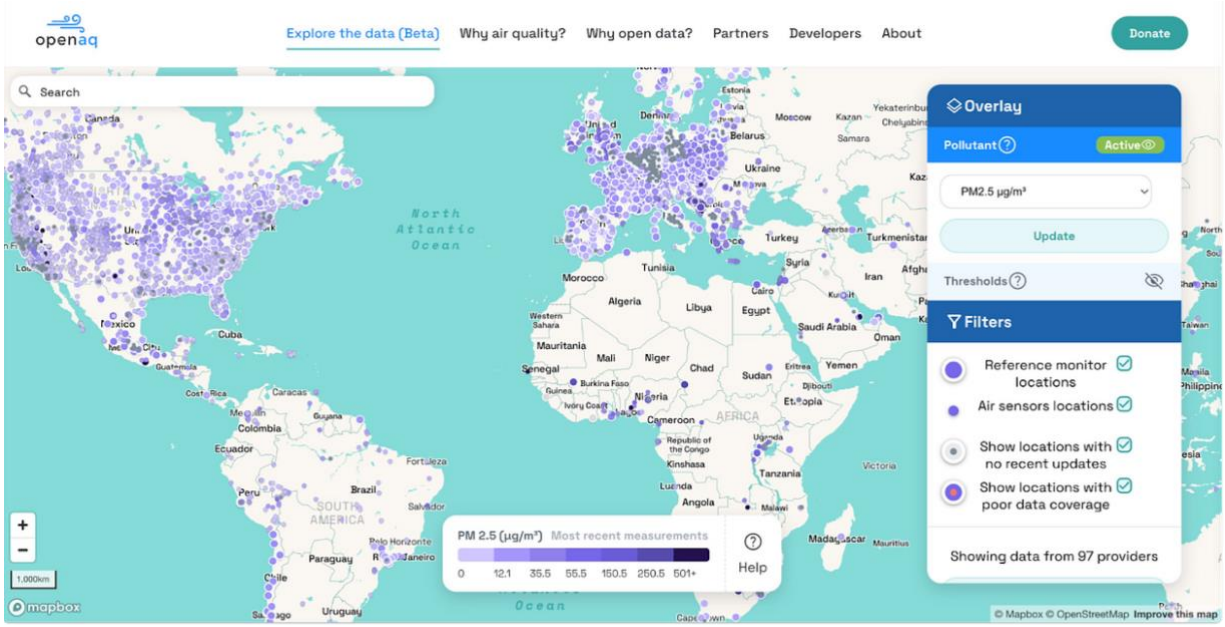

Рис. 1.4. Інтерфейс відкритого API платформи OpenAQ

IQAir AirVisual демонструє підхід до зручної візуалізації індексів якості повітря для широкого кола користувачів (рис.1.5). Сервіс має зрозумілий інтерфейс, глобальне покриття та наочне подання рівнів забруднення. Його доцільно розглядати як приклад якісної користувацької візуалізації. Основним обмеженням є залежність від зовнішньої платформи, її джерел даних, правил доступу та формату подання результатів.

Рис. 1.5. Аналітична панель сервісу IQAir AirVisual

Copernicus CAMS (рис.1.6) використовується для атмосферного моніторингу, прогнозування та аналізу великих просторових наборів екологічних даних. Це потужне джерело інформації для оцінювання стану атмосфери, однак для локальних завдань потрібен додатковий програмний шар, який забезпечує адаптацію, фільтрацію, збереження та представлення даних у форматі, зручному для конкретної організації або навчального проєкту.

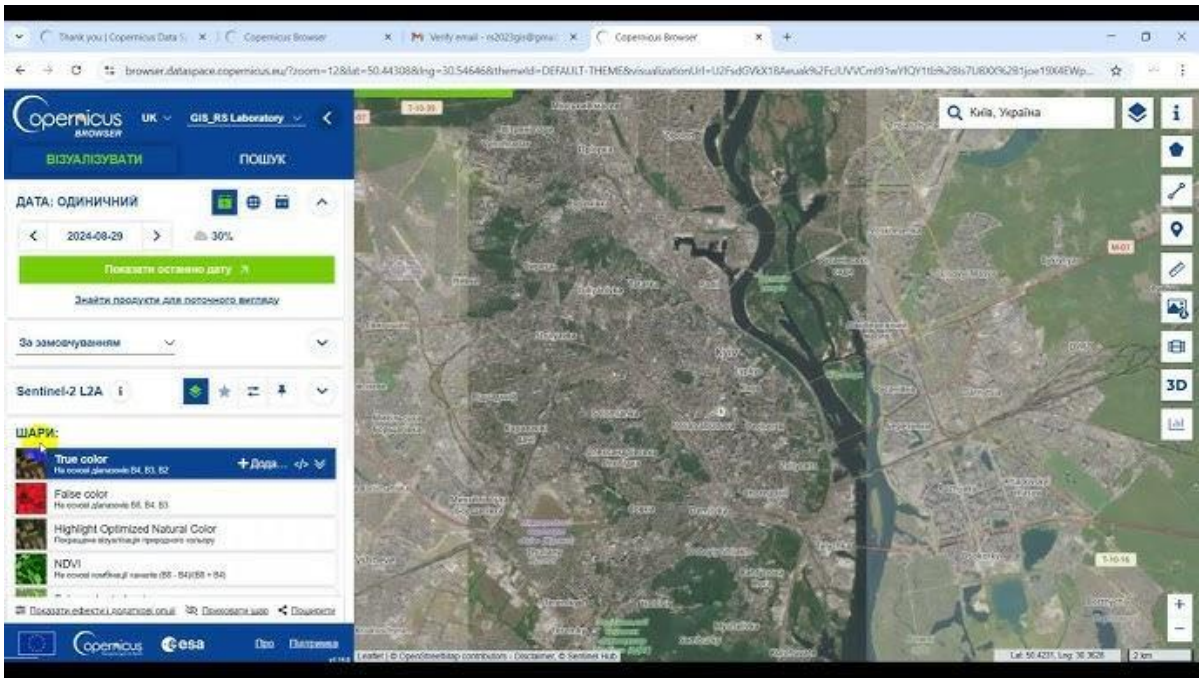

Рис. 1.6. Картографічний інтерфейс сервісів Copernicus

Порівняльну характеристику розглянутих рішень і розроблюваного вебсервісу наведено в таблиці 1.2

Таблиця 1.2

Порівняння існуючих рішень та розроблюваного вебсервісу

Критерій	EcoCity	SaveEcoBot	OpenAQ	IQAir AirVisual	Copernicus CAMS	Розроблюваний вебсервіс
Збір екологічних вимірювань із сенсорів/API	так	так	так	так	так	так
Підтримка різних джерел даних	обмежено	так	обмежено	обмежено	частково	так

Продовження таблиці 1.2

Оновлення даних у близькому до реального часу режимі	так	частково	так	так	частково	так
Агрегація, нормативний аналіз і візуалізація	так	обмежено	так	так	обмежено	так
Незалежність від одного постачальника даних	ні	частково	ні	ні	частково	так
Можливість налаштування правил і модулів	обмежено	обмежено	низька	обмежено	низька	висока

Аналіз існуючих рішень показує, що більшість платформ ефективно виконує окремі функції: відображення екологічної карти, доступ до відкритих даних, візуалізацію індексів якості повітря або роботу з атмосферними прогнозами. Водночас вони не завжди забезпечують повний цикл роботи з даними в межах одного вебсервісу: приймання вимірювань із різних джерел, збереження у власній базі даних, налаштування правил контролю, формування сповіщень, побудову аналітичних графіків і створення звітів.

Отже, розроблення власного вебсервісу екологічного моніторингу є доцільним, оскільки він дає змогу самостійно визначати структуру бази даних, перелік екологічних показників, правила аналізу, формати візуалізації, ролі користувачів і механізми доступу. Такий підхід забезпечує більшу гнучкість, незалежність від одного постачальника даних і можливість адаптації системи до конкретних задач моніторингу довкілля.

1.3 Моделювання предметної області

Моделювання предметної області використовується для формалізації ролей користувачів, сценаріїв роботи та послідовності оброблення екологічних даних. UML-моделі дають змогу перейти від загального опису вебсервісу до конкретної програмної структури.

У моделі прецедентів основними акторами є адміністратор, оператор моніторингу, еколог-аналітик, зареєстрований користувач, сенсорний шлюз і зовнішній сервіс даних. Кожен актор виконує визначений набір дій у межах своєї ролі (рис.1.7).

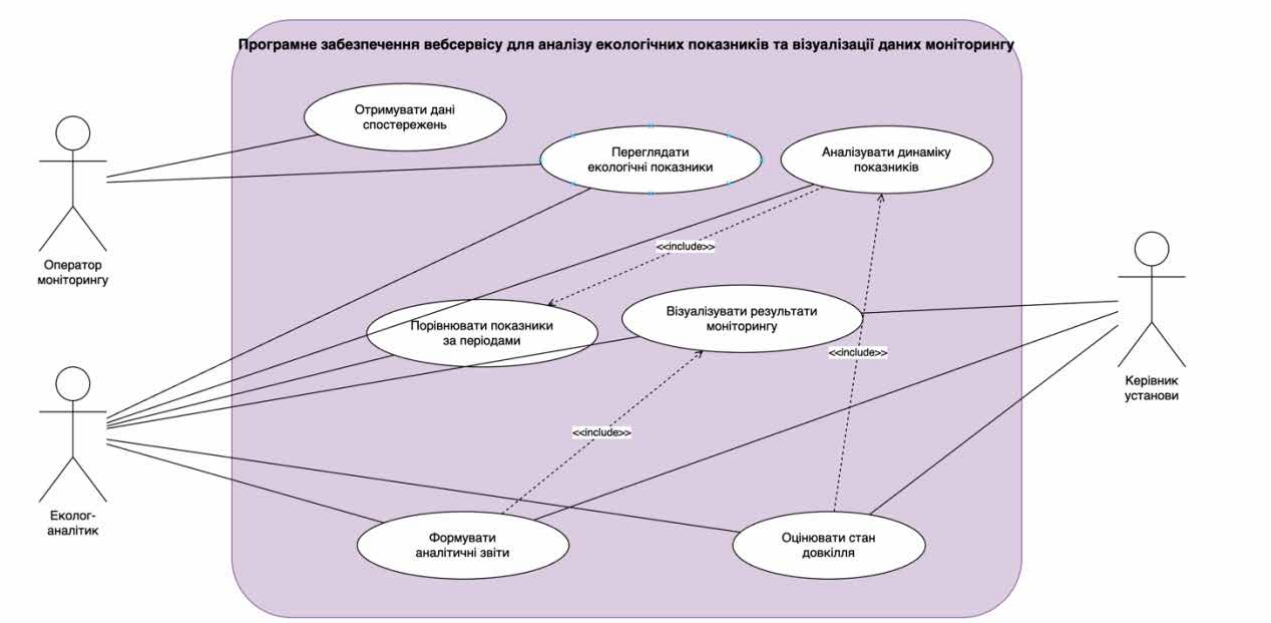

Рис. 1.7. Діаграма прецедентів вебсервісу екологічного моніторингу

Оператор моніторингу відповідає за перегляд поточних вимірювань, контроль стану пунктів спостереження та підтвердження подій. Еколог-аналітик формує звіти, аналізує динаміку показників і налаштовує правила оцінювання.

Адміністратор керує користувачами, ролями, довідниками, джерелами даних і параметрами безпеки. Зовнішні сервіси та сенсорні шлюзи передають дані до API, після чого вони проходять валідацію і зберігаються у базі.

Діаграма послідовності описує сценарій надходження вимірювання: джерело надсилає пакет до API, сервер перевіряє ключ доступу, виконує валідацію структури, нормалізує параметри та записує результат у сховище часових рядів.

Після збереження вимірювання запускається перевірка правил. Якщо значення перевищує норматив або різко відхиляється від попередніх даних, формується подія, створюється запис у журналі та ініціюється сповіщення відповідальних користувачів (рис.1.8).

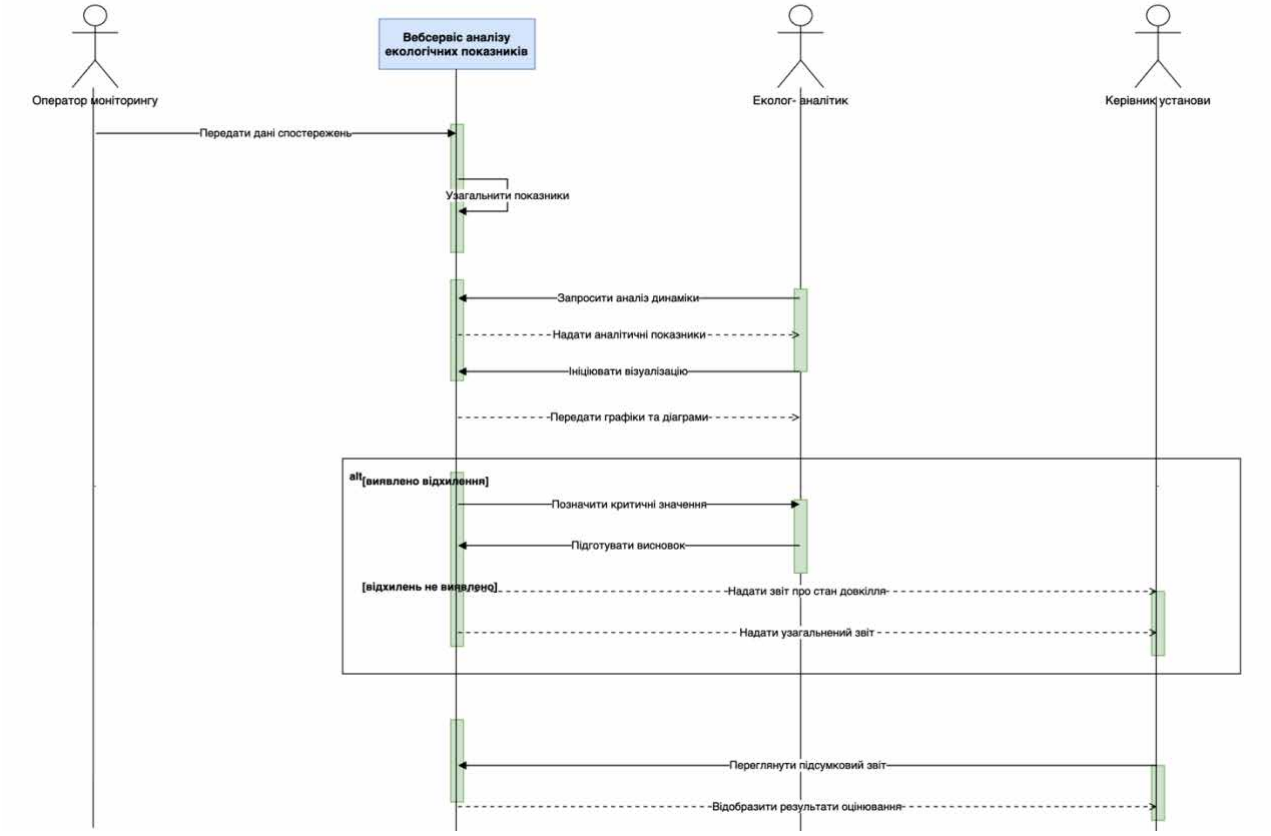

Рис. 1.8. Діаграма послідовності оброблення пакета екологічних вимірювань

Діаграма активності відображає повний цикл: отримання даних, перевірка джерела, очищення, агрегація, порівняння з нормативами, оновлення графіків, відображення карти та формування звітного результату (рис.1.9).

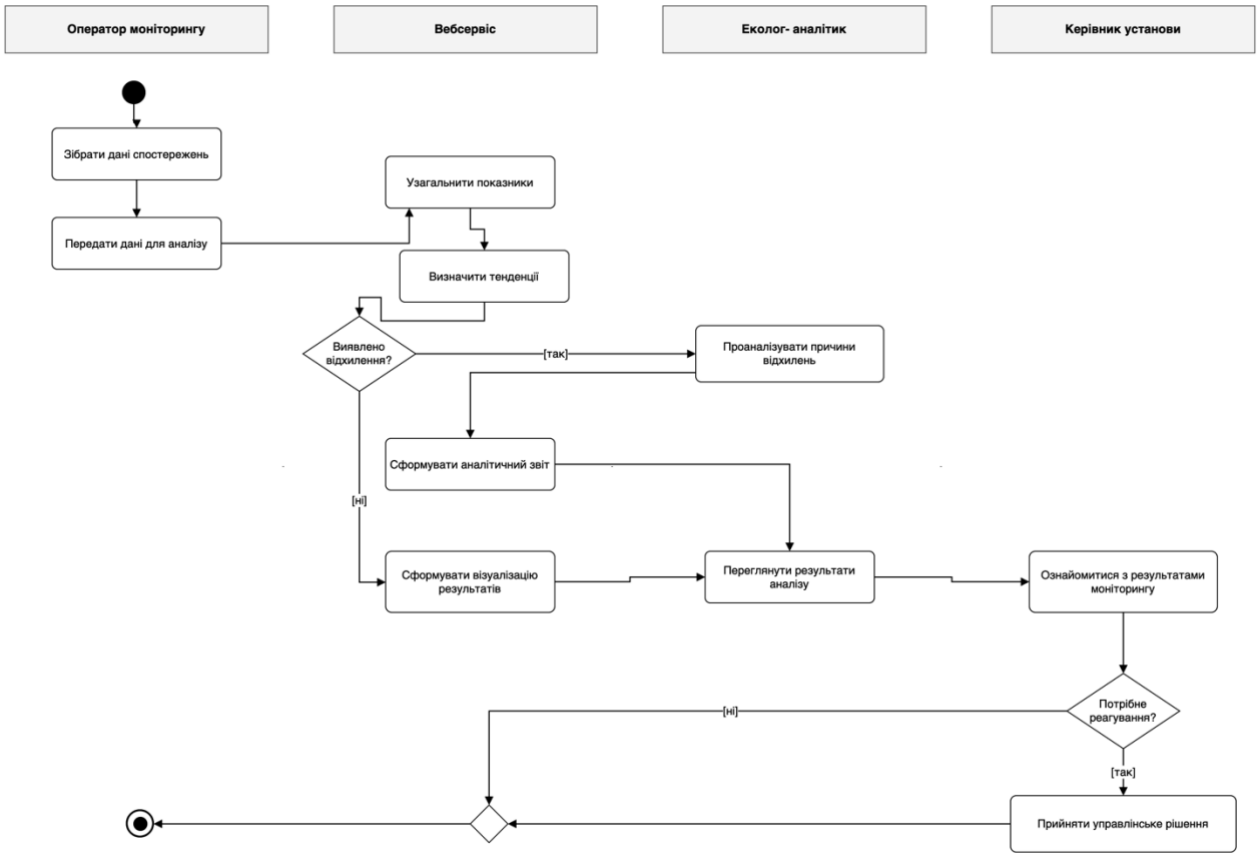

Рис. 1.9. Діаграма активності процесу збору, аналізу та візуалізації екологічних даних

На наступному етапі нормалізовані дані зберігаються у базі даних і передаються до аналітичного модуля. У межах цього модуля виконується агрегація показників за часовими інтервалами, порівняння з установленими нормативами, визначення перевищень і формування аналітичних результатів. Якщо система виявляє критичне значення або різку зміну показника, створюється подія моніторингу, записується інформація до журналу та формується сповіщення для відповідальних користувачів.

Результати оброблення відображаються у веб-інтерфейсі у вигляді графіків, картографічних шарів, таблиць вимірювань і звітів. Така модель дає змогу простежити повний цикл роботи вебсервісу: від отримання екологічних даних до їх аналізу, візуалізації та інформування користувачів. Побудовані UML-діаграми створюють основу для подальшого проєктування бази даних, компонентної структури та алгоритмів програмного забезпечення.

1.4 Аналіз вимог розроблюваного вебсервісу

Аналіз вимог є необхідним етапом проєктування вебсервісу екологічного моніторингу, оскільки саме на цьому етапі визначаються основні функції системи, умови її експлуатації, технічні обмеження та вимоги до захисту даних. Для розроблюваного вебсервісу вимоги сформовано з урахуванням потреб приймання екологічних вимірювань із різних джерел, збереження показників у базі даних, побудови графіків, візуалізації результатів на карті та формування аналітичних звітів.

Насамперед визначено функціональні вимоги, які описують набір дій, що повинна виконувати система під час роботи з екологічними даними. Вони охоплюють приймання вимірювань від сенсорів, зовнішніх API та файлів імпорту, перевірку коректності даних, збереження часових рядів, керування пунктами моніторингу, побудову графіків, формування сповіщень і створення звітів. Узагальнений перелік функціональних вимог наведено в табл. 1.3.

Таблиця 1.3

Функціональні вимоги до вебсервісу екологічного моніторингу

№	Вимога
1	Приймання екологічних вимірювань від сенсорів, CSV/JSON-файлів і зовнішніх API
2	Валідація, очищення та нормалізація пакетів екологічних вимірювань
3	Збереження екологічних показників у сховищі часових рядів
4	Формування сеансів моніторингу та періодів спостереження
5	Обчислення середніх, максимальних, мінімальних значень та індексів якості
6	Виявлення перевищень нормативів та аномальних змін показників
7	Надсилання сповіщень відповідальним користувачам
8	Формування аналітичних звітів, графіків і дашбордів

Після визначення функціональних можливостей системи доцільно сформувати нефункціональні вимоги, які характеризують якість роботи вебсервісу. Вони не описують окремі функції напряму, але визначають швидкодію, доступність,

масштабованість, надійність, зручність використання та можливість подальшого розширення системи. Для вебсервісу екологічного моніторингу особливо важливими є своєчасне оновлення дашбордів, стабільна робота з потоковими даними та підтримка збільшення кількості пунктів спостереження. Основні нефункціональні вимоги подано в табл. 1.4.

Таблиця 1.4

Нефункціональні вимоги до вебсервісу

№	Вимога
1	Оброблення екологічних потоків у режимі, близькому до реального часу
2	Висока доступність сервісів (24/7)
3	Масштабованість при зростанні кількості пунктів моніторингу
4	Відмовостійкість та підтримка механізмів повторної обробки (retry)
5	Мінімальна затримка між отриманням вимірювання та оновленням дашборду
6	Можливість інтеграції з зовнішніми BI- та ERP-системами

Як видно з табл. 1.4, вебсервіс повинен забезпечувати оброблення екологічних даних у режимі, близькому до реального часу, підтримувати безперервну роботу основних сервісів і залишатися працездатним у разі помилок окремих джерел даних. Також важливо передбачити масштабованість системи, оскільки кількість сенсорів, пунктів моніторингу та користувачів може поступово збільшуватися.

Окрему групу становлять технічні вимоги, які визначають програмні та інфраструктурні умови реалізації вебсервісу. До них належать підтримка REST API, використання захищеного протоколу передавання даних, автентифікація користувачів і джерел вимірювань, застосування бази даних для часових і просторових показників, резервне копіювання та можливість розгортання системи в локальному або хмарному середовищі. Перелік технічних вимог наведено в табл. 1.5.

Таблиця 1.5

Технічні вимоги до вебсервісу екологічного моніторингу

№	Вимога
1	Підтримка REST API для приймання екологічних вимірювань
2	Використання захищеного протоколу HTTPS
3	Підтримка автентифікації користувачів і джерел даних (JWT/API key/mTLS)
4	Використання бази часових і просторових даних для збереження показників
5	Наявність механізмів резервного копіювання та відновлення даних
6	Можливість розгортання у хмарному або локальному середовищі
7	Підтримка контейнеризації та мікросервісної архітектури

Технічні вимоги, наведені в табл. 1.5, формують основу для вибору архітектури та технологічного стеку системи. Вебсервіс має підтримувати структурований обмін даними через API, захищене передавання інформації, збереження вимірювань з прив’язкою до часу та координат, а також можливість подальшої інтеграції з картографічними, аналітичними та зовнішніми інформаційними сервісами.

Оскільки вебсервіс працює з екологічними даними, користувацькими обліковими записами, службовими ключами доступу та журналами подій, важливо окремо визначити вимоги до інформаційної безпеки. Вони передбачають автентифікацію користувачів і сенсорних джерел, авторизацію відповідно до ролей, шифрування даних, захист API від несанкціонованих запитів, журналювання дій та контроль цілісності службової інформації. Основні вимоги до інформаційної безпеки подано в табл. 1.6.

Таблиця 1.6

Вимоги до інформаційної безпеки вебсервісу

№	Вимога
1	Обов’язкова автентифікація користувачів та сенсорів екологічної мережі
2	Шифрування даних під час передавання та зберігання
3	Розмежування прав доступу відповідно до ролей
4	Захист API від несанкціонованих запитів (rate limiting, логування)

Продовження таблиці 1.6

5	Забезпечення цілісності та незмінності журналів подій
6	Реєстрація та моніторинг інцидентів безпеки

Вимоги, наведені в табл. 1.3-1.6, комплексно описують функціональні, якісні, технічні та безпекові характеристики розроблюваного вебсервісу. Їх виконання дає змогу створити систему, здатну приймати екологічні вимірювання з різних джерел, перевіряти та зберігати їх, виконувати аналітичну обробку, будувати графіки, формувати звіти, відображати дані на карті та інформувати користувачів про критичні зміни показників.

Отже, проведений аналіз вимог створює основу для подальшого проєктування інформаційного та програмного забезпечення вебсервісу екологічного моніторингу. Сформовані вимоги дозволяють перейти до побудови моделі даних, визначення компонентної структури, вибору технологій реалізації та розроблення алгоритмів оброблення екологічних показників.

1.5 Постановка завдання

У межах кваліфікаційної роботи необхідно спроєктувати та реалізувати програмне забезпечення вебсервісу, що забезпечує повний цикл роботи з екологічними показниками: від приймання вимірювань до їх візуалізації, аналізу та експорту звітів.

Розроблювана система має моделювати взаємодію сенсорних джерел, серверного API, бази даних, аналітичного модуля, підсистеми правил, сервісу сповіщень і клієнтського вебінтерфейсу.

Вхідними даними є пакети вимірювань із часовою міткою, координатами, кодом пункту моніторингу, типом показника, одиницею вимірювання, числовим значенням та службовими атрибутами джерела.

Вихідними даними є очищені та збережені записи, агреговані показники, карти забруднення, графіки динаміки, повідомлення про перевищення нормативів, експортовані звіти та журнали дій користувачів.

Основними функціями системи є приймання даних через API, валідація, нормалізація, збереження, порівняння з нормативами, побудова графіків, відображення на карті, формування звітів і керування користувачами.

- імітацію надходження пакетів екологічних вимірювань від сенсорних сенсорів;

- автентифікація користувачів та сенсорів екологічної мережі;

- валідацію та нормалізацію екологічних даних;

- збереження даних у базі часових рядів;

- обчислення похідних метрик та формування сесій сеансів моніторингу;

- перевірку екологічних вимірювань відповідно до налаштованих правил і порогів;

- створення записів попереджень і надсилання сповіщень;

- формування звітів і візуалізацію показників у веб-інтерфейсі;

- ведення журналу подій та забезпечення механізмів резервного копіювання.

Програмна реалізація повинна мати модульну вебархітектуру з окремими рівнями клієнтського інтерфейсу, серверної логіки, доступу до даних, аналітики, сповіщень та інтеграцій із зовнішніми джерелами.

Постановка завдання полягає у створенні вебсервісу, який забезпечує надійне збереження екологічних даних, зрозумілу візуалізацію результатів моніторингу та підтримку прийняття рішень на основі актуальних показників довкілля.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних вебсервісу екологічного моніторингу побудована з урахуванням структури джерел даних, типів екологічних показників та зв'язків між ними. На рис. 2.1 подано ER-діаграму, яка відображає основні сутності системи: DataSource, SensorSource, ExternalAPISource, IndicatorCategory, Indicator та DataSource_Indicator.

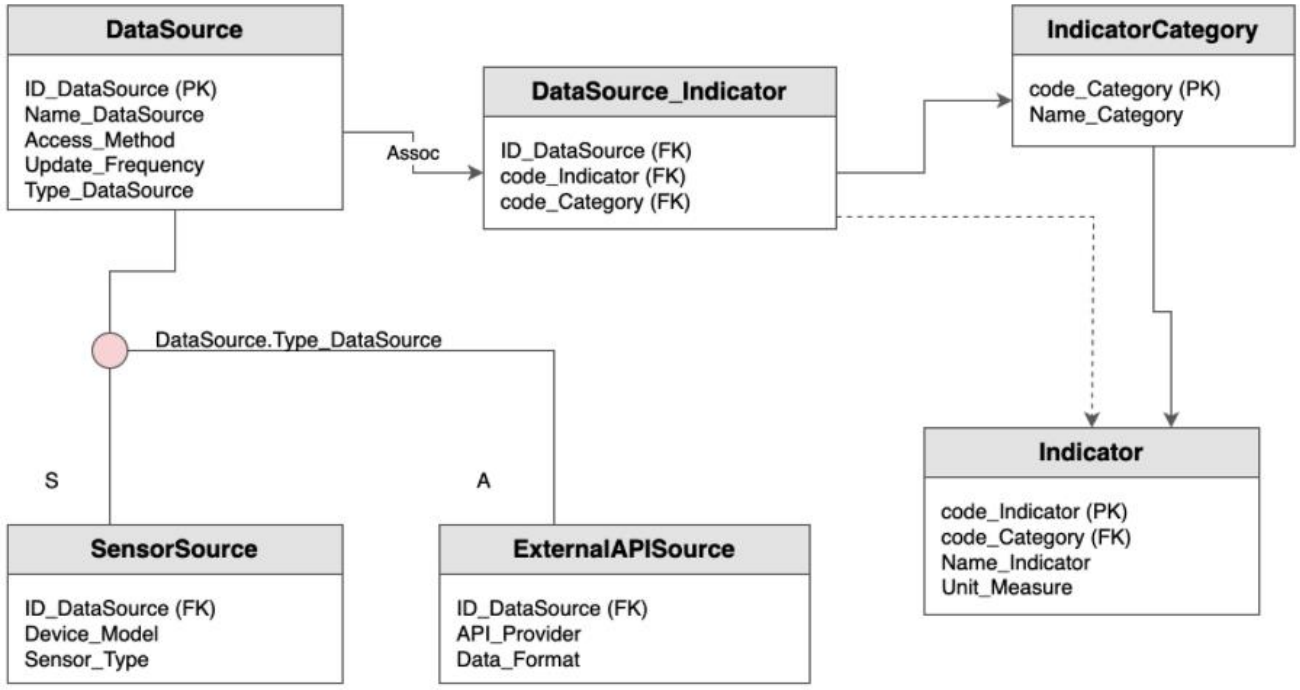

Рис. 2.1. ER-діаграма логічної моделі даних вебсервісу екологічного моніторингу

Центральною сутністю моделі є DataSource, яка описує джерело надходження екологічних даних. Вона містить первинний ключ ID_DataSource, назву джерела, спосіб доступу, частоту оновлення та тип джерела. Атрибут Type_DataSource використовується для розмежування джерел на сенсорні пристрої та зовнішні API.

Для уточнення типів джерел у моделі використано спеціалізацію. Сутність SensorSource зберігає характеристики фізичних сенсорів, зокрема модель пристрою та тип сенсора. Сутність ExternalAPISource описує зовнішні програмні джерела

даних, містить назву API-провайдера та формат отримання даних. Обидві сутності мають зовнішній ключ ID_DataSource, що забезпечує зв’язок із базовою сутністю DataSource.

Сутність IndicatorCategory призначена для класифікації екологічних показників за категоріями, наприклад показники якості повітря, води або метеорологічні параметри. Первинним ключем є code_Category. Сутність Indicator описує конкретний показник моніторингу, містить код показника, назву, одиницю вимірювання та зовнішній ключ code_Category, який пов’язує показник із відповідною категорією.

Зв’язок між джерелами даних і показниками реалізовано через проміжну сутність DataSource_Indicator. Її використання є необхідним, оскільки одне джерело даних може передавати кілька екологічних показників, а один показник може надходити з кількох різних джерел. Тому зв’язок багато-до-багатьох між DataSource та Indicator розкладено на два зв’язки один-до-багатьох через асоціативну таблицю. Це відповідає вимогам нормалізації реляційної моделі.

Запропонована структура відповідає першій нормальній формі, оскільки всі атрибути сутностей є атомарними, а повторювані групи даних відсутні. Модель також відповідає другій нормальній формі, оскільки неключові атрибути залежать від повного первинного ключа відповідної сутності, а дані про джерела, категорії та показники винесені в окремі таблиці.

Відповідність третій нормальній формі забезпечується тим, що в моделі відсутні транзитивні залежності між неключовими атрибутами. Наприклад, назва категорії зберігається тільки в IndicatorCategory, одиниця вимірювання показника - тільки в Indicator, а технічні параметри сенсора - тільки в SensorSource. Завдяки цьому не дублюються довідкові значення, зменшується ризик помилок під час оновлення та спрощується супровід бази даних.

Нормалізована структура ER-моделі дає змогу уникнути аномалій вставлення, оновлення й видалення даних. Нове джерело можна додати без дублювання показників, нову категорію - без зміни структури джерел, а новий показник - без

повторного внесення інформації про API або сенсор. Такий підхід підвищує цілісність даних і забезпечує зручне розширення вебсервісу.

2.2 Діаграма класів і кооперації

Для опису об’єктно-орієнтованої структури вебсервісу екологічного моніторингу побудовано діаграму класів, наведену на рис. 2.2. Вона відображає основні програмні сутності системи, їх атрибути та зв’язки, що виникають під час збереження, аналізу й візуалізації екологічних даних.

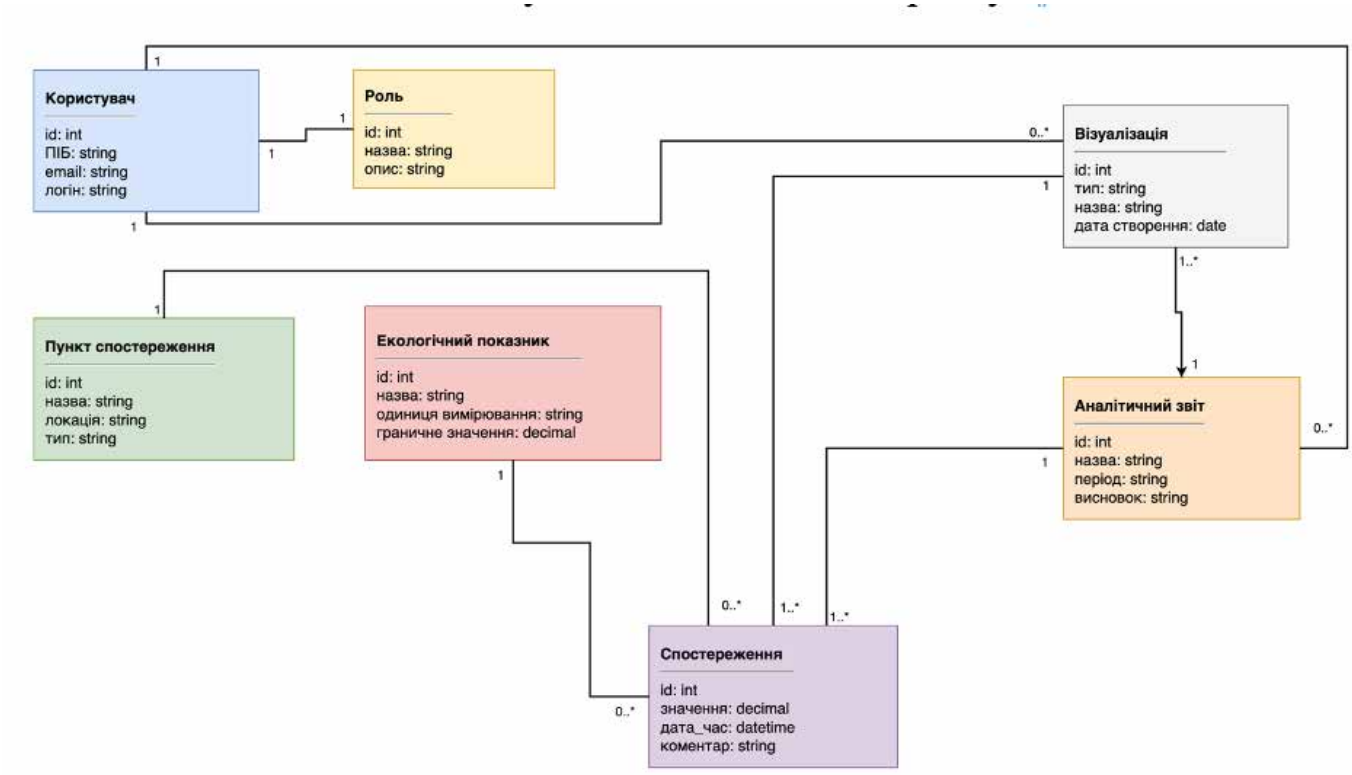

Рисунок 2.2 – Діаграма класів вебсервісу екологічного моніторингу

У моделі виділено сім основних класів: Користувач, Роль, Пункт спостереження, Екологічний показник, Спостереження, Візуалізація та Аналітичний звіт. Такий склад класів відповідає основним процесам вебсервісу: керуванню користувачами, роботі з пунктами моніторингу, фіксації екологічних вимірювань, побудові графіків і формуванню звітних матеріалів.

Клас Користувач описує особу, яка працює із системою. Він містить ідентифікатор, ПІБ, електронну пошту та логін. Через зв’язок із класом Роль

визначаються права доступу користувача до функцій вебсервісу. Наприклад, адміністратор може керувати довідниками й обліковими записами, а еколог-аналітик має доступ до аналізу показників, перегляду спостережень і формування звітів.

Клас Роль призначений для збереження інформації про тип доступу користувача. Він містить ідентифікатор, назву та опис ролі. Винесення ролей в окремий клас є доцільним, оскільки дозволяє не дублювати інформацію про права доступу в кожному обліковому записі та спрощує подальше розширення системи.

Клас Пункт спостереження описує місце, у якому здійснюється екологічний моніторинг. До його атрибутів належать ідентифікатор, назва, локація та тип пункту. Один пункт спостереження може містити багато записів спостережень, що відповідає логіці накопичення даних у часі.

Клас Екологічний показник визначає параметр, який контролюється системою. Він містить назву показника, одиницю вимірювання та граничне значення. Наявність граничного значення дає змогу виконувати автоматичне порівняння фактичних вимірювань із допустимими межами та виявляти перевищення.

Клас Спостереження є центральним класом моделі, оскільки саме в ньому фіксуються фактичні екологічні дані. Він містить ідентифікатор, значення, дату й час фіксації та коментар. Кожне спостереження пов’язане з конкретним пунктом спостереження та певним екологічним показником. Завдяки цьому система може зберігати часові ряди вимірювань і виконувати подальшу аналітичну обробку.

Клас Візуалізація відповідає за подання результатів моніторингу у вигляді графіків, діаграм або картографічних елементів. Він містить тип, назву та дату створення. Візуалізація формується на основі одного або кількох спостережень і використовується користувачем для швидкого аналізу стану довкілля.

Клас Аналітичний звіт узагальнює результати аналізу за вибраний період. Він містить назву, період і висновок. Звіт формується користувачем на основі спостережень та може включати створені візуалізації. Такий зв’язок дозволяє поєднати числові дані, графічне представлення та текстові висновки в одному результаті.

Зв’язки між класами відображають логіку роботи вебсервісу. Користувач має визначену роль, обирає пункт спостереження, переглядає спостереження та використовує візуалізації. Пункт спостереження містить багато спостережень, а екологічний показник фіксується у відповідних записах вимірювань. Спостереження можуть відображатися у візуалізаціях і узагальнюватися в аналітичних звітах. Така структура забезпечує цілісне представлення предметної області та є придатною для подальшої програмної реалізації.

Для уточнення взаємодії між класами побудовано кооперації, які показують типові сценарії роботи системи. Перша кооперація, наведена на рис. 2.3, описує сценарій вибору пункту спостереження та перегляду пов’язаних з ним вимірювань.

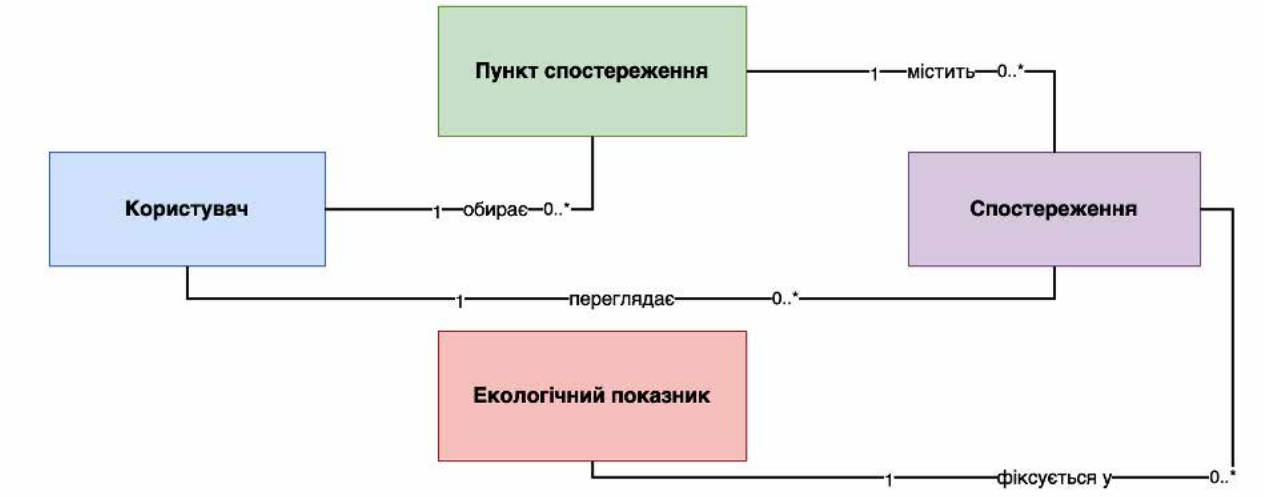

Рисунок 2.3 – Кооперація класів під час вибору пункту спостереження та перегляду спостережень

У цьому сценарії користувач обирає пункт спостереження, після чого система отримує пов’язані з ним записи спостережень. Кожне спостереження містить значення конкретного екологічного показника. Така кооперація описує базову дію користувача, пов’язану з переглядом даних моніторингу за певним місцем спостереження.

Друга кооперація, подана на рис. 2.4, відображає процес аналізу екологічного показника та подання результатів у вигляді візуалізації.

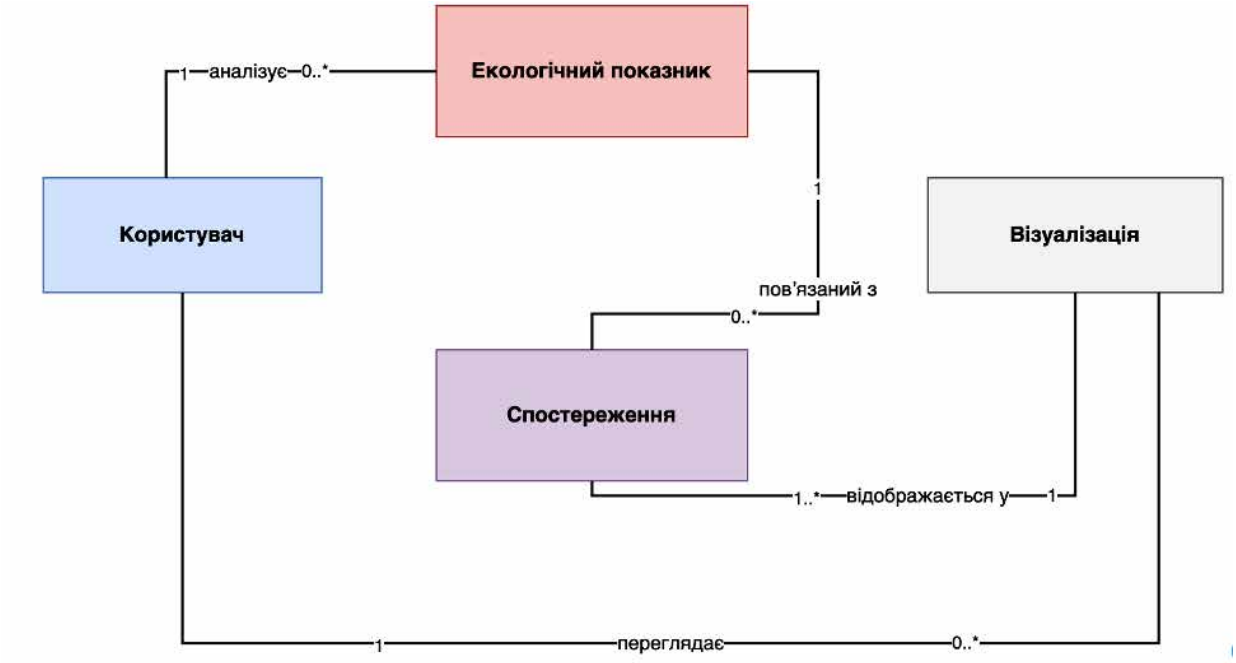

Рисунок 2.4 – Кооперація класів під час аналізу екологічного показника та побудови візуалізації

У межах цього сценарію користувач аналізує вибраний екологічний показник, який пов’язаний із набором спостережень. Отримані спостереження передаються до візуалізації, де відображаються у вигляді графіка або іншого наочного представлення. Така взаємодія забезпечує швидке виявлення змін показників у часі та оцінювання динаміки екологічного стану.

Третя кооперація, наведена на рис. 2.5, описує процес формування аналітичного звіту.

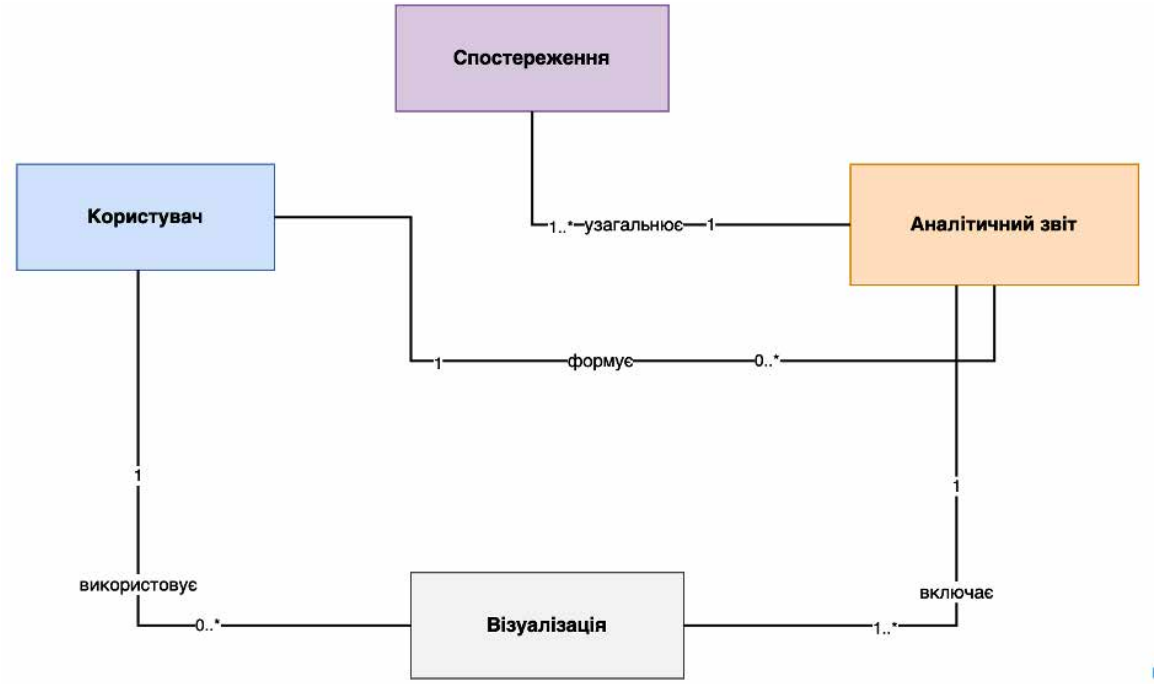

Рисунок 2.5 – Кооперація класів під час формування аналітичного звіту

У цьому сценарії користувач формує аналітичний звіт на основі спостережень за вибраний період. Спостереження узагальнюються у звіті, а створені візуалізації можуть бути включені до його складу як графічне підтвердження результатів аналізу. Такий підхід дозволяє поєднати первинні дані моніторингу, результати їх оброблення та висновки спеціаліста.

Отже, діаграма класів і кооперації формалізують програмну структуру вебсервісу екологічного моніторингу. Діаграма класів визначає основні об’єкти системи, їх атрибути та зв’язки, а кооперації уточнюють порядок взаємодії між цими об’єктами у ключових сценаріях: перегляд спостережень, аналіз екологічних показників і формування аналітичного звіту. Отримані моделі є основою для подальшого проєктування компонентів, пакетної структури та алгоритмів роботи системи.

2.3 Діаграма компонентів

Компонентна модель показує архітектуру вебсервісу на рівні клієнтської частини, серверних сервісів, сховищ даних, аналітичних модулів і зовнішніх інтеграцій.

Клієнтський вебінтерфейс забезпечує перегляд карти, графіків, таблиць вимірювань, подій і звітів. Усі операції виконуються через захищені запити до API-шлюзу.

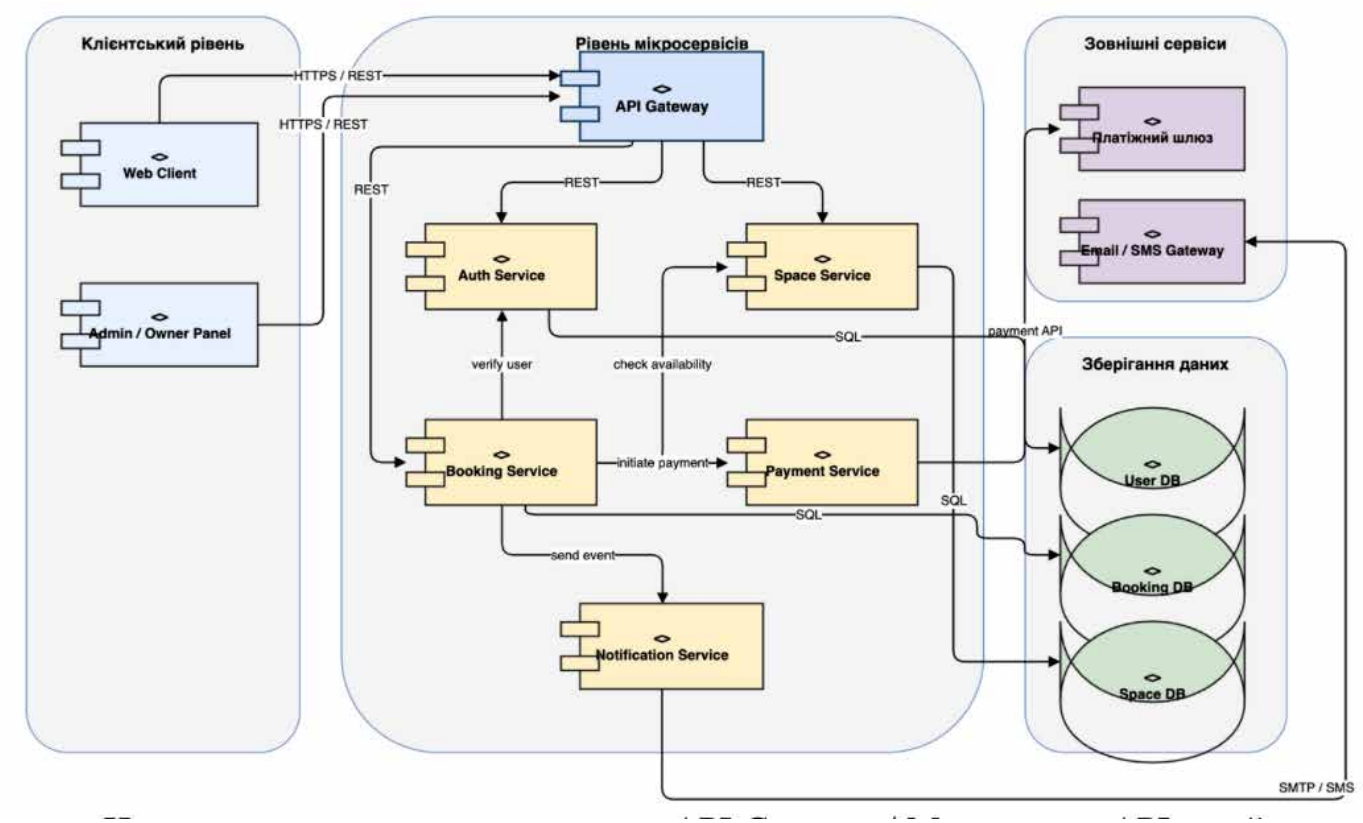

Рис. 2.6. Діаграма компонентів вебсервісу аналізу екологічних показників

API-шлюз приймає запити користувачів і сенсорних джерел, перевіряє доступ, маршрутизує дані до відповідних модулів і повертає структуровані відповіді у форматі JSON.

Сервіс збирання та нормалізації відповідає за перевірку пакетів вимірювань, приведення одиниць до єдиного формату, синхронізацію часу та підготовку записів до збереження.

Аналітичний сервіс виконує агрегування даних, розрахунок статистичних показників, перевірку нормативів, пошук аномалій і підготовку даних для графічного відображення.

Сервіс звітності формує табличні та графічні результати за обраними фільтрами, а також забезпечує експорт у поширені формати для подальшого використання.

База даних зберігає довідники, користувачів, пункти моніторингу, вимірювання, нормативи, події, звіти та журнали. Для просторових запитів передбачено використання географічних типів даних.

Компонентна структура забезпечує масштабованість і розширюваність вебсервісу, оскільки окремі модулі можна змінювати або доповнювати без порушення роботи всієї системи.

Таблиця 2.2

Характеристика основних компонентів вебсервісу екологічного моніторингу

Компонент	Призначення	Основна взаємодія
Веб-клієнт / Environmental Dashboard	Забезпечує інтерфейс для перегляду карти, графіків, звітів і екологічних подій	Взаємодіє з API-шлюзом через REST API
API-шлюз	Приймає зовнішні запити, карти спостереженьзує їх до внутрішніх сервісів і контролює доступ до API	Отримує запити від клієнта, сенсорів та зовнішніх API
Сервіс автентифікації та керування доступом	Виконує перевірку користувачів, сенсорів, токенів і ролей доступу	Взаємодіє з API-шлюзом, БД та іншими сервісами
Сервіс збору та нормалізації екологічних даних	Приймає сирі пакети екологічних вимірювань, перевіряє їх і приводить до єдиного формату	Отримує дані від API-шлюзу, записує нормалізовані екологічні дані
Сервіс аналітики, правил і попереджень	Аналізує параметри, перевіряє правила моніторингу та формує події попереджень	Читає екологічні дані, записує події та передає команди сповіщень
Сервіс звітності, дашбордів і калібрування	Формує аналітичні звіти, показники дашбордів і дані для планування калібрування сенсорів і перевірки якості даних	Читає агреговані дані з БД та передає результати у веб-клієнт

Запропонована компонентна структура є модульною, тому окремі частини вебсервісу можна змінювати або доповнювати без повної перебудови системи. Наприклад, можна додати нове джерело екологічних даних, підключити інший картографічний сервіс, розширити набір аналітичних правил або змінити формат звітності. Це забезпечує масштабованість, зручність супроводу та можливість подальшого розвитку вебсервісу екологічного моніторингу.

2.4 Діаграма пакетів

Діаграма пакетів використовується для подання логічної організації програмного коду вебсервісу екологічного моніторингу. Вона показує, як основні частини системи згруповані за призначенням і які залежності існують між клієнтським інтерфейсом, прикладною логікою, платформними сервісами, аналітикою, даними та зовнішніми інтеграціями. Діаграму пакетів наведено на рис. 2.7.

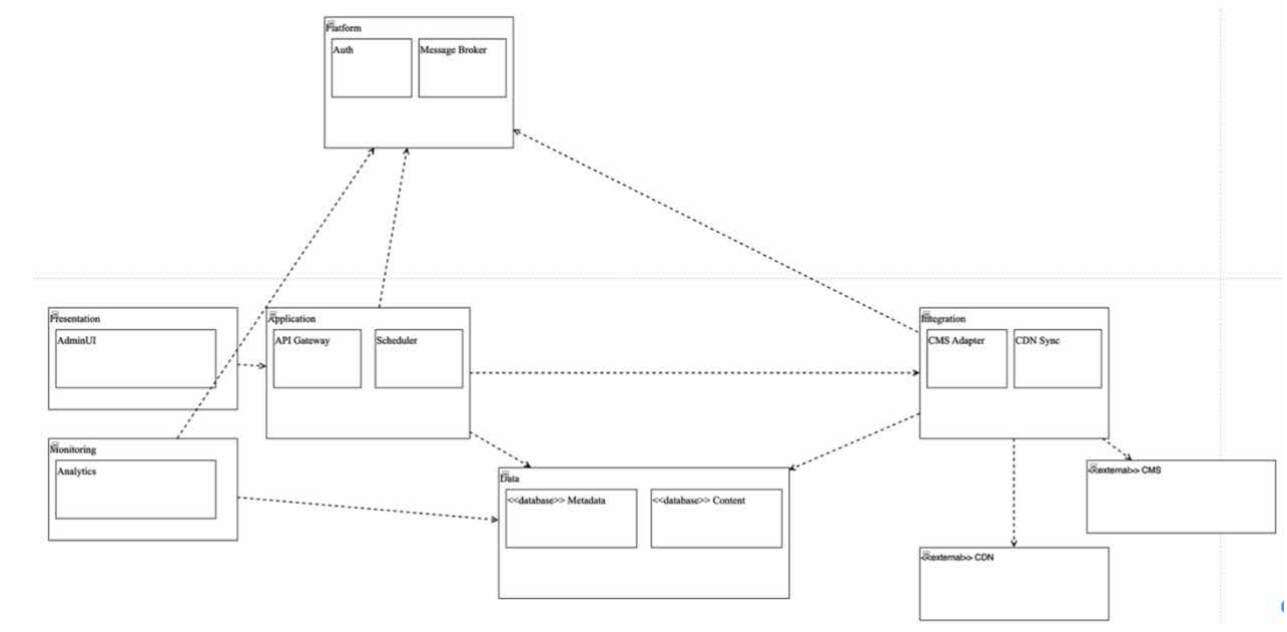

Рисунок 2.7 – Діаграма пакетів вебсервісу екологічного моніторингу

Пакет Presentation відповідає за інтерфейс користувача. У його складі розміщується модуль AdminUI, через який користувачі переглядають карту пунктів спостереження, графіки екологічних показників, таблиці вимірювань, події

перевищення та сформовані звіти. Цей пакет не працює з базою даних напряму, а передає запити до прикладного рівня.

Пакет Application містить основну прикладну логіку вебсервісу. До нього входять API Gateway та Scheduler. API Gateway приймає запити від клієнтської частини й зовнішніх джерел даних, перевіряє доступ і передає їх до потрібних модулів. Scheduler використовується для періодичного оновлення екологічних вимірювань, запуску імпорту даних, перевірки нормативів і формування планових звітів.

Пакет Platform об’єднує загальні службові механізми, необхідні для стабільної роботи системи. Модуль Auth забезпечує автентифікацію користувачів, перевірку ролей і контроль доступу. Message Broker використовується для передавання подій між сервісами, наприклад під час надходження нового вимірювання, виявлення перевищення або запуску сповіщення.

Пакет Monitoring відповідає за аналітичну частину вебсервісу. Його модуль Analytics виконує оброблення екологічних показників, розрахунок середніх, мінімальних і максимальних значень, порівняння з граничними нормами та підготовку даних для візуалізації. Результати роботи цього пакета використовуються у дашбордах і звітах.

Пакет Data призначений для роботи зі сховищами даних. У ньому виділено базу Metadata, де зберігаються користувачі, ролі, пункти спостереження, типи показників і налаштування системи, а також базу Content, яка містить екологічні вимірювання, події, результати аналізу, звіти та журнали дій. Такий поділ спрощує обслуговування даних і зменшує залежність між довідковою та операційною інформацією.

Пакет Integration забезпечує взаємодію з зовнішніми сервісами. До нього входять адаптери для підключення відкритих екологічних API, картографічних сервісів, сервісів сповіщень або зовнішніх сховищ. Через цей пакет вебсервіс може отримувати додаткові дані, синхронізувати результати та передавати інформацію до сторонніх систем.

Залежності між пакетами побудовані так, щоб зберегти модульність системи. Клієнтська частина звертається до Application, прикладний рівень використовує

Platform для безпеки та обміну подіями, Monitoring отримує дані з Data, а Integration забезпечує зв’язок із зовнішніми джерелами. Завдяки такій структурі окремі пакети можна змінювати або розширювати без повної перебудови вебсервісу.

Отже, діаграма пакетів відображає логічний поділ програмного забезпечення вебсервісу екологічного моніторингу на взаємопов’язані частини. Запропонована структура є компактною, зрозумілою та придатною для реалізації, оскільки розмежовує інтерфейс, прикладну логіку, аналітику, роботу з даними, платформні сервіси та зовнішні інтеграції.

2.5 Висновки до другого розділу

У другому розділі спроєктовано інформаційне та програмне забезпечення вебсервісу екологічного моніторингу. Розроблено логічну модель даних, яка описує користувачів, пункти спостереження, сенсори, вимірювання, нормативи, події та звіти.

Побудована ER-модель відповідає вимогам нормалізації, забезпечує відсутність надлишкового дублювання та підтримує коректні зв’язки між довідковими, вимірювальними та аналітичними даними.

Діаграма класів і кооперації відобразили основні програмні абстракції та сценарії роботи: реєстрацію пункту моніторингу, оброблення вимірювання і формування попередження про перевищення нормативу.

Компонентна та пакетна моделі підтвердили доцільність модульної архітектури, у якій окремо виділено клієнтський інтерфейс, API, збирання даних, аналітику, базу даних, звітність і зовнішні інтеграції.

Отримані моделі створюють основу для переходу до фізичної реалізації бази даних, вибору технологічного стеку та розроблення алгоритмів програмних модулів вебсервісу.

У другому розділі спроєктовано інформаційне та програмне забезпечення вебсервісу екологічного моніторингу. Розроблено логічну модель даних, яка описує

користувачів, пункти спостереження, сенсори, вимірювання, нормативи, події та звіти.

3 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ ТА РОЗРОБЛЕННЯ АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибору технологічних засобів та інструментів реалізації

Для реалізації вебсервісу обрано технологічний стек, орієнтований на швидке створення API, роботу з часовими рядами, просторовими даними та інтерактивну візуалізацію результатів моніторингу.

Серверну частину доцільно реалізувати мовою Python із використанням FastAPI, оскільки цей фреймворк підтримує типізацію, автоматичну документацію OpenAPI, зручну валідацію запитів і високу продуктивність для REST-сервісів.

Клієнтський рівень може бути побудований на React, що дає змогу створювати компонентний інтерфейс із дашбордами, фільтрами, таблицями, графіками та інтерактивною картою.

Для зберігання даних обрано SQLlite із розширеннями PostGIS і TimescaleDB. Така конфігурація поєднує реляційні зв’язки, просторові запити та ефективну роботу з часовими рядами вимірювань.

Таблиця 3.1

Технологічні засоби реалізації вебсервісу екологічного моніторингу

Технологічний засіб	Призначення в системі	Обґрунтування вибору
Python	Реалізація бізнес-логіки вебсервісу	Простота розробки, зручна робота з даними, файлами та алгоритмами
FastAPI + React	Побудова графічного інтерфейсу	Дозволяє створити API та інтерактивний вебінтерфейс із дашбордами
SQLLite/PostGIS	Серверна база даних	Забезпечує реляційне, просторове та часово-рядове збереження даних
SQLAlchemy / asyncpg	Робота з базою даних	Забезпечує створення таблиць, запити, додавання та оновлення записів

Прожлвження таблиці 3.1

Leaflet / Chart.js / ECharts	Візуалізація екологічних вимірювань	Дає змогу будувати графіки концентрації забруднювачів, температури, концентрацій повітряного середовища та рівня забруднення
hashlib / bcrypt	Захист паролів	Забезпечує збереження паролів у вигляді хешів
CSV / PDF export	Формування звітів	Дозволяє експортувати результати аналізу та журнали подій
Docker / Docker Compose	Формування інсталяційного пакету	Дає змогу зібрати вебсервіс у виконуваний файл для запуску на комп’ютері користувача
Git	Контроль версій	Дозволяє зберігати історію змін програмного коду

Для картографічної візуалізації доцільно використовувати Leaflet або React Leaflet, а для графіків - Chart.js чи ECharts. Вони забезпечують побудову динаміки показників, порівняння територій і відображення критичних значень.

Архітектура вебсервісу передбачає поділ на модулі авторизації, керування пунктами моніторингу, приймання вимірювань, аналітики, правил, сповіщень, звітності та адміністрування.

Обраний стек забезпечує масштабованість, підтримку контейнерного розгортання, інтеграцію із зовнішніми API та можливість поступового розширення функцій без зміни базової структури системи.

3.2 Фізична модель даних

Фізична модель даних конкретизує логічну структуру у вигляді таблиць, типів полів, ключів, індексів і обмежень цілісності. Вона орієнтована на PostgreSQL із підтримкою просторових і часових даних.

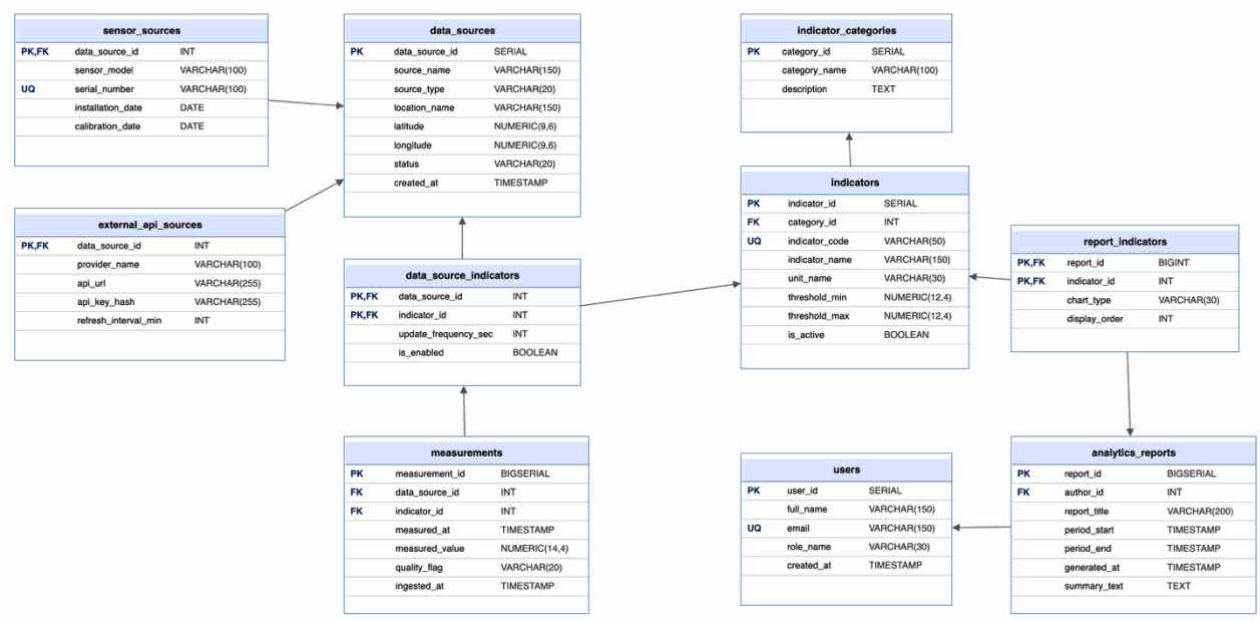

Рис. 3.1. Фізична модель даних вебсервісу екологічного моніторингу

До складу бази входять таблиці roles, users, monitoring_stations, sensors, parameter_types, measurements, indicator_norms, alerts, notifications, reports та audit_logs.

Таблиця roles зберігає ролі користувачів, а users - облікові записи, контактні дані, хеші паролів і статус активності. Зв’язок між ними реалізовано через зовнішній ключ role_id.

Таблиця monitoring_stations призначена для збереження пунктів екологічного моніторингу. У ній фіксуються назва, координати, тип території, статус і джерело даних. До кожного пункту може бути прив’язано один або кілька сенсорів, що зберігаються в таблиці sensors. Для сенсорів передбачено код, тип показника, одиницю вимірювання, дату калібрування та час останнього з’єднання із системою.

Таблиця monitoring_stations містить назву пункту, координати, тип території, статус і опис джерела. Для просторових операцій використовується поле geometry або geography з індексом PostGIS.

Таблиця monitoring_sessions використовується для збереження періодів спостереження. Вона містить час початку та завершення, кількість вимірювань, середні значення й статус періоду. За потреби сеанс пов’язується з типом показника з таблиці parameter_types. Таблиця calibration_records призначена для обліку

калібрування сенсорів: типу перевірки, дати, відповідального користувача, наступної дати контролю та результату.

Таблиця sensors описує вимірювальні канали, прив’язані до пунктів моніторингу. Вона містить тип сенсора, код показника, одиницю вимірювання, дату калібрування та статус активності.

Центральною таблицею є measurements, у якій зберігаються фактичні значення показників із часовою міткою, посиланням на сенсор, параметром, значенням, ознакою якості та службовим payload.

Таблиця 3.2

Характеристика таблиць фізичної моделі даних

№	Назва таблиці	Призначення таблиці	Первинний ключ	Зовнішні ключі
1	roles	Зберігання ролей користувачів системи: адміністратор, оператор, еколог-лаборант, користувач	role_id	-
2	users	Зберігання облікових записів користувачів, їх контактних даних, паролів і статусу активності	user_id	role_id
3	monitoring_stations	Зберігання пунктів моніторингу: назва, координати, тип території, статус	vehicle_id	-
4	sensors	Зберігання сенсорів і джерел даних, прив’язаних до пунктів моніторингу	device_id	vehicle_id
5	parameter_types	Зберігання типів показників, одиниць вимірювання та описів параметрів	driver_id	-
6	measurements	Зберігання екологічних вимірювань: час, координати, показник, значення, якість даних	packet_id	device_id, vehicle_id
7	monitoring_sessions	Зберігання періодів спостереження з часом початку, завершення та агрегованими значеннями	trip_id	vehicle_id, device_id, driver_id
8	indicator_norms	Зберігання нормативних меж і правил контролю екологічних показників	rule_id	-

Продовження таблиці 3.2

9	environmental_alerts	Зберігання попереджень, що виникають у разі перевищення нормативів	alert_id	rule_id, vehicle_id, device_id, packet_id
10	notifications	Зберігання повідомлень, які надсилаються користувачам після виникнення попередженьи	notification_id	alert_id, user_id
11	calibration_records	Зберігання записів про калібрування сенсорів і перевірку якості даних	maintenance_id	vehicle_id

Нормативні межі зберігаються в indicator_norms. Під час аналізу вони використовуються для створення записів alerts, якщо виміряне значення перевищує допустимий рівень або має аномальну динаміку.

Таблиці notifications, reports та audit_logs забезпечують інформування користувачів, формування звітних матеріалів і відстеження службових дій у системі.

Для прискорення запитів передбачено індекси за station_id, sensor_id, parameter_id, measured_at, а також просторовий індекс для координат пунктів моніторингу.

Запропонована фізична модель забезпечує цілісність, масштабованість і придатність до аналітичних запитів, необхідних для побудови графіків, карт і звітів вебсервісу.

3.3 Алгоритмізація програмних модулів вебсервісу

Алгоритмізація програмних модулів вебсервісу екологічного моніторингу виконується для опису послідовності дій, які забезпечують введення, перевірку, збереження, аналіз, візуалізацію та формування звітів за екологічними показниками. Побудовані блок-схеми уточнюють логіку роботи основних модулів системи та показують порядок переходу від первинних даних до аналітичного результату.

Перший алгоритм описує процес введення або завантаження екологічних показників до вебсервісу (рис. 3.2).

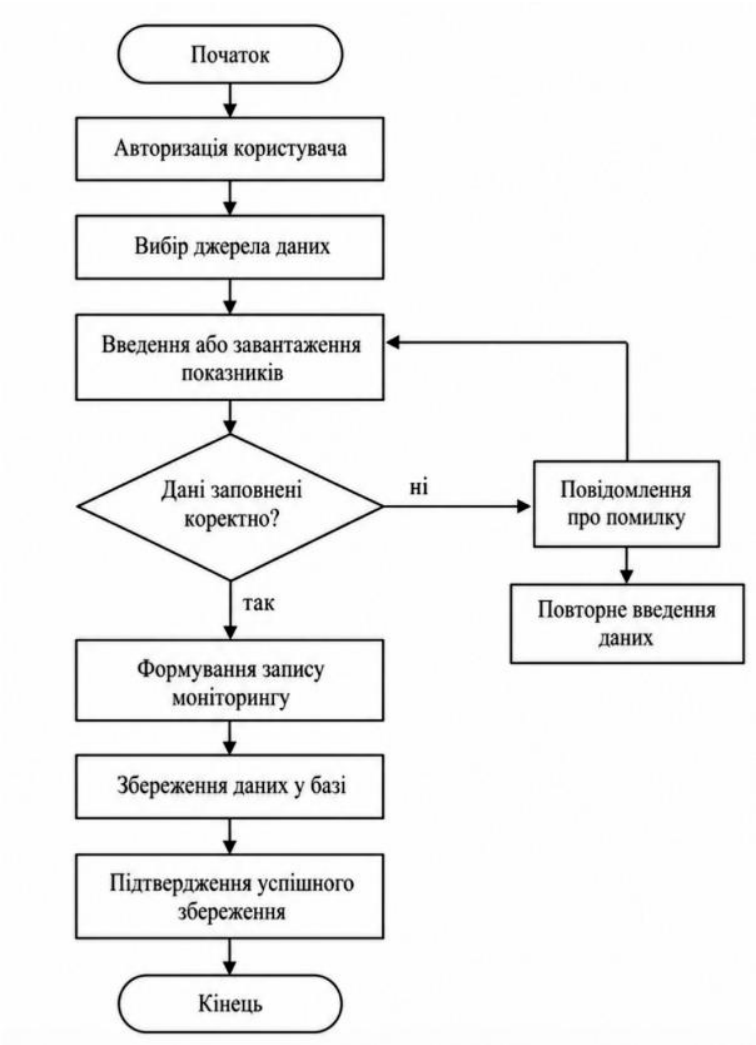

Рис. 3.2. Алгоритм введення та збереження екологічних показників

Робота алгоритму починається з авторизації користувача, після чого він обирає джерело даних. Показники можуть бути введені вручну або завантажені з файлу. Далі система перевіряє коректність заповнення обов’язкових полів. Якщо виявлено помилку, користувач отримує повідомлення та повторно вводить дані. Якщо дані заповнені правильно, формується запис моніторингу, який зберігається у базі даних, після чого система підтверджує успішне збереження.

Другий алгоритм деталізує процес перевірки та нормалізації отриманого запису моніторингу (рис. 3.3).

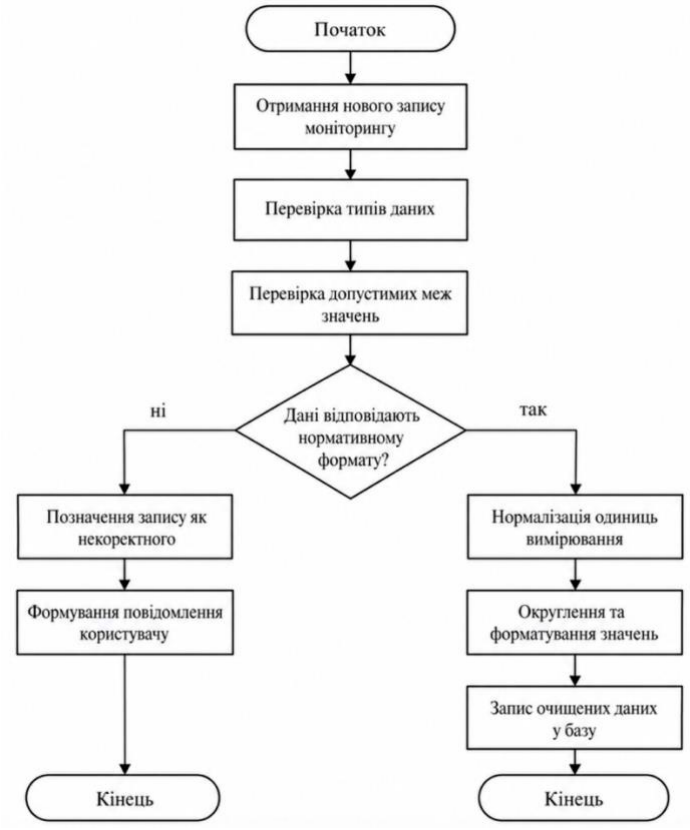

Рис. 3.3. Алгоритм перевірки та нормалізації екологічних даних

На цьому етапі система отримує новий запис моніторингу та перевіряє типи даних, допустимі межі значень і відповідність встановленому формату. Якщо запис не відповідає вимогам, він позначається як некоректний, а користувачу формується повідомлення. Якщо дані є коректними, виконується нормалізація одиниць вимірювання, округлення та форматування значень. Після цього очищені дані записуються до бази.

Третій алгоритм описує аналітичну обробку екологічних показників (рис. 3.4).

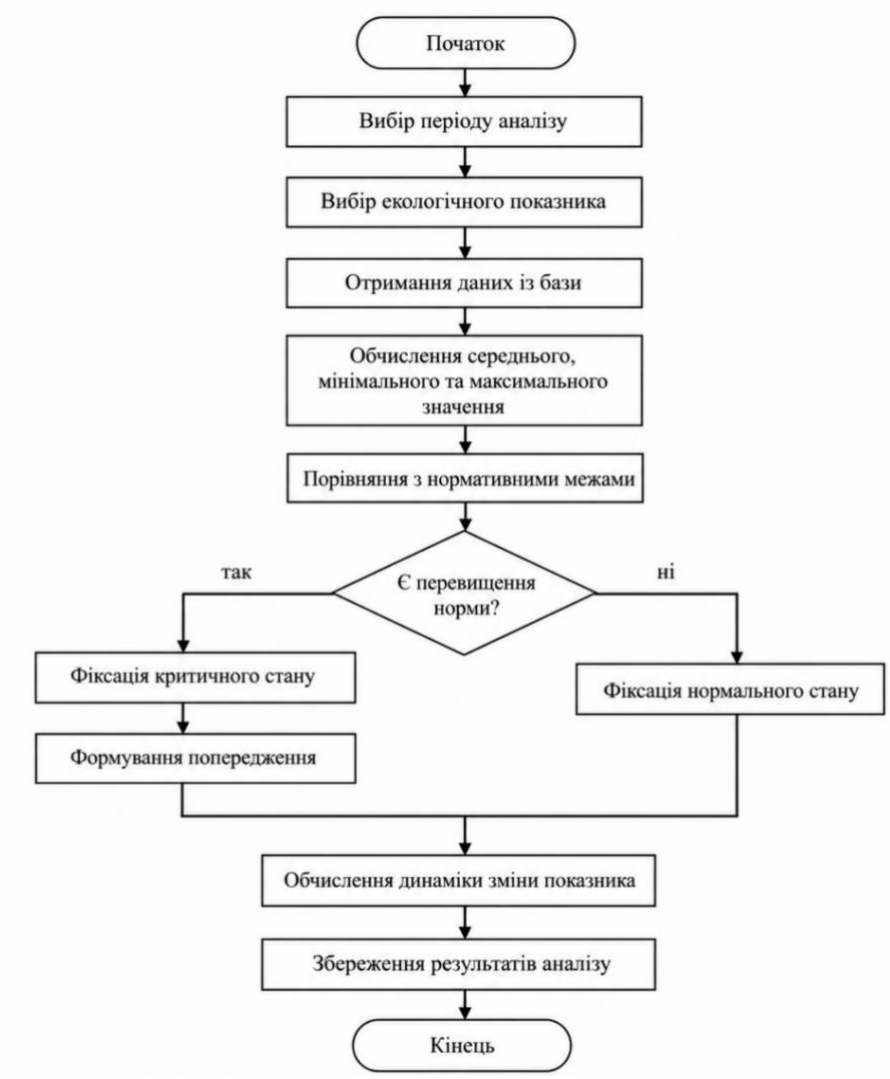

Рис. 3.4. Алгоритм аналізу екологічних показників

Спочатку користувач або система обирає період аналізу та потрібний екологічний показник. Після цього вебсервіс отримує відповідні дані з бази та обчислює середнє, мінімальне й максимальне значення за вибраний проміжок часу. Далі результати порівнюються з нормативними межами. У разі перевищення норми фіксується критичний стан і формується попередження. Якщо перевищень немає, система фіксує нормальний стан показника. Завершальним етапом є обчислення динаміки зміни показника та збереження результатів аналізу.

Четвертий алгоритм визначає порядок побудови графіків, діаграм або інших засобів візуалізації екологічних даних (рис. 3.5).

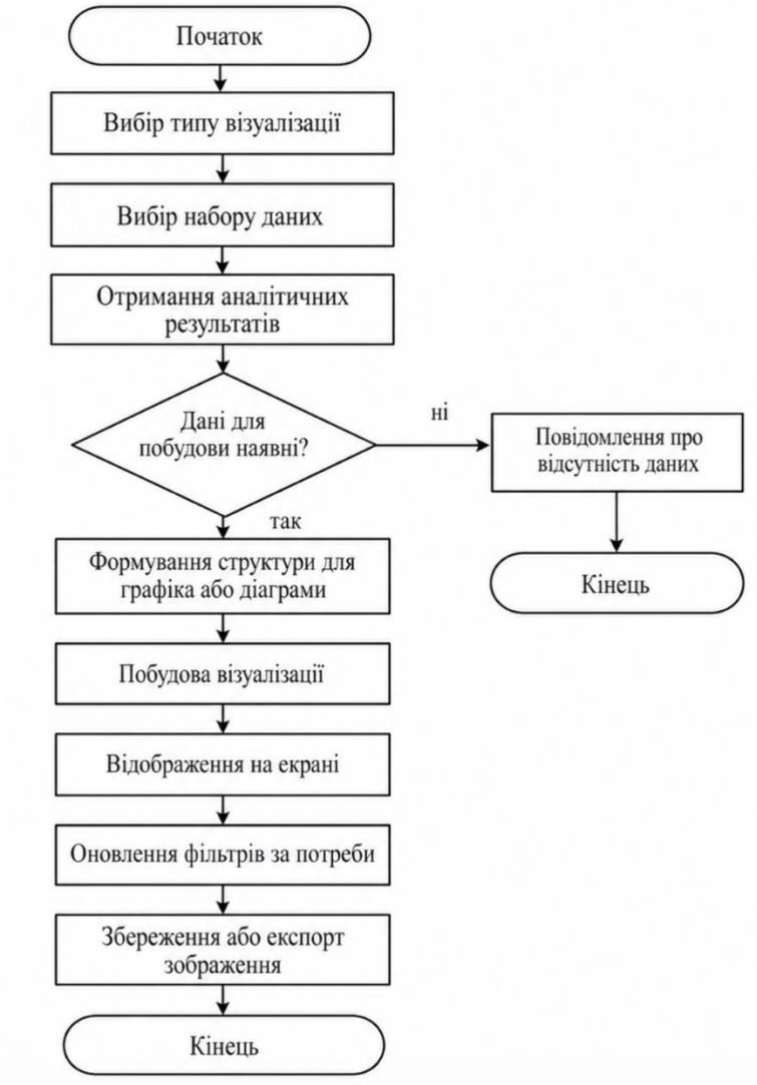

Рис. 3.5. Алгоритм побудови візуалізації екологічних даних

Процес починається з вибору типу візуалізації та набору даних. Далі система отримує аналітичні результати й перевіряє, чи достатньо даних для побудови графіка або діаграми. Якщо дані відсутні, користувачу виводиться повідомлення про неможливість побудови візуалізації. Якщо дані наявні, формується структура для графіка, виконується побудова візуалізації та її відображення на екрані. За потреби користувач може оновити фільтри, змінити параметри відображення або експортувати результат.

П’ятий алгоритм описує формування аналітичного звіту за результатами екологічного моніторингу (рис. 3.6).

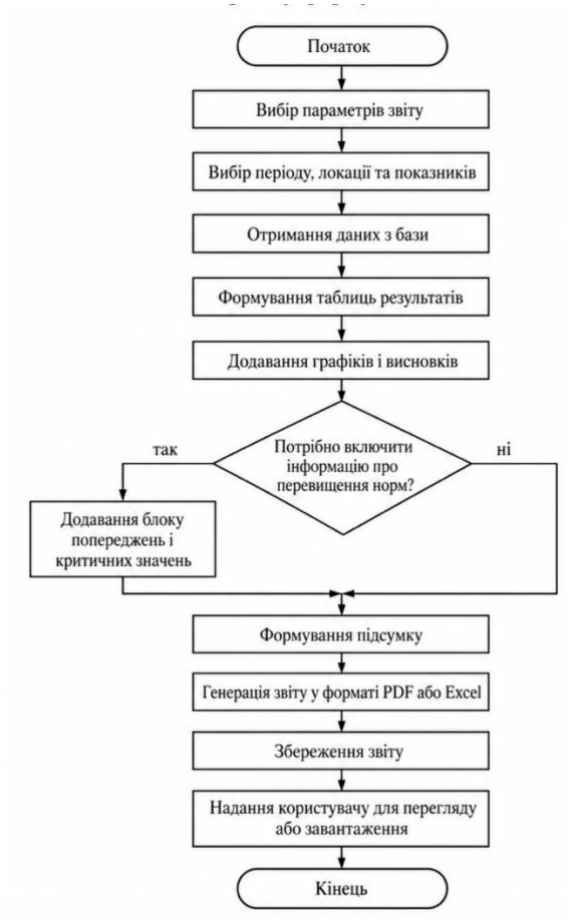

Рис. 3.6. Алгоритм формування аналітичного звіту

На початку користувач обирає параметри звіту: період, локацію та перелік екологічних показників. Далі система отримує дані з бази, формує таблиці результатів, додає графіки та текстові висновки. Після цього перевіряється, чи потрібно включити інформацію про перевищення нормативів. Якщо така інформація потрібна, до звіту додається блок попереджень і критичних значень. Потім формується підсумок, генерується файл звіту у форматі PDF або Excel, зберігається у системі та надається користувачу для перегляду або завантаження.

Запропоновані алгоритми охоплюють ключові процеси вебсервісу: введення даних, перевірку коректності, нормалізацію, аналітичну обробку, візуалізацію та звітність. Їх використання дає змогу формалізувати роботу програмних модулів, зменшити кількість помилок під час оброблення екологічних показників і забезпечити послідовну логіку функціонування системи.

3.4 Висновки до третього розділу

У третьому розділі обґрунтовано технологічні засоби реалізації вебсервісу для аналізу екологічних показників. Запропоновано використання FastAPI, React, SQLLite, PostGIS, TimescaleDB, Leaflet і засобів контейнеризації.

Розроблено фізичну модель даних, яка визначає таблиці, первинні та зовнішні ключі, індекси, просторові поля й структуру часових рядів вимірювань.

Описано алгоритми основних програмних модулів: приймання вимірювання, завантаження дашборду, оброблення подій інтерфейсу, виконання API-запитів і транзакційне збереження даних.

Отримані рішення формують технічну основу для реалізації вебсервісу, здатного збирати, перевіряти, аналізувати та візуалізувати екологічні показники у вебсередовищі.

У третьому розділі обґрунтовано технологічні засоби реалізації вебсервісу для аналізу екологічних показників. Запропоновано використання FastAPI, React, PostgreSQL, PostGIS, TimescaleDB, Leaflet і засобів контейнеризації.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ СИСТЕМИ

4.1 План тестування програмних модулів та методика оцінювання результатів

Тестування програмних модулів вебсервісу проводиться для перевірки коректності приймання екологічних даних, роботи API, збереження вимірювань, побудови графіків, формування картографічних шарів і створення попереджень.

План тестування охоплює функціональні, інтеграційні, навантажувальні та інтерфейсні перевірки. Особлива увага приділяється правильності валідації JSON-пакетів, швидкодії серверної логіки, цілісності бази даних і стабільності оновлення дашборду.

Методика оцінювання результатів базується на порівнянні фактичної поведінки системи з очікуваними результатами. Для цього використовуються тестові набори вимірювань, журнали запитів, часові мітки та контрольні записи у базі даних.

Таблиця 4.1

План тестування програмних модулів вебсервісу моніторингу

№	Тестовий сценарій	Модуль	Очікуваний результат
1	Передавання екологічних вимірювань із сенсорного джерела	Модуль збору даних	Дані успішно отримані сервером
2	Зчитування показників сенсора	IoT / MQTT / REST модуль	Параметри коректно обробляються
3	Запис показників у PostgreSQL/PostGIS	Сервіс збереження даних	Дані збережені без помилок
4	Перевірка нормативних меж	Модуль аналізу	Виявлення критичних відхилень
5	Формування екологічного попередження	Аналітичний модуль	Подія створюється коректно
6	Надсилання сповіщення	Сервіс повідомлень	Повідомлення доставлене користувачу
7	Перегляд історії екологічних показників	API / PostgreSQL/PostGIS	Дані відображаються коректно

Ключовими критеріями є час відповіді API, відсоток успішно оброблених вимірювань, відсутність втрат пакетів, правильність створення подій, коректність відображення графіків і карт.

- час реакції серверної частини;
- відсоток успішно оброблених повідомлень;
- коректність збереження екологічних вимірювань у PostgreSQL/PostGIS;
- стабільність роботи FastAPI при навантаженні;
- правильність формування діагностичних подій;
- відсутність втрат пакетів екологічних вимірювань;
- коректність відображення даних у клієнтському вебсервісу.

Під час навантажувального тестування перевіряється здатність вебсервісу обробляти інтенсивний потік екологічних вимірювань від кількох пунктів моніторингу без критичних затримок або помилок запису.

Комплексне тестування дозволяє встановити рівень готовності системи до експериментального використання та виявити модулі, які потребують оптимізації перед розгортанням.

Тестування програмних модулів вебсервісу проводиться для перевірки коректності приймання екологічних даних, роботи API, збереження вимірювань, побудови графіків, формування картографічних шарів і створення попереджень.

4.2 Тестування вебсервісу аналізу екологічних показників

Тестування вебсервісу екологічного моніторингу виконано для перевірки працездатності основних функціональних модулів: авторизації, реєстрації користувачів, перегляду дашборду, керування пунктами моніторингу, внесення екологічних вимірювань, імпорту CSV/JSON, аналітики, візуалізації та роботи REST API. Перевірка проводилася в локальному середовищі запуску Node.js із використанням підготовленої бази даних і тестових екологічних показників.

На першому етапі перевірено модуль входу в систему. Користувач вводить email і пароль, після чого система перевіряє облікові дані та відкриває робочу панель. Окремо перевірено реакцію на незаповнені поля, неправильний пароль і некоректний формат email. Інтерфейс входу наведено на рис. 4.1.

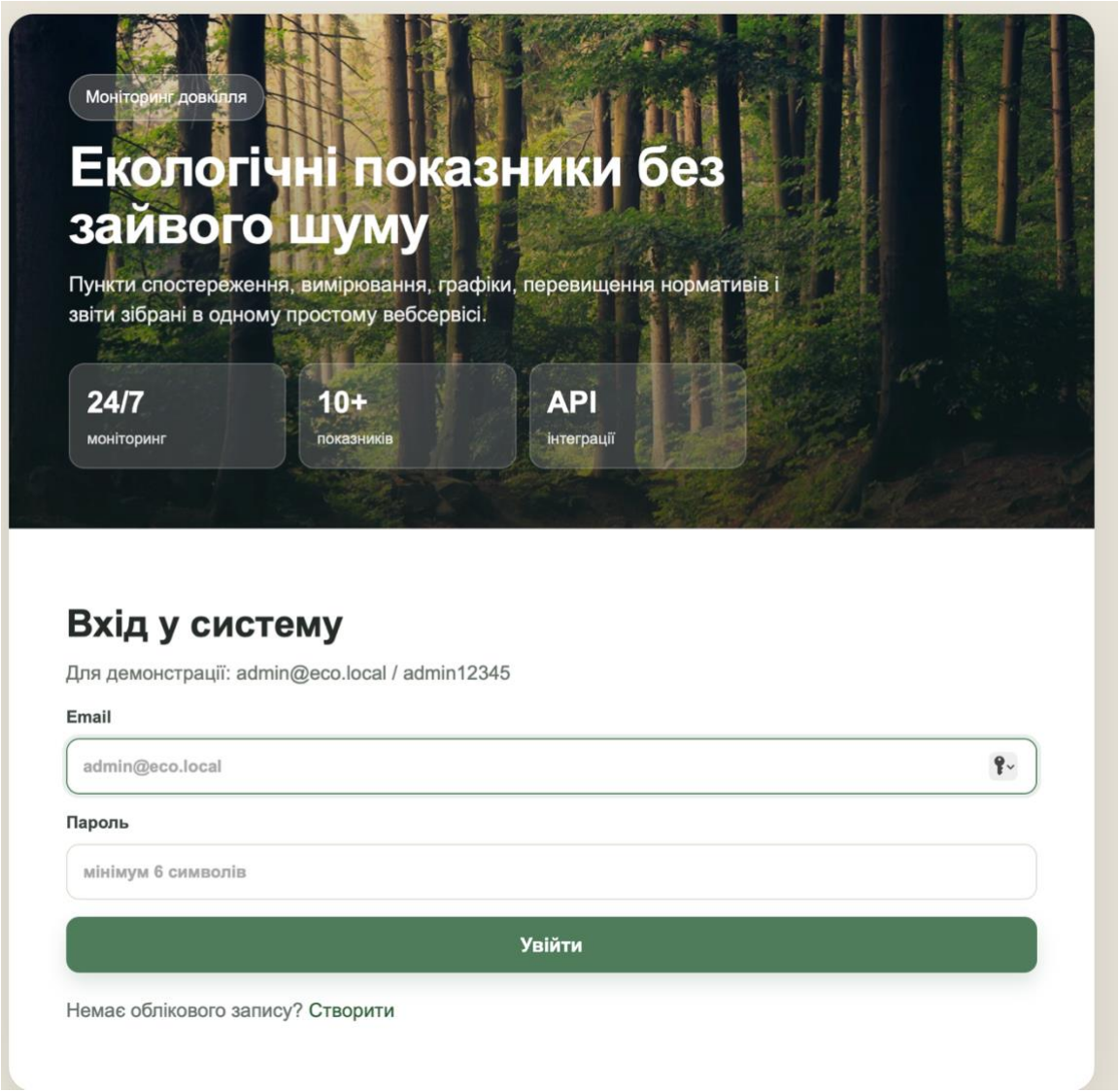

Рис. 4.1. Вікно авторизації користувача

Далі протестовано модуль реєстрації. Новий користувач вводить ПІБ, email і пароль, після чого система створює обліковий запис із роллю аналітика. Перевірено обмеження на мінімальну довжину пароля, обов’язковість заповнення полів і недопущення повторної реєстрації з однаковою електронною адресою. Вікно реєстрації користувача подано на рис. 4.2.

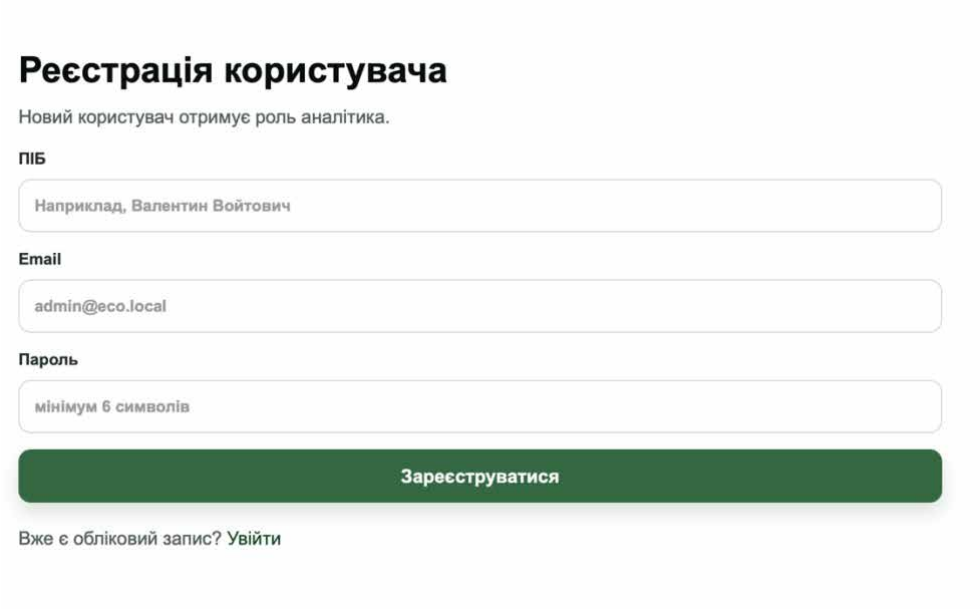

Рис. 4.2. Вікно реєстрації користувача

Після успішного входу перевірено головну панель вебсервісу. Дашборд відображає кількість пунктів моніторингу, загальну кількість вимірювань, кількість сповіщень і розрахований індекс якості. Також перевірено коректність відображення графіка динаміки PM2.5, умовної карти пунктів і розподілу останніх значень за показниками. Результат роботи головної панелі наведено на рис. 4.3.

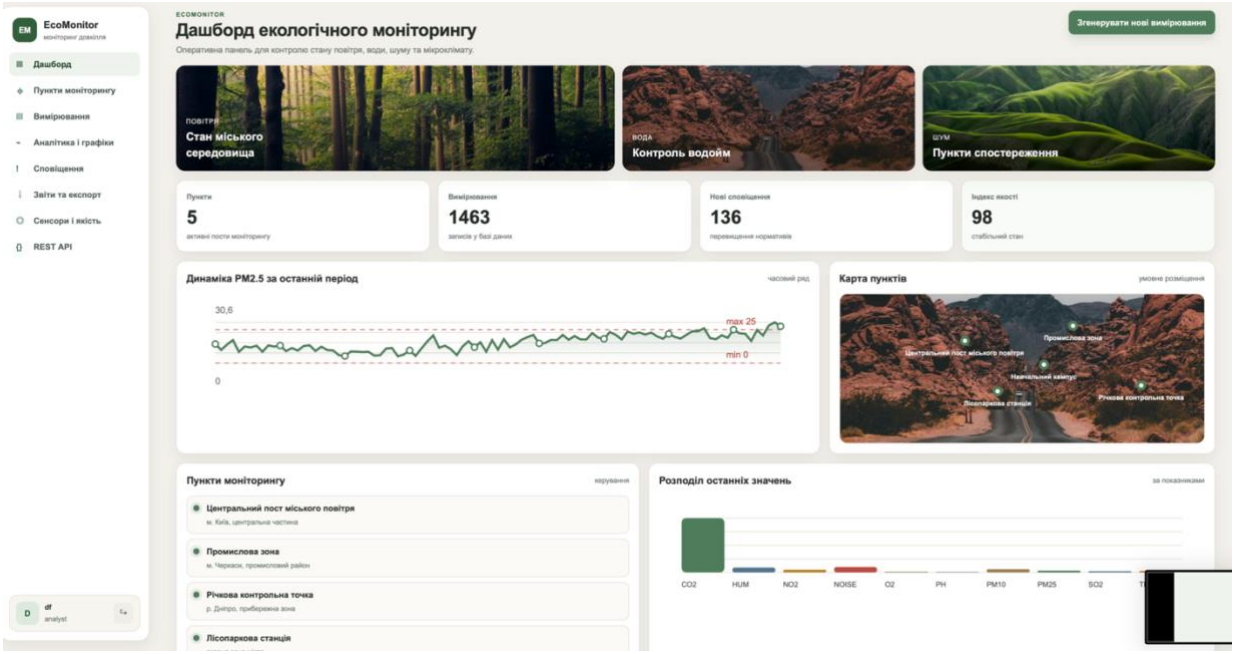

Рис. 4.3. Дашборд екологічного моніторингу

Окремо протестовано модуль керування пунктами моніторингу. Перевірено створення нового пункту, введення назви, типу, локації, координат і опису. Після

збереження пункт відображається у загальному списку з відповідним статусом. Також перевірено коректність відображення вже наявних пунктів, зокрема навчального кампусу, лісопаркової станції, річкової контрольної точки, промислової зони та центрального поста міського повітря. Результат тестування цього модуля показано на рис. 4.4.

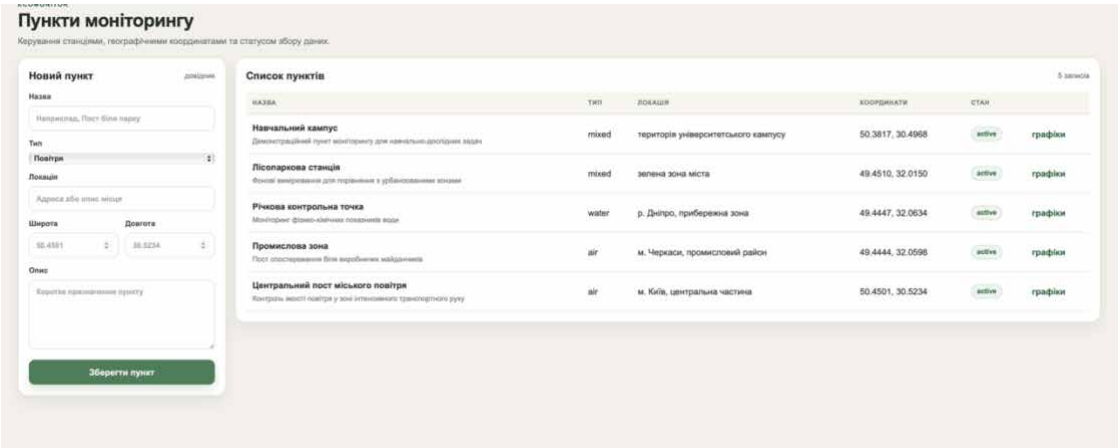

Рис. 4.4. Модуль керування пунктами моніторингу

Наступним етапом було тестування модуля екологічних вимірювань. Система підтримує ручне додавання показника, вибір пункту моніторингу, вибір типу показника, зазначення сеансу спостереження, значення, дати й джерела даних. Також перевірено імпорт даних у форматах CSV і JSON. Після збереження нові записи відображаються у таблиці останніх вимірювань із позначенням джерела та якості даних. Інтерфейс цього модуля наведено на рис. 4.5.

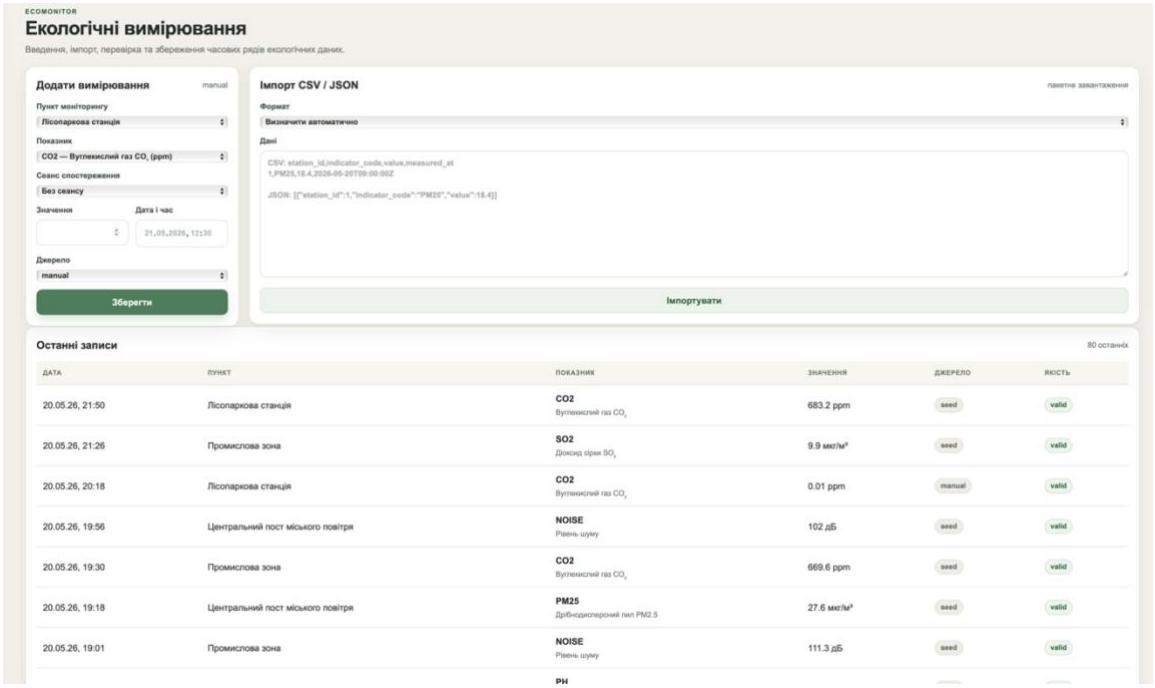

Рис. 4.5. Модуль введення та імпорту екологічних вимірювань

Під час перевірки аналітичного модуля протестовано вибір пункту моніторингу, екологічного показника та періоду аналізу. Система будує часовий ряд, відображає нормативні межі, розраховує кількість записів, мінімальне, середнє й максимальне значення, а також індекс якості. Додатково перевірено побудову діаграми середніх значень і структури сповіщень за рівнем критичності. Результат тестування аналітики та візуалізації наведено на рис. 4.6.

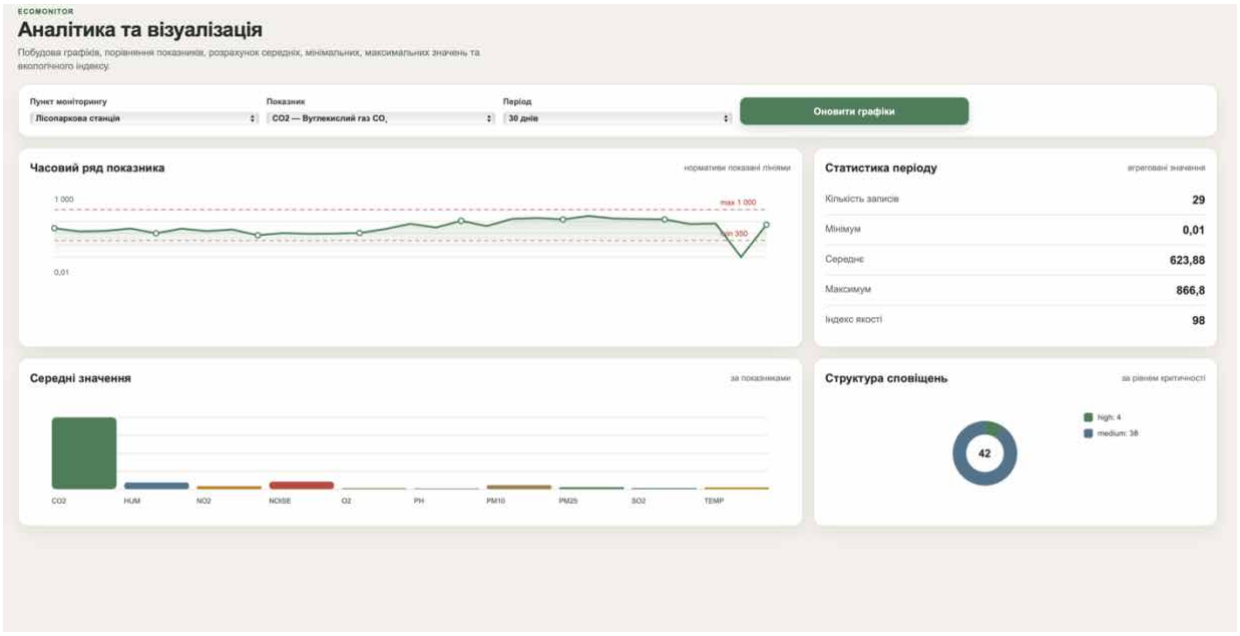

Рис. 4.6. Модуль аналітики та візуалізації екологічних показників

Також протестовано REST API для зовнішніх джерел даних. Перевірено можливість передавання нового вимірювання через POST-запит до маршруту /api/v1/measurements із використанням ключа доступу. Окремо перевірено отримання останніх записів через GET-запит із фільтрацією за пунктом моніторингу та кодом показника. Інтерфейс із прикладами API-запитів наведено на рис. 4.7.

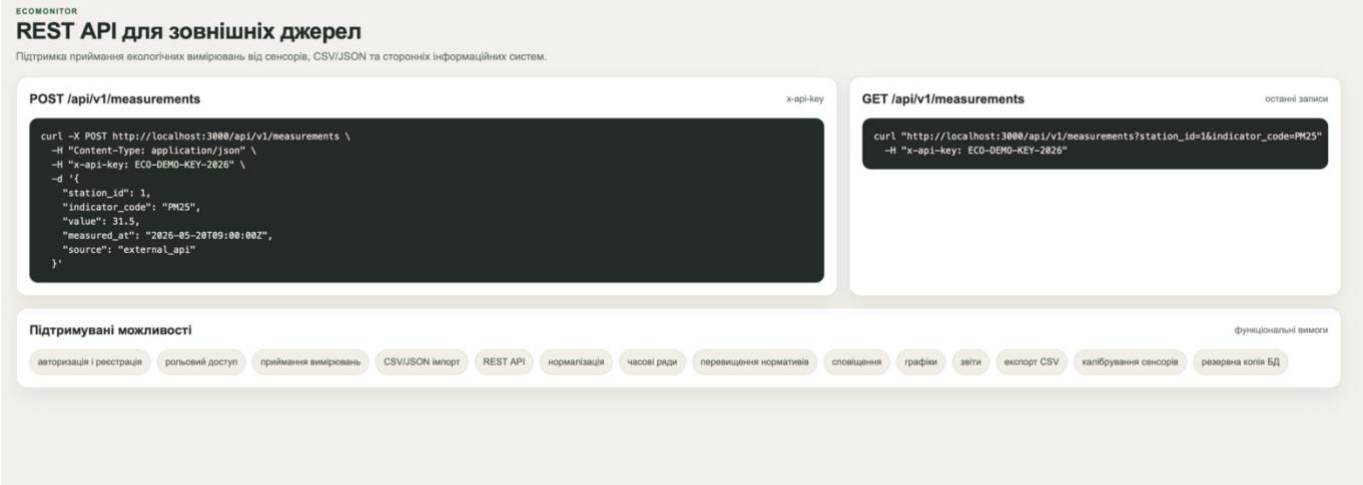

Рис. 4.7. Перевірка REST API для зовнішніх джерел даних

Узагальнені результати тестування основних модулів системи наведено в табл. 4.2.

Таблиця 4.2

Результати тестування основних модулів вебсервісу

Модуль	Перевірена дія	Очікуваний результат	Результат
Авторизація	Вхід за коректним email і паролем	Користувач переходить до дашборду	Виконано
Авторизація	Введення неправильного пароля	Система не дозволяє вхід і повідомляє про помилку	Виконано
Реєстрація	Створення нового користувача	Обліковий запис створюється, користувач отримує роль аналітика	Виконано
Реєстрація	Повторний email	Система не допускає дублювання облікового запису	Виконано
Дашборд	Завантаження зведених показників	Відображаються пункти, вимірювання, сповіщення та індекс якості	Виконано
Пункти моніторингу	Додавання нового пункту	Новий пункт зберігається та з’являється у списку	Виконано
Вимірювання	Ручне додавання показника	Запис зберігається у базі та відображається в таблиці	Виконано

Продовження таблиці 4.2

Імпорт даних	Завантаження CSV/JSON	Дані розпізнаються, перевіряються та додаються до списку вимірювань	Виконано
Аналітика	Побудова графіка за період	Система формує часовий ряд і статистику	Виконано
Візуалізація	Відображення діаграм і структури сповіщень	Дані подаються у графічному вигляді	Виконано
REST API	POST-запит на додавання вимірювання	API приймає пакет і створює новий запис	Виконано
REST API	GET-запит на отримання вимірювань	API повертає структуровану відповідь за заданими параметрами	Виконано

За результатами тестування встановлено, що вебсервіс коректно виконує основні функції, передбачені функціональними вимогами. Авторизація та реєстрація забезпечують контрольований доступ до системи, модуль пунктів моніторингу дає змогу вести довідник станцій, а модуль вимірювань підтримує як ручне введення, так і імпорт екологічних даних. Аналітичний блок правильно формує статистику, графіки та візуальні показники, а REST API забезпечує можливість інтеграції із зовнішніми джерелами.

Отже, тестування підтвердило працездатність розробленого вебсервісу екологічного моніторингу. Система забезпечує повний цикл роботи з екологічними показниками: від авторизації користувача та введення даних до їх збереження, аналізу, візуалізації та отримання через API.

4.3 Результати тестування та аналіз ефективності вебсервісу

За результатами тестування встановлено, що вебсервіс екологічного моніторингу коректно виконує основні функції, передбачені технічним завданням. Система стабільно приймає екологічні вимірювання, перевіряє їхню структуру, зберігає записи у базі даних і оновлює інформацію в інтерфейсі користувача.

Під час перевірки розглядалися два основні сценарії роботи: штатний режим моніторингу та режим комплексного критичного стану. У першому випадку система

обробляла звичайні вимірювання без перевищення нормативів. У другому випадку перевірялася реакція на критичні значення показників, формування попереджень і передавання інформації до журналу подій. Основні кількісні результати тестування наведено в табл. 4.2.

Таблиця 4.2

Загалом тестування підтвердило, що розроблена архітектура придатна для подальшого розгортання та може бути розширена додатковими джерелами даних і новими аналітичними правилами.

Показник	Значення	Очікувана вимога	Висновок
Середній час обробки вимірювання	92 мс	≤ 200 мс	виконується
p95 затримки	138 мс	≤ 200 мс	виконується
p99 затримки	181 мс	≤ 250 мс	виконується
Частота обробки пакетів вимірювань	68 msg/s	≥ 30 msg/s	виконується
Успішне збереження екологічних даних	99.8 %	≥ 99 %	виконується
Втрата пакетів вимірювань	0.3 %	≤ 1 %	виконується
Формування екологічних попереджень	100 %	≥ 99 %	виконується
Надсилання сповіщень	99.6 %	≥ 99 %	виконується

Дані табл. 4.2 показують, що середній час обробки одного вимірювання становить 92 мс, що менше від допустимого значення 200 мс. Значення p95 і p99 також не перевищують встановлені межі, тому затримка оброблення даних не створює помітних перешкод для оновлення дашборду та перегляду результатів моніторингу.

Частота обробки пакетів становила 68 повідомлень за секунду при мінімальній вимозі 30 повідомлень за секунду. Це підтверджує, що система має достатній запас продуктивності для роботи з кількома пунктами моніторингу та може обробляти як ручні записи, так і пакети, отримані через API або імпорт CSV/JSON.

Перевірка цілісності даних показала, що коректні записи зберігалися без втрат, а некоректні пакети відхилялися із фіксацією причини помилки в журналі. Показник

успішного збереження становив 99,8 %, а втрата пакетів не перевищила 0,3 %, що відповідає заданим вимогам.

Аналітичний модуль правильно визначав перевищення граничних значень, формував події відповідного рівня критичності та передавав їх до підсистеми сповіщень. У контрольних сценаріях помилкових спрацювань не виявлено. Надсилання повідомлень виконувалося стабільно, а показник успішної доставки становив 99,6 %.

Інтерфейс користувача забезпечив коректний перегляд карти пунктів моніторингу, графіків, таблиць вимірювань, статистичних показників і журналу подій. Фільтрація за періодом, пунктом спостереження та типом показника працювала правильно й не призводила до втрати або некоректного відображення даних.

Отже, результати тестування підтвердили працездатність і достатню ефективність розробленого вебсервісу. Система відповідає основним вимогам щодо швидкодії, надійності збереження даних, виявлення перевищень, формування сповіщень і зручності роботи користувача. Отримані результати дають підстави вважати архітектуру придатною для подальшого розгортання, підключення нових джерел даних і розширення аналітичних правил.

4.4 Висновки до четвертого розділу

У четвертому розділі виконано перевірку працездатності вебсервісу екологічного моніторингу. Тестування охопило основні функціональні модулі системи: авторизацію та реєстрацію користувачів, приймання екологічних вимірювань, роботу REST API, збереження даних у базі, побудову графіків, відображення пунктів моніторингу, формування сповіщень і підготовку звітів.

Для оцінювання системи було сформовано план тестування, який включав функціональні, інтеграційні та навантажувальні сценарії. Такий підхід дав змогу перевірити не лише окремі можливості вебсервісу, а й узгоджену роботу його

компонентів: клієнтського інтерфейсу, серверної частини, бази даних, аналітичного модуля та зовнішнього API.

Результати тестування підтвердили, що система коректно приймає, перевіряє, зберігає та відображає екологічні показники. Аналітичний модуль правильно визначає перевищення нормативних значень, формує події відповідного рівня критичності та забезпечує підготовку даних для графіків і звітів. Інтерфейс вебсервісу забезпечує зручний перегляд таблиць, картографічних елементів, динаміки показників і результатів аналізу.

Проведена перевірка показала, що розроблена програмна реалізація є працездатною та відповідає основним функціональним і технічним вимогам. Вебсервіс може бути використаний як основа для подальшого розширення, зокрема підключення нових джерел екологічних даних, додавання аналітичних правил, удосконалення візуалізації та розгортання системи в реальному середовищі моніторингу.

ВИСНОВКИ

У кваліфікаційній роботі вирішено практичне завдання проєктування програмного забезпечення вебсервісу для аналізу екологічних показників та візуалізації даних моніторингу. Актуальність роботи обґрунтовано потребою централізованого збирання, перевірки, збереження й подання екологічної інформації у зручному для користувача форматі.

У першому розділі виконано аналіз предметної області екологічного моніторингу. Розглянуто джерела екологічних даних, особливості роботи з часовими рядами, просторову прив’язку вимірювань, роль сенсорів, зовнішніх API та вебінтерфейсу в системах моніторингу довкілля. Також проаналізовано існуючі рішення, зокрема EcoCity, SaveEcoBot, OpenAQ, IQAir AirVisual та Copernicus CAMS. Визначено, що наявні платформи добре виконують окремі функції, однак не завжди забезпечують гнучке поєднання власних джерел даних, аналітичних правил, візуалізації, звітності та керування доступом.

На основі аналізу предметної області сформовано функціональні, нефункціональні, технічні та безпекові вимоги до вебсервісу. Визначено, що система має підтримувати приймання екологічних вимірювань через API, ручне введення та імпорт CSV/JSON, валідацію і нормалізацію показників, збереження даних у базі, перевірку нормативних меж, побудову графіків, відображення пунктів моніторингу, формування звітів і розмежування прав користувачів.

У другому розділі розроблено логічну модель даних, діаграму класів, кооперації, компонентну та пакетну структуру програмного забезпечення. Побудовані моделі описують основні сутності системи: користувачів, ролі, джерела даних, пункти спостереження, екологічні показники, записи вимірювань, візуалізації та аналітичні звіти. Логічна модель даних побудована з урахуванням нормалізації, що зменшує дублювання інформації та забезпечує цілісність зв’язків між об’єктами.

У третьому розділі обґрунтовано вибір технологічних засобів реалізації, спроєктовано фізичну модель бази даних та описано алгоритми роботи основних програмних модулів. Для реалізації вебсервісу обрано підхід із серверним API,

клієнтським дашбордом, базою даних і модулями оброблення екологічних показників. Алгоритми охоплюють введення та збереження даних, перевірку і нормалізацію вимірювань, аналітичну обробку, побудову візуалізацій та формування звітів.

У четвертому розділі проведено тестування розробленого вебсервісу. Перевірено авторизацію і реєстрацію користувачів, роботу дашборду, керування пунктами моніторингу, ручне додавання вимірювань, імпорт CSV/JSON, функціонування REST API, побудову графіків, формування сповіщень і відображення аналітичних результатів. Результати тестування підтвердили коректність приймання, збереження, оброблення та візуалізації екологічних даних.

У результаті досягнуто поставленої мети: спроєктовано програмне забезпечення вебсервісу, яке забезпечує аналіз екологічних показників, візуалізацію моніторингових даних, формування подій перевищення нормативів і підтримку звітності. Розроблені моделі, алгоритми та програмні рішення можуть бути використані як основа для подальшого впровадження системи, підключення реальних сенсорних джерел, розширення аналітичних правил і розвитку функцій екологічного моніторингу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про охорону навколишнього природного середовища : Закон України від 25.06.1991 № 1264-XII. URL: https://zakon.rada.gov.ua/go/1264-12 (дата звернення: 21.05.2026).

2. Про охорону атмосферного повітря : Закон України від 16.10.1992 № 2707-XII. URL: https://zakon.rada.gov.ua/go/2707-12 (дата звернення: 21.05.2026).

3. Деякі питання здійснення державного моніторингу в галузі охорони атмосферного повітря : Постанова Кабінету Міністрів України від 14.08.2019 № 827. URL: https://zakon.rada.gov.ua/go/827-2019-%D0%BF (дата звернення: 21.05.2026).

4. Міністерство захисту довкілля та природних ресурсів України. Екологічний моніторинг. URL: https://mepr.gov.ua/diyalnist/napryamky/ekologichnyj-monitoryng/ (дата звернення: 21.05.2026).

5. Міністерство захисту довкілля та природних ресурсів України. Екологічний моніторинг довкілля. URL: https://mepr.gov.ua/diyalnist/napryamky/ekologichnyj-monitoryng/ekologichnyj-monitoryng-dovkillya/ (дата звернення: 21.05.2026).

6. WHO global air quality guidelines: particulate matter PM2.5 and PM10, ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide. Geneva : World Health Organization, 2021. URL: https://www.who.int/publications/i/item/9789240034228 (дата звернення: 21.05.2026).

7. Air Quality Download Service API. European Environment Agency. URL: https://www.eea.europa.eu/en/datahub/datahubitem-view/778ef9f5-6293-4846-badd-56a29c70880d (дата звернення: 21.05.2026).

8. Copernicus Atmosphere Monitoring Service. Data. URL: https://atmosphere.copernicus.eu/data (дата звернення: 21.05.2026).

9. CAMS global atmospheric composition forecasts. Atmosphere Data Store. URL: https://ads.atmosphere.copernicus.eu/datasets/cams-global-atmospheric-composition-forecasts (дата звернення: 21.05.2026).

10. Copernicus Atmosphere Monitoring Service. Documentation. URL: https://atmosphere.copernicus.eu/documentation (дата звернення: 21.05.2026).

11. EcoCity: Reborn. Якість повітря в реальному часі. URL: https://reborn.eco-city.org.ua/ (дата звернення: 21.05.2026).

12. Eco-City. Громадський моніторинг стану якості повітря. URL: https://eco-city.org.ua/faq (дата звернення: 21.05.2026).

13. SaveEcoBot. The first environmental system in Ukraine. URL: https://www.saveecobot.com/en (дата звернення: 21.05.2026).

14. SaveEcoBot. Air quality in Ukraine online: Air quality monitoring map. URL: https://www.saveecobot.com/en/maps (дата звернення: 21.05.2026).

15. OpenAQ Docs. OpenAQ API Documentation. URL: https://docs.openaq.org/ (дата звернення: 21.05.2026).

16. OpenAQ REST API. URL: https://api.openaq.org/ (дата звернення: 21.05.2026).

17. IQAir. AirVisual API Documentation. URL: https://api-docs.iqair.com/ (дата звернення: 21.05.2026).

18. IQAir. AirVisual API. URL: https://www.iqair.com/commercial-air-quality-monitors/api (дата звернення: 21.05.2026).

19. Shelestov A., Lavreniuk M., Kussul N., Novikov A., Skakun S. Air Quality Estimation in Ukraine Using SDG 11.6.2 Indicator. Remote Sensing. 2021. Vol. 13, No. 23. URL: https://www.mdpi.com/2072-4292/13/23/4769 (дата звернення: 21.05.2026).

20. Копняк В. Є., Мокін В. Б., Варчук І. В. Визначення середньорічного тренду показників якості атмосферного повітря регіону за даними громадського моніторингу.
URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/49186/24325.pdf (дата звернення: 21.05.2026).

21. Node.js. API Documentation. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 21.05.2026).

22. Node.js. Introduction to Node.js. URL: https://nodejs.org/learn/getting-started/introduction-to-nodejs (дата звернення: 21.05.2026).

23. Express.js. API Reference. URL: https://expressjs.com/en/api/ (дата звернення: 21.05.2026).

24. Express.js. Routing. URL: https://expressjs.com/en/guide/routing.html (дата звернення: 21.05.2026).

25. PostgreSQL. Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 21.05.2026).

26. PostgreSQL. JSON Functions and Operators. URL: https://www.postgresql.org/docs/current/functions-json.html(дата звернення: 21.05.2026).

27. SQLite Documentation. URL: https://sqlite.org/docs.html (дата звернення: 21.05.2026).

28. PostGIS Documentation. URL: https://postgis.net/documentation/ (дата звернення: 21.05.2026).

29. PostGIS Manual. URL: https://postgis.net/docs/ (дата звернення: 21.05.2026).

30. Leaflet. An open-source JavaScript library for mobile-friendly interactive maps. URL: https://leafletjs.com/ (дата звернення: 21.05.2026).

31. Leaflet API Reference. URL: https://leafletjs.com/reference.html (дата звернення: 21.05.2026).

32. Chart.js Documentation. URL: https://www.chartjs.org/docs/ (дата звернення: 21.05.2026).

33. MDN Web Docs. Fetch API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (дата звернення: 21.05.2026).

34. OpenAPI Specification. Version 3.1.0. URL: https://swagger.io/specification/ (дата звернення: 21.05.2026).

35. Bray T. The JavaScript Object Notation (JSON) Data Interchange Format : RFC 8259. Internet Engineering Task Force, 2017. URL: https://datatracker.ietf.org/doc/html/rfc8259 (дата звернення: 21.05.2026).

36. Shafranovich Y. Common Format and MIME Type for Comma-Separated Values (CSV) Files : RFC 4180. Internet Engineering Task Force, 2005. URL: https://www.ietf.org/rfc/rfc4180.txt (дата звернення: 21.05.2026).

37. OWASP API Security Top 10 2023. URL: https://owasp.org/API-Security/editions/2023/en/0x11-t10/ (дата звернення: 21.05.2026).

38. OWASP Application Security Verification Standard. URL: https://owasp.org/www-project-application-security-verification-standard/ (дата звернення: 21.05.2026).

39. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. URL: https://www.iso.org/standard/35733.html(дата звернення: 21.05.2026).

40. ISO 19115-1:2014. Geographic information — Metadata. URL: https://www.iso.org/standard/53798.html (дата звернення: 21.05.2026).

ДОДАТОК А

Фрагмент програмного коду створення бази даних вебсервісу екологічного моніторингу

```
-- schema.sql
-- Створення основних таблиць бази даних вебсервісу екологічного моніторингу

CREATE TABLE roles (
    role_id SERIAL PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL UNIQUE,
    description TEXT
);

CREATE TABLE users (
    user_id SERIAL PRIMARY KEY,
    role_id INTEGER NOT NULL,
    full_name VARCHAR(120) NOT NULL,
    email VARCHAR(120) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_users_roles
        FOREIGN KEY (role_id) REFERENCES roles(role_id)
        ON DELETE RESTRICT
);

CREATE TABLE monitoring_stations (
    station_id SERIAL PRIMARY KEY,
    station_code VARCHAR(40) NOT NULL UNIQUE,
    station_name VARCHAR(120) NOT NULL,
    region VARCHAR(120),
    locality VARCHAR(120),
    latitude NUMERIC(10, 7) NOT NULL,
    longitude NUMERIC(10, 7) NOT NULL,
    area_type VARCHAR(60),
    source_description TEXT,
    status VARCHAR(30) DEFAULT 'active',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE parameter_types (
    parameter_id SERIAL PRIMARY KEY,
    parameter_code VARCHAR(40) NOT NULL UNIQUE,
    parameter_name VARCHAR(120) NOT NULL,
    unit_measure VARCHAR(30) NOT NULL,
    parameter_group VARCHAR(60),
    description TEXT
);

CREATE TABLE sensors (
    sensor_id SERIAL PRIMARY KEY,
    station_id INTEGER NOT NULL,
    parameter_id INTEGER NOT NULL,
    sensor_code VARCHAR(60) NOT NULL UNIQUE,
    device_model VARCHAR(100),
    sensor_type VARCHAR(80),
    calibration_date DATE,
    last_connection TIMESTAMP,
    status VARCHAR(30) DEFAULT 'active',
    CONSTRAINT fk_sensors_stations
        FOREIGN KEY (station_id) REFERENCES monitoring_stations(station_id)
        ON DELETE CASCADE,
```

```
    CONSTRAINT fk_sensors_parameters
        FOREIGN KEY (parameter_id) REFERENCES parameter_types(parameter_id)
        ON DELETE RESTRICT
);

CREATE TABLE measurements (
    measurement_id BIGSERIAL PRIMARY KEY,
    station_id INTEGER NOT NULL,
    sensor_id INTEGER,
    parameter_id INTEGER NOT NULL,
    measured_at TIMESTAMP NOT NULL,
    value NUMERIC(12, 4) NOT NULL,
    quality_status VARCHAR(40) DEFAULT 'valid',
    source_type VARCHAR(40),
    raw_payload JSONB,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_measurements_stations
        FOREIGN KEY (station_id) REFERENCES monitoring_stations(station_id)
        ON DELETE CASCADE,
    CONSTRAINT fk_measurements_sensors
        FOREIGN KEY (sensor_id) REFERENCES sensors(sensor_id)
        ON DELETE SET NULL,
    CONSTRAINT fk_measurements_parameters
        FOREIGN KEY (parameter_id) REFERENCES parameter_types(parameter_id)
        ON DELETE RESTRICT
);

CREATE TABLE indicator_norms (
    norm_id SERIAL PRIMARY KEY,
    parameter_id INTEGER NOT NULL,
    limit_value NUMERIC(12, 4) NOT NULL,
    norm_type VARCHAR(60) NOT NULL,
    period_type VARCHAR(40),
    description TEXT,
    is_active BOOLEAN DEFAULT TRUE,
    CONSTRAINT fk_norms_parameters
        FOREIGN KEY (parameter_id) REFERENCES parameter_types(parameter_id)
        ON DELETE CASCADE
);

CREATE TABLE alerts (
    alert_id BIGSERIAL PRIMARY KEY,
    measurement_id BIGINT NOT NULL,
    norm_id INTEGER,
    alert_level VARCHAR(30) NOT NULL,
    alert_message TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    is_confirmed BOOLEAN DEFAULT FALSE,
    confirmed_by INTEGER,
    confirmed_at TIMESTAMP,
    CONSTRAINT fk_alerts_measurements
        FOREIGN KEY (measurement_id) REFERENCES measurements(measurement_id)
        ON DELETE CASCADE,
    CONSTRAINT fk_alerts_norms
        FOREIGN KEY (norm_id) REFERENCES indicator_norms(norm_id)
        ON DELETE SET NULL,
    CONSTRAINT fk_alerts_users
        FOREIGN KEY (confirmed_by) REFERENCES users(user_id)
        ON DELETE SET NULL
);

CREATE TABLE notifications (
    notification_id BIGSERIAL PRIMARY KEY,
```

```
    alert_id BIGINT NOT NULL,
    user_id INTEGER NOT NULL,
    channel VARCHAR(30) NOT NULL,
    message TEXT NOT NULL,
    status VARCHAR(30) DEFAULT 'created',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    sent_at TIMESTAMP,
    CONSTRAINT fk_notifications_alerts
        FOREIGN KEY (alert_id) REFERENCES alerts(alert_id)
        ON DELETE CASCADE,
    CONSTRAINT fk_notifications_users
        FOREIGN KEY (user_id) REFERENCES users(user_id)
        ON DELETE CASCADE
);

CREATE TABLE reports (
    report_id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL,
    report_name VARCHAR(150) NOT NULL,
    period_from TIMESTAMP NOT NULL,
    period_to TIMESTAMP NOT NULL,
    report_format VARCHAR(20) DEFAULT 'pdf',
    file_path VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT fk_reports_users
        FOREIGN KEY (user_id) REFERENCES users(user_id)
        ON DELETE CASCADE
);

CREATE TABLE audit_logs (
    log_id BIGSERIAL PRIMARY KEY,
    user_id INTEGER,
    action_type VARCHAR(80) NOT NULL,
    entity_name VARCHAR(80),
    entity_id VARCHAR(80),
    action_description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ip_address VARCHAR(45),
    CONSTRAINT fk_audit_logs_users
        FOREIGN KEY (user_id) REFERENCES users(user_id)
        ON DELETE SET NULL
);

CREATE INDEX idx_measurements_station_time
    ON measurements(station_id, measured_at);

CREATE INDEX idx_measurements_parameter_time
    ON measurements(parameter_id, measured_at);

CREATE INDEX idx_alerts_created_at
    ON alerts(created_at);

CREATE INDEX idx_notifications_user_status
    ON notifications(user_id, status);
```

ДОДАТОК Б

Фрагмент програмного коду серверної частини FastAPI для приймання екологічних вимірювань

```
# main.py
# Серверний модуль приймання, валідації та збереження екологічних вимірювань

from datetime import datetime
from typing import Optional, Dict, Any

from fastapi import FastAPI, HTTPException, Header, Depends
from pydantic import BaseModel, Field
from sqlalchemy import create_engine, text
from sqlalchemy.orm import sessionmaker, Session

DATABASE_URL =
"postgresql+psycopg2://eco_user:eco_password@localhost:5432/eco_monitoring"

engine = create_engine(DATABASE_URL, pool_pre_ping=True)
SessionLocal = sessionmaker(bind=engine, autoflush=False, autocommit=False)

app = FastAPI(
    title="Environmental Monitoring Web Service",
    description="API для приймання, перевірки та аналізу екологічних показників",
    version="1.0.0"
)

API_KEYS = {
    "sensor-demo-key": "demo_sensor_gateway"
}


class MeasurementIn(BaseModel):
    station_code: str = Field(..., min_length=2, max_length=40)
    parameter_code: str = Field(..., min_length=2, max_length=40)
    measured_at: datetime
    value: float
    unit_measure: str = Field(..., max_length=30)
    sensor_code: Optional[str] = None
    source_type: str = "sensor"
    raw_payload: Optional[Dict[str, Any]] = None


class MeasurementResponse(BaseModel):
    measurement_id: int
    status: str
    alert_created: bool
    message: str


def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()


def check_api_key(x_api_key: str = Header(...)):
    if x_api_key not in API_KEYS:
        raise HTTPException(status_code=401, detail="Недійсний ключ доступу до API")
    return API_KEYS[x_api_key]
```

```
def get_station_id(db: Session, station_code: str) -> int:
    result = db.execute(
        text("SELECT station_id FROM monitoring_stations WHERE station_code =
:code"),
        {"code": station_code}
    ).fetchone()

    if not result:
        raise HTTPException(status_code=404, detail="Пункт моніторингу не знайдено")

    return result.station_id


def get_parameter_id(db: Session, parameter_code: str) -> int:
    result = db.execute(
        text("SELECT parameter_id FROM parameter_types WHERE parameter_code =
:code"),
        {"code": parameter_code}
    ).fetchone()

    if not result:
        raise HTTPException(status_code=404, detail="Тип екологічного показника не
знайдено")

    return result.parameter_id


def get_sensor_id(db: Session, sensor_code: Optional[str]) -> Optional[int]:
    if not sensor_code:
        return None

    result = db.execute(
        text("SELECT sensor_id FROM sensors WHERE sensor_code = :code"),
        {"code": sensor_code}
    ).fetchone()

    return result.sensor_id if result else None


def validate_measurement_value(value: float) -> None:
    if value < 0:
        raise HTTPException(
            status_code=400,
            detail="Значення екологічного показника не може бути від’ємним"
        )

    if value > 100000:
        raise HTTPException(
            status_code=400,
            detail="Значення показника виходить за межі допустимого діапазону"
        )


def create_alert_if_needed(db: Session, measurement_id: int, parameter_id: int,
value: float) -> bool:
    norm = db.execute(
        text("""
            SELECT norm_id, limit_value, norm_type
            FROM indicator_norms
            WHERE parameter_id = :parameter_id
              AND is_active = TRUE
```

```
            ORDER BY limit_value ASC
            LIMIT 1
        """),
        {"parameter_id": parameter_id}
    ).fetchone()

    if not norm:
        return False

    if value <= float(norm.limit_value):
        return False

    alert_level = "warning"
    if value >= float(norm.limit_value) * 1.5:
        alert_level = "critical"

    alert_message = (
        f"Зафіксовано перевищення нормативного значення. "
        f"Поточне значення: {value}; допустима межа: {norm.limit_value}."
    )

    db.execute(
        text("""
            INSERT INTO alerts (
                measurement_id, norm_id, alert_level, alert_message
            )
            VALUES (
                :measurement_id, :norm_id, :alert_level, :alert_message
            )
        """),
        {
            "measurement_id": measurement_id,
            "norm_id": norm.norm_id,
            "alert_level": alert_level,
            "alert_message": alert_message
        }
    )

    return True


@app.post("/api/v1/measurements", response_model=MeasurementResponse)
def receive_measurement(
    payload: MeasurementIn,
    db: Session = Depends(get_db),
    source_name: str = Depends(check_api_key)
):
    validate_measurement_value(payload.value)

    station_id = get_station_id(db, payload.station_code)
    parameter_id = get_parameter_id(db, payload.parameter_code)
    sensor_id = get_sensor_id(db, payload.sensor_code)

    result = db.execute(
        text("""
            INSERT INTO measurements (
                station_id,
                sensor_id,
                parameter_id,
                measured_at,
                value,
                quality_status,
                source_type,
```

```
                raw_payload
            )
            VALUES (
                :station_id,
                :sensor_id,
                :parameter_id,
                :measured_at,
                :value,
                :quality_status,
                :source_type,
                CAST(:raw_payload AS jsonb)
            )
            RETURNING measurement_id
        """),
        {
            "station_id": station_id,
            "sensor_id": sensor_id,
            "parameter_id": parameter_id,
            "measured_at": payload.measured_at,
            "value": payload.value,
            "quality_status": "valid",
            "source_type": payload.source_type,
            "raw_payload": "{}" if payload.raw_payload is None else
str(payload.raw_payload).replace("'", '"')
        }
    ).fetchone()

    measurement_id = result.measurement_id

    alert_created = create_alert_if_needed(
        db=db,
        measurement_id=measurement_id,
        parameter_id=parameter_id,
        value=payload.value
    )

    db.execute(
        text("""
            INSERT INTO audit_logs (
                user_id,
                action_type,
                entity_name,
                entity_id,
                action_description
            )
            VALUES (
                NULL,
                'CREATE_MEASUREMENT',
                'measurements',
                :entity_id,
                :description
            )
        """),
        {
            "entity_id": str(measurement_id),
            "description": f"Отримано вимірювання від джерела {source_name}"
        }
    )

    db.commit()

    return MeasurementResponse(
        measurement_id=measurement_id,
```

```
        status="saved",
        alert_created=alert_created,
        message="Вимірювання успішно прийнято та збережено"
    )


@app.get("/api/v1/stations/{station_code}/latest")
def get_latest_station_measurements(
    station_code: str,
    db: Session = Depends(get_db)
):
    station_id = get_station_id(db, station_code)

    rows = db.execute(
        text("""
            SELECT
                p.parameter_code,
                p.parameter_name,
                p.unit_measure,
                m.value,
                m.measured_at,
                m.quality_status
            FROM measurements m
            JOIN parameter_types p ON p.parameter_id = m.parameter_id
            WHERE m.station_id = :station_id
              AND m.measured_at = (
                  SELECT MAX(m2.measured_at)
                  FROM measurements m2
                  WHERE m2.station_id = m.station_id
                    AND m2.parameter_id = m.parameter_id
              )
            ORDER BY p.parameter_name
        """),
        {"station_id": station_id}
    ).fetchall()

    return [
        {
            "parameter_code": row.parameter_code,
            "parameter_name": row.parameter_name,
            "unit_measure": row.unit_measure,
            "value": float(row.value),
            "measured_at": row.measured_at,
            "quality_status": row.quality_status
        }
        for row in rows
    ]
```

ДОДАТОК В

Фрагмент програмного коду аналітичного модуля вебсервісу

```
# analytics_service.py
# Модуль агрегації екологічних показників і підготовки даних для графіків

from datetime import datetime
from typing import List, Dict, Any

from sqlalchemy import text
from sqlalchemy.orm import Session


class AnalyticsService:
    def __init__(self, db: Session):
        self.db = db

    def get_parameter_dynamics(
        self,
        station_id: int,
        parameter_id: int,
        date_from: datetime,
        date_to: datetime
    ) -> List[Dict[str, Any]]:
        rows = self.db.execute(
            text("""
                SELECT
                    date_trunc('hour', measured_at) AS period_hour,
                    AVG(value) AS avg_value,
                    MIN(value) AS min_value,
                    MAX(value) AS max_value,
                    COUNT(*) AS measurement_count
                FROM measurements
                WHERE station_id = :station_id
                  AND parameter_id = :parameter_id
                  AND measured_at BETWEEN :date_from AND :date_to
                  AND quality_status = 'valid'
                GROUP BY date_trunc('hour', measured_at)
                ORDER BY period_hour
            """),
            {
                "station_id": station_id,
                "parameter_id": parameter_id,
                "date_from": date_from,
                "date_to": date_to
            }
        ).fetchall()

        return [
            {
                "period": row.period_hour.isoformat(),
                "avg_value": round(float(row.avg_value), 3),
                "min_value": round(float(row.min_value), 3),
                "max_value": round(float(row.max_value), 3),
                "measurement_count": row.measurement_count
            }
            for row in rows
        ]

    def calculate_station_summary(
        self,
```

```
    station_id: int,
    date_from: datetime,
    date_to: datetime
) -> Dict[str, Any]:
    rows = self.db.execute(
        text("""
            SELECT
                p.parameter_code,
                p.parameter_name,
                p.unit_measure,
                AVG(m.value) AS avg_value,
                MAX(m.value) AS max_value,
                MIN(m.value) AS min_value,
                COUNT(m.measurement_id) AS total_count
            FROM measurements m
            JOIN parameter_types p ON p.parameter_id = m.parameter_id
            WHERE m.station_id = :station_id
              AND m.measured_at BETWEEN :date_from AND :date_to
              AND m.quality_status = 'valid'
            GROUP BY p.parameter_code, p.parameter_name, p.unit_measure
            ORDER BY p.parameter_name
        """),
        {
            "station_id": station_id,
            "date_from": date_from,
            "date_to": date_to
        }
    ).fetchall()

    return {
        "station_id": station_id,
        "date_from": date_from.isoformat(),
        "date_to": date_to.isoformat(),
        "parameters": [
            {
                "parameter_code": row.parameter_code,
                "parameter_name": row.parameter_name,
                "unit_measure": row.unit_measure,
                "avg_value": round(float(row.avg_value), 3),
                "max_value": round(float(row.max_value), 3),
                "min_value": round(float(row.min_value), 3),
                "total_count": row.total_count
            }
            for row in rows
        ]
    }

def get_exceedance_statistics(
    self,
    date_from: datetime,
    date_to: datetime
) -> List[Dict[str, Any]]:
    rows = self.db.execute(
        text("""
            SELECT
                s.station_name,
                p.parameter_name,
                a.alert_level,
                COUNT(a.alert_id) AS alert_count
            FROM alerts a
            JOIN measurements m ON m.measurement_id = a.measurement_id
            JOIN monitoring_stations s ON s.station_id = m.station_id
            JOIN parameter_types p ON p.parameter_id = m.parameter_id
```

```
            WHERE a.created_at BETWEEN :date_from AND :date_to
            GROUP BY s.station_name, p.parameter_name, a.alert_level
            ORDER BY alert_count DESC
        """),
        {
            "date_from": date_from,
            "date_to": date_to
        }
    ).fetchall()

    return [
        {
            "station_name": row.station_name,
            "parameter_name": row.parameter_name,
            "alert_level": row.alert_level,
            "alert_count": row.alert_count
        }
        for row in rows
    ]

def prepare_report_data(
    self,
    station_id: int,
    date_from: datetime,
    date_to: datetime
) -> Dict[str, Any]:
    summary = self.calculate_station_summary(station_id, date_from, date_to)

    alerts = self.db.execute(
        text("""
            SELECT
                a.alert_level,
                a.alert_message,
                a.created_at,
                m.value,
                p.parameter_name,
                p.unit_measure
            FROM alerts a
            JOIN measurements m ON m.measurement_id = a.measurement_id
            JOIN parameter_types p ON p.parameter_id = m.parameter_id
            WHERE m.station_id = :station_id
              AND a.created_at BETWEEN :date_from AND :date_to
            ORDER BY a.created_at DESC
        """),
        {
            "station_id": station_id,
            "date_from": date_from,
            "date_to": date_to
        }
    ).fetchall()

    summary["alerts"] = [
        {
            "alert_level": row.alert_level,
            "alert_message": row.alert_message,
            "created_at": row.created_at.isoformat(),
            "value": float(row.value),
            "parameter_name": row.parameter_name,
            "unit_measure": row.unit_measure
        }
        for row in alerts
    ]
    return summary
```

ДОДАТОК Г

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Маслиган Віталій Романович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Програмне забезпечення вебсервісу для аналізу екологічних показників та візуалізації даних моніторингу» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - [ ] інструменти генеративного ШІ не використовувалися.
    - [ ] інструменти ШІ (вказати назву: **ChatGPT**, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - [ ] перекладу іншомовних джерел;
        - [+] стилістичного редагування та перевірки граматики;
        - [+] допомоги у написанні/відлагодженні програмного коду;
        - [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **Віталій Маслиган**

(підпис)