Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Інформаційно-аналітична система моніторингу та прогнозування в галузі управління водними ресурсами

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ–22–1
_____/В. С. Сорокін /

Керівник кваліфікаційної роботи _____ / О. Г. Рибальченко /

Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг

2026

Криворізький національний університет

Факультет:Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри
_____ А. М. Стрюк
«_» _____ 2026р.

ЗАВДАННЯ
на кваліфікаційну роботу
студенту групи ІПЗ–22–1 Сорокін Віталій Сергійович

1. Тема: Інформаційно-аналітична система моніторингу та прогнозування в галузі управління водними ресурсами затверджено наказом по КНУ № 284-с від «28» травень 2026 р.
2. Термін подання студентом закінченої роботи: «05» червня 2026 р.
3. Вихідні дані по роботі: розроблення інформаційно-аналітичної система збору, збереження, обробку та візуалізацію даних про стан водних ресурсів, а також прогнозування їх змін для прийняття управлінських рішень.
4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): аналіз існуючих інформаційних систем моніторингу та управління водними ресурсами, проектування архітектури інформаційно-аналітичної системи, програмна реалізація системи, робота з базою даних, проведення тестування та оцінку ефективності.
5. Перелік ілюстративного матеріалу: схеми структурної системи , блок–схеми алгоритму, діаграми послідовності, знімки екранних форм інтерфейсу.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Огляд літератури та нормативно-правової бази з управління водними ресурсами	27.02.2026 – 29.02.2026
2	Аналіз існуючих інформаційно-аналітичних систем моніторингу водних ресурсів	30.02.2026 – 09.03.2026
3	Збір та порівняльний аналіз методів моніторингу та прогнозування гідрологічних і гідрохімічних параметрів	10.03.2026– 18.03.2026
4	Підготовка матеріалів першого розділу роботи	19.03.2026 – 23.03.2026
5	Розроблення математичних моделей прогнозування стану водних ресурсів	24.03.2026– 30.03.2026
6	Проектування архітектури системи та структури бази даних	1.04.2026 – 04.04.2026
7	Підготовка матеріалів другого та третього розділів	05.04.2026– 13.05.2026
8	Розроблення програмного забезпечення	14.05.2026 – 20.05.2026
9	Проведення тестування, верифікації моделей та оцінка ефективності системи	21.05.2026 – 26.05.2026
10	Оформлення пояснювальної записки	27.05.2026 – 05.06.2026

Дата видачі завдання: «26» лютого 2026 р.

Студент _____/ В. С. Сорокін

Керівник роботи _____/О. Г. Рибальченко

РЕФЕРАТ

ІНФОРМАЦІЙНА СИСТЕМА, ЕКОЛОГІЧНИЙ МОНІТОРИНГ, ВОДОКОРИСТУВАННЯ, ЯКІСТЬ ВОД, БАЛАНСОВИЙ ЗВІТ, ВЕБЗАСТОСУНОК, REACT, C#, ASP.NET CORE, TYPESCRIPT.

Пояснювальна записка: 50 с., 6 табл., 31 рис., 1 дод., 28 джерел.

Метою кваліфікаційної роботи є розробка інформаційно-аналітичної системи для обліку, моніторингу та аналізу використання водних ресурсів, контролю екологічних нормативів якості води та автоматизації формування регламентної звітності підприємств.

У роботі проведено системний аналіз предметної області, досліджено вимоги європейського та національного нормативно-правового регулювання у сфері управління водними ресурсами. Виявлено недоліки існуючих процесів обробки екологічних даних та обґрунтовано необхідність їхньої автоматизації для забезпечення достовірності екологічного контролю.

Спроєктовано клієнт-серверну архітектуру програмного рішення. Для оптимізації серверної частини та ізоляції бізнес-логіки застосовано архітектурний патерн CQRS у поєднанні з централізованим диспетчером MediatR. Розроблено логічну та фізичну структуру реляційної бази даних, що забезпечує збереження нормативно-довідкової інформації (реєстри річкових басейнів, забруднюючих речовин), дозвільних документів, протоколів лабораторних вимірювань якості води та звітів про водокористування.

Програмно реалізовано серверну систему на базі платформи ASP.NET Core та клієнтський односторінковий застосунок з використанням бібліотеки React і TypeScript. Впроваджено функціонал автоматизованого розрахунку гідрологічних балансів, динамічного контролю дотримання нормативних лімітів забору та скиду води. Реалізовано аналітичні модулі для оцінки багаторічних трендів водоспоживання та просторової візуалізації екологічного навантаження на водні об'єкти з прив'язкою до територіальних класифікаторів.

Отримане програмне забезпечення мінімізує вплив людського фактора під час агрегації екологічних показників, підвищує швидкість ідентифікації перевищень нормативів гранично допустимих скидів (ГДС) та оптимізує процеси обміну даними між суб'єктами господарювання і контролюючими органами.

ABSTRACT

INFORMATION SYSTEM, ECOLOGICAL MONITORING, WATER USAGE, WATER QUALITY, BALANCE REPORT, WEB APPLICATION, REACT, C#, ASP.NET CORE, TYPESCRIPT.

Explanatory note: 50 p., 6 tables, 31 figures, 1 appendix, 28 sources.

The objective of the qualification work is the development of an information and analytical system for the accounting, monitoring, and analysis of water resources usage, the control of environmental water quality standards, and the automation of regulatory reporting generation for enterprises.

A system analysis of the subject area was conducted, and the requirements of European and national regulatory frameworks in the field of water resources management were investigated. The shortcomings of existing environmental data processing procedures were identified, and the necessity of their automation to ensure the reliability of environmental control was substantiated.

The client-server architecture of the software solution was designed. To optimize the server side and isolate business logic, the CQRS architectural pattern was applied in combination with the MediatR centralized dispatcher. The logical and physical structure of a relational database was developed, ensuring the storage of regulatory and reference information (registers of river basins, pollutants), permitting documents, protocols of laboratory water quality measurements, and water usage reports.

The server system was implemented based on the ASP.NET Core platform, and the client single-page application was built using the React library and TypeScript. The functionality for automated calculation of hydrological balances and dynamic control of compliance with regulatory limits for water intake and discharge was implemented. Analytical modules were deployed to assess long-term water consumption trends and provide spatial visualization of the ecological load on water bodies, linked to territorial classifiers.

The resulting software minimizes the human factor during the aggregation of environmental indicators, increases the speed of identifying exceedances of maximum allowable discharge (MAD) standards, and optimizes data exchange processes between business entities and regulatory authorities.

ЗМІСТ

ВСТУП

Раціональне використання водних ресурсів та контроль за їх якісним і кількісним станом є однією з ключових умов забезпечення екологічної безпеки та сталого економічного розвитку. В умовах інтеграції України до європейського екологічного та правового простору виникає гостра потреба у модернізації національної системи екологічного моніторингу відповідно до вимог ЄС.

Існуючі процеси збору, обробки, верифікації та агрегації даних на рівні екологічних служб підприємств-водокористувачів та хіміко-аналітичних лабораторій характеризуються високим рівнем децентралізації, використанням застарілих паперових носіїв або ізольованих електронних таблиць, а також значним впливом людського фактора. Це призводить до тривалого часу обробки інформації, затримок у виявленні перевищень нормативів гранично допустимих скидів (ГДС) забруднюючих речовин, помилок під час порічного розрахунку гідрологічних балансів та ускладнює оперативний аналіз антропогенного навантаження на водні екосистеми. Таким чином, розробка та впровадження інтегрованої інформаційно-аналітичної системи для автоматизації обліку водокористування, верифікації лабораторних вимірювань та контролю дотримання екологічних нормативів якості води є актуальним технічним завданням.

Мета роботи – розробка інформаційно-аналітичної системи для автоматизації процесів обліку, моніторингу та аналізу використання водних ресурсів, динамічного контролю дотримання екологічних нормативів якості води та автоматизації формування регламентної звітності підприємств.

Практичне значення отриманих результатів полягає у створенні кросплатформного програмного забезпечення, яке дозволяє мінімізувати вплив людського фактора під час агрегації показників, підвищити швидкість ідентифікації перевищень нормативів лімітів дозволів та ГДС, автоматизувати контроль відповідності хімічного стану вод державним стандартам, а також оптимізувати процеси обміну аналітичною інформацією.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА БІЗНЕС-ВИМОГИ ДО ОБ'ЄКТА ПРОЄКТУВАННЯ

1.1 Характеристика предметної області

Предметну область проєктування складають процеси збору, первинної реєстрації, консолідації, порівняльного аналізу та формування регламентної звітності щодо якості водних ресурсів, обсягів водоспоживання і скидів зворотних (стічних) вод. Зазначений комплекс процесів протікає на рівні екологічних служб промислових підприємств-водокористувачів та безпосередньо взаємодіючих із ними хіміко-аналітичних лабораторій [1]. Головною метою функціонування цього об'єкта є забезпечення безперервного моніторингу екологічного навантаження на навколишнє природне середовище, дотримання встановлених законодавством нормативів водокористування та своєчасне подання достовірних звітних даних до державних органів контролю [3].

Організація діяльності об’єкта проєктування регламентується чинним законодавством України, зокрема Водним кодексом та Порядком ведення державного обліку водокористування [1]. Результати функціонування є першоджерелом для нарахування рентної плати за спеціальне водокористування та формування Державного водного кадастру [2].

Процес збору та обробки інформації щодо водокористування та якості стічних вод характеризується взаємодією кількох ключових суб'єктів. Кожен із них виконує чітко визначену роль у загальному контурі обробки даних, починаючи від безпосереднього вимірювання фізико-хімічних показників і закінчуючи прийняттям рішень на державному рівні (Таблиця 1.1).

Таблиця 1.1 Суб'єкти інформаційної взаємодії та їхні функціональні ролі в системі.

Суб'єкт процесу	Інформаційна роль	Основні функції в процесі обробки інформації	Джерела та приймачі даних
Хіміко-аналітична лабораторія	Постачальник первинних вимірювань	Проведення інструментальних досліджень проб води, визначення фактичних концентрацій речовин, оформлення паперових протоколів аналізу.	Джерело: відібрані проби води. Приймач: еколог підприємства.
Екологічна служба (еколог підприємства)	Консолідатор, аналітик та реєстратор даних	Реєстрація даних вимірювань та первинного обліку води; порівняння концентрацій із нормативами; розрахунок маси скидів; формування річного звіту за формою № 2ТП-водгосп.	Джерело: протоколи лабораторії, журнали обліку, нормативні акти.[4] Приймач: Держводагентство, ДПС.
Держводагентство (басейнові управління)	Отримувач та верифікатор регламентної звітності	Прийом електронної форми звіту № 2ТП-водгосп через веб-портал; перевірка достовірності даних; систематизація даних у Державному водному кадастрі.	Джерело: екологічна служба підприємства.[1] Приймач: державні реєстри та кадастри.
Державна податкова служба (ДПС)	Контролюючий орган фіскального спрямування	Прийом квартальних податкових декларацій з рентної плати разом із копією електронного звіту № 2ТП-водгосп; контроль правильності нарахування податкових зобов'язань.	Джерело: екологічна служба підприємства. Приймач: інтегрована податкова база даних.

Визначена структура суб'єктів взаємодії демонструє багаторівневий характер обробки даних екологічного моніторингу. Існуючий розподіл функціональних ролей між внутрішніми підрозділами підприємства та зовнішніми контролюючими органами зумовлює необхідність проєктування багаторівневої рольової моделі управління доступом (RBAC) у цільовій інформаційній системі. Це забезпечить ізоляцію даних та розмежування зон відповідальності учасників процесу.

1.2 Характеристика інформаційних потоків у стані «As-Is»

Аналіз існуючого стану інформаційних потоків свідчить про високий рівень децентралізації та фрагментації даних на підприємстві. Інформація циркулює переважно у вигляді паперових документів або ізольованих файлів офісних форматів, що потребує постійного ручного втручання для її агрегації та обробки (Рисунок 1.1) [4].

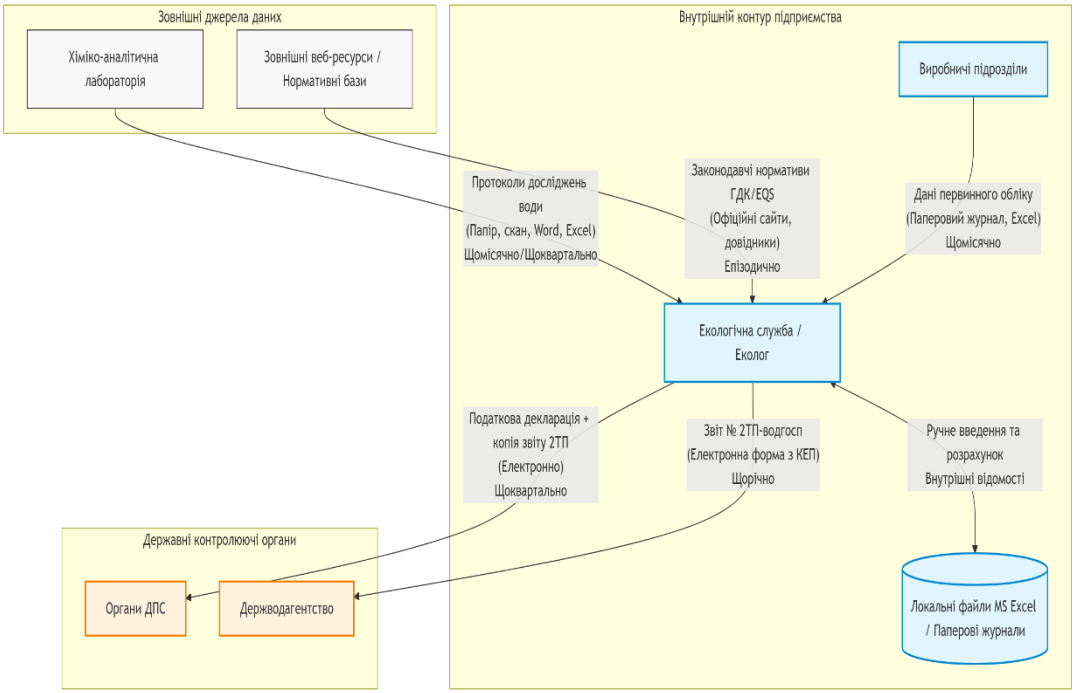


Рисунок 1.1 Інформаційні потоки у поточному стані

Існуючий підхід до ведення екологічного моніторингу та підготовки звітності має низку суттєвих методологічних та технологічних недоліків (Таблиця 1.2). Вони знижують загальну ефективність роботи екологічної служби, сповільнюють обробку даних та створюють серйозні юридичні й фінансові ризики для діяльності підприємства.

Таблиця 1.2 Порівняння характеристик процесів у станах «As-Is» та «To-Be».

Критерій порівняння	Поточний стан	Цільовий автоматизований стан
Спосіб збереження даних	Децентралізовані локальні файли Excel та паперові носії.	Єдина централізована реляційна база даних.
Ідентифікація речовин	Ручний пошук у законодавчих актах; відсутність зв'язку з CAS-номерами.	Автоматизований довідник речовин, інтегрований із CAS-реєстром.
Контроль нормативів (ГДК/EQS)	Візуальне зіставлення екологом фактичних даних із нормативами.	Автоматичний розрахунок кратності перевищень та миттєва сигналізація про відхилення.
Розрахунок маси скидів	Ручне введення формул в Excel для кожного випуску та місяця.	Автоматичний розрахунок за вбудованими алгоритмами на основі первинних даних.
Формування форми № 2ТП-водгосп	Ручне агрегування показників та заповнення веб-форми на порталі.	Автоматична генерація звіту № 2ТП-водгосп у потрібному форматі для імпорту.

Характеристики у станах «As-Is» та «To-Be» демонструють, що існуюча паперово-таблична модель збору, обробки та аналізу екологічних даних вичерпала свій ресурс. Вона не здатна забезпечити необхідну швидкість, точність та системність управління інформацією в умовах посилення вимог державного екологічного контролю та цифровізації звітності.

1.3 Нормативний базис Європейського Союзу

Європейське законодавство у сфері управління водними ресурсами формує суворий набір бізнес-вимог до проектування інформаційних систем екологічного моніторингу. Нормативні акти ЄС виступають джерелом функціональних обмежень, алгоритмів класифікації та правил валідації даних, що визначають поведінку системи на етапах збору, математичної обробки та регламентованого експорту інформації. Система повинна трансформувати сирі масиви даних у структурований актив комплаєнсу, забезпечуючи повну відстежуваність та доказовість результатів.

1.3.1 Вимоги до ідентифікації даних

Фундаментальною вимогою Директиви 2000/60/ЄС (Водна рамкова директива, ВРД) є впровадження жорсткої просторової ієрархічної моделі даних для об'єктів моніторингу. Інформаційна система повинна підтримувати облік просторових локацій не як ізольованих географічних точок, а як залежних елементів єдиної гідрологічної мережі. Основною одиницею ідентифікації та обліку в системі виступає сутність «масив води». Бізнес-правило диктує необхідність прив'язки кожного масиву до відповідного річкового басейну, що формує топологічну залежність об'єктів у системі [10].

Зазначений нормативний акт вимагає від системи забезпечення процедури категоризації кожного водного масиву. Інтерфейс ідентифікації об'єктів повинен містити обов'язковий атрибут походження, значення якого суворо обмежені трьома варіантами: природні, штучні та істотно змінені масиви. Залежно від присвоєної категорії, система повинна активувати механізм динамічного обмеження доступних станів для оцінки. Для об'єктів з категорією «природні» алгоритм ідентифікації повинен активувати шкалу «екологічного стану» (екологічний та хімічний стан). Натомість для штучних та істотно змінених об'єктів система зобов'язана блокувати використання стандартної шкали та застосовувати метрику «екологічного потенціалу».

Директиви 2008/105/ЄС та 2013/39/ЄС (Директиви стандартів якості середовища) формують функціональні вимоги до модуля ведення нормативно-довідкової інформації щодо хімічних параметрів. Довідник речовин у системі повинен містити спеціалізовані бінарні атрибути для маркування кожного хімічного елемента чи сполуки як «пріоритетної речовини» (Priority Substances, PS) або «пріоритетної небезпечної речовини» (Priority Hazardous Substances, PHS). Ця системна категоризація безпосередньо впливає на подальшу логіку обробки даних: для маркерів PHS система повинна активувати аналітичний модуль контролю довгострокового тренду на повне припинення викидів протягом встановленого 20-річного циклу, тоді як для маркерів PS дозволяється контроль виключно в межах статичних встановлених лімітів без вимоги повної елімінації [11].

Додатковою вимогою до ідентифікаційної моделі даних є запровадження ідентифікатора матриці моніторингу. Система повинна на рівні метаданих розрізняти результати лабораторних досліджень, отримані безпосередньо з проб води, та результати моніторингу біоти (риби, молюски, донні відкладення). Нормативний акт диктує правило: для речовин зі схильністю до біоакумуляції система повинна застосовувати нормативні ліміти виключно до тих записів, які ідентифіковані атрибутом біологічної матриці, блокуючи застосування цих лімітів до водних зразків [12].

Специфічні директиви ЄС встановлюють додаткові параметри ідентифікації для специфічних типів масивів вод, що вимагає розширення атрибутивного складу геопросторових об'єктів у базі даних. Наприклад, Нітратна директива (91/676/EEC) вимагає наявності в системі інструментарію для ідентифікації та полігонального виділення спеціальних «Нітратно-вразливих зон» (NVZs). Будь-який суб'єкт господарювання або точка моніторингу, просторові координати якої перетинаються з полігоном NVZ, повинна автоматично успадковувати суворіші обмеження щодо лімітів внесення азоту (не більше 170 кг на гектар на рік), що впливає на алгоритми валідації [13].

1.3.2 Вимоги до алгоритмів обробки даних

Функціональні вимоги до модуля математичної обробки та автоматичної валідації безпосередньо базуються на концепції Екологічних нормативів якості середовища (EQS). Аналітичний рушій цільової системи зобов'язаний підтримувати два повністю незалежні алгоритми розрахунку перевищень, які активуються паралельно [11].

Перший алгоритм заснований на показнику AA-EQS (Annual Average). Його функціональна специфікація вимагає агрегації всіх підтверджених вимірювань для конкретної речовини в межах певного водного об'єкта за повний календарний рік. Система повинна обчислити середнє арифметичне значення концентрації і порівняти отриманий результат з лімітом AA-EQS, призначеним для контролю хронічного екологічного впливу.

Другий алгоритм базується на показнику MAC-EQS і функціонує як тригер реального часу. Кожен внесений результат аналізу миттєво порівнюється з лімітом для виявлення гострої токсичності та пікових скидів. У разі перевищення нормативу система автоматично генерує сповіщення та фіксує інцидент.

Згідно з Директивою 2009/90/ЄС, підсистема імпорту та контролю якості лабораторних даних повинна забезпечувати обов'язкову фіксацію межі квантифікації (LOQ).

Алгоритм вхідного контролю виконує математичну валідацію показника LOQ відносно встановлених нормативів EQS. Дані з недостатньою точністю методу автоматично маркуються та виключаються з розрахунків комплаєнсу.

Розрахунковий апарат системи імплементує логіку умовної підстановки. Для показників концентрації, що є нижчими за межу квантифікації (< LOQ), під час агрегації даних застосовуються відповідні розрахункові значення.

Згідно з Директивою 2006/118/ЄС, підсистема аналізу підземних вод має реалізовувати алгоритми визначення довгострокових трендів зміни концентрацій. При

фіксації висхідного тренду забруднення система повинна предиктивно генерувати сповіщення про ризик, навіть якщо поточні показники ще не досягли порогових значень.

Відповідно до Директиви 91/271/ЄЕС, система повинна автоматично розраховувати інженерну метрику «еквівалент населення» (р.е.) на основі добового показника біохімічного споживання кисню. Значення цієї метрики визначає нормативні вимоги до рівня очищення, які автоматично застосовуються до об'єкта генерації стічних вод.

Додатково архітектура системи має забезпечувати гнучкість для динамічного підключення нових маркерних мікрозабруднювачів до алгоритмів обчислення ефективності четвертинної стадії очищення на очисних спорудах.

1.3.3 Вимоги до звітності та експорту інформації

Вимоги до підсистеми генерації звітності за європейськими стандартами базуються на концепції повної хронологічної версійності даних. Оскільки плани управління річковими басейнами розробляються на фіксовані шестирічні цикли, модуль звітності повинен мати функціонал створення статичних «зрізів» даних із прив'язкою до конкретного управлінського циклу [10].

Критичною вимогою є впровадження часової версійності нормативних довідників. Європейський список пріоритетних речовин, що підлягають моніторингу («Watch List»), підлягає нормативному оновленню кожні два роки. Стандарти EQS також регулярно переглядаються [14]. Система не має права здійснювати деструктивне оновлення записів довідника. Вона повинна зберігати історію змін нормативних значень із зазначенням дати початку та дати закінчення їхньої дії. Бізнес-правило формування історичної звітності вимагає, щоб будь-який перерахунок ретроспективних показників комплаєнсу за минулі роки здійснювався виключно з використанням тих параметрів EQS, які були юридично чинними на момент фіксації вимірювання. Це запобігає ризику каскадного пошкодження та викривлення історичної аналітики екологічного стану при кожній зміні законодавства.

Модуль експорту даних повинен бути спроектований із врахуванням вимог інтеграції із загальноєвропейською системою WISE (Water Information System for Europe). Це диктує жорстку специфікацію вихідних форматів. Система зобов'язана генерувати пакети даних у структурованому форматі XML, сумісному зі стандартами обміну даними Європейського агентства з довкілля (EEA), використовуючи кодування схем Reportnet 3. Під час формування експортного пакету система повинна застосовувати єдиний класифікатор просторових даних INSPIRE для геокодування об'єктів [12].

1.4 Національний нормативно-правовий ландшафт України

Національний екологічний нормативний ландшафт України характеризується перехідним станом, поєднуючи поступову імплементацію європейських директив із збереженням діючих індивідуальних методик нормування підприємств. Цей дуалізм формує специфічний масив бізнес-вимог. Інформаційна система повинна підтримувати паралельне виконання різних алгоритмів обробки даних, забезпечуючи водокористувачів інструментарієм як для внутрішнього оперативного контролю, так і для формування статистичної регламентованої звітності.

1.4.1 Вимоги до ідентифікації суб'єктів, об'єктів та речовин

Основою архітектури безпеки та управління доступом у системі виступає рольова модель (Role-Based Access Control, RBAC), контури якої визначені Постановою Кабінету Міністрів України № 758 [7]. Система повинна реалізувати багатокористувацьку структуру із суворим розмежуванням прав на чотирьох ієрархічних рівнях доступу.

Таблиця 1.3 Основі функціональні обов'язків суб'єктів державного екологічного моніторингу.

Суб'єкт в ІС	Рольовий доступ та функції	Тип даних, що вносяться / обробляються	Періодичність взаємодії
Міндовкілля	Національний адміністратор, затвердження нормативів та ведення реєстрів	Узагальнені дані моніторингу, реєстр лімітів гранично допустимих скидів	Постійно, річний цикл планування
Держводагентство	Басейновий оператор, контроль питних водозаборів	Результати спостережень питних джерел, звіти водокористувачів	Щомісячно / Щоквартально
Підприємства-водокористувачі	Оператор джерела забруднення	Обсяги зворотних вод, фактичні концентрації речовин	Згідно з графіком контролю, річна форма 2ТП

Функціональна вимога ізоляції даних диктує, що облікові записи з роллю «Оператор джерела забруднення» (представники підприємств) повинні мати повноваження виключно

на внесення, редагування та запис транзакційних даних у межах зареєстрованих за їхнім підприємством просторових точок моніторингу (водовипусків). Натомість, користувачі з ієрархічною роллю «Басейновий оператор» повинні наділятися системними привілеями на перегляд, агрегацію та валідацію масивів даних від усіх суб'єктів господарювання, які територіально належать до закріпленого за ними гідрографічного району, без права прямого редагування первинних лабораторних результатів підприємств.

Вимоги до ідентифікації хімічних параметрів на національному рівні визначені Наказом Міндовкілля № 45. Документ безальтернативно вимагає переходу від застарілих текстових найменувань забруднювачів до єдиних міжнародних ідентифікаторів. Первинним ключем для ідентифікації будь-якої речовини у довідниках системи, процесах валідації та налаштуваннях лімітів повинен виступати виключно міжнародний код CAS (Chemical Abstracts Service) [8]. Ця архітектурна вимога повністю усуває ризик дублювання сутностей внаслідок використання хімічних синонімів, торгових назв або помилок транслітерації.

Проте, враховуючи перехідний період, система повинна підтримувати механізм крос-мапінгу. Довідник має містити інструмент для логічного зв'язування унікального коду CAS із застарілими вітчизняними номенклатурними кодами, які продовжують використовуватися податковими органами у фіскальній звітності (Додаток А).

1.4.2 Вимоги до обробки даних

Операційним ядром інформаційної системи має бути розрахунковий модуль, функціональність якого суворо регламентується Методичними рекомендаціями [9]. Система повинна імплементувати складну математичну логіку розрахунку гранично допустимих скидів (ГДС).

На відміну від європейських EQS, які є константами у довіднику, український показник ГДС є обчислюваною, динамічною величиною. Бізнес-правило вимагає від системи безперервного розрахунку максимально допустимої маси забруднюючої речовини, яку дозволено скинути зі стічними водами за одиницю часу для кожного індивідуального водовипуску підприємства. Розрахунковий рушій системи повинен приймати на вхід множину змінних параметрів: фонову концентрацію речовини у приймальному водному об'єкті до місця скиду, фактичну витрату стічних вод підприємства, актуальну витрату води у водному об'єкті та розрахунковий коефіцієнт змішування.

Функціональним обмеженням є те, що математична модель ГДС повинна функціонувати в режимі постійного перерахунку. Будь-яка зміна гідрологічних параметрів річки або збільшення фонового забруднення, зафіксоване системою через інтерфейс

введення, повинна негайно ініціювати автоматичний каскадний перерахунок діючих лімітів ГДС для пов'язаного водовипуску. Після розрахунку ліміту, система активує алгоритм валідації: фактична розрахована маса скиду порівнюється з допустимим лімітом ГДС. При виявленні математичного перевищення, система ініціює подію порушення, що трансформує сирі дані в активний механізм попередження екологічних інцидентів.

Настання події наднормативного скиду активує вимоги до модуля фінансової аналітики, засновані на Наказі № 116. Система повинна інтегрувати алгоритм визначення розмірів відшкодування збитків за екологічну шкоду [9]. Процедура обробки ініціює підпрограму, яка використовує як вхідні дані зафіксовану величину перевищення (масу понад ГДС). Алгоритм застосовує до цієї маси специфічний коефіцієнт токсичності для конкретної речовини (ідентифікованої за CAS) та враховує часовий інтервал (тривалість) наднормативного скиду. Результатом виконання алгоритму є автоматична генерація оціночної вартості екологічного збитку в національній валюті. Даний показник прикріплюється системою як обов'язковий атрибут до протоколу інциденту.

1.4.3 Вимоги до генерації регламентованої статистичної звітності

Процес державного обліку водокористування формує чіткий бізнес-регламент щодо функціонування системної підсистеми ETL (Extract, Transform, Load) для автоматичної генерації річної форми звітності № 2ТП-водгосп [5].

Система повинна забезпечувати обробку даних у межах суворих часових обмежень: звітний період охоплює виключно календарний рік, а крайній термін подання (валідації) пакету даних фіксується системою як 01 лютого року, наступного за звітним. Алгоритм відбору даних повинен фільтрувати суб'єктів звітування за нормативними критеріями. Звіт формується обов'язково, якщо система фіксує сумарний забір води із підземних чи поверхневих джерел в обсязі понад 5 метрів кубічних на добу, передачу зворотних вод до систем водовідведення або здійснення прямого скиду зворотних вод незалежно від їх об'єму, а також для підприємств гідроенергетики.

Логіка трансформації даних у звітному модулі є надзвичайно ресурсоємною. Система зобов'язана автоматично агрегувати десятки тисяч щоденних атомарних транзакцій (записи лабораторних журналів, покази витратомірів). Ці первинні дані конвертуються: обсяги використаної води підсумовуються в тисячах кубічних метрів, а маси скинутих хімічних речовин – у тоннах. Ключова функціональна вимога алгоритму агрегації полягає у необхідності перехресного групування цих мас за специфічними нормативними категоріями. Система повинна самостійно класифікувати загальні обсяги зворотних вод на «нормативно-очищені», «забруднені без очищення» та «недостатньо очищені», спираючись

на історію спрацювання тригерів перевищення ГДС у розрахунковому модулі протягом року.

Вимога до ергономіки та цілісності даних диктує впровадження системного інструменту «Data Cloning». Інтерфейс підготовки звіту повинен містити функцію ініціації перенесення статичної інфраструктурної топології (географічні координати, дозвільні характеристики точок забору та водовипусків) із затвердженого звіту минулого періоду у поточний чернетковий статус нового звітного періоду [2]. Це бізнес-обмеження спрямоване на мінімізацію ризиків помилок ручного введення та забезпечення наскрізної простежуваності інфраструктурних змін.

Завершальним етапом пайплайну звітності є процедура криптографічного захисту та експорту. Система не має права дозволяти експорт даних у незахищених текстових форматах. Вона повинна інтегрувати функціонал накладання кваліфікованого електронного підпису (КЕП) безпосередньо на згенерований масив даних. Зашифрований пакет передається через API або завантажується на Єдиний державний вебпортал електронних послуг «Портал Дія» чи Портал електронних послуг Держводагентства. Також система повинна підтримувати управління статусами життєвого циклу звіту, дозволяючи переведення поданого звіту у статус «Коригування» шляхом формування виправленого (уточненого) пакету даних у період до 01 березня [3].

1.5 Аналіз існуючих аналогів

Для всебічного та об'єктивного оцінювання існуючих рішень на ринку програмного забезпечення у сфері екологічного моніторингу та управління охороною праці, промисловою і екологічною безпекою (відомих у міжнародній практиці як EHS — Environment, Health, and Safety), доцільно провести детальну категоризацію доступних продуктів. Архітектурний та функціональний аналіз дозволяє розподілити їх на три основні класи: державні та відомчі макрорівневі системи, комерційні ERP-модулі глобального корпоративного рівня, та закордонні спеціалізовані інформаційно-аналітичні платформи оцінки якості вод. Кожен із цих класів формувався під впливом різних стейкхолдерів та бізнес-вимог, що визначає їхні сильні сторони та критичні обмеження.

Державні інформаційні ресурси, такі як національний портал «ЕкоСистема» та електронні сервіси Держводагентства, архітектурно спроектовані як системи макрорівня (Рисунок 1.2-1.3). Їхнім основним призначенням є збір, агрегація та довгострокове зберігання консолідованої звітності для виконання функцій державного контролю та формування макроекономічної статистики. Портал Держводагентства приймає річну форму

№ 2ТП-водгосп виключно у вигляді попередньо розрахованих, підсумкових результатів із застосуванням кваліфікованого електронного підпису [7].

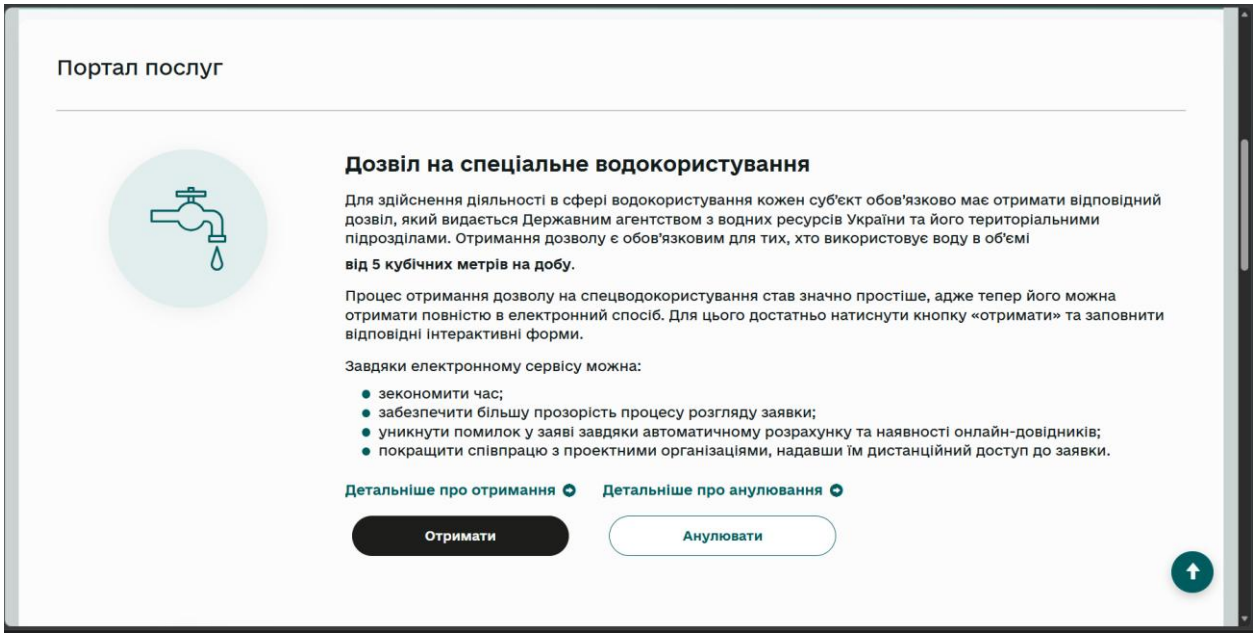


Рисунок 1.2 - Демонстрація порталу «ЕкоСистема»

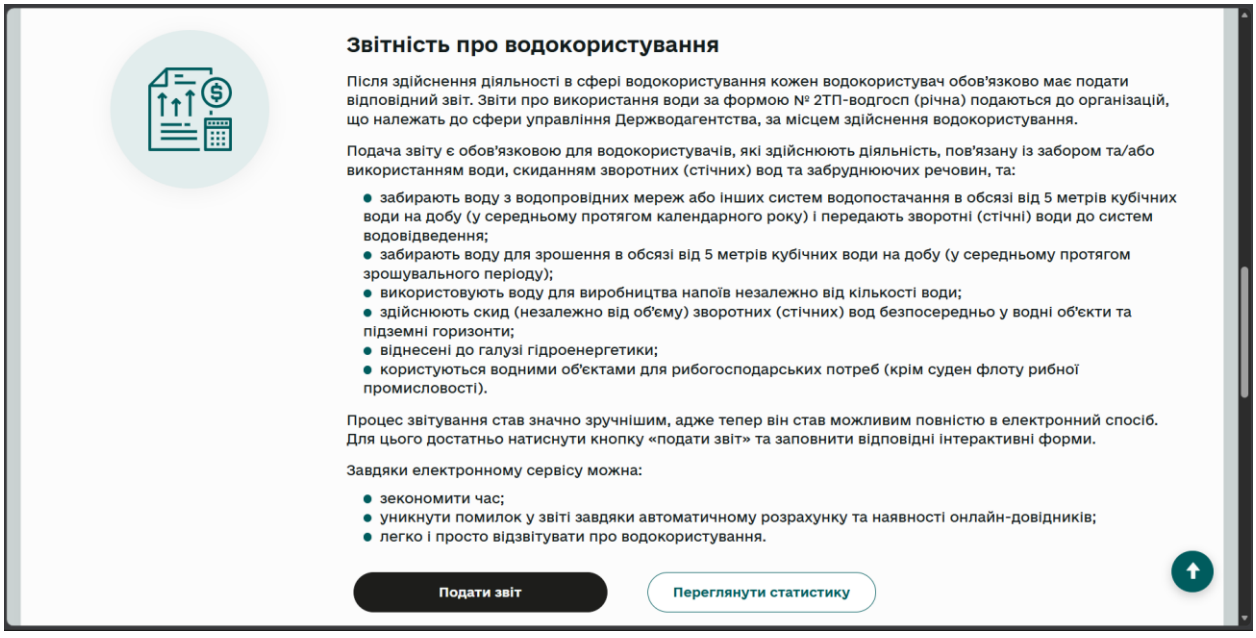


Рисунок 1.3 – Демонстрація порталу «ЕкоСистема»

Архітектура таких систем реалізує парадигму приймача даних. Вони очікують готових агрегованих масивів і не містять інструментарію для щоденного первинного обліку чи проміжної математичної обробки показників. Обчислення мас скидів та зіставлення їх із

дозволами виконуються поза межами системи, що унеможливлює наскрізну відстежуваність даних на мікрорівні підприємства.

Платформи фіксують екологічний стан ретроспективно, що виключає можливість оперативного моніторингу. Вони позбавлені рушіїв обробки правил у режимі реального часу, які б дозволяли автоматично порівнювати показники поточної проби води з динамічними нормативами гранично допустимих скидів (ГДС) та генерувати системні сповіщення про відхилення (Рисунок 1.4-1.7).

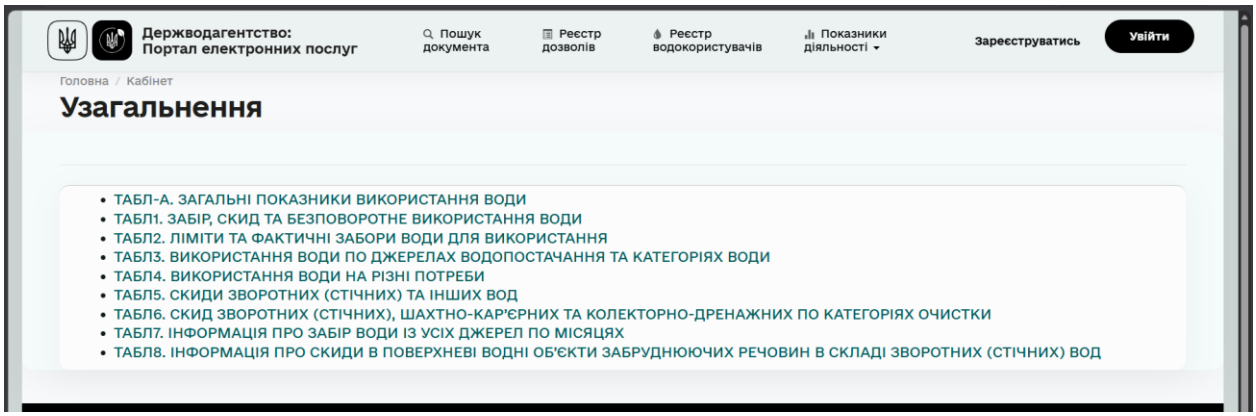


Рисунок 1.4 – Демонстрація процесу формування звіту на порталі «ЕкоСистема»

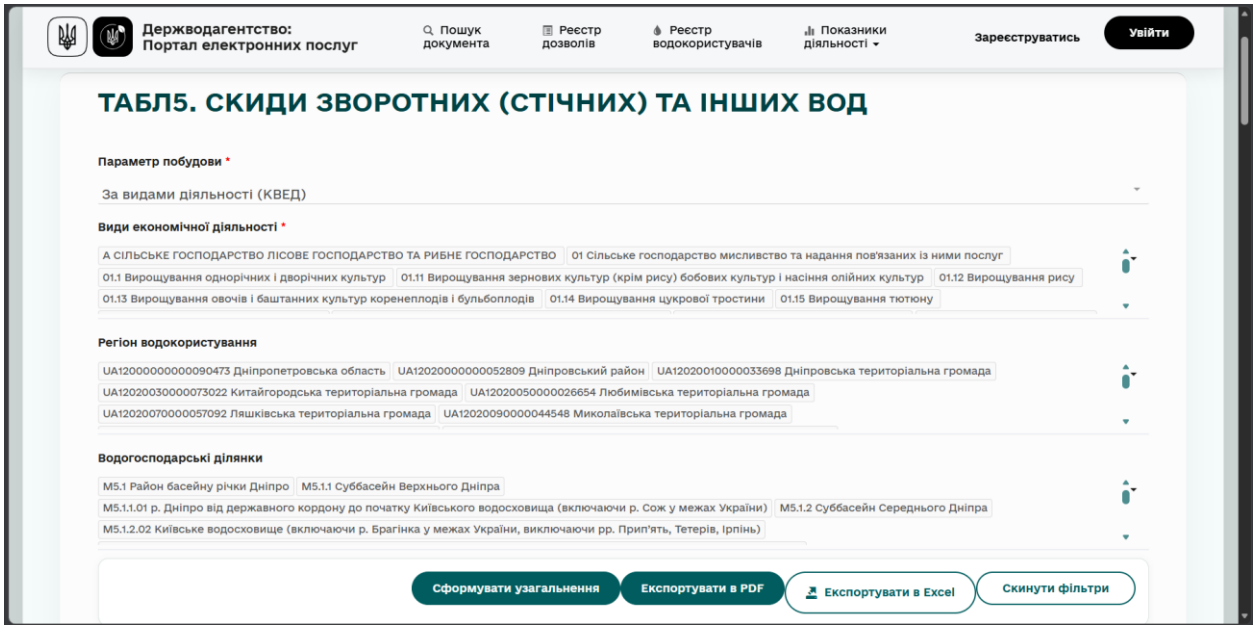
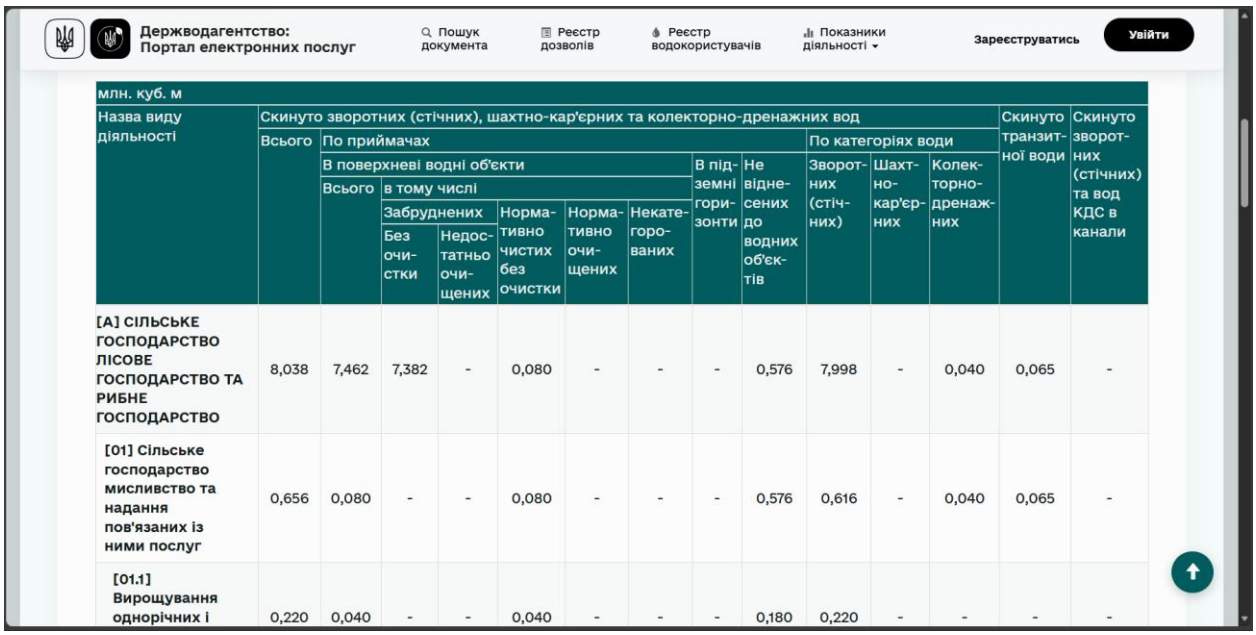


Рисунок 1.5 – Демонстрація процесу формування звіту на порталі «ЕкоСистема»



млн. куб. м

Назва виду діяльності	Скинуто зворотних (стічних), шахтно-кар'єрних та колекторно-дренажних вод												Скинуто транзитної води	Скинуто зворотних (стічних) та вод КДС в канали
	Всього	По приймачах								По категоріях води				
		В поверхневі водні об'єкти						В підземні горизонти	Не віднесених до водних об'єктів	Зворотних (стічних)	Шахтно-кар'єрних	Колекторно-дренажних		
		Всього	в тому числі											
			Забруднених		Нормативно чистих без очистки	Нормативно очищених	Некатегорованих							
			Без очистки	Недостатньо очищених										
[A] СІЛЬСЬКЕ ГОСПОДАРСТВО ЛІСОВЕ ГОСПОДАРСТВО ТА РИБНЕ ГОСПОДАРСТВО	8,038	7,462	7,382	-	0,080	-	-	-	0,576	7,998	-	0,040	0,065	-
[01] Сільське господарство мисливство та надання пов'язаних із ними послуг	0,656	0,080	-	-	0,080	-	-	-	0,576	0,616	-	0,040	0,065	-
[01.1] Вирощування однорічних і	0,220	0,040	-	-	0,040	-	-	-	0,180	0,220	-	-	-	-

Рисунок 1.6 – Демонстрація процесу формування звіту на порталі «ЕкоСистема»

Функціональним бар'єром для інтеграції є жорстке обмеження довідників. Архітектура державних сервісів не підтримує ведення деталізованих локальних баз хімічних мікрокомпонентів за ідентифікаторами CAS, які є базовою вимогою для щоденної роботи заводської або акредитованої лабораторії.

Клас комерційних ERP-систем представлений корпоративними рішеннями з інтегрованими модулями управління екологічною безпекою (EHS), такими як продукти SAP або Oracle (Рисунок 1.7-1.8). Архітектура цих систем забезпечує управління даними речовин, відстеження їхнього життєвого циклу та нативну інтеграцію з виробничими процесами [16].

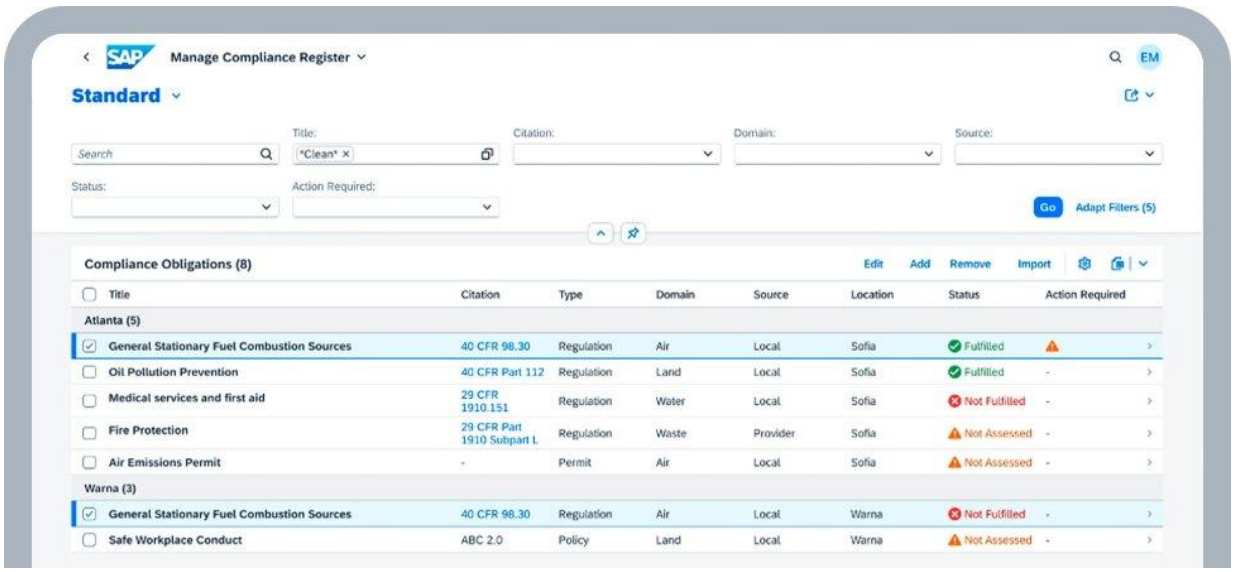


Рисунок 1.7 – Демонстрація модуля EHS продукту SAP

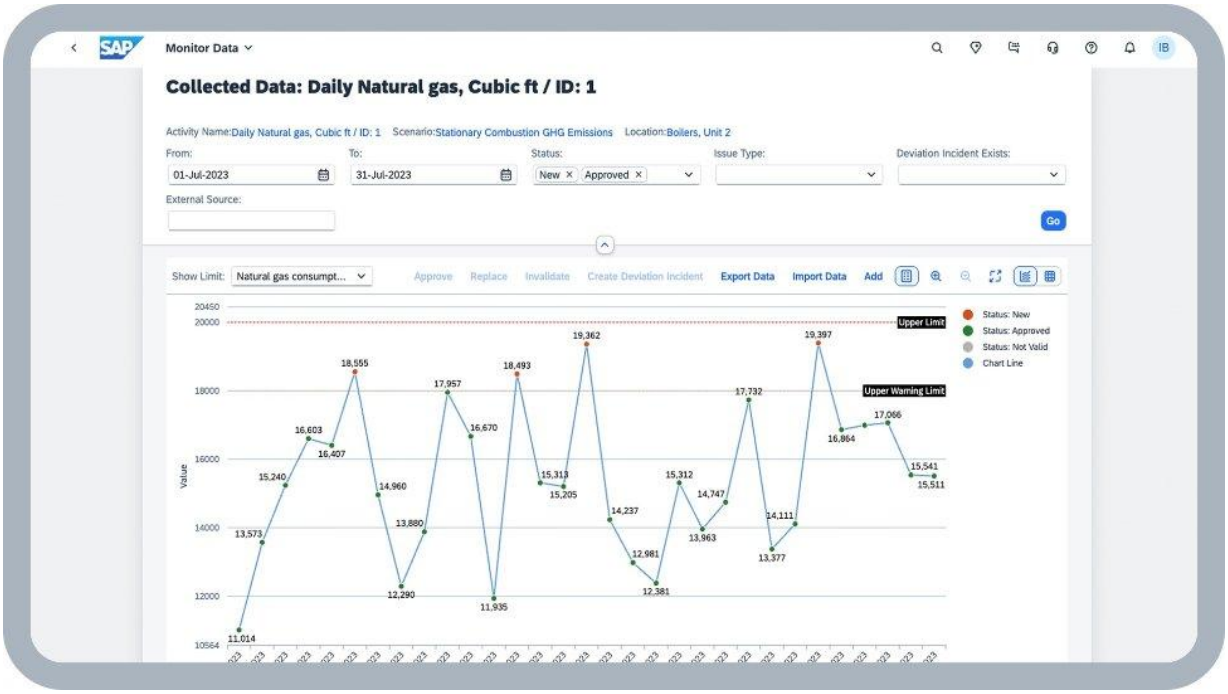


Рисунок 1.8 – Демонстрація модуля EHS продукту SAP

Ключовим бар'єром для локального впровадження є висока сукупна вартість володіння (TCO). Розгортання EHS-модулів передбачає значні витрати на ліцензування, адаптацію інфраструктури та реінжиніринг суміжних бізнес-процесів. Це робить їх використання економічно недоцільним для вирішення вузькоспеціалізованих завдань моніторингу на рівні окремого підприємства чи водоканалу.

Значним обмеженням виступає відсутність локалізації регламентованої звітності. Базова конфігурація звітів корпоративних платформ орієнтована на стандарти США (EPA) або ЄС (E-PRTR). Автоматична генерація української статистичної форми № 2ТП-водгосп неможлива без створення нових сутностей у базі даних та масштабної кастомізації платформи через механізми Z-розробки.

Вбудовані розрахункові алгоритми цих систем є несумісними зі специфічними вітчизняними методиками нормування. Імплементація розрахунку нормативів гранично допустимого скидання (ГДС) на основі методик розбавлення та врахування фонової якості води вимагає розробки унікальної кастомної програмної логіки поверх стандартного функціоналу.

Закордонні спеціалізовані інформаційно-аналітичні платформи (наприклад, інфраструктура WISE) та лабораторні інформаційні системи (LIMS) архітектурно спроектовані для управління екологічними даними за стандартами ЄС (Рисунок 1.9-1.10).

Вони забезпечують нативну підтримку ідентифікації речовин за CAS-номерами та автоматизований облік за європейськими екологічними нормативами якості (EQS) [17].

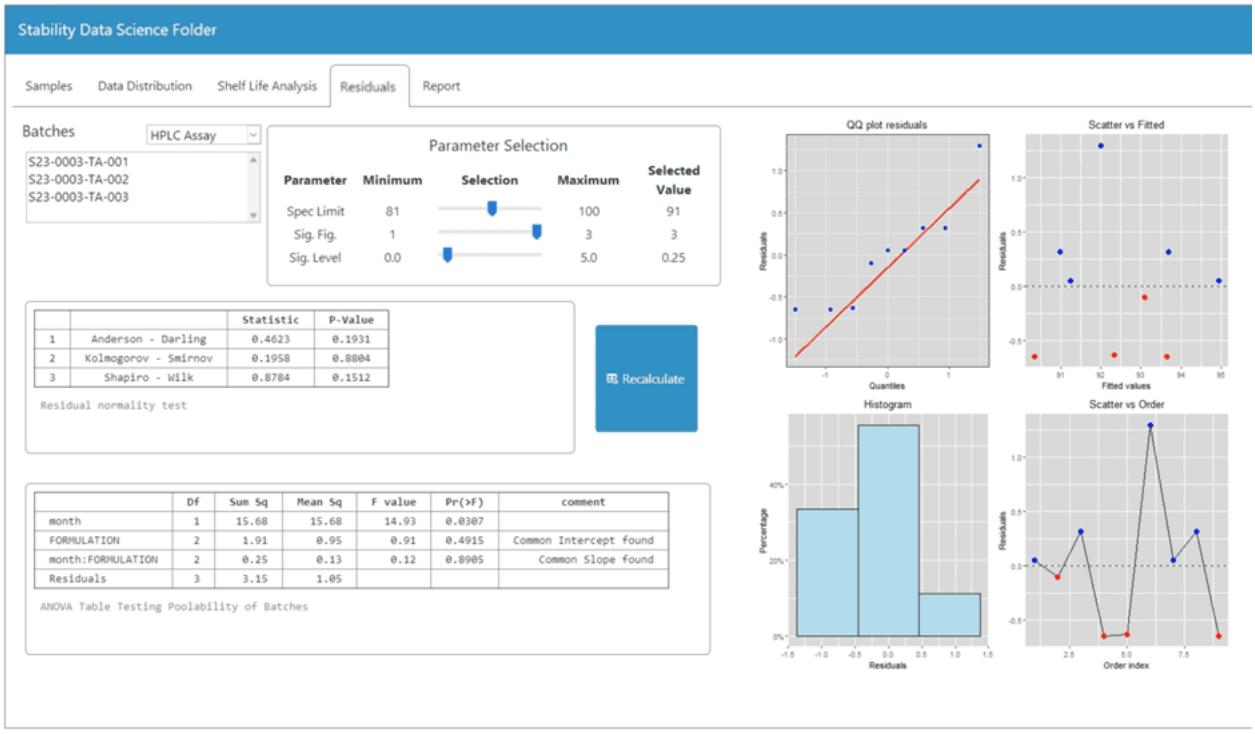


Рисунок 1.9 – Демонстрація продукту LabWare LIMS 2026

Рисунок 1.10 – Демонстрація продукту LabWare LIMS 2026

Основним бар'єром для впровадження цих систем на вітчизняних підприємствах є несумісність базових розрахункових алгоритмів з українською парадигмою нормування. Закордонні платформи не підтримують специфічні вітчизняні методики обчислення гранично допустимих скидів (ГДС), які базуються на гідрологічних параметрах водного об'єкта та динамічних фонових концентраціях забруднювачів.

Такі системи позбавлені локалізованої підсистеми звітності, оскільки їхня архітектура орієнтована на експорт даних у форматах XML для загальноєвропейських баз. Вони принципово не здатні автоматично генерувати регламентовану вітчизняну форму № 2ТП-водгосп або податкову звітність. Додатковим функціональним обмеженням для локального використання є відсутність інтегрованих національних довідників та територіальних класифікаторів, зокрема КАТОТТГ.

Критичним недоліком виступає архітектурна невідповідність вимогам перехідного періоду. Платформи спроектовані під єдину систему європейського екологічного нормування і не містять технічних рішень для ведення обов'язкового в Україні подвійного обліку. Відсутність такого функціоналу унеможливлює паралельну верифікацію результатів лабораторних досліджень за європейськими стандартами EQS та діючими вітчизняними лімітами ГДК/ГДС.

Обґрунтування розробки інформаційної системи як нового архітектурного рішення потребує порівняльного аналізу існуючих класів програмного забезпечення. Зіставлення базується на технічних, функціональних та експлуатаційних критеріях, що випливають із визначених бізнес-вимог та специфіки українського екологічного законодавства перехідного періоду. Оцінювання структуровано за чотирма критеріями, які охоплюють повний життєвий цикл обробки інформації — від первинної ідентифікації хімічних речовин до формування кінцевої регламентованої звітності (Таблиця 1.5).

Таблиця 1.4 Зведена матриця порівняльного аналізу архітектурних та функціональних характеристик систем екологічного моніторингу водних ресурсів

Критерій оцінювання	Державні та відомчі системи	ERP EHS-модулі	Закордонні платформи
Архітектурна роль	Кінцевий приймач та макроагрегатор	Комплексне управління підприємством	Екоаналітика (виконання директив ЄС)
Довідник речовин	Частково	Повністю	Повністю
Математичні тренди	Відсутні на мікрорівні	Підтримується	Повністю підтримується
Генерація 2ТП-водгосп (ETL)	Лише прийом та валідація	Відсутня	Відсутня
Вартість володіння (TCO)	Безкоштовно (фінансується державою)	Екстремально висока	Середня/Висока (SaaS/On-prem)
Відповідність нормам України	Повна	Низька	Нульова

Продовження таблиці 1.4

Критерій оцінювання	Державні та відомчі системи	ERP EHS-модулі	Закордонні платформи
Автоматизація роботи еколога	Низький	Дуже високий	Середній
Верифікація ГДК / розрахунок ГДС (Укр.)	Відсутня	Потребує кастомізації (Z-розробка)	Відсутня
Підтримка стандартів EQS / PNEC	Лише на макрорівні	Можлива через налаштування	Реалізовано (базово + EQR)

Результати порівняльного аналізу свідчать, що наявні державні та комерційні рішення не забезпечують необхідного рівня автоматизації щоденної операційної діяльності еколога підприємства. Відсутність спеціалізованого функціоналу для локального розрахунку нормативів ГДС, неможливість автоматичної генерації специфічних звітних форм та висока сукупна вартість володіння корпоративними ERP-модулями роблять існуючі аналоги непридатними для комплексного задоволення бізнес-вимог.

1.6 Обґрунтування розробки системи

Аналіз існуючих рішень свідчить про відсутність програмного забезпечення проміжного рівня, здатного автоматизувати операційну діяльність екологічних підрозділів та забезпечити інтеграцію з державними порталами приймання звітності. Для виконання вимог перехідного періоду екологічного законодавства проектована система повинна імплементувати низку архітектурних рішень.

База даних проектується з використанням міжнародних кодів CAS як первинних ключів ідентифікації хімічних речовин. Це усуває дублювання записів у локальних довідниках та забезпечує архітектурну сумісність із європейськими реєстрами даних.

Розрахунковий рушій системи забезпечує паралельний фоновий контроль результатів лабораторних досліджень. Обробка даних відбувається одночасно за двома

нормативними парадигмами: вітчизняними методиками розрахунку гранично допустимих скидів (ГДК/ГДС) та європейськими екологічними нормативами якості води (EQS/PNEC).

Для генерації регламентованої звітності впроваджується підсистема ETL. Алгоритм здійснює автоматичну трансформацію та агрегацію транзакційних даних у формати, сумісні з логікою державної статистичної форми № 2ТП-водгосп.

Зниження сукупної вартості володіння (ТСО) досягається проєктуванням кросплатформного вебдодатка. Архітектура модульного моноліту дозволяє локальне розгортання на апаратній інфраструктурі підприємства, що є раціональною альтернативою впровадженню корпоративних ERP-систем.

1.7 Завдання, які має вирішувати об'єкт проєктування

Для забезпечення ефективного управління водними ресурсами цільова інформаційна система має вирішувати комплекс взаємопов'язаних завдань. Першим завданням є автоматизований збір та централізована реєстрація даних водокористування, що охоплює результати лабораторних вимірювань хімічного стану вод та показники приладів обліку.

Другим завданням виступає здійснення динамічного контролю за дотриманням нормативів гранично допустимих скидів та лімітів дозвільної документації на спеціальне водокористування із забезпеченням миттєвої ідентифікації нормативних перевищень.

Третім завданням є автоматизація обчислювальних процесів, зокрема розрахунок річних гідрологічних балансів підприємств у розрізі окремих об'єктів моніторингу та видів водогосподарських робіт.

Четверте завдання полягає у забезпеченні аналітичної підтримки шляхом ретроспективного формування багаторічних трендів водоспоживання та просторової візуалізації екологічного навантаження на водні екосистеми з використанням державних територіальних класифікаторів.

П'ятим завданням є автоматична генерація регламентованої статистичної звітності та підготовка валідованих масивів даних для експорту відповідно до стандартів національних контролюючих органів та європейських інформаційних платформ.

2 АРХІТЕКТУРА РІШЕННЯ

2.1 Парадигма клієнт-серверної взаємодії та розподілені обчислення

Проектування інформаційної системи екологічного моніторингу та державного обліку водокористування вимагає створення стійкої, масштабованої та безпечної програмної інфраструктури, яка здатна надійно функціонувати в умовах промислового підприємства чи басейнового управління. В основі архітектури рішення лежить класична клієнт-серверна модель взаємодії, яка логічно та фізично розділяє систему на презентаційний рівень (клієнтську частину) та шар обробки бізнес-логіки й управління даними (серверну частину).

Такий розподіл обов'язків є критично важливим для інформаційної системи екологічного моніторингу з кількох причин. По-перше, він забезпечує безпеку та ізоляцію персистентного сховища даних (бази даних). Клієнтські пристрої лаборантів, екологів та державних інспекторів ніколи не отримують прямого доступу до таблиць бази даних. Вся взаємодія здійснюється виключно через стандартизований прикладний програмний інтерфейс (API), який розгорнуто на захищеному сервері додатків. Це дозволяє централізовано впроваджувати політики безпеки, валідації та автентифікації на стороні сервера.

По-друге, клієнт-серверна модель дозволяє раціонально розподілити обчислювальне навантаження в мережі. У системі моніторингу водних ресурсів значна частина операцій пов'язана з інтерактивною візуалізацією часових рядів, відображенням просторових об'єктів на карті (ГІС-компоненти) та заповненням об'ємних ієрархічних форм звітів. Перенесення логіки рендерингу, управління станом інтерфейсу та первинної фільтрації даних на клієнтські пристрої (за рахунок використання потужностей браузера) дозволяє значно знизити навантаження на сервер. Серверна інфраструктура вивільняється для виконання своїх безпосередніх, обчислювально важких завдань: розрахунку гідрологічних балансів, перевірки перевищень лімітів ГДК за складними алгоритмами та пакетної генерації регламентованих звітів.

2.2 Односторінковий додаток як механізм клієнтського середовища

Клієнтську частину системи реалізовано у вигляді односторінкового додатка (Single Page Application, SPA). На відміну від традиційного підходу Server-Side Rendering (SSR), за якого при кожній дії користувача сервер повністю генерує нову HTML-сторінку та передає її через мережу, парадигма SPA передбачає одноразове завантаження базової оболонки веб-

додатка (HTML, CSS та JavaScript-коду) під час першого звернення. Усі подальші переходи між розділами системи, фільтрація даних та відправка форм здійснюються асинхронно, без перезавантаження сторінки в браузері. Клієнтська частина обмінюється із сервером виключно сирими, структурованими даними у форматі JSON за допомогою асинхронних HTTP-запитів.

Застосування SPA є технологічно та ергономічно обґрунтованим рішенням для екологічної інформаційної системи через максимальна інтерактивність та реактивність інтерфейсу, адже користувач (наприклад, інженер-еколог) отримує миттєвий відгук на свої дії при роботі з аналітичними панелями. Динамічна зміна часових інтервалів або вибір конкретних забруднювальних речовин на графіку призводить до миттєвого перемальовування відповідного UI-компонента.

Оскільки через мережу передається лише корисне навантаження у вигляді JSON-об'єктів, обсяг трафіку знижується в рази порівняно з SSR. Це особливо важливо для промислових об'єктів або лабораторій, які можуть використовувати мобільні канали зв'язку або мережі з обмеженою пропускною здатністю для передачі щоденних показників.

Візуалізація точок контролю (випусків стічних вод, водозаборів) вимагає інтеграції складних JavaScript-бібліотек. У SPA-додатку картографічний контекст ініціалізується один раз і зберігається в пам'яті клієнта протягом усієї сесії. При перегляді деталей конкретного випуску карта фокусується на відповідній геокоординаті без необхідності повторного завантаження важких шарів карти з геопорталів, що суттєво прискорює роботу аналітика.

2.3 Архітектурний патерн CQRS

Однією з головних інженерних проблем при проєктуванні високонавантажених інформаційних систем є використання єдиної доменної моделі для операцій читання та запису даних (класичний підхід CRUD). У системі екологічного моніторингу ця проблема проявляється особливо гостро через різку асиметрію між процесами збору даних та їхнього аналітичного споживання.

З одного боку, система постійно приймає великий потік транзакційних даних: щоденні показники витратомірів про об'єми забору води, результати хімічних аналізів стічних вод від лабораторії (що вимагають миттєвої перевірки на відповідність лімітам ГДК/ГДС). Ці операції запису вимагають суворої консистентності, перевірки складних бізнес-правил, нормалізації таблиць бази даних для запобігання аномаліям та використання ACID-транзакцій для забезпечення цілісності.

З іншого боку, екологи та державні інспектори щодня генерують складні аналітичні запити: будують довгострокові тренди зміни якості води, розраховують інтегральні маси скидів, зводять квартальні та річні баланси водовикористання. Ці операції читання вимагають вилучення мільйонів історичних записів із бази даних, їхнього групування, умовної агрегації та проекції у складні деревоподібні структури (наприклад, для таблиць форми № 2ТП-водгосп).

Якщо реалізувати обидва ці потоки на базі єдиної моделі даних, виникає конфлікт інтересів. Модель, спроєктована для швидкого та консистентного запису, є вкрай неефективною для побудови складних аналітичних вибірок, оскільки вимагає виконання десятків важких SQL-операцій JOIN над глибоко нормалізованими таблицями. І навпаки, спроба денормалізувати базу даних для прискорення аналітики створює критичні ризики порушення цілісності при реєстрації первинних вимірювань та суттєво уповільнює транзакції через необхідність оновлення численних індексів.

Інтеграція архітектурного патерна CQRS (Command Query Responsibility Segregation) розділяє систему на два незалежних тракти обробки інформації. Тракт команд (Command Stack) обробляє виключно операції мутації стану: створення, оновлення та видалення сутностей. Він оперує глибоко нормалізованою моделлю запису, яка оптимізована для забезпечення вимог ACID, захисту бізнес-інваріантів та швидкої фіксації транзакцій. Команди формулюються в термінах намірів користувача і повертають лише підтвердження успіху або ідентифікатор створеного об'єкта.

Тракт запитів (Query Stack) виконує операції читання та отримання інформації. Він використовує модель, яка ізольована від бізнес-логіки та оптимізована для швидкого повернення проекцій даних клієнту. Запити застосовують прямі механізми доступу до бази, минаючи складні доменні сутності. Такий підхід усуває накладні витрати на гідратацію об'єктів, відстеження змін та навігацію по графу зв'язків у пам'яті сервера додатків.

2.4 Патерн Mediator як централізована шина маршрутизації

Сегрегація відповідальності на команди та запити автоматично утворює велику кількість атомарних класів-повідомлень (Commands/Queries) та відповідних їм класів-обробників (Handlers). Необхідно організувати ефективний, гнучкий та слабозв'язаний механізм передачі управління від зовнішніх точок входу (HTTP-контролерів REST API) до внутрішніх обробників бізнес-логіки.

Пряме впровадження залежностей у конструктори контролерів спричиняє жорстке зв'язування та порушує принцип єдиної відповідальності (SRP). Перевантаження класу

численними сервісами (репозиторіями, валідаторами) робить код громіздким і унеможливлює ізольоване модульне тестування.

Для вирішення цієї проблеми в архітектуру системи впроваджено поведінковий патерн Mediator. Патерн Mediator функціонує як централізована внутрішньосистемна шина повідомлень. Контролер презентаційного шару взаємодіє виключно з одним абстрактним інтерфейсом — IMediator. Отримавши HTTP-запит, контролер десеріалізує його у відповідний об'єкт повідомлення та відправляє його в шину.

Посередник, використовуючи контейнер інверсії управління (IoC), динамічно знаходить зареєстрований у системі обробник для цього конкретного типу повідомлення, ініціалізує його залежності, передає повідомлення на виконання та повертає результат назад контролеру.

Використання цього підходу забезпечує стратегічні переваги для розробки та еволюції системи. Досягається абсолютний декоплінг (слабка зв'язність), за якого контролери та обробники бізнес-логіки повністю ізольовані та спілкуються виключно через абстрактні контракти повідомлень. Це дозволяє модифікувати обробники сценаріїв використання без внесення змін у код контролерів API.

Графи залежностей спрощуються, оскільки кожен контролер отримує єдину залежність — IMediator. Додатково забезпечується можливість масштабування. Наявність посередника дозволяє перевести обробку команд із синхронного режиму на асинхронні черги повідомлень (RabbitMQ, MassTransit) під час переходу на мікросервісну архітектуру без зміни інтерфейсу взаємодії з клієнтом.

2.5 Реалізація парадигми тонких контролерів

У багатьох традиційних веб-системах контролери беруть на себе занадто багато обов'язків: валідацію вхідних даних, координацію транзакцій бази даних, виконання бізнес-логіки та формування HTTP-відповідей. Такий антипатерн називається "товстими контролерами". Він ускладнює супровід системи, оскільки бізнес-логіка виявляється нерозривно заплутаною з інфраструктурними деталями веб-фреймворку (заголовки HTTP, об'єкти запиту, контекст сесії), що унеможливлює її повторне використання у консольних додатках, фонових задачах чи інтеграційних тестах.

Архітектура розроблюваної системи суворо дотримується парадигми тонких контролерів. Контролери Web API спроєктовані як максимально прості, анемічні інфраструктурні адаптери, відповідальність яких обмежена виключно комунікаційними завданнями на рівні протоколу HTTP.

Життєвий цикл обробки мережевого запиту в тонкому контролері системи є суворо регламентованим і складається з таких фаз:

1. Під час прийому запиту контролер перехоплює вхідний HTTP-запит на певному маршруті.
2. Тіло запиту (JSON) десеріалізується у строго типізований об'єкт команди або запиту, відбувається трансляція з термінів мережевого протоколу в терміни об'єктної моделі програми.
3. Сформоване повідомлення делегується в посередник IMediator. Контролер переходить у стан асинхронного очікування, звільняючи потік веб-сервера для обробки інших мережевих запитів.
4. Отриманий від обробника результат обчислень контролер упаковує у відповідний DTO.
5. Контролер повертає клієнту HTTP-відповідь із семантично коректним статус-кодом.

У тонких контролерах системи повністю відсутні умовні оператори розгалуження, логіка звернення до репозиторіїв чи пряме управління транзакціями. Код є шаблонним і легко піддається автоматичній генерації або покриттю інтеграційними тестами.

2.6 Ізольовані класи-обробники сценаріїв використання

Перенесення прикладної логіки з контролерів вимагає створення надійного середовища для її виконання. У розробленій системі вся бізнес-поведінка інкапсульована в ізольованих класах-обробниках. Кожен клас відповідає за реалізацію одного єдиного сценарію використання системи, строго дотримуючись принципу єдиної відповідальності (Single Responsibility Principle, SRP).

Запропонована архітектура усуває регресійні помилки, характерні для традиційних сервісних класів. Кодові бази сценаріїв використання фізично ізольовані в окремих файлах та класах. Внесення змін в один сценарій не впливає на суміжні процеси, оскільки вони не мають спільного стану в оперативній пам'яті.

Розділення логіки забезпечує простоту модульного тестування. Кожен обробник тестується ізольовано шляхом ініціалізації класу та передачі імітацій (моків) необхідних репозиторіїв та інфраструктурних сервісів через конструктор.

Структура проекту значно підвищує читабельність та відстежуваність коду. Функціональні можливості системи прямо відображаються у переліку класів у директоріях Commands та Queries. Кожен клас у шарі бізнес-логіки виступає безпосереднім інженерним втіленням конкретної бізнес-вимоги.

2.7 Domain-Driven Design, DDD

Проектування фізичної схеми бази даних інформаційної системи моніторингу водних ресурсів базується на принципах предметно-орієнтованого проектування (DDD). Предметна область екологічного моніторингу та державного обліку водокористування була декомпонована на ізольовані обмежені контексти. Кожен контекст інкапсулює власну бізнес-логіку, правила валідації та структуру збереження даних, що відображено у реляційній схемі бази даних СУБД MS SQL Server.

Довідковий блок містить структури даних, що характеризуються низькою частотою змін, високою частотою зчитування та слугують єдиним джерелом істини для класифікації та групування інформації у системі. До цього блоку відносяться:

1. Реєстр забруднювальних речовин (Pollutant) є центральним елемент довідкового блоку. Він моделює хімічні, фізичні та біологічні агенти, які підлягають екологічному контролю. Структура таблиці включає унікальні міжнародні та національні коди: CasNumber (код Chemical Abstracts Service), EuNumber (код Європейського Союзу) та StateCode (національний екологічний код забруднювача). На рівні СУБД налаштовано обмеження унікальності (UNIQUE) для цих полів, що виключає виникнення проблеми синонімії та дублювання речовин.

2. Довідник галузей економіки (IndustryBranch) класифікує підприємства-водокористувачі за видами економічної діяльності. Для відображення ієрархічної структури (секції, розділи, групи) застосовано патерн Adjacency List (список суміжності) із визначенням рекурсивного зовнішнього ключа ParentBranchId, що дозволяє виконувати аналітичну агрегацію екологічного навантаження на макроекономічному рівні.

3. Довідник річкових басейнів (RiverBasin) відображає гідрографічне районування території України відповідно до Водного кодексу. Басейни мають просторову ієрархію: район річкового басейну, суббасейн, водогосподарська ділянка.

4. Довідник видів робіт водокористування (WaterManagementWork) класифікує технологічні процеси, що супроводжуються забором чи скидом води. Таблиця містить нормативні коефіцієнти безповоротного водоспоживання та відсотки втрат води, які використовуються в обчислювальному ядрі системи.

Блок суб'єктів та об'єктів моделює топологію мережі екологічного контролю та антропогенного навантаження на водні екосистеми. Він включає:

1. Підприємства-водокористувачі (MonitoringSubject) виступають коренем агрегації для суб'єктів господарювання, що здійснюють спеціальне водокористування. Таблиця

містить код ЄДРПОУ (з налаштованим обмеженням контрольної суми для валідації вводу) та код КАТОТТГ (для просторової прив'язки до адміністративно-територіального устрою).

2. Фізичні об'єкти моніторингу (MonitoringObject) – це фізичні точки в просторі (водозабори, свердловини, випуски зворотних вод), де здійснюється відбір проб або вимірювання витрати води. Таблиця містить географічні атрибути (Latitude, Longitude), що дозволяють виконувати геопросторові запити та інтегруватися з ГІС-компонентами інтерфейсу. Кожен об'єкт має обов'язковий зовнішній ключ до таблиці RiverBasin.

Блок нормативів та дозволів відповідає за збереження регуляторної та дозвільної бази, яка визначає легітимні рамки антропогенного впливу:

1. Екологічні стандарти якості (EnvironmentalQualityStandard), які моделюють екологічні нормативи, зокрема стандарти якості (EQS) та гранично допустимі концентрації (ГДК) для речовин з таблиці Pollutant. Для таблиці налаштовано підтримку темпоральності (системного версіонування): кожне значення нормативу містить поля ValidFrom та ValidTo, що дозволяє системі коректно оцінювати якість води в історичній ретроспективі.

2. Екологічні дозволи підприємств (EcologicalPermit), які є кореннем агрегації, що фіксує юридичне право підприємства на спеціальне водокористування. Таблиця містить номер дозволу, дату видачі та термін дії.

Транзакційний блок консолідує дані, що характеризуються найвищою частотою операцій запису, виконуючи роль аудиторського сліду системи, а саме:

1. Лабораторні протоколи якості води (WaterQualityProtocol) реєструють результати інструментально-лабораторного аналізу проб води на об'єкті MonitoringObject. Таблиця містить метадані (виконавець, методика вимірювань, дата відбору проби) та посилання на вкладені результати вимірювань.

2. Результати хімічного аналізу протоколу (QualityParameterResult) відображають деталізації, що містить фактичні концентрації речовин (Pollutant) у пробі та показники інструментальної похибки.

3. Звіти водоспоживання (WaterUsageReport) агрегують дані державного статистичного спостереження про об'єми забраної, використаної та скинутої води підприємством MonitoringSubject за календарний рік.

4. Записи звіту водоспоживання (WaterUsageRecord) надають деталізації, що міститять помісячні об'єми води в розрізі джерел постачання, категорій якості води та видів робіт.

5. Акти розрахунку гідрологічного балансу (HydrologicalBalanceAct) відображають документ, що містить зведені дані про приплив, відтік, випаровування та антропогенне вилучення водних мас, а також розраховану дельту розбіжностей балансу.

2.8 Логічна схема та зв'язки таблиць бази даних

Перенесення доменної моделі в реляційну схему базується на суворому дотриманні правил посилальної цілісності та мінімізації надлишковості інформації. Усі зв'язки між таблицями розділені на три логічні категорії: зв'язки агрегації, композиційні зв'язки та довідкові зв'язки.

Зв'язки агрегації ілюструють підпорядкованість життєвих циклів сутностей, де дочірні записи мають самостійне бізнес-значення, проте їх існування зумовлене наявністю батьківської сутності.

Таблиця 2.1 Специфікація зв'язків агрегації в реляційній схемі бази даних.

Батьківська таблиця (1)	Дочірня таблиця (N)	Тип реляційного зв'язку	Стратегія деструкції	Обґрунтування в предметній області моніторингу вод
MonitoringSubject	EcologicalPermit	Неідентифікуючий	ON DELETE RESTRICT	Збереження регуляторної історії суб'єкта контролю.[1]
MonitoringSubject	WaterUsageReport	Неідентифікуючий	ON DELETE RESTRICT	Забезпечення незмінності державної статистичної звітності.[1]
MonitoringObject	WaterQualityProtocol	Неідентифікуючий	ON DELETE RESTRICT	Запобігання втраті просторового та гідрологічного контексту вимірювань.[1]
RiverBasin	MonitoringObject	Неідентифікуючий	ON DELETE RESTRICT	Збереження топологічної цілісності гідрографічної мережі.[1]

Використання стратегії ON DELETE RESTRICT є фундаментальним архітектурним вибором. Воно забороняє видалення запису підприємства або точки моніторингу з реєстру, якщо з ними асоційований хоча б один історичний документ. Це гарантує безстрокове збереження правової цілісності та аудиторського сліду для податкових перевірок та формування Державного водного кадастру.

Композиційні зв'язки репрезентують жорстку залежність типу "частина-ціле". У парадигмі DDD це відповідає внутрішній структурі агрегату, де дочірні елементи позбавлені глобальної ідентичності поза межами свого кореня агрегації. У реляційній схемі такі зв'язки реалізуються як ідентифікуючі відношення із каскадним видаленням (ON DELETE CASCADE) або через використання композитних первинних ключів.

Таблиця 2.2 Специфікація композиційних зв'язків у базі даних.

Корінь агрегації	Композиційний елемент	Кардинальність	Стратегія первинного ключа дочірньої таблиці
EcologicalPermit	PermitLimit	1 : N	Композитний первинний ключ: [PermitId, PollutantId]
WaterUsageReport	WaterUsageRecord	1 : N	Сурогатний первинний ключ: RecordId з обов'язковим унікальним індексом
WaterQualityProtocol	QualityParameterResult	1 : N	Композитний первинний ключ: [ProtocolId, PollutantId]

Використання композитного первинного ключа [PermitId, PollutantId] на рівні схеми гарантує, що в межах одного дозволу неможливо створити два конфліктуючі ліміти для однієї і тієї ж хімічної речовини, запобігаючи аномаліям на етапі проектування бази даних. Політика ON DELETE CASCADE гарантує, що при анулюванні або видаленні батьківського документа всі підпорядковані записи деталізації знищуються автоматично, унеможливлюючи появу "орфанних" (осиротілих) записів у базі даних.

Довідкові зв'язки забезпечують денормалізацію операційних даних шляхом створення реляційних посилань на сутності Master Data.

Специфічною інженерною рисою довідкових зв'язків у системі є їхня семантична незмінність у контексті зафіксованої транзакції. Схема бази даних спроектована таким чином, що зовнішні ключі, які вказують на довідники, не підлягають модифікації операцією UPDATE після затвердження документа. Якщо в затвердженому лабораторному протоколі виникає необхідність змінити ідентифікатор досліджуваної речовини, бізнес-правила відхиляють пряму зміну запису. Замість цього створюється нова ревізія протоколу. Це запобігає прихованій фальсифікації історичних екологічних даних та гарантує надійність аудиторського сліду.

2.9 Проектування API та інтеграційна взаємодія

Проектування прикладного програмного інтерфейсу (API) інформаційної системи базується на глибокій інтеграції архітектурного стилю REST та патерну CQRS. В основу розробленої архітектури покладено фундаментальний принцип логічного та фізичного розділення операцій модифікації стану системи (команд) та операцій отримання даних (запитів).

Класичний підхід REST, орієнтований виключно на CRUD-операції з ресурсами, визнано недостатнім для реалізації складної бізнес-логіки предметної області екологічного моніторингу, оскільки наміри користувача не завжди можуть бути однозначно виражені через стандартні дієслова HTTP (POST, GET, PUT, DELETE). Відповідно, простір імен API спроектовано з урахуванням семантики предметної області та суворо поділено на дві логічні групи: транзакційні та аналітичні контролери.

2.10 Транзакційні контролери

Транзакційні контролери призначені виключно для обробки операцій мутації стану (команд CQRS). Вони відповідають за ініціацію процесів створення, оновлення або видалення доменних агрегатів. Ці компоненти оперують моделями запису, які оптимізовані для забезпечення абсолютної цілісності даних, підтримки доменних інваріантів та швидкого виконання транзакцій у базі даних.

Ресурсні контролери не здійснюють безпосереднього доступу до бази даних, виконуючи роль фасадів, які приймають HTTP-запити, десеріалізують їх у структури команд та передають на виконання до внутрішньої шини IMediator. Це забезпечує повну ізоляцію доменного рівня від транспортного протоколу HTTP.

Приклади транзакційних маршрутів у системі:

• POST /api/v1/monitoring-subjects – створення нового суб'єкта моніторингу;

• POST /api/v1/monitoring-subjects/{id}/permits – реєстрація нового екологічного дозволу;

• PUT /api/v1/water-usage-reports/{id} – оновлення показників річного звіту водоспоживання;

• DELETE /api/v1/water-quality-protocols/{id} – анулювання лабораторного протоколу (м'яке видалення).

2.11 Аналітичні контролери

Аналітичні контролери спроектовані для обробки запитів на читання, які вимагають багатовимірної агрегації, фільтрації та проекції даних з багатьох таблиць одночасно. Вимоги системи передбачають необхідність формування складних зведений звітів, побудови часових графіків водоспоживання та обчислення екологічних індексів у реальному часі.

Для вирішення цієї проблеми виокремлено спеціалізовану групу аналітичних контролерів, маршрутизація яких будується навколо операцій вибірки, де основним ідентифікатором простору імен є маршрут /api/v1/analytics. Особливістю реалізації цих контролерів є застосування виключно HTTP-методу GET для отримання оптимізованих структур моделей читання. Повна відсутність операцій зміни стану у цьому просторі імен гарантує безпеку та ідемпотентність запитів.

Приклади аналітичних маршрутів у системі:

• GET /api/v1/analytics/water-usage/trends – отримання щорічних трендів водоспоживання для побудови клієнтських графіків;

• GET /api/v1/analytics/river-basins/{id}/quality-index – розрахунок комплексного індексу забруднення для басейну річки;

• GET /api/v1/analytics/monitoring-subjects/{id}/permit-compliance – порівняльний аналіз фактичних скидів із дозволеними лімітами.

Такий архітектурний дизайн гарантує, що ресурсоємні операції обчислення аналітичних показників не впливатимуть на пропускну здатність системи під час масового надходження транзакційних даних від користувачів або зовнішніх сенсорів.

2.12 Формати передачі даних та стандартизація

Механізми інформаційного обміну між клієнтським додатком та серверною частиною жорстко стандартизовано для забезпечення абсолютної передбачуваності

контрактів, підвищення ефективності серіалізації та гарантування безпеки мережевого зв'язку.

Передача даних між зовнішнім клієнтським та внутрішнім серверним рівнями абстракції здійснюється виключно через спеціалізовані об'єкти передачі даних — DTO. DTO проектуються як максимально прості, ізольовані структури, які не містять жодної бізнес-логіки, методів поведінки або зв'язків з інфраструктурою бази даних.

Доменні сутності та агрегати предметної області ніколи не серіалізуються безпосередньо у HTTP-відповідь API. Пряма серіалізація сутностей порушує інкапсуляцію доменного рівня та розкриває внутрішню структуру бази даних, створюючи загрози інформаційній безпеці. Такий підхід формує жорстку залежність між фізичною схемою даних та зовнішнім контрактом системи, через що будь-яка зміна структури таблиць призводить до руйнування клієнтського застосунку. Використання об'єктів передачі даних (DTO) усуває ці недоліки, дозволяючи змінювати внутрішню доменну модель або схему бази даних незалежно від контрактів API та забезпечуючи повну зворотну сумісність для клієнтів.

3 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Проєктування структурної схеми системи

Для формалізації статичної архітектури інформаційної системи та визначення міжкомпонентних меж розроблено структурну схему застосунку. Декомпозиція системи на модулі дозволяє зафіксувати склад підсистем, визначити зони їхньої відповідальності та логіку об'єднання функціональних елементів у ізольовані архітектурні шари (Рисунок 3.1).

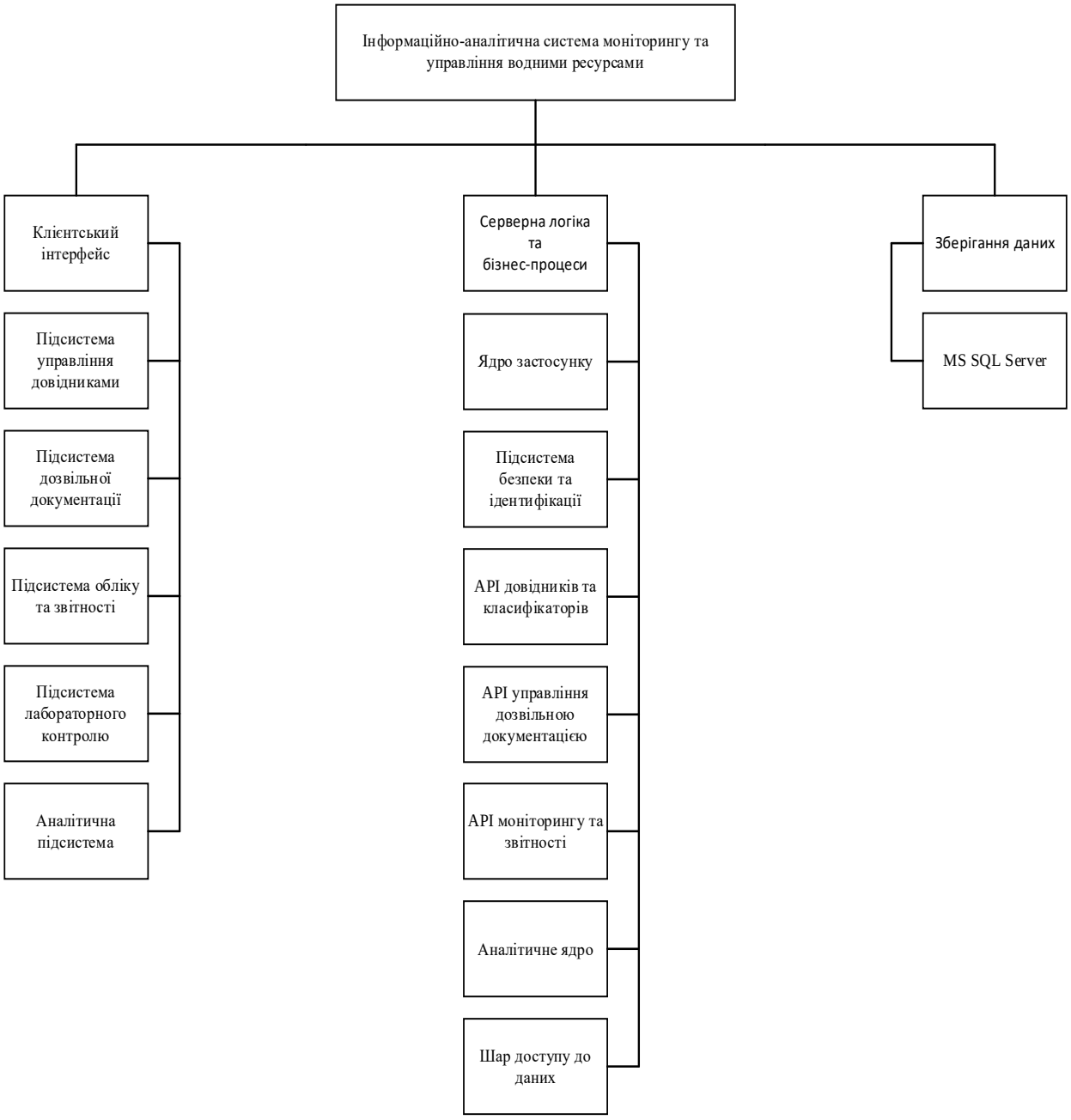


Рисунок 3.1– Демонстрація структурної схеми системи

Схема інформаційної системи побудована за багаторівневим принципом. Компонентна структура наочно демонструє поділ системи на три ключові блоки: клієнтський інтерфейс, серверна логіки та рівень персистенції даних. Таке проєктування забезпечує суворе дотримання принципу розділення відповідальності, де кожен рівень взаємодіє із суміжними виключно через стандартизовані інтерфейси та протоколи обміну даними.

3.2 Екологічне навантаження

Алгоритм формування звіту з екологічного навантаження призначений для просторового аналізу обсягів водокористування в межах визначеної територіальної громади за вказаний проміжок часу (Рисунок 3.2).

У контексті бізнес-процесів його виконання розпочинається з ідентифікації цільових водних об'єктів, які територіально належать до заданого регіону згідно з класифікатором КАТОТТГ. За необхідності радіус аналізу може бути точково звужений до конкретної станції спостереження. Далі система звертається до бази даних і здійснює агрегацію фактичних показників з усіх зареєстрованих звітів підприємств за обраний звітний період.

Отримані масиви інформації проходять етап класифікації для кожного окремого об'єкта моніторингу. Алгоритм підсумовує загальний об'єм забору свіжої води та загальний об'єм скиду зворотних вод, після чого розраховує гідрологічне сальдо. Ця різниця між забором та скидом є ключовим індикатором, який відображає чисте безповоротне вилучення ресурсу з екосистеми.

На фінальному етапі згруповані дані ранжуються за рівнем забору від найбільшого до найменшого, акцентуючи увагу на найбільш експлуатованих водоймах. Результатом роботи алгоритму є консолідована вибірка, яка дозволяє оцінити сумарний антропогенний вплив на конкретні водні об'єкти регіону без необхідності ручного пошуку та зведення розрізнених звітів окремих водокористувачів.

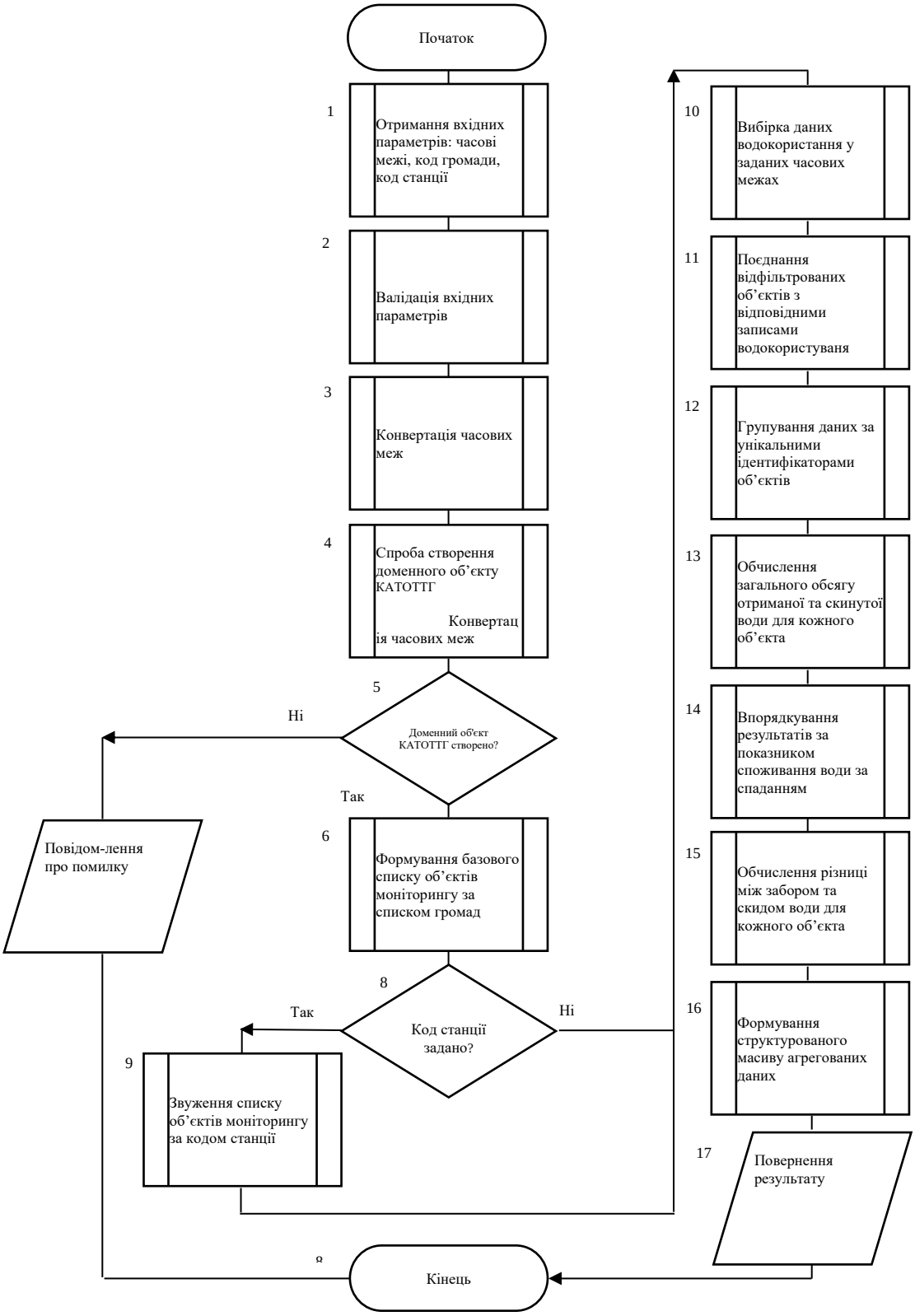


Рисунок 3.2 – Блок-схема алгоритму екологічного навантаження

Візуалізація просторового розподілу екологічного навантаження забезпечує користувача інструментом оперативної оцінки антропогенного впливу на водні об'єкти в межах обраних територіальних одиниць. Завдяки інтеграції з класифікатором КАТОТТГ, система дозволяє локалізувати зони найінтенсивнішого безповоротного вилучення ресурсів, трансформуючи сирі табличні дані звітів у наочний аналітичний зріз для підтримки прийняття управлінських рішень.

3.3 Динаміка водоспоживання

Алгоритм формування звіту з динаміки водоспоживання призначений для ретроспективного аналізу використання водних ресурсів конкретним підприємством протягом заданого багаторічного періоду (Рисунок 3.3).

У контексті бізнес-процесів виконання алгоритму розпочинається з ідентифікації суб'єкта господарювання за його унікальним державним кодом ЄДРПОУ. Після підтвердження наявності підприємства в екологічному реєстрі система здійснює вибірку всіх затверджених звітів про водокористування, які належать цьому суб'єкту і потрапляють у вказаний часовий діапазон.

Далі алгоритм проводить хронологічну агрегацію даних, групуючи всі фактичні вимірювання в розрізі окремих років. Для кожного звітного року обчислюється сумарний об'єм забраної свіжої води та загальний об'єм скинутих зворотних вод, після чого визначається річний гідрологічний баланс. Цей баланс відображає обсяг безповоротно спожитого ресурсу на виробничі потреби.

Сформований масив даних сортується за зростанням років. Результатом роботи алгоритму є консолідований аналітичний зріз, який дозволяє екологам підприємства та державним інспекторам відстежувати тренди водоспоживання, оцінювати ефективність впровадження водозберігаючих технологій та виявляти системні аномалії в експлуатації природних ресурсів.

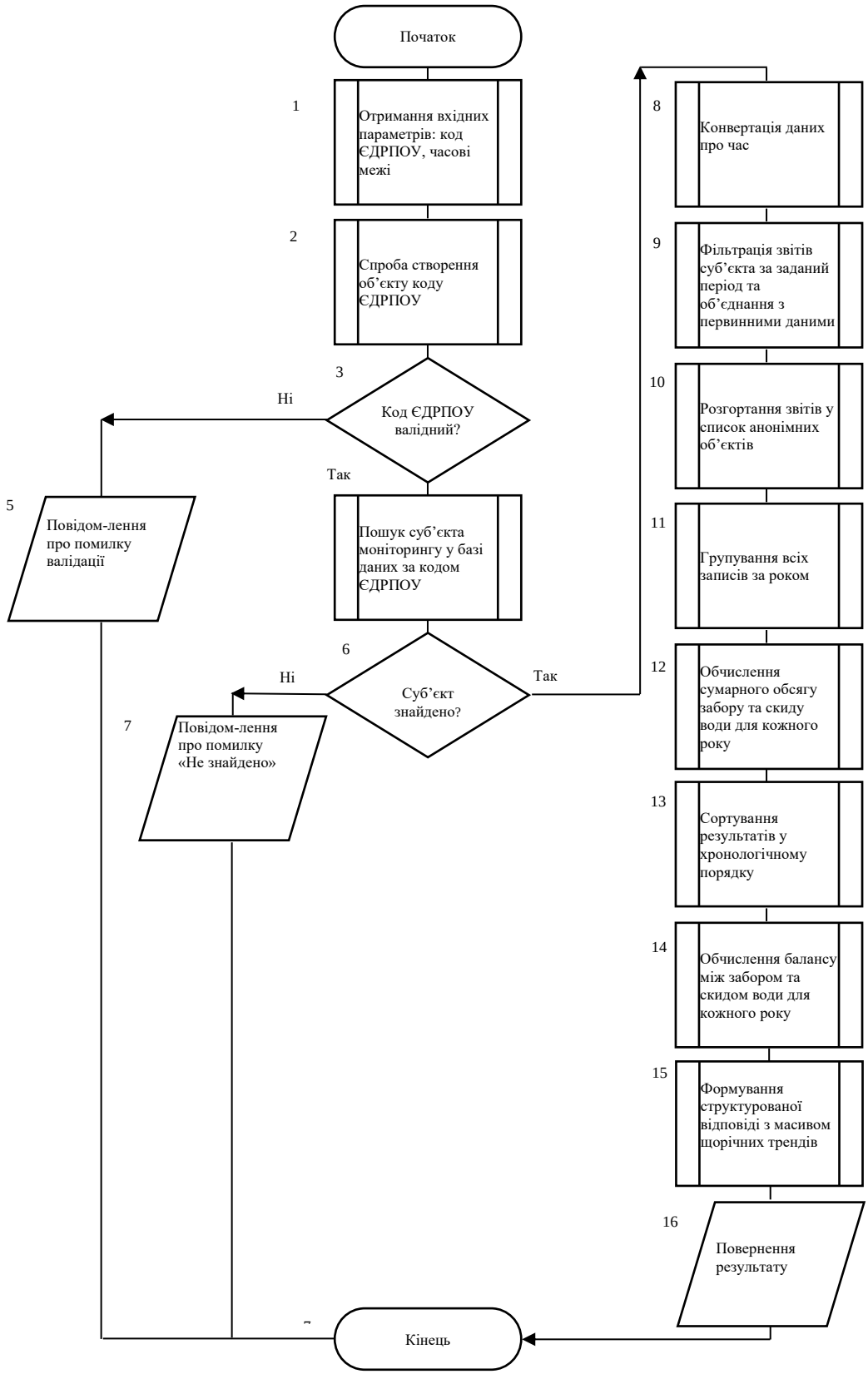


Рисунок 3.3 – Блок-схема алгоритму динаміки водоспоживання

Графічне відображення ретроспективної динаміки водоспоживання унаочнює багаторічні тренди взаємодії конкретного підприємства з водними екосистемами. Розрахунок річного гідрологічного балансу та його представлення у вигляді часових рядів дозволяє екологічним службам оцінювати реальну ефективність впроваджених технологій зворотного водопостачання та виявляти системні аномалії в експлуатації природних ресурсів протягом тривалих часових проміжків.

3.4 Зведений баланс водовикористання

Алгоритм формування балансового звіту за екологічним дозволом призначений для автоматизованого контролю за дотриманням підприємством встановлених квот на спеціальне водокористування протягом визначеного року (Рисунок 3.4).

У контексті бізнес-процесів його виконання розпочинається з пошуку діючих дозвільних документів. Система перевіряє наявність дозволів для вказаного суб'єкта господарювання, які були чинними в межах обраного звітного розрахункового року. Після ідентифікації валідних документів алгоритм звертається до підсистеми звітності і вилучає всі затверджені фактичні показники водокористування підприємства за цей самий період, ігноруючи чернетки.

Отримані масиви даних проходять етап ієрархічного групування та порівняльного аналізу. Алгоритм розмежовує інформацію в розрізі окремих дозволів, об'єктів моніторингу (точок забору чи скиду води) та конкретних видів водогосподарських робіт. Для кожного нормативного ліміту, закріпленого в дозволі, система підсумовує фактично використані або скинуті об'єми води згідно з поданими звітами. Далі обчислюється різниця між затвердженою квотою та фактичним використанням, що дозволяє миттєво визначити залишок ліміту або зафіксувати факт його перевищення.

Результатом роботи алгоритму є деталізований багаторівневий звіт. Цей документ надає екологам підприємств та державним інспекторам прозору картину використання водних ресурсів, повністю автоматизуючи процес звірки дозволених та фактичних показників і виключаючи необхідність ручного зіставлення паперової дозвільної документації зі звітною.

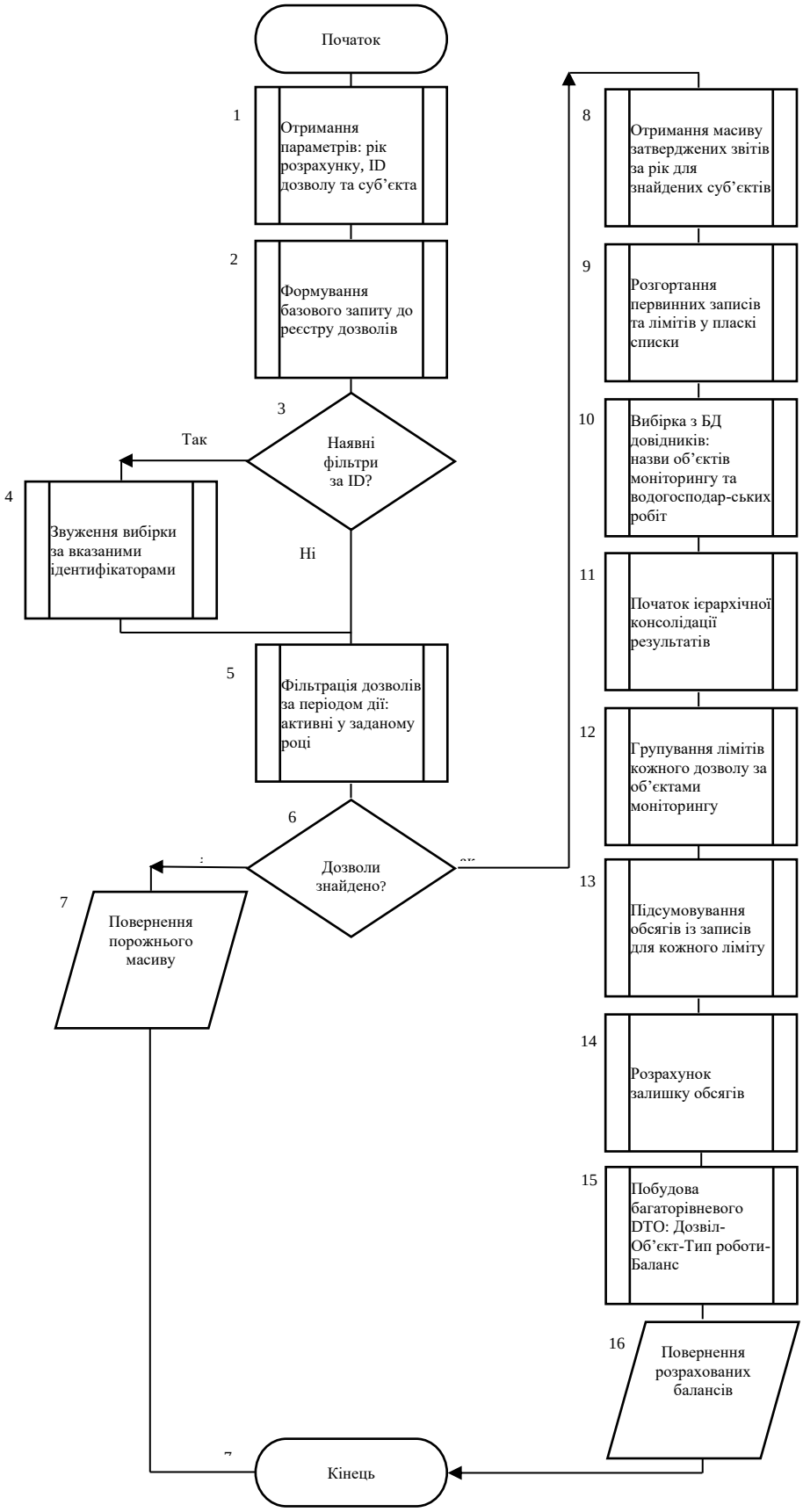


Рисунок 3.4 – Блок-схема алгоритму зведеного балансу водовикористання

Екранна форма моніторингу балансу екологічних дозволів виступає ключовим інструментом операційного контролю за дотриманням встановлених державою квот на спеціальне водокористування. Автоматичне обчислення залишків лімітів у розрізі об'єктів та видів робіт дозволяє запобігти перевищенню дозволених об'ємів забору чи скиду води, повністю усуваючи необхідність ручного зіставлення паперової документації з поточними показниками діяльності.

3.5 Створення діаграми послідовностей

Для деталізації динаміки взаємодії архітектурних компонентів системи під час виконання ключових аналітичних операцій розроблено діаграму послідовностей. Цей тип моделювання дозволяє зафіксувати точний хронологічний порядок обміну запитами та відповідями між користувацьким інтерфейсом, прикладним рівнем серверної логіки та реляційним сховищем даних.

Для прикладу розглянемо процес формування звіту гідрологічного балансу водоспоживання (Рисунок 3.5). Наочне відображення життєвого циклу запиту на часових лініях дозволяє чітко розмежувати зони відповідальності між клієнтським застосунком, що ініціює подію, ізольованими обробниками бізнес-процесів на стороні сервера та базою даних, яка виконує транзакційні операції.

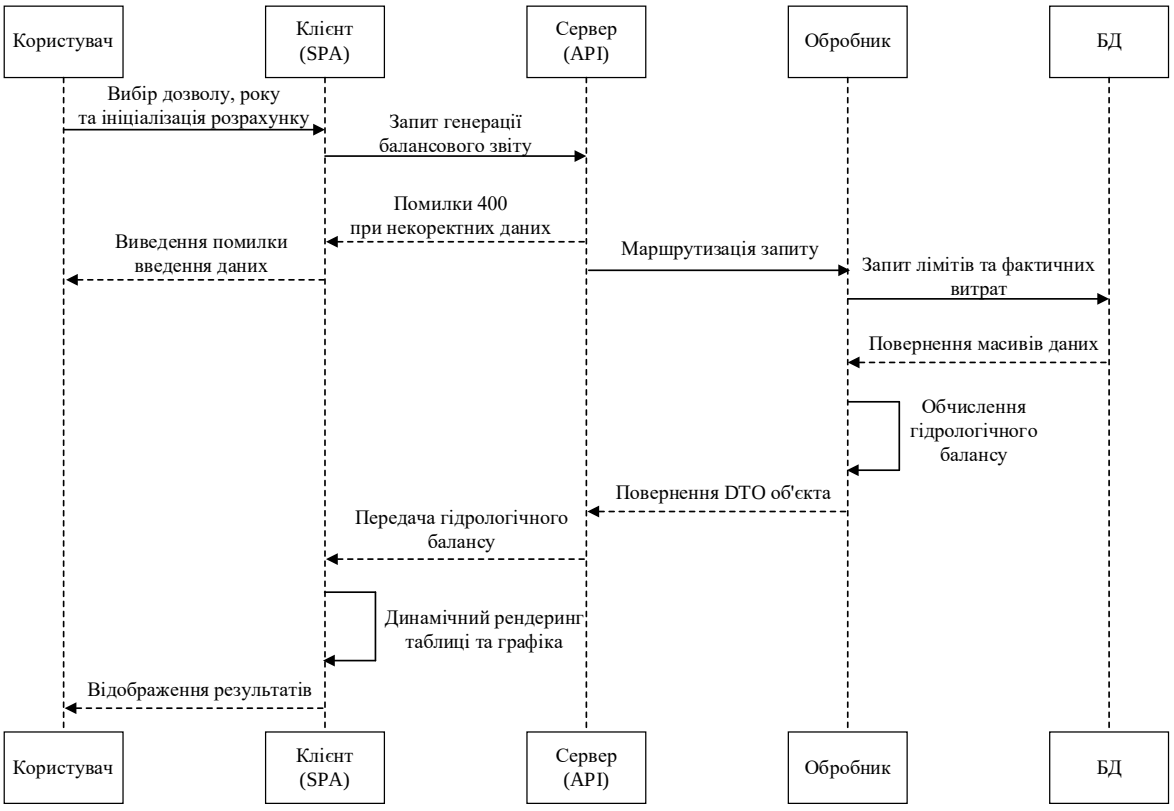


Рисунок 3.5 – Діаграма послідовностей процесу формування звіту гідрологічного балансу водовикористання

Подібний підхід до проектування забезпечує можливість превентивного виявлення логічних помилок у структурі міжкомпонентних зв'язків та оптимізації алгоритмів асинхронного обміну даними ще до етапу безпосередньої програмної реалізації системи.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Вибір технічного стеку

Розробка масштабованої, відмовостійкої та безпечної інформаційної системи екологічного моніторингу вимагає застосування суворих інженерних підходів. Комбінація екосистеми .NET 8+ із використанням бібліотеки MediatR для реалізації CQRS та Entity Framework Core для об'єктно-реляційного відображення складної доменної моделі гарантує архітектурну чистоту та створює надійний бар'єр проти помилок бізнес-логіки. Використання статичної типізації TypeScript у поєднанні з декларативною міццю React дозволяє будувати високоінтерактивні інтерфейси, мінімізуючи ризики розсинхронізації контрактів даних. Використання транзакційно-орієнтованої СУБД MS SQL Server із застосуванням умовної агрегації є фундаментальною гарантією цілісності, швидкодії та консистентності екологічних баз даних [18, 19, 20, 21].

4.2 Реалізація серверної частини

Шар представлення, що відповідає за обробку HTTP-запитів та взаємодію з клієнтськими застосунками:

```
public static class DependencyInjection
{
    public static IServiceCollection AddPresentation(this
     IServiceCollection services)
    {
        services.AddCors(options =>
        {
            options.AddPolicy("AllowViteFrontend", policyBuilder
     =>
            {

     policyBuilder.WithOrigins("http://localhost:5173")
                  .AllowAnyHeader()
                  .AllowAnyMethod()
                  .AllowCredentials();
            });
        });

        services.AddMapping();
        return services;
    }
}
```

Логіка конфігурації сервісів бізнес-рівня. Завданням цього методу є ініціалізація інфраструктури для впровадження CQRS-конвеєра та автоматизованої перевірки вхідних даних.

```
public static IServiceCollection AddApplication(this
     IServiceCollection services)
{

     services.AddValidatorsFromAssembly(Assembly.GetExecutingAsse
     mbly());

    services.AddMediatR(cfg =>
    {

     cfg.RegisterServicesFromAssembly(typeof(DependencyInjection)
     .Assembly);
        cfg.AddOpenBehavior(typeof(ValidationBehavior<,>));
    });

    return services;
}
```

Реєстрація інфраструктурних залежностей, зокрема для роботи з базою даних:

```
public static class DependencyInjection
{
    public static IServiceCollection AddInfrastructure(this
     IServiceCollection services, ConfigurationManager
     configuration)
    {
        services.AddDbContext<AquaMonitorDbContext>(options =>

     options.UseSqlServer(configuration.GetConnectionString("Defa
     ultConnection"),
                b =>
     b.MigrationsAssembly(typeof(AquaMonitorDbContext).Assembly.F
     ullName)
            ));

        services.AddSingleton<IDateTimeProvider,
     DateTimeProvider>();
        services.AddAuth(configuration);

        services.AddScoped<IUserRepository, UserRepository>();
        services.AddScoped<IApplicationDbContext,
     AquaMonitorDbContext>();
        return services;
    }
```

```
private static IServiceCollection AddAuth(this
IServiceCollection services, ConfigurationManager
configuration)
{
    var jwtSettings = new JwtSettings();

    configuration.Bind(JwtSettings.SectionName,
jwtSettings);

    services.AddSingleton(Options.Create(jwtSettings));
    services.AddSingleton<IJwtTokenGenerator,
JwtTokenGenerator>();

    services.AddAuthentication(defaultScheme:
JwtBearerDefaults.AuthenticationScheme)
        .AddJwtBearer(options =>
        {
            options.TokenValidationParameters = new
TokenValidationParameters
            {
                ValidateAudience = true,
                ValidateIssuer = true,
                ValidateLifetime = true,
                ValidateIssuerSigningKey = true,
                ValidIssuer = jwtSettings.Issuer,
                ValidAudience = jwtSettings.Audience,
                IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings.Secr
et))
            };

            // Блок для перехоплення та логування причин
помилки 401
            options.Events = new JwtBearerEvents
            {
                OnAuthenticationFailed = context =>
                {
                    Console.WriteLine($"[AUTH FAILED]
{context.Exception.Message}");
                    return Task.CompletedTask;
                },
                OnChallenge = context =>
                {
                    Console.WriteLine($"[AUTH CHALLENGE]
{context.Error} - {context.ErrorDescription}");
                    return Task.CompletedTask;
                }
            };
        });

    services.AddSingleton<IPasswordHasher,
PasswordHasher>();
```

```
        return services;
    }
}
```

На рівні HTTP-контролерів результати, отримані від диспетчера MediatR, обробляються за допомогою механізму функціонального зіставлення зразків. Наочним прикладом є обробка маршруту в контролері AnalyticsController:

```
[HttpGet("subjects/{edrpouCode}/water-usage-trends")]
public async Task<IActionResult> GetWaterUsageTrends(
    string edrpouCode,
    [FromQuery] int startYear,
    [FromQuery] int endYear,
    CancellationToken cancellationToken)
{
    var query = new GetWaterUsageTrendsQuery(edrpouCode,
     startYear, endYear);
    var result = await _mediator.Send(query, cancellationToken);

    return result.Match(
        trends => Ok(trends),
        errors => Problem(errors)
    );
}
```

4.3 Реалізація клієнтської частини та візуалізація

Ключовим аспектом архітектури клієнтської частини є впровадження механізму управління доступом на основі ролей. Оскільки інформаційна система обробляє конфіденційні дані щодо водоспоживання, скидів та екологічного навантаження, доступ до певних модулів має бути суворо регламентований. Захист маршрутів, які вимагають авторизації та наявності специфічних привілеїв, реалізовано через спеціалізований компонент-обгортку ProtectedRoute, який виконує функцію проміжного програмного забезпечення на рівні маршрутизації.

Фактична реалізація базового компонента захисту маршрутів наведена у наступному фрагменті коду:

```
// @/shared/auth/ProtectedRoute.tsx
import {Navigate, Outlet, useLocation} from 'react-router-dom';
import {useAuth} from "@/shared/auth/useAuth.ts";
import type {RepresentativeRole} from "@/shared/auth/roles.ts";

interface ProtectedRouteProps {
    allowedRoles?: RepresentativeRole;
}
```

```
export const ProtectedRoute = ({ allowedRoles }:
     ProtectedRouteProps) => {
    const { token, roles } = useAuth();
    const location = useLocation();

    // Перевірка наявності активної сесії (токена)
    if (!token) {
        // 'state' зберігає об'єкт location для майбутнього
     navigate() в LoginPage
        return <Navigate to="/login" state={{ from: location }}
     replace />;
    }

    // Перевірка наявності відповідних прав доступу, якщо такі
     вимагаються маршрутом
    if (allowedRoles &&!roles.some(role =>
     allowedRoles.includes(role))) {
        return <Navigate to="/unauthorized" replace />;
    }

    // Рендеринг дочірніх компонентів маршруту за умови успішної
     верифікації
    return <Outlet />;
};
```

Інтеграція даного компонента у глобальну конфігурацію роутера забезпечує строгу ізоляцію приватних сегментів застосунку. Приклад використання у декларативному описі шляхів через об'єктно-орієнтований підхід передбачає обгортання дочірніх маршрутів у ProtectedRoute. Нижче наведено фрагмент конфігурації (з файлу router.tsx), що демонструє захист аналітичних дашбордів:

```
// @/app/router.tsx
import { createBrowserRouter } from 'react-router-dom';
import { ProtectedRoute } from
     '@/shared/auth/ProtectedRoute.tsx';
import { EcologistRoles } from '@/shared/auth/roles.ts';
import { WaterBodyLoadPage } from
     '@/pages/WaterBodyLoadPage.tsx';

export const router = createBrowserRouter(} />,
        children:
    }
]);
```

Надійний та консистентний обмін даними між клієнтським застосунком та серверною частиною є фундаментом для коректного функціонування інформаційної системи. Комунікаційний шар інкапсульовано за допомогою популярної бібліотеки axios,

яка базується на технології XMLHttpRequest та сучасних стандартах Promise. Для централізованого управління мережевими запитами спроектовано єдиний конфігурований екземпляр клієнта apiClient.

Нижче наведено повний вихідний код налаштування екземпляра apiClient:

```
// @/shared/api/apiClient.ts
import axios, { AxiosError } from 'axios';
import { useAuth } from "@/shared/auth/useAuth.ts";
import { AppError, ProblemDetails } from
    "@/shared/api/types.ts";
import { useSubjectStore } from
    "@/shared/store/useSubjectStore.ts";

// Створення ізольованого екземпляра з базовим URL з
    конфігурації
export const apiClient = axios.create({
    baseURL: import.meta.env.VITE_API_URL ||
    'http://localhost:5145',
});

// Налаштування інтерцептора для вихідних запитів
apiClient.interceptors.request.use((config) => {
    // Отримання токена поза контекстом React-компонентів
    const token = useAuth.getState().token;

    if (token && config.headers) {
        config.headers.Authorization = `Bearer ${token}`;
    }

    // Динамічна ін'єкція ідентифікатора мультитенантності
    const activeSubjectId =
    useSubjectStore.getState().activeSubjectId;
    if (activeSubjectId && config.headers) {
        config.headers = activeSubjectId;
    }

    return config;
});

// Налаштування інтерцептора для вхідних відповідей
apiClient.interceptors.response.use(
    (response) => response,
    (error: AxiosError<ProblemDetails>) => {
        // Перехоплення закінчення сесії
        if (error.response?.status === 401) {
            useAuth.getState().logout();
            if (!window.location.pathname.includes('/login')) {
                window.location.href = '/login';
            }
```

```
            return Promise.reject([{ code: 'Auth.Unauthorized',
        description: 'Сесія завершена' }]);
        }

        const data = error.response?.data;

        // Трансформація стандарту RFC 7807 (Problem Details) у
        внутрішній тип AppError
        if (data?.errors) {
            const apiErrors: AppError =
        Object.entries(data.errors).map(([code, messages]) => ({
                code: code,
                description: messages // Береться перше
        повідомлення з масиву помилок для ключа
            }));
            return Promise.reject(apiErrors);
        }

        // Страхувальний блок для невідомих серверних збоїв
        return Promise.reject();
    }
);
```

Такий підхід до перехоплення гарантує, що жоден необроблений виняток від API не призведе до критичного збою клієнтського застосунку. Замість генерування неконтрольованих JavaScript-помилок, функція завжди повертає об'єкт відхиленого промісу (Promise.reject) із суворо структурованим та прогнозованим масивом AppError.

Для забезпечення масштабованості кодової бази та підтримки високої швидкодії системи управління станом реалізовано на основі гібридного підходу. Традиційні монолітні рішення (такі як Redux) часто призводять до надлишкового обсягу шаблонного коду. Натомість, спроектована архітектура строго розділяє глобальний синхронний стан користувацького інтерфейсу та асинхронний стан віддалених даних. Це дозволяє оптимізувати життєвий цикл компонентів та ізолювати логіку кешування мережевих відповідей.

Наступний фрагмент коду демонструє реалізацію глобального сховища з типізацією та персистентністю:

```
// @/shared/store/useSubjectStore.ts
import { create } from 'zustand';
import { persist } from 'zustand/middleware';

// Декларування контракту стану для підтримки TypeScript
interface SubjectState {
    activeSubjectId: string | null;
    setActiveSubjectId: (id: string | null) => void;
```

```
    clearSubject: () => void;
}

// Ініціалізація глобального хука сховища з обгорткою persist
export const useSubjectStore = create<SubjectState>()(
    persist(
        (set) => ({
            activeSubjectId: null,
            // Мутаційні функції для оновлення стану
            setActiveSubjectId: (id) => set({ activeSubjectId:
     id }),
            clearSubject: () => set({ activeSubjectId: null }),
        }),
        { name: 'active-subject-storage' } // Ключ для
     LocalStorage
    )
);
```

Для управління асинхронними даними, які безперервно змінюються на стороні сервера (аналітичні зведення, реєстри суб'єктів), інтегровано бібліотеку @tanstack/react-query.

Фрагмент реалізації виклику асинхронного запиту через useQuery наведено нижче:

```
// Фрагмент з @/pages/WaterUsageTrendPage.tsx
//  Зберігання об'єкта параметрів після проходження локальної
     валідації
const [queryParams, setQueryParams] = useState<{edrpou: string,
     startYear: number, endYear: number} | null>(null);

// Підключення до серверного стану
const { data, isLoading, isError, refetch } = useQuery({
    queryKey: ['water-usage-trends', queryParams],
    queryFn: () => analyticsApi.getWaterUsageTrends(
        queryParams!.edrpou,
        queryParams!.startYear,
        queryParams!.endYear
    ),
    enabled:!!queryParams,  //  Відкладене завантаження:  запит
     блокується, поки queryParams === null
    retry:  false  //  Вимкнення  каскадних  повторів  при  помилках
     (напр. 404, якщо підприємство не знайдено)
});
```

Для централізованого доступу до інструментів обробки та візуалізації агрегованих даних спроектовано сторінку аналітичної панелі управління (Рисунок 4.1). Даний компонент виконує функцію інтегрованого робочого простору (дашборду), який забезпечує

єдину точку входу до спеціалізованих модулів дослідження показників водокористування та оцінки екологічного навантаження на довкілля.

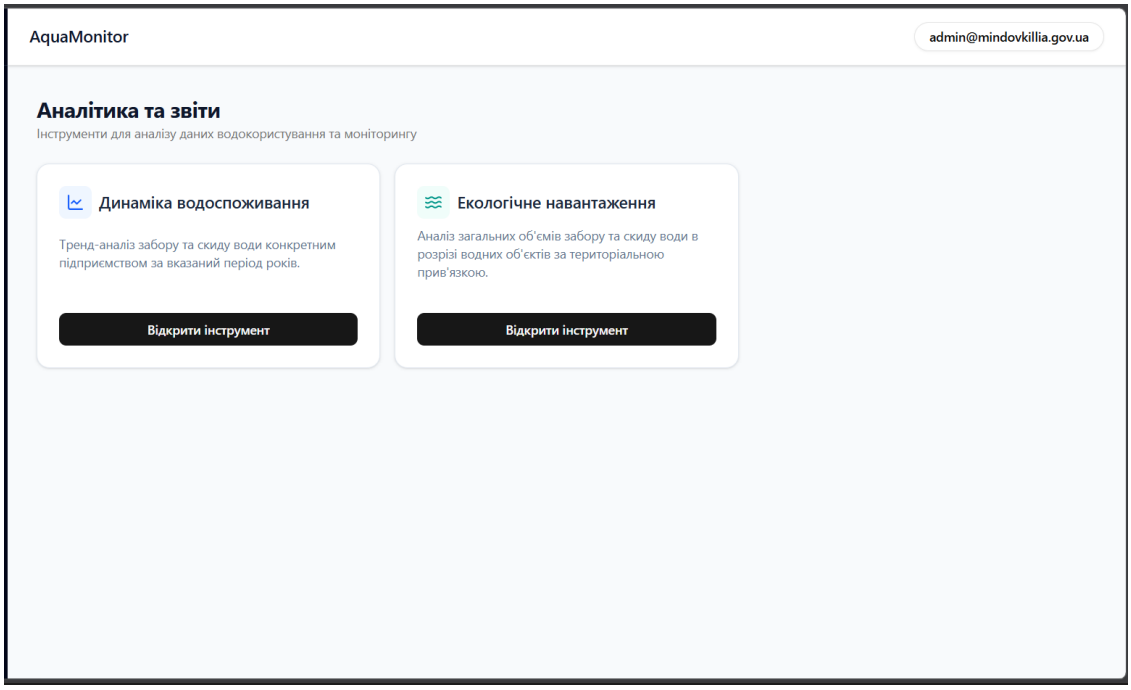


Рисунок 4.1 – Демонстрація сторінки аналітичної панелі

Архітектура інтерфейсу базується на використанні модульної системи компонування, яка спирається на адаптивну сітку (grid-cols-1 md:grid-cols-2) для коректного відображення елементів незалежно від роздільної здатності екрана. Основними структурними одиницями сторінки є інформаційні картки (на базі компонента Card), кожна з яких інкапсулює опис окремої аналітичної підсистеми. Зокрема, реалізовано блоки переходу до модуля аналізу багаторічних трендів водоспоживання за конкретними суб'єктами господарювання та до модуля просторової оцінки навантаження на водні об'єкти з прив'язкою до територіального класифікатора КАТОТТГ. Для семантичного та візуального розмежування розділів застосовано спеціалізовані векторні іконки (TrendingUp, Waves) з бібліотеки lucide-react.

Для проведення ретроспективного аналізу динаміки використання водних ресурсів спроектовано сторінку модуля WaterUsageTrendPage (Рисунок 4.2-4.4). Компонент забезпечує відображення багаторічних трендів водоспоживання та водовідведення для обраного суб'єкта господарювання.

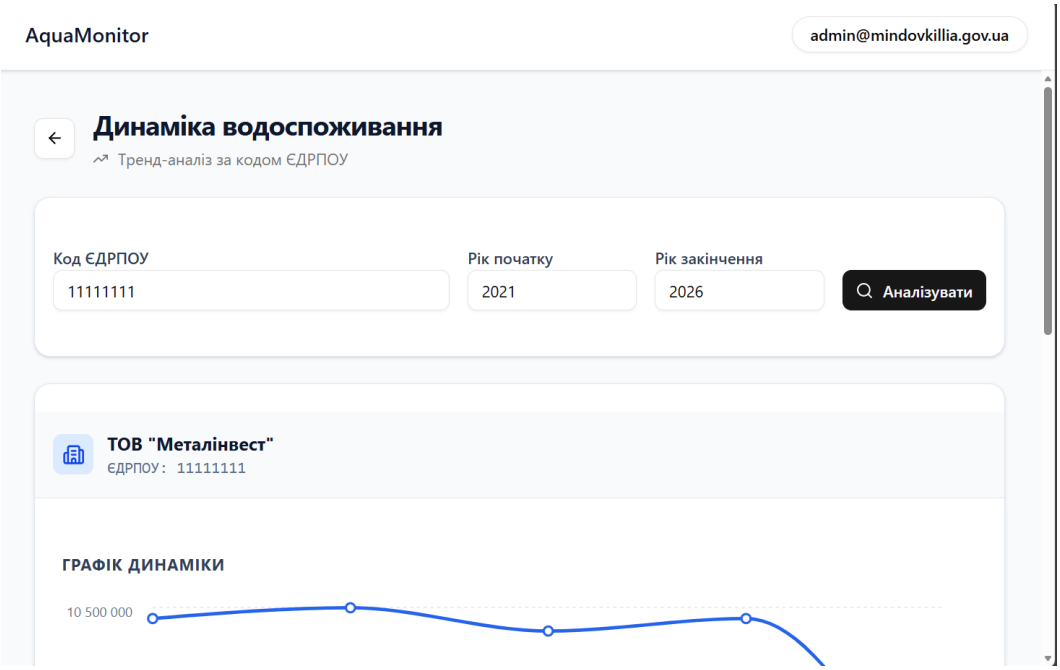


Рисунок 4.2 – Демонстрація звіту «Динаміка водоспоживання»

Рисунок 4.3 – Демонстрація звіту «Динаміка водоспоживання»

Рік	Забір води (м³)	Скид води (м³)	Безповоротне споживання (м³)
2021	10 060 000,00	3 285 000,00	6 775 000,00
2022	10 482 500,00	3 419 000,00	7 063 500,00
2023	9 570 000,00	3 115 000,00	6 455 000,00
2024	10 060 000,00	3 394 000,00	6 666 000,00
2025	735 000,00	875 000,00	-140 000,00

Рисунок 4.4 – Демонстрація звіту «Динаміка водоспоживання»

Інтерфейс модуля містить панель конфігурації часових меж (початковий та кінцевий роки), яка дозволяє користувачеві гнучко налаштовувати період дослідження. Асинхронне завантаження аналітичної вибірки реалізовано за допомогою хука useQuery, який ініціює запит до сервера виключно після валідації введених параметрів (наявність ідентифікатора, коректність років) та генерації відповідного об'єкта queryParams. Управління станом завантаження дозволяє блокувати інтерфейс і виводити інформаційні повідомлення під час очікування відповіді від API.

Для проведення комплексного аналізу антропогенного впливу на гідрологічні екосистеми з прив'язкою до територіального устрою розроблено сторінку модуля WaterBodyLoadPage (Рисунок 4.5-4.7). Компонент забезпечує візуалізацію та табличне представлення агрегованих даних щодо об'ємів водокористування в межах заданого класифікатора адміністративно-територіальних одиниць (КАТОТТГ). Інтерфейс логічно розділено на блок конфігурації параметрів фільтрації та робочу область відображення результатів аналітики.

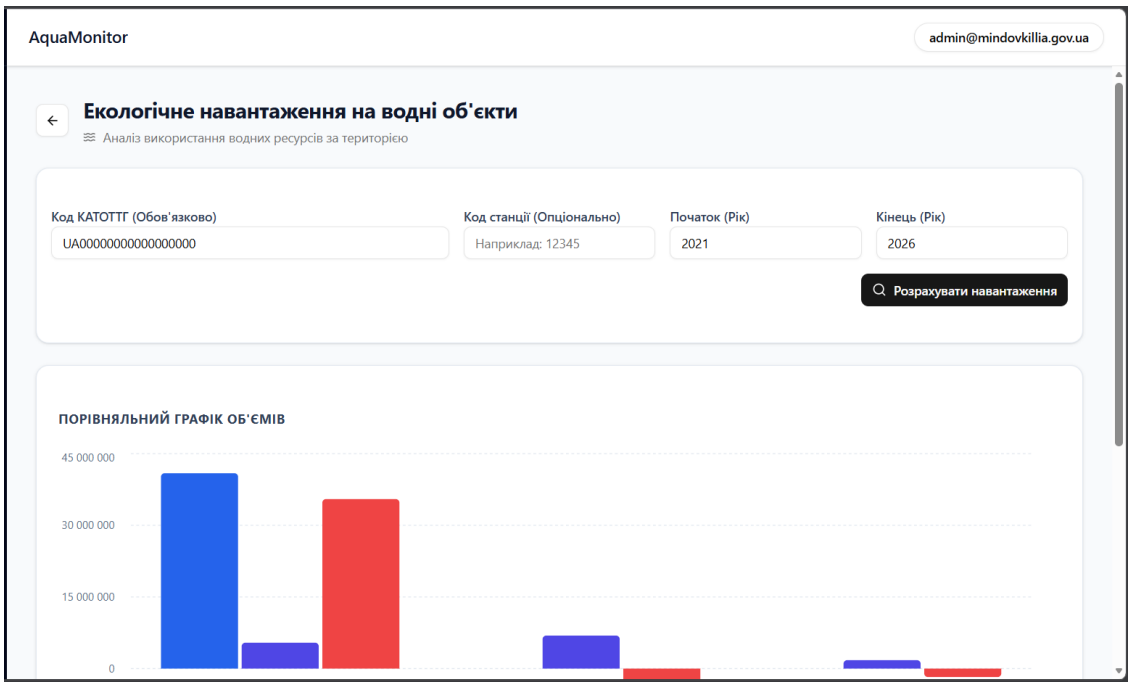


Рисунок 4.5 – Демонстрація звіту «Екологічне навантаження на водні об’єкти»

Рисунок 4.6 – Демонстрація звіту «Екологічне навантаження на водні об’єкти»

Рисунок 4.7 – Демонстрація звіту «Екологічне навантаження на водні об’єкти»

Панель фільтрації дозволяє користувачеві визначити часовий діапазон (початковий та кінцевий роки), базовий код КАТОТТГ та, за потреби, код конкретної станції моніторингу. Збір параметрів реалізовано за допомогою локального стану компонента, а ініціалізація відправки мережевого запиту до API виконується виключно після підтвердження введених даних через генерацію об'єкта queryParams. Асинхронне отримання аналітичної вибірки обробляється хуком useQuery, що гарантує кешування результатів та керування станом завантаження інтерфейсу під час виконання складних запитів на стороні сервера.

Для управління нормативно-дозвільною документацією підприємства реалізовано сторінку реєстру екологічних дозволів (Рисунок 1.8). Компонент забезпечує відображення переліку дозволів на спеціальне водокористування, які прив'язані до поточного активного суб'єкта моніторингу. Ідентифікація суб'єкта виконується автоматично через отримання ідентифікатора з глобального стану за допомогою useSubjectStore.

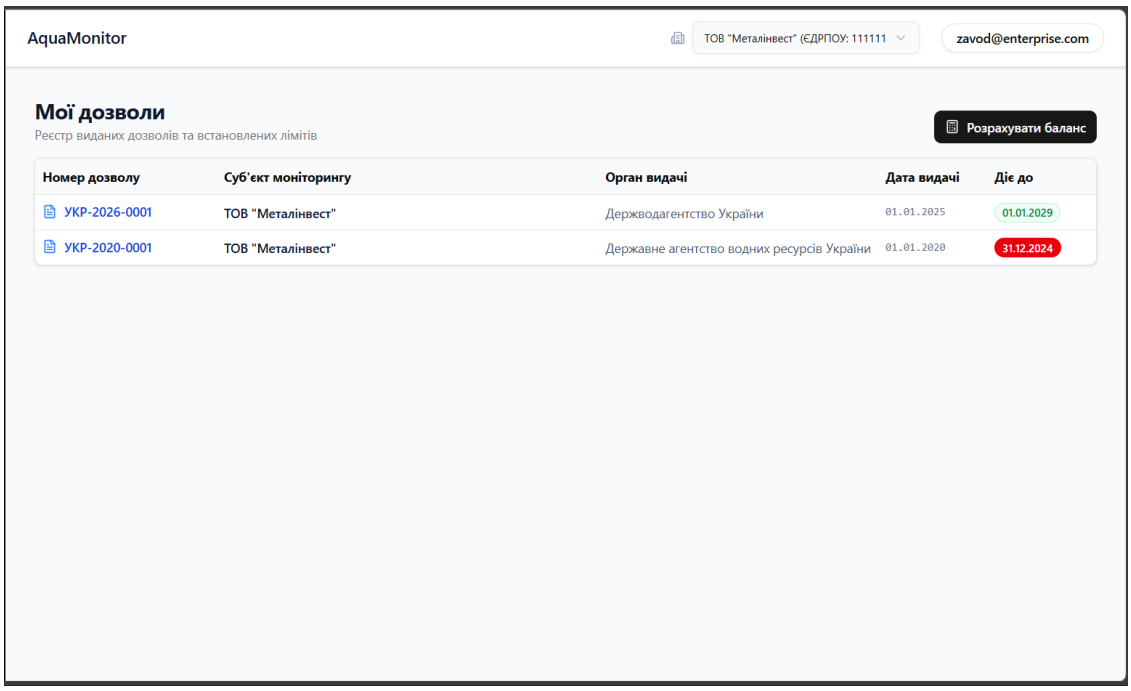


Рисунок 4.8 – Демонстрація сторінки переліку наявних дозволів суб’єкту

Інтерфейс побудовано на базі табличного представлення даних (компонент Table), де кожному запису відповідають ключові атрибути: номер дозволу, дати початку та закінчення дії, а також орган видачі. Для візуального контролю актуальності документів впроваджено систему динамічних статусів: алгоритм на стороні клієнта аналізує дату закінчення дії дозволу (expirationDate) і маркує його індикатором «Активний» або «Протермінований». Панель керування сторінки містить швидкі дії: кнопку ініціалізації додавання нового дозволу та окрему кнопку для переходу до модуля генерації балансового звіту (/permits/balance-report).

Для забезпечення можливості детального перегляду параметрів конкретного дозвільного документа спроектовано сторінку EcologicalPermitDetailPage (Рисунок 4.9). Ідентифікація необхідного дозволу здійснюється через отримання параметра id безпосередньо з навігаційного маршруту (URL) за допомогою хука useParams, після чого ініціюється асинхронний запит до API через useQuery.

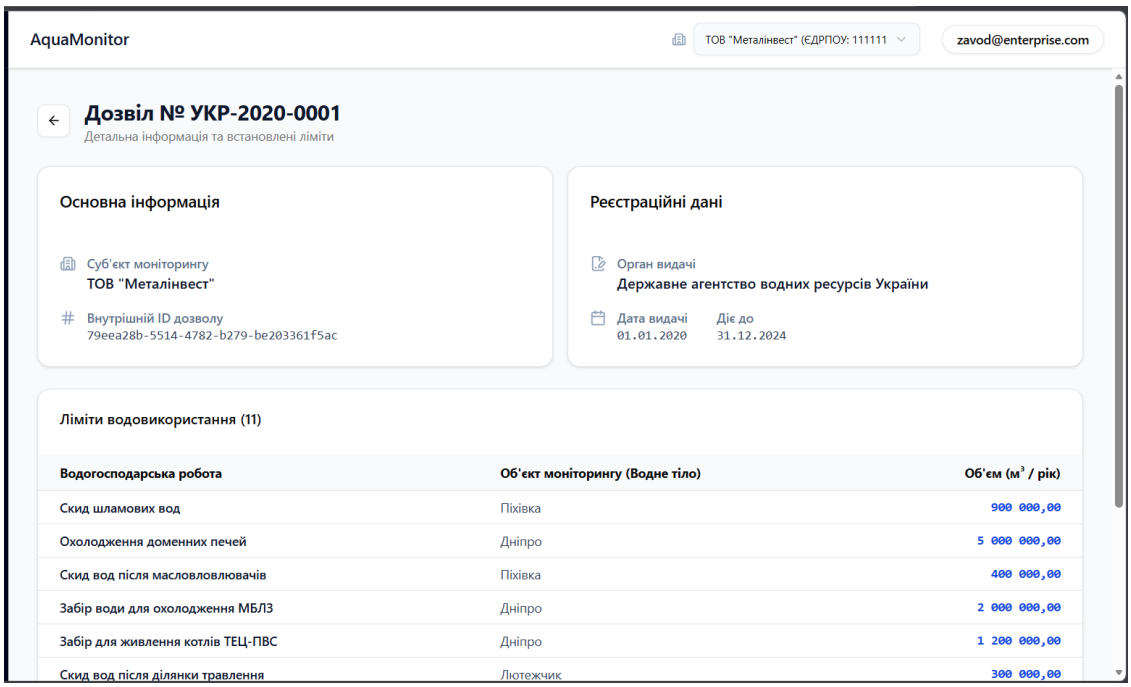


Рисунок 4.9 – Демонстрація сторінки деталізації дозволу

Верхня частина інтерфейсу містить інформаційну панель (на базі компонента Card), де консолідовано загальні метадані документа: реєстраційний номер, дату видачі, дату закінчення терміну дії та найменування органу, що видав дозвіл. Аналогічно до загального реєстру, на сторінці реалізовано алгоритм візуальної індикації статусу документа, який автоматично порівнює поточну дату з датою завершення дії дозволу (expirationDate) та відповідним чином маркує його як «Активний» або «Протермінований».

Нижній блок інтерфейсу призначений для відображення встановлених квот (лімітів) водокористування. Для оптимізації сприйняття великих масивів інформації застосовано компонент вкладок (Tabs), який логічно розділяє ліміти за напрямком гідрологічного потоку. Масив квот, отриманий із сервера, фільтрується на стороні клієнта на дві ізольовані колекції: обмеження забору води (Intake) та обмеження скиду стічних вод (Discharge). Кожна з категорій відображається у власній вкладці за допомогою табличного компонента, що містить назву водного об'єкта, нормативний код водогосподарської роботи та затверджений річний ліміт у кубічних метрах.

Для автоматизації процесу контролю за дотриманням встановлених лімітів спроектовано сторінку формування звіту балансу дозволу (Рисунок 4.10). Цей модуль виконує порівняльний аналіз фактичних об'ємів водоспоживання та водовідведення із затвердженими квотами згідно з обраним екологічним дозволом за вказаний звітний рік.

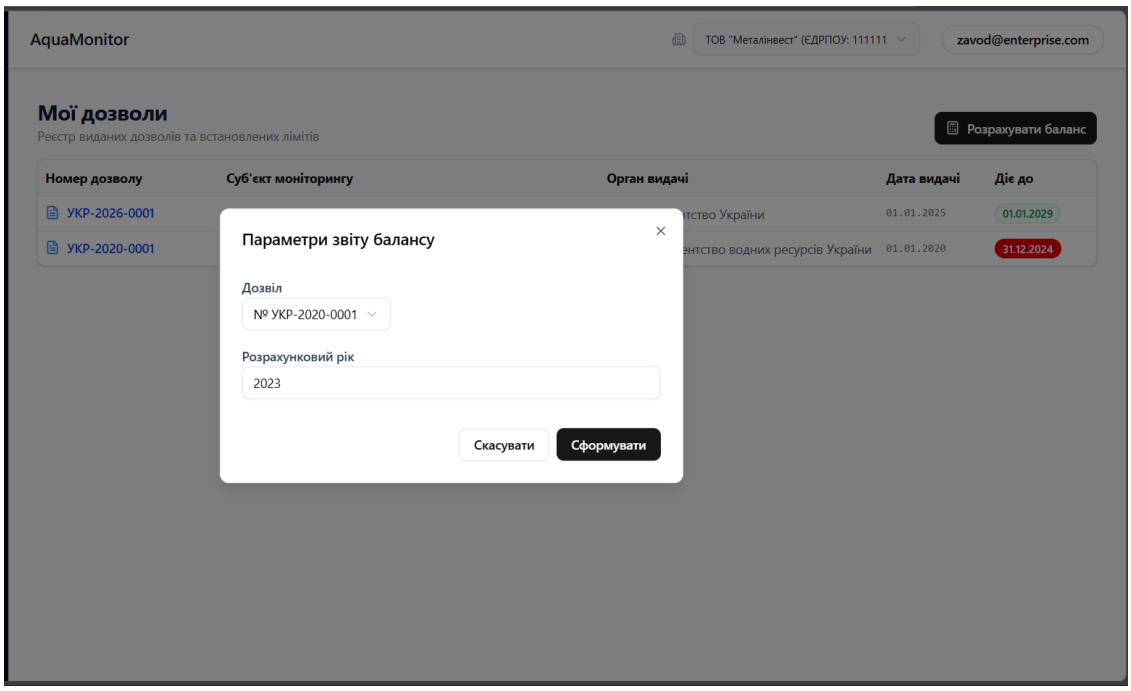


Рисунок 4.10 – Демонстрація визначення параметрів для розрахунку балансу водовикористання

Інтерфейс логічно поділено на панель параметрів формування запиту та область відображення розрахованих результатів. Панель параметрів містить випадаючий список для вибору цільового дозволу, дані для якого завантажуються асинхронно через useQuery, та поле введення цільового звітного року. Відправка параметрів до API та генерація звіту обробляється за допомогою хука useMutation, що дозволяє керувати станом завантаження.

Результати розрахунків сегментуються на два ізольовані блоки: «Забір води» та «Скид води». Для кожного блоку виводиться деталізована таблиця з прив'язкою до конкретного водного об'єкта, яка відображає встановлений ліміт, фактично використаний об'єм та розраховану різницю (дельту). Вбудована логіка рендерингу автоматично маркує результати відповідними значками (Badges): зелений статус «Норма», якщо фактичний об'єм не перевищує ліміт, та червоний статус «Перевищення», якщо зафіксовано порушення квоти (Рисунок 4.11-4.13).

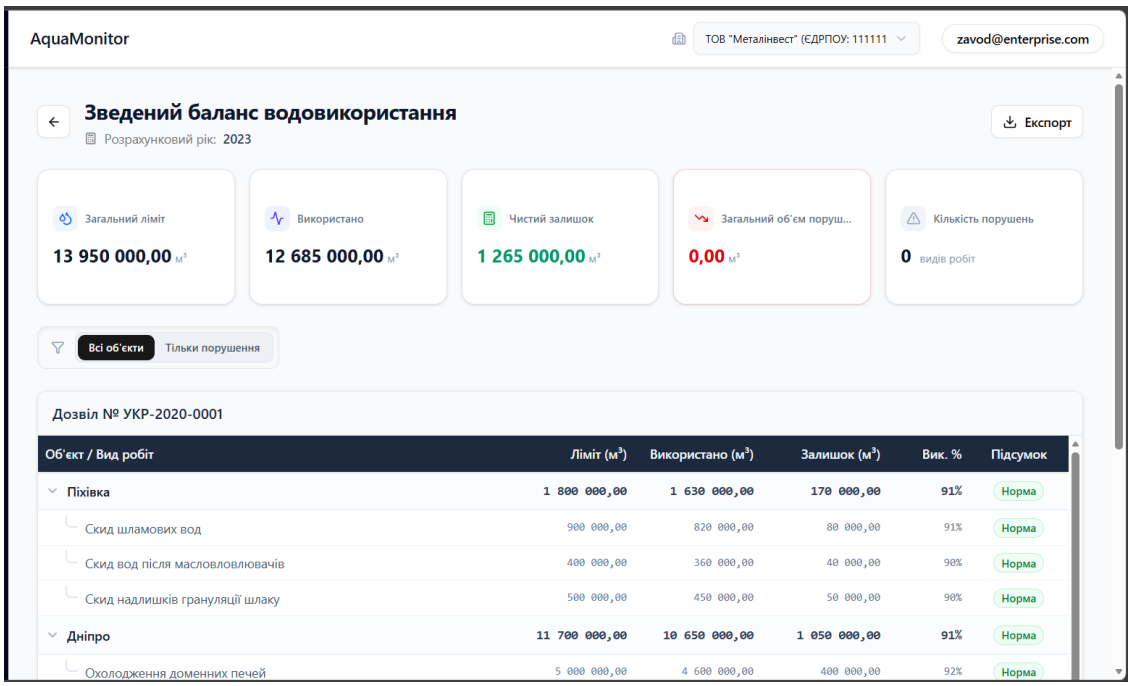


Рисунок 4.11 – Демонстрація звіту «Зведений баланс водовикористання»

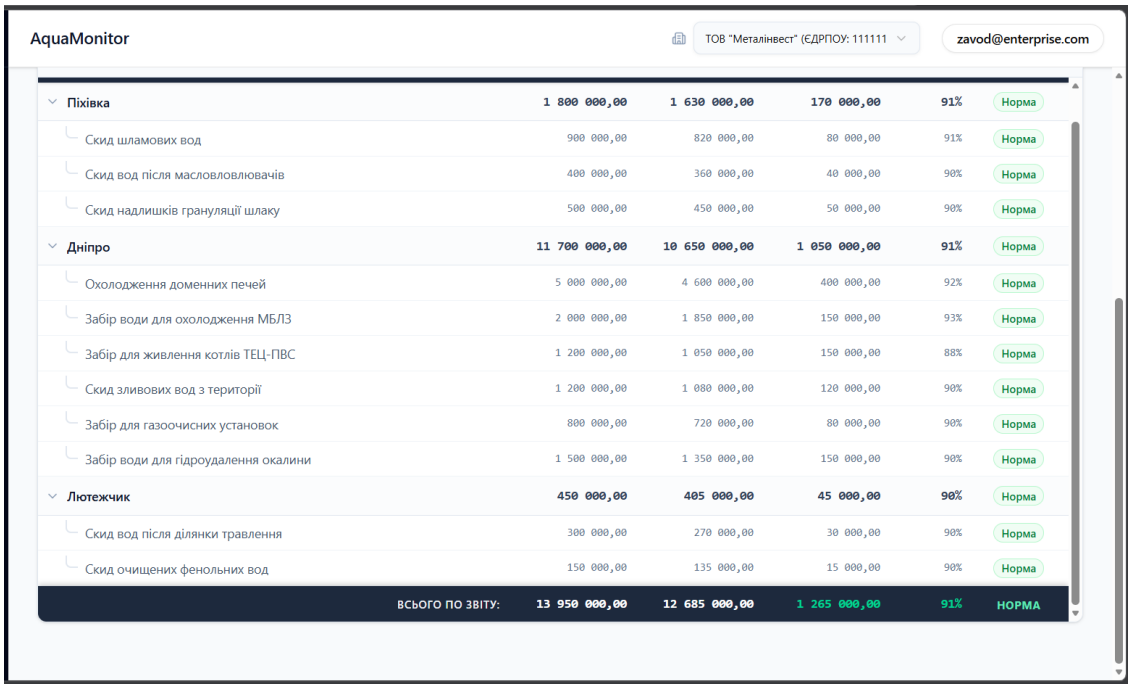


Рисунок 4.12 – Демонстрація звіту «Зведений баланс водовикористання»

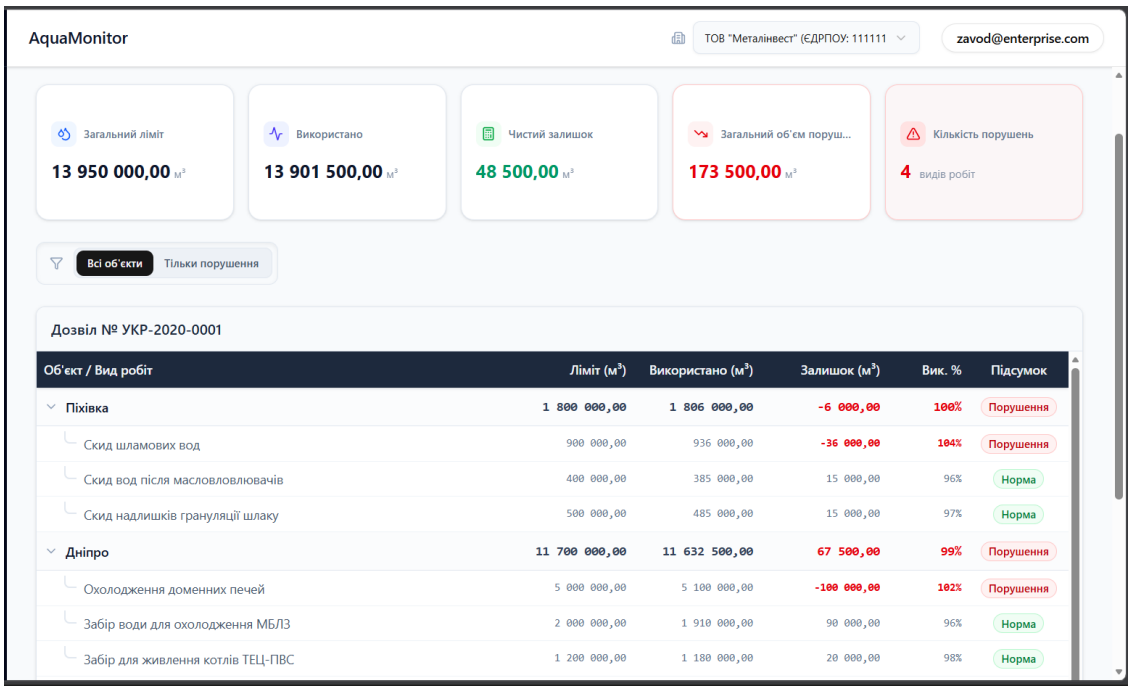


Рисунок 4.13 – Демонстрація звіту «Зведений баланс водовикористання»

Додатково, для забезпечення комплексної аналітики, результати візуалізуються за допомогою гістограми порівняння об'ємів, реалізованої на базі компонента BarChart з бібліотеки Recharts. Графік дозволяє наочно оцінити співвідношення фактичних показників та дозволених лімітів у розрізі кожного водного об'єкта, забезпечуючи інтерактивність через використання спливаючих підказок з україномовною локалізацією метрик (Рисунок 4.14).

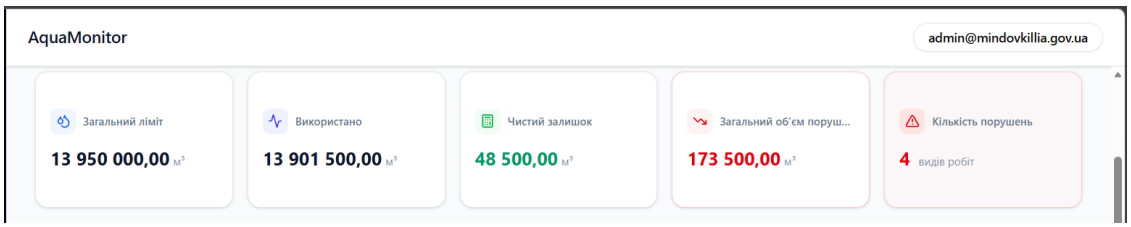


Рисунок 4.14 – Демонстрація звіту «Зведений баланс водовикористання»

Для забезпечення процесу обліку та управління поданою державною звітністю спроектовано сторінку реєстру звітів водокористування (Рисунок 4.15). Компонент відповідає за відображення історичних та поточних звітів обраного суб'єкта господарювання. Завантаження масиву даних ініціюється автоматично за допомогою хука useQuery на основі ідентифікатора підприємства, що вилучається з глобального стану сховища useSubjectStore.

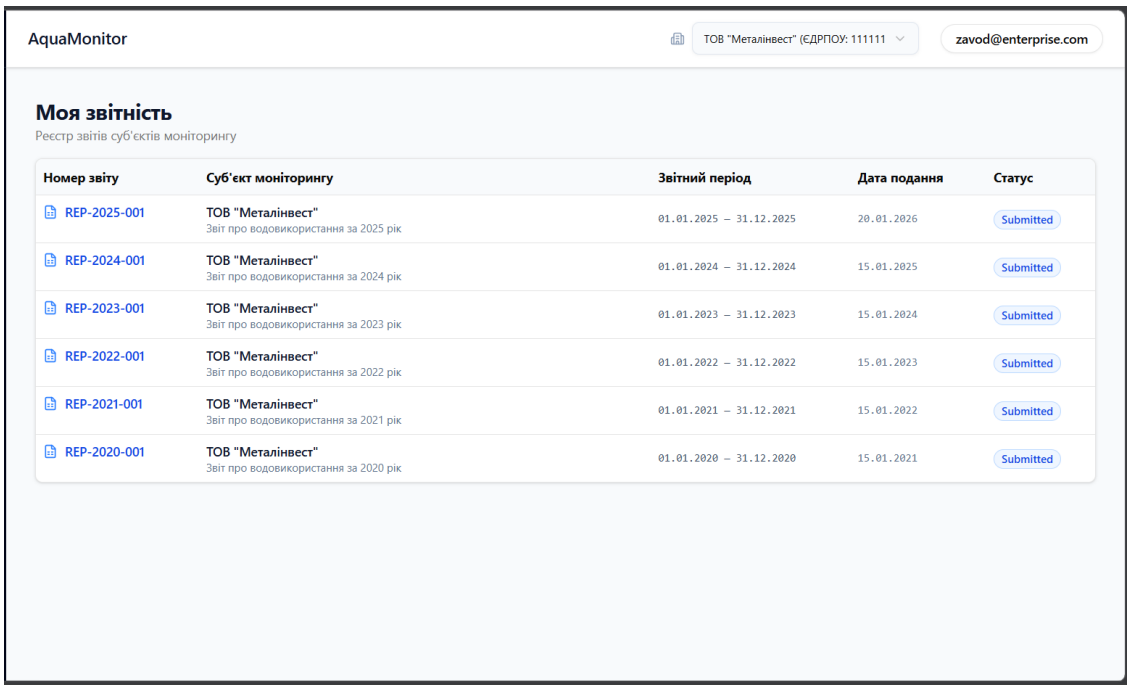


Рисунок 4.15 – Демонстрація переліку поданих звітів з водовикористання

Основна робоча область реалізована у вигляді таблиці, яка консолідує ключові агреговані метрики: звітний рік, сумарні об'єми забору та скиду води, а також поточний стан обробки документа системою. Для забезпечення швидкої візуальної ідентифікації етапів життєвого циклу звіту застосовано функцію динамічного мапінгу статусних кодів у відповідні кольорові індикатори (Badges) з україномовною локалізацією. Навігація до детальної інформації реалізується через обробник події кліку на рядок таблиці, який перенаправляє користувача за відповідним маршрутом з ідентифікатором звіту.

Детальний перегляд показників водоспоживання за конкретний звітний період реалізовано на сторінці WaterUsageReportDetailPage (Рисунок 4.16). Архітектура компонента передбачає отримання унікального ідентифікатора звіту з параметрів маршрутизації (URL) та подальше асинхронне завантаження повної структури документа через API.

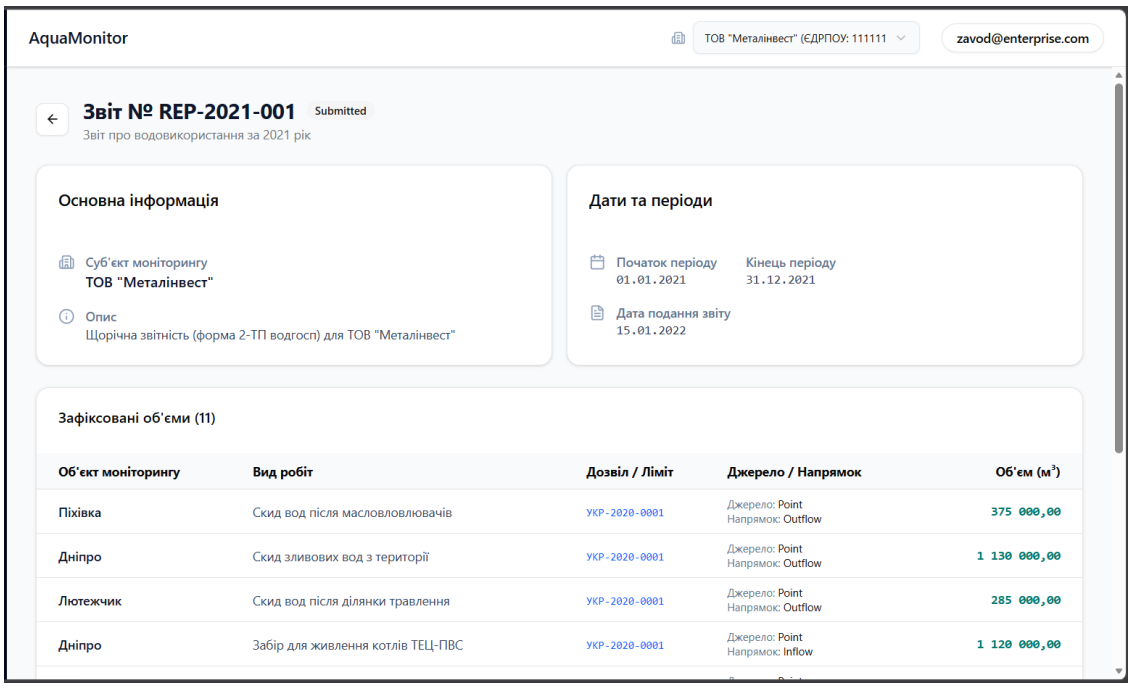


Рисунок 4.16 – Демонстрація сторінки деталізації звіту з водовикористання

Для забезпечення систематизації та відстежуваності результатів лабораторного контролю зворотних та природних вод реалізовано сторінку реєстру протоколів якості води (Рисунок 4.17). Даний компонент виступає центральним інформаційним хабом для екологів та представників державних інспекцій, дозволяючи переглядати перелік зареєстрованих документів аналізу.

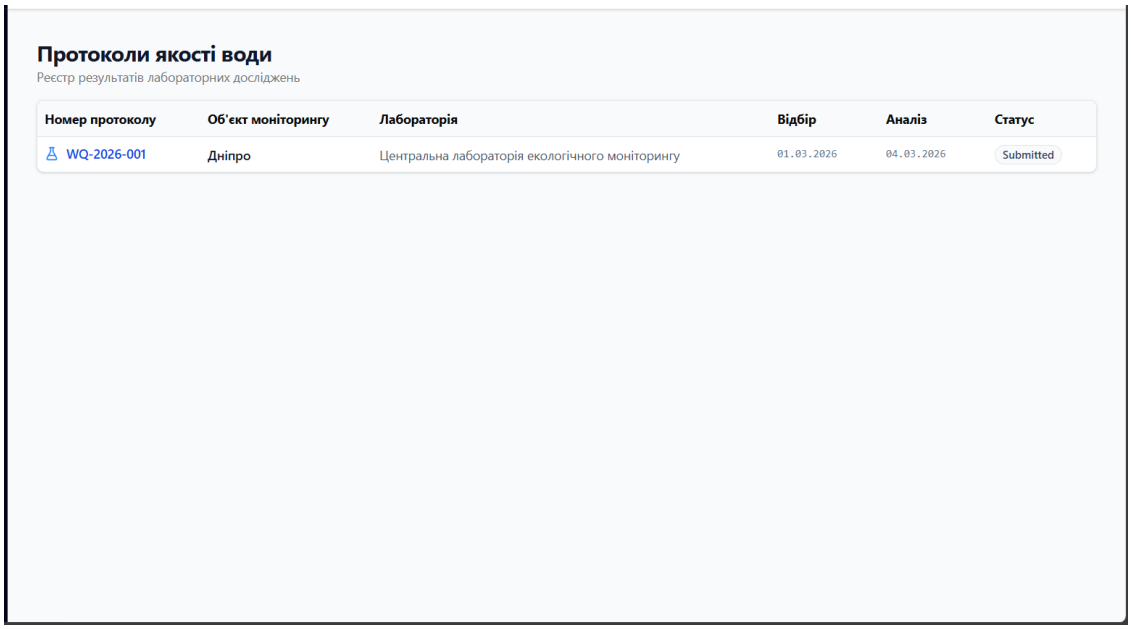


Рисунок 4.17 – Демонстрація переліку поданих протоколів з якості води

Для оперативного візуального моніторингу стану документів впроваджено функцію динамічного кодування статусів («В роботі», «Затверджено», «Відхилено»). Обробник кліку на рядок таблиці забезпечує імперативну маршрутизацію користувача на сторінку повної деталізації конкретного протоколу.

Для поглибленого аналізу хімічного складу проб у розрізі окремого документа спроектовано сторінку детальної інформації протоколу якості води (Рисунок 4.18-4.19). Ініціалізація компонента супроводжується вилученням унікального ідентифікатора протоколу з параметрів URL-маршруту за допомогою хука useParams та виконанням асинхронного запиту на отримання повної доменної моделі об'єкта.

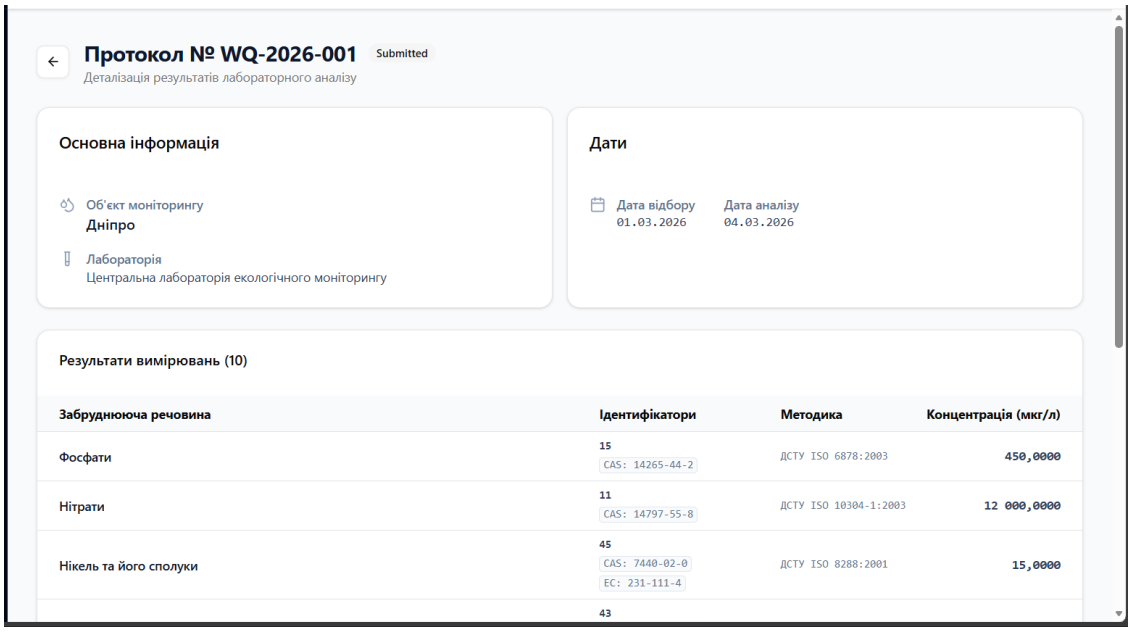


Рисунок 4.18 – Демонстрація деталізації протоколу з якості води

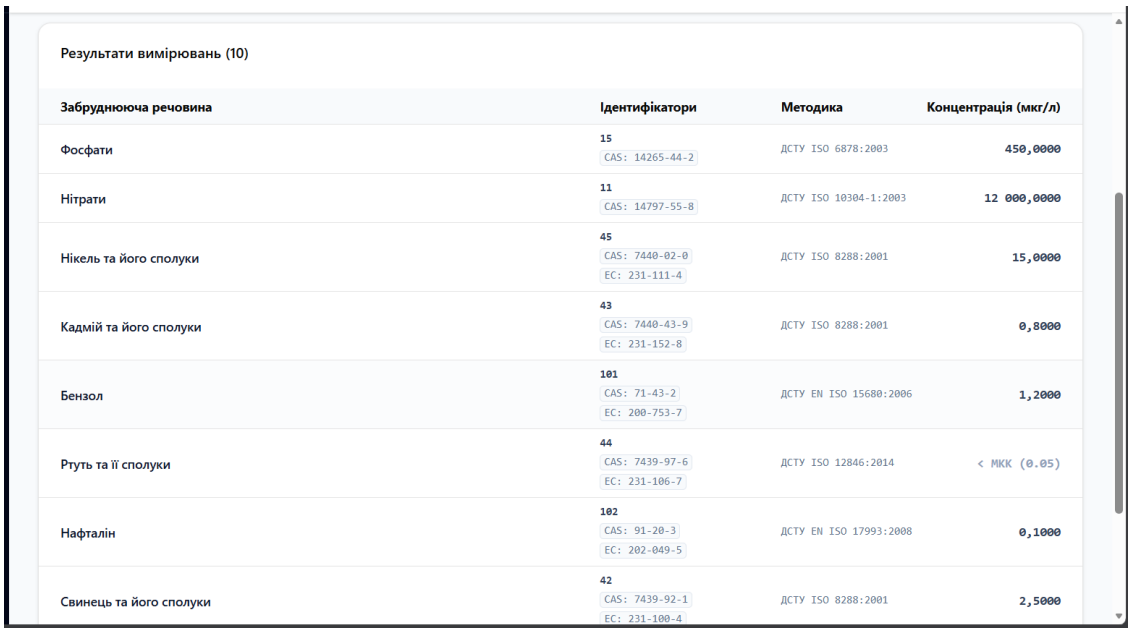


Рисунок 4.19 – Демонстрація деталізації протоколу з якості води

Для структурування значних обсягів інформації застосовано механізм вкладок, який логічно ізолює показники забору води від показників скиду стічних вод. У кожній вкладці формується таблиця з деталізацією по конкретних водних об'єктах, де відображаються затверджені ліміти, фактичні об'єми використання та розрахована математична різниця (дельта). Від'ємні значення дельти автоматично підсвічуються червоним кольором для сигналізації про перевищення ліміту. Додатково, на рівні клієнтської логіки реалізовано алгоритм агрегації загальних об'ємів, який виконує підсумовування показників з масиву записів для виведення у верхній інформаційній панелі компонента.

Для забезпечення централізованого доступу до реєстрів та класифікаторів предметної області спроектовано сторінку панелі управління довідниками (Рисунок 4.20-4.27). Даний компонент виконує роль єдиного навігаційного центру (дашборду) для адміністрування базових сутностей системи, необхідних для коректного функціонування транзакційних та аналітичних модулів.

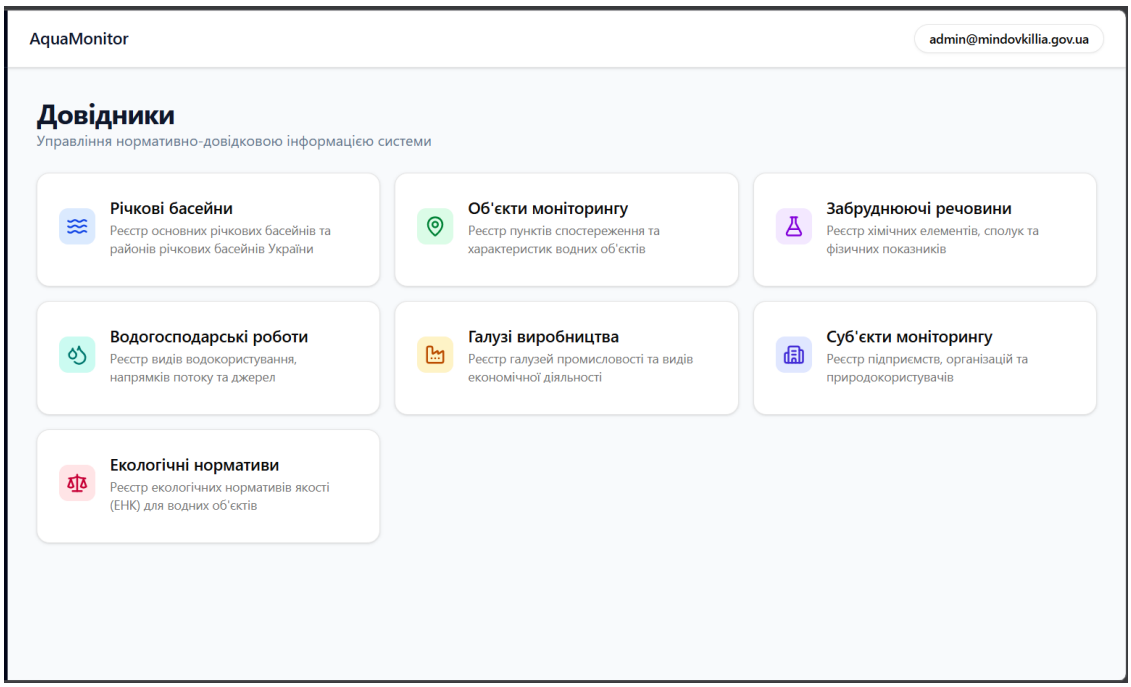


Рисунок 4.20 – Демонстрація сторінки довідникової інформації

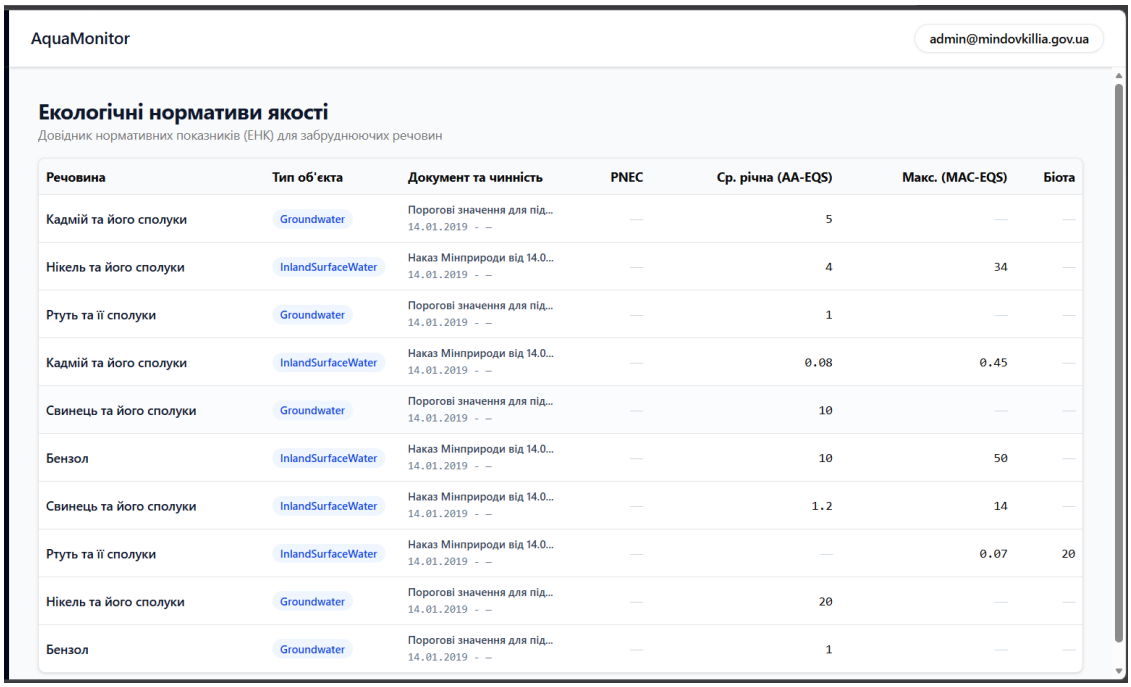


Рисунок 4.21 – Демонстрація довідника екологічних нормативів якості

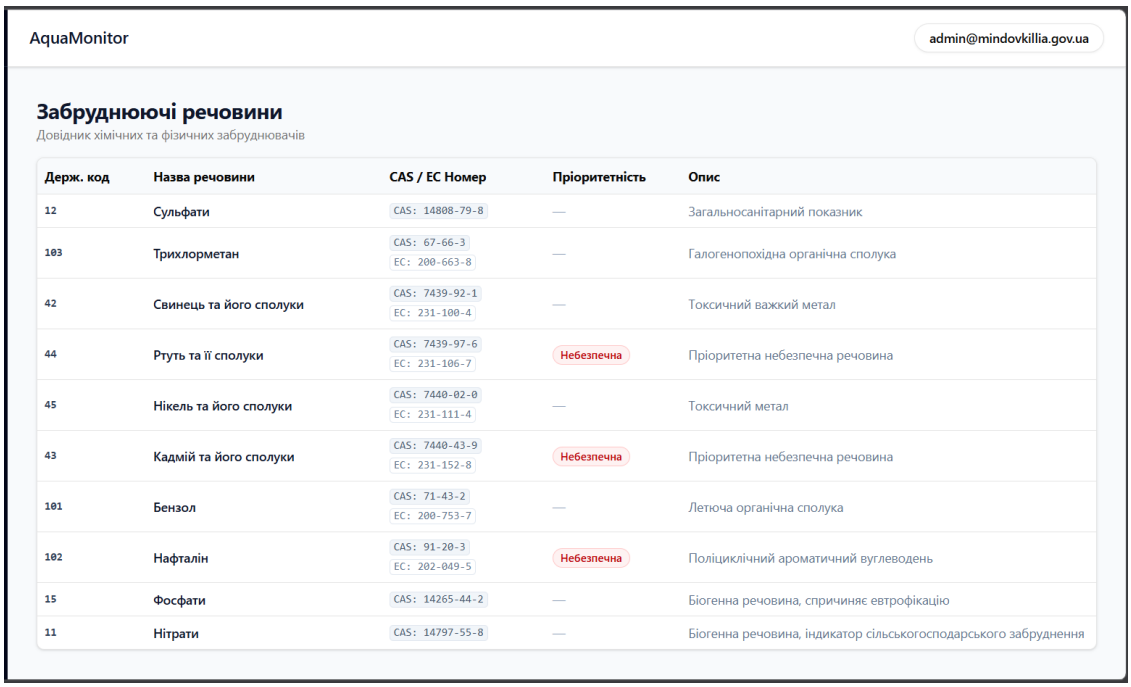


Рисунок 4.22 – Демонстрація довіника забруднюючих речовин

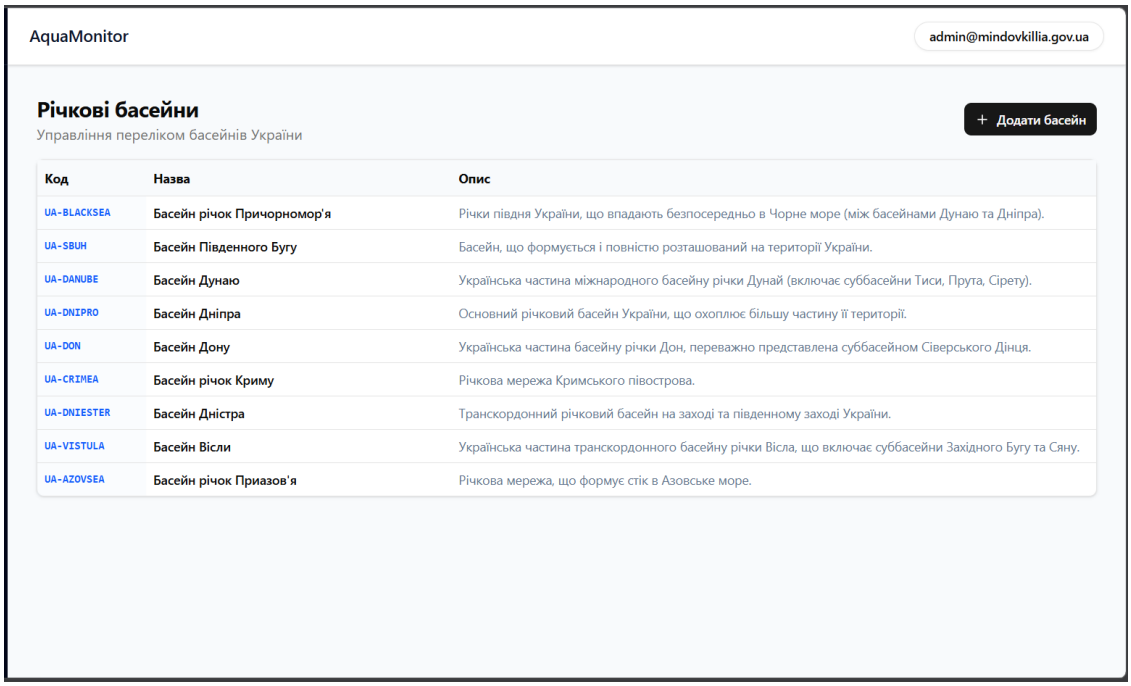


Рисунок 4.23 – Демонстрація довідника річкових басейнів

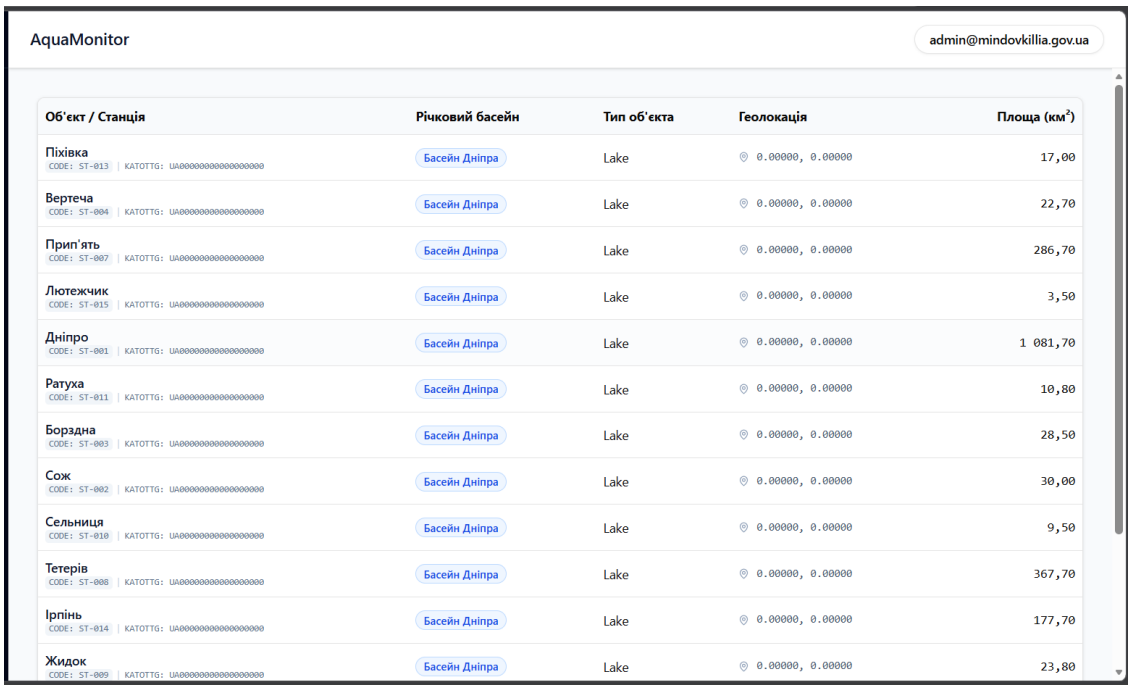


Рисунок 4.24 – Демонстрація довідника об’єктів моніторингу

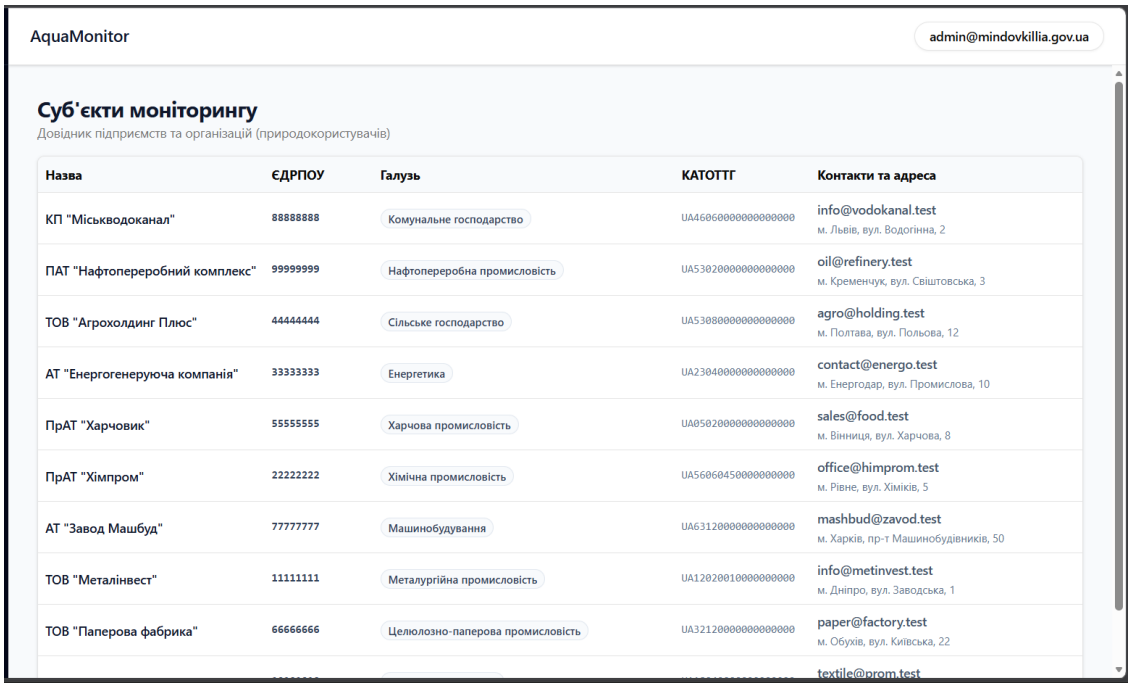


Рисунок 4.25 – Демонстрація довідника суб’єктів моніторингу

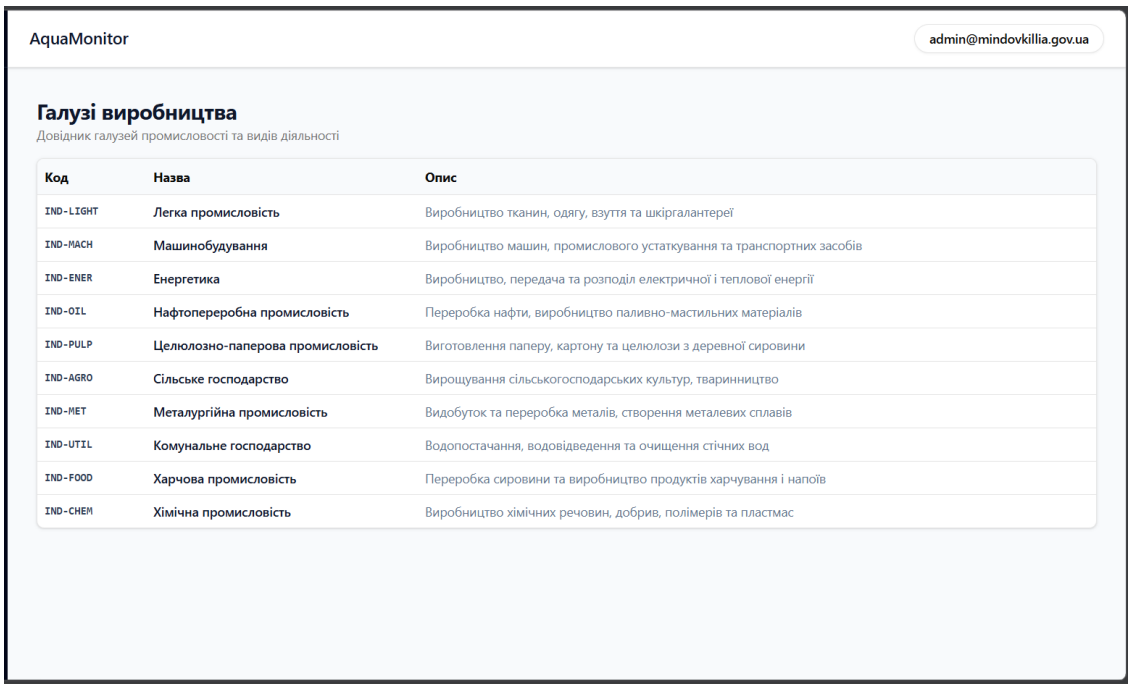


Рисунок 4.26 – Демонстрація довідника галузей виробництва

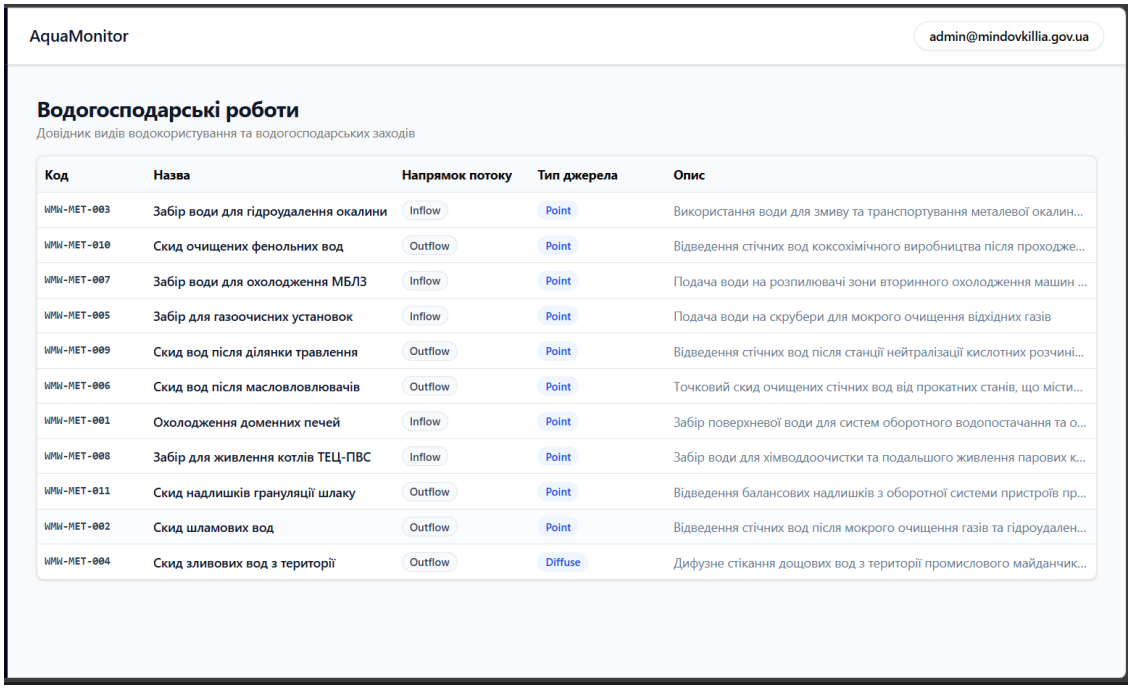


Рисунок 4.27 – Демонстрація довідника водогосподарських робіт

Інтерфейс побудовано за принципом модульного дашборду, що складається з інтерактивних інформаційних карток (на базі перевикористовуваного компонента Card), організованих за допомогою адаптивної сітки (grid-cols-1 md:grid-cols-2 lg:grid-cols-3). Кожен елемент панелі логічно згрупований та інкапсулює доступ до відповідного підмодуля управління: екологічними нормативами якості (EQS) (Рисунок 4.21), реєстрами

забруднюючих речовин за CAS-номерами (Рисунок 4.22), класифікаторами водних об'єктів та річкових басейнів (Рисунок 4.23-4.24), а також довідниками суб'єктів моніторингу (Рисунок 4.25). Для покращення візуальної ідентифікації розділів застосовано семантичні іконки з бібліотеки lucide-react.

Програмна маршрутизація між розділами реалізована за допомогою хука useNavigate з бібліотеки react-router-dom, який імперативно перенаправляє користувача на відповідні шляхи при виникненні події натискання на елемент управління.

4.4 Організація зберігання даних

Центральним інфраструктурним вузлом, що забезпечує виконання операцій читання та запису даних, є клас AquaMonitorDbContext, який розміщено в просторі імен AquaMonitor.Infrastructure.Persistence. Цей клас успадковується від базового класу Microsoft.EntityFrameworkCore.DbContext та імплементує прикладний інтерфейс абстракції IApplicationDbContext. В архітектурі системи він виконує фундаментальну роль координатора транзакцій, реалізуючи класичний патерн об'єктно-орієнтованого проектування Unit of Work.

```
public class AquaMonitorDbContext : DbContext,
     IApplicationDbContext
{
    public
     AquaMonitorDbContext(DbContextOptions<AquaMonitorDbContext>
     options)
         : base(options) { }

    // --- 1. АДМІНІСТРАТИВНА ЧАСТИНА ---
    public DbSet<User> Users => Set<User>();
    public DbSet<SubjectRepresentative> SubjectRepresentatives
     => Set<SubjectRepresentative>();

    // --- 2. ДОВІДНИКИ ---
    public DbSet<MonitoringSubject> MonitoringSubjects =>
     Set<MonitoringSubject>();
    public DbSet<WaterManagementWork> WaterManagementWorks =>
     Set<WaterManagementWork>();
    public DbSet<Pollutant> Pollutants => Set<Pollutant>();
    public DbSet<RiverBasin> RiverBasins => Set<RiverBasin>();
    public DbSet<MonitoringObject> MonitoringObjects =>
     Set<MonitoringObject>();

    public DbSet<IndustryBranch> IndustryBranches =>
     Set<IndustryBranch>();
```

```
    public DbSet<EnvironmentalQualityStandard>
     EnvironmentalQualityStandards =>
     Set<EnvironmentalQualityStandard>();

    // --- 3. ДОКУМЕНТИ (Агрегати) ---
    public DbSet<EcologicalPermit> EcologicalPermits =>
     Set<EcologicalPermit>();
    public DbSet<PermitLimit> PermitLimits =>
     Set<PermitLimit>();
    public DbSet<WaterUsageReport> WaterUsageReports =>
     Set<WaterUsageReport>();
    public DbSet<WaterQualityProtocol> WaterQualityProtocols =>
     Set<WaterQualityProtocol>();

    // --- 4. АНАЛІТИКА ---
    public DbSet<HydrologicalBalanceAct> HydrologicalBalanceActs
     => Set<HydrologicalBalanceAct>();

    protected override void OnModelCreating(ModelBuilder
     modelBuilder)
    {
        base.OnModelCreating(modelBuilder);


     modelBuilder.ApplyConfigurationsFromAssembly(typeof(AquaMoni
     torDbContext).Assembly);
    }
}
```

Для проектування фізичної структури бази даних застосовано підхід конфігурації через Fluent API, який реалізується шляхом створення окремих класів, що імплементують інтерфейс IEntityTypeConfiguration<T>. WaterUsageReportConfigurations демонструє механізми трансляції високорівневої моделі агрегата WaterUsageReport у низькорівневу реляційну схему MS SQL Server:

```
public class WaterUsageReportConfigurations
: IEntityTypeConfiguration<WaterUsageReport>
{
    public void Configure(EntityTypeBuilder<WaterUsageReport>
     builder)
    {
        ConfigureBase(builder);
        ConfigureProperties(builder);
        ConfigureRelationships(builder);
    }
```

Внутрішня структура класу WaterUsageReportConfigurations декомпозована на три спеціалізовані приватні методи: ConfigureBase, ConfigureProperties та

ConfigureRelationships, виклик яких агреговано в основному методі Configure(EntityTypeBuilder<WaterUsageReport> builder). Така декомпозиція значно покращує читабельність та зручність супроводу конфігураційного коду.

ВИСНОВОК

У кваліфікаційній роботі вирішено актуальне завдання щодо розробки інформаційно-аналітичної системи для автоматизації процесів обліку, моніторингу та аналізу використання водних ресурсів.

У процесі виконання роботи виконано системний аналіз предметної області, досліджено національні стандарти та вимоги європейського законодавства. Аналіз існуючих процесів виявив децентралізацію, залежність від застарілих форматів збереження даних та високий ризик людської помилки, що стало основою для формування вимог до функціоналу цільової системи.

Спроєктовано клієнт-серверну архітектуру програмного рішення. Застосування архітектурного патерну CQRS у поєднанні з диспетчером MediatR забезпечило розділення відповідальності між операціями читання та запису, ізоляцію бізнес-логіки та масштабованість серверної частини. Розроблено структуру реляційної бази даних для ефективного збереження нормативно-довідкової інформації (реєстри річкових басейнів, забруднюючих речовин), дозвільних документів, протоколів лабораторного контролю та звітів водокористувачів.

Програмно реалізовано серверну підсистему (API) на платформі ASP.NET Core із налаштуванням механізмів автентифікації, авторизації та глобального контролю контексту суб'єкта моніторингу. Впроваджено алгоритми автоматичного розрахунку гідрологічних балансів та виявлення відхилень від нормативних лімітів. Клієнтський односторінковий застосунок (SPA) розроблено з використанням бібліотеки React і TypeScript. Реалізовано інтерфейси для управління реєстрами, перегляду деталізованих протоколів якості води та візуалізації аналітичних даних, включаючи інструменти для оцінки багаторічних трендів водоспоживання підприємств та просторового аналізу екологічного навантаження.

Впровадження розробленого програмного рішення усуває ручну консолідацію даних, мінімізує вплив людського фактора під час розрахунків та значно прискорює ідентифікацію перевищень нормативів гранично допустимих скидів. Оптимізовано процеси обміну екологічною інформацією між суб'єктами господарювання та контролюючими органами. Поставленої мети дослідження досягнуто повністю, усі визначені завдання виконано.

ПЕРЕЛІК ПОСИЛАНЬ

1. Звіт про використання води за формою 2-ТП (водгосп) [Електронний ресурс] // БУВР річок Приазов'я. URL: https://buvrpb.davr.gov.ua/vodni-resursy/zvit-pro-vykorystannia-vody-za-formoiu-2-tp-vodhosp.

2. У яких звітних періодах подаються дозвільні документи разом із декларацією з рентної плати за спеціальне водокористування [Електронний ресурс] // Блог бухгалтера від M.E.Doc. URL: https://medoc.ua/blog/u-jakih-zvitnih-periodah-podajutsja-dozvilni-dokumenti-razom-iz-deklaraciju-z-rentno-plati-za-specialne-vodokoristuvannja-.

3. З 1 січня до 1 лютого 2026 року триває прийом звітів про водокористування: правила та спосіб подачі [Електронний ресурс] // Державне агентство водних ресурсів України. URL: https://www.davr.gov.ua/news/z-1-sichnya-do-1-lyutogo-2026-roku-trivaye-prijom-zvitiv-pro-vodokoristuvannya-pravila-ta-sposib-podachi.

4. Звітність про використання води за формами № 2ТП-водгосп (річна) і № 7-ГР (підземні води) (річна) [Електронний ресурс] // Онлайн-консультант еколога підприємства. URL: https://ecologiya.com.ua/articles/286270-zvitnist-pro-vykorystannya-vody-za-formamy-no-2tp-vodhosp-richna-i-no-7-hr-pidzemni.

5. Наказ Міндовкілля № 332 від 01.04.2024 р. «Про затвердження екологічних нормативів якості води для визначення екологічного стану масиву поверхневих вод...» [Електронний ресурс] // Леонорм. URL: http://www.leonorm.lviv.ua/P/NL_DOC/2024/NakMDPR332.htm.

6. Звіт про використання води за формою №2ТП-водгосп (річна): інструкція [Електронний ресурс] // 7eminar. URL: https://7eminar.ua/news/3404-zvit-pro-vikoristannya-vodi-forma-2tp-vodgosp-ricna-instrukciya.

7. Моніторинг поверхневих вод [Електронний ресурс] // Державне агентство водних ресурсів України. URL: https://www.davr.gov.ua/monitoring-poverhnevih-vod1.

8. Наказ Мінприроди від 06.02.2017 № 45 «Про затвердження Переліку забруднюючих речовин для визначення хімічного стану масивів поверхневих і підземних вод...» [Електронний ресурс] // FAOLEX. URL: https://faolex.fao.org/docs/pdf/ukr170168.pdf.

9. Про затвердження "Методики розрахунку розмірів відшкодування збитків, заподіяних державі внаслідок порушення законодавства про охорону та раціональне використання водних ресурсів" [Електронний ресурс] // Інформаційно-аналітична система «Парус». URL: http://consultant.parus.ua/?doc=01G590A6DE.

10. The EU Water Framework Directive [Електронний ресурс] // Federal Ministry of Agriculture and Forestry, Climate and Environmental Protection, Regions and Water

Management. URL: https://www.bmluk.gv.at/en/topics/water/water-management/the-eu-water-framework-directive.html.

11. Directive 2013/39/EU of the European Parliament and of the Council of 12 August 2013 amending Directives 2000/60/EC and 2008/105 [Електронний ресурс] // EUR-Lex. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32013L0039&from=EN.

12. River pollution by priority chemical substances under the Water Framework Directive: A provisional pan-European assessment [Електронний ресурс] // PMC. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC6391594/.

13. Nitrates [Електронний ресурс] // Environment - European Commission. URL: https://environment.ec.europa.eu/topics/water/nitrates_en.

14. Surface water [Електронний ресурс] // Environment - European Commission. URL: https://environment.ec.europa.eu/topics/water/surface-water_en.

15. WISE - SoE WQ Data Reporting Tool and Help Manual v.1.1 [Електронний ресурс] // ResearchGate. URL: https://www.researchgate.net/publication/297731174_WISE_-_SoE_WQ_Data_Reporting_Tool_and_Help_Manual_v11.

16. Environment, Health and Safety (EHS) [Електронний ресурс] // SAP Help Portal. URL: https://help.sap.com/docs/EHS.

17. Water Information System for Europe (WISE) [Електронний ресурс] // Glossaire eau, milieu marin et biodiversité. URL: https://glossaire.eauetbiodiversite.fr/en/concept/water-information-system-europe-wise.

18. .NET 8+ [Електронний ресурс] // .NET documentation. URL: https://learn.microsoft.com/en-us/dotnet/.

19. CQRS pattern [Електронний ресурс] // Command Query Responsibility Segregation. URL: https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs.

20. MediatR [Електронний ресурс] // URL: https://mediatr.io/.

21. React [Електронний ресурс] // URL: https://react.dev/.

22. Palermo S. A., Maiolo M., Brusco A. C., Turco M., Pirouz B. Smart Technologies for Water Resource Management: An Overview // Sensors. 2022. T. 22. C. 6225. DOI: 10.3390/s22166225.

23. Olagunju T. E. GIS-based integrated water quality assessment and pollution hotspot mapping of the Oyun River–Reservoir system, Nigeria // African Journal of Aquatic Science. 2026. DOI: 10.2989/16085914.2026.2630907.

24. Choudhury M., Jyethi D. S., Dutta J. [та ін.]. Spatial analysis reveals groundwater potential // Frontiers in Water. 2026. DOI: 10.3389/frwa.2026.1801919.

25. Medeiros R. B., Berezuk A. G., Pinto A. L. [та ін.]. Calidad de Las Aguas Superficiales En Sistemas Kársticos. Un Estudio de La Cuenca Hidrográfica Del Río Formoso... // Investigaciones Geográficas. 2022. Т. 78. С. 107–129.

26. Buitink J., Tsiokanos A., Geertsema T. [та ін.]. Water balance simulation models // Geoscientific Model Development. 2026. Т. 19. С. 2551–2575. DOI: 10.5194/gmd-19-2551-2026.

27. Shekhar S., Jha M. Groundwater level prediction of Varanasi wells during pre-monsoon and post-monsoon using intelligence approach // Arabian Journal of Geosciences. 2022. Т. 15. С. 88. DOI: 10.1007/s12517-021-08875-6.

ДОДАТОК А – КЛАСИФІКАЦІЯ РЕЧОВИН ЗГІДНО З НАКАЗОМ № 45

Назва хімічної речовини або параметра	Міжнародний код CAS	Нормативна група за Наказом № 45	Системні обмеження та коментарі для ІС
Алахлор	15972-60-8	Пріоритетні речовини (хімічний стан поверхневих вод)	Нормується в мкг/л
Антрацен	120-12-7	Пріоритетні речовини (хімічний стан поверхневих вод)	Нормується в мкг/л
Кадмій та його сполуки	7440-43-9	Важкі метали (хімічний стан поверхневих вод)	Залежить від класу жорсткості води
Ртуть та її сполуки	7439-97-6	Важкі метали (хімічний стан поверхневих вод)	Високотоксичний елемент
Бензо(а)пірен	50-32-8	Поліциклічні ароматичні вуглеводні (ПАВ)	Маркується як "Не застосовується" безпосередньо у загальному переліку хімічного стану без супутніх ПАВ
Поліхлоровані дибензо-п-діоксини (ПХДД)	Не застосовується	Токсичні ізомери	Оцінка вимагає розрахунку за факторами токсичного еквівалента ВООЗ 2005 (ТЕ)
Нітрати	Не застосовується	Хімічний стан підземних вод	Нормується для підземних горизонтів
Біохімічне споживання кисню (БСК_5)	Не застосовується	Екологічний потенціал штучних/змінених масивів	Фізико-хімічний інтегральний показник

ДОДАТОК Б – КОД СЕРВЕРНОЇ ЧАСТИНИ

AquaMonitor.API/Program.cs

```
var builder = WebApplication.CreateBuilder(args);
{
    builder.Services
        .AddPresentation()
        .AddApplication()
        .AddInfrastructure(builder.Configuration);

    builder.Services.AddControllers();
}

var app = builder.Build();
{
    app.UseExceptionHandler("/error");
    app.Map("/error", (HttpContext context) =>
    {
        Exception? exception =
     context.Features.Get<IExceptionHandlerFeature>()?.Error;
        return Results.Problem();
    } );

    app.UseHttpsRedirection();

    app.UseCors("AllowViteFrontend");

    app.UseAuthentication();
    app.UseAuthorization();
    app.UseMiddleware<SubjectContextMiddleware>();

    app.MapControllers();

    app.Run();
}
```

AquaMonitor/API/ DependencyInjection.cs

```
public static class DependencyInjection
{
    public static IServiceCollection AddPresentation(this
     IServiceCollection services)
    {
        services.AddCors(options =>
        {
            options.AddPolicy("AllowViteFrontend", policyBuilder
     =>
            {

     policyBuilder.WithOrigins("http://localhost:5173")
                    .AllowAnyHeader()
                    .AllowAnyMethod()
                    .AllowCredentials();
```

```
            });
        });

        services.AddMapping();

        return services;
    }
}
```

AquaMonitor/API/ AnalyticsController.cs

```
[Route("analytics")]
[Authorize]
public class AnalyticsController : ApiController
{
    private readonly ISender _mediator;
    private readonly IMapper _mapper;

    public AnalyticsController(ISender mediator, IMapper mapper)
    {
        _mediator = mediator;
        _mapper = mapper;
    }

    [HttpGet("subjects/{edrpouCode}/water-usage-trends")]
    public async Task<IActionResult> GetWaterUsageTrends(
        [FromRoute] string edrpouCode,
        [FromQuery] int startYear,
        [FromQuery] int endYear,
        CancellationToken cancellationToken)
    {
        var query = new GetWaterUsageTrendsQuery(edrpouCode,
     startYear, endYear);
        var result = await _mediator.Send(query,
     cancellationToken);

        return result.Match(
            trends => Ok(trends),
            errors => Problem(errors)
        );
    }

    [HttpGet("water-bodies/load")]
    public async Task<IActionResult> GetWaterBodiesLoad(
        [FromQuery] int startYear,
        [FromQuery] int endYear,
        [FromQuery] string katottgCode,
        [FromQuery] string? stationCode,
        CancellationToken cancellationToken)
    {
        var query = new GetWaterBodyLoadQuery(startYear,
     endYear, katottgCode, stationCode);
        var result = await _mediator.Send(query,
     cancellationToken);
```

```
        return result.Match(
            loads => Ok(loads),
            errors => Problem(errors)
        );
    }
}
```

AquaMonitor/API/ EcologicalPermitsController.cs

```
[Route("ecological-permits")]
[Authorize]
public class EcologicalPermitsController : ApiController
{
    private readonly ISender _mediator;
    private readonly IMapper _mapper;

    public EcologicalPermitsController(ISender mediator, IMapper
     mapper)
    {
        _mediator = mediator;
        _mapper = mapper;
    }

    [HttpPost]
    [Authorize(Roles = "SystemAdmin")]
    public async Task<IActionResult>
     Create(CreateEcologicalPermitRequest request)
    {
        var command =
     _mapper.Map<CreateEcologicalPermitCommand>(request);

        var result = await _mediator.Send(command);

        return result.Match(
            commandResult => Ok(commandResult),
            errors => Problem(errors)
        );
    }

    [HttpPut("{id:guid}")]
    [Authorize(Roles = "SystemAdmin")]
    public async Task<IActionResult> Update(Guid id,
     UpdateEcologicalPermitRequest request)
    {
        var command =
     _mapper.Map<UpdateEcologicalPermitCommand>(request);
        command = command with { Id = id };

        var result = await _mediator.Send(command);

        return result.Match(
            _ => NoContent(),
            errors => Problem(errors)
```

```
        );
    }

    [HttpDelete("{id:guid}")]
    [Authorize(Roles = "SystemAdmin")]
    public async Task<IActionResult> Delete(Guid id)
    {
        var command = new DeleteEcologicalPermitCommand(id);

        var result = await _mediator.Send(command);

        return result.Match(
            _ => NoContent(),
            errors => Problem(errors)
        );
    }

    [HttpGet("{id:guid}")]
    [RequireRole(requireStrictSubjectContext: true,
     RepresentativeRole.EnterpriseOwner,
     RepresentativeRole.EnterpriseEcologist)]
    public async Task<IActionResult> GetById(Guid id)
    {
        Guid? activeSubjectId =
     HttpContext.Items.TryGetValue("ActiveSubjectId", out var
     obj) && obj is MonitoringSubjectId subjectId
            ? subjectId.Value
            : null;

        var query = new
     GetEcologicalPermitByIdQuery(activeSubjectId, id);

        var result = await _mediator.Send(query);

        return result.Match(
            queryResult => Ok(queryResult),
            errors => Problem(errors)
        );
    }

    [HttpGet]
    [Authorize(Roles =
     "NationalInspector,BasinWaterManager,SystemAdmin")]
    public async Task<IActionResult> GetAllHeaders()
    {
        var query = new GetAllEcologicalPermitHeadersQuery();

        var result = await _mediator.Send(query);

        return result.Match(
            queryResult => Ok(queryResult),
            errors => Problem(errors)
        );
```

```
}

[HttpGet("me")]
[RequireRole(requireStrictSubjectContext: false,
 RepresentativeRole.EnterpriseOwner,
 RepresentativeRole.EnterpriseEcologist)]
public async Task<IActionResult> GetMy()
{
    var activeSubjectId =
 ((MonitoringSubjectId)HttpContext.Items["ActiveSubjectId"]!)
 .Value;

    var query = new
 GetMyEcologicalPermitHeadersQuery(activeSubjectId);

    var result = await _mediator.Send(query);

    return result.Match(
        queryResult => Ok(queryResult),
        errors => Problem(errors)
    );
}

[HttpGet("balances")]
[RequireRole(requireStrictSubjectContext: true,
 RepresentativeRole.EnterpriseOwner,
 RepresentativeRole.EnterpriseEcologist)]
public async Task<IActionResult> GetBalances(
    [FromQuery] Guid? permitId,
    [FromQuery] int? year,
    CancellationToken cancellationToken)
{
    if (!permitId.HasValue && !year.HasValue)
    {
        return BadRequest(new { error = "At least one
 parameter (permitId or year) must be provided." });
    }

    if (year.HasValue && (year < 2000 || year > 2100))
    {
        return BadRequest(new { error = "Invalid calculation
 year parameter." });
    }

    int targetYear = year ?? DateTime.UtcNow.Year;

    Guid? activeSubjectId =
 HttpContext.Items.TryGetValue("ActiveSubjectId", out var
 obj) && obj is MonitoringSubjectId subjectId
        ? subjectId.Value
        : null;
```

```
        var query = new
    GetEcologicalPermitBalancesQuery(activeSubjectId, permitId,
    targetYear);

        var result = await _mediator.Send(query,
    cancellationToken);

        return result.Match(
            balances => Ok(balances),
            errors => Problem(errors)
        );
    }
}
```

AquaMonitor/API/ WaterUsageReportsController.cs

```
[Route("water-usage-reports")]
[Authorize]
public class WaterUsageReportsController : ApiController
{
    private readonly ISender _mediator;
    private readonly IMapper _mapper;

    public WaterUsageReportsController(ISender mediator, IMapper
     mapper)
    {
        _mediator = mediator;
        _mapper = mapper;
    }

    [HttpGet("{id:guid}")]
    [RequireRole(requireStrictSubjectContext: true,
     RepresentativeRole.EnterpriseOwner,
     RepresentativeRole.EnterpriseEcologist,
        RepresentativeRole.NationalInspector,
     RepresentativeRole.BasinWaterManager,
     RepresentativeRole.SystemAdmin)]
    public async Task<IActionResult> GetById(Guid id)
    {
        Guid? activeSubjectId =
     HttpContext.Items.TryGetValue("ActiveSubjectId", out var
     obj) && obj is MonitoringSubjectId subjectId
            ? subjectId.Value
            : null;

        var query = new
     GetWaterUsageReportByIdQuery(activeSubjectId, id);

        var result = await _mediator.Send(query);

        return result.Match(
            queryResult => Ok(queryResult),
            errors => Problem(errors)
        );
```

```
    }

    [HttpGet]
    [RequireRole(requireStrictSubjectContext: true,
     RepresentativeRole.NationalInspector,
     RepresentativeRole.BasinWaterManager,
        RepresentativeRole.SystemAdmin)]
    public async Task<IActionResult> GetAllHeaders()
    {
        var query = new GetAllWaterUsageHeadersQuery();

        var result = await _mediator.Send(query);

        return result.Match(
            queryResult => Ok(queryResult),
            errors => Problem(errors)
        );
    }

    [HttpGet("me")]
    [RequireRole(requireStrictSubjectContext: false,
     RepresentativeRole.EnterpriseOwner,
     RepresentativeRole.EnterpriseEcologist)]
    public async Task<IActionResult> GetMy(CancellationToken
     cancellationToken)
    {
        var activeSubjectId =
     ((MonitoringSubjectId)HttpContext.Items["ActiveSubjectId"]!)
     .Value;

        var query = new
     GetMyWaterUsageReportHeadersQuery(activeSubjectId);

        var result = await _mediator.Send(query,
     cancellationToken);

        return result.Match(
            subjects => Ok(subjects),
            errors => Problem(errors)
        );
    }
}
```

AquaMonitor/API/ WaterQualityProtocolsController.cs

```
[Route("water-quality-protocols")]
[Authorize]
public class WaterQualityProtocolsController : ApiController
{
    private readonly ISender _mediator;
    private readonly IMapper _mapper;

    public WaterQualityProtocolsController(ISender mediator,
     IMapper mapper)
```

```
    {
        _mediator = mediator;
        _mapper = mapper;
    }

    [HttpGet("{id:guid}")]
    [RequireRole(requireStrictSubjectContext: true,
     RepresentativeRole.NationalInspector,
     RepresentativeRole.StateLaboratoryEcologist)]
    public async Task<IActionResult> GetById(Guid id)
    {
        var query = new GetWaterQualityProtocolByIdQuery(id);

        var result = await _mediator.Send(query);

        return result.Match(
            queryResult => Ok(queryResult),
            errors => Problem(errors)
        );
    }

    [HttpGet]
    [RequireRole(requireStrictSubjectContext: true,
     RepresentativeRole.NationalInspector,
     RepresentativeRole.StateLaboratoryEcologist)]
    public async Task<IActionResult> GetAllHeaders()
    {
        var query = new
     GetAllWaterQualityProtocolHeadersQuery();

        var result = await _mediator.Send(query);

        return result.Match(
            queryResult => Ok(queryResult),
            errors => Problem(errors)
        );
    }
}
```

AquaMonitor/API/ EnvironmentalStandardsController.cs

```
[Route("environmental-quality-standards")]
[Authorize]
public class EnvironmentalStandardsController : ApiController
{
    private readonly ISender _mediator;
    private readonly IMapper _mapper;

    public EnvironmentalStandardsController(ISender mediator,
     IMapper mapper)
    {
        _mediator = mediator;
        _mapper = mapper;
    }
```

```
[HttpPost]
public async Task<IActionResult>
 Create(CreateEnvironmentalStandardRequest request)
{
    var command =
 _mapper.Map<CreateEnvironmentalStandardCommand>(request);

    var result = await _mediator.Send(command);

    return result.Match(
        commandResult => Ok(commandResult),
        errors => Problem(errors)
    );
}

[HttpPut("{id:guid}")]
public async Task<IActionResult> Update(Guid id,
 UpdateEnvironmentalStandardRequest request)
{
    var command =
 _mapper.Map<UpdateEnvironmentalStandardCommand>(request);
    command = command with { Id = id };

    var result = await _mediator.Send(command);

    return result.Match(
        _ => NoContent(),
        errors => Problem(errors)
    );
}

[HttpDelete("{id:guid}")]
public async Task<IActionResult> Delete(Guid id)
{
    var command = new
 DeleteEnvironmentalStandardCommand(id);

    var result = await _mediator.Send(command);

    return result.Match(
        _ => NoContent(),
        errors => Problem(errors)
    );
}

[HttpGet("{id:guid}")]
public async Task<IActionResult> GetById(Guid id)
{
    var query = new GetEnvironmentalStandardByIdQuery(id);

    var result = await _mediator.Send(query);
```

```
        return result.Match(
            queryResult => Ok(queryResult),
            errors => Problem(errors)
        );
    }

    [HttpGet]
    public async Task<IActionResult> GetAll()
    {
        var query = new GetAllEnvironmentalStandardQuery();

        var result = await _mediator.Send(query);

        return result.Match(
            queryResult => Ok(queryResult),
            errors => Problem(errors)
        );
    }
}
```

AquaMonitor/Application/DependencyInjection.cs

```
public static class DependencyInjection
{
    public static IServiceCollection AddApplication(this
     IServiceCollection services)
    {

     services.AddValidatorsFromAssembly(Assembly.GetExecutingAsse
     mbly());

        services.AddMediatR(cfg =>
        {

     cfg.RegisterServicesFromAssembly(typeof(DependencyInjection)
     .Assembly);

            cfg.AddOpenBehavior(typeof(ValidationBehavior<,>));
        });

        return services;
    }
}
```

AquaMonitor/Application/Analytics/Queries/GetWaterBodyLoad.cs

```
public class GetWaterBodyLoadQueryHandler
    : IRequestHandler<GetWaterBodyLoadQuery,
     ErrorOr<WaterBodyLoadResponse>>
{
    private readonly IApplicationDbContext _context;

    public GetWaterBodyLoadQueryHandler(IApplicationDbContext
     context)
```

```
{
    _context = context;
}

public async Task<ErrorOr<WaterBodyLoadResponse>>
 Handle(GetWaterBodyLoadQuery request, CancellationToken
 cancellationToken)
{
    var startBoundary = new DateTime(request.StartYear, 1,
 1);
    var endBoundary = new DateTime(request.EndYear, 12, 31,
 23, 59, 59);

    var katottgCodeResult =
 KatottgCode.Create(request.KatottgCode);
    if (katottgCodeResult.IsError)
    {
        return katottgCodeResult.Errors;
    }

    var katottgCode = katottgCodeResult.Value;

    var objectQuery =
 _context.MonitoringObjects.AsNoTracking()
        .Where(o => o.KatottgCode == katottgCode);

    if (!string.IsNullOrWhiteSpace(request.StationCode))
    {
        objectQuery = objectQuery.Where(o => o.StationCode
 == request.StationCode);
    }

    var groupedData = await _context.WaterUsageReports
        .AsNoTracking()
        .Where(report => report.ReportingPeriodStart >=
 startBoundary && report.ReportingPeriodStart <= endBoundary)
        .SelectMany(report => report.Records)
        .Join(
            objectQuery,
            record => record.MonitoringObjectId,
            obj => obj.Id,
            (record, obj) => new
            {
                ObjectId = obj.Id,
                ObjectName = obj.Name,
                record.RecordedFlowDirection,
                record.Volume
            })
        .GroupBy(x => new { x.ObjectId, x.ObjectName })
        .Select(g => new
        {
            g.Key.ObjectId,
            g.Key.ObjectName,
```

```
                TotalIntake = g.Sum(x => x.RecordedFlowDirection
        == FlowDirection.Inflow ? x.Volume : 0),
                TotalDischarge = g.Sum(x =>
        x.RecordedFlowDirection == FlowDirection.Outflow ? x.Volume
        : 0)
            })
            .OrderByDescending(x => x.TotalIntake)
            .ToListAsync(cancellationToken);

        var loads = groupedData
            .Select(x => new WaterBodyLoadItem(
                x.ObjectId.Value,
                x.ObjectName,
                x.TotalIntake,
                x.TotalDischarge,
                x.TotalIntake - x.TotalDischarge
            ))
            .ToList();

        return new WaterBodyLoadResponse(loads);
    }
}
```

AquaMonitor/Application/Analytics/Queries/ GetWaterUsageTrends.cs

```
public class
    GetWaterUsageTrendsQueryHandler :
     IRequestHandler<GetWaterUsageTrendsQuery,
     ErrorOr<WaterUsageTrendResponse>>
{
    private readonly IApplicationDbContext _context;

    public GetWaterUsageTrendsQueryHandler(IApplicationDbContext
     context)
    {
        _context = context;
    }

    public async Task<ErrorOr<WaterUsageTrendResponse>>
     Handle(GetWaterUsageTrendsQuery request,
        CancellationToken cancellationToken)
    {
        var edrpouResult =
     EdrpouCode.Create(request.EdrpouCode);
        if (edrpouResult.IsError)
        {
            return edrpouResult.Errors;
        }

        var edrpou = edrpouResult.Value;

        var subject = await _context.MonitoringSubjects
            .AsNoTracking()
            .Where(s => s.EdrpouCode == edrpou)
```

```
        .Select(s => new { s.Id, s.Name, EdrpouCode =
s.EdrpouCode.Value })
        .FirstOrDefaultAsync(cancellationToken);

    if (subject is null)
    {
        return Error.NotFound(
            code: "MonitoringSubject.NotFound",
            description: "No entity with the specified
EDRPOU code was found.");
    }

    var startBoundary = new DateTime(request.StartYear, 1,
1);
    var endBoundary = new DateTime(request.EndYear, 12, 31,
23, 59, 59);

    var groupedData = await _context.WaterUsageReports
        .AsNoTracking()
        .Where(r => r.MonitoringSubjectId == subject.Id
                    && r.ReportingPeriodStart >=
startBoundary
                    && r.ReportingPeriodStart <=
endBoundary)
        .SelectMany(
            report => report.Records,
            (report, record) => new
            {
                Year = report.ReportingPeriodStart.Year,
                record.RecordedFlowDirection,
                record.Volume
            })
        .GroupBy(x => x.Year)
        .Select(g => new
        {
            Year = g.Key,
            TotalInflow = g.Sum(x => x.RecordedFlowDirection
== FlowDirection.Inflow ? x.Volume : 0),
            TotalOutflow = g.Sum(x =>
x.RecordedFlowDirection == FlowDirection.Outflow ? x.Volume
: 0)
        })
        .OrderBy(t => t.Year)
        .ToListAsync(cancellationToken);

    var trends = groupedData
        .Select(x => new YearlyWaterUsageTrend(
            x.Year,
            x.TotalInflow,
            x.TotalOutflow,
            x.TotalInflow - x.TotalOutflow
        ))
        .ToList();
```

```
        return new WaterUsageTrendResponse(
            subject.Id.Value,
            subject.Name,
            subject.EdrpouCode,
            trends
        );
    }
}
```

AquaMonitor/Application/EcologicalPermits/Queries/GetBalance.cs

```
public class GetEcologicalPermitBalancesQueryHandler
    : IRequestHandler<GetEcologicalPermitBalancesQuery,
     ErrorOr<IReadOnlyList<PermitBalanceResponse>>>
{
    private readonly IApplicationDbContext _context;

    public
     GetEcologicalPermitBalancesQueryHandler(IApplicationDbContex
     t context)
    {
        _context = context;
    }

    public async
     Task<ErrorOr<IReadOnlyList<PermitBalanceResponse>>> Handle(
        GetEcologicalPermitBalancesQuery request,
     CancellationToken cancellationToken)
    {
        var permitQuery = _context.EcologicalPermits
            .Include(p => p.Limits)
            .AsNoTracking()
            .AsQueryable();

        if (request.PermitId.HasValue)
        {
            var permitId =
     EcologicalPermitId.Create(request.PermitId.Value);
            permitQuery = permitQuery.Where(p => p.Id ==
     permitId);
        }

        if (request.ActiveSubjectId.HasValue)
        {
            var subjectId =
     MonitoringSubjectId.Create(request.ActiveSubjectId.Value);
            permitQuery = permitQuery.Where(p =>
     p.MonitoringSubjectId == subjectId);
        }

        var targetDateStart = new
     DateTime(request.CalculationYear, 1, 1).ToUniversalTime();
```

```
    var targetDateEnd = new
DateTime(request.CalculationYear, 12, 31, 23, 59,
59).ToUniversalTime();

    permitQuery = permitQuery.Where(p =>
        p.IssueDate <= targetDateEnd && p.ExpirationDate >=
targetDateStart);

    var permits = await
permitQuery.ToListAsync(cancellationToken);

    if (!permits.Any())
    {
        return new List<PermitBalanceResponse>();
    }

    var subjectIds = permits.Select(p =>
p.MonitoringSubjectId).Distinct().ToList();

    var reports = await _context.WaterUsageReports
        .Include(r => r.Records)
        .AsNoTracking()
        .Where(r =>
subjectIds.Contains(r.MonitoringSubjectId)
                    && r.ReportingPeriodEnd.Year ==
request.CalculationYear
                    && r.ReportStatus != ReportStatus.Draft)
        .ToListAsync(cancellationToken);

    var allRecordsForYear = reports.SelectMany(r =>
r.Records).ToList();

    var allLimits = permits.SelectMany(p =>
p.Limits).ToList();
    var objectIds = allLimits.Select(l =>
l.MonitoringObjectId).Distinct().ToList();
    var workIds = allLimits.Select(l =>
l.WaterManagementWorkId).Distinct().ToList();

    var objectNames = await _context.MonitoringObjects
        .Where(o => objectIds.Contains(o.Id))
        .ToDictionaryAsync(o => o.Id, o => o.Name,
cancellationToken);

    var workNames = await _context.WaterManagementWorks
        .Where(w => workIds.Contains(w.Id))
        .ToDictionaryAsync(w => w.Id, w => w.Name,
cancellationToken);

    var result = new List<PermitBalanceResponse>();

    foreach (var permit in permits)
    {
```

```
        var objectBalances = new
List<ObjectBalanceResponse>();
        var groupedLimits = permit.Limits.GroupBy(l =>
l.MonitoringObjectId);

        foreach (var objectGroup in groupedLimits)
        {
            var objectId = objectGroup.Key;
            var objectName =
objectNames.GetValueOrDefault(objectId, "Невідомо");

            var worksBalance = new
List<WorkTypeBalanceResponse>();

            foreach (var limit in objectGroup)
            {
                var usedVolume = allRecordsForYear
                    .Where(r => r.PermitLimitId == limit.Id)
                    .Sum(r => r.Volume);

                var remainingVolume = limit.VolumeLimit -
usedVolume;

                var workName =
workNames.GetValueOrDefault(limit.WaterManagementWorkId,
"Невідомо");

                worksBalance.Add(new
WorkTypeBalanceResponse(
                    limit.WaterManagementWorkId.Value,
                    workName,
                    limit.VolumeLimit,
                    usedVolume,
                    remainingVolume
                ));
            }

            objectBalances.Add(new ObjectBalanceResponse(
                objectId.Value,
                objectName,
                worksBalance
            ));
        }

        result.Add(new PermitBalanceResponse(
            permit.Id.Value,
            permit.PermitNumber.Value,
            request.CalculationYear,
            objectBalances
        ));
    }

    return result;
```

```
    }
}
```

AquaMonitor/Infrastructure/ DependencyInjection.cs

```
public static class DependencyInjection
{
    public static IServiceCollection AddInfrastructure(this
     IServiceCollection services, ConfigurationManager
     configuration)
    {
        services.AddDbContext<AquaMonitorDbContext>(options =>

     options.UseSqlServer(configuration.GetConnectionString("Defa
     ultConnection"),
                b =>
     b.MigrationsAssembly(typeof(AquaMonitorDbContext).Assembly.F
     ullName)
            ));

        services.AddSingleton<IDateTimeProvider,
     DateTimeProvider>();
        services.AddAuth(configuration);

        services.AddScoped<IUserRepository, UserRepository>();
        services.AddScoped<IApplicationDbContext,
     AquaMonitorDbContext>();
        return services;
    }

    private static IServiceCollection AddAuth(this
     IServiceCollection services, ConfigurationManager
     configuration)
    {
        var jwtSettings = new JwtSettings();

        configuration.Bind(JwtSettings.SectionName,
     jwtSettings);

        services.AddSingleton(Options.Create(jwtSettings));
        services.AddSingleton<IJwtTokenGenerator,
     JwtTokenGenerator>();

        services.AddAuthentication(defaultScheme:
     JwtBearerDefaults.AuthenticationScheme)
            .AddJwtBearer(options =>
            {
                options.TokenValidationParameters = new
     TokenValidationParameters
                {
                    ValidateAudience = true,
                    ValidateIssuer = true,
                    ValidateLifetime = true,
                    ValidateIssuerSigningKey = true,
```

```
                    ValidIssuer = jwtSettings.Issuer,
                    ValidAudience = jwtSettings.Audience,
                    IssuerSigningKey = new
    SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings.Secr
    et))
                };

                // Блок для перехоплення та логування причин
    помилки 401
                options.Events = new JwtBearerEvents
                {
                    OnAuthenticationFailed = context =>
                    {
                        Console.WriteLine($"[AUTH FAILED]
    {context.Exception.Message}");
                        return Task.CompletedTask;
                    },
                    OnChallenge = context =>
                    {
                        Console.WriteLine($"[AUTH CHALLENGE]
    {context.Error} - {context.ErrorDescription}");
                        return Task.CompletedTask;
                    }
                };
            });

        services.AddSingleton<IPasswordHasher,
    PasswordHasher>();

        return services;
    }
}
```

AquaMonitor/Infrastructure/Persistence/AquaMonitorDbContext.cs

```
public class AquaMonitorDbContext : DbContext,
    IApplicationDbContext
{
    public
    AquaMonitorDbContext(DbContextOptions<AquaMonitorDbContext>
    options)
        : base(options) { }

    // --- 1. АДМІНІСТРАТИВНА ЧАСТИНА ---
    public DbSet<User> Users => Set<User>();
    public DbSet<SubjectRepresentative> SubjectRepresentatives
    => Set<SubjectRepresentative>();

    // --- 2. ДОВІДНИКИ ---
    public DbSet<MonitoringSubject> MonitoringSubjects =>
    Set<MonitoringSubject>();
    public DbSet<WaterManagementWork> WaterManagementWorks =>
    Set<WaterManagementWork>();
    public DbSet<Pollutant> Pollutants => Set<Pollutant>();
```

```
    public DbSet<RiverBasin> RiverBasins => Set<RiverBasin>();
    public DbSet<MonitoringObject> MonitoringObjects =>
     Set<MonitoringObject>();

    public DbSet<IndustryBranch> IndustryBranches =>
     Set<IndustryBranch>();

    public DbSet<EnvironmentalQualityStandard>
     EnvironmentalQualityStandards =>
     Set<EnvironmentalQualityStandard>();

    // --- 3. ДОКУМЕНТИ (Агрегати) ---
    public DbSet<EcologicalPermit> EcologicalPermits =>
     Set<EcologicalPermit>();
    public DbSet<PermitLimit> PermitLimits =>
     Set<PermitLimit>();
    public DbSet<WaterUsageReport> WaterUsageReports =>
     Set<WaterUsageReport>();
    public DbSet<WaterQualityProtocol> WaterQualityProtocols =>
     Set<WaterQualityProtocol>();

    // --- 4. АНАЛІТИКА ---
    public DbSet<HydrologicalBalanceAct> HydrologicalBalanceActs
     => Set<HydrologicalBalanceAct>();

    protected override void OnModelCreating(ModelBuilder
     modelBuilder)
    {
        base.OnModelCreating(modelBuilder);


     modelBuilder.ApplyConfigurationsFromAssembly(typeof(AquaMoni
     torDbContext).Assembly);
    }
}
```

AquaMonitor/Infrastructure/Persistence/Configurations/

WaterQualityProtocolConfigurations.cs

```
public class WaterQualityProtocolConfigurations :
     IEntityTypeConfiguration<WaterQualityProtocol>
{
    public void
     Configure(EntityTypeBuilder<WaterQualityProtocol> builder)
    {
        ConfigureBase(builder);
        ConfigureProperties(builder);
        ConfigureRelationships(builder);
    }

    private void
     ConfigureBase(EntityTypeBuilder<WaterQualityProtocol>
     builder)
```

```
{
    builder.ToTable("WaterQualityProtocols");

    builder.HasKey(x => x.Id);
    builder.Property(x => x.Id)
        .ValueGeneratedNever()
        .HasConversion(
            id => id.Value,
            value => WaterQualityProtocolId.Create(value));
}

private void
 ConfigureProperties(EntityTypeBuilder<WaterQualityProtocol>
 builder)
{
    builder.Property(x => x.ProtocolNumber)
        .HasMaxLength(50)
        .IsRequired();

    builder.HasIndex(x => x.ProtocolNumber)
        .IsUnique();

    builder.Property(x => x.LaboratoryName)
        .HasMaxLength(200)
        .IsRequired();

    builder.Property(x => x.SamplingDate)
        .HasColumnType("date")
        .IsRequired();

    builder.Property(x => x.AnalysisDate)
        .HasColumnType("date")
        .IsRequired();

    builder.Property(x => x.Status)
        .HasConversion<string>()
        .HasMaxLength(20)
        .IsRequired();
}

private void
 ConfigureRelationships(EntityTypeBuilder<WaterQualityProtoco
 l> builder)
{
    builder.Property(x => x.MonitoringObjectId)
        .HasConversion(id => id.Value, v =>
 MonitoringObjectId.Create(v))
        .IsRequired();

    builder.HasOne<MonitoringObject>()
        .WithMany()
        .HasForeignKey(x => x.MonitoringObjectId)
        .OnDelete(DeleteBehavior.Restrict);
```

```
        var navigation =
    builder.Metadata.FindNavigation(nameof(WaterQualityProtocol.
    Results));

    navigation!.SetPropertyAccessMode(PropertyAccessMode.Field);

        builder.HasMany(x => x.Results)
            .WithOne()
            .HasForeignKey(x => x.WaterQualityProtocolId)
            .OnDelete(DeleteBehavior.Cascade);
    }
}

public class QualityParameterResultConfigurations :
    IEntityTypeConfiguration<QualityParameterResult>
{
    public void
    Configure(EntityTypeBuilder<QualityParameterResult> builder)
    {
        builder.ToTable("QualityParameterResults");

        builder.HasKey(x => x.Id);
        builder.Property(x => x.Id)
            .ValueGeneratedNever()
            .HasConversion(
                id => id.Value,
                value =>
    QualityParameterResultId.Create(value));

        builder.Property(x => x.MethodologyCode)
            .HasMaxLength(50)
            .IsRequired();

        builder.Property(x => x.IsBelowLimitOfQuantification)
            .IsRequired();

        builder.OwnsOne(x => x.Value, nav =>
        {
            nav.Property(c => c.ValueMicrograms)
                .HasColumnName("ConcentrationValue")
                .IsRequired();
        });

        builder.Property(x => x.PollutantId)
            .HasConversion(id => id.Value, v =>
    PollutantId.Create(v))
            .IsRequired();

        builder.HasOne<Pollutant>()
            .WithMany()
            .HasForeignKey(x => x.PollutantId)
            .OnDelete(DeleteBehavior.Restrict);
```

```
        builder.Property(x => x.WaterQualityProtocolId)
            .HasConversion(id => id.Value, v =>
     WaterQualityProtocolId.Create(v));
    }
}
```

AquaMonitor/Infrastructure/Persistence/Configurations/

WaterUsageReportConfigurations.cs

```
public class WaterUsageReportConfigurations :
     IEntityTypeConfiguration<WaterUsageReport>
{
    public void Configure(EntityTypeBuilder<WaterUsageReport>
     builder)
    {
        ConfigureBase(builder);
        ConfigureProperties(builder);
        ConfigureRelationships(builder);
    }

    private void
     ConfigureBase(EntityTypeBuilder<WaterUsageReport> builder)
    {
        builder.ToTable("WaterUsageReports");

        builder.HasKey(x => x.Id);
        builder.Property(x => x.Id)
            .ValueGeneratedNever()
            .HasConversion(
                id => id.Value,
                value => WaterUsageReportId.Create(value));
    }

    private void
     ConfigureProperties(EntityTypeBuilder<WaterUsageReport>
     builder)
    {
        builder.Property(x => x.Name)
            .HasMaxLength(200)
            .IsRequired();

        builder.Property(x => x.Description)
            .HasMaxLength(1000);

        builder.Property(x => x.ReportNumber)
            .HasMaxLength(50)
            .IsRequired();

        builder.HasIndex(x => x.ReportNumber)
            .IsUnique();

        builder.Property(x => x.SubmissionDate);
```

```
        builder.Property(x => x.ReportingPeriodStart)
            .HasColumnType("date")
            .IsRequired();

        builder.Property(x => x.ReportingPeriodEnd)
            .HasColumnType("date")
            .IsRequired();

        builder.Property(x => x.ReportStatus)
            .HasConversion<string>()
            .HasMaxLength(20)
            .IsRequired();
    }

    private void
     ConfigureRelationships(EntityTypeBuilder<WaterUsageReport>
     builder)
    {
        builder.Property(x => x.MonitoringSubjectId)
            .HasConversion(id => id.Value, v =>
     MonitoringSubjectId.Create(v))
            .IsRequired();

        builder.HasOne<MonitoringSubject>()
            .WithMany()
            .HasForeignKey(x => x.MonitoringSubjectId)
            .OnDelete(DeleteBehavior.Restrict);

        var navigation =
     builder.Metadata.FindNavigation(nameof(WaterUsageReport.Reco
     rds));

     navigation.SetPropertyAccessMode(PropertyAccessMode.Field);

        builder.HasMany(x => x.Records)
            .WithOne()
            .HasForeignKey(x => x.WaterUsageReportId)
            .OnDelete(DeleteBehavior.Cascade);
    }
}

public class WaterUsageRecordConfigurations :
     IEntityTypeConfiguration<WaterUsageRecord>
{
    public void Configure(EntityTypeBuilder<WaterUsageRecord>
     builder)
    {
        builder.ToTable("WaterUsageRecords");

        builder.HasKey(x => x.Id);
        builder.Property(x => x.Id)
            .ValueGeneratedNever()
```

```
            .HasConversion(id => id.Value, v =>
        WaterUsageRecordId.Create(v));

            builder.Property(x => x.WaterUsageReportId)
                .HasConversion(id => id.Value, v =>
        WaterUsageReportId.Create(v));

            builder.Property(x => x.MonitoringObjectId)
                .HasConversion(id => id.Value, v =>
        MonitoringObjectId.Create(v))
                .IsRequired();

            builder.Property(x => x.WaterManagementWorkId)
                .HasConversion(id => id.Value, v =>
        WaterManagementWorkId.Create(v))
                .IsRequired();

            builder.Property(x => x.PermitLimitId)
                .HasConversion(id => id.Value, v =>
        PermitLimitId.Create(v))
                .IsRequired();

            builder.Property(x => x.EcologicalPermitId)
                .HasConversion(id => id.Value, v =>
        EcologicalPermitId.Create(v))
                .IsRequired();

            builder.Property(x => x.Volume).IsRequired();

            builder.Property(x => x.RecordedFlowDirection)
                .HasConversion<string>()
                .HasMaxLength(20)
                .IsRequired();

            builder.Property(x => x.RecordedSourceType)
                .HasConversion<string>()
                .HasMaxLength(20)
                .IsRequired();
        }
    }
```

AquaMonitor/Infrastructure/Persistence/Configurations/

EcologicalPermitConfigurations.cs

```
public class EcologicalPermitConfigurations :
        IEntityTypeConfiguration<EcologicalPermit>
    {
        public void Configure(EntityTypeBuilder<EcologicalPermit>
        builder)
        {
            ConfigureBase(builder);
            ConfigureProperties(builder);
            ConfigureRelationships(builder);
```

```
}

private void
 ConfigureBase(EntityTypeBuilder<EcologicalPermit> builder)
{
    builder.ToTable("EcologicalPermits");

    builder.HasKey(x => x.Id);

    builder.Property(x => x.Id)
        .ValueGeneratedNever()
        .HasConversion(
            id => id.Value,
            value => EcologicalPermitId.Create(value));
}

private void
 ConfigureProperties(EntityTypeBuilder<EcologicalPermit>
 builder)
{
    builder.Property(x => x.PermitNumber)
        .HasConversion(
            number => number.Value,
            value =>
 EcologicalPermitNumber.Create(value).Value
        )
        .HasColumnName("PermitNumber")
        .HasMaxLength(50)
        .IsRequired()

 .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

    builder.HasIndex(x => x.PermitNumber)
        .IsUnique();

    builder.Property(x => x.IssuingAuthority)
        .HasMaxLength(200)
        .IsRequired()

 .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

    builder.Property(x => x.IssueDate)
        .HasColumnType("date")
        .IsRequired()

 .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

    builder.Property(x => x.ExpirationDate)
        .HasColumnType("date")
        .IsRequired()

 .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);
}
```

```
    private void
     ConfigureRelationships(EntityTypeBuilder<EcologicalPermit>
     builder)
    {
        builder.Property(x => x.MonitoringSubjectId)
            .HasConversion(
                id => id.Value,
                value => MonitoringSubjectId.Create(value))
            .IsRequired()

     .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

        builder.HasOne<MonitoringSubject>()
            .WithMany()
            .HasForeignKey(x => x.MonitoringSubjectId)
            .OnDelete(DeleteBehavior.Restrict);

        builder.HasMany(x => x.Limits)
            .WithOne()
            .HasForeignKey(x => x.EcologicalPermitId)
            .OnDelete(DeleteBehavior.Cascade);


     builder.Metadata.FindNavigation(nameof(EcologicalPermit.Limi
     ts))!
            .SetPropertyAccessMode(PropertyAccessMode.Field);
    }
}

public class PermitLimitConfigurations :
     IEntityTypeConfiguration<PermitLimit>
{
    public void Configure(EntityTypeBuilder<PermitLimit>
     builder)
    {
        builder.ToTable("PermitLimits");

        builder.HasKey(x => x.Id);

        builder.Property(x => x.Id)
            .ValueGeneratedNever()
            .HasConversion(
                id => id.Value,
                value => PermitLimitId.Create(value));

        builder.Property(x => x.VolumeLimit)
            .IsRequired()

     .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

        builder.Property(x => x.WaterManagementWorkId)
```

```
            .HasConversion(id => id.Value, v =>
        WaterManagementWorkId.Create(v))
            .IsRequired()

        .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

            builder.Property(x => x.MonitoringObjectId)
            .HasConversion(id => id.Value, value =>
        MonitoringObjectId.Create(value))
            .IsRequired()

        .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

            builder.Property(x => x.EcologicalPermitId)
            .HasConversion(id => id.Value, v =>
        EcologicalPermitId.Create(v))
            .IsRequired()

        .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

            builder.HasIndex(x => new { x.EcologicalPermitId,
        x.WaterManagementWorkId, x.MonitoringObjectId })
            .IsUnique();
    }
}
```

AquaMonitor/Infrastructure/Persistence/Configurations/

EnvironmentalQualityStandardConfigurations.cs

```
public class EnvironmentalQualityStandardConfigurations :
    IEntityTypeConfiguration<EnvironmentalQualityStandard>
{
    public void
    Configure(EntityTypeBuilder<EnvironmentalQualityStandard>
    builder)
    {
        ConfigureBase(builder);
        ConfigureSimpleProperties(builder);
        ConfigureValueObjects(builder);
        ConfigureRelationships(builder);
    }

    private void
    ConfigureBase(EntityTypeBuilder<EnvironmentalQualityStandard
    > builder)
    {
        builder.ToTable("EnvironmentalQualityStandards");

        builder.HasKey(x => x.Id);

        builder.Property(x => x.Id)
            .ValueGeneratedNever()
            .HasConversion(
```

```
            id => id.Value,
            value =>
 EnvironmentalQualityStandardId.Create(value));
}

private void
 ConfigureSimpleProperties(EntityTypeBuilder<EnvironmentalQua
 lityStandard> builder)
{
    builder.Property(x => x.WaterMacroCategory)
        .HasConversion<string>()
        .HasMaxLength(50)
        .IsRequired()

 .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

    builder.Property(x => x.ValidFrom)
        .HasColumnType("date")
        .IsRequired()

 .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

    builder.Property(x => x.ValidTo)
        .HasColumnType("date")
        .IsRequired(false);
}

private void
 ConfigureValueObjects(EntityTypeBuilder<EnvironmentalQuality
 Standard> builder)
{
    builder.Property(x => x.RegulatoryDocument)
        .HasConversion(
            doc => doc.Value,
            value =>
 RegulatoryDocumentReference.Create(value).Value
        )
        .HasColumnName("RegulatoryDocument")
        .HasMaxLength(500)
        .IsRequired()

 .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

    builder.Property(x => x.AnnualAverageLimit)
        .HasConversion(
            c => c.ValueMicrograms,
            v => Concentration.FromMicrograms(v).Value
        )
        .HasColumnName("AnnualAverage_Value")
        .IsRequired(false)

 .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);
```

```
            builder.Property(x => x.MaxAllowableConcentration)
                .HasConversion(
                    c => c.ValueMicrograms,
                    v => Concentration.FromMicrograms(v).Value
                )
                .HasColumnName("MAC_Value")
                .IsRequired(false)

        .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

            builder.Property(x => x.PnecLimit)
                .HasConversion(
                    c => c.ValueMicrograms,
                    v => Concentration.FromMicrograms(v).Value
                )
                .HasColumnName("PNEC_Value")
                .IsRequired(false)

        .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

            builder.Property(x => x.BiotaLimit)
                .HasConversion(
                    c => c.ValueMicrograms,
                    v => Concentration.FromMicrograms(v).Value
                )
                .HasColumnName("Biota_Value")
                .IsRequired(false)

        .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);
        }

        private void
        ConfigureRelationships(EntityTypeBuilder<EnvironmentalQualit
        yStandard> builder)
        {
            builder.Property(x => x.PollutantId)
                .HasConversion(
                    id => id.Value,
                    value => PollutantId.Create(value)
                )
                .IsRequired()

        .Metadata.SetAfterSaveBehavior(PropertySaveBehavior.Ignore);

            builder.HasOne<Pollutant>()
                .WithMany()
                .HasForeignKey(x => x.PollutantId)
                .OnDelete(DeleteBehavior.Restrict);

            builder.HasIndex(x => new { x.PollutantId,
        x.WaterMacroCategory, x.ValidFrom });
        }
}
```

ДОДАТОК В – КОД КЛІЄНТСЬКОЇ ЧАСТИНИ

features/analytics/pages/AnalyticsDashboardPage

```
export const AnalyticsDashboardPage = () => {
    const navigate = useNavigate();

    return (
        <div className="space-y-6">
            <div>
                <h1 className="text-2xl font-bold tracking-tight
    text-slate-900">Аналітика та звіти</h1>
                <p className="text-muted-foreground text-
    sm">Інструменти для аналізу даних водокористування та
    моніторингу</p>
            </div>

            <div className="grid grid-cols-1 md:grid-cols-2
    lg:grid-cols-3 gap-4">
                <Card className="hover:shadow-sm border-slate-
    200 transition-all flex flex-col">
                    <CardHeader className="pb-3 flex-1">
                        <CardTitle className="flex items-center
    gap-2 text-lg text-slate-800">
                            <div className="p-2 bg-blue-50
    rounded-md">
                                <LineChart className="text-blue-
    600" size={20} />
                            </div>
                            Динаміка водоспоживання
                        </CardTitle>
                        <CardDescription className="text-slate-
    500">
                            Тренд-аналіз забору та скиду води
    конкретним підприємством за вказаний період років.
                        </CardDescription>
                    </CardHeader>
                    <CardContent>
                        <Button className="w-full" onClick={()
    => navigate('/analytics/water-usage-trends')}>
                            Відкрити інструмент
                        </Button>
                    </CardContent>
                </Card>

                <Card className="hover:shadow-sm border-slate-
    200 transition-all flex flex-col">
                    <CardHeader className="pb-3 flex-1">
                        <CardTitle className="flex items-center
    gap-2 text-lg text-slate-800">
                            <div className="p-2 bg-teal-50
    rounded-md">
                                <Waves className="text-teal-600"
    size={20} />
```

```
                            </div>
                            Екологічне навантаження
                        </CardTitle>
                        <CardDescription className="text-slate-
    500">
                            Аналіз загальних об'ємів забору та
    скиду води в розрізі водних об'єктів за територіальною
    прив'язкою.
                        </CardDescription>
                    </CardHeader>
                    <CardContent>
                        <Button className="w-full" onClick={()
    => navigate('/analytics/water-bodies-load')}>
                            Відкрити інструмент
                        </Button>
                    </CardContent>
                </Card>
            </div>
        </div>
    );
};
```

features/analytics/pages/WaterUsageTrendPage

```
export const WaterUsageTrendPage = () => {
    const navigate = useNavigate();
    const currentYear = new Date().getFullYear();

    const [edrpou, setEdrpou] = useState('');
    const [startYear, setStartYear] =
     useState<string>((currentYear - 5).toString());
    const [endYear, setEndYear] =
     useState<string>(currentYear.toString());

    const [queryParams, setQueryParams] = useState<{edrpou:
     string, startYear: number, endYear: number} | null>(null);

    const { data, isLoading, isError, refetch } = useQuery({
        queryKey: ['water-usage-trends', queryParams],
        queryFn: () =>
     analyticsApi.getWaterUsageTrends(queryParams!.edrpou,
     queryParams!.startYear, queryParams!.endYear),
        enabled: !!queryParams,
        retry: false
    });

    const handleSearch = () => {
        if (edrpou.trim() && startYear && endYear) {
            const newEdrpou = edrpou.trim();
            const newStartYear = parseInt(startYear, 10);
            const newEndYear = parseInt(endYear, 10);

            if (
                queryParams?.edrpou === newEdrpou &&
```

```
            queryParams?.startYear === newStartYear &&
            queryParams?.endYear === newEndYear
        ) {
            refetch();
        } else {
            setQueryParams({
                edrpou: newEdrpou,
                startYear: newStartYear,
                endYear: newEndYear
            });
        }
    }
};

const formatVol = (val: number) => val.toLocaleString('uk-
UA', { minimumFractionDigits: 2, maximumFractionDigits: 2
});

const formatTooltip = (value: any, name: any) => {
    if (value === undefined || name === undefined) return
[];

    const translations: Record<string, string> = {
        totalIntakeVolume: 'Забір води',
        totalDischargeVolume: 'Скид води',
        consumptiveUseVolume: 'Безповоротне споживання'
    };
    return [`${formatVol(Number(value))} м³`,
translations[String(name)] || String(name)];
};

return (
    <div className="space-y-6">
        <div className="flex items-center gap-4">
            <Button variant="outline" size="icon"
onClick={() => navigate('/analytics')}>
                <ArrowLeft size={18} />
            </Button>
            <div className="flex-1">
                <h1 className="text-2xl font-bold tracking-
tight text-slate-900">
                    Динаміка водоспоживання
                </h1>
                <p className="text-muted-foreground text-sm
flex items-center gap-2 mt-1">
                    <TrendingUp size={14} /> Тренд-аналіз за
кодом ЄДРПОУ
                </p>
            </div>
        </div>

        <Card className="border-slate-200 shadow-sm bg-
white">
```

```
            <CardContent className="p-4 flex flex-col
md:flex-row gap-4 items-end">
                <div className="space-y-2 flex-1">
                    <label className="text-sm font-medium
text-slate-700">Код ЄДРПОУ</label>
                    <Input
                        placeholder="Введіть 8 або 10 цифр"
                        value={edrpou}
                        onChange={(e) =>
setEdrpou(e.target.value)}
                    />
                </div>
                <div className="space-y-2 w-full md:w-
[150px]">
                    <label className="text-sm font-medium
text-slate-700">Рік початку</label>
                    <Input
                        type="number"
                        value={startYear}
                        onChange={(e) =>
setStartYear(e.target.value)}
                    />
                </div>
                <div className="space-y-2 w-full md:w-
[150px]">
                    <label className="text-sm font-medium
text-slate-700">Рік закінчення</label>
                    <Input
                        type="number"
                        value={endYear}
                        onChange={(e) =>
setEndYear(e.target.value)}
                    />
                </div>
                <Button onClick={handleSearch} className="w-
full md:w-auto flex items-center gap-2">
                    <Search size={16} /> Аналізувати
                </Button>
            </CardContent>
        </Card>

        {isLoading && <div className="p-10 text-center text-
slate-500">Завантаження даних...</div>}

        {isError && (
            <div className="p-10 text-center text-red-500
border border-red-200 bg-red-50 rounded-md">
                Суб'єкт не знайдено або помилка завантаження
даних
            </div>
        )}

        {data && (
```

```
            <div className="space-y-6">
                <Card className="border-slate-200 shadow-
sm">
                    <CardContent className="p-4 flex items-
center gap-3 bg-slate-50 border-b border-slate-200">
                        <div className="p-2 bg-blue-100
text-blue-700 rounded-md"><Building2 size={20} /></div>
                        <div>
                            <h3 className="font-bold text-
slate-900">{data.subjectName}</h3>
                            <p className="text-sm text-
slate-500 font-mono">ЄДРПОУ: {data.edrpouCode}</p>
                        </div>
                    </CardContent>

                    {data.trends.length > 0 && (
                        <div className="p-6 border-b border-
slate-200">
                            <h4 className="text-sm font-bold
text-slate-700 mb-6 uppercase tracking-wider">Графік
динаміки</h4>
                            <div className="h-[400px] w-
full">
                                <ResponsiveContainer
width="100%" height="100%">
                                    <LineChart
                                        data={data.trends}
                                        margin={{ top: 5,
right: 30, left: 20, bottom: 5 }}
                                    >
                                        <CartesianGrid
strokeDasharray="3 3" vertical={false} stroke="#e2e8f0" />
                                        <XAxis
                                            dataKey="year"
                                            axisLine={false}
                                            tickLine={false}
                                            tick={{ fill:
'#64748b', fontSize: 12 }}
                                            dy={10}
                                        />
                                        <YAxis
                                            axisLine={false}
                                            tickLine={false}
                                            tick={{ fill:
'#64748b', fontSize: 12 }}

tickFormatter={(value) => value.toLocaleString('uk-UA')}
                                            dx={-10}
                                        />
                                        <Tooltip

formatter={formatTooltip}
```

```
                                                labelStyle={{
fontWeight: 'bold', color: '#0f172a', marginBottom: '8px' }}
                                                contentStyle={{
borderRadius: '8px', border: '1px solid #e2e8f0', boxShadow:
'0 4px 6px -1px rgb(0 0 0 / 0.1)' }}
                                            />
                                            <Legend

wrapperStyle={{paddingTop: '20px'}}

formatter={(value: any) => (
                                                    <span
className="text-slate-700 font-medium">
       {value === 'totalIntakeVolume' ? 'Забір води' :
           value === 'totalDischargeVolume' ? 'Скид води' :
               'Безповоротне споживання'}
   </span>
                                                )}
                                            />
                                            <Line
                                                type="monotone"

dataKey="totalIntakeVolume"
                                                stroke="#2563eb"
                                                strokeWidth={3}
                                                dot={{ r: 4,
strokeWidth: 2 }}
                                                activeDot={{ r:
6 }}
                                            />
                                            <Line
                                                type="monotone"

dataKey="totalDischargeVolume"
                                                stroke="#4f46e5"
                                                strokeWidth={3}
                                                dot={{ r: 4,
strokeWidth: 2 }}
                                                activeDot={{ r:
6 }}
                                            />
                                            <Line
                                                type="monotone"

dataKey="consumptiveUseVolume"
                                                stroke="#ef4444"
                                                strokeWidth={3}
                                                dot={{ r: 4,
strokeWidth: 2 }}
                                                activeDot={{ r:
6 }}
                                            />
                                        </LineChart>
```

```
                                </ResponsiveContainer>
                            </div>
                        </div>
                    )}

                    <div className="p-0">
                        <Table>
                            <TableHeader className="bg-
slate-100">
                                <TableRow>
                                    <TableHead
className="font-bold text-slate-700 w-
[100px]">Рік</TableHead>
                                    <TableHead
className="font-bold text-slate-700 text-right">Забір води
(м³)</TableHead>
                                    <TableHead
className="font-bold text-slate-700 text-right">Скид води
(м³)</TableHead>
                                    <TableHead
className="font-bold text-slate-700 text-right">Безповоротне
споживання (м³)</TableHead>
                                </TableRow>
                            </TableHeader>
                            <TableBody>
                                {data.trends.length === 0 ?
(
                                    <TableRow>
                                        <TableCell
colSpan={4} className="text-center py-8 text-slate-500">
                                            Дані за вказаний
період відсутні
                                        </TableCell>
                                    </TableRow>
                                ) : (
                                    data.trends.map((trend)
=> (
                                        <TableRow
key={trend.year} className="hover:bg-slate-50 transition-
colors">
                                            <TableCell
className="font-bold text-slate-
900">{trend.year}</TableCell>
                                            <TableCell
className="text-right font-mono text-blue-
700">{formatVol(trend.totalIntakeVolume)}</TableCell>
                                            <TableCell
className="text-right font-mono text-indigo-
700">{formatVol(trend.totalDischargeVolume)}</TableCell>
                                            <TableCell
className="text-right font-mono font-semibold text-slate-
700">{formatVol(trend.consumptiveUseVolume)}</TableCell>
                                        </TableRow>
```

```
                                        ))
                                    )}
                                </TableBody>
                            </Table>
                        </div>
                    </Card>
                </div>
            )}
        </div>
    );
};
```

features/analytics/pages/WaterBodyLoadPage

```
export const WaterBodyLoadPage = () => {
    const navigate = useNavigate();
    const currentYear = new Date().getFullYear();

    const [katottgCode, setKatottgCode] = useState('');
    const [stationCode, setStationCode] = useState('');
    const [startYear, setStartYear] =
     useState<string>((currentYear - 1).toString());
    const [endYear, setEndYear] =
     useState<string>(currentYear.toString());

    const [queryParams, setQueryParams] = useState<{katottgCode:
     string, stationCode: string, startYear: number, endYear:
     number} | null>(null);

    const { data, isLoading, isError, refetch } = useQuery({
        queryKey: ['water-bodies-load', queryParams],
        queryFn: () =>
     analyticsApi.getWaterBodiesLoad(queryParams!.startYear,
     queryParams!.endYear, queryParams!.katottgCode,
     queryParams!.stationCode),
        enabled: !!queryParams,
        retry: false
    });

    const handleSearch = () => {
        if (katottgCode.trim() && startYear && endYear) {
            const newKatottg = katottgCode.trim();
            const newStation = stationCode.trim();
            const newStartYear = parseInt(startYear, 10);
            const newEndYear = parseInt(endYear, 10);

            if (
                queryParams?.katottgCode === newKatottg &&
                queryParams?.stationCode === newStation &&
                queryParams?.startYear === newStartYear &&
                queryParams?.endYear === newEndYear
            ) {
                refetch();
            } else {
```

```
            setQueryParams({
                katottgCode: newKatottg,
                stationCode: newStation,
                startYear: newStartYear,
                endYear: newEndYear
            });
        }
    }
};

const formatVol = (val: number) => val.toLocaleString('uk-
UA', { minimumFractionDigits: 2, maximumFractionDigits: 2
});

const formatTooltip = (value: any, name: any) => {
    if (value === undefined || name === undefined) return
[];
    const translations: Record<string, string> = {
        totalIntakeVolume: 'Забір води',
        totalDischargeVolume: 'Скид води',
        consumptiveUseDelta: 'Дельта (Безповоротне)'
    };
    return [`${formatVol(Number(value))} м³`,
translations[String(name)] || String(name)];
};

return (
    <div className="space-y-6">
        <div className="flex items-center gap-4">
            <Button variant="outline" size="icon"
onClick={() => navigate('/analytics')}>
                <ArrowLeft size={18} />
            </Button>
            <div className="flex-1">
                <h1 className="text-2xl font-bold tracking-
tight text-slate-900">
                    Екологічне навантаження на водні об'єкти
                </h1>
                <p className="text-muted-foreground text-sm
flex items-center gap-2 mt-1">
                    <Waves size={14} /> Аналіз використання
водних ресурсів за територією
                </p>
            </div>
        </div>

        <Card className="border-slate-200 shadow-sm bg-
white">
            <CardContent className="p-4 grid grid-cols-1
md:grid-cols-5 gap-4 items-end">
                <div className="space-y-2 md:col-span-2">
                    <label className="text-sm font-medium
text-slate-700">Код КАТОТТГ (Обов'язково)</label>
```

```
                    <Input
                        placeholder="Введіть код території"
                        value={katottgCode}
                        onChange={(e) =>
setKatottgCode(e.target.value)}
                    />
                </div>
                <div className="space-y-2">
                    <label className="text-sm font-medium
text-slate-700">Код станції (Опціонально)</label>
                    <Input
                        placeholder="Наприклад: 12345"
                        value={stationCode}
                        onChange={(e) =>
setStationCode(e.target.value)}
                    />
                </div>
                <div className="space-y-2">
                    <label className="text-sm font-medium
text-slate-700">Початок (Рік)</label>
                    <Input
                        type="number"
                        value={startYear}
                        onChange={(e) =>
setStartYear(e.target.value)}
                    />
                </div>
                <div className="space-y-2">
                    <label className="text-sm font-medium
text-slate-700">Кінець (Рік)</label>
                    <Input
                        type="number"
                        value={endYear}
                        onChange={(e) =>
setEndYear(e.target.value)}
                    />
                </div>
                <div className="md:col-span-5 flex justify-
end">
                    <Button onClick={handleSearch}
className="w-full md:w-auto flex items-center gap-2"
disabled={!katottgCode.trim() || !startYear || !endYear}>
                        <Search size={16} /> Розрахувати
навантаження
                    </Button>
                </div>
            </CardContent>
        </Card>

        {isLoading && <div className="p-10 text-center text-
slate-500">Розрахунок даних...</div>}

        {isError && (
```

```
            <div className="p-10 text-center text-red-500
border border-red-200 bg-red-50 rounded-md">
                Помилка завантаження даних. Перевірте
правильність параметрів.
            </div>
        )}

        {data && (
            <div className="space-y-6">
                <Card className="border-slate-200 shadow-sm
overflow-hidden">
                    {data.loads.length > 0 && (
                        <div className="p-6 border-b border-
slate-200">
                            <h4 className="text-sm font-bold
text-slate-700 mb-6 uppercase tracking-wider">Порівняльний
графік об'ємів</h4>
                            <div className="h-[400px] w-
full">
                                <ResponsiveContainer
width="100%" height="100%">
                                    <BarChart
                                        data={data.loads}
                                        margin={{ top: 5,
right: 30, left: 20, bottom: 5 }}
                                    >
                                        <CartesianGrid
strokeDasharray="3 3" vertical={false} stroke="#e2e8f0" />
                                        <XAxis

dataKey="waterBodyName"
                                            axisLine={false}
                                            tickLine={false}
                                            tick={{ fill:
'#64748b', fontSize: 12 }}
                                            dy={10}
                                        />
                                        <YAxis
                                            axisLine={false}
                                            tickLine={false}
                                            tick={{ fill:
'#64748b', fontSize: 12 }}

tickFormatter={(value) => value.toLocaleString('uk-UA')}
                                            dx={-10}
                                        />
                                        <Tooltip

formatter={formatTooltip}
                                            labelStyle={{
fontWeight: 'bold', color: '#0f172a', marginBottom: '8px' }}
```

```
                                        contentStyle={{
borderRadius: '8px', border: '1px solid #e2e8f0', boxShadow:
'0 4px 6px -1px rgb(0 0 0 / 0.1)' }}
                                    />
                                    <Legend
                                        wrapperStyle={{
paddingTop: '20px' }}

formatter={(value: any) => (
                                            <span
className="text-slate-700 font-medium">
                                                {value
=== 'totalIntakeVolume' ? 'Забір води' :

value === 'totalDischargeVolume' ? 'Скид води' :

'Дельта (Безповоротне)'}
                                            </span>
                                        )}
                                    />
                                    <Bar
dataKey="totalIntakeVolume" fill="#2563eb" radius={[4, 4, 0,
0]} />
                                    <Bar
dataKey="totalDischargeVolume" fill="#4f46e5" radius={[4, 4,
0, 0]} />
                                    <Bar
dataKey="consumptiveUseDelta" fill="#ef4444" radius={[4, 4,
0, 0]} />
                                </BarChart>
                            </ResponsiveContainer>
                        </div>
                    </div>
                )}

                <div className="p-0">
                    <Table>
                        <TableHeader className="bg-
slate-100">
                            <TableRow>
                                <TableHead
className="font-bold text-slate-700">Водний
об'єкт</TableHead>
                                <TableHead
className="font-bold text-slate-700 text-right w-
[180px]">Забір води (м³)</TableHead>
                                <TableHead
className="font-bold text-slate-700 text-right w-
[180px]">Скид води (м³)</TableHead>
                                <TableHead
className="font-bold text-slate-700 text-right w-
[180px]">Дельта (м³)</TableHead>
                            </TableRow>
```

```
                                    </TableHeader>
                                    <TableBody>
                                        {data.loads.length === 0 ? (
                                            <TableRow>
                                                <TableCell
    colSpan={4} className="text-center py-8 text-slate-500">
                                                    Дані за вказаний
    період відсутні
                                                </TableCell>
                                            </TableRow>
                                        ) : (
                                            data.loads.map((load) =>
    (
                                                <TableRow
    key={load.monitoringObjectId} className="hover:bg-slate-50
    transition-colors">
                                                    <TableCell
    className="font-bold text-slate-
    900">{load.waterBodyName}</TableCell>
                                                    <TableCell
    className="text-right font-mono text-blue-
    700">{formatVol(load.totalIntakeVolume)}</TableCell>
                                                    <TableCell
    className="text-right font-mono text-indigo-
    700">{formatVol(load.totalDischargeVolume)}</TableCell>
                                                    <TableCell
    className="text-right font-mono font-semibold text-red-
    600">{formatVol(load.consumptiveUseDelta)}</TableCell>
                                                </TableRow>
                                            ))
                                        )}
                                    </TableBody>
                                </Table>
                            </div>
                        </Card>
                    </div>
                )}
            </div>
        );
};
```

src/features/ecological-permits/pages/EcologicalPermitsListPage.tsx

```
export const PermitBalanceReportPage = () => {
    const navigate = useNavigate();
    const [searchParams] = useSearchParams();

    const rawPermitId = searchParams.get('permitId');
    const permitId = rawPermitId && rawPermitId !== 'all' ?
     rawPermitId : null;

    const rawYear = searchParams.get('year');
    const year = rawYear ? parseInt(rawYear, 10) : new
     Date().getFullYear();
```

```
// Стан для фільтрації та розгортання
const [showOnlyViolations, setShowOnlyViolations] =
 useState(false);
const [collapsedObjects, setCollapsedObjects] =
 useState<Record<string, boolean>>({});

const toggleCollapse = (objectId: string) => {
    setCollapsedObjects(prev => ({ ...prev, [objectId]:
 !prev[objectId] }));
};

const { data: balances, isLoading, isError } = useQuery({
    queryKey: ['permit-balances', permitId, year],
    queryFn: () => permitBalancesApi.getBalances(permitId,
 year)
});

if (isLoading) return <div className="p-10 text-center text-
 slate-500">Розрахунок балансу...</div>;
if (isError || !balances) return <div className="p-10 text-
 center text-red-500">Помилка завантаження даних</div>;
if (balances.length === 0) return <div className="p-10 text-
 center text-slate-500">Дані за вказаними параметрами
 відсутні</div>;

const formatVol = (val: number) => val.toLocaleString('uk-
 UA', { minimumFractionDigits: 2, maximumFractionDigits: 2
 });

const getPercentageText = (used: number, allowed: number) =>
 {
    if (allowed === 0) return used > 0 ? '> 100%' : '0%';
    const percentage = Math.round((used / allowed) * 100);
    return `${percentage}%`;
};

const renderStatus = (hasViolation: boolean) => {
    if (hasViolation) {
        return <Badge variant="destructive" className="bg-
 red-50 text-red-700 border-red-200 hover:bg-red-100 shadow-
 none text-xs px-2 py-0.5">Порушення</Badge>;
    }
    return <Badge variant="outline" className="bg-green-50
 text-green-700 border-green-200 shadow-none text-xs px-2 py-
 0.5">Норма</Badge>;
};

const handleExport = (format: 'xlsx' | 'csv') => {
    if (!balances || balances.length === 0) return;

    const exportData: any[] = [];
```

```
    balances.forEach(balance => {
        const filteredObjects = balance.objects.filter(obj
=> {
            if (!showOnlyViolations) return true;
            return obj.works.some(w => w.remainingVolume <
0);
        });

        filteredObjects.forEach(obj => {
            obj.works.forEach(work => {
                const hasWorkViolation =
work.remainingVolume < 0;
                if (showOnlyViolations && !hasWorkViolation)
return;

                exportData.push({
                    'Номер дозволу': balance.permitNumber,
                    "Об'єкт моніторингу": obj.objectName,
                    'Вид робіт': work.workTypeName,
                    'Ліміт (м³)': work.allowedVolume,
                    'Використано (м³)': work.usedVolume,
                    'Залишок (м³)': work.remainingVolume,
                    'Виконання':
getPercentageText(work.usedVolume, work.allowedVolume),
                    'Статус': hasWorkViolation ? 'Порушення'
: 'Норма'
                });
            });
        });
    });

    if (exportData.length === 0) return;

    const worksheet = XLSX.utils.json_to_sheet(exportData);

    // Налаштування ширини колонок
    worksheet['!cols'] = [
        { wch: 15 }, // Номер дозволу
        { wch: 30 }, // Об'єкт
        { wch: 40 }, // Вид робіт
        { wch: 15 }, // Ліміт (Індекс 3)
        { wch: 15 }, // Використано (Індекс 4)
        { wch: 15 }, // Залишок (Індекс 5)
        { wch: 15 }, // Виконання
        { wch: 15 }, // Статус
    ];

    // Примусове застосування числового формату для Excel
    if (format === 'xlsx' && worksheet['!ref']) {
        const range =
XLSX.utils.decode_range(worksheet['!ref']);
        // Починаємо з 1, щоб пропустити рядок із
заголовками
```

```
        for (let R = range.s.r + 1; R <= range.e.r; ++R) {
            // Колонки Ліміт (3), Використано (4), Залишок
(5)
            for (let C = 3; C <= 5; ++C) {
                const cellRef = XLSX.utils.encode_cell({ c:
C, r: R });
                const cell = worksheet[cellRef];

                // Якщо комірка існує і її тип - число ('n')
                if (cell && cell.t === 'n') {
                    // Задаємо формат: роздільник тисяч, два
знаки після коми
                    cell.z = '#,##0.00';
                }
            }
        }
    }

    const workbook = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(workbook, worksheet,
'Баланс');

    const fileName =
`Баланс_водовикористання_${year}.${format}`;
    XLSX.writeFile(workbook, fileName, { bookType: format
});
};

// Розрахунок глобальних KPI
let globalAllowed = 0;
let globalUsed = 0;
let globalRemaining = 0;
let globalViolationsCount = 0;
let globalViolationVolume = 0; // Нове: Загальний об'єм
порушень

balances.forEach(b => {
    b.objects.forEach(o => {
        o.works.forEach(w => {
            globalAllowed += w.allowedVolume;
            globalUsed += w.usedVolume;
            globalRemaining += w.remainingVolume;
            if (w.remainingVolume < 0) {
                globalViolationsCount++;
                globalViolationVolume +=
Math.abs(w.remainingVolume); // Сумуємо абсолютні значення
перевищень
            }
        });
    });
});

// Допоміжний компонент для картки KPI
```

```
const KpiCard = ({ icon: Icon, title, value, unit = "м³",
 color = "default" }) => {
    const colors = {
        default: { bg: "bg-blue-50", text: "text-blue-600",
 border: "border-slate-200", valText: "text-slate-900" },
        indigo: { bg: "bg-indigo-50", text: "text-indigo-
 600", border: "border-slate-200", valText: "text-slate-900"
 },
        green: { bg: "bg-green-50", text: "text-green-600",
 border: "border-slate-200", valText: "text-emerald-600" },
        red: { bg: "bg-red-50", text: "text-red-600",
 border: "border-red-200", valText: "text-red-600" },
        neutral: { bg: "bg-slate-100", text: "text-slate-
 500", border: "border-slate-200", valText: "text-slate-900"
 },
    };
    const c = colors[color];
    const valFormatted = formatVol(value);

    return (
        <Card className={`bg-white shadow-sm ${c.border}
 overflow-hidden`}>
            <CardContent className="p-4 flex flex-col gap-
 3">
                <div className="flex items-center gap-2">
                    <div className={`p-1.5 ${c.bg} ${c.text}
 rounded-md shrink-0`}>
                        <Icon size={16} />
                    </div>
                    <p className="text-xs font-medium text-
 slate-500 truncate" title={title}>
                        {title}
                    </p>
                </div>
                <div className="min-w-0">
                    <h3 className={`text-xl font-bold
 ${c.valText} truncate`} title={`${valFormatted} ${unit}`}>
                        {valFormatted}
                        <span className="text-xs font-normal
 text-slate-500 ml-1 opacity-80">{unit}</span>
                    </h3>
                </div>
            </CardContent>
        </Card>
    );
};

return (
    <div className="space-y-6">
        <div className="flex items-center justify-between
 gap-4">
            <div className="flex items-center gap-4">
```

```
                <Button variant="outline" size="icon"
onClick={() => navigate(-1)}>
                    <ArrowLeft size={18}/>
                </Button>
                <div>
                    <h1 className="text-2xl font-bold
tracking-tight text-slate-900">
                        Зведений баланс водовикористання
                    </h1>
                    <p className="text-muted-foreground
text-sm flex items-center gap-2 mt-1">
                        <Calculator size={14}/>
Розрахунковий рік: <span
                        className="font-semibold text-slate-
700">{year}</span>
                    </p>
                </div>
            </div>

            <DropdownMenu>
                <DropdownMenuTrigger asChild>
                    <Button variant="outline"
className="flex items-center gap-2">
                        <Download size={16}/>
                        Експорт
                    </Button>
                </DropdownMenuTrigger>
                <DropdownMenuContent align="end">
                    <DropdownMenuItem onClick={() =>
handleExport('xlsx')} className="cursor-pointer">
                        Завантажити як Excel (.xlsx)
                    </DropdownMenuItem>
                    <DropdownMenuItem onClick={() =>
handleExport('csv')} className="cursor-pointer">
                        Завантажити як CSV (.csv)
                    </DropdownMenuItem>
                </DropdownMenuContent>
            </DropdownMenu>
        </div>

        {/* Віджети KPI - 5 колонок на великих екранах */}
        <div className="grid grid-cols-1 sm:grid-cols-2
lg:grid-cols-5 gap-4">
            <KpiCard icon={Droplets} title="Загальний ліміт"
value={globalAllowed} color="default"/>
            <KpiCard icon={Activity} title="Використано"
value={globalUsed} color="indigo"/>
            <KpiCard icon={Calculator} title="Чистий
залишок" value={globalRemaining}
                     color={globalRemaining < 0 ? "red" :
"green"}/>
            <KpiCard icon={TrendingDown} title="Загальний
об'єм порушень" value={globalViolationVolume}
```

```
                    color="red"/>

            <Card
                className={`${globalViolationsCount > 0 ?
'bg-red-50/50 border-red-200' : 'bg-white border-slate-200'}
shadow-sm overflow-hidden`}>
                <CardContent className="p-4 flex flex-col
gap-3">
                    <div className="flex items-center gap-
2">
                        <div
                            className={`p-1.5 rounded-md
shrink-0 ${globalViolationsCount > 0 ? 'bg-red-100 text-red-
600' : 'bg-slate-100 text-slate-400'}`}>
                            <AlertTriangle size={16}/>
                        </div>
                        <p className="text-xs font-medium
text-slate-500 truncate" title="Кількість порушень">
                            Кількість порушень
                        </p>
                    </div>
                    <div className="min-w-0">
                        <h3 className={`text-xl font-bold
${globalViolationsCount > 0 ? 'text-red-600' : 'text-slate-
900'} truncate`}
                            title={`${globalViolationsCount}
видів робіт`}>
                            {globalViolationsCount} <span
className="text-xs font-normal text-slate-500 ml-1 opacity-
80">видів робіт</span>
                        </h3>
                    </div>
                </CardContent>
            </Card>
        </div>

        {/* Панель фільтрації */}
        <div className="flex items-center gap-3 bg-slate-50
p-1.5 rounded-lg border border-slate-200 inline-flex shadow-
inner">
            <Filter size={15} className="text-slate-400 ml-
2" />
            <div className="flex bg-slate-200/50 p-0.5
rounded-md border border-slate-200">
                <Button
                    variant={!showOnlyViolations ? "default"
: "ghost"}
                    size="sm"
                    onClick={() =>
setShowOnlyViolations(false)}
                    className="h-7 text-xs px-3"
                >
                    Всі об'єкти
```

```
                </Button>
                <Button
                    variant={showOnlyViolations ?
"destructive" : "ghost"}
                    size="sm"
                    onClick={() =>
setShowOnlyViolations(true)}
                    className={`h-7 text-xs px-3
${!showOnlyViolations ? "text-slate-600 hover:text-red-600
hover:bg-red-50" : ""}`}
                >
                    Тільки порушення
                </Button>
            </div>
        </div>

        {balances.map((balance) => {
            // Фільтрація об'єктів
            const filteredObjects =
balance.objects.filter(obj => {
                if (!showOnlyViolations) return true;
                return obj.works.some(w => w.remainingVolume
< 0);
            });

            if (filteredObjects.length === 0) return null;
// Ховаємо таблицю, якщо немає даних під фільтр

            const totalAllowed =
filteredObjects.reduce((sum, obj) => sum +
obj.works.reduce((wSum, w) => wSum + w.allowedVolume, 0),
0);
            const totalUsed = filteredObjects.reduce((sum,
obj) => sum + obj.works.reduce((wSum, w) => wSum +
w.usedVolume, 0), 0);
            const totalRemaining =
filteredObjects.reduce((sum, obj) => sum +
obj.works.reduce((wSum, w) => wSum + w.remainingVolume, 0),
0);
            const hasTotalViolation =
filteredObjects.some(obj => obj.works.some(w =>
w.remainingVolume < 0));

            return (
                <div key={balance.permitId}
className="rounded-lg border border-slate-200 bg-white
shadow-sm overflow-hidden mb-8">
                    <div className="bg-slate-50 px-4 py-3
border-b border-slate-200">
                        <h2 className="font-semibold text-
slate-800">Дозвіл № {balance.permitNumber}</h2>
                    </div>
```

```
                    {/* Контейнер для скролу з фіксованим
заголовком */}
                    <div className="max-h-[600px] overflow-
y-auto relative scrollbar-thin scrollbar-thumb-slate-200
scrollbar-track-transparent">
                        <Table className="border-collapse">
                            <TableHeader className="sticky
top-0 z-10 shadow-sm">
                                <TableRow className="bg-
slate-800 hover:bg-slate-800 border-none">
                                    <TableHead
className="text-slate-100 font-semibold bg-slate-800 text-sm
h-10">Об'єкт / Вид робіт</TableHead>
                                    <TableHead
className="text-slate-100 font-semibold text-right w-[140px]
bg-slate-800 text-sm h-10">Ліміт (м³)</TableHead>
                                    <TableHead
className="text-slate-100 font-semibold text-right w-[140px]
bg-slate-800 text-sm h-10">Використано (м³)</TableHead>
                                    <TableHead
className="text-slate-100 font-semibold text-right w-[140px]
bg-slate-800 text-sm h-10">Залишок (м³)</TableHead>
                                    <TableHead
className="text-slate-100 font-semibold text-right w-[90px]
bg-slate-800 text-sm h-10">Вик. %</TableHead>
                                    <TableHead
className="text-slate-100 font-semibold text-center w-
[110px] bg-slate-800 text-sm h-10">Підсумок</TableHead>
                                </TableRow>
                            </TableHeader>
                            <TableBody>
                                {filteredObjects.map((obj)
=> {
                                    const objAllowed =
obj.works.reduce((sum, w) => sum + w.allowedVolume, 0);
                                    const objUsed =
obj.works.reduce((sum, w) => sum + w.usedVolume, 0);
                                    const objRemaining =
obj.works.reduce((sum, w) => sum + w.remainingVolume, 0);
                                    const hasObjectViolation
= obj.works.some(w => w.remainingVolume < 0);
                                    const isCollapsed =
collapsedObjects[obj.monitoringObjectId];

                                    return (
                                        <React.Fragment
key={obj.monitoringObjectId}>
                                            <TableRow

className="bg-slate-50/50 hover:bg-slate-100/70 border-b
border-slate-200 cursor-pointer transition-colors"
                                                onClick={()
=> toggleCollapse(obj.monitoringObjectId)}
```

```
                                        >
                                            <TableCell
className="font-semibold text-slate-900 py-2.5">
                                                <div
className="flex items-center gap-2">

{isCollapsed ? <ChevronRight size={16} className="text-
slate-400" /> : <ChevronDown size={16} className="text-
slate-400" />}

{obj.objectName}
                                                </div>
                                            </TableCell>
                                            <TableCell
className="text-right font-mono font-semibold text-slate-700
py-2.5 text-sm">{formatVol(objAllowed)}</TableCell>
                                            <TableCell
className="text-right font-mono font-semibold text-slate-700
py-2.5 text-sm">{formatVol(objUsed)}</TableCell>
                                            <TableCell
className={`text-right font-mono font-semibold py-2.5 text-
sm ${hasObjectViolation ? 'text-red-600' : 'text-slate-
700'}`}>{formatVol(objRemaining)}</TableCell>
                                            <TableCell
className={`text-right font-mono font-semibold py-2.5 text-
sm ${hasObjectViolation ? 'text-red-600' : 'text-slate-
700'}`}>

{getPercentageText(objUsed, objAllowed)}
                                            </TableCell>
                                            <TableCell
className="text-center py-2.5">

{renderStatus(hasObjectViolation)}
                                            </TableCell>
                                        </TableRow>

                                        {!isCollapsed &&
obj.works.map((work) => {
                                            const
hasWorkViolation = work.remainingVolume < 0;

                                            // Якщо
увімкнено фільтр, приховуємо нормальні роботи в проблемному
об'єкті
                                            if
(showOnlyViolations && !hasWorkViolation) return null;

                                            return (

<TableRow key={work.workTypeId} className="hover:bg-slate-
50/50 border-b border-slate-100">
```

```
<TableCell className="pl-6 py-2">

<div className="flex items-center relative">

{/* Візуальна лінія ієрархії */}

<div className="absolute left-1.5 top-1/2 -mt-3.5 w-3.5 h-
3.5 border-l-2 border-b-2 border-slate-200 rounded-bl-
sm"></div>

<span className="text-slate-600 text-sm pl-
7">{work.workTypeName}</span>

</div>

</TableCell>

<TableCell className="text-right font-mono text-xs text-
slate-500 py-2">{formatVol(work.allowedVolume)}</TableCell>

<TableCell className="text-right font-mono text-xs text-
slate-500 py-2">{formatVol(work.usedVolume)}</TableCell>

<TableCell className={`text-right font-mono text-xs py-2
${hasWorkViolation ? 'text-red-600 font-semibold' : 'text-
slate-500'}`}>{formatVol(work.remainingVolume)}</TableCell>

<TableCell className={`text-right font-mono text-xs py-2
${hasWorkViolation ? 'text-red-600 font-semibold' : 'text-
slate-500'}`}>

{getPercentageText(work.usedVolume, work.allowedVolume)}


</TableCell>

<TableCell className="text-center py-2">

{renderStatus(hasWorkViolation)}

</TableCell>

</TableRow>
                                                    );
                                                })}
                                            </React.Fragment>
                                        );
                                    })}

                                    {/* Підсумковий рядок,
прилипає до низу */}
```

```
                                    <TableRow className="bg-
    slate-800 hover:bg-slate-800 sticky bottom-0 z-10 border-
    none shadow-[0_-3px_5px_-1px_rgba(0,0,0,0.15)] h-10">
                                        <TableCell
    className="font-semibold text-slate-100 uppercase text-right
    text-xs tracking-wider">Всього по звіту:</TableCell>
                                        <TableCell
    className="text-right font-mono font-bold text-white text-
    sm">{formatVol(totalAllowed)}</TableCell>
                                        <TableCell
    className="text-right font-mono font-bold text-white text-
    sm">{formatVol(totalUsed)}</TableCell>
                                        <TableCell
    className={`text-right font-mono font-bold text-sm
    ${hasTotalViolation ? 'text-red-400' : 'text-emerald-
    400'}`}>{formatVol(totalRemaining)}</TableCell>
                                        <TableCell
    className={`text-right font-mono font-bold text-sm
    ${hasTotalViolation ? 'text-red-400' : 'text-emerald-
    400'}`}>

    {getPercentageText(totalUsed, totalAllowed)}
                                        </TableCell>
                                        <TableCell
    className="text-center">
                                            {hasTotalViolation
                                                ? <span
    className="text-red-300 font-bold uppercase text-xs
    tracking-wider">Порушення</span>
                                                : <span
    className="text-emerald-300 font-bold uppercase text-xs
    tracking-wider">Норма</span>}
                                        </TableCell>
                                    </TableRow>
                                </TableBody>
                            </Table>
                        </div>
                    </div>
                );
            })}
        </div>
    );
};
```