НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Веб-застосунок для віддаленого керування та моніторингу роботи систем очищення води

Виконав студент IV курсу, групи ІП-96
(шифр групи)

Карімов Артем Васильович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викл., Халус О. А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викл., Вітковська І. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент професор кафедри ІСТ, д.т.н., проф., Корнага Я. І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Карімову Артему Васильовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Веб-застосунок для віддаленого керування та моніторингу роботи систем очищення води

керівник проєкту Халус Олена Андріївна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 31 » травня 2023 р. № 2101-с

2. Термін подання студентом проєкту « 19 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: загальні положення, аналіз предметної області, розроблення функціональних вимог, розроблення нефункціональних вимог.
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення, аналіз безпеки даних.
3) Аналіз якості та тестування програмного забезпечення: аналіз якості програмного забезпечення, опис процесів тестування, опис контрольного прикладу.
4) Впровадження та супровід програмного забезпечення: розгортання програмного забезпечення, підтримка програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема бази даних

2) Схема структурна класів програмного забезпечення

3) Схема бізнес-процесу керування системою зворотного осмосу

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» березня 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	17.03.2023	
3	Постановка та формалізація задачі	31.03.2023	
4	Розробка інформаційного забезпечення	14.04.2023	
5	Алгоритмізація задачі	21.04.2023	
6	Обґрунтування вибору використаних технічних засобів	28.04.2023	
7	Розробка програмного забезпечення	05.05.2023	
8	Налагодження програми	08.05.2023	
9	Виконання графічних документів	12.04.2023	
10	Оформлення пояснювальної записки	19.04.2023	
11	Подання ДП на попередній захист	02.06.2023	
12	Подання ДП рецензенту	12.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент _____ Артем КАРІМОВ
(підпис) (ініціали, прізвище)

Керівник _____ Олена ХАЛУС
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 92 таблиці, 34 рисунків та 8 джерел – загалом 105 сторінок.

Дипломний проєкт присвячений полегшенню й прискоренню роботи з системами зворотного осмосу.

Мета: зменшити частоту виїздів сервісних спеціалістів до апаратів очищення води, прискорити швидкість реакції сервісної служби, запобігати виникненню аварійних ситуацій.

Об'єкт дослідження: системи очищення води.

Предмет дослідження: керування системами очищення води.

У першому розділі розглянуто загальні положення, проаналізовано предметну область та засоби розробки програмного забезпечення, розроблено функціональні та нефункціональні вимоги.

У другому розділі змодельовано програмне забезпечення, спроектовано архітектуру програмного забезпечення, проаналізовано безпеку даних.

У третьому розділі проаналізовано якість програмного забезпечення, описано процеси тестування та контрольний приклад.

У четвертому розділі описано процес розгортання та підтримки програмного забезпечення.

Програмне забезпечення впроваджено на платформі Docker Swarm.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, СИСТЕМА ОЧИЩЕННЯ ВОДИ.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 92 tables, 34 figures and 8 sources – in total 105 pages.

The purpose of the diploma project is reducing the frequency of visits by service specialists to water purification devices, speeding up the response rate of the service, preventing the occurrence of alarm situations.

The object of study is water purification system.

The subject of study is control of water purification systems.

In the first chapter the general provisions are considered, the subject area and software development tools are analyzed, functional and non-functional requirements are developed.

In the second chapter the software is modeled, the software architecture is designed, data security is analyzed.

In the third chapter the software quality is analyzed, testing processes and a control example are described.

In the fourth chapter the processes of deploying and maintaining the software are described.

The software is implemented on the platform Docker Swarm.

KEYWORDS: WEB APPLICATION, SOFTWARE, WATER PURIFICATION SYSTEM.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-9619.045440.01.91	Технічне завдання	13	
3	А4	КПІ.ІП-9619.045440.02.81	Пояснювальна записка	105	
4	А4	КПІ.ІП-9619.045440.03.12	Текст програми	71	
5	А4	КПІ.ІП-9619.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-9619.045440.05.34	Керівництво користувача	22	
7	А3	КПІ.ІП-9619.045440.06.99	Схема бази даних	1	
8	А3	КПІ.ІП-9619.045440.06.99	Схема структурна класів	1	
			програмного забезпечення		
11	А3	КПІ.ІП-9619.045440.06.99	Схема бізнес-процесу керування	1	
			системою зворотного осмосу		

					КПІ.ІП-9619.045440.00.90		
Змін.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Карімов А.В.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Халус О.А.					
						КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІП-96	
Н.контр.		Вітковська І.І.					
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ВІДДАЛЕНОГО КЕРУВАННЯ ТА МОНІТОРИНГУ РОБОТИ СИСТЕМ ОЧИЩЕННЯ ВОДИ

Технічне завдання

КПІ.ІП-9619.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Артем КАРІМОВ

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: ВЕБ-ЗАСТОСУНОК ДЛЯ ВІДДАЛЕНОГО КЕРУВАННЯ ТА МОНІТОРИНГУ РОБОТИ СИСТЕМ ОЧИЩЕННЯ ВОДИ

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення веб-застосунку для віддаленого керування та моніторингу роботи систем очищення води [045440], котра використовується для віддаленого керування й моніторингу роботи систем очищення води та призначена для користувачів систем зворотного осмосу.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосунку для віддаленого керування та моніторингу роботи систем очищення води є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для полегшення й прискорення роботи з системами зворотного осмосу.

Метою розробки є зменшення частоти виїздів сервісних спеціалістів до апаратів очищення води, прискорення швидкості реакції сервісної служби, запобігання виникненню аварійних ситуацій.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функцій:

4.1.1 Користувацького інтерфейсу

– кожна сторінка, крім сторінок логіну та скидання пароля, має зручну бічну панель для навігації між сторінками;

– кожна сторінка, крім сторінок логіну та скидання пароля, має хедер, де можна вибрати філію, обрати мову інтерфейсу та перейти на сторінку користувача;

– інформація зі списками відображається у вигляді таблиць, а статистична інформація – у вигляді карток та графіків;

– ввід інформації користувачем відбувається за допомогою форм, які знаходяться в модальних вікнах;

– усі сторінки мають інтерфейс двома мовами:

1) українською
2) англійською;

– усі сторінки мають два інтерфейси:

1) світлий
2) темний.

На рисунку 4.1 зображені прототипи екранних форм. Як можна побачити з рисунку, увійти в застосунок можна, якщо ввести коректні дані на сторінці логіну й натиснути на кнопку «Вхід» або «Sign In», якщо інтерфейс англійською мовою. Далі вся навігація між сторінками відбувається за допомогою бокового меню. Також зі сторінки зі списком апаратів можна потрапити на сторінку з детальним станом апарата, якщо натиснути на рядок у таблиці з інформацією про цей апарат.

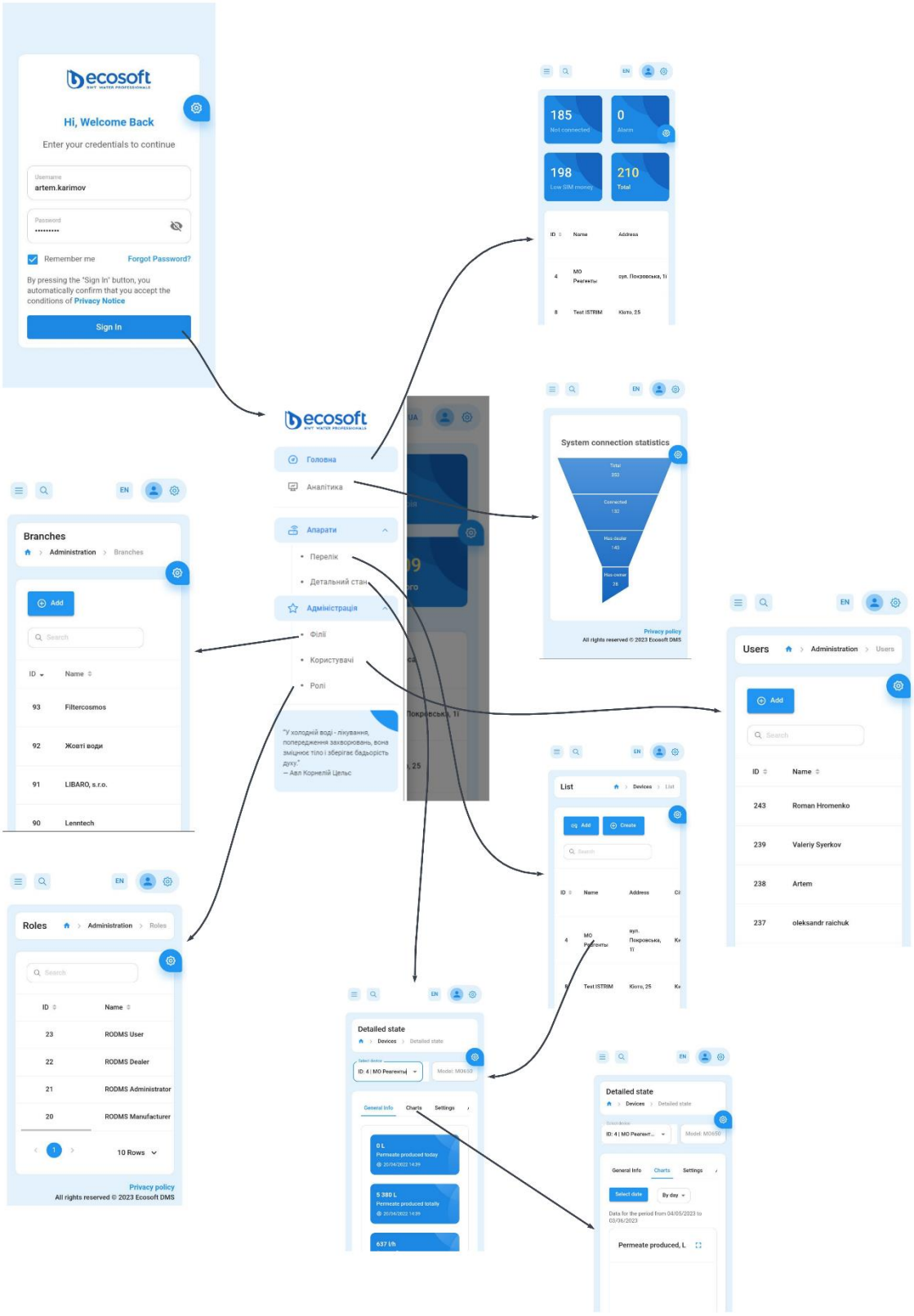


Рисунок 4.1 – Прототипи екранних форм

4.1.2 Для користувача:

– відображення списку апаратів у таблиці;

– відображення детального стану вибраного апарату у вигляді таблиць, графіків, карток;

– зміна персональних даних користувача;

– зміна пароля акаунту користувача.

4.1.3 Для адміністратора системи:

– перегляд дашборду;

– перегляд аналітики;

– відображення списку апаратів у таблиці;

– відображення детального стану вибраного апарату у вигляді таблиць, графіків, карток;

– зміна персональних даних користувача;

– зміна пароля акаунту користувача.

– адміністрування по філіях;

– адміністрування по ролях.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. У разі введення некоректних даних або даних, що не пройшли валідацію, передбачити попередження про це користувача. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинне функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесора: Intel Core i5;
– об’єм оперативної пам’яті: 8 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт;

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 16 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинне працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.), Unix або MacOS.

4.5.1 Вимоги до вхідних даних

Особливі вимоги до вхідних даних не висуваються.

4.5.2 Вимоги до вихідних даних

Вихідні дані на сторінці з історією аварій мають бути представлені у вигляді файлу з розширенням .xlsx, який можна завантажити.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування JavaScript для клієнтької частини та Java для серверної частини.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Visual Studio Code для клієнтької частини та IntelliJ Idea для серверної частини.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді репозиторіїв у базі систем керування версіями Bitbucket.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;

– технічне завдання;

– текст програми;

– програма та методика тестування;

– керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема бази даних;

– схема структурна класів програмного забезпечення;

– схема бізнес-процесу керування системою зворотного осмосу.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки наведені в таблиці 6.1.

Таблиця 6.1 – Стадії та етапи розробки

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: Веб-застосунок для віддаленого керування та моніторингу роботи систем очищення води

КПІ.ІП-9619.045440.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
API	–	Application programming interface, прикладний програмний Інтерфейс
JSON	–	JavaScript Object Notation
JWT	–	JSON Web Token
URL	–	Uniform Resource Locator – єдиний вказівник на ресурс
БД	–	База даних.
VDMS	–	Vending Device Management System – система моніторингу та керування торгівельними апаратами
RODMS	–	Reverse Osmosis Device Management System – система моніторингу та керування апаратами зворотного осмосу
HTTP	–	Hypertext Transfer Protocol – протокол передачі гіпертексту
ПЗ	–	Програмне забезпечення

ВСТУП

Вода – одна з найголовніших речовин, необхідних для життя людини та всього живого. З давніх часів перед людьми стояла задача очищення води. Інструменти, якими це роблять пройшли великий розвиток, і в наші часи є найпотужніші апарати, здатні швидко очистити велику кількість води.

Одним із таких інструментів є зворотний осмос. Апарати зворотного осмосу можуть розташовуватися як у житлових будинках, так і на заправках, заводах тощо. Однак для того, щоб переглянути статистику по такому апарату, необхідно знаходитися біля нього. Веб-застосунок, що буде створено на дипломному проєктованні призначений для віддаленого моніторингу роботи такої системи. Крім того, він дозволить ще й керувати апаратами: створювати, змінювати дані.

Крім того, додаток матиме адмінську панель, де можна буде створювати нових користувачів, нові філії та редагувати їхні значення.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Зворотний осмос – це процес, при якому вода під тиском проходить через спеціальну напівпроникну мембрану, яка затримує різні мікроелементи, розчинені в ній, із ефективністю в 96-99% [1]. Такі можливості дані мембрани мають завдяки порам зі сверхмалими діаметрами, які пропускають виключно водяні молекули й гази. Оскільки системи зворотного осмосу здатні очищати воду від хімічних, органічних та механічних забруднень, вони є універсальними й використовуються не тільки в житлових будинках, а й в апаратах із водою, на заводах, на автомийках тощо.

Для моніторингу роботи таких систем у веб-застосунку можна переглянути наступну статистику:

– кількість очищеної води за сьогодні;
– кількість очищеної води за весь час експлуатації системи;
– середню продуктивність за сьогодні;
– стан системи;
– навність аварій.

1.2 Змістовний опис і аналіз предметної області

Веб-застунок для віддаленого керування та моніторингу роботи систем очищення води – це DMS (device management system, система керування апаратами). Функціонал, який мають DMS застосунки, зазвичай наступний:

– перегляд даних у табличному чи інших виглядах;
– створення користувачів;
– створення апаратів;
– зміна даних апаратів;
– видалення апаратів;
– завантаження звітів;
– перегляд логів;

– налаштування;

– адміністрування.

Може бути велика кількість різних ролей, залежно від яких, користувачі мають доступ до того чи іншого функціоналу додатку. Задачі, які дане програмне забезпечення виконує, – це отримання даних від апаратів очищення води, відображення цих даних користувачеві в різних форматах, а також керування цими системами.

Проблеми, які можуть виникати в таких застосунках, наступні:

– відсутність зв’язку на контролері в апараті в момент, коли потрібно відправити дані;

– нерівномірне навантаження на серверну частину з боку апаратів зворотного осмосу. Вони відправляють пакет даних нерівномірно, а за своєю логікою, наприклад, коли в них відбуваються певні події. Можливі пікові черги, коли, наприклад, 20 апаратів одночасно відправляють дані, а потім нічого не відбувається.

У даному веб-застосунку перша проблема розв’язується тим, що в середині контролера є достатньо пам’яті для зберігання подій. Друга ж проблема вирішується за допомогою асинхронності обробки даних від контролерів та асинхронної відправки даних на контролер. У цьому випадку апарати можуть давати відповідь не одразу, вони можуть виходити на зв’язок тоді, коли це буде потрібно, оскільки вони є ініціаторами зв’язку. Коли апарати передають дані, вони обробляються незалежно від того, чи є з ними зв’язок, а відповідь буде надана за потреби в доступному сеансі зв’язку. Наприклад, апарат має певні дані, ці дані перетворюються в повідомлення, асинхронно відправляються, RODMS сервіс їх вичитує, обробляє й зберігає тоді, коли йому зручно. А коли сервер має дані, які повинні бути відправлені до апарату, він їх відправляє через брокер повідомлень, middleware (проміжне програмне забезпечення) зберігає дані в базу даних і віддає апарату, коли той виходить на зв’язок.

1.3 Аналіз існуючих технологій та успішних ІТ-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації веб-застосунку віддаленого керування та моніторингу роботи систем очищення води. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Найбільш відомими архітектурами програмного забезпечення є:

– монолітна архітектура;

– мікросервісна архітектура;

– подійно-орієнтована архітектура.

Монолітна архітектура – це традиційна модель програмного забезпечення, що побудована як уніфікована одиниця, яка є автономною й незалежною від інших програм [3]. Перевагами такої архітектури є:

– легке розгортання (один виконуваний файл або директорія спрощують розгортання);

– розробка (застосунок розробляти легше, коли його створено з однією кодовою базою);

– легке налагодження (завдяки тому, що весь код знаходиться в одному місці, простіше виконати запити та знайти проблеми).

Мікросервісна архітектура – це модель, яка спирається на серію незалежно розгорнутих сервісів [3]. Ці сервіси мають власну бізнес-логіку та базу даних. Це й відрізняє дану архітектуру від монолітної. Оновлення, тестування, розгортання та масштабування відбуваються в межах кожного сервіса. Перевагами такої архітектури є:

– самостійне розгортання (оскільки мікросервіси – окремі одиниці, це дозволяє швидко, легко й незалежно розгортати окремі функції);

– висока надійність (оскільки розробники можуть розгортати зміни для окремого сервісу без імовірності вивести з ладу всю програму);

– гнучкість у виборі технологій (оскільки кожен сервіс – окрема одиниця, розробники мають свободу у виборі інструментів, які їм потрібні для розробки).

Подійно-орієнтована архітектура використовує події для ініціювання та обміну даними між роз’єднаними сервісами та поширена в сучасних застосунках, створених за допомогою мікросервісів [4]. Перевагами такої архітектури є:

– самостійні масштабування та падіння (оскільки сервіси роз’єднані, вони знають тільки маршрутизатор подій, а одне про одного – ні, тому сервіси є сумісними, але якщо один впаде, інші продовжать працювати);

– швидкість розробки (оскільки маршрутизатор усуває потребу в інтенсивній координації між службами виробника та споживача, прискорюється процес розробки).

У застосунку, що розробляється в рамках дипломного проєктування, використовуватимуться мікросервісна та подійно-орієнтована архітектури. Перша використовуватиметься, тому що застосунку необхідний буде доступ до сервісу з користувачами (user service), який використовує інший застосунок, тому наший сервісу має розгортатися окремо від того вервісу, оскільки зміни можуть нашкодити одразу двом застосункам, а самостійне розгортання і є особливістю мікросервісної архітектури. Друга використовуватиметься, оскільки сервер будуватиметься, використовуючи мікрофреймворк WebFlux, основою якого є подійно-орієнтована архітектура. Цю технологію було обрано була обрана, оскільки вона надасть можливість масштабувати програму з обмеженою, фіксованою кількістю потоків і меншими вимогами до пам’яті, використовуючи доступну обчислювальну потужність.

Найбільш відомими архітектурними патернами є:

– багаторівнева архітектура;

– клієнт-серверна архітектура;
– брокер;
– peer-to-peer (рівний до рівного);
– event bus;
– MVC (model-view-controller, модель-вид-контролер).

У багаторівневій архітектурі компоненти організано в горизонтальні рівні: рівень презентації (містить логіку з тим, як представлені дані), рівень бізнесу (містить бізнес-логіку), рівень стійкості (використовується для таких функцій, як об’єктно-реляційне відображення), рівень бази даних (де зберігаються дані).

У клієнт-серверній архітектурі багато клієнтів (віддалених процесорів) запитують та отримують послуги від централізованого сервера (хост-комп’ютера) [5]. Клієнтські комп’ютери забезпечують інтерфейс, який дозволяє користувачеві комп’ютера запитувати послуги сервера та відображати результати, які повертає сервер. Сервери очікують надходження запитів від клієнтів і потім відповідають на них.

Брокер – архітектурний патерн, що відповідає за комунікацію компонентів. Брокер пересилає запит від клієнта на відповідний сервер і повертає відповідь назад клієнту. Клієнтські та серверні компоненти спілкуються один з одним лише через брокера. Клієнт і сервер не знають одне одного.

Peer-to-peer – архітектурний патерн, в основі якого децентралізована мережа комп’ютерів. Кожен комп’ютер має має однакові функції – є одночасно клієнтом та сервером.

Event bus – архітектурний патерн, за допомогою якого різні компоненти можуть спілкуватися один із одним, не знаючи один про одного. Компонент може надіслати подію в event bus, не знаючи, хто її підбере або скільки інших її підбере. Компоненти також можуть слухати події від event bus, не знаючи, хто надіслав події.

MVC – архітектурний патерн, який розділяє програму на три основні логічні компоненти: модель (уся логіка даних, із якою працює користувач), вид (використовується для всієї логіки інтерфейсу користувача програми) та контролер (обробляє всю бізнес-логіку та вхідні запити, маніпулює даними да допомогою компонента «модель» та взаємодіє з компонентом «вид» для відтворення кінцевого результату).

У застосунку, що розробляється в рамках дипломного проєктування, використовуватимуться наступні патерни:

– багаторівнева архітектура (на рівні бази даних буде власне база даних, на рівні стікості будуть репозиторії, на рівні бізнесу будуть сервіси, на рівні презентації будуть контролери);

– клієнт-серверна архітектура (адже маємо веб-застосунок, де на клієнті дані відображатимуться й можна буде відправити запити на їхнє створення, зміну й видалення на сервер, що їх оброблятиме й даватиме відповідь на них);

– брокер (у застосунку використовуватиметься брокер повідомлень RabbitMQ для комунікації між сервером та апарами зворотного осмосу);

– event bus (його функції також виконуватиме брокер повідомлень RabbitMQ).

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Оскільки програмне забезпечення має клієнт-серверну архітектуру, то необхідно визначитися з технологіями, за допомогою яких розроблятимуться клієнтська та серверна частини застосунку.

Для клієнтської частини використовуються мова гіпертекстової розмітки HTML, мова стилів CSS (каскадні таблиці стилів) та мова програмування JavaScript. Для більш зручної роботи на клієнті використовуються JavaScript бібліотеки та фреймвоки. Найбільш популярними з них на даний момент є бібліотека React і фреймворки Angular та Vue. Кожен із них дуже зручний тим, що для побудови користувацьких

інтерфейсів використовує компоненти – незалежні частини інтерфейсу, придатні для повторного використанні в різних файлах. У бібліотеці React розмітка, стилі та логіка можуть знаходитися в одному файлі, іноді стилі виносяться в окремий файл. Також для бібліотеки створено найбільшу кількість інших бібліотек для роботи з формами, HTTP запитами, стилями тощо. У фреймворку Angular якщо компоненти середні чи великі, то розмітка, стилі та логіка знаходяться в окремих файлах, що може бути незручно. Крім того, для використоння цього фреймворка необіхдні знання мови TypeScript. У фреймворку Vue також розмітка, стилі та логіка знаходяться в одному файлі, однак інших допоміжних бібліотек трохи менше, ніж у бібліотеки React, а також менша спільнота розробників, тому менша ймовірність знайти відповідь на запитання, які виникатимуть під час розробки. Зважаючи на все вище сказане, я обрав бібліотеку React для розробки клієнтської частини веб-застосунку.

Для серверної частини може використовуватися велика кількість мов програмування. Згідно зі щорічним опитуванням платформи Stack Overflow найбільш популярними мовами для цього є JavaScript (фреймворк Express), C# (фреймворк .NET) та Java (фреймворк Spring Boot) [2]. Оскільки JavaScript – не строго типізована мова програмування, на відміну від інших двох, то розробка серверів та виправлення помилок може бути більш важкою та тривалою, тому ця мова не використовуватиметься для написання додатку. І .NET, і Spring Boot широко використовуються для написання серверних частин веб-застосунків, тому мають великі спільноти розробників, що працюють із ними. Але у зв’язку з широким застосуванням платформи Spring Boot для написання сервісів, було обраного саме його для написання серверної частини.

Для клієнтської частини додатків зазвичай використовуються текстовий редактор Visual Studio Code та IDE WebStorm. Для розробки даного додатку використовуватиметься Visual Studio Code, оскільки він має дуже велику кількість різноманітних розширень для полегшення роботи, а також є

безкоштовним, на відміну від WebStorm. Для серверної частини використовуватиметься IntelliJ Idea – найбільш зручний на популярний IDE для мови прогрмування Java.

1.3.3 Аналіз відомих програмних продуктів

Готовим продуктом, який частково чи повністю реалізує функціонал, описаний у технічному завданні, може бути VDMS (Vending Device Management System, система моніторингу та керування торгівельними апаратами) [6]. У цьому застосунку також можна переглядати інформацію про апарати (у даному випадку не про апарати зворотного осмосу, як у RODMS, а про апарати з водою), створювати апарати та користувачів, редагувати дані апаратів та користувачів, видаляти апарати та користувачів, завантажувати технічні звіти. Крім того, можна працювати з інкасаціями, адмініструванням, відправляти запити на оновлення програмного забезпечення апаратів.

Для порівняння курсової роботи з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Веб-застосунок для віддаленого керування та моніторингу роботи систем очищення води	Система моніторингу та керування торгівельними апаратами	Пояснення

Продовження таблиці 1.1

Авторизація зареєстрованих користувачів	+	+	Застосунок має авторизувати користувача й переправити його на головну сторінку, якщо були введені коректні дані
Відображення списку апаратів доступних філій	+	+	Застосунок має відображати дані апаратів доступних користувачеві філій у табличному вигляді
Відображення детального стану конкретного апарата	+	+	Застосунок має відображати детальний стан вибраного користувачем апарата у вигляді графіків, таблиць тощо
Створення нового апарата	+	+	Користувач може створити апарат, якщо він увів дані у форму, і вони пройшли валідацію

Продовження таблиці 1.1

Редагування апарата	+	+	Користувач може відредагувати апарат, змінивши дані й пройшовши валідацію значень у формі
Пошук апарата за ключовими словами	+	+	Користувач має змогу знайти апарати за ключовими словами на сторінці зі списком апаратів
Створення користувачів	+	+	Користувач може створювати інших користувачів, увівши коректні дані у форму
Створення філій	+	+	Користувач може створювати філії, увівши коректні дані у форму

Продовження таблиці 1.1

Створення ролей	–	+	Користувач може створювати ролі, вибравши всі дозволи (на перегляд, створення, редагування, видалення, завантаження), які користувачі з цими ролями матимуть
Завантаження звітів	+	+	Користувач може завантажувати звіти у форматі .xlsx
Редагування персональних даних	+	+	Користувач може відредагувати персональні дані в налаштуваннях акаунта
Зміна пароля	+	+	Користувач може змінити пароль у налаштуваннях акаунта
Скидання пароля	+	+	Користувач може скинути пароль на сторінці авторизації

1.4 Аналіз вимог до програмного забезпечення

Головною функцією програмного забезпечення є моніторинг систем очищення води. Більше функцій можна побачити на рисунках 1.1, 1.2, 1.3 та 1.4.

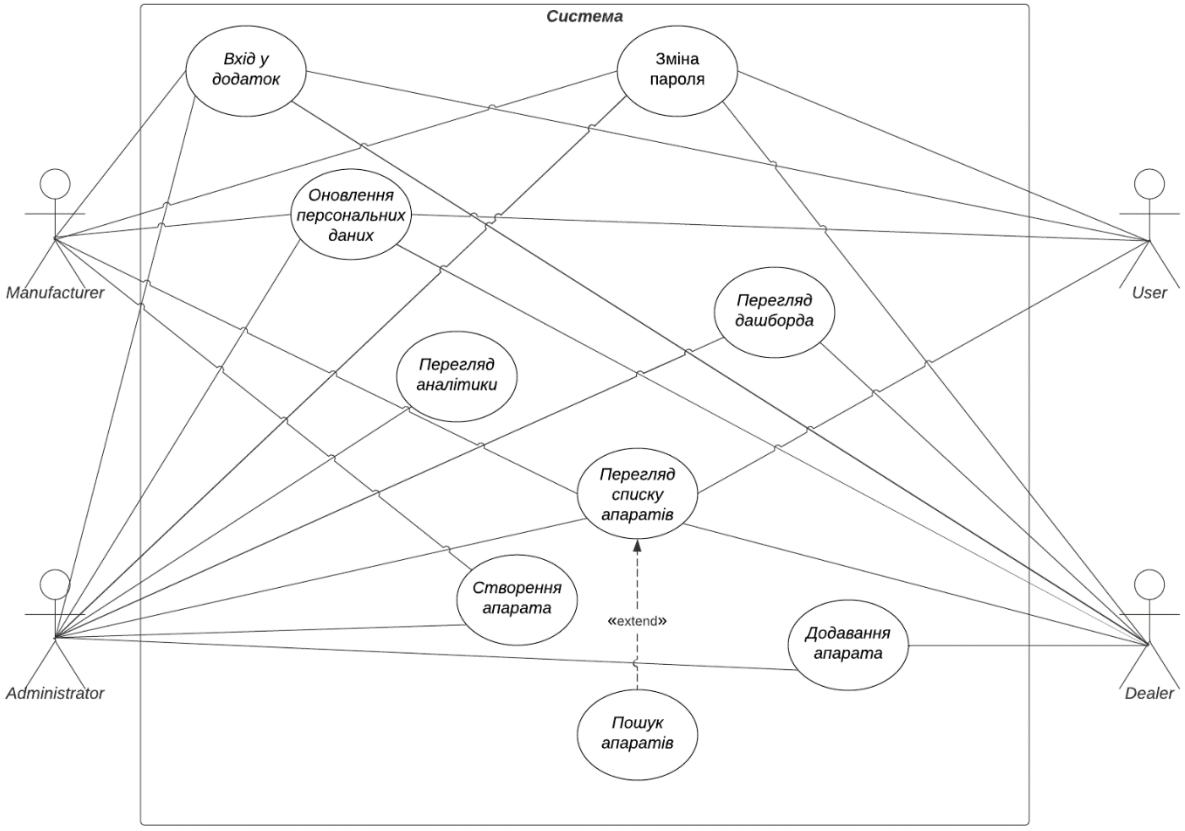


Рисунок 1.1 – Схема структурна варіантів використання

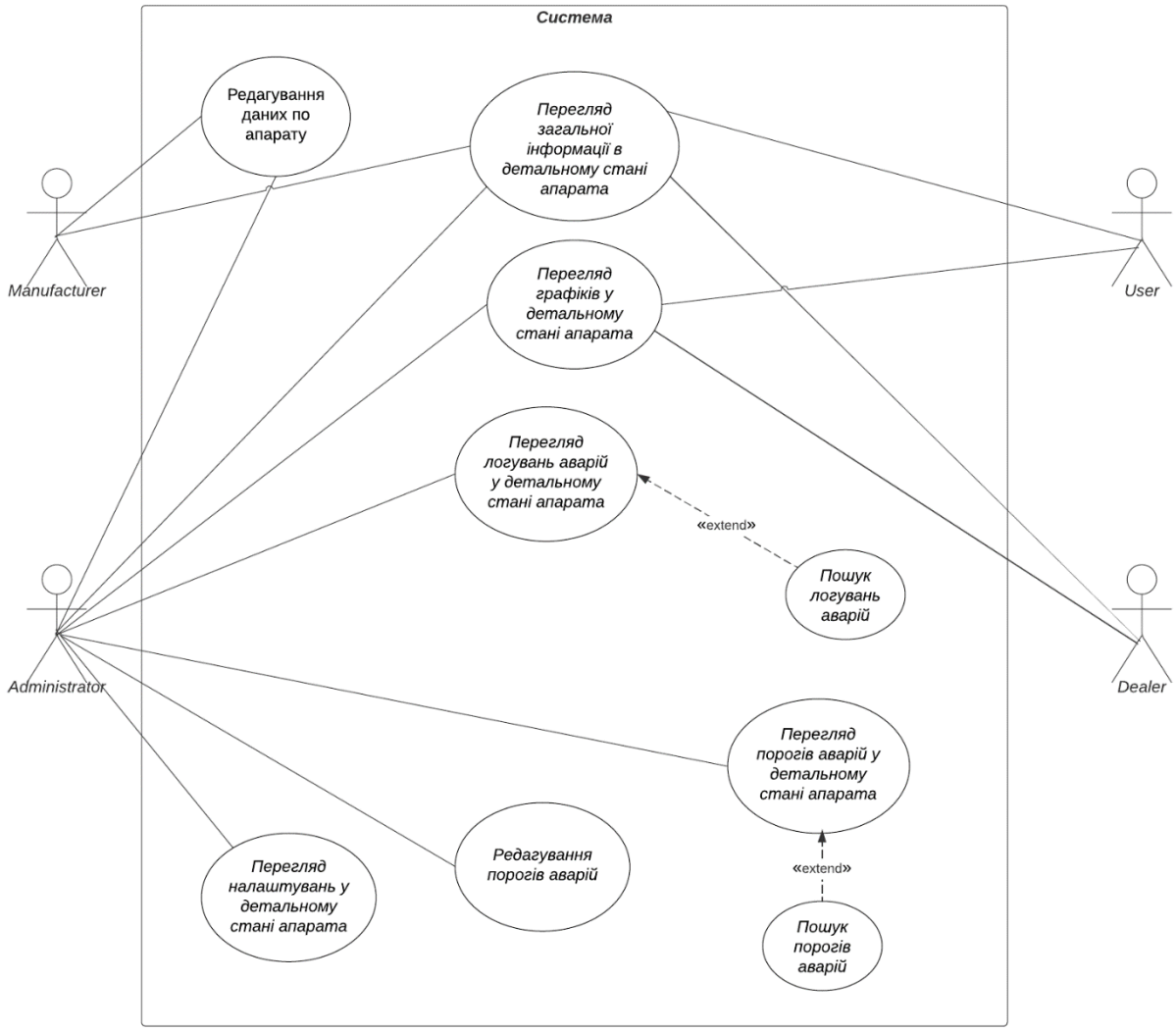


Рисунок 1.2 – Схема структурна варіантів використання

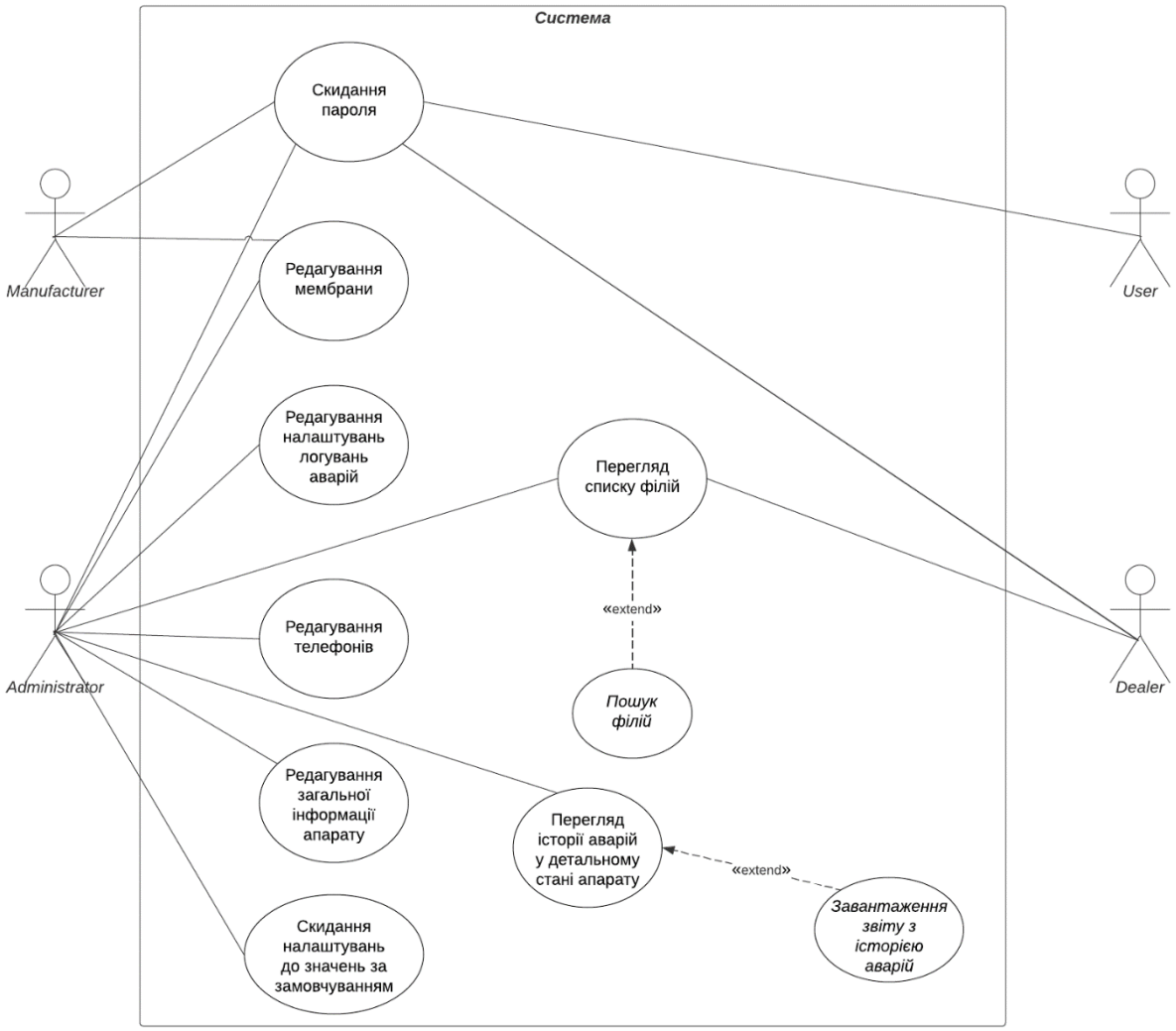


Рисунок 1.3 – Схема структурна варіантів використання

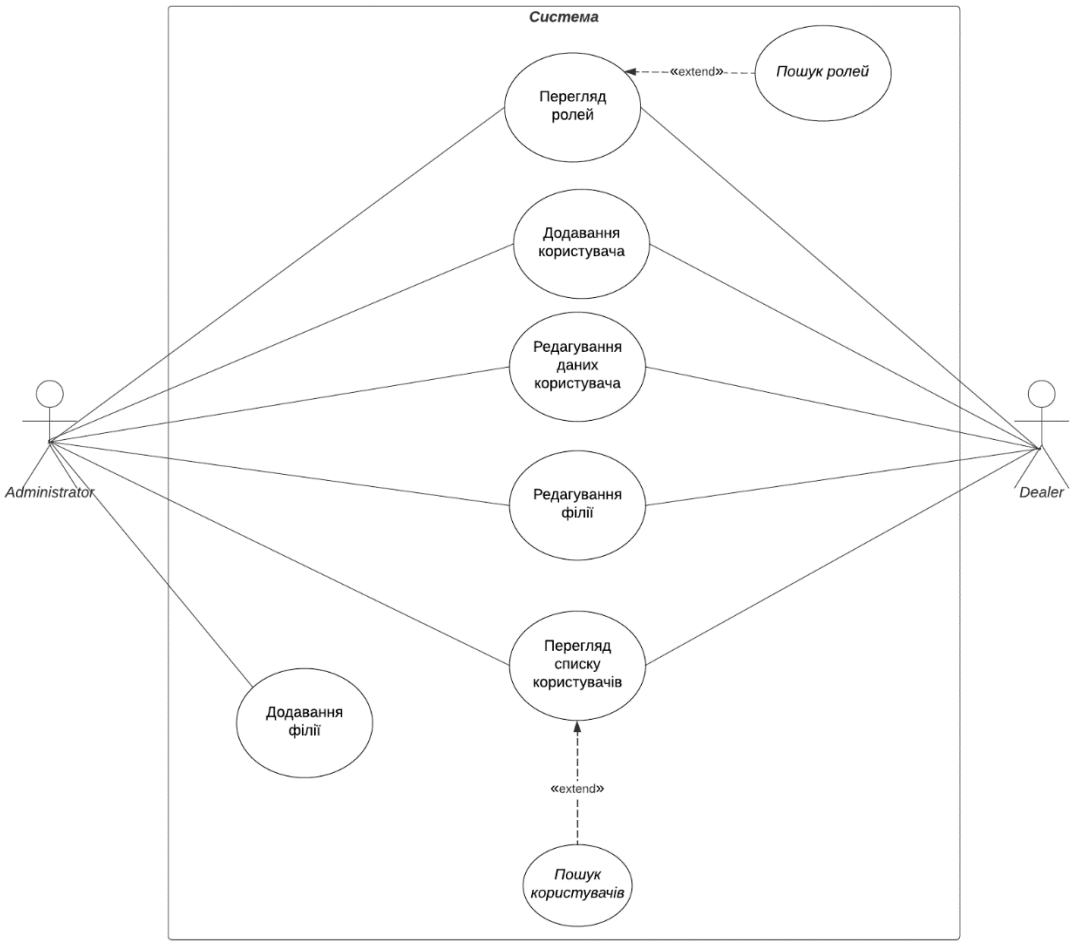


Рисунок 1.4 – Схема структурна варіантів використання

В таблицях 1.2 – 1.37 наведені варіанти використання програмного забезпечення.

Таблиця 1.2 – Варіант використання UC-01

Use case name	Вхід у додаток
Use case ID	UC-01
Goals	Авторизація користувача в системі
Actors	Administrator, Manufacturer, Deader, User
Trigger	Користувач бажає авторизуватися в системі
Pre-conditions	-

Продовження таблиці 1.2

Flow of Events	Користувач переходить на сторінку логіну. В поля форми вводяться ім’я користувача та пароль. Після заповнення даних, користувач натискає кнопку «Вхід». Якщо були введені коректні дані, користувач переходить на головну сторінку додатку.
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні.
Post-Condition	Користувача авторизовано, додаток отримав список дозволів, що має користувач

Таблиця 1.3 – Варіант використання UC-02

Use case name	Зміна пароля
Use case ID	UC-02
Goals	Зміна пароля для входу в систему
Actors	Administrator, Manufacturer, Deader, User
Trigger	Користувач бажає змінити пароль
Pre-conditions	Користувача було авторизовано, і він має доступ доступ до налаштувань акаунту
Flow of Events	Користувач переходить на сторінку з налаштуваннями акаунту. Далі переходить на вкладку «Зміна Пароля», у форму вводить актуальний пароль, новий пароль та підтвердження нового пароля, натискає на кнопку «Зміна Пароля». Якщо був уведений правильний поточний пароль та всі введені дані пройшли валідацію, користувач отримує повідомлення про те, що пароль було успішно змінено.
Extension	У випадку введення неправильного поточного пароля, користувач отримує повідомлення про це.
Post-Condition	Пароль користувача для входу в систему змінено.

Таблиця 1.4 – Варіант використання UC-03

Use case name	Оновлення персональних даних
Use case ID	UC-03
Goals	Оновлення персональних даних зареєстрованого користувача
Actors	Administrator, Manufacturer, Deader, User
Trigger	Користувач бажає оновити персональні дані
Pre-conditions	Користувача було авторизовано, і він має доступ доступ до налаштувань акаунту
Flow of Events	Користувач переходить на сторінку з налаштуваннями акаунту. Змінює персональну та/або контактну інформацію, натискає на кнопку «Оновити профіль». Якщо введені дані пройшли валідацію, користувач отримує повідомлення про те, що персональні дані було успішно змінено.
Extension	-
Post-Condition	Персональні дані користувача оновлено

Таблиця 1.5 – Варіант використання UC-04

Use case name	Перегляд дашборда
Use case ID	UC-04
Goals	Перегляд дашборда зі списком апаратів та статистикою
Actors	Administrator, Deader
Trigger	Користувач бажає переглянути дашборд зі списком апаратів та/або статистичні дані
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду дашборда
Flow of Events	Користувач переходить на головну сторінку. Йому відображаються дані про апарати в табличному вигляді та статистичні дані у вигляді карток
Extension	У випадку якщо поточна філія не має апаратів, користувач отримує про це повідомлення
Post-Condition	Користувач може переглянути дашборд та статистичні дані

Таблиця 1.6 – Варіант використання UC-05

Use case name	Перегляд аналітики
Use case ID	UC-05
Goals	Перегляд статистики підключення систем
Actors	Administrator
Trigger	Користувач бажає переглянути статистику підключення систем
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду перегляду аналітики
Flow of Events	Користувач переходить на головну з аналітикою. Йому відображається статика підключення систем у вигляді діаграми
Extension	-
Post-Condition	Користувача може переглянути статистику підключення систем

Таблиця 1.7 – Варіант використання UC-06

Use case name	Перегляд списку апаратів
Use case ID	UC-06
Goals	Перегляд список апаратів зворотного осмосу у доступних філіях
Actors	Administrator, Manufacturer, Deader, User
Trigger	Користувач бажає переглянути список апаратів зворотного осмосу у доступних йому філіях
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду апаратів
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Перелік». Йому відображається перелік апаратів поточної філії у вигляді таблиці. Змінити філію можна в хедері, якщо користувач має дозвіл на це
Extension	У випадку якщо поточна філія не має апаратів, користувач отримує про це повідомлення

Post-Condition	Користувач може переглянути список апаратів поточної філії та інших доступних йому філій

Таблиця 1.8 – Варіант використання UC-07

Use case name	Створення апарата
Use case ID	UC-07
Goals	Створрення нового апарата
Actors	Administrator, Manufacturer
Trigger	Користувач бажає створити новий апарат
Pre-conditions	Користувача було авторизовано, і він має доступ до створення апаратів
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Перелік». Йому відображається перелік апаратів поточної філії у вигляді таблиці. Над таблицею користувач натискає кнопку «Створити». З'являється модальне вікно з формою. Користувач заповнює форму. Якщо введені дані пройшли валідацію, користувач отримує повідомлення про те, що апарат було створено. Після цього модальне вікно закривається
Extension	Якщо дані не пройшли валідацію, користувач отримує повідомлення про те, у яких саме полях некоректні дані
Post-Condition	Користувач створив новий апарат

Таблиця 1.9 – Варіант використання UC-08

Use case name	Додавання апарата
Use case ID	UC-08
Goals	Додавання апарата до філії
Actors	Administrator, Deader
Trigger	Користувач бажає додати створений апарат до доступних йому філій

Продовження таблиці 1.9

Pre-conditions	Користувача було авторизовано, і він має доступ до додавання апаратів
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Перелік». Йому відображається перелік апаратів поточної філії у вигляді таблиці. Над таблицею користувач натискає кнопку «Додати». З'являється модальне вікно з формою. Користувач заповнює форму. Якщо введені дані пройшли валідацію, користувач отримує повідомлення про те, що апарат було додано. Після цього модальне вікно закривається
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні
Post-Condition	Користувач додав новий апарат до вибраної філії

Таблиця 1.10 – Варіант використання UC-09

Use case name	Пошук апаратів
Use case ID	UC-09
Goals	Пошук апаратів за ключовим словом
Actors	Administrator, Manufacturer, Deader, User
Trigger	Користувач бажає знайти апарати за ключовим словом
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду апаратів
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Перелік». Йому відображається перелік апаратів поточної філії у вигляді таблиці. Над таблицею є поле для пошуку. Користувач вводить туди ключові слова, і на кожне введення літери отримує список апаратів, які відповідають тому, що було введено
Extension	У випадку, якщо апаратів за пошуком не знайшлося, користувачу нічого не відображається
Post-Condition	Користувач знайшов або не знайшов апарати за своїм пошуком

Таблиця 1.11 – Варіант використання UC-10

Use case name	Редагування даних по апарату
Use case ID	UC-10
Goals	Редагування даних по обраному апарату
Actors	Administrator, Manufacturer
Trigger	Користувач бажає відредагувати дані по обраному апарату
Pre-conditions	Користувача було авторизовано, і він має доступ до редагування апаратів
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Перелік». Йому відображається перелік апаратів поточної філії у вигляді таблиці. У таблиці є колонка «Дія». При натисканні на рядок цієї колонки, відкривається модальне вікно з даними апарата для редагування. Користувач змінює дані. Якщо змінені дані пройшли валідацію, користувач отримує повідомлення про те, що дані про апарат було відредаговано. Після цього модальне вікно закривається
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні
Post-Condition	Користувач відредагував дані про обраний апарат

Таблиця 1.12 – Варіант використання UC-11

Use case name	Перегляд загальної інформації в детальному стані апарата
Use case ID	UC-11
Goals	Перегляд загальної інформації про апарат
Actors	Administrator, Manufacturer, Deader, User
Trigger	Користувач бажає переглянути загальну інформацію про апарат
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду загальної інформації в детальному стані апарата

Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається загальна інформація про цей апарат
Extension	-
Post-Condition	Користувач може переглянути загальну інформацію про апарат

Таблиця 1.13 – Варіант використання UC-12

Use case name	Перегляд графіків у детальному стані апарата
Use case ID	UC-12
Goals	Перегляд графіків, що відображають дані про апарат
Actors	Administrator, Deader, User
Trigger	Користувач бажає переглянути графіки, що відображають дані про апарат
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду графіків у детальному стані апарата
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Графіки». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображаються 6 графіків, які відображають дані про апарат
Extension	У випадку, якщо даних для графіків немає, користувач отримує повідомлення про це
Post-Condition	Користувача може переглянути графіки, які відображають дані про апарат

Таблиця 1.14 – Варіант використання UC-13

Use case name	Перегляд налаштувань у детальному стані апарата
Use case ID	UC-13
Goals	Перегляд налаштувань обраного апарата

Actors	Administrator
Trigger	Користувач бажає переглянути налаштування обраного ним апарата
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду налаштувань у детальному стані апарата
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Налаштування». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображаються доступні налаштування
Extension	У випадку, якщо доступних налаштувань немає, користувач отримує повідомлення про це
Post-Condition	Користувач може переглянути налаштування обраного ним апарата

Таблиця 1.15 – Варіант використання UC-14

Use case name	Перегляд логувань аварій у детальному стані апарата
Use case ID	UC-14
Goals	Перегляд логувань аварій обраного апарата
Actors	Administrator
Trigger	Користувач бажає переглянути логування аварій обраного ним апарата
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду логувань аварій у детальному стані апарата
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Логування Аварій». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається таблиця з аваріями
Extension	-
Post-Condition	Користувач може переглянути логування аварій обраного ним апарата

Таблиця 1.16 – Варіант використання UC-15

Use case name	Пошук логувань аварій
Use case ID	UC-15
Goals	Пошук аварій за ключовими словами
Actors	Administrator
Trigger	Користувач бажає знайти аварії за ключовими словами
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду логувань аварій у детальному стані апарата
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Логування Аварій». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається таблиця з аваріями. Над таблицею є поле для пошуку. Користувач вводить туди ключові слова, і на кожне введення літери отримує список аварій, які відповідають тому, що було введено
Extension	У випадку, якщо аварій за пошуком не знайшлося, користувачу нічого не відображається
Post-Condition	Користувач знайшов або не знайшов аварії за своїм пошуком

Таблиця 1.17 – Варіант використання UC-16

Use case name	Перегляд порогів аварій у детальному стані апарата
Use case ID	UC-16
Goals	Перегляд порогів аварій обраного апарата
Actors	Administrator
Trigger	Користувач бажає переглянути пороги аварій обраного ним апарата
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду порогів аварій у детальному стані апарата

Продовження таблиці 1.17

Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Пороги Аварій». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається таблиця з порогами аварій
Extension	-
Post-Condition	Користувач може переглянути пороги аварій обраного ним апарата

Таблиця 1.18 – Варіант використання UC-17

Use case name	Пошук порогів аварій
Use case ID	UC-17
Goals	Пошук порогів аварій за ключовими словами
Actors	Administrator
Trigger	Користувач бажає знайти пороги аварій за ключовими словами
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду порогів аварій у детальному стані апарата
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Пороги Аварій». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається таблиця з порогами аварій. Над таблицею є поле для пошуку. Користувач вводить туди ключові слова, і на кожне введення літери отримує список порогів аварій, які відповідають тому, що було введено
Extension	У випадку, якщо порогів аварій за пошуком не знайшлося, користувачу нічого не відображається
Post-Condition	Користувач знайшов або не знайшов пороги аварій за своїм пошуком

Таблиця 1.19 – Варіант використання UC-18

Use case name	Редагування порогів аварій
Use case ID	UC-18
Goals	Редагування значення вибраного порогу аварії
Actors	Administrator
Trigger	Користувач бажає відредагувати значення вибраного ним порогу аварії
Pre-conditions	Користувача було авторизовано, і він має доступ до редагування порогів аварій у детальному стані апарата
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Пороги Аварій». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається таблиця з порогами аварій. У таблиці є колонка «Дія». При натисканні на рядок цієї колонки, відкривається модальне вікно з даними порогу аварії для редагування. Користувач змінює дані. Якщо змінені дані пройшли валідацію, користувач отримує повідомлення про те, що дані про поріг аварії було відредаговано. Після цього модальне вікно закривається
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні.
Post-Condition	Користувача відредагував дані про обраний поріг аварії

Таблиця 1.20 – Варіант використання UC-19

Use case name	Скидання пароля
Use case ID	UC-19
Goals	Скидання поточного пароля
Actors	Administrator, Manufacturer, Deader, User
Trigger	Користувач бажає скинути свій поточний пароль
Pre-conditions	Користувач не може увійни в систему через те, що забув свій поточний пароль

Продовження таблиці 1.20

Flow of Events	Користувач переходить на сторінку логіну. Натискає на «Забули пароль?» Відбувається перехід на сторнку з формою, куди потрібно ввести електронну пошту, на яку прийде код, який потрібно буде використати для скидання. Далі з’являється форма для введення коду та нового пароля. Якщо введені дані пройшли валідацію, то користувач отримає повідомлення про те, що пароль було успішно змінено, і його перенаправляє застосунок на сторінку з логіном
Extension	У випадку введення електронної пошти, що незареєстрована в додатку, користувач отримує повідомлення, що такого користувача не знайдено. У випадку введення неправильного коду, користувач отримує про це повідомлення
Post-Condition	Пароль користувача скинуто

Таблиця 1.21 – Варіант використання UC-20

Use case name	Перегляд історії аварій у детальному стані апарата
Use case ID	UC-20
Goals	Перегляд історії аварій вибраного апарата
Actors	Administrator
Trigger	Користувач бажає переглянути сторію аварій вибраного ним апарата
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду історії аварій у детальному стані апарата
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Історія Аварій». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається історія аварій
Extension	У випадку, якщо аварій в аварата немає, користувач отримує про це повідомлення
Post-Condition	Користувач може переглянути історію аварій обраного ним апарата

Таблиця 1.22 – Варіант використання UC-21

Use case name	Завантаження звіту з історією аварій
Use case ID	UC-21
Goals	Завантаження звіту з історією аварій обраного апарату
Actors	Administrator
Trigger	Користувач бажає завантажити звіт із історією аварій обраного ним апарату
Pre-conditions	Користувача було авторизовано, і він має доступ до перегляду історії аварій у детальному стані апарата
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Історія Аварій». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається історія аварій. Натиснувши на кнопку «Завантажити», користувач може завантажити ці дані у форматі .xlsx
Extension	-
Post-Condition	Користувач завантажив звіт із історією аварій обраного ним апарату

Таблиця 1.23 – Варіант використання UC-22

Use case name	Редагування загальної інформації апарату
Use case ID	UC-22
Goals	Редагування загальної інформації обраного апарату
Actors	Administrator
Trigger	Користувач бажає відредагувати загальну інформацію обраного ним апарату
Pre-conditions	Користувача було авторизовано, і він має доступ до редагування загальної інформації апарата

Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається загальна інформація про цей апарат. Після того, як користувач натиснув на кпопку «Редагувати» у картці з інформацією про апарат, відкривається модальне вікно з даними апарата для редагування. Користувач змінює дані. Якщо змінені дані пройшли валідацію, користувач отримує повідомлення про те, що дані про апарат було відредаговано. Після цього модальне вікно закривається
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні
Post-Condition	Користувач відредагував дані про обраний апарат

Таблиця 1.24 – Варіант використання UC-23

Use case name	Редагування телефонів
Use case ID	UC-23
Goals	Редагування телефонів обраного апарату
Actors	Administrator
Trigger	Користувач бажає відредагувати телефони обраного ним апарату
Pre-conditions	Користувача було авторизовано, і він має доступ до редагування телефонів апарата

Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається загальна інформація про обраний апарат. Після того, як користувач натиснув на кпопку «Редагувати» у картці з телефонами апарату (телефони системи, сервісного центру, SMS), відкривається модальне вікно з телефонними даними апарата для редагування. Користувач змінює дані. Якщо змінені дані пройшли валідацію, користувач отримує повідомлення про те, що телефони було відредаговано. Після цього модальне вікно закривається
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні
Post-Condition	Користувач відредагував телефони обраного апарата

Таблиця 1.25 – Варіант використання UC-24

Use case name	Редагування мембрани
Use case ID	UC-24
Goals	Редагування мембрани обраного апарату
Actors	Administrator
Trigger	Користувач бажає відредагувати мембрану обраного ним апарату
Pre-conditions	Користувача було авторизовано, і він має доступ до редагування мембрани апарата

Продовження таблиці 1.25

Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається загальна інформація про цей апарат. Після того, як користувач натиснув на кпопку «Редагувати» у картці з інформацією про мебрану, відкривається модальне вікно з даними мембрани для редагування. Користувач змінює дані. Якщо змінені дані пройшли валідацію, користувач отримує повідомлення про те, що дані про мембрану було відредаговано. Після цього модальне вікно закривається
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні
Post-Condition	Користувач відредагував дані про мембрану обраного апарата

Таблиця 1.26 – Варіант використання UC-25

Use case name	Скидання налаштувань до значень за замовчуванням
Use case ID	UC-25
Goals	Скидання налаштувань апарата до значень за замовчуванням
Actors	Administrator
Trigger	Користувач бажає скинути налаштування обраного ним апарата до значень за замочуванням
Pre-conditions	Користувача було авторизовано, і він має доступ до скидання налаштувань до значень за замовчуванням
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Пороги Аварій». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається таблиця з порогами аварій. Натиснувши на кнопку «Дефолтні Налаштування» та підтвердивши свою дію, користувач може скинути налаштування до значень за замовчуванням
Extension	-
Post-Condition	Користувач скинув налаштування до значень за замовчуванням

Таблиця 1.27 – Варіант використання UC-26

Use case name	Редагування налаштувань логувань аварій
Use case ID	UC-26
Goals	Редагування налаштувань логувань аварій обраного апарата
Actors	Administrator
Trigger	Користувач бажає відредагувати налаштування логувань аварій обраного ним апарата
Pre-conditions	Користувача було авторизовано, і він має доступ до редагування налаштувань логувань аварій
Flow of Events	Користувач натискає на вкладку «Апарати» в боковому меню й вибирає дочірню вкладку «Детальний стан», переходить на сторінці на вкладку «Логування Аварій». Далі обирає апарат зі списку апаратів, які йому доступні, і йому відображається таблиця з аваріями. У таблиці є колонка «Увімкнено». Змінюючи значення в чекбоксі, користувач змінює налаштування для аварії в конкретному рядку
Extension	-
Post-Condition	Користувач відредагував налаштування логувань аварій обраного ним апарата

Таблиця 1.28 – Варіант використання UC-27

Use case name	Перегляд списку філій
Use case ID	UC-27
Goals	Перегляд списку філій системи
Actors	Administrator, Deader
Trigger	Користувач бажає переглянути список філій системи
Pre-conditions	Користувача було авторизовано, і він має доступ на перегляд списку філій
Flow of Events	Користувач натискає на вкладку «Адміністрація» в боковому меню й вибирає дочірню вкладку «Філії». Йому відображається список філій у вигляді таблиці
Extension	-

Post-Condition	Користувач може переглянути список філій системи

Таблиця 1.29 – Варіант використання UC-28

Use case name	Пошук філій
Use case ID	UC-28
Goals	Пошук філій за ключовими словами
Actors	Administrator, Deader
Trigger	Користувач бажає знайти філії за ключовими словами
Pre-conditions	Користувача було авторизовано, і він має доступ на перегляд списку філій
Flow of Events	Користувач натискає на вкладку «Адміністрація» в боковому меню й вибирає дочірню вкладку «Філії». Йому відображається список філій у вигляді таблиці. Над таблицею є поле для пошуку. Користувач вводить туди ключові слова, і на кожне введення літери отримує список філій, які відповідають тому, що було введено
Extension	У випадку, якщо філій за пошуком не знайшлося, користувачу нічого не відображається
Post-Condition	Користувач знайшов або не знайшов філії за своїм пошуком

Таблиця 1.30 – Варіант використання UC-29

Use case name	Додавання філії
Use case ID	UC-29
Goals	Додавання нової філії
Actors	Administrator
Trigger	Користувач бажає додати нову філію
Pre-conditions	Користувача було авторизовано, і він має доступ на додавання філії

Продовження таблиці 1.30

Flow of Events	Користувач натискає на вкладку «Адміністрація» в боковому меню й вибирає дочірню вкладку «Філії». Йому відображається список філій у вигляді таблиці. Над таблицею є кнопка «Додати». Після того, як користувач її натиснув, відкривається модальне вікно з формою. Якщо введені користувачем дані пройшли валідацію, користувач отримує повідомлення про те, що філію було додано. Після цього модальне вікно закривається
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні
Post-Condition	Користувач створив нову філію

Таблиця 1.31 – Варіант використання UC-30

Use case name	Редагування філії
Use case ID	UC-30
Goals	Редагування філії системи
Actors	Administrator, Deader
Trigger	Користувач бажає відредагувати дані вибраної ним філії
Pre-conditions	Користувача було авторизовано, і він має доступ на редагування філії
Flow of Events	Користувач натискає на вкладку «Адміністрація» в боковому меню й вибирає дочірню вкладку «Філії». Йому відображається список філій у вигляді таблиці. У таблиці є колонка «Дія». При натисканні на рядок цієї колонки, відкривається модальне вікно з даними філії для редагування. Користувач змінює дані. Якщо змінені дані пройшли валідацію, користувач отримує повідомлення про те, що дані про філію було відредаговано. Після цього модальне вікно закривається
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні
Post-Condition	Користувач відредагував дані по філії

Таблиця 1.32 – Варіант використання UC-31

Use case name	Перегляд списку користувачів
Use case ID	UC-31
Goals	Перегляд списку користувачів системи
Actors	Administrator, Deader
Trigger	Користувач бажає переглянути список користувачів системи
Pre-conditions	Користувача було авторизовано, і він має доступ на перегляд списку користувачів
Flow of Events	Користувач натискає на вкладку «Адміністрація» в боковому меню й вибирає дочірню вкладку «Користувачі». Йому відображається список користувачів у вигляді таблиці
Extension	-
Post-Condition	Користувач може переглянути список користувачів системи

Таблиця 1.33 – Варіант використання UC-32

Use case name	Пошук користувачів
Use case ID	UC-32
Goals	Пошук користувачів за ключовими словами
Actors	Administrator, Deader
Trigger	Користувач бажає знайти користувачів за ключовими словами
Pre-conditions	Користувача було авторизовано, і він має доступ на перегляд списку користувачів
Flow of Events	Користувач натискає на вкладку «Адміністрація» в боковому меню й вибирає дочірню вкладку «Користувачі». Йому відображається список користувачів у вигляді таблиці. Над таблицею є поле для пошуку. Користувач вводить туди ключові слова, і на кожне введення літери отримує список користувачів, які відповідають тому, що було введено
Extension	У випадку, якщо користувачів за пошуком не знайшлося, користувачу нічого не відображається

Post-Condition	Користувач знайшов або не знайшов користувачів системи за своїм пошуком

Таблиця 1.34 – Варіант використання UC-33

Use case name	Додавання користувача
Use case ID	UC-33
Goals	Додавання нового користувача системи
Actors	Administrator, Deader
Trigger	Користувач бажає додати нового користувача системи
Pre-conditions	Користувача було авторизовано, і він має доступ на додавання користувача
Flow of Events	Користувач натискає на вкладку «Адміністрація» в боковому меню й вибирає дочірню вкладку «Користувачі». Йому відображається список користувачів у вигляді таблиці. Над таблицею є кнопка «Додати». Після того, як користувач її натиснув, відкривається модальне вікно з формою. Якщо введені користувачем дані пройшли валідацію, користувач отримує повідомлення про те, що користувача було додано. Після цього модальне вікно закривається
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні
Post-Condition	Користувач створив нового користувача системи

Таблиця 1.35 – Варіант використання UC-34

Use case name	Редагування даних користувача
Use case ID	UC-34
Goals	Редагування даних користувача системи
Actors	Administrator, Deader
Trigger	Користувач бажає відредагувати дані вибраного ним користувача системи

Продовження таблиці 1.35

Pre-conditions	Користувача було авторизовано, і він має доступ на редагування користувача
Flow of Events	Користувач натискає на вкладку «Адміністрація» в боковому меню й вибирає дочірню вкладку «Користувачі». Йому відображається список користувачів у вигляді таблиці. У таблиці є колонка «Дія». При натисканні на рядок цієї колонки, відкривається модальне вікно з даними користувача системи для редагування. Користувач змінює дані. Якщо змінені дані пройшли валідацію, користувач отримує повідомлення про те, що дані про користувача було відредаговано. Після цього модальне вікно закривається
Extension	У випадку введення некоректних даних, користувач отримує повідомлення про те, у якому саме полі дані не правильні
Post-Condition	Користувач відредагував дані по користувачу системи

Таблиця 1.36 – Варіант використання UC-35

Use case name	Перегляд ролей
Use case ID	UC-35
Goals	Перегляд списку ролей системи
Actors	Administrator, Deader
Trigger	Користувач бажає переглянути список ролей системи
Pre-conditions	Користувача було авторизовано, і він має доступ на перегляд списку ролей
Flow of Events	Користувач натискає на вкладку «Адміністрація» в боковому меню й вибирає дочірню вкладку «Ролі». Йому відображається список ролей у вигляді таблиці
Extension	-
Post-Condition	Користувач може переглянути список ролей системи

Таблиця 1.37 – Варіант використання UC-36

Use case name	Пошук ролей
Use case ID	UC-36
Goals	Пошук ролей за ключовими словами
Actors	Administrator, Deader
Trigger	Користувач бажає знайти ролі за ключовими словами
Pre-conditions	Користувача було авторизовано, і він має доступ на перегляд списку ролей
Flow of Events	Користувач натискає на вкладку «Адміністрація» в боковому меню й вибирає дочірню вкладку «Ролі». Йому відображається список ролей у вигляді таблиці. Над таблицею є поле для пошуку. Користувач вводить туди ключові слова, і на кожне введення літери отримує список ролей, які відповідають тому, що було введено
Extension	У випадку, якщо ролей за пошуком не знайшлося, користувачу нічого не відображається
Post-Condition	Користувач знайшов або не знайшов ролі системи за своїм пошуком

1.4.1 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. На рисунку 1.5 наведено загальну модель вимог, а в таблицях 1.38 – 1.52 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 1.6.

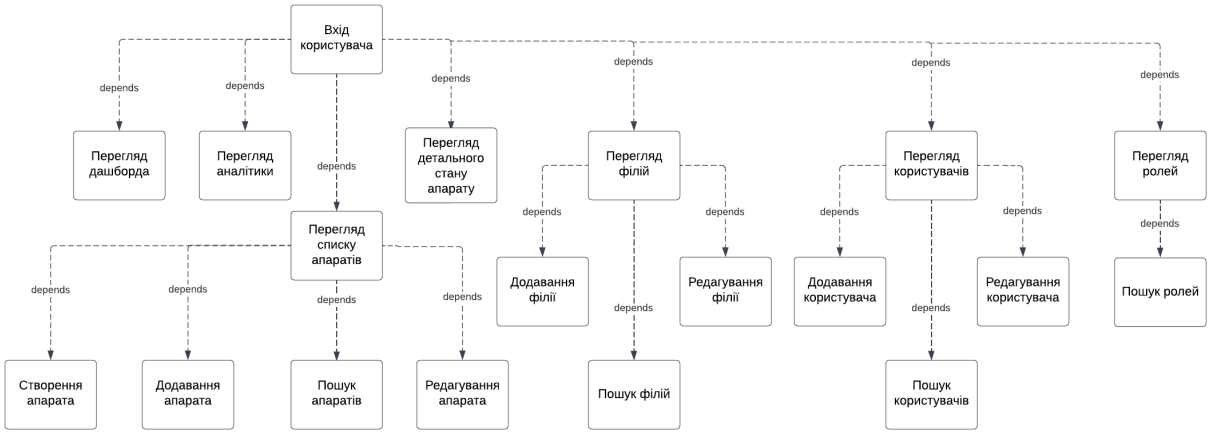


Рисунок 1.5 – Модель вимог у загальному вигляді

Таблиця 1.38 – Функціональна вимога FR-1

Назва	Вхід у застосунок
Опис	Система повинна надавати можливість авторизації користувачеві шляхом введення імені користувача та пароля

Таблиця 1.39 – Функціональна вимога FR-2

Назва	Скидання пароля
Опис	Система повинна надавати можливість користувачеві скинути поточний пароль та зберегти новий шляхом введення електронної пошти та коду, що на неї надійде

Таблиця 1.40 – Функціональна вимога FR-3

Назва	Перегляд дашборду
Опис	Система повинна надавати можливість переглядати дашборд авторизованому користувачеві

Таблиця 1.41 – Функціональна вимога FR-4

Назва	Перегляд аналітики
Опис	Система повинна надавати можливість переглядати аналітику авторизованому користувачеві

Таблиця 1.42 – Функціональна вимога FR-5

Назва	Перегляд списку апаратів
Опис	Система повинна надавати можливість переглядати список апаратів авторизованому користувачеві

Таблиця 1.43 – Функціональна вимога FR-6

Назва	Керування апаратами
Опис	Система повинна надавати можливість керувати апаратами авторизованому користувачеві

Таблиця 1.44 – Функціональна вимога FR-7

Назва	Перегляд детального стану апаратів
Опис	Система повинна надавати можливість переглядати детальний стан апаратів авторизованому користувачеві

Таблиця 1.45 – Функціональна вимога FR-8

Назва	Керування детальним станом апаратів
Опис	Система повинна надавати можливість керувати детальним станом апаратів авторизованому користувачеві

Таблиця 1.46 – Функціональна вимога FR-9

Назва	Перегляд списку філій
Опис	Система повинна надавати можливість переглядати список філій авторизованому користувачеві

Таблиця 1.47 – Функціональна вимога FR-10

Назва	Керування філіями
Опис	Система повинна надавати можливість керувати філіями авторизованому користувачеві

Таблиця 1.48 – Функціональна вимога FR-11

Назва	Перегляд списку користувачів
Опис	Система повинна надавати можливість переглядати список користувачів авторизованому користувачеві

Таблиця 1.49 – Функціональна вимога FR-12

Назва	Керування користувачами
Опис	Система повинна надавати можливість керувати користувачами авторизованому користувачеві

Таблиця 1.50 – Функціональна вимога FR-13

Назва	Перегляд списку ролей
Опис	Система повинна надавати можливість переглядати список ролей авторизованому користувачеві

Таблиця 1.51 – Функціональна вимога FR-14

Назва	Оновлення персональних даних
Опис	Система повинна надавати можливість оновлювати персональні дані авторизованому користувачеві

Таблиця 1.52 – Функціональна вимога FR-15

Назва	Зміна пароля
Опис	Система повинна надавати можливість змінювати пароль авторизованому користувачеві шляхом введення поточного пароля, нового пароля та підтвердження нового пароля

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11	FR-12	FR-13	FR-14	FR-15
UC-1	+														
UC-2															+
UC-3														+	
UC-4			+												
UC-5				+											
UC-6					+										
UC-7						+									
UC-8						+									
UC-9					+										
UC-10						+									
UC-11							+								
UC-12							+								
UC-13							+								
UC-14							+								
UC-15							+								
UC-16							+								
UC-17							+								
UC-18								+							
UC-19		+													
UC-20							+								
UC-21							+								
UC-22								+							
UC-23								+							
UC-24								+							
UC-25								+							
UC-26								+							
UC-27									+						
UC-28									+						
UC-29										+					
UC-30										+					
UC-31											+				
UC-32											+				
UC-33												+			
UC-34												+			
UC-35													+		
UC-36													+		

Рисунок 1.6 – Матриця трасування вимог

1.4.2 Розроблення нефункціональних вимог

У веб-застосунку, що розроблятиметься в рамках дипломного проєктування, виділимо наступні нефункціональні вимоги:

– система має два інтерфейси:
    1) світлий;
    2) темний;
– система має інтерфейси трьома різними шрифтами, що використовуються у всіх компонентах:
    1) Roboto;
    2) Poppins;
    3) Inter;
– система має інферфейси двома мовами:
    1) українською;
    2) англійською;
– система є пристосованою до підтримування та відновлення;
– система є надійною;
– система повинна працювати на різних платформах та різних за розміром девайсах без зміни зручності її використання.

1.5 Постановка задачі

Веб-застосунок призначений для спрощеного віддаленого керування апаратами й спостереження за ними за допомогою легкого інтерфейсу.

Метою розробки є зменшення частоти виїздів сервісних спеціалістів до апаратів очищення води, прискорення швидкості реакції сервісної служби, запобігання виникненню аварійних ситуацій.

Веб-застосунок має виконувати наступні основні задачі:

– відображення списку апаратів;
– відображення детального стану вибраних користувачами апаратів;
– створення апаратів;
– редагування апаратів;
– створення філій;
– створення користувачів.

Висновки до розділу

У цьому розділі були розглянуті загальні положення, було описано та проаналізовано предметну область. Були проаналізовані відомі технічні рішення: архітектури програмного забезпечення, архітектурні патерни. Були наведені їхні переваги та обґрунтування, чому саме вони використовуватимуться в роботі.

Було проаналізовано програмні засоби й засоби розробки, які найкраще використовувати для застосунку, та обґрунтовано вибір мов програмування, бібліотек, фреймворків, текстового редактора та інтегрованого середовища розробки.

Було проведено порівняльний аналіз застосунку з іншим програмним продуктом, який майже повністю реалізує його функціонал.

Крім того, було проведено аналіз вимог до програмного забезпечення: розроблено варіанти використання, функціональні та нефункціональні вимоги, створено матрицю трасування вимог.

Отже, відбулися постановка задачі та наналіз вимог до програмного забезпечення.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для опису бізнес процесів програмного забезпечення використовуються BPMN моделі (рисунки 2.1 та 2.2).

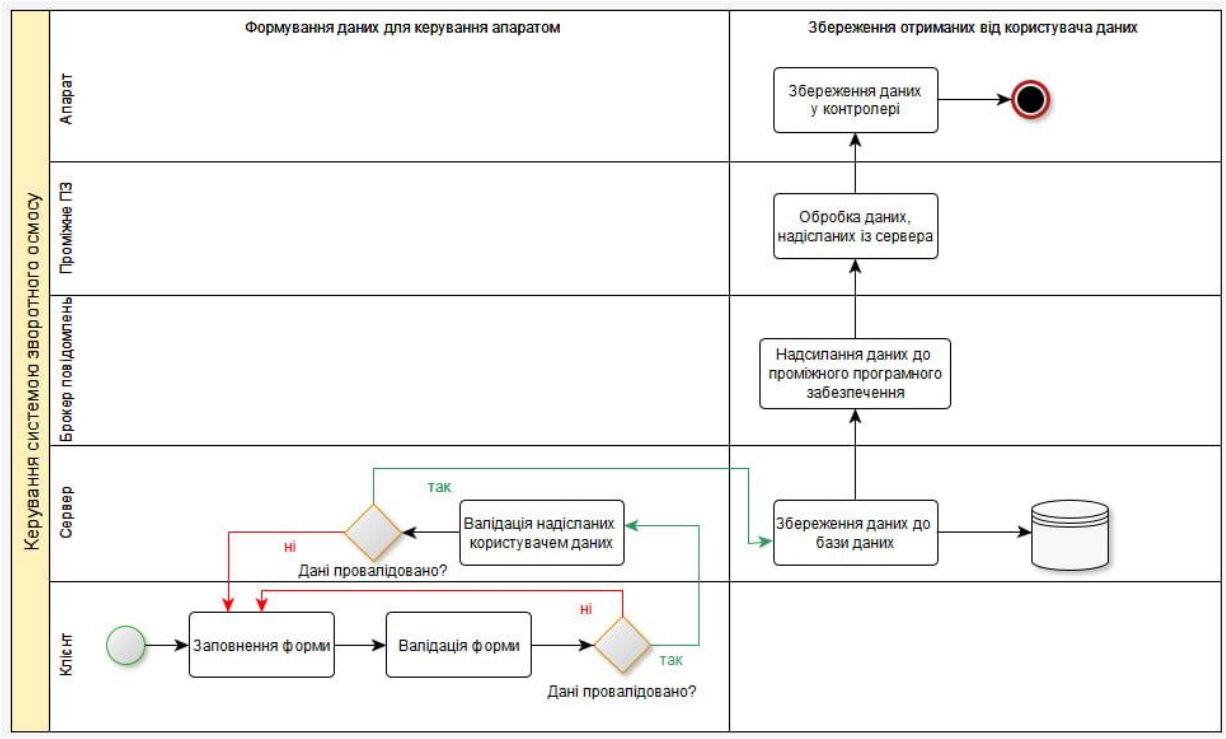


Рисунок 2.1 – Схема бізнес-процесу керування системою зворотного осмосу

Опис послідовності керування системою зворотного осмосу:

– користувач заповнює форму, яка проходить валідацію у клієнтській частині застосунку;;

– якщо дані пройшли валідацію, вони відправляються на сервер, в іншому випадку користувачеві потрібно буде ввести їх ще раз;

– дані проходять валідацію в серверні частині застосунку;

– якщо дані пройшли другу валідацію, вони зберігаються в базі даних, в іншому випадку користувачеві потрібно буде ввести їх ще раз;

– дані через брокер повідомлень надходять до проміжного програмного забезпечення, де вони обробляються, після чого надсилаються до апарату;

– оброблені дані зберігаються в контролері апарата зворотного осмосу.

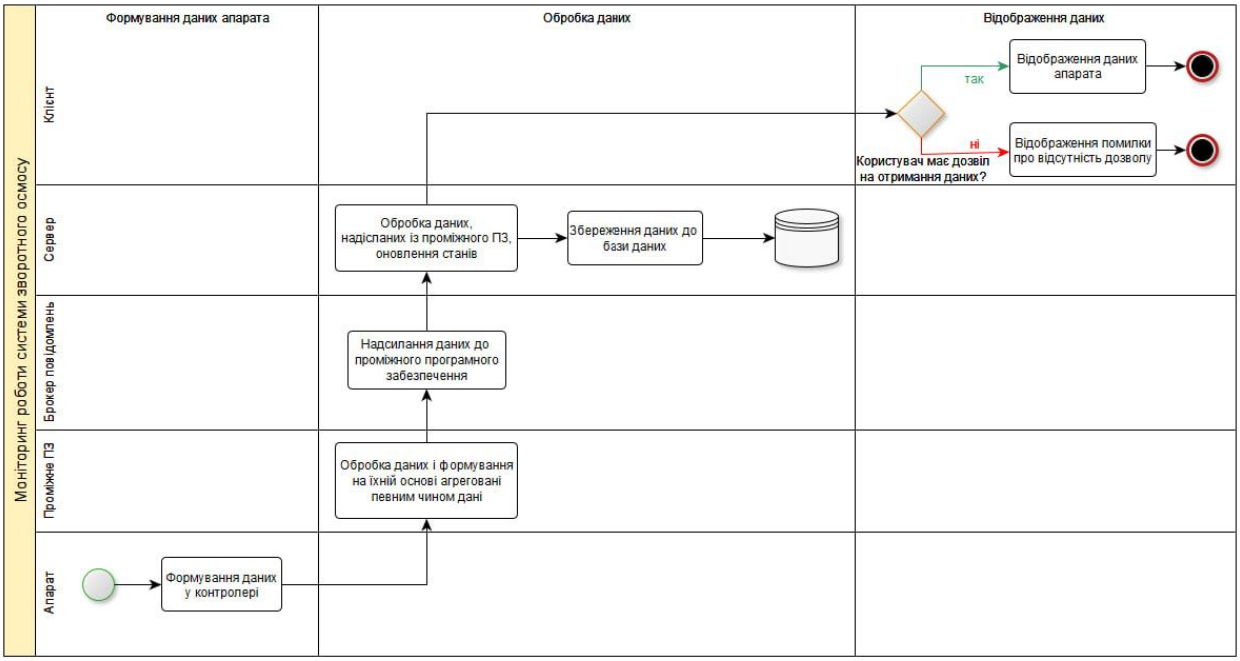


Рисунок 2.2 – Схема бізнес-процесу моніторингу роботи системи зворотного осмосу

Опис послідовності моніторингу роботи системи зворотного осмосу:

– у контролері апарата формуються дані;

– ці дані надсилаються до проміжного програмного забезпечення, яке обробляє їх і формує на основі них агреговані певним чином дані;

– сформовані дані через брокер повідомлень надсилаються на сервер;

– сервер обробляє ці дані (рахує статистику, записує логи, оновлює певні стани) та записує їх до бази даних;створення філій;

– дані за допомогою запитів користувача потрапляють на клієнтську частину застосунку, якщо користувач має лозволи на отримання цих даних, в

іншому випадку користувач отримує повідомлення про те, що дозволів на це в нього немає.

2.2 Архітектура програмного забезпечення

Веб-застосунок для віддаленого керування та моніторингу роботи систем очищення води має архітектуру, показану на рисунку 2.3.

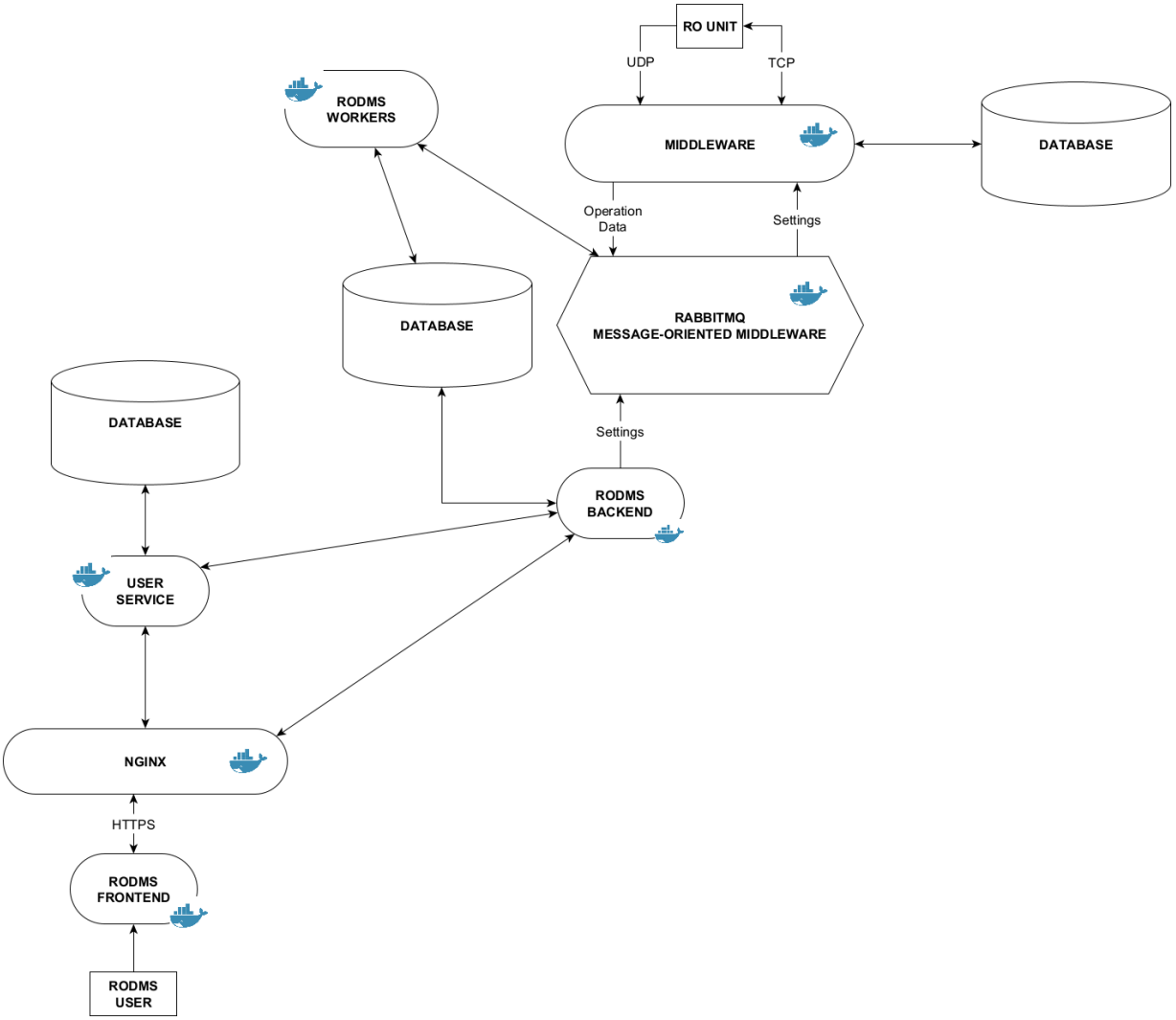


Рисунок 2.3 – Архітектура веб-застосунку для віддаленого керування та моніторингу роботи систем очищення води

RO Unit – це апарат зворотного осмосу. В апаратах є контролери, вони передають дані в middleware (проміжне програмне забезпечення), воно обробляє їх і формує на основі них агреговані певним чином дані, які надсилає в брокер повідомлень, а з брокера повідомлень дані відправляються на сервер, який їх обробляє (рахує статистику, записує логи, оновлює певні стани) та

записує до бази даних. Основний сервіс RODMS читає дані та записує їх до бази даних, оброблюючи з клієнтської частини додатку як запити на отримання даних, так і на їхню модифікацію. У разі останніх сервер може надсилати дані брокер повідомлень, і вони через проміжне програмне забезпечення надходять до апаратів. Для створення нових користувачів та для автентифікації використовується identity service (сервіс ідентифікації).

Основними сутностями в застосунку є апарат, користувач та філія. ER діаграма цих сутностей зображена на рисунку 2.4.

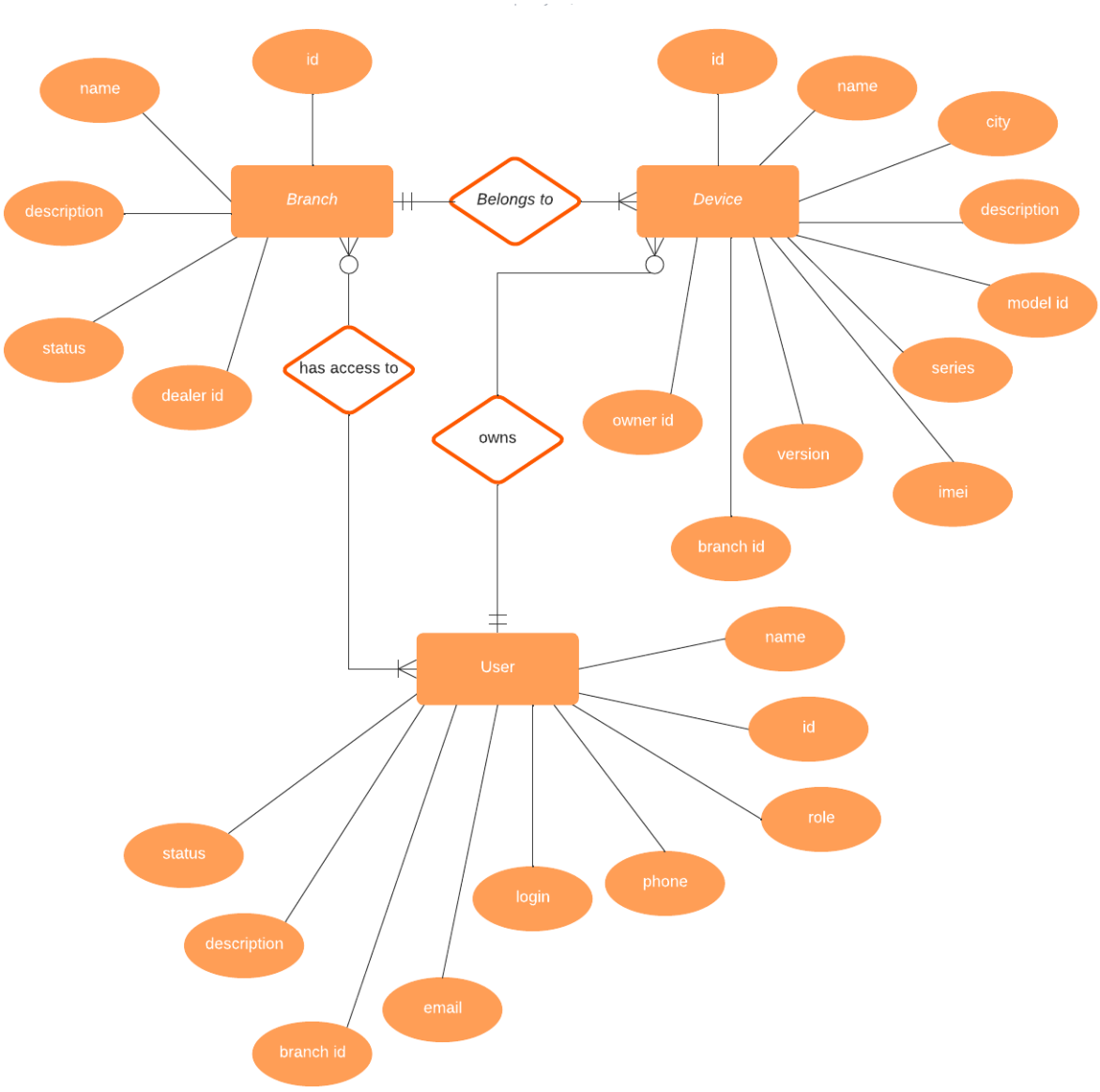


Рисунок 2.4 – ER діаграма сутностей User, Branch, Device

2.3 Конструювання програмного забезпечення

Одним із патернів архітектури програмного забезпечення, що використовується в розробці застосунку, є багаторівнева архітектура. На рівні бази даних знаходиться власне база даних, на рівні стікості знаходяться репозиторії, на рівні бізнесу знаходяться сервіси, на рівні презентації – контролери. Контролери та сервіси представлені в програмному забезпеченні за допомогою класів. Ці класи, а також зв'язки між ними, представлені в діаграмі класів на рисунку 2.5.

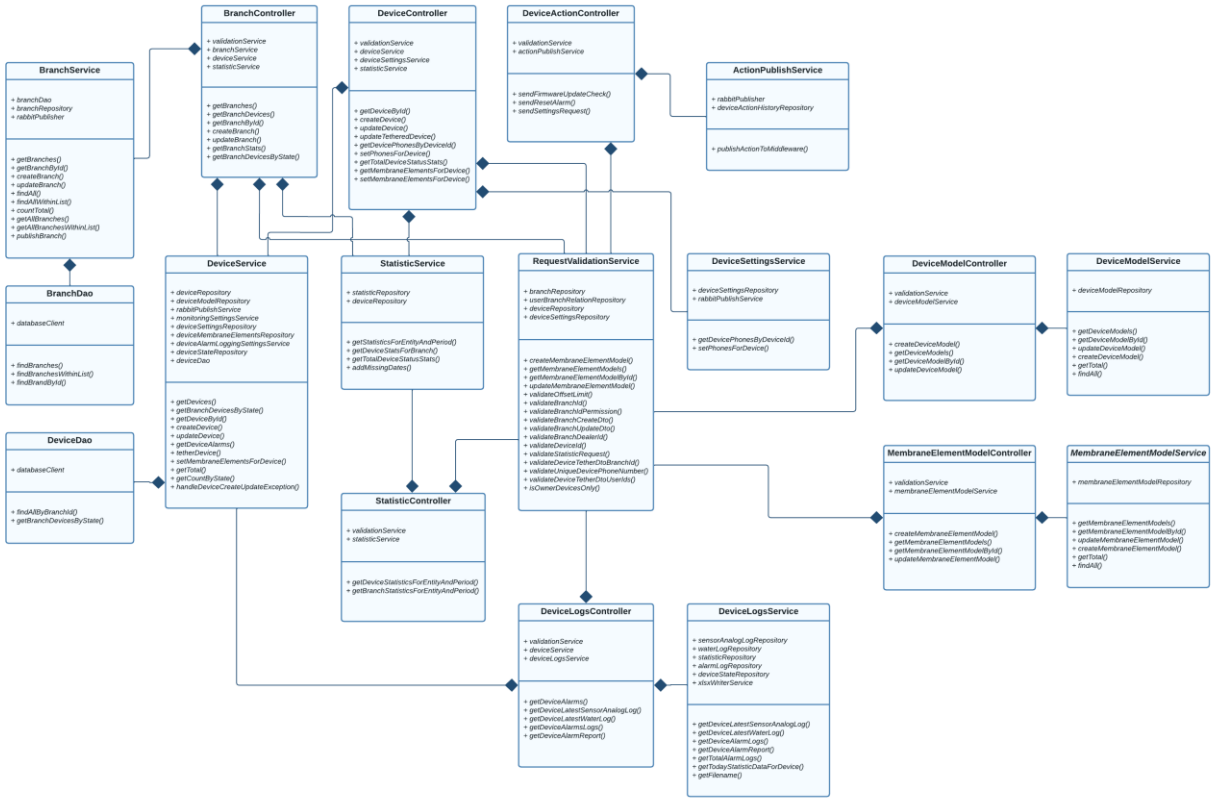


Рисунок 2.5 – Схема структурна класів

У якості системи управління базами даних використовується MariaDB. База даних сервера призначена для зберігання користувачів, даних апаратів, філій, аварій. Схема бази даних зображена на рисунку 2.6. Опис таблиць бази даних наведено в таблицях 2.1 - 2.13.

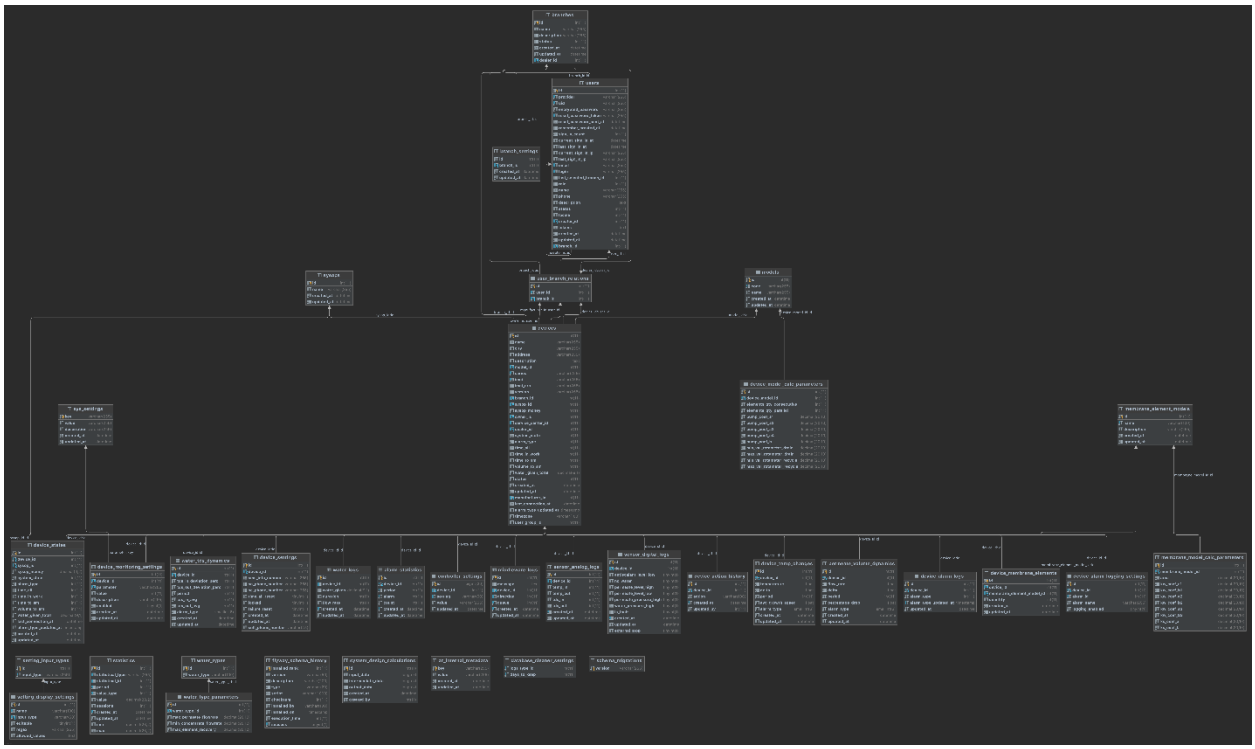


Рисунок 2.6 – Схема бази даних

Таблиця 2.1 – Опис таблиці users

Таблиця	Назва поля	Тип даних	Опис
users	id	int(11)	ідентифікатор користувача
	role	int(11)	роль користувача
	name	varchar(255)	ім’я користувача
	phone	varchar(255)	телефон користувача
	login	varchar(255)	логін користувача
	email	varchar(255)	електронна пошта користувача
	status	int(11)	статус користувача

Продовження таблиці 2.1

	branch_id	int(11)	ідентифікатор філії, до якої користувач має доступ
	created_at	datetime	дата й час створення запису
	updated_at	datetime	дата й час оновлення запису

Таблиця 2.2 – Опис таблиці branches

Таблиця	Назва поля	Тип даних	Опис
branches	id	int(11)	ідентифікатор філії
	name	varchar(255)	назва філії
	description	varchar(255)	опис філії
	status	int(11)	статус філії
	dealer_id	int(11)	дилер філії
	created_at	datetime	дата й час створення запису
	updated_at	datetime	дата й час оновлення запису

Таблиця 2.3 – Опис таблиці devices

Таблиця	Назва поля	Тип даних	Опис
devices	id	int(11)	ідентифікатор апарата
	city	varchar(255)	місто, у якому знаходиться апарат
	name	varchar(255)	ім’я апарата

Продовження таблиці 2.3

	description	text	опис апарата
	model id	int(11)	ідентифікатор моделі апарата
	series	varchar(255)	серія апарата
	imei	varchar(255)	IMEI апарата
	branch id	int(11)	ідентифікатор філії, до якої належить апарат
	owner id	int(11)	ідентифікатор користувача, який володіє апаратом
	version	varchar(255)	версія апарата
	created_at	datetime	дата й час створення запису
	updated_at	datetime	дата й час оновлення запису

Таблиця 2.4 – Опис таблиці membrane_element_models

Таблиця	Назва поля	Тип даних	Опис
membrane_element_models	id	int(11)	ідентифікатор мембрани
	device_id	int(11)	ідентифікатор апарата
	action	varchar(100)	назва дії
	created_at	datetime	дата й час створення запису
	updated_at	datetime	дата й час оновлення запису

Таблиця 2.5 – Опис таблиці device_alarm_logging_settings

Таблиця	Назва поля	Тип даних	Опис
device_alarm_logging_settings	id	int(11)	ідентифікатор налаштування
	device_id	int(11)	ідентифікатор апарата
	alarm_id	int(11)	ідентифікатор аварії
	alarm_name	varchar(255)	назва аварії
	updated_by	int(11)	ідентифікатор користувача, який оновив налаштування

Таблиця 2.6 – Опис таблиці device_alarm_logs

Таблиця	Назва поля	Тип даних	Опис
device_alarm_logs	id	int(11)	ідентифікатор логування
	device_id	int(11)	ідентифікатор апарата
	action	varchar(100)	назва дії
	created_at	datetime	дата й час створення запису
	updated_by	int(11)	ідентифікатор користувача, який оновив налаштування

Таблиця 2.7 – Опис таблиці device_membrane_elements

Таблиця	Назва поля	Тип даних	Опис
device_membrane_elements	id	int(11)	ідентифікатор логування
	device_id	int(11)	ідентифікатор апарата

Продовження таблиці 2.7

	membrane_element_model_id	int(11)	ідентифікатор моделі мембранного елемента
	quantity	int(11)	кількість елементів
	created_at	datetime	дата й час створення запису
	updated_by	int(11)	ідентифікатор користувача, який оновив налаштування

Таблиця 2.8 – Опис таблиці device_monitoring_settings

Таблиця	Назва поля	Тип даних	Опис
device_monitoring_settings	id	int(11)	ідентифікатор налаштування
	device_id	int(11)	ідентифікатор апарата
	parameter	varchar(255)	ім’я параметру
	value	int(11)	значення налаштування
	description	varchar(255)	опис налаштування
	enabled	tinyint(1)	ідентифікатор того, чи ввімкнене налаштування
	created_at	datetime	дата й час створення запису
	updated_at	datetime	дата й час оновлення запису

Таблиця 2.9 – Опис таблиці device_states

Таблиця	Назва поля	Тип даних	Опис
device_states	id	int(11)	ідентифікатор стану
	device_id	int(11)	ідентифікатор апарата
	sysop_id	int(11)	ідентифікатор системного адміністратора
	system_state	int(11)	стан системи
	alarm_type	int(11)	тип аварії
	last_connection_at	datetime	дата й час останнього підключення до апарата
	created_at	datetime	дата й час створення запису
	updated_at	datetime	дата й час оновлення запису

Таблиця 2.10 – Опис таблиці models

Таблиця	Назва поля	Тип даних	Опис
models	id	int(11)	ідентифікатор моделі
	code	varchar(255)	код моделі
	name	varchar(255)	назва моделі
	created_at	datetime	дата й час створення запису
	updated_by	int(11)	ідентифікатор користувача, який оновив налаштування

Таблиця 2.11 – Опис таблиці sensor_analog_logs

Таблиця	Назва поля	Тип даних	Опис
sensor_analog_logs	id	int(11)	ідентифікатор логування
	device_id	int(11)	ідентифікатор апарата
	temp_in	varchar(255)	значення температури води на вході
	temp_out	int(11)	значення температури води на виході
	tds_in	varchar(255)	загальний солевміст на вході
	tds_out	tinyint(1)	загальний солевміст на виході
	created_at	datetime	дата й час створення запису
	updated_at	datetime	дата й час оновлення запису

Таблиця 2.12 – Опис таблиці water_logs

Таблиця	Назва поля	Тип даних	Опис
water_logs	id	int(11)	ідентифікатор логування
	device_id	int(11)	ідентифікатор апарата
	water_given	decimal(9,1)	кількість води
	duration	int(11)	тривалість очищення
	flow_rate	int(11)	швидкість потоку

Продовження таблиці 2.12

	created_at	datetime	дата й час створення запису
	updated_at	datetime	дата й час оновлення запису

Таблиця 2.13 – Опис таблиці water_tds_dynamics

Таблиця	Назва поля	Тип даних	Опис
water_tds_dynamics	id	int(11)	ідентифікатор логування
	device_id	int(11)	ідентифікатор апарата
	tds_in_deviation_perc	int(11)	відхилення у відсотках вхідного загального солевмісту
	tds_out_deviation_perc	int(11)	відхилення у відсотках вихідного загального солевмісту
	period	int(11)	період, протягом якого визначається відхилення по загальному солевмісту
	tds_in_avg	int(11)	середнє значення солевмісту на вході
	tds_out_avg	int(11)	середнє значення солевмісту на виході
	alarm_type	smallint(6)	тип аварії
	created_at	datetime	дата й час створення запису
	updated_at	datetime	дата й час оновлення запису

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується в розробці наведено в таблиці 2.14.

Таблиця 2.14 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	IntelliJ IDEA	Середовище розробки програмного забезпечення серверної частини дипломної роботи.
2	Visual Studio Code	Графічний редактор для розробки клієнтської частини дипломної роботи.
3	Swagger	Програмне забезпечення, необхідне для тестування REST запитів. Використовувалось для тестування API інтерфейсів.
4	Spring Boot	Фреймворк на мові програмування Java для розробки серверної частини дипломної роботи
5	WebFlux	Фреймворк для створення реактивної серверної частини застосунку
6	React	Бібліотека на мові програмування JavsScript, для створення користувацьких інтерфейсів

7	Redux	Бібліотека на мові програмування JavaScript, призначена для керування станом програм
8	React Router	Бібліотека на мові програмування JavaScript, призначена для доступу до стану роутера та навігації всередині компонентів
9	Axios	HTTP-клієнт для браузера та Node.js, в основі якого лежать проміси.
10	React Hook Form	Бібліотека на мові програмування JavaScript, призначена для зручної роботи з формами
11	Yup	Бібліотека на мові програмування JavaScript, призначена для валідації форм
12	i18next	Фреймворк на мові програмування JavaScript, призначений для інтернаціоналізації програми

2.4 Аналіз безпеки даних

У застосунку для безпечної передачі інформації між сервером та клієнтом використовується стандарт JWT (JSON Web Tokens). JWT – це зашифрований рядок, що складається з трьох частин: заголовка, корисних даних та підпису, – розділених крапкою. Розшифрований JWT – це JSON об’єкт.

При вході в застосунок сервер надсилає на клієнт два таких токени: access token (токен доступу) та refresh token (токен оновлення). Вони записуються в LocalStorage (об’єкт, що зберігає дані в браузері навіть після

його закриття), щоб можна було мати доступ до застосунку навіть через години після останнього використання без необхідності нового входу в додаток. Тривалість життя токена доступу (access token) складає одну годину, а токена оновлення – 24 години. Токен доступу надсилається на сервер з кожним запитом у полі «Authentication» (автентифікація) заголовку запиту. Сервер розшифровує токен та отримує дані про дозволи користувача. Якщо серед них є ті, які потрібні для обробки запиту користувача, сервер його оброблює й надсилає відповідну відповідь клієнтові. Інакше клієнт отримує відповідь із повідомленням про те, що в нього немає доступу до певних даних чи на зміну цих даних.

Так само при вході в застосунок сервер надсилає на клієнт ці дозволи й залежно від них застосунок показує, наприклад лише ті вкладки в меню, які доступні користувачеві. Також за опомогою цієї інформації можна захистити застосунок від уведення користувачем URL (uniform resource locator, єдиний вказівник на ресурс), що веде на захищені сторінки.

У випадку, якщо на сервер прийшов запит, а токен доступу вже протермінований, відправляється відповідь із 401 кодом (Unauthorized, неавторизований). Після цього клієнт відсилає запит з токеном оновлення, якщо цей токен ще не протемінований, то клієнтові приходить відповідь відповідно до запиту та нові токен доступу й токен оновлення.

Висновки до розділу

У цьому розділі були представлені бізнес процеси системи за допомогою BPMN діаграм. Також була представлена діаграма з архітектурою застосунку та розписані її елементи та їхні функції.

Також було створено ER діаграму із сутностями User (користувач), Branch (філія), Device (апарат). Також були наведені описи таблиць users, branches, devices. Був наведений опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці програмного забезпечення.

Крім того, було розписано, за допомогою чого реалізована безпека даних у програмному забезпеченні, і як це реалізовано на клієнтській та на серверній частинах застосунку.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Для оцінки якості програмного забезпечення треба визначити метрики якості ПЗ, за якими система, створена в рамках дипломного проєктування, може бути протестована.

Такими метриками є:

– здатність підтримувати код (характеризує легкість і швидкість, із якою код може бути змінений або розширений для оптимізації наявної функціональності та підвищення якості продукту [7]);

– ефективність (досягається шляхом видалення зайвого коду, антипатернів, вразливостей та дотримання стандартів кодування [7]);

– надійність (характеризує частоту провалів, здатність витримувати відмову компонентів, здатність відновлюватися після несправного стану [7]);

– здатність піддаватися аналізу (характеризує здатність діагностувати недоліки чи причини несправностей або ідентифікувати частини, які необхідно модифікувати [7]);

– безпека (захищає програмне забезпечення від зловмисних атак і загроз, що допомагає програмному забезпеченню зберегти свою цілісність та автентифікацію [7]).

Для того, щоб проаналізувати програмне забезпечення за вказаними метриками буде використана платформа аналітики коду Embold [7]. Дана платформа була обрана, оскільки вона надає такі параметри моніторингу, як ключові індикатори ефективності, спеціальні контрольні точки якості, що допомагає оцінити інженерний та бізнес вплив різних різних проблем у коді. Після проведення аналізу платформа надає як загальний рейтинг у вигляді оцінки від -5 до 5, так і оцінки кожної з метрик окремо від 0 до 100.

На рисунках 3.1 – 3.6 показані результати аналізу коду застосунку за допомогою платформи Embold.

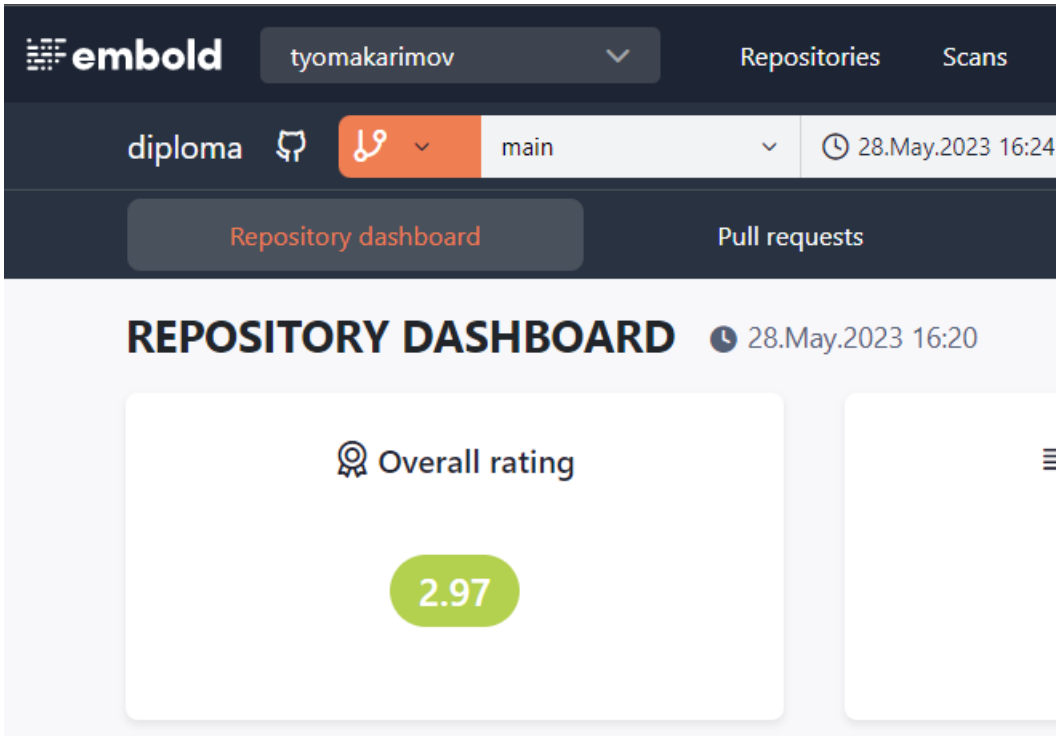


Рисунок 3.1 – Загальний рейтинг аналізу коду

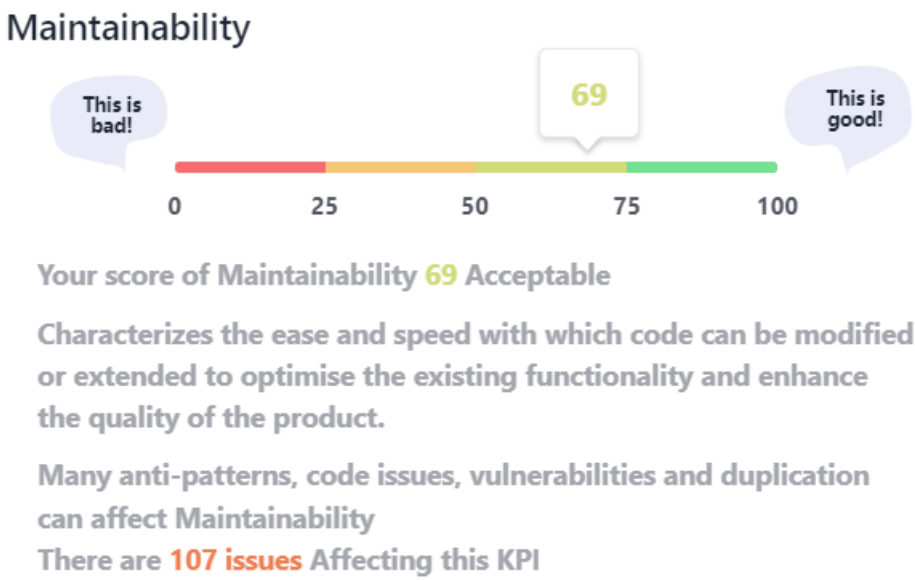


Рисунок 3.2 – Оцінка аналізу коду за метрикою здатності підтримувати код

Рисунок 3.3 – Оцінка аналізу коду за метрикою ефективності

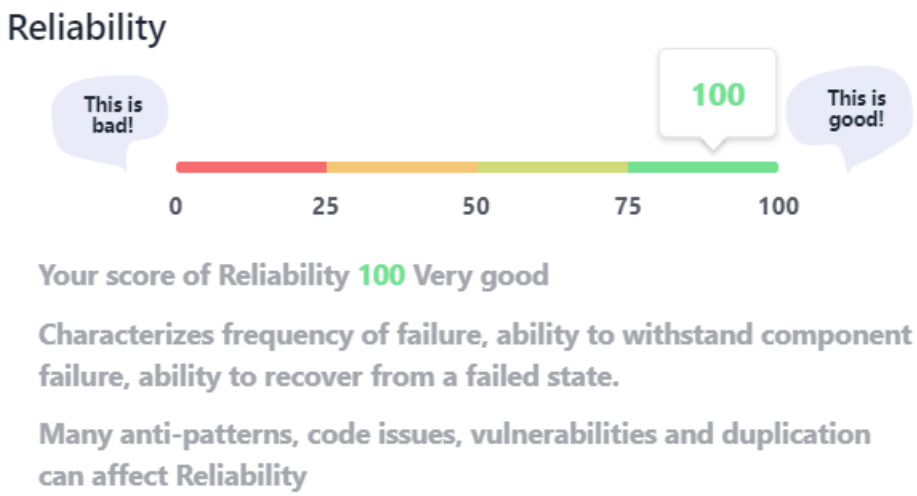


Рисунок 3.4 – Оцінка аналізу коду за метрикою надійності

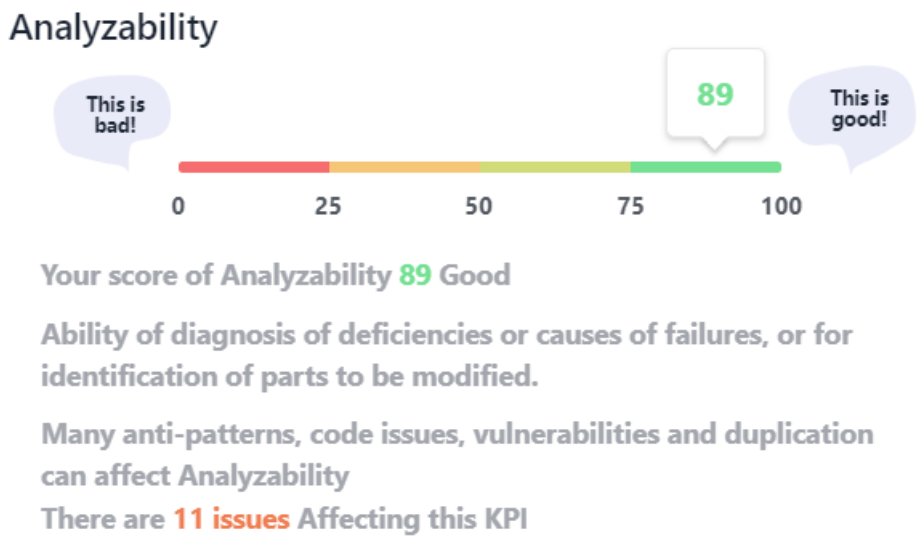


Рисунок 3.5 – Оцінка аналізу коду за метрикою здатності піддаватися аналізу

Рисунок 3.6 – Оцінка аналізу коду за метрикою безпеки

Як можемо побачити з рисунків 3.2 та 3.4 операційні нефункціональні вимоги виконані, адже оцінка здатності системи до підтримки коду є прийнятною, а оцінка надійності – дуже доброю.

3.2 Опис процесів тестування

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 3.1 – 3.30.

Таблиця 3.1 – Тест 1.1

Тест	Вхід у застосунок
Модуль	Вхід у застосунок
Номер тесту	1.1
Початковий стан системи	Користувач знаходиться на сторінці логіну
Вхідні дані	Ім’я користувача, пароль
Опис проведення тесту	У відповідні поля вводяться: ім’я зареєстрованого в системі користувача, коректний пароль. Після цього натискається кнопка «Вхід».
Очікуваний результат	Вхід у застосунок проходить успішно. Відбувається перенаправлення на головну сторінку з дашбордом.
Фактичний результат	Вхід у застосунок проходить успішно. Відбувається перенаправлення на головну сторінку з дашбордом.

Таблиця 3.2 – Тест 1.2

Тест	Вхід у застосунок із некоректним логіном
Модуль	Вхід у застосунок
Номер тесту	1.2
Початковий стан системи	Користувач знаходиться на сторінці логіну
Вхідні дані	Ім’я користувача, пароль
Опис проведення тесту	У відповідні поля вводяться: ім’я незареєстрованого в системі користувача, пароль. Після цього натискається кнопка «Вхід».
Очікуваний результат	Отримується повідомлення про те, що логін не вірний, і користувач залишається на сторінці входу в застосунок.
Фактичний результат	Отримується повідомлення про те, що логін не вірний, і користувач залишається на сторінці входу в застосунок.

Таблиця 3.3 – Тест 1.3

Тест	Вхід у застосунок із некоректним паролем
Модуль	Вхід у застосунок
Номер тесту	1.3
Початковий стан системи	Користувач знаходиться на сторінці логіну
Вхідні дані	Ім’я користувача, пароль
Опис проведення тесту	У відповідні поля вводяться: ім’я зареєстрованого в системі користувача, некоректний пароль. Після цього натискається кнопка «Вхід».
Очікуваний результат	Отримується повідомлення про те, що пароль не вірний, і користувач залишається на сторінці входу в застосунок.
Фактичний результат	Отримується повідомлення про те, що пароль не вірний, і користувач залишається на сторінці входу в застосунок.

Таблиця 3.4 – Тест 2.1

Тест	Відновлення пароля
Модуль	Відновлення пароля
Номер тесту	2.1
Початковий стан системи	Користувач знаходиться на сторінці логіну й не може увійти в застосунок, бо забув пароль
Вхідні дані	Електронна пошта, код, пароль

Продовження таблиці 3.4

Опис проведення тесту	Натискається посилання «Забули пароль?», після чого відбувається перенаправлення на сторінку скидання пароля, де вводиться електронна адреса, яка прив’язана до акаунту в застосунку, для того, щоб на цю електронну пошту було надіслано код, за допомогою якого можна буде відновити пароль. Після цього натискається кнопка «Надіслати», після чого з’являється форма з полями для введення коду та нового пароля. Вводиться код, який може містити тільки цифри, і їх має бути рівно 8, та новий пароль. Після цього натискається кнопка «Відновити».
Очікуваний результат	Отримується повідомлення про те, що пароль було успішно змінено, після чого відбувається перенаправлення на сторінку логіну.
Фактичний результат	Отримується повідомлення про те, що пароль було успішно змінено, після чого відбувається перенаправлення на сторінку логіну.

Таблиця 3.5 – Тест 2.2

Тест	Відновлення пароля із некоректною електронною поштою
Модуль	Відновлення пароля
Номер тесту	2.2
Початковий стан системи	Користувач знаходиться на сторінці логіну й не може увійти в застосунок, бо забув пароль
Вхідні дані	Електронна пошта, код, пароль

Продовження таблиці 3.5

Опис проведення тесту	Натискається посилання «Забули пароль?», після чого відбувається перенаправлення на сторінку скидання пароля, де вводиться електронна адреса, яка не прив’язана до жодного акаунту в застосунку. Після цього натискається кнопка «Надіслати».
Очікуваний результат	Отримується повідомлення про те, що користувача з такою електронною поштою не було знайдено.
Фактичний результат	Отримується повідомлення про те, що користувача з такою електронною поштою не було знайдено.

Таблиця 3.6 – Тест 2.3

Тест	Відновлення пароля із некоректним кодом
Модуль	Відновлення пароля
Номер тесту	2.3
Початковий стан системи	Користувач знаходиться на сторінці логіну й не може увійти в застосунок, бо забув пароль
Вхідні дані	Електронна пошта, код, пароль
Опис проведення тесту	Натискається посилання «Забули пароль?», після чого відбувається перенаправлення на сторінку скидання пароля, де вводиться електронна адреса, яка прив’язана до акаунту в застосунку, для того, щоб на цю електронну пошту було надіслано код, за допомогою якого можна буде відновити пароль. Після цього натискається кнопка «Надіслати», після чого з’являється форма з полями для введення коду та нового пароля. Далі вводиться неправильний код та новий пароль. Після цього натискається кнопка «Відновити».

Продовження таблиці 3.6

Очікуваний результат	Отримується повідомлення про те, що було введено невірний код.
Фактичний результат	Отримується повідомлення про те, що було введено невірний код.

Таблиця 3.7 – Тест 3.1

Тест	Оновлення персональних даних
Модуль	Оновлення персональних даних
Номер тесту	3.1
Початковий стан системи	Користувач знаходиться на головній сторінці застосунку
Вхідні дані	Ім’я, прізвище, опис, номер телефону, електронна пошта
Опис проведення тесту	Натискається кнопка з іконками користувача та зубчастого колеса справа зверху й вибирається опція «Налаштування». Відбувається перехід на сторінку з профілем. Щонайменше одне з полів із персональною інформацією (ім’я, прізвище, опис) чи контактною інформацією (номер телефону, електронна пошта) змінюються. Натискається кнопка «Оновити профіль».
Очікуваний результат	Користувач отримує повідомлення про те, що профіль було успішно оновлено.
Фактичний результат	Отримує повідомлення про те, що профіль було успішно оновлено.

Таблиця 3.8 – Тест 4.1

Тест	Зміна пароля
Модуль	Зміна пароля
Номер тесту	4.1

Продовження таблиці 3.8

Початковий стан системи	Користувач знаходиться на головній сторінці застосунку
Вхідні дані	Актуальний пароль, новий пароль
Опис проведення тесту	Натискається кнопка з іконками користувача та зубчастого колеса справа зверху й вибирається опція «Налаштування». Відбувається перехід на сторінку з профілем та на вкладку зі зміною пароля. Заповнюються поля з актуальним паролем, новим паролем та підтвердженням нового пароля. Натискається кнопка «Змінити пароль».
Очікуваний результат	Отримується повідомлення про те, що пароль було успішно змінено.
Фактичний результат	Користувач отримує повідомлення про те, що пароль було успішно змінено.

Таблиця 3.9 – Тест 4.2

Тест	Зміна пароля із некоректним актуальним паролем
Модуль	Зміна пароля
Номер тесту	4.2
Початковий стан системи	Користувач знаходиться на головній сторінці застосунку
Вхідні дані	Актуальний пароль, новий пароль

Продовження таблиці 3.9

Опис проведення тесту	Натискається кнопка з іконками користувача та зубчастого колеса справа зверху й вибирається опція «Налаштування». Відбувається перехід на сторінку з профілем та на вкладку зі зміною пароля. Заповнюються поля з актуальним паролем, куди вводить невірний пароль, новим паролем та підтвердженням нового пароля. Натискається кнопка «Змінити пароль».
Очікуваний результат	Отримується повідомлення про те, що поточний пароль не вірний.
Фактичний результат	Отримується повідомлення про те, що поточний пароль не вірний.

Таблиця 3.10 – Тест 5.1

Тест	Зміна поточної філії
Модуль	Зміна поточної філії
Номер тесту	5.1
Початковий стан системи	Користувач знаходиться на головній сторінці застосунку
Вхідні дані	-
Опис проведення тесту	У хедері натискається ім’я поточної філії, з’являється випадаючий список, вибирається інша філія.
Очікуваний результат	Змінюються апарати в таблиці й дані в картках на головній сторінці застосунку та на всіх інших сторінках.
Фактичний результат	Змінюються апарати в таблиці й дані в картках на головній сторінці застосунку та на всіх інших сторінках.

Таблиця 3.11 – Тест 6.1

Тест	Зміна кольору інтерфейсу
Модуль	Нефункціональні вимоги до інтерфейсу
Номер тесту	6.1
Початковий стан системи	Авторизований користувач знаходиться на будь-якій сторінці застосунку. Колір його інтерфейсу – білий
Вхідні дані	-
Опис проведення тесту	Натискається кнопка з зубчастим колесом, яке крутиться, що знаходиться посередині справа. З того ж боку з’являється бокове меню. Натискається невибрана опція картці «Mode».
Очікуваний результат	Колір інтерфейсу змінюється на протилежний – чорний
Фактичний результат	Колір інтерфейсу змінюється на протилежний – чорний

Таблиця 3.12 – Тест 6.2

Тест	Зміна шрифту інтерфейсу
Модуль	Нефункціональні вимоги до інтерфейсу
Номер тесту	6.2
Початковий стан системи	Авторизований користувач знаходиться на будь-якій сторінці застосунку. Шрифт, що використовується на сторінках застосунку, – Roboto
Вхідні дані	-
Опис проведення тесту	Натискається кнопка з зубчастим колесом, яке крутиться, що знаходиться посередині справа. З того ж боку з’являється бокове меню. Натискається опція «Poppins» у картці «Font Family».

Продовження таблиці 3.12

Очікуваний результат	Шрифт, що використовується на сторінках застосунку змінюється на Poppins.
Фактичний результат	Шрифт, що використовується на сторінках застосунку змінюється на Poppins.

Таблиця 3.13 – Тест 6.3

Тест	Зміна мови інтерфейсу
Модуль	Нефункціональні вимоги до інтерфейсу
Номер тесту	6.3
Початковий стан системи	Авторизований користувач знаходиться на будь-якій сторінці застосунку. Мова, що використовується на сторінках застосунку, – українська
Вхідні дані	-
Опис проведення тесту	Натискається кнопка з кодом поточної мови інтерфейсу (UA). У випадаючому списку обирається опція «English (EN)».
Очікуваний результат	Мова, що використовується на сторінках застосунку змінюється на англійську.
Фактичний результат	Мова, що використовується на сторінках застосунку змінюється на англійську.

Таблиця 3.14 – Тест 7.1

Тест	Створення апарата
Модуль	Створення апарата
Номер тесту	7.1
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку

Продовження таблиці 3.14

Вхідні дані	IMEI, серійний номер, модель
Опис проведення тесту	Натискається вкладка «Апарати» в боковому меню й вибирається дочірня вкладка «Перелік». Відображається перелік апаратів поточної філії у вигляді таблиці, над якою натискається кнопка «Створити». З’являється модальне вікно з формою. Вводяться IMEI, серійний номер та вибирається модель із випадаючого списку. Натискається кнопка «Зберегти».
Очікуваний результат	Отримується повідомлення про те, що апарат було успішно створено.
Фактичний результат	Отримується повідомлення про те, що апарат було успішно створено.

Таблиця 3.15 – Тест 8.1

Тест	Додавання апарата
Модуль	Додавання апарата
Номер тесту	8.1
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	Ім’я, адреса, IMEI, серійний номер, модель, власник, опис
Опис проведення тесту	Натискається вкладка «Апарати» в боковому меню й вибирається дочірня вкладка «Перелік». Відображається перелік апаратів поточної філії у вигляді таблиці, над якою натискається кнопка «Додати». З’являється модальне вікно з формою. Вводяться ім’я, адреса, IMEI, серійний номер та вибираються модель та власник із випадаючого списку, вводиться опис. Натискається кнопка «Зберегти».

Продовження таблиці 3.15

Очікуваний результат	Отримується повідомлення про те, що апарат було успішно додано.
Фактичний результат	Отримується повідомлення про те, що апарат було успішно додано.

Таблиця 3.16 – Тест 9.1

Тест	Оновлення апарата
Модуль	Оновлення апарата
Номер тесту	9.1
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	IMEI, серійний номер, модель
Опис проведення тесту	Натискається вкладка «Апарати» в боковому меню й вибирається дочірня вкладка «Перелік». Відображається перелік апаратів поточної філії у вигляді таблиці, у якій є колонка «Дія». Натискається рядок цієї колонки, відкривається модальне вікно з даними апарата для редагування. Змінюється щонайменше одне з наступних значень: IMEI, серійний номер, модель. Натискається кнопка «Зберегти».
Очікуваний результат	Отримується повідомлення про те, що апарат було успішно оновлено.
Фактичний результат	Отримується повідомлення про те, що апарат було успішно оновлено.

Таблиця 3.17 – Тест 10.1

Тест	Вибір апарата на сторінці детального стану
Модуль	Детальний стан апарата
Номер тесту	10.1
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	-
Опис проведення тесту	Натискається вкладка «Апарати» в боковому меню й вибирається дочірня вкладка «Детальний стан». Натискається поле з запонювачем «Оберіть апарат». З’являється випадаючий список із ID та назвою апаратів, натискається один із них.
Очікуваний результат	Підвантажується загальна інформація про апарат.
Фактичний результат	Підвантажується загальна інформація про апарат.

Таблиця 3.18 – Тест 10.2

Тест	Редагування загальної інформації апарата
Модуль	Детальний стан апарата
Номер тесту	10.2
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	Ім’я, адреса, власник, опис

Продовження таблиці 3.18

Опис проведення тесту	Натискається вкладка «Апарати» в боковому меню й вибирається дочірня вкладка «Детальний стан». Натискається поле з запонювачем «Оберіть апарат». З'являється випадаючий список із ID та назвою апаратів, натискається один із них. Підвантажується загальна інформація про апарат, натискається кнопка «Редагувати» у картці з інформацією про апарат, відкривається модальне вікно з даними апарата для редагування. Змінюється щонайменше одне з наступних значень: ім'я, адреса, власник, опис. Натискається кнопка «Зберегти».
Очікуваний результат	Отримується повідомлення про те, що апарат було успішно відредаговано.
Фактичний результат	Отримується повідомлення про те, що апарат було успішно відредаговано.

Таблиця 3.19 – Тест 10.3

Тест	Редагування номерів телефонів апарата
Модуль	Детальний стан апарата
Номер тесту	10.3
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	Номер телефону SMS, номер телефону сервісного центру, номер телефону системи

Продовження таблиці 3.19

Опис проведення тесту	Натискається вкладка «Апарати» в боковому меню й вибирається дочірня вкладка «Детальний стан». Натискається поле з запонювачем «Оберіть апарат». З'являється випадаючий список із ID та назвою апаратів, натискається один із них. Підвантажується загальна інформація про апарат, натискається кнопка «Редагувати» у картці з номерами телефонів, відкривається модальне вікно з формою. Змінюється щонайменше одне з наступних значень: номер телефону SMS, номер телефону сервісного центру, номер телефону системи. Натискається кнопка «Зберегти».
Очікуваний результат	Отримується повідомлення про те, що телефони апарата було успішно відредаговано.
Фактичний результат	Отримується повідомлення про те, що телефони апарата було успішно відредаговано.

Таблиця 3.20 – Тест 10.4

Тест	Редагування мембрани апарата
Модуль	Детальний стан апарата
Номер тесту	10.4
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	Назва мембрани, кількість

Продовження таблиці 3.20

Опис проведення тесту	Натискається вкладка «Апарати» в боковому меню й вибирається дочірня вкладка «Детальний стан». Натискається поле з запонювачем «Оберіть апарат». З'являється випадаючий список із ID та назвою апаратів, натискається один із них. Підвантажується загальна інформація про апарат, натискається кпопка «Редагувати» у картці з даними про мембрану апарата, відкривається модальне вікно з формою. Змінюється щонайменше одне з наступних значень: назва мембрани, кількість. Натискається кнопка «Зберегти».
Очікуваний результат	Отримується повідомлення про те, що мембрану апарата було успішно відредаговано.
Фактичний результат	Отримується повідомлення про те, що мембрану апарата було успішно відредаговано.

Таблиця 3.21 – Тест 10.5

Тест	Оновлення налаштувань логувань аварій
Модуль	Детальний стан апарата
Номер тесту	10.5
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	Назва мембрани, кількість

Продовження таблиці 3.21

Опис проведення тесту	Натискається вкладка «Апарати» в боковому меню й вибирається дочірня вкладка «Детальний стан». Натискається поле з запонювачем «Оберіть апарат». З’являється випадаючий список із ID та назвою апаратів, натискається один із них. Підвантажується загальна інформація про апарат, натискається таб «Логування Аварій», з’являється таблиця зі списком аварій та інформацією в колонці «Увімкнено» про те, чи вони включені до логування. Змінюється значення в чекбоксі в колонці «Увімкнено» одного з рядків на протилежне.
Очікуваний результат	Отримується повідомлення про те, що налаштування логувань аварій було успішно оновлено.
Фактичний результат	Отримується повідомлення про те, що налаштування логувань аварій було успішно оновлено.

Таблиця 3.22 – Тест 10.6

Тест	Завантаження історії аварій
Модуль	Детальний стан апарата
Номер тесту	10.6
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	Назва мембрани, кількість

Опис проведення тесту	Натискається вкладка «Апарати» в боковому меню й вибирається дочірня вкладка «Детальний стан». Натискається поле з запонювачем «Оберіть апарат». З’являється випадаючий список із ID та назвою апаратів, натискається один із них. Підвантажується загальна інформація про апарат, натискається таб «Історія Аварій», обирається період дат, натискається кнопка «Завантажити».
Очікуваний результат	Завантажується файл із історією аварій у форматі .xlsx.
Фактичний результат	Завантажується файл із історією аварій у форматі .xlsx.

Таблиця 3.23 – Тест 11.1

Тест	Додавання філії
Модуль	Додавання філії
Номер тесту	11.1
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	Ім’я філії
Опис проведення тесту	Натискається вкладка «Адміністрація» в боковому меню й вибирається дочірня вкладка «Філії», з’яляється список філій. Натискається кнопка «Створити», з’являється форма, вводиться значення імені філії. Натискається кнопка «Зберегти».
Очікуваний результат	Отримується повідомлення про те, що філію було успішно додано.

Продовження таблиці 3.23

Фактичний результат	Отримується повідомлення про те, що філію було успішно додано.

Таблиця 3.24 – Тест 12.1

Тест	Редагування філії
Модуль	Редагування філії
Номер тесту	12.1
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	Ім’я філії
Опис проведення тесту	Натискається вкладка «Адміністрація» в боковому меню й вибирається дочірня вкладка «Філії», з’являється список філій у вигляді таблиці, у якій є колонка «Дія». Натискається рядок цієї колонки, відкривається модальне вікно з формою, змінюється значення імені філії. Натискається кнопка «Зберегти».
Очікуваний результат	Отримується повідомлення про те, що філію було успішно відредаговано.
Фактичний результат	Отримується повідомлення про те, що філію було успішно відредаговано.

Таблиця 3.25 – Тест 13.1

Тест	Додавання користувача
Модуль	Додавання користувача
Номер тесту	13.1
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку

Продовження таблиці 3.25

Вхідні дані	Логін, ім’я, електронна пошта, номер телефону, філія, роль, пароль
Опис проведення тесту	Натискається вкладка «Адміністрація» в боковому меню й вибирається дочірня вкладка «Користувачі», з’являється список філій. Натискається кнопка «Створити», з’являється форма, вводяться логін, ім’я, електронна пошта, номер телефону, філія, роль, пароль. Натискається кнопка «Зберегти».
Очікуваний результат	Отримується повідомлення про те, що користувача було успішно додано.
Фактичний результат	Отримується повідомлення про те, що користувача було успішно додано.

Таблиця 3.26 – Тест 14.1

Тест	Редагування користувача
Модуль	Редагування користувача
Номер тесту	14.1
Початковий стан системи	Авторизований користувач знаходиться на головній сторінці застосунку
Вхідні дані	Логін, ім’я, електронна пошта, номер телефону, філія, роль, пароль

Продовження таблиці 3.26

Опис проведення тесту	Натискається вкладка «Адміністрація» в боковому меню й вибирається дочірня вкладка «Користувачі», з’являється список користувачів у вигляді таблиці, у якій є колонка «Дія». Натискається рядок цієї колонки, відкривається модальне вікно з формою, змінюється щонайменше одне з наступних значень: логін, ім’я, електронна пошта, номер телефону, філія, роль, пароль. Натискається кнопка «Зберегти».
Очікуваний результат	Отримується повідомлення про те, що дані користувача було успішно відредаговано.
Фактичний результат	Отримується повідомлення про те, що дані користувача було успішно відредаговано.

3.3 Опис контрольного прикладу

Застосунок, що розробляється в рамках дипломного проєктування, має два найголовніших завдання: моніторинг роботи систем очищення води та віддалене керування ними. У якості контрольного прикладу розглянемо процес моніторингу роботи систем. Крім того, будуть показані способи отримання даних, сформованих у результаті моніторингу, та адміністративних даних.

Початковим станом є знаходження на сторінці логіну (рисунок 3.7).

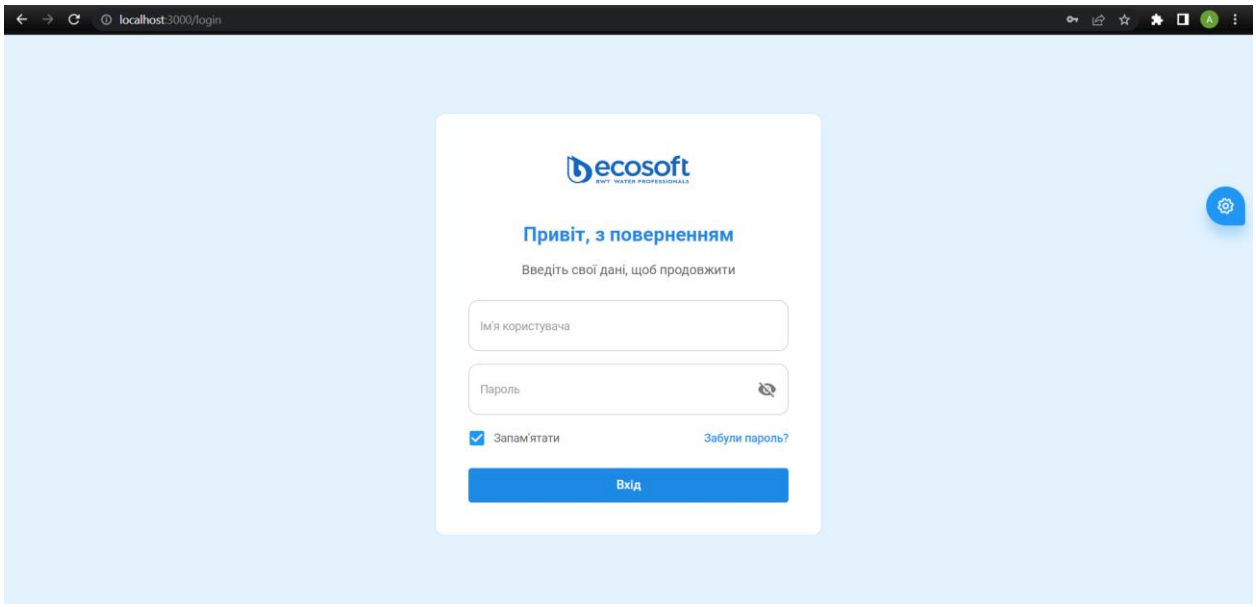


Рисунок 3.7 – Сторінка логіну

Далі вводяться ім’я користувача та пароль (рисунок 3.8).

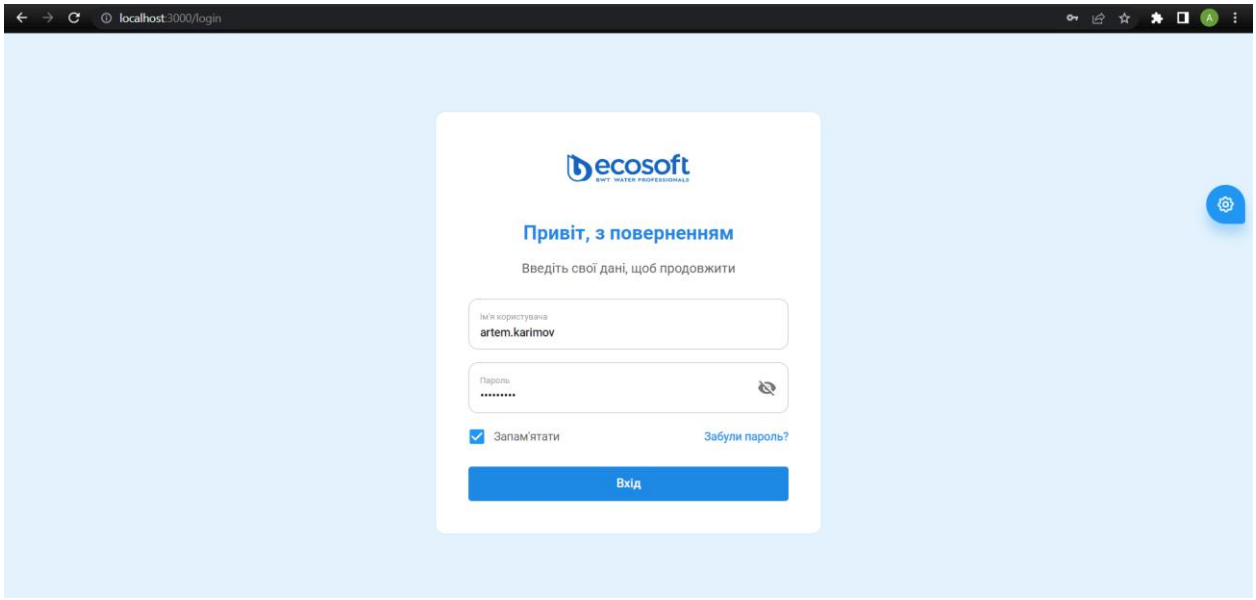


Рисунок 3.8 – Введені дані на сторінці логіну

Далі натискається кнопка «Вхід», та відбувається перехід на головну сторінку застосунку (рисунок 3.9).

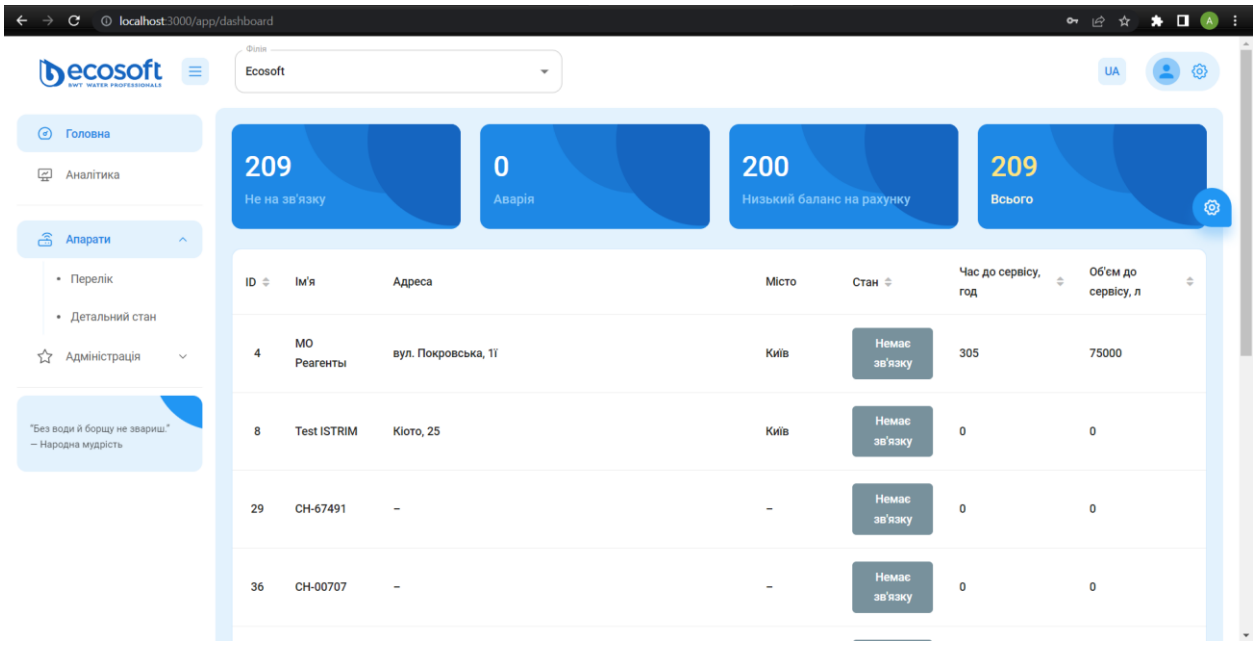


Рисунок 3.9 – Головна сторінка застосунку

Далі натискається вкладка «Аналітика» в боковому меню для перегляду аналітики (рисунок 3.10).

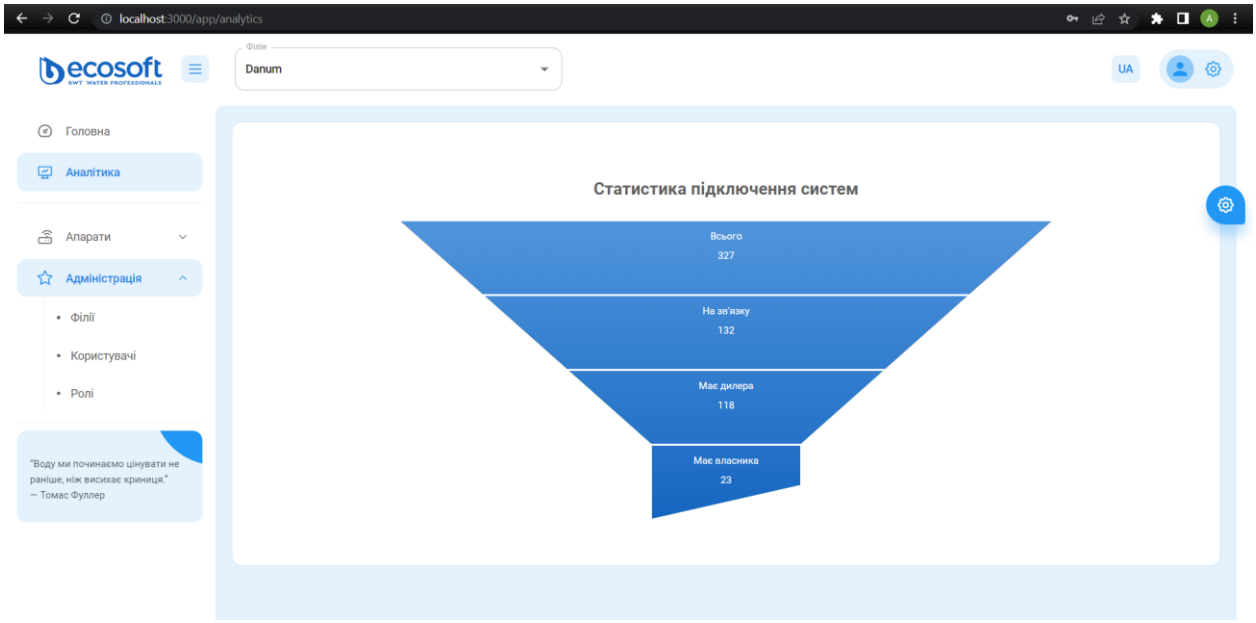


Рисунок 3.10 – Сторінка з аналітикою

Далі натискається вкладка «Апарати» в боковому меню й вибирається дочірня вкладка «Перелік», відбувається перехід на сторінку зі списком апаратів зворотного осмосу поточної філії (рисунок 3.11).

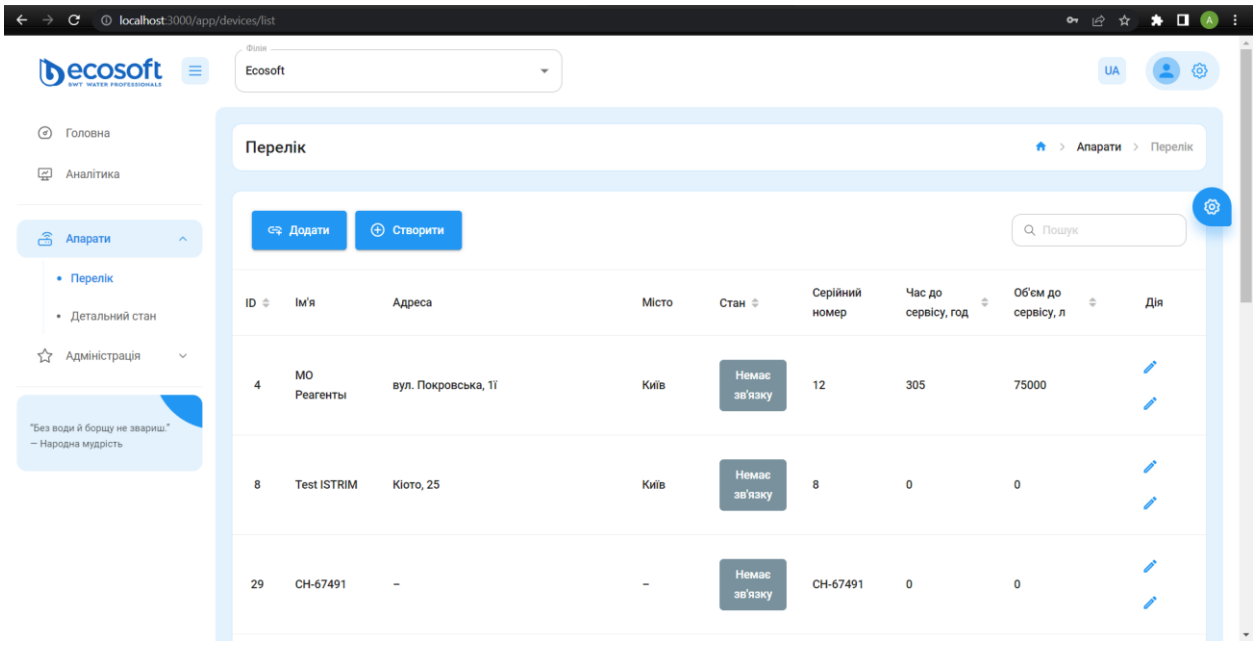


Рисунок 3.11 – Сторінка зі списком апаратів

Далі вибирається сусідня вкладка «Детальний стан». Відбувається перехід на сторінку з детальним станом апарата (рисунок 3.12).

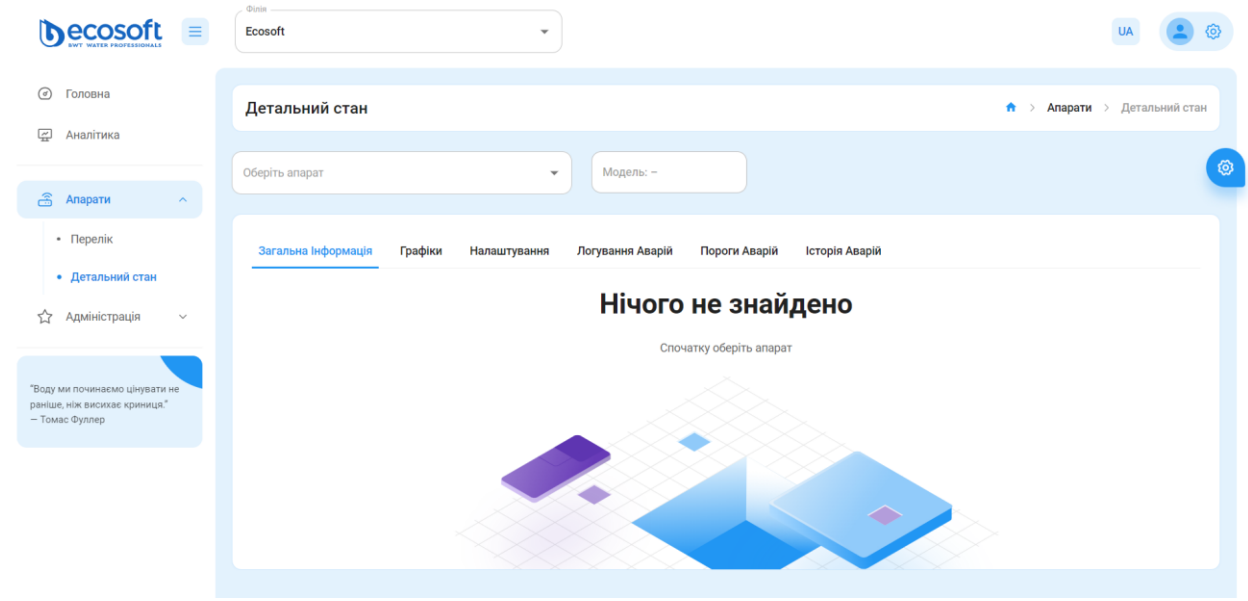


Рисунок 3.12 – Сторінка з детальним станом апарата

Після цього натискається поле «Оберіть апарат» та в списку, що випадає, обирається апарат. Відображається загальна інформація про апарат (рисунок 3.13).

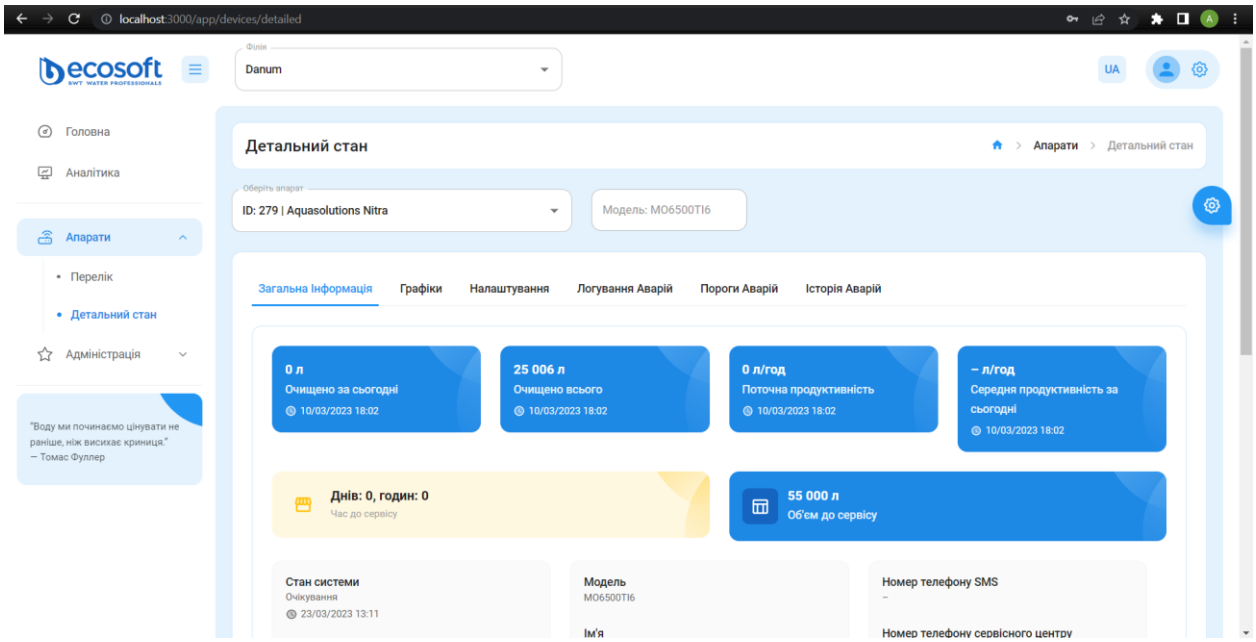


Рисунок 3.13 – Відображення загальної інформації про обраний апарат на сторінці з детальним станом апарата

Далі натискається таб «Графіки» на цій сторінці. Натискається кнопка «Оберіть дату», з'являється компонент для вибору проміжку дат, дані за який мають відобразитися на графіках (рисунок 3.14).

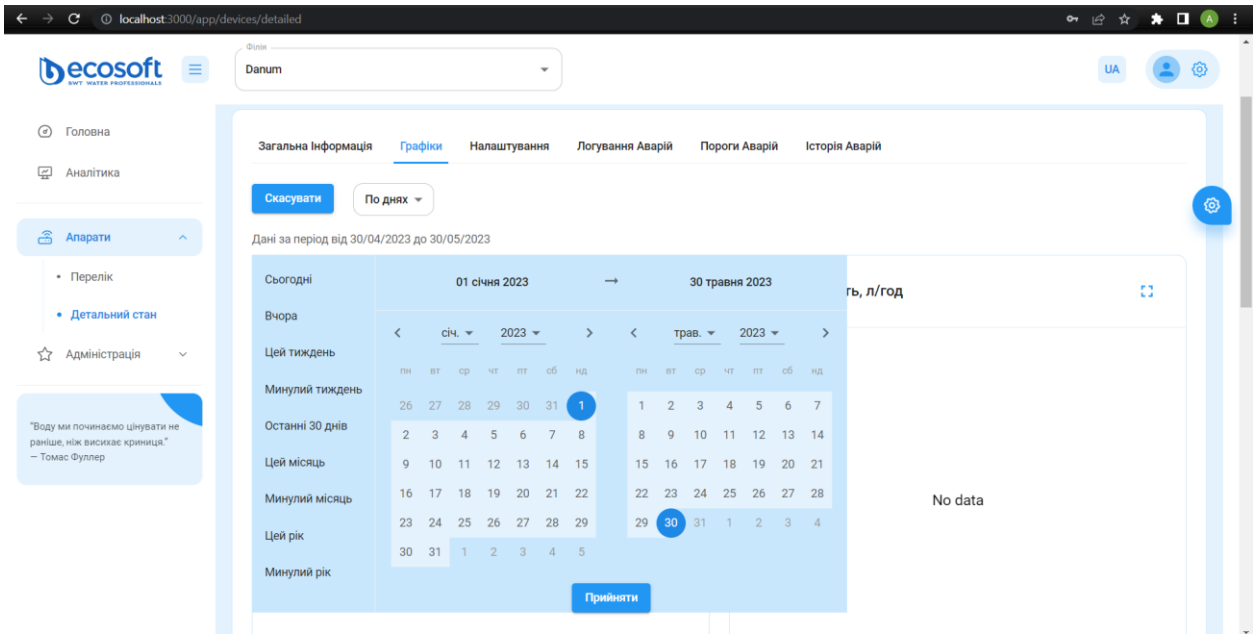


Рисунок 3.14 – Вибір проміжку дат

Натискається кнопка «Прийняти». Підвантажуються дані про кількість очищеної води за обраний період, а також продуктивність, загальний

солевміст вихідної води, загальний солевміст перміату, селективність, температуру й відображаються на графіках (рисунок 3.15).

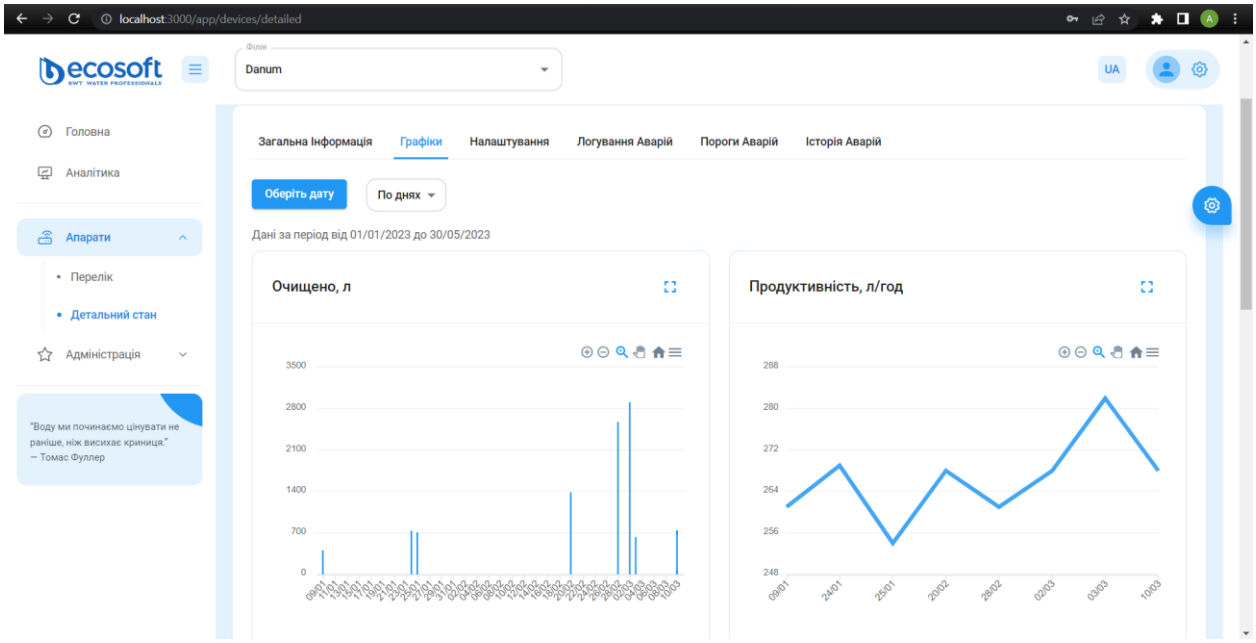


Рисунок 3.15 – Відображення даних апарата у вигляді графіків

Далі натискається таб «Налаштування» на цій сторінці. Відображаються доступні налаштування або повідомлення про те, що їх немає (рисунок 3.16).

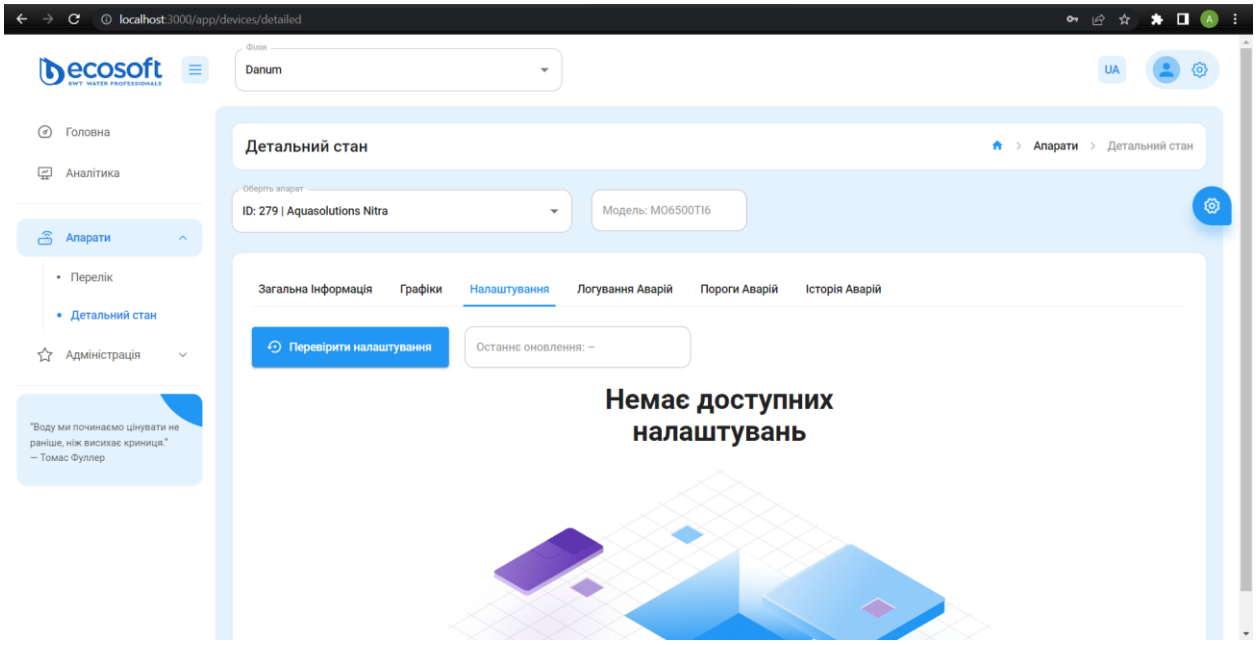


Рисунок 3.16 – Відображення налаштувань апарата

Далі натискається таб «Логування Аварій» на цій сторінці. Відображається список аварій і те, чи вони увімкнені для логування (рисунок 3.17).

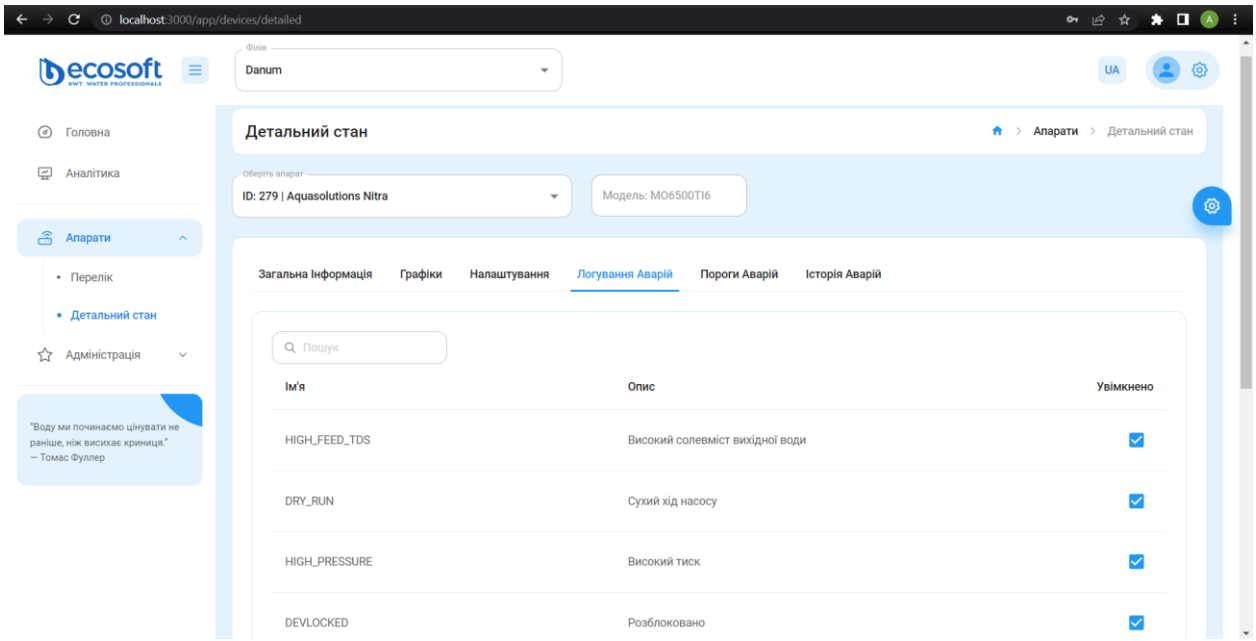


Рисунок 3.17 – Відображення можливих аварій

Далі натискається таб «Пороги Аварій» на цій сторінці. Відображається список порогів аварій і те, чи вони увімкнені (рисунок 3.18).

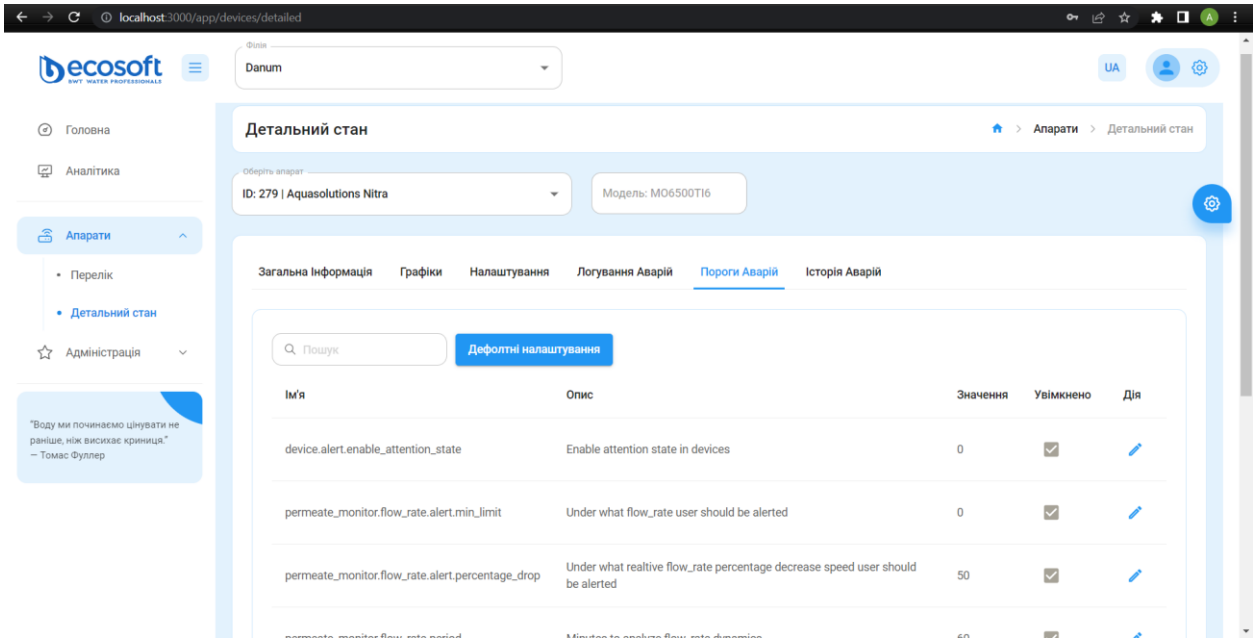


Рисунок 3.18 – Відображення списку порогів аварій

Далі натискається таб «Історія Аварій» на цій сторінці. Натискається кнопка «Оберіть дату», з'являється компонент для вибору проміжку дат, список аварій за який має відобразитися в таблиці. Натискається кнопка «Прийняти». Відображається список аварій даного апарату (рисунок 3.19).

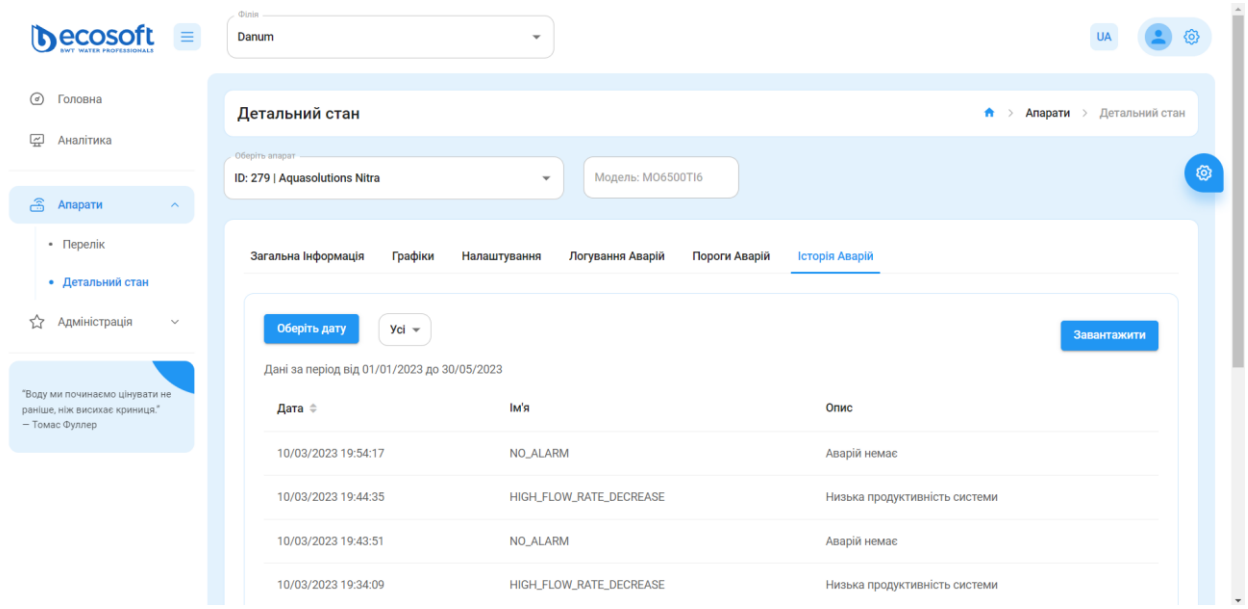


Рисунок 3.19 – Відображення списку аварій, які були в апарата

Далі натискається вкладка «Адміністрація» в боковому меню й вибирається дочірня вкладка «Філії», відбувається перехід на сторінку зі списком філій (рисунок 3.20).

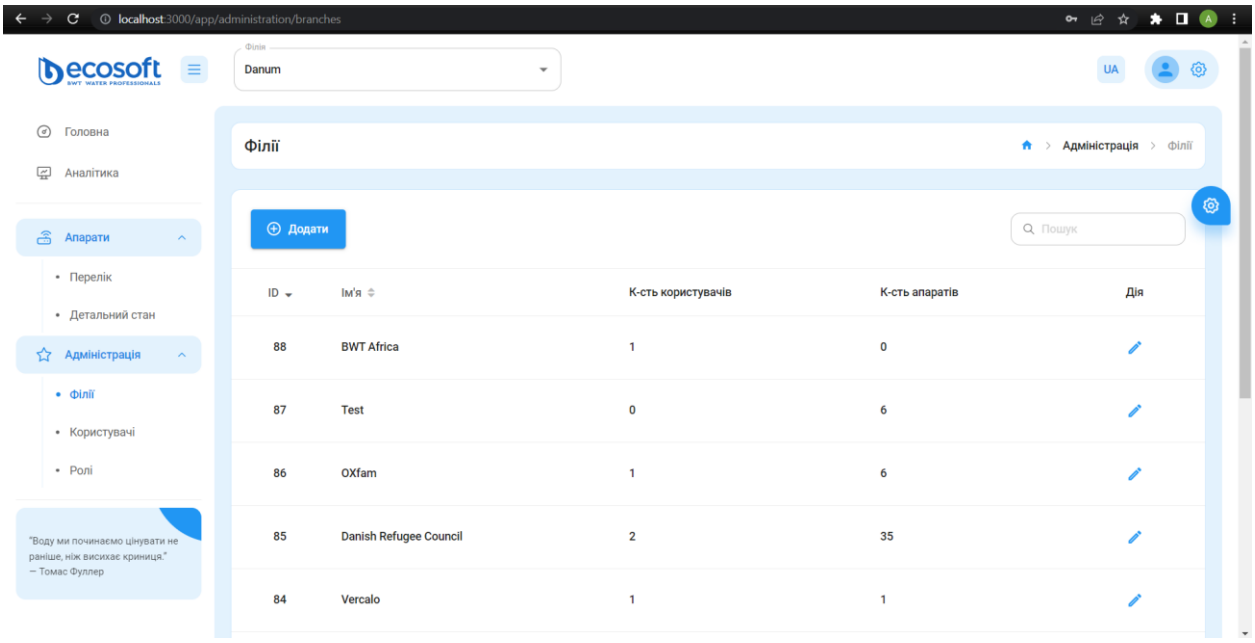


Рисунок 3.20 – Сторінка зі списком філій

Далі вибирається сусідня вкладка «Користувачі». Відбувається перехід на сторінку зі списком користувачів (рисунок 3.21).

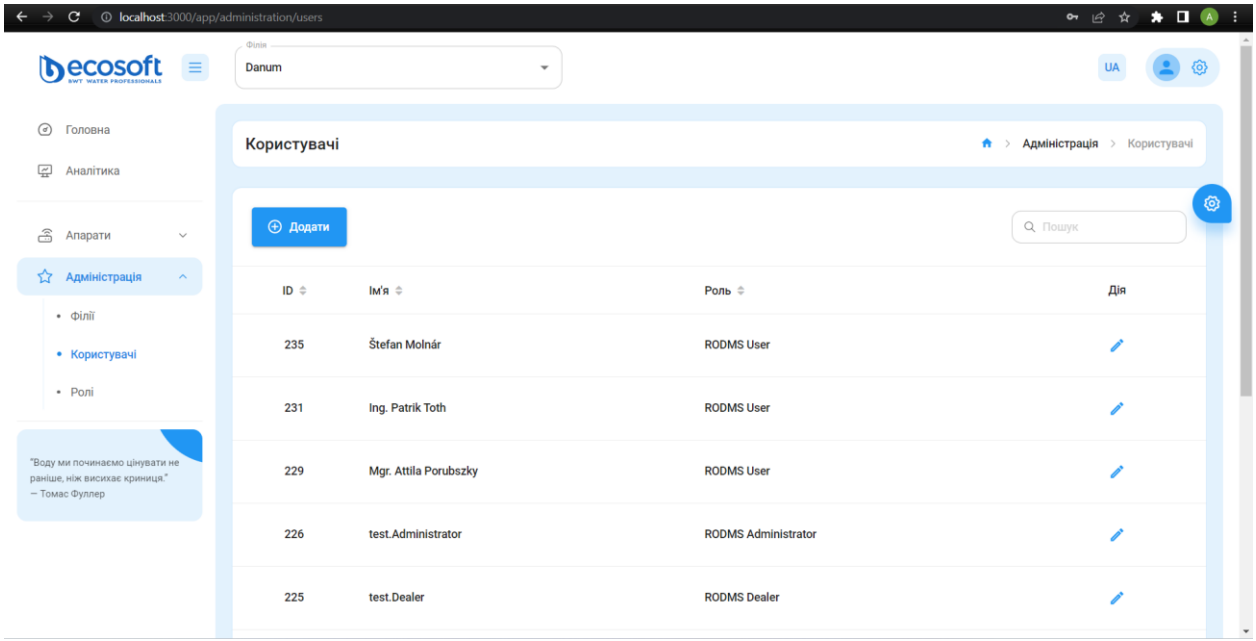


Рисунок 3.21 – Сторінка зі списком користувачів

Висновки до розділу

У даному розділі відбулося тестування веб-застосунку для віддаленого керування та моніторингу роботи систем очищення води. Були обрані метрики якості програмного забезпечення за якими система тестувалася. Тестування відбувалося за допомогою платформи аналітики коду Embold, яка після проведення аналізу коду надає загальний рейтинг у вигляді оцінки від -5 до 5. Програмне забезпечення, що розроблялося в рамках дипломного проєктування, отримало оцінку 2.97. Крім того, відбувся аналіз за кожною з метрик. Найнижчий бал був отриманий за метрикою здатності підтримувати код, хоча він є прийнятним. За всіма іншими метриками: ефективність, надійність, здатність піддаватися аналізу, безпека – бали були добрими або дуже добрими. Після тестування було з'ясовано, що програмне забезпечення виконує такі операційні нефункціональні вимоги, як здатність піддаватися аналізу та здатність підтримувати код.

Також було виконане мануальне тестування програмного забезпечення. Були протестовані такі модулі: вхід у застосунок, відновлення пароля, оновлення персональних даних, зміна пароля, зміна поточної філії, нефункціональні вимоги до інтерфейсу, створення апарата, оновлення апарата, детальний стан апарата, додавання філії, редагування філії, додавання користувача, редагування користувача.

Крім того, було описано контрольний приклад. У якості контрольного прикладу було розглянуто процес моніторингу роботи систем очищення води. Також були показані способи отримання даних, сформованих у результаті моніторингу, та адміністративних даних.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Клієнтську та серверну частини програмного було вирішено розгорнути на платформі Docker Swarm. Для розгортання було використано веб-сервіс для хостингу проєктів на базі систем керування версіями Bitbucket, який надає можливості для постійної інтеграції й розгортання [8].

Розгортання починається, коли новий код застосунку доставляється в репозиторій у гілку master. Тоді в середовищі Bitbucket створюється Docker image за допомогою Dockerfile, що знаходиться в проєкті. Цей image збирається за допомогою інструкцій, написаних у файлі bitbucket-pipelines.yml. За допомогою цього файлу відбувається підміна змінних середовища, будується image (команда "docker build"), відбувається авторизація у віддалений репозиторій, де зберігаються реєстр images (команда "docker login"), та відправляємо збудований image на зберігання (команда "docker push"). Цей image підмінюється в сервісі клієнта або сервера за допомогою команди "docker service update --image". Повністю сервіси розгортаються в Docker Swarm за допомогою команди "docker stack deploy".

На рисунку 4.1 можна побачити інформацію про розгортання клієнтської і серверної частин застосунку. Сервіс «osmos-test_osmos-frontend-react» – це клієнтська частина, а «osmos-test_rodms-api» – серверна. Для обох можна побачити найсвіжіший image, побачити статус та подивитися логи.

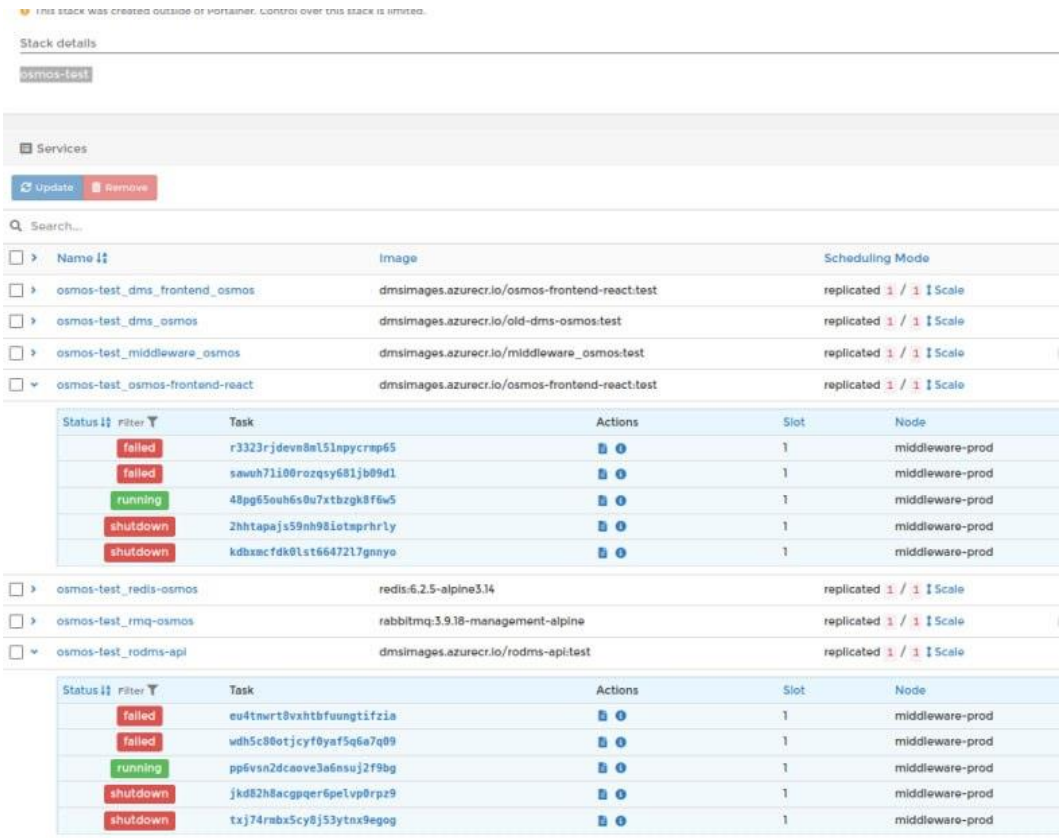


Рисунок 4.1 - Інформація про розгортання програмного забезпечення

4.2 Підтримка програмного забезпечення

І клієнтська, і серверна частини застосунку з часом змінюються: додається новий функціонал, виправляються помилки. І користувач повинен мати змогу отримати актуальну версію застосунку. Для автоматизації цього процесу також може бути використаний веб-сервіс для хостингу проєктів Bitbucket. Кожна нова версія клієнтської та серверної частин зберігається у веб-сервісі Bitbucket за допомогою системи контролю версій Git. Якщо в Bitbucket надсилається нова версія якоїсь із частин застосунку, то Bitbucket за допомогою pipeline (інструмент для безперервної інтеграції та безперервної доставки) стягує оновлений програмний код. Потім виконує дії з файлу bitbucket-pipelines.yml відповідно до гілки, яка була надіслана у віддалений репозиторій. Далі він автоматично викатує image та підмінює попередній image на щойно викачений.

Висновки до розділу

У даному розділі був розглянутий процес впровадження програмного забезпечення. Розгортання застосунку відбувається на платформі Docker Swarm за допомогою веб-сервісу Bitbucket. Був розглянутий процес побудови Docker image за допомогою інструкцій файлу bitbucket-pipelines.yml у клієнтській та серверній частині веб-застосунку. Крім того, було розглянуто процес підтримки програмного забезпечення, який також відбувається за допомогою Bitbucket. Завдяки його інструментам для безперервної інтеграції та безперервної доставки викат image, у якому зберігається нова версія застосунку, відбувається автоматично.

ВИСНОВКИ

У рамках дипломного проєктування було створено веб-застосунок для віддаленого керування та моніторингу роботи систем очищення води.

Спочатку були проаналізовані архітектури програмного забезпечення, архітектурні патерни, програмні засоби й засоби розробки, переваги вибраних мов програмування, бібліотек, фреймворків, текстового редактора та інтегрованого середовища розробки. Далі був проведений аналіз вимог до програмного забезпечення та порівняльний аналіз застосунку з іншим програмним продуктом, який також є системою керування апаратами.

Наступним кроком було спроєктовано систему та представлено архітектуру застосунку. Також було вказано, за допомогою чого реалізована безпека даних у програмному забезпеченні, і як це реалізовано у застосунку.

Далі відбулося тестування веб-застосунку за обраними метриками за допомогою платформи аналітики коду Embold. Програмне забезпечення отримало оцінку 2.97. Крім того, було виконане мануальне тестування програмного забезпечення та описано контрольний приклад, у якості якого було розглянуто процес моніторингу роботи систем очищення води.

У кінці був розглянутий процес розгортання програмного забезпечення, що відбувається на платформі Docker Swarm за допомогою веб-сервісу Bitbucket, а також процес підтримки програмного забезпечення, який теж відбувається за допомогою інструментів цього веб-сервісу.

Програмне забезпечення, розроблене в рамках дипломного проєктування, – це система моніторингу та керування апаратами зворотного осмосу, яка дозволяє швидко отримувати необхідну про апарати інформацію без необхідності знаходитися біля них, а також керувати ними без необхідності виїздів до них сервісних спеціалістів.

Застосунок може бути використаний тими людьми чи підприємствами, які володіють апаратами та бажають швидко й зручно дізнаватися стан систем та керувати ними.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Принцип роботи зворотного осмосу – що таке зворотний осмос, користь чи шкода? [Електронний ресурс]. – Режим доступу : https://modernsys.com.ua/uk/princip-raboty-obratnogo-osmosa-chto-takoe-obratnyy-osmos-polza-ili-vred.html

2) Stack Overflow Developer Survey 2022 [Електронний ресурс]. – Режим доступу : https://survey.stackoverflow.co/2022

3) Chandler Harris. Microservices vs. monolithic architecture [Електронний ресурс]. – Режим доступу : https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith

4) What is an Event-Driven Architecture? [Електронний ресурс]. – Режим доступу : https://aws.amazon.com/event-driven-architecture

5) Електронна енциклопедія Britannica: client-server architecture [Електронний ресурс]. – Режим доступу : https://www.britannica.com/technology/client-server-architecture

6) Vending Device Management System [Електронний ресурс]. – Режим доступу : https://dms.ecosoft.ua

7) Embold [Електронний ресурс]. – Режим доступу : https://embold.io

8) Bitbucket [Електронний ресурс]. – Режим доступу : https://bitbucket.org

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015375930

Дата перевірки:
01.06.2023 19:32:11 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
01.06.2023 19:34:24 EEST

ID користувача:
76913

Назва документа: **ІП-96_Карімов_ПЗ**

Кількість сторінок: **91** Кількість слів: **14136** Кількість символів: **108880** Розмір файлу: **3.52 MB** ID файлу: **1015041720**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

7.7%
Схожість

Найбільша схожість: **3.01%** з джерелом з Бібліотеки (ID файлу: **1015036057**)

1.72% Джерела з Інтернету	96	Сторінка 93
7.58% Джерела з Бібліотеки	245	Сторінка 94

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за **8 слів** та **0%**)

Немає вилучених Інтернет-джерел

0% Вилученого тексту з Бібліотеки	8	Сторінка 94

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	22 сторінки

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ВІДДАЛЕНОГО КЕРУВАННЯ ТА МОНІТОРИНГУ РОБОТИ СИСТЕМ ОЧИЩЕННЯ ВОДИ

Текст програми

КПІ.ІП-9619.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Артем КАРІМОВ

Київ – 2023

Файл App.jsx

```
import React, { useEffect, useState } from 'react'
import { useSelector } from 'react-redux'
import { useRoutes } from 'react-router-dom'

import { ThemeProvider } from '@mui/material/styles'
import { CssBaseline, StyledEngineProvider } from '@mui/material'

import themes from './themes'

import './App.css'
import NavigationScroll from './components/layout/NavigationScroll'
import routes from './routes/routes'
import Locales from './components/helpers/Locales'

const App = () => {
  const [appRoutes, setAppRoutes] = useState(routes(false))
  const customization = useSelector((state) => state.customization)
  const { authUser } = useSelector((state) => state.auth)

  useEffect(() => {
    setAppRoutes(routes(Boolean(authUser), authUser?.authorityIds))
  }, [authUser])

  return (
    <StyledEngineProvider injectFirst>
      <ThemeProvider theme={themes(customization)}>
        <CssBaseline />
        <Locales>
          <NavigationScroll>{useRoutes(appRoutes)}</NavigationScroll>
        </Locales>
      </ThemeProvider>
    </StyledEngineProvider>
  )
}

export default App
```

Файл pages/Administration/Branches/index.jsx

```
import React, { useEffect, useState } from 'react'
import { useDispatch, useSelector } from 'react-redux'
import {
  Button,
  Grid,
  InputAdornment,
  OutlinedInput,
  Pagination,
  Tooltip,
} from '@mui/material'
import { FormattedMessage, injectIntl } from 'react-intl'
import PropTypes from 'prop-types'
import { IconSearch } from '@tabler/icons'
import AddCircleOutlineOutlinedIcon from '@mui/icons-material/AddCircleOutlineOutlined'

import MainCard from 'components/ui/cards/MainCard'
import { gridSpacing } from 'themes/common/constants'
import { calculateOffset, countPagesAmount } from 'helpers/pagination.helper'

import PageSizer from 'components/helpers/PageSizer'
```

```
import { OrderType, PageSize, PaginationEnum } from 'common/enums/enums'

import useDialog from 'hooks/useDialog'
import usePagination from 'hooks/usePagination'
import useSearch from 'hooks/useSearch'
import useSort from 'hooks/useSort'
import { clearSubmitted } from 'store/common/actions'
import * as branchActionCreator from 'store/administration/branches/actions'
import { hasAccess } from 'common/permissions/permissions'
import { HttpMethod } from 'common/enums/http/http-method.enum'
import BranchUpdate from './components/BranchUpdate'
import BranchAdd from './components/BranchAdd'
import BranchList from './components/BranchList'

const Branches = ({ intl }) => {
  const dispatch = useDispatch()
  const { authUser } = useSelector((state) => state.auth)

  const [rows, setRows] = useState([])

  const {
    openAdd,
    openEdit,
    handleOpenAddDialog,
    handleCloseAddDialog,
    handleOpenEditDialog,
    handleCloseEditDialog,
  } = useDialog()

  const {
    anchorEl,
    page,
    pageSize,
    setPage,
    handlePageSizerClick,
    handlePageSizerClose,
    handlePageChange,
  } = usePagination()

  const { submitted } = useSelector((state) => state.common)
  const { branches, branchById } = useSelector((state) => state.branches)

  const { keyword, setKeyword, handleSearch } = useSearch()

  const { orderBy, orderType, handleRequestSort, setOrderBy, setOrderType } =
    useSort('desc', 'id')

  useEffect(() => {
    dispatch(
      branchActionCreator.getBranches({
        limit: PaginationEnum.DEFAULT_LIMIT,
        offset: PaginationEnum.DEFAULT_OFFSET,
        orderBy,
        orderType,
      })
    )
  }, [])

  useEffect(() => {
    if (submitted) {
      setPage(PaginationEnum.DEFAULT_PAGE)
      setOrderBy(null)
      setOrderType(OrderType.NONE)
```

```
    setKeyword(null)

    dispatch(
      branchActionCreator.getBranches({
        limit: PaginationEnum.DEFAULT_LIMIT,
        offset: PaginationEnum.DEFAULT_OFFSET,
        orderBy,
        orderType,
      })
    )

    dispatch(clearSubmitted())
  }
}, [submitted])

useEffect(() => {
  if (branches) {
    setRows(branches.data)
  }
}, [branches])

const handlePageNChange = (p) =>
  handlePageChange(
    p,
    branchActionCreator.getBranches({
      limit: pageSize,
      offset: calculateOffset(p, pageSize),
      keyword,
      orderBy,
      orderType,
    })
  )

const handleSearchByKeyword = (e) => {
  handleSearch(
    e,
    branchActionCreator.getBranches({
      limit: pageSize,
      offset: PaginationEnum.DEFAULT_OFFSET,
      keyword: e.target.value,
      orderBy,
      orderType,
    })
  )
  setPage(PaginationEnum.DEFAULT_PAGE)
}

const handleSort = (property) => {
  handleRequestSort(property, (newOrderBy, newOrderType) =>
    branchActionCreator.getBranches({
      limit: pageSize,
      offset: PaginationEnum.DEFAULT_OFFSET,
      keyword,
      orderBy: newOrderBy,
      orderType: newOrderType,
    })
  )
  setPage(PaginationEnum.DEFAULT_PAGE)
}

const handleSelectPageClose = (ps) =>
  handlePageSizerClose(
    ps,
```

```
    branchActionCreator.getBranches({
      limit: ps,
      offset: PaginationEnum.DEFAULT_OFFSET,
      keyword,
      orderBy,
      orderType,
    })
  )

const handleEdit = (id) => {
  dispatch(branchActionCreator.getBranch(id))
  handleOpenEditDialog()
}

return (
  <MainCard
    title={
      <Grid
        container
        alignItems="center"
        justifyContent="space-between"
        spacing={gridSpacing}
      >
        {hasAccess(HttpMethod.POST, '/branches', authUser?.authorityIds) && (
          <>
            <Grid item>
              <Tooltip
                title={<FormattedMessage id="add" defaultMessage="Add" />}
              >
                <Button
                  variant="contained"
                  size="medium"
                  startIcon={<AddCircleOutlineOutlinedIcon />}
                  sx={{ px: 2.75, py: 1.5 }}
                  onClick={handleOpenAddDialog}
                >
                  <FormattedMessage id="add" defaultMessage="Add" />
                </Button>
              </Tooltip>
              <BranchAdd
                open={openAdd}
                handleCloseDialog={handleCloseAddDialog}
              />
            </Grid>
          </>
        )}
        <Grid item>
          <OutlinedInput
            id="input-search-list"
            placeholder={intl.formatMessage({ id: 'search' })}
            onChange={handleSearchByKeyword}
            startAdornment={
              <InputAdornment position="start">
                <IconSearch stroke={1.5} size="1rem" />
              </InputAdornment>
            }
            size="small"
          />
        </Grid>
      </Grid>
    }
    content={false}
  >
```

```
      <BranchList
        rows={rows}
        orderBy={orderBy}
        orderType={orderType}
        handleRequestSort={handleSort}
        handleOpenDialog={handleEdit}
      />
      {branchById && (
        <BranchUpdate
          branch={branchById || {}}
          open={openEdit}
          handleCloseDialog={handleCloseEditDialog}
        />
      )}
      <Grid item xs={12} sx={{ p: 3 }}>
        <Grid container justifyContent="space-between" spacing={gridSpacing}>
          <Grid item>
            <Pagination
              count={branches ? countPagesAmount(branches.total, pageSize) : 1}
              onChange={(_, p) => handlePageNChange(p)}
              page={page}
              color="primary"
            />
          </Grid>
          <Grid item>
            <PageSizer
              anchorEl={anchorEl}
              pageSize={pageSize}
              handleClick={handlePageSizerClick}
              handleClose={handleSelectPageClose}
              sizeOptions={[
                PageSize.SIZE_10,
                PageSize.SIZE_20,
                PageSize.SIZE_50,
              ]}
            />
          </Grid>
        </Grid>
      </Grid>
    </MainCard>
  )
}

/* eslint-disable react/forbid-prop-types */
Branches.propTypes = {
  intl: PropTypes.object.isRequired,
}

export default injectIntl(Branches)
```

Файл pages/Administration/Roles/index.jsx

```
import React, { useEffect, useState } from 'react'
import { useDispatch, useSelector } from 'react-redux'
import PropTypes from 'prop-types'
import { Grid, InputAdornment, OutlinedInput, Pagination } from '@mui/material'
import { IconSearch } from '@tabler/icons'
```

```
import { injectIntl } from 'react-intl'

import MainCard from 'components/ui/cards/MainCard'
import { gridSpacing } from 'themes/common/constants'
import * as rolesActionCreator from 'store/administration/roles/actions'
import { calculateOffset, countPagesAmount } from 'helpers/pagination.helper'
import PageSizer from 'components/helpers/PageSizer'
import { PageSize, PaginationEnum } from 'common/enums/enums'

import usePagination from 'hooks/usePagination'
import useSearch from 'hooks/useSearch'
import useSort from 'hooks/useSort'
import RoleList from './components/RoleList'

const Roles = ({ intl }) => {
  const dispatch = useDispatch()

  const [rows, setRows] = useState([])

  const {
    anchorEl,
    page,
    pageSize,
    setPage,
    handlePageSizerClick,
    handlePageSizerClose,
    handlePageChange,
  } = usePagination()

  const { keyword, handleSearch } = useSearch()

  const { orderBy, orderType, handleRequestSort } = useSort()

  const { roles } = useSelector((state) => state.roles)

  useEffect(() => {
    dispatch(
      rolesActionCreator.getRoles({
        limit: PaginationEnum.DEFAULT_LIMIT,
        offset: PaginationEnum.DEFAULT_OFFSET,
      })
    )
```

```
}, [])

useEffect(() => {
  if (roles) {
    setRows(roles.data)
  }
}, [roles])

const handlePageNChange = (p) =>
  handlePageChange(
    p,
    rolesActionCreator.getRoles({
      limit: pageSize,
      offset: calculateOffset(p, pageSize),
      keyword,
      orderBy,
      orderType,
    })
  )

const handleSort = (property) => {
  handleRequestSort(property, (newOrderBy, newOrderType) =>
    rolesActionCreator.getRoles({
      limit: pageSize,
      offset: PaginationEnum.DEFAULT_OFFSET,
      keyword,
      orderBy: newOrderBy,
      orderType: newOrderType,
    })
  )
  setPage(PaginationEnum.DEFAULT_PAGE)
}

const handleSearchByKeyword = (e) => {
  handleSearch(
    e,
    rolesActionCreator.getRoles({
      limit: pageSize,
      offset: PaginationEnum.DEFAULT_OFFSET,
      keyword: e.target.value,
      orderBy,
      orderType,
```

```
    })
  )
  setPage(PaginationEnum.DEFAULT_PAGE)
}

const handleSelectPageClose = (ps) =>
  handlePageSizerClose(
    ps,
    rolesActionCreator.getRoles({
      limit: ps,
      offset: PaginationEnum.DEFAULT_OFFSET,
      keyword,
      orderBy,
      orderType,
    })
  )

return (
  <MainCard
    title={
      <Grid
        container
        alignItems="center"
        justifyContent="space-between"
        spacing={gridSpacing}
      >
        <Grid item>
          <OutlinedInput
            id="input-search-list"
            placeholder={intl.formatMessage({ id: 'search' })}
            onChange={handleSearchByKeyword}
            startAdornment={
              <InputAdornment position="start">
                <IconSearch stroke={1.5} size="1rem" />
              </InputAdornment>
            }
            size="small"
          />
        </Grid>
      </Grid>
    }
    content={false}
```

```
    >
      <RoleList
        rows={rows}
        orderBy={orderBy}
        orderType={orderType}
        handleRequestSort={handleSort}
      />
      <Grid item xs={12} sx={{ p: 3 }}>
        <Grid container justifyContent="space-between" spacing={gridSpacing}>
          <Grid item>
            <Pagination
              count={roles ? countPagesAmount(roles.total, pageSize) : 1}
              onChange={(_, p) => handlePageNChange(p)}
              page={page}
              color="primary"
            />
          </Grid>
          <Grid item>
            <PageSizer
              anchorEl={anchorEl}
              pageSize={pageSize}
              handleClick={handlePageSizerClick}
              handleClose={handleSelectPageClose}
              sizeOptions={[
                PageSize.SIZE_10,
                PageSize.SIZE_20,
                PageSize.SIZE_50,
              ]}
            />
          </Grid>
        </Grid>
      </Grid>
    </MainCard>
  )
}

/* eslint-disable react/forbid-prop-types */
Roles.propTypes = {
  intl: PropTypes.object.isRequired,
}

export default injectIntl(Roles)
```

Файл pages/Administration/Users/index.jsx

```
import React, { useEffect, useState } from 'react'
import { useDispatch, useSelector } from 'react-redux'
import PropTypes from 'prop-types'
import {
  Button,
  Grid,
  InputAdornment,
  OutlinedInput,
  Pagination,
  Tooltip,
} from '@mui/material'
import { IconSearch } from '@tabler/icons'
import AddCircleOutlineOutlinedIcon from '@mui/icons-material/AddCircleOutlineOutlined'
import { FormattedMessage, injectIntl } from 'react-intl'

import MainCard from 'components/ui/cards/MainCard'
import { gridSpacing } from 'themes/common/constants'
import { calculateOffset, countPagesAmount } from 'helpers/pagination.helper'
import * as usersActionCreator from 'store/administration/users/actions'
import PageSizer from 'components/helpers/PageSizer'
import { OrderType, PageSize, PaginationEnum } from 'common/enums/enums'

import { clearSubmitted } from 'store/common/actions'
import useDialog from 'hooks/useDialog'
import usePagination from 'hooks/usePagination'
import useSearch from 'hooks/useSearch'
import useSort from 'hooks/useSort'
import { hasAccess } from 'common/permissions/permissions'
import { HttpMethod } from 'common/enums/http/http-method.enum'
import UserList from './components/UserList'
import UserAdd from './components/UserAdd'
import UserUpdate from './components/UserUpdate'

const Users = ({ intl }) => {
  const dispatch = useDispatch()
  const { authUser } = useSelector((state) => state.auth)

  const [rows, setRows] = useState([])

  const {
```

```
  openAdd,
  openEdit,
  handleOpenAddDialog,
  handleCloseAddDialog,
  handleOpenEditDialog,
  handleCloseEditDialog,
} = useDialog()

const {
  anchorEl,
  page,
  pageSize,
  setPage,
  handlePageSizerClick,
  handlePageSizerClose,
  handlePageChange,
} = usePagination()

const { keyword, setKeyword, handleSearch } = useSearch()

const { orderBy, setOrderBy, orderType, setOrderType, handleRequestSort } =
  useSort()

const { submitted } = useSelector((state) => state.common)
const { users, userById } = useSelector((state) => state.users)
const {
  selectedBranch: { id: branchId },
} = useSelector((state) => state.header)

useEffect(() => {
  dispatch(
    usersActionCreator.getUsers({
      branchId,
      limit: PaginationEnum.DEFAULT_LIMIT,
      offset: PaginationEnum.DEFAULT_OFFSET,
    })
  )
}, [branchId])

useEffect(() => {
  if (submitted) {
    setPage(PaginationEnum.DEFAULT_PAGE)
```

```
    setOrderBy(null)
    setOrderType(OrderType.NONE)
    setKeyword(null)

    dispatch(
      usersActionCreator.getUsers({
        branchId,
        limit: PaginationEnum.DEFAULT_LIMIT,
        offset: PaginationEnum.DEFAULT_OFFSET,
      })
    )

    dispatch(clearSubmitted())
  }
}, [submitted])

useEffect(() => {
  if (users) {
    setRows(users.data)
  }
}, [users])

useEffect(() => {
  if (!openEdit) {
    dispatch(usersActionCreator.clearUserById())
  }
}, [openEdit])

const handlePageNChange = (p) =>
  handlePageChange(
    p,
    usersActionCreator.getUsers({
      branchId,
      limit: pageSize,
      offset: calculateOffset(p, pageSize),
      keyword,
      orderBy,
      orderType,
    })
  )

const handleSearchByKeyword = (e) => {
```

```
  handleSearch(
    e,
    usersActionCreator.getUsers({
      branchId,
      limit: pageSize,
      offset: PaginationEnum.DEFAULT_OFFSET,
      keyword: e.target.value,
      orderBy,
      orderType,
    })
  )
  setPage(PaginationEnum.DEFAULT_PAGE)
}

const handleSort = (property) => {
  handleRequestSort(property, (newOrderBy, newOrderType) =>
    usersActionCreator.getUsers({
      branchId,
      limit: pageSize,
      offset: PaginationEnum.DEFAULT_OFFSET,
      keyword,
      orderBy: newOrderBy,
      orderType: newOrderType,
    })
  )
  setPage(PaginationEnum.DEFAULT_PAGE)
}

const handleSelectPageClose = (ps) =>
  handlePageSizerClose(
    ps,
    usersActionCreator.getUsers({
      branchId,
      limit: ps,
      offset: PaginationEnum.DEFAULT_OFFSET,
      keyword,
      orderBy,
      orderType,
    })
  )

const handleEdit = (id) => {
```

```
  dispatch(usersActionCreator.getUser(id))
  handleOpenEditDialog()
 }

 return (
  <MainCard
   title={
    <Grid
     container
     alignItems="center"
     justifyContent="space-between"
     spacing={gridSpacing}
    >
     {hasAccess(HttpMethod.POST, '/users', authUser?.authorityIds) && (
      <Grid item>
       <Tooltip
        title={<FormattedMessage id="add" defaultMessage="Add" />}
       >
        <Button
         variant="contained"
         size="medium"
         startIcon={<AddCircleOutlineOutlinedIcon />}
         sx={{ px: 2.75, py: 1.5 }}
         onClick={handleOpenAddDialog}
        >
         <FormattedMessage id="add" defaultMessage="Add" />
        </Button>
       </Tooltip>
       {openAdd && (
        <UserAdd
         open={openAdd}
         handleCloseDialog={handleCloseAddDialog}
        />
       )}
      </Grid>
     )}
     <Grid item>
      <OutlinedInput
       id="input-search-list"
       placeholder={intl.formatMessage({ id: 'search' })}
       onChange={handleSearchByKeyword}
       startAdornment={
```

```
            <InputAdornment position="start">
              <IconSearch stroke={1.5} size="1rem" />
            </InputAdornment>
          }
          size="small"
        />
      </Grid>
    </Grid>
  }
  content={false}
>
  <UserList
    rows={rows}
    orderBy={orderBy}
    orderType={orderType}
    handleRequestSort={handleSort}
    handleOpenDialog={handleEdit}
  />
  {openEdit && userById && (
    <UserUpdate
      user={userById}
      open={openEdit}
      handleCloseDialog={handleCloseEditDialog}
    />
  )}
  <Grid item xs={12} sx={{ p: 3 }}>
    <Grid container justifyContent="space-between" spacing={gridSpacing}>
      <Grid item>
        <Pagination
          count={users ? countPagesAmount(users.total, pageSize) : 1}
          onChange={(_, p) => handlePageNChange(p)}
          page={page}
          color="primary"
        />
      </Grid>
      <Grid item>
        <PageSizer
          anchorEl={anchorEl}
          pageSize={pageSize}
          handleClick={handlePageSizerClick}
          handleClose={handleSelectPageClose}
          sizeOptions={[
```

```
              PageSize.SIZE_10,
              PageSize.SIZE_20,
              PageSize.SIZE_50,
            ]}
          />
        </Grid>
      </Grid>
    </Grid>
  </MainCard>
 )
}

/* eslint-disable react/forbid-prop-types */
Users.propTypes = {
 intl: PropTypes.object.isRequired,
}

export default injectIntl(Users)
```

Файл pages/Analytics/index.jsx

```
import React, { useEffect, useMemo } from 'react'
import { Grid } from '@mui/material'
import { gridSpacing } from 'themes/common/constants'
import { FunnelChart } from 'react-funnel-pipeline'

import 'react-funnel-pipeline/dist/index.css'
import MainCard from 'components/ui/cards/MainCard'
import { useDispatch, useSelector } from 'react-redux'
import { analyticsActionCreator } from 'store/actions'
import i18next from 'i18next'
import useWindowDimensions from '../../hooks/useWindowDimensions'

const backgroundImages = [
 'linear-gradient(to bottom, #5196dc, #4189d5)',
 'linear-gradient(to bottom, #4189d5, #327dce)',
 'linear-gradient(to bottom, #327dce, #2572c8)',
 'linear-gradient(to bottom, #2572c8, #1565c0)',
]

const Chart = ({ locale, data, height }) =>
 useMemo(
```

```
    () => (
      <FunnelChart
        title={i18next.t('system_connection_statistics')}
        getRowNameStyle={() => ({
          fontSize: 12,
          position: 'relative',
          zIndex: 999,
        })}
        chartHeight={height}
        getRowStyle={({ backgroundImage }) => ({
          height: 90,
          backgroundImage,
        })}
        data={data?.map(({ label, count, backgroundImage }) => ({
          name: i18next.t(label.toLowerCase()),
          value: count,
          backgroundImage,
        }))}
      />
    ),
    [data, locale, height]
  )

  const Analytics = () => {
    const dispatch = useDispatch()

    const { locale } = useSelector((state) => state.customization)
    const { deviceStats } = useSelector((state) => state.analytics)

    const chartData = deviceStats?.map((value, index) => ({
      ...value,
      backgroundImage: backgroundImages[index],
    }))

    const { height } = useWindowDimensions()

    useEffect(() => {
      dispatch(analyticsActionCreator.getDeviceStatsByStatus())
    }, [])

    return (
      <Grid container spacing={gridSpacing}>
```

```
      <Grid item xs={12} md={12} lg={12}>
        <MainCard
          content={false}
          sx={{
            paddingTop: 5,
            paddingBottom: 5,
            height: height > 700 ? `${height - 200}px` : `${height - 100}px`,
            display: 'flex',
            justifyContent: 'center',
            alignItems: 'center',
          }}
        >
          <Grid item xs={12} lg={9} md={12}>
            {deviceStats && locale && (
              <Chart data={chartData} locale={locale} height={height} />
            )}
          </Grid>
        </MainCard>
      </Grid>
    </Grid>
  )
}

export default Analytics
```

Файл pages/Auth/Login/index.jsx

```
import React from 'react'
import { Link } from 'react-router-dom'

import { useTheme } from '@mui/material/styles'
import { Grid, Stack, Typography, useMediaQuery } from '@mui/material'

import { FormattedMessage } from 'react-intl'

import Logo from 'components/ui/Logo'
import AuthFooter from 'components/ui/cards/AuthFooter'
import LoginForm from './forms/LoginForm'
import AuthWrapper from './components/AuthWrapper'
import AuthCardWrapper from './components/AuthCardWrapper'

const Login = () => {
```

```
const theme = useTheme()
const matchDownSM = useMediaQuery(theme.breakpoints.down('md'))

return (
  <AuthWrapper>
    <Grid
      container
      direction="column"
      justifyContent="flex-end"
      sx={{ minHeight: '100vh' }}
    >
      <Grid item xs={12}>
        <Grid
          container
          justifyContent="center"
          alignItems="center"
          sx={{ minHeight: 'calc(100vh - 68px)' }}
        >
          <Grid item sx={{ m: { xs: 1, sm: 3 }, mb: 0 }}>
            <AuthCardWrapper>
              <Grid
                container
                spacing={2}
                alignItems="center"
                justifyContent="center"
              >
                <Grid item sx={{ mb: 3 }}>
                  <Link to="/login">
                    <Logo />
                  </Link>
                </Grid>
                <Grid item xs={12}>
                  <Grid
                    container
                    direction={matchDownSM ? 'column-reverse' : 'row'}
                    alignItems="center"
                    justifyContent="center"
                  >
                    <Grid item>
                      <Stack
                        alignItems="center"
                        justifyContent="center"
```

```
                spacing={1}
               >
                <Typography
                 color={theme.palette.primary.dark}
                 gutterBottom
                 variant={matchDownSM ? 'h3' : 'h2'}
                >
                 <FormattedMessage id="hi" defaultMessage="Hi" />,{' '}
                 <FormattedMessage
                  id="welcome_back"
                  defaultMessage="Welcome back"
                 />
                </Typography>
                <Typography
                 variant="caption"
                 fontSize="16px"
                 textAlign={matchDownSM ? 'center' : 'inherit'}
                >
                 <FormattedMessage
                  id="enter_credentials"
                  defaultMessage="Enter your credentials to continue"
                 />
                </Typography>
               </Stack>
              </Grid>
             </Grid>
            </Grid>
            <Grid item xs={12}>
             <LoginForm />
            </Grid>
           </Grid>
          </AuthCardWrapper>
         </Grid>
        </Grid>
       </Grid>
       <Grid item xs={12} sx={{ m: 3, mt: 1 }}>
        <AuthFooter />
       </Grid>
      </Grid>
     </AuthWrapper>
    )
}
```

```
export default Login
```

Файл pages/Auth/PasswordReset/index.jsx

```
import React from 'react'
import { useSelector } from 'react-redux'
import { Link } from 'react-router-dom'
import { useTheme } from '@mui/material/styles'
import { Grid, Typography, useMediaQuery } from '@mui/material'
import { FormattedMessage } from 'react-intl'

import Logo from 'components/ui/Logo'
import AuthFooter from 'components/ui/cards/AuthFooter'
import AuthWrapper from './components/AuthWrapper'
import AuthCardWrapper from './components/AuthCardWrapper'
import PasswordResetForm from './forms/PasswordResetForm'

const PasswordReset = () => {
  const theme = useTheme()
  const matchDownSM = useMediaQuery(theme.breakpoints.down('md'))

  const { codeIsSent } = useSelector((state) => state.auth)

  return (
    <AuthWrapper>
      <Grid
        container
        direction="column"
        justifyContent="flex-end"
        sx={{ minHeight: '100vh' }}
      >
        <Grid item xs={12}>
          <Grid
            container
            justifyContent="center"
            alignItems="center"
            sx={{ minHeight: 'calc(100vh - 68px)' }}
          >
            <Grid item sx={{ m: { xs: 1, sm: 3 }, mb: 0 }}>
              <AuthCardWrapper>
                <Grid
```

```
            container
            spacing={2}
            alignItems="center"
            justifyContent="center"
          >
            <Grid item sx={{ mb: 3 }}>
              <Link to="/login">
                <Logo />
              </Link>
            </Grid>
            {codeIsSent ? (
              <Typography
                variant="caption"
                fontSize="14px"
                marginLeft="20px"
                textAlign={matchDownSM ? 'center' : 'inherit'}
              >
                <FormattedMessage
                  id="reset_password_message"
                  defaultMessage="Enter code from email and new password."
                />
              </Typography>
            ) : (
              <Typography
                variant="caption"
                fontSize="14px"
                marginLeft="20px"
                textAlign={matchDownSM ? 'center' : 'inherit'}
              >
                <FormattedMessage
                  id="reset_code_message"
                  defaultMessage="You will receive a code to change your password via email. In addition to the
'Inbox' folder, check also 'Spam' folder."
                />
              </Typography>
            )}
            <Grid item xs={12} style={{ paddingTop: 8 }}>
              <PasswordResetForm />
            </Grid>
          </Grid>
        </AuthCardWrapper>
      </Grid>
```

```
          </Grid>
        </Grid>
        <Grid item xs={12} sx={{ m: 3, mt: 1 }}>
          <AuthFooter />
        </Grid>
      </Grid>
    </AuthWrapper>
  )
}

export default PasswordReset
```

Файл pages/Dashboard/index.jsx

```
import React, { useEffect, useState } from 'react'
import { Grid, Pagination } from '@mui/material'
import { gridSpacing } from 'themes/common/constants'
import { useDispatch, useSelector } from 'react-redux'
import { injectIntl } from 'react-intl'
import * as dashboardActionCreator from 'store/dashboard/actions'
import PropTypes from 'prop-types'

import { PaginationEnum } from 'common/enums/app/pagination.enum'
import { calculateOffset, countPagesAmount } from 'helpers/pagination.helper'
import PageSizer from 'components/helpers/PageSizer'
import { PageSize } from 'common/enums/app/page-size.enum'
import usePagination from 'hooks/usePagination'
import useSort from 'hooks/useSort'
import MainCard from 'components/ui/cards/MainCard'

import ClickableCard from './components/TodayPurifiedCard'
import DashboardDeviceList from './components/DashboardDeviceList'

const Dashboard = ({ intl }) => {
  const dispatch = useDispatch()
  const [active, setActive] = useState('TOTAL')

  const [rows, setRows] = useState([])

  const {
    selectedBranch: { id: branchId },
  } = useSelector((state) => state.header)
```

```
const { volumeUnit } = useSelector((state) => state.customization)

const { branchStats, branchDevices } = useSelector((state) => state.dashboard)

const {
  anchorEl,
  page,
  pageSize,
  setPage,
  handlePageSizerClick,
  handlePageSizerClose,
  handlePageChange,
} = usePagination()

const { orderBy, orderType, handleRequestSort } = useSort()

useEffect(() => {
  dispatch(dashboardActionCreator.getBranchStats(branchId))
}, [branchId])

useEffect(() => {
  if (branchDevices) {
    setRows(branchDevices.data)
  }
}, [branchDevices])

useEffect(() => {
  setPage(PaginationEnum.DEFAULT_PAGE)
  if (active === 'TOTAL') {
    dispatch(
      dashboardActionCreator.getBranchDevicesByState({
        id: branchId,
        params: {
          limit: PaginationEnum.DEFAULT_LIMIT,
          offset: PaginationEnum.DEFAULT_OFFSET,
          volumeUnit,
        },
      })
    )
  } else {
    dispatch(
```

```
      dashboardActionCreator.getBranchDevicesByState({
        id: branchId,
        state: active,
        params: {
          limit: PaginationEnum.DEFAULT_LIMIT,
          offset: PaginationEnum.DEFAULT_OFFSET,
          volumeUnit,
        },
      })
    )
  }
}, [branchId, active, volumeUnit])

const handlePageNChange = (p) => {
  if (active === 'TOTAL') {
    handlePageChange(
      p,
      dashboardActionCreator.getBranchDevicesByState({
        id: branchId,
        params: {
          limit: pageSize,
          offset: calculateOffset(p, pageSize),
          volumeUnit,
          orderBy,
          orderType,
        },
      })
    )
  } else {
    handlePageChange(
      p,
      dashboardActionCreator.getBranchDevicesByState({
        id: branchId,
        state: active,
        params: {
          limit: pageSize,
          offset: calculateOffset(p, pageSize),
          volumeUnit,
          orderBy,
          orderType,
        },
      })
```

```
    )
  }
}

const handleSelectPageClose = (ps) => {
  if (active === 'TOTAL') {
    handlePageSizerClose(
      ps,
      dashboardActionCreator.getBranchDevicesByState({
        id: branchId,
        params: {
          limit: ps,
          offset: PaginationEnum.DEFAULT_OFFSET,
          volumeUnit,
          orderBy,
          orderType,
        },
      })
    )
  } else {
    handlePageSizerClose(
      ps,
      dashboardActionCreator.getBranchDevicesByState({
        id: branchId,
        state: active,
        params: {
          limit: ps,
          offset: PaginationEnum.DEFAULT_OFFSET,
          volumeUnit,
          orderBy,
          orderType,
        },
      })
    )
  }
}

const handleSort = (property) => {
  handleRequestSort(property, (newOrderBy, newOrderType) => {
    if (active === 'TOTAL') {
      dispatch(
        dashboardActionCreator.getBranchDevicesByState({
```

```
          id: branchId,
          params: {
            limit: pageSize,
            offset: calculateOffset(page, pageSize),
            volumeUnit,
            orderBy: newOrderBy,
            orderType: newOrderType,
          },
        })
      )
    } else {
      dispatch(
        dashboardActionCreator.getBranchDevicesByState({
          id: branchId,
          state: active,
          params: {
            limit: pageSize,
            offset: calculateOffset(page, pageSize),
            volumeUnit,
            orderBy: newOrderBy,
            orderType: newOrderType,
          },
        })
      )
    }
  })
}

return (
  <Grid container spacing={gridSpacing}>
    <Grid item xs={12}>
      {branchStats && (
        <Grid container spacing={gridSpacing}>
          {branchStats.map(({ label, count }) => (
            <Grid key={label} item lg={3} md={4} sm={4} xs={6}>
              <ClickableCard
                isLoading={false}
                isActive={active === label}
                label={intl.formatMessage({ id: label.toLowerCase() })}
                count={count}
                onClick={() => setActive(label)}
              />
```

```
          </Grid>
        ))}
      </Grid>
    )}
  </Grid>
  <Grid item xs={12}>
    <MainCard content={false}>
      <DashboardDeviceList
        rows={rows}
        orderBy={orderBy}
        orderType={orderType}
        handleRequestSort={handleSort}
      />
      <Grid item xs={12} sx={{ p: 3 }}>
        <Grid
          container
          justifyContent="space-between"
          spacing={gridSpacing}
        >
          <Grid item>
            <Pagination
              count={
                branchDevices
                  ? countPagesAmount(branchDevices.total, pageSize)
                  : 1
              }
              onChange={(_, p) => handlePageNChange(p)}
              page={page}
              color="primary"
            />
          </Grid>
          <Grid item>
            <PageSizer
              anchorEl={anchorEl}
              pageSize={pageSize}
              handleClick={handlePageSizerClick}
              handleClose={handleSelectPageClose}
              sizeOptions={[
                PageSize.SIZE_10,
                PageSize.SIZE_20,
                PageSize.SIZE_50,
              ]}
```

```
            />
          </Grid>
        </Grid>
      </Grid>
    </MainCard>
  </Grid>
 </Grid>
 )
}

/* eslint-disable react/forbid-prop-types */
Dashboard.propTypes = {
  intl: PropTypes.object.isRequired,
}

export default injectIntl(Dashboard)
```

Файл pages/Devices/List/index.jsx

```
import React, { useEffect, useState } from 'react'
import {
  Button,
  Grid,
  InputAdornment,
  OutlinedInput,
  Pagination,
  Tooltip,
} from '@mui/material'
import { IconSearch } from '@tabler/icons'
import AddCircleOutlineOutlinedIcon from '@mui/icons-material/AddCircleOutlineOutlined'
import AddLinkIcon from '@mui/icons-material/AddLink'

import MainCard from 'components/ui/cards/MainCard'
import { gridSpacing } from 'themes/common/constants'
import { calculateOffset, countPagesAmount } from 'helpers/pagination.helper'
import * as devicesActionCreator from 'store/devices/list/actions'
import PageSizer from 'components/helpers/PageSizer'
import { OrderType, PageSize, PaginationEnum } from 'common/enums/enums'
import { useDispatch, useSelector } from 'react-redux'
import PropTypes from 'prop-types'
import { FormattedMessage, injectIntl } from 'react-intl'
import useDialog from 'hooks/useDialog'
```

```
import usePagination from 'hooks/usePagination'
import useSearch from 'hooks/useSearch'
import useSort from 'hooks/useSort'
import { clearSubmitted } from 'store/common/actions'
import { hasAccess } from 'common/permissions/permissions'
import { HttpMethod } from 'common/enums/http/http'
import DeviceList from './components/DeviceList'
import DeviceAdd from './components/DeviceAdd'
import DeviceUpdate from './components/DeviceUpdate'
import DeviceCreate from './components/DeviceCreate'
import DeviceUpdateCreated from './components/DeviceUpdateCreated'

const List = ({ intl }) => {
  const dispatch = useDispatch()

  const [rows, setRows] = useState([])

  const {
    openAdd,
    openEdit,
    handleOpenAddDialog,
    handleCloseAddDialog,
    handleOpenEditDialog,
    handleCloseEditDialog,
  } = useDialog()

  const {
    openAdd: openCreate,
    openEdit: openEditCreated,
    handleOpenAddDialog: handleOpenCreateDialog,
    handleCloseAddDialog: handleCloseCreateDialog,
    handleOpenEditDialog: handleOpenEditCreatedDialog,
    handleCloseEditDialog: handleCloseEditCreatedDialog,
  } = useDialog()

  const {
    anchorEl,
    page,
    pageSize,
    setPage,
    handlePageSizerClick,
    handlePageSizerClose,
```

```
  handlePageChange,
} = usePagination()

const { volumeUnit } = useSelector((state) => state.customization)
const { authUser } = useSelector((state) => state.auth)

const { keyword, setKeyword, handleSearch } = useSearch()

const { orderBy, setOrderBy, orderType, setOrderType, handleRequestSort } =
  useSort()

const { devices, deviceById } = useSelector((state) => state.devices)
const { submitted } = useSelector((state) => state.common)
const {
  selectedBranch: { id: branchId },
} = useSelector((state) => state.header)

useEffect(() => {
  dispatch(
    devicesActionCreator.getDevices({
      branchId,
      params: {
        limit: PaginationEnum.DEFAULT_LIMIT,
        offset: PaginationEnum.DEFAULT_OFFSET,
        volumeUnit,
      },
    })
  )
}, [branchId, volumeUnit])

useEffect(() => {
  if (!openEditCreated) {
    dispatch(devicesActionCreator.clearDeviceById())
  }
}, [openEditCreated, openEdit])

useEffect(() => {
  if (submitted) {
    setPage(PaginationEnum.DEFAULT_PAGE)
    setOrderBy(null)
    setOrderType(OrderType.NONE)
    setKeyword(null)
```

```
    dispatch(
      devicesActionCreator.getDevices({
        branchId,
        params: {
          limit: PaginationEnum.DEFAULT_LIMIT,
          offset: PaginationEnum.DEFAULT_OFFSET,
          volumeUnit,
        },
      })
    )

    dispatch(clearSubmitted())
  }
}, [submitted])

useEffect(() => {
  if (devices) {
    setRows(devices.data)
  }
}, [devices])

const handlePageNChange = (p) =>
  handlePageChange(
    p,
    devicesActionCreator.getDevices({
      branchId,
      params: {
        limit: pageSize,
        offset: calculateOffset(p, pageSize),
        keyword,
        orderBy,
        orderType,
        volumeUnit,
      },
    })
  )

const handleSort = (property) => {
  handleRequestSort(property, (newOrderBy, newOrderType) =>
    devicesActionCreator.getDevices({
      branchId,
```

```
      params: {
        limit: pageSize,
        offset: PaginationEnum.DEFAULT_OFFSET,
        keyword,
        orderBy: newOrderBy,
        orderType: newOrderType,
        volumeUnit,
      },
    })
  )
  setPage(PaginationEnum.DEFAULT_PAGE)
}

const handleSearchByKeyword = (e) => {
  handleSearch(
    e,
    devicesActionCreator.getDevices({
      branchId,
      params: {
        limit: pageSize,
        offset: PaginationEnum.DEFAULT_OFFSET,
        keyword: e.target.value,
        orderBy,
        orderType,
        volumeUnit,
      },
    })
  )
  setPage(PaginationEnum.DEFAULT_PAGE)
}

const handleSelectPageClose = (ps) =>
  handlePageSizerClose(
    ps,
    devicesActionCreator.getDevices({
      branchId,
      params: {
        branchId,
        limit: ps,
        offset: PaginationEnum.DEFAULT_OFFSET,
        keyword,
        orderBy,
```

```
        orderType,
        volumeUnit,
      },
    })
  )

const handleEdit1 = (id) => {
  dispatch(
    devicesActionCreator.getDevice({
      id,
    })
  )
  handleOpenEditCreatedDialog()
}

const handleEdit2 = (id) => {
  dispatch(
    devicesActionCreator.getDevice({
      id,
    })
  )
  handleOpenEditDialog()
}

return (
  <MainCard
    title={
      <Grid
        container
        alignItems="center"
        justifyContent="space-between"
        spacing={gridSpacing}
      >
        <Grid item>
          {hasAccess(
            HttpMethod.POST,
            '/devices/tether',
            authUser?.authorityIds
          ) && (
            <Tooltip
              title={<FormattedMessage id="add" defaultMessage="Add" />}
              style={{
```

```
            marginRight: 10,
          }}
        >
          <Button
            variant="contained"
            size="medium"
            startIcon={<AddLinkIcon />}
            sx={{ px: 2.75, py: 1.5 }}
            onClick={handleOpenAddDialog}
          >
            <FormattedMessage id="add" defaultMessage="Add" />
          </Button>
        </Tooltip>
      )}
      {hasAccess(HttpMethod.POST, '/devices', authUser?.authorityIds) && (
        <Tooltip
          title={<FormattedMessage id="create" defaultMessage="Create" />}
        >
          <Button
            variant="contained"
            size="medium"
            startIcon={<AddCircleOutlineOutlinedIcon />}
            sx={{ px: 2.75, py: 1.5 }}
            onClick={handleOpenCreateDialog}
          >
            <FormattedMessage id="create" defaultMessage="Create" />
          </Button>
        </Tooltip>
      )}
      {openAdd && (
        <DeviceAdd
          open={openAdd}
          handleCloseDialog={handleCloseAddDialog}
        />
      )}
      {openCreate && (
        <DeviceCreate
          open={openCreate}
          handleCloseDialog={handleCloseCreateDialog}
        />
      )}
    </Grid>
```

```
      <Grid item>
        <OutlinedInput
          id="input-search-list"
          placeholder={intl.formatMessage({ id: 'search' })}
          onChange={handleSearchByKeyword}
          startAdornment={
            <InputAdornment position="start">
              <IconSearch stroke={1.5} size="1rem" />
            </InputAdornment>
          }
          size="small"
        />
      </Grid>
    </Grid>
  }
  content={false}
>
  {rows && (
    <DeviceList
      rows={rows}
      orderBy={orderBy}
      orderType={orderType}
      handleRequestSort={handleSort}
      handleOpenEdit1Dialog={handleEdit1}
      handleOpenEdit2Dialog={handleEdit2}
    />
  )}
  {openEdit && deviceById && (
    <DeviceUpdate
      device={deviceById}
      open={openEdit}
      handleCloseDialog={handleCloseEditDialog}
    />
  )}
  {openEditCreated && deviceById && (
    <DeviceUpdateCreated
      device={deviceById}
      open={openEditCreated}
      handleCloseDialog={handleCloseEditCreatedDialog}
    />
  )}
  <Grid item xs={12} sx={{ p: 3 }}>
```

```
        <Grid container justifyContent="space-between" spacing={gridSpacing}>
          <Grid item>
            <Pagination
              count={
                devices && devices.total
                  ? countPagesAmount(devices.total, pageSize)
                  : 1
              }
              onChange={(_, p) => handlePageNChange(p)}
              page={page}
              color="primary"
            />
          </Grid>
          <Grid item>
            <PageSizer
              anchorEl={anchorEl}
              pageSize={pageSize}
              handleClick={handlePageSizerClick}
              handleClose={handleSelectPageClose}
              sizeOptions={[
                PageSize.SIZE_10,
                PageSize.SIZE_20,
                PageSize.SIZE_50,
              ]}
            />
          </Grid>
        </Grid>
      </Grid>
    </MainCard>
  )
}

/* eslint-disable react/forbid-prop-types */
List.propTypes = {
  intl: PropTypes.object.isRequired,
}

export default injectIntl(List)
```

Файл pages/Devices/DetailedState/index.jsx

```
import React, { useEffect, useState, useMemo } from 'react'
```

```
import PropTypes from 'prop-types'
import { Link, useLocation } from 'react-router-dom'
import { FormattedMessage, injectIntl } from 'react-intl'
import { Autocomplete, Box, Tab, Tabs, TextField } from '@mui/material'

import MainCard from 'components/ui/cards/MainCard'
import {
  getValueOrDash,
  isLoading,
  getDetailedStateTabIndex,
} from 'utils/utils'
import { styled } from '@mui/material/styles'
import { shouldForwardProp } from '@mui/system'
import { useDispatch, useSelector } from 'react-redux'
import { PaginationEnum } from 'common/enums/app/pagination.enum'
import * as devicesActionCreator from 'store/devices/list/actions'
import { clearSubmitted } from 'store/common/actions'
import { hasAccess } from 'common/permissions/permissions'
import { HttpMethod } from 'common/enums/http/http-method.enum'
import useDialog from 'hooks/useDialog'

import { useTheme } from '@mui/styles'
import ChartInfo from './components/ChartInfo'
import SystemInfo from './components/SystemInfo'
import SystemSettings from './components/SystemSettings'
import AlarmLogging from './components/AlarmLogging'
import NoContentCard from './components/NoContentCard'
import SettingsUpdate from './components/SettingsUpdate'
import AlarmThresholds from './components/AlarmThresholds'
import AlarmHistory from './components/AlarmHistory'

const AutocompleteStyle = styled(Autocomplete, { shouldForwardProp })(
  ({ theme }) => ({
    width: 434,
    paddingRight: 16,
    marginBottom: 24,
    '& input': {
      background:
        theme.palette.mode === 'dark'
          ? theme.palette.background.default
          : '#fff',
      paddingLeft: '4px !important',
```

```
    },
    '& .MuiAutocomplete-inputRoot': {
      background:
        theme.palette.mode === 'dark'
          ? theme.palette.background.default
          : '#fff',
    },
    [theme.breakpoints.down('lg')]: {
      width: 250,
    },
    [theme.breakpoints.down('md')]: {
      width: '100%',
      marginLeft: 4,
      backgroundColor:
        theme.palette.mode === 'dark'
          ? theme.palette.background.default
          : '#fff',
    },
  })
)

/* eslint-disable react/forbid-prop-types */
const TabPanel = ({ children, value, index, ...other }) => (
  <div
    role="tabpanel"
    hidden={value !== index}
    id={`simple-tabpanel-${index}`}
    aria-labelledby={`simple-tab-${index}`}
    {...other}
  >
    {value === index && <Box sx={{ p: 0 }}>{children}</Box>}
  </div>
)

TabPanel.propTypes = {
  // eslint-disable-next-line react/require-default-props
  children: PropTypes.node,
  index: PropTypes.any.isRequired,
  value: PropTypes.any.isRequired,
}

const a11yProps = (index) => ({
```

```
  id: `simple-tab-${index}`,
  'aria-controls': `simple-tabpanel-${index}`,
 })

const DetailedState = ({ intl }) => {
  const location = useLocation()
  const theme = useTheme()
  const [searchValue, setSearchValue] = useState('')
  const [selectedDevice, setSelectedDevice] = useState({})
  const [tabValue, setTabValue] = useState(0)
  const { volumeUnit } = useSelector((state) => state.customization)
  const handleChange = (event, newValue) => {
   setTabValue(newValue)
  }

  const { authUser } = useSelector((state) => state.auth)

  const {
   selectedBranch: { id: branchId },
  } = useSelector((state) => state.header)
  const { submitted } = useSelector((state) => state.common)

  const dispatch = useDispatch()

  const [open, setOpen] = useState(false)

  const {
   devices,
   deviceById,
   waterLogs,
   analogLogs,
   devicePhones,
   deviceMembranes,
   loading,
  } = useSelector((state) => state.devices)

  const hasAccessToCharts = useMemo(
   () =>
    hasAccess(
     HttpMethod.GET,
     '/statistics/devices/*/*',
     authUser?.authorityIds
```

```
    ),
  [authUser]
)

const hasAccessToSettings = useMemo(
  () =>
    hasAccess(HttpMethod.GET, '/devices/*/settings', authUser?.authorityIds),
  [authUser]
)

const hasAccessToAlarmLogging = useMemo(
  () =>
    hasAccess(
      HttpMethod.GET,
      '/devices/*/alarm/settings',
      authUser?.authorityIds
    ),
  [authUser]
)
const hasAccessToAlarmThresholds = useMemo(
  () =>
    hasAccess(
      HttpMethod.GET,
      '/devices/*/monitoring-settings',
      authUser?.authorityIds
    ),
  [authUser]
)

const hasAccessToAlarmHistory = useMemo(
  () =>
    hasAccess(
      HttpMethod.GET,
      '/devices/*/alarms/logs',
      authUser?.authorityIds
    ),
  [authUser]
)

const tabIndices = useMemo(
  () => ({
    generalInfo: 0,
```

```
    charts: 1,
    settings: getDetailedStateTabIndex([hasAccessToCharts], 2),
    alarmLogging: getDetailedStateTabIndex(
      [hasAccessToCharts, hasAccessToSettings],
      3
    ),
    alarmThresholds: 4,
    alarmHistory: 5,
  }),
  [hasAccessToCharts, hasAccessToSettings]
)

useEffect(() => {
  dispatch(devicesActionCreator.clearDeviceById())
}, [branchId])

useEffect(() => {
  if (deviceById) {
    setSelectedDevice({
      id: deviceById.id,
      name: deviceById.name,
    })
  }
}, [deviceById, volumeUnit])

useEffect(() => {
  if (submitted) {
    dispatch(clearSubmitted())
  }
}, [submitted])

const handleOpen = () => {
  dispatch(
    devicesActionCreator.getDevices({
      branchId,
      params: {
        limit: PaginationEnum.DEFAULT_LIMIT_POPUP,
        offset: PaginationEnum.DEFAULT_OFFSET,
      },
    })
  )
  setOpen(true)
```

```
}

const handleSelectDevice = (value) => {
  setSelectedDevice(value)
  dispatch(
    devicesActionCreator.getDevice({
      id: value.id,
      volumeUnit,
    })
  )
  dispatch(devicesActionCreator.getDevicePhones(value.id))
  dispatch(devicesActionCreator.getDeviceMembranes(value.id))
  dispatch(
    devicesActionCreator.getDeviceWaterLogsLatest({
      id: value.id,
      volumeUnit,
    })
  )
  dispatch(devicesActionCreator.getDeviceAnalogLogsLatest(value.id))
}

useEffect(() => {
  if (location.state) {
    handleSelectDevice(location.state)
  }
}, [location.state])

useEffect(() => {
  if (deviceById) {
    handleSelectDevice(deviceById)
  }
}, [volumeUnit])

const { openEdit, handleOpenEditDialog, handleCloseEditDialog } = useDialog()

const parseSearchValue = (value) => {
  const parsed = value.split(' | ')
  if (parsed.length < 2) {
    return parsed[0]
  }

  return parsed[1]
```

```
}

const handleInputChange = (value) => {
  const parsed = parseSearchValue(value)

  setSearchValue(parsed)
  dispatch(
    devicesActionCreator.getDevices({
      branchId,
      params: {
        limit: PaginationEnum.DEFAULT_LIMIT_POPUP,
        offset: PaginationEnum.DEFAULT_OFFSET,
        keyword: parsed,
      },
    })
  )
}

const handleEdit = (data) => {
  dispatch(devicesActionCreator.setSettingData(data))
  handleOpenEditDialog()
}

return (
  <>
    <div
      style={{
        display: 'flex',
        gap: '0.5rem',
      }}
    >
      <AutocompleteStyle
        id="select-device"
        open={open}
        placeholder="Select device"
        onOpen={handleOpen}
        value={deviceById && selectedDevice}
        loading={isLoading(loading)}
        autoSelect
        loadingText={
          <FormattedMessage id="loading" defaultMessage="Loading..." />
        }
```

```
    onChange={(_, value) => handleSelectDevice(value)}
    onInputChange={(_, value) => handleInputChange(value)}
    onClose={() => setOpen(false)}
    isOptionEqualToValue={(option, value) => option.id === value.id}
    getOptionLabel={(option) =>
      option.id ? `ID: ${option.id} | ${option.name}` : searchValue
    }
    options={isLoading(loading) ? [] : devices?.data || []}
    disableClearable
    renderInput={(params) => (
      <TextField
        {...params}
        label={
          <FormattedMessage
            id="select_device"
            defaultMessage="Select device"
          />
        }
      />
    )}
  />
  <TextField
    disabled
    sx={{
      '& input': {
        background:
          theme.palette.mode === 'dark'
            ? theme.palette.background.default
            : '#fff',
      },
    }}
    value={`${intl.formatMessage({ id: 'model' })}: ${getValueOrDash(
      deviceById?.model?.name
    )}`}
  />
</div>
<MainCard>
  <div>
    {openEdit && (
      <SettingsUpdate
        open={openEdit}
        handleCloseDialog={handleCloseEditDialog}
```

```
  />
)}
<Tabs
  value={tabValue}
  indicatorColor="primary"
  onChange={handleChange}
  sx={{
    mb: 3,
    minHeight: 'auto',
    '& button': {
      minWidth: 100,
    },
    '& a': {
      minHeight: 'auto',
      minWidth: 10,
      py: 1.5,
      px: 1,
      mr: 2.25,
      color: 'grey.600',
    },
    '& a.Mui-selected': {
      color: 'primary.main',
    },
  }}
  aria-label="simple tabs example"
  variant="scrollable"
>
  <Tab
    component={Link}
    to="#"
    label={
      <FormattedMessage
        id="general_info"
        defaultMessage="General info"
      />
    }
    {...a11yProps(tabIndices.generalInfo)}
  />
  {hasAccessToCharts && (
    <Tab
      component={Link}
      to="#"
```

```
    label={<FormattedMessage id="charts" defaultMessage="Charts" />}
    {...a11yProps(tabIndices.charts)}
   />
  )}
  {hasAccessToSettings && (
   <Tab
    component={Link}
    to="#"
    label={
     <FormattedMessage id="settings" defaultMessage="Settings" />
    }
    {...a11yProps(tabIndices.charts)}
   />
  )}
  {hasAccessToAlarmLogging && (
   <Tab
    component={Link}
    to="#"
    label={
     <FormattedMessage
      id="alarm_logging"
      defaultMessage="Alarm logging"
     />
    }
    {...a11yProps(tabIndices.alarmLogging)}
   />
  )}

  {hasAccessToAlarmThresholds && (
   <Tab
    component={Link}
    to="#"
    label={
     <FormattedMessage
      id="alarm_thresholds"
      defaultMessage="Alarm thresholds"
     />
    }
    {...a11yProps(tabIndices.alarmThresholds)}
   />
  )}
  {hasAccessToAlarmHistory && (
```

```
    <Tab
     component={Link}
     to="#"
     label={
      <FormattedMessage
       id="alarm_history"
       defaultMessage="Alarm history"
      />
     }
     {...a11yProps(tabIndices.alarmHistory)}
    />
   )}
  </Tabs>
  <TabPanel value={tabValue} index={tabIndices.generalInfo}>
   {deviceById && waterLogs && analogLogs && devicePhones ? (
    <SystemInfo
     deviceData={deviceById}
     phonesData={devicePhones}
     waterLogs={waterLogs}
     analogLogs={analogLogs}
     membranesData={deviceMembranes}
    />
   ) : (
    <NoContentCard
     messageId="nothing_to_show"
     defaultMessage="Nothing to show"
     hintMessageId="select_device_first"
     hintDefaultMessage="Select device first"
    />
   )}
  </TabPanel>
  {hasAccessToCharts && (
   <TabPanel value={tabValue} index={tabIndices.charts}>
    {deviceById ? (
     <ChartInfo deviceId={deviceById.id} />
    ) : (
     <NoContentCard
      messageId="nothing_to_show"
      defaultMessage="Nothing to show"
      hintMessageId="select_device_first"
      hintDefaultMessage="Select device first"
     />
```

```
    )}
  </TabPanel>
)}
{hasAccessToSettings && (
  <TabPanel value={tabValue} index={tabIndices.settings}>
    {deviceById ? (
      <SystemSettings
        deviceId={deviceById.id}
        intl={intl}
        handleOpenDialog={handleEdit}
      />
    ) : (
      <NoContentCard
        messageId="nothing_to_show"
        defaultMessage="Nothing to show"
        hintMessageId="select_device_first"
        hintDefaultMessage="Select device first"
      />
    )}
  </TabPanel>
)}
{hasAccessToAlarmLogging && (
  <TabPanel value={tabValue} index={tabIndices.alarmLogging}>
    {deviceById ? (
      <AlarmLogging deviceId={deviceById.id} intl={intl} />
    ) : (
      <NoContentCard
        messageId="nothing_to_show"
        defaultMessage="Nothing to show"
        hintMessageId="select_device_first"
        hintDefaultMessage="Select device first"
      />
    )}
  </TabPanel>
)}

{hasAccessToAlarmThresholds && (
  <TabPanel value={tabValue} index={tabIndices.alarmThresholds}>
    {deviceById ? (
      <AlarmThresholds deviceId={deviceById.id} intl={intl} />
    ) : (
      <NoContentCard
```

```
                messageId="nothing_to_show"
                defaultMessage="Nothing to show"
                hintMessageId="select_device_first"
                hintDefaultMessage="Select device first"
              />
            )}
          </TabPanel>
        )}
        {hasAccessToAlarmHistory && (
          <TabPanel value={tabValue} index={tabIndices.alarmHistory}>
            {deviceById ? (
              <AlarmHistory deviceId={deviceById.id} intl={intl} />
            ) : (
              <NoContentCard
                messageId="nothing_to_show"
                defaultMessage="Nothing to show"
                hintMessageId="select_device_first"
                hintDefaultMessage="Select device first"
              />
            )}
          </TabPanel>
        )}
      </div>
    </MainCard>
  </>
 )
}

DetailedState.propTypes = {
 intl: PropTypes.object.isRequired,
}

export default injectIntl(DetailedState)
```

Файл main/java/com.ecosoft.rodms_api/controller/BranchController.java

```
package com.ecosoft.rodms_api.controller;

import com.ecosoft.rodms_api.config.security.CustomAuthentication;
import com.ecosoft.rodms_api.constant.DeviceStatsKeys;
import com.ecosoft.rodms_api.constant.OrderType;
import com.ecosoft.rodms_api.constant.RegexConstant;
```

```
import com.ecosoft.rodms_api.constant.VolumeUnit;
import com.ecosoft.rodms_api.dto.request.BranchCreateDto;
import com.ecosoft.rodms_api.dto.request.BranchUpdateDto;
import com.ecosoft.rodms_api.dto.response.EntityCount;
import com.ecosoft.rodms_api.dto.response.EntityResponseList;
import com.ecosoft.rodms_api.dto.response.LabeledEntityResponseList;
import com.ecosoft.rodms_api.dto.response.branch.BranchDto;
import com.ecosoft.rodms_api.dto.response.device.DeviceResponseDto;
import com.ecosoft.rodms_api.service.BranchService;
import com.ecosoft.rodms_api.service.DeviceService;
import com.ecosoft.rodms_api.service.StatisticService;
import com.ecosoft.rodms_api.service.validation.RequestValidationService;
import com.ecosoft.rodms_api.util.ConditionValidation;
import com.ecosoft.rodms_api.validation.ValidDeviceStatsKey;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.media.Schema;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

import javax.validation.Valid;
import javax.validation.constraints.Min;
import javax.validation.constraints.Pattern;
import java.security.Principal;
import java.util.List;
import java.util.Objects;

@AllArgsConstructor
@RestController
@RequestMapping("/branches")
```

```
@Validated
@Tag(name = "Branches", description = "Main CRUD for branches")
public class BranchController {

    private final RequestValidationService validationService;
    private final BranchService branchService;
    private final DeviceService deviceService;
    private final StatisticService statisticService;

    @GetMapping
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<EntityResponseList<BranchDto>> getBranches(@RequestParam @Min(0) Integer offset,
                                                           @RequestParam @Min(1) Integer limit,
                                                           @RequestParam(required = false) String keyword,
                                                           @RequestParam(required = false) @Pattern(regexp =
RegexConstant.ORDER_TYPE) String orderType,
                                                           Principal principal) {
        return validationService.validateOffsetLimit(offset, limit)
                .then(branchService.getBranches(limit, offset, keyword, orderType, (CustomAuthentication) principal));
    }

    @GetMapping("/{id}/devices")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<EntityResponseList<DeviceResponseDto>> getBranchDevices(@PathVariable Integer id,
                                                                        @RequestParam @Min(0) Integer offset,
                                                                        @RequestParam @Min(1) Integer limit,
                                                                        @RequestParam(required = false) String keyword,
                                                                        @Schema(type = "string", allowableValues = {"asc", "desc"})
                                                                        @RequestParam(required = false) @Pattern(regexp =
RegexConstant.ORDER_TYPE) String orderType,
                                                                        @Schema(type = "string", allowableValues = {"simBalance",
"waterGivenTotal",
                                                                                "volumeToSm", "timeInWork", "timeToSm",
                                                                                "lastConnectionAt", "alarmTypeUpdatedAt",
                                                                                "alarmType", "systemState", "id"})
                                                                        @RequestParam(required = false) String orderBy,
                                                                        @RequestParam(required = false) VolumeUnit volumeUnit,
                                                                        Principal principal) {
        String finalOrderType = ConditionValidation.validateConditionString(orderType) ? orderType :
OrderType.ORDER_TYPE_ASC.getOrder();
        var checkedVolumeUnit = Objects.nonNull(volumeUnit) ? volumeUnit : VolumeUnit.LITER;
        CustomAuthentication auth = (CustomAuthentication) principal;
```

```
        boolean ownerDevicesOnly = validationService.isOwnerDevicesOnly(auth);

        return validationService.validateBranchId(id)
                .then(validationService.validateOffsetLimit(offset, limit))
                .then(deviceService.getDevices(id, limit, offset, keyword, finalOrderType, orderBy, checkedVolumeUnit,
ownerDevicesOnly, auth.getUserId()));
    }

    @GetMapping("/{id}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<BranchDto> getBranchById(@PathVariable @Min(1) Integer id, Principal principal) {
        return validationService.validateBranchId(id)
                .then(validationService.validateBranchIdPermission(id, principal))
                .then(branchService.getBranchById(id));
    }

    @PostMapping
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> createBranch(@RequestBody @Valid BranchCreateDto dto) {
        return validationService.validateBranchCreateDto(dto)
                .then(branchService.createBranch(dto));
    }

    @PutMapping("/{id}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> updateBranch(@PathVariable @Min(1) Integer id, @RequestBody @Valid
BranchUpdateDto dto) {
        return validationService.validateBranchId(id)
                .then(validationService.validateBranchUpdateDto(dto))
                .then(branchService.updateBranch(id, dto));
    }

    @GetMapping("/{id}/stats")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<List<EntityCount>> getBranchStats(@PathVariable @Min(1) Integer id) {
        return validationService.validateBranchId(id)
                .then(statisticService.getDeviceStatsForBranch(id));
    }

    @GetMapping("/{id}/devices/{state}")
```

```
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<LabeledEntityResponseList<DeviceResponseDto>> getBranchDevicesByState(@PathVariable
Integer id,
                                                                                      @RequestParam @Min(0) Integer offset,
                                                                                      @RequestParam @Min(1) Integer limit,
                                                                                      @Schema(type = "string", allowableValues = {"alarm", "non-
connecting", "low-sim-money"})
                                                                                      @PathVariable @ValidDeviceStatsKey String state,
                                                                                      @RequestParam(required = false) String keyword,
                                                                                      @Schema(type = "string", allowableValues = {"asc", "desc"})
                                                                                      @RequestParam(required = false) @Pattern(regexp =
RegexConstant.ORDER_TYPE) String orderType,
                                                                                      @Schema(type = "string", allowableValues = {"simBalance",
"waterGivenTotal",
                                                                                              "volumeToSm", "timeInWork", "timeToSm",
                                                                                              "lastConnectionAt", "alarmTypeUpdatedAt",
                                                                                              "alarmType", "systemState", "id"})
                                                                                      @RequestParam(required = false) String orderBy,
                                                                                      @RequestParam(required = false) VolumeUnit volumeUnit) {
        String finalOrderType = ConditionValidation.validateConditionString(orderType) ? orderType :
OrderType.ORDER_TYPE_ASC.getOrder();
        var checkedVolumeUnit = Objects.nonNull(volumeUnit) ? volumeUnit : VolumeUnit.LITER;
        String deviceState = DeviceStatsKeys.convertToUpperUnderscore(state);
        return validationService.validateBranchId(id)
                .then(validationService.validateOffsetLimit(offset, limit))
                .then(deviceService.getBranchDevicesByState(id, limit, offset, deviceState, keyword, finalOrderType,
orderBy, checkedVolumeUnit));
    }
}
```

Файл main/java/com.ecosoft.rodms_api/controller/DeviceActionController.java

```
package com.ecosoft.rodms_api.controller;

import com.ecosoft.rodms_api.constant.Action;
import com.ecosoft.rodms_api.service.ActionPublishService;
import com.ecosoft.rodms_api.service.validation.RequestValidationService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
```

```
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

import javax.validation.constraints.Min;

@AllArgsConstructor
@RestController
@RequestMapping("/devices")
@Validated
@Tag(name = "Devices actions", description = "Publish requests for device actions")
public class DeviceActionController {
    private RequestValidationService validationService;
    private ActionPublishService actionPublishService;

    @PutMapping("/{id}/firmware-update")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.ACCEPTED)
    public Mono<Void> sendFirmwareUpdateCheck(@PathVariable @Min(1) Integer id) {
        return validationService.validateDeviceId(id)
                .then(actionPublishService.publishActionToMiddleware(id, Action.FIRMWARE_UPDATE));
    }

    @PutMapping("/{id}/reset-alarm")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.ACCEPTED)
    public Mono<Void> sendResetAlarm(@PathVariable @Min(1) Integer id) {
        return validationService.validateDeviceId(id)
                .then(actionPublishService.publishActionToMiddleware(id, Action.RESET_ALARM));
    }

    @PutMapping("/{id}/request-settings")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.ACCEPTED)
    public Mono<Void> sendSettingsRequest(@PathVariable @Min(1) Integer id) {
        return validationService.validateDeviceId(id)
                .then(actionPublishService.publishActionToMiddleware(id, Action.REQUEST_SETTINGS));
    }
```

}

Файл main/java/com.ecosoft.rodms_api/controller/DeviceController.java

```
package com.ecosoft.rodms_api.controller;

import com.ecosoft.rodms_api.constant.VolumeUnit;
import com.ecosoft.rodms_api.dto.request.DeviceCreateUpdateDto;
import com.ecosoft.rodms_api.dto.request.DeviceMembraneElementsCreateDto;
import com.ecosoft.rodms_api.dto.request.DevicePhonesDto;
import com.ecosoft.rodms_api.dto.request.DeviceTetherDto;
import com.ecosoft.rodms_api.dto.request.DeviceUpdateDto;
import com.ecosoft.rodms_api.dto.response.EntityCount;
import com.ecosoft.rodms_api.dto.response.device.DeviceMembraneElementsDto;
import com.ecosoft.rodms_api.dto.response.device.DeviceResponseDto;
import com.ecosoft.rodms_api.service.DeviceService;
import com.ecosoft.rodms_api.service.DeviceSettingsService;
import com.ecosoft.rodms_api.service.StatisticService;
import com.ecosoft.rodms_api.service.validation.RequestValidationService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

import javax.validation.Valid;
import javax.validation.constraints.Min;
import java.security.Principal;
import java.util.List;
import java.util.Objects;
```

```
@AllArgsConstructor
@RestController
@RequestMapping("/devices")
@Validated
@Tag(name = "Devices", description = "Main CRUD for devices")
public class DeviceController {
    private RequestValidationService validationService;
    private DeviceService deviceService;
    private DeviceSettingsService deviceSettingsService;
    private StatisticService statisticService;

    @GetMapping("/{id}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<DeviceResponseDto> getDeviceById(@PathVariable @Min(1) Integer id,
                                                 @RequestParam(required = false) VolumeUnit volumeUnit) {
        var checkedVolumeUnit = Objects.nonNull(volumeUnit) ? volumeUnit : VolumeUnit.LITER;
        return validationService.validateDeviceId(id).then(deviceService.getDeviceById(id, checkedVolumeUnit));
    }

    @PostMapping
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> createDevice(@RequestBody @Valid DeviceCreateUpdateDto dto) {
        return deviceService.createDevice(dto);
    }

    @PutMapping("/{id}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> updateDevice(@PathVariable Integer id, @RequestBody @Valid DeviceCreateUpdateDto
dto) {
        return validationService.validateDeviceId(id)
                .then(deviceService.updateDevice(id, dto));
    }

    @PutMapping("/tether/{id}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> updateTetheredDevice(@PathVariable Integer id, @RequestBody @Valid
DeviceUpdateDto dto) {
        return validationService.validateDeviceId(id)
                .then(deviceService.updateDevice(id, dto));
```

```
    }

    @GetMapping("/{id}/phones")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<DevicePhonesDto> getDevicePhonesByDeviceId(@PathVariable @Min(1) Integer id) {
        return validationService.validateDeviceId(id)
                .then(deviceSettingsService.getDevicePhonesByDeviceId(id));
    }

    @PutMapping("/{id}/phones")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> setPhonesForDevice(@PathVariable @Min(1) Integer id,
                                         @RequestBody @Valid DevicePhonesDto dto) {
        return validationService.validateDeviceId(id)
                .then(validationService.validateUniqueDevicePhoneNumber(id, dto))
                .then(deviceSettingsService.setPhonesForDevice(id, dto));
    }

    @GetMapping("/stats/by-status")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<List<EntityCount>> getTotalDeviceStatusStats() {
        return statisticService.getTotalDeviceStatusStats();
    }

    @PostMapping("/tether")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> createDevice(@RequestBody @Valid DeviceTetherDto dto, Principal principal) {
        return validationService.validateDeviceTetherDtoBranchId(dto, principal)
                .flatMap(branchId -> validationService.validateBranchId(branchId).thenReturn(branchId))
                .flatMap(branchId -> validationService.validateDeviceTetherDtoUserIds(dto).thenReturn(branchId))
                .flatMap(branchId -> deviceService.tetherDevice(branchId, dto));
    }

    @GetMapping("/{id}/membranes")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<DeviceMembraneElementsDto> getMembraneElementsForDevice(@PathVariable @Min(1)
Integer id) {
        return validationService.validateDeviceId(id)
                .then(deviceService.getMembraneElementsForDevice(id));
    }
```

```
    @PutMapping("/{id}/membranes")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> setMembraneElementsForDevice(@PathVariable @Min(1) Integer id,
                                                   @RequestBody @Valid DeviceMembraneElementsCreateDto dto) {
        return validationService.validateDeviceId(id)
                .then(deviceService.setMembraneElementsForDevice(id, dto));
    }
}
```

Файл main/java/com.ecosoft.rodms_api/controller/DeviceLogsController.java

```
package com.ecosoft.rodms_api.controller;

import com.ecosoft.rodms_api.constant.AlarmType;
import com.ecosoft.rodms_api.constant.OrderType;
import com.ecosoft.rodms_api.constant.RegexConstant;
import com.ecosoft.rodms_api.constant.VolumeUnit;
import com.ecosoft.rodms_api.dto.request.DeviceAlarmLogsRequest;
import com.ecosoft.rodms_api.dto.response.EntityResponseList;
import com.ecosoft.rodms_api.dto.response.device.DeviceAlarmsDto;
import com.ecosoft.rodms_api.dto.response.log.AlarmDto;
import com.ecosoft.rodms_api.dto.response.log.SensorAnalogLogDto;
import com.ecosoft.rodms_api.dto.response.log.WaterLogDto;
import com.ecosoft.rodms_api.service.DeviceLogsService;
import com.ecosoft.rodms_api.service.DeviceService;
import com.ecosoft.rodms_api.service.validation.RequestValidationService;
import com.ecosoft.rodms_api.util.ConditionValidation;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.AllArgsConstructor;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import java.util.Objects;

@AllArgsConstructor
@RestController
@RequestMapping("/devices")
@Validated
@Tag(name = "Device logs", description = "Device alarm logs and latest data")
public class DeviceLogsController {
    private RequestValidationService validationService;
    private DeviceService deviceService;
    private DeviceLogsService deviceLogsService;

    public static final String ATTACHMENT = "attachment; filename=";

    @GetMapping("/{id}/alarms/latest")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<DeviceAlarmsDto> getDeviceAlarms(@PathVariable @Min(1) Integer id,
                                                 @RequestParam(required = false) VolumeUnit volumeUnit) {
        var checkedVolumeUnit = Objects.nonNull(volumeUnit) ? volumeUnit : VolumeUnit.LITER;
        return validationService.validateDeviceId(id)
                .then(deviceService.getDeviceAlarms(id, checkedVolumeUnit));
    }

    @GetMapping("/{id}/logs/analog/latest")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<SensorAnalogLogDto> getDeviceLatestSensorAnalogLog(@PathVariable @Min(1) Integer id) {
        return validationService.validateDeviceId(id)
                .then(deviceLogsService.getDeviceLatestSensorAnalogLog(id));
    }

    @GetMapping("/{id}/logs/water/latest")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<WaterLogDto> getDeviceLatestWaterLog(@PathVariable @Min(1) Integer id,
                                                     @RequestParam(required = false) VolumeUnit volumeUnit) {
        var checkedVolumeUnit = Objects.nonNull(volumeUnit) ? volumeUnit : VolumeUnit.LITER;
```

```
        return validationService.validateDeviceId(id)
                .then(deviceLogsService.getDeviceLatestWaterLog(id, checkedVolumeUnit));
    }

    @GetMapping("/{id}/alarms/logs")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<EntityResponseList<AlarmDto>> getDeviceAlarmsLogs(@PathVariable @Min(1) Integer id,
                                                  @RequestParam @Min(0) Integer offset,
                                                  @RequestParam @Min(1) Integer limit,
                                                  @RequestParam @NotNull Long startDate,
                                                  @RequestParam @NotNull Long endDate,
                                                  @RequestParam(required = false) AlarmType alarmType,
                                                  @RequestParam(required = false) @Pattern(regexp =
RegexConstant.ORDER_TYPE) String orderType) {
        String finalOrderType = ConditionValidation.validateConditionString(orderType) ? orderType :
OrderType.ORDER_TYPE_DESC.getOrder();
        DeviceAlarmLogsRequest request = new DeviceAlarmLogsRequest(id, startDate, endDate, limit, offset,
alarmType, finalOrderType);
        return validationService.validateDeviceId(id)
                .then(validationService.validateOffsetLimit(offset, limit))
                .then(deviceLogsService.getDeviceAlarmLogs(request));
    }

    @GetMapping("/{id}/alarms/logs/xlsx")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<ResponseEntity<InputStreamResource>> getDeviceAlarmReport(@PathVariable @Min(1) Integer
id,
                                                  @RequestParam @NotNull Long startDate,
                                                  @RequestParam @NotNull Long endDate,
                                                  @RequestParam(required = false) AlarmType alarmType,
                                                  @RequestParam(required = false, defaultValue = "Europe/Kiev")
String timezone) {
        return validationService.validateDeviceId(id)
                .then(deviceLogsService.getDeviceAlarmReport(id, startDate, endDate, alarmType, timezone))
                .flatMap(report ->
                        Mono.just(ResponseEntity.ok()
                                .header(HttpHeaders.CONTENT_DISPOSITION, ATTACHMENT + report.getFilename())
                                .header(HttpHeaders.CONTENT_TYPE,
MediaType.APPLICATION_OCTET_STREAM_VALUE)
                                .body(report.getResource())));
    }
}
```

Файл main/java/com.ecosoft.rodms_api/controller/DeviceModelController.java

```
package com.ecosoft.rodms_api.controller;

import com.ecosoft.rodms_api.constant.RegexConstant;
import com.ecosoft.rodms_api.dto.request.DeviceModelCreateUpdateDto;
import com.ecosoft.rodms_api.dto.response.EntityResponseList;
import com.ecosoft.rodms_api.dto.response.model.DeviceModelDto;
import com.ecosoft.rodms_api.service.DeviceModelService;
import com.ecosoft.rodms_api.service.validation.RequestValidationService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

import javax.validation.Valid;
import javax.validation.constraints.Min;
import javax.validation.constraints.Pattern;

@AllArgsConstructor
@RestController
@RequestMapping("/models")
@Validated
@Tag(name = "Device models", description = "Main CRUD for device models")
public class DeviceModelController {
    private RequestValidationService requestValidationService;
    private DeviceModelService deviceModelService;
```

```
    @PostMapping
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> createDeviceModel(@RequestBody @Valid DeviceModelCreateUpdateDto dto) {
        return deviceModelService.createDeviceModel(dto);
    }

    @GetMapping
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<EntityResponseList<DeviceModelDto>> getDeviceModels(@RequestParam @Min(0) Integer
offset,
                                                    @RequestParam @Min(1) Integer limit,
                                                    @RequestParam(required = false) String keyword,
                                                    @RequestParam(required = false)
                                                    @Pattern(regexp = RegexConstant.ORDER_TYPE) String orderType,
                                                    @RequestParam(required = false, defaultValue = "false") Boolean
forCalculation) {
        return requestValidationService.validateOffsetLimit(offset, limit)
                .then(deviceModelService.getDeviceModels(limit, offset, keyword, orderType, forCalculation));
    }

    @GetMapping("/{id}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<DeviceModelDto> getDeviceModelById(@PathVariable @Min(1) Integer id) {
        return deviceModelService.getDeviceModelById(id);
    }

    @PutMapping("/{id}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> updateDeviceModel(@PathVariable @Min(1) Integer id, @RequestBody @Valid
DeviceModelCreateUpdateDto dto) {
        return deviceModelService.updateDeviceModel(id, dto);
    }
}
```

Файл main/java/com.ecosoft.rodms_api/controller/BranchController.java

```
import React, { useEffect, useState } from 'react'
import { useDispatch, useSelector } from 'react-redux'
import PropTypes from 'prop-types'
```

Файл main/java/com.ecosoft.rodms_api/controller/DeviceSettingsController.java

```
package com.ecosoft.rodms_api.controller;

import com.ecosoft.rodms_api.dto.request.AlarmLoggingSettingUpdateDto;
import com.ecosoft.rodms_api.dto.request.DeviceMonitoringSettingUpdateDto;
import com.ecosoft.rodms_api.dto.request.SettingUpdateDTO;
import com.ecosoft.rodms_api.dto.response.ControllerSettingDTO;
import com.ecosoft.rodms_api.dto.response.DatedEntityResponseList;
import com.ecosoft.rodms_api.dto.response.EntityResponseList;
import com.ecosoft.rodms_api.dto.response.setting.AlarmLoggingSettingDTO;
import com.ecosoft.rodms_api.dto.response.setting.DeviceMonitoringSettingDto;
import com.ecosoft.rodms_api.dto.response.setting.MonitoringSettingDTO;
import com.ecosoft.rodms_api.dto.response.setting.SettingValuesDTO;
import com.ecosoft.rodms_api.service.ControllerSettingService;
import com.ecosoft.rodms_api.service.DeviceAlarmLoggingSettingsService;
import com.ecosoft.rodms_api.service.DeviceMonitoringSettingsService;
import com.ecosoft.rodms_api.service.validation.RequestValidationService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

import javax.validation.Valid;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
```

```
@AllArgsConstructor
@RestController
@RequestMapping("/devices")
@Validated
@Tag(name = "Device settings", description = "Settings of device controller & monitoring")
public class DeviceSettingController {
    private final RequestValidationService validationService;
    private final ControllerSettingService controllerSettingService;
    private final DeviceMonitoringSettingsService deviceMonitoringSettingsService;
    private final DeviceAlarmLoggingSettingsService deviceAlarmLoggingSettingsService;

    @GetMapping("/{id}/settings")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<DatedEntityResponseList<ControllerSettingDTO>>
getControllerSettingsByDeviceId(@PathVariable @Min(1) Integer id,
                                                    @RequestParam @NotNull @Min(1) Integer limit,
                                                    @RequestParam @NotNull @Min(0) Integer offset,
                                                    @RequestParam(required = false) String keyword) {
        return validationService.validateDeviceId(id)
                .then(controllerSettingService.getControllerSettingsByDeviceId(id, limit, offset, keyword));
    }

    @PutMapping("/{id}/settings/{name}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> changeControllerSetting(@PathVariable @Min(1) Integer id,
                                    @PathVariable @NotBlank String name,
                                    @RequestBody @Valid SettingUpdateDTO dto) {
        return validationService.validateDeviceId(id)
                .then(controllerSettingService.handleNewControllerSetting(id, name, dto));
    }

    @GetMapping("/settings/{name}/values")
    public Mono<SettingValuesDTO> getSettingValues(@PathVariable String name) {
        return controllerSettingService.getSettingDisplaySettingByName(name);
    }

    @GetMapping("/monitoring-settings")
    public Mono<EntityResponseList<MonitoringSettingDTO>> getAllSystemSettings() {
        return deviceMonitoringSettingsService.getAllSystemSettings();
    }
```

```
    @GetMapping("/{id}/monitoring-settings")
    public Mono<EntityResponseList<DeviceMonitoringSettingDto>> getDeviceMonitoringSettings(@PathVariable
@Min(1) Integer id,
                                                                @RequestParam @NotNull @Min(1) Integer limit,
                                                                @RequestParam @NotNull @Min(0) Integer offset,
                                                                @RequestParam(required = false) String keyword) {
        return deviceMonitoringSettingsService.getDeviceMonitoringSettings(id, limit, offset, keyword);
    }

    @PutMapping("/{id}/monitoring-settings")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> updateDeviceMonitoringSetting(@PathVariable @Min(1) Integer id,
                                                    @RequestBody @Valid DeviceMonitoringSettingUpdateDto dto) {
        return validationService.validateDeviceId(id)
                .then(deviceMonitoringSettingsService.updateDeviceMonitoringSetting(id, dto));
    }

    @PostMapping("/{id}/monitoring-settings/defaults")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> setDeviceMonitoringSettingsFromDefaults(@PathVariable @Min(1) Integer id) {
        return validationService.validateDeviceId(id)
                .then(deviceMonitoringSettingsService.setDeviceMonitoringSettingsFromDefaults(id));
    }

    @GetMapping("/{id}/alarm/settings")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<EntityResponseList<AlarmLoggingSettingDTO>>
getAlarmLoggingSettingsByDeviceId(@PathVariable @Min(1) Integer id,
                                                                @RequestParam @NotNull @Min(1) Integer limit,
                                                                @RequestParam @NotNull @Min(0) Integer offset,
                                                                @RequestParam(required = false) String keyword) {
        return validationService.validateDeviceId(id)
                .then(deviceAlarmLoggingSettingsService.getAlarmLoggingSettingsByDeviceId(id, limit, offset,
keyword));
    }

    @PutMapping("/{deviceId}/alarm/{alarmId}/settings")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> updateAlarmLoggingSettingForDevice(@PathVariable @Min(1) Integer deviceId,
```

```
                                    @PathVariable @Min(1) Integer alarmId,
                                    @RequestBody @Valid AlarmLoggingSettingUpdateDto dto) {
        return validationService.validateDeviceId(deviceId)
                .then(deviceAlarmLoggingSettingsService.updateAlarmLoggingSettingForDevice(deviceId, alarmId,
dto));
    }
}
```

Файл main/java/com.ecosoft.rodms_api/controller/MembraneElementController.java

```
package com.ecosoft.rodms_api.controller;

import com.ecosoft.rodms_api.constant.RegexConstant;
import com.ecosoft.rodms_api.dto.request.MembraneElementModelCreateUpdateDto;
import com.ecosoft.rodms_api.dto.response.EntityResponseList;
import com.ecosoft.rodms_api.dto.response.model.MembraneElementModelDto;
import com.ecosoft.rodms_api.service.MembraneElementModelService;
import com.ecosoft.rodms_api.service.validation.RequestValidationService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.AllArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

import javax.validation.Valid;
import javax.validation.constraints.Min;
import javax.validation.constraints.Pattern;

@AllArgsConstructor
```

```
@RestController
@RequestMapping("/membranes")
@Validated
@Tag(name = "Membrane element models", description = "Main CRUD for membrane element models")
public class MembraneElementModelController {
    private final RequestValidationService requestValidationService;
    private final MembraneElementModelService membraneElementModelService;

    @PostMapping
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public Mono<Void> createMembraneElementModel(@RequestBody @Valid
MembraneElementModelCreateUpdateDto dto) {
        return membraneElementModelService.createMembraneElementModel(dto);
    }

    @GetMapping
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<EntityResponseList<MembraneElementModelDto>>
getMembraneElementModels(@RequestParam @Min(0) Integer offset,
                                                    @RequestParam @Min(1) Integer limit,
                                                    @RequestParam(required = false) String keyword,
                                                    @RequestParam(required = false) @Pattern(regexp =
RegexConstant.ORDER_TYPE) String orderType,
                                                    @RequestParam(required = false, defaultValue = "false")
Boolean forCalculation) {
        return requestValidationService.validateOffsetLimit(offset, limit)
                .then(membraneElementModelService.getMembraneElementModels(limit, offset, keyword, orderType,
forCalculation));
    }

    @GetMapping("/{id}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<MembraneElementModelDto> getMembraneElementModelById(@PathVariable @Min(1) Integer
id) {
        return membraneElementModelService.getMembraneElementModelById(id);
    }

    @PutMapping("/{id}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    @ResponseStatus(HttpStatus.NO_CONTENT)
```

```
    public Mono<Void> updateMembraneElementModel(@PathVariable @Min(1) Integer id, @RequestBody
@Valid MembraneElementModelCreateUpdateDto dto) {
        return membraneElementModelService.updateMembraneElementModel(id, dto);
    }
}
```

Файл main/java/com.ecosoft.rodms_api/controller/StatisticController.java

```
package com.ecosoft.rodms_api.controller;

import com.ecosoft.rodms_api.constant.StatisticPeriod;
import com.ecosoft.rodms_api.constant.StatisticalValueType;
import com.ecosoft.rodms_api.constant.VolumeUnit;
import com.ecosoft.rodms_api.dto.request.StatisticRequest;
import com.ecosoft.rodms_api.dto.response.statistic.StatisticDto;
import com.ecosoft.rodms_api.dto.response.statistic.StatisticResponseList;
import com.ecosoft.rodms_api.service.StatisticService;
import com.ecosoft.rodms_api.service.validation.RequestValidationService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.AllArgsConstructor;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

import javax.validation.constraints.Min;
import java.security.Principal;
import java.util.Objects;

@AllArgsConstructor
@RestController
@RequestMapping("/statistics")
@Validated
@Tag(name = "Statistics", description = "Aggregated operational data for devices and branches ")
public class StatisticController {
    private RequestValidationService validationService;
    private StatisticService statisticService;
```

```
    @GetMapping("/devices/{id}/{valueType}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<StatisticResponseList<StatisticDto>> getDeviceStatisticsForEntityAndPeriod(
            @PathVariable @Min(1) Integer id,
            @RequestParam Long startDate,
            @RequestParam Long endDate,
            @PathVariable StatisticalValueType valueType,
            @RequestParam(required = false) StatisticPeriod period,
            @RequestParam(required = false) VolumeUnit volumeUnit,
            Principal principal) {

        var statisticPeriod = Objects.nonNull(period) ? period : StatisticPeriod.DAY;
        var checkedVolumeUnit = Objects.nonNull(volumeUnit) ? volumeUnit : VolumeUnit.LITER;

        var request = new StatisticRequest(StatisticRequest.DEVICE, id, startDate, endDate, valueType,
statisticPeriod);
        return validationService.validateStatisticRequest(principal, request)
                .then(statisticService.getStatisticsForEntityAndPeriod(request, checkedVolumeUnit));
    }

    @GetMapping("/branches/{id}/{valueType}")
    @Operation(security = @SecurityRequirement(name = "bearerAuth"))
    public Mono<StatisticResponseList<StatisticDto>> getBranchStatisticsForEntityAndPeriod(
            @PathVariable @Min(1) Integer id,
            @RequestParam Long startDate,
            @RequestParam Long endDate,
            @PathVariable StatisticalValueType valueType,
            @RequestParam(required = false) StatisticPeriod period,
            @RequestParam(required = false) VolumeUnit volumeUnit,
            Principal principal) {

        var statisticPeriod = Objects.nonNull(period) ? period : StatisticPeriod.DAY;
        var checkedVolumeUnit = Objects.nonNull(volumeUnit) ? volumeUnit : VolumeUnit.LITER;

        var request = new StatisticRequest(StatisticRequest.BRANCH, id, startDate, endDate, valueType,
statisticPeriod);
        return validationService.validateStatisticRequest(principal, request)
                .then(statisticService.getStatisticsForEntityAndPeriod(request, checkedVolumeUnit));
    }
}
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ВІДДАЛЕНОГО КЕРУВАННЯ ТА МОНІТОРИНГУ РОБОТИ СИСТЕМ ОЧИЩЕННЯ ВОДИ

Програма та методика тестування

КПІ.ІП-9619.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Артем КАРІМОВ

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є веб-застосунок для віддаленого керування та моніторингу роботи систем очищення води. Оскільки це веб-застосунок, то використовувати його можна на пристроях, на яких можна зайти у веб-браузери, такі як Google Chrome, Safari, Microsoft Edge, Mozilla Firefox, Opera та інші.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка виконання поставлених нефункціональних вимог як операційних, так і до інтерфейсу;

– перевірка сумісності веб-застосунку з останніми версіями сучасних браузерів (Chrome, Safari, Microsoft Edge, Mozilla Firefox, Opera);

– перевірка сумісності застосунку з різними операційними системами (Windows, MacOS, Linux);

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– системне тестування – перевіряється усе програмне забезпечення в цілому;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення

– тестування безпеки – тестування для перевірки вразливостей програмного забезпечення.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

– динамічне тестування на відповідність функціональним вимогам;

– тестування на мобільних пристроях із різною роздільною здатністю екрану;

– тестування на мобільних пристроях із різною версією операційної системи;

– тестування на персональних комп’ютерах із різною версією операційної системи;

– тестування на виведення повідомлень про помилку, коли це необхідно;

– тестування зміни орієнтації екрану;

– тестування інтерфейсу користувача;

– тестування зручності використання застосунку.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ВІДДАЛЕНОГО КЕРУВАННЯ ТА МОНІТОРИНГУ РОБОТИ СИСТЕМ ОЧИЩЕННЯ ВОДИ

Керівництво користувача

КПІ.ІП-9619.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Артем КАРІМОВ

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

RODMS – це веб-застосунок, призначений для віддаленого керування та моніторингу роботи апаратів очищення води користувачами систем зворотного осмосу. Він дозволяє швидко отримувати необхідну про апарати інформацію без необхідності знаходитися біля них, а також керувати ними без необхідності виїздів до них сервісних спеціалістів.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– тип процесора: Intel Core i5;

– об’єм оперативної пам’яті: 8 Гб;

– підключення до мережі Інтернет зі швидкістю від 20 мегабіт.

2.2 Завантаження застосунку

Оскільки в рамках дипломного проєктування було створено веб-застосунок, то для запуску необхідно відкрити його в будь-якому браузері на будь-якій платформі, на якій можна вийти в браузер.

2.3 Перевірка коректної роботи

Після запуску застосунку в браузері користувач має перейти на сторінку логіну, якщо він користується додатком уперше або не заходив у нього протягом останніх 24 годин. Якщо ж пройшло менше, ніж 24 години з моменту останнього заходу користувача в застосунок, то має відбутися перехід на головну сторінку, що доступна авторизованому користувачеві – сторінку з дашбордом.

3 ВИКОНАННЯ ПРОГРАМИ

Як було сказано в попередньому розділі, після запуску застосунку в браузері користувач має перейти на сторінку логіну, якщо він користується додатком уперше або не заходив у нього протягом останніх 24 годин. На сторінці знаходиться форма з полями для вводу імені користувача та пароля (рисунок 3.1).

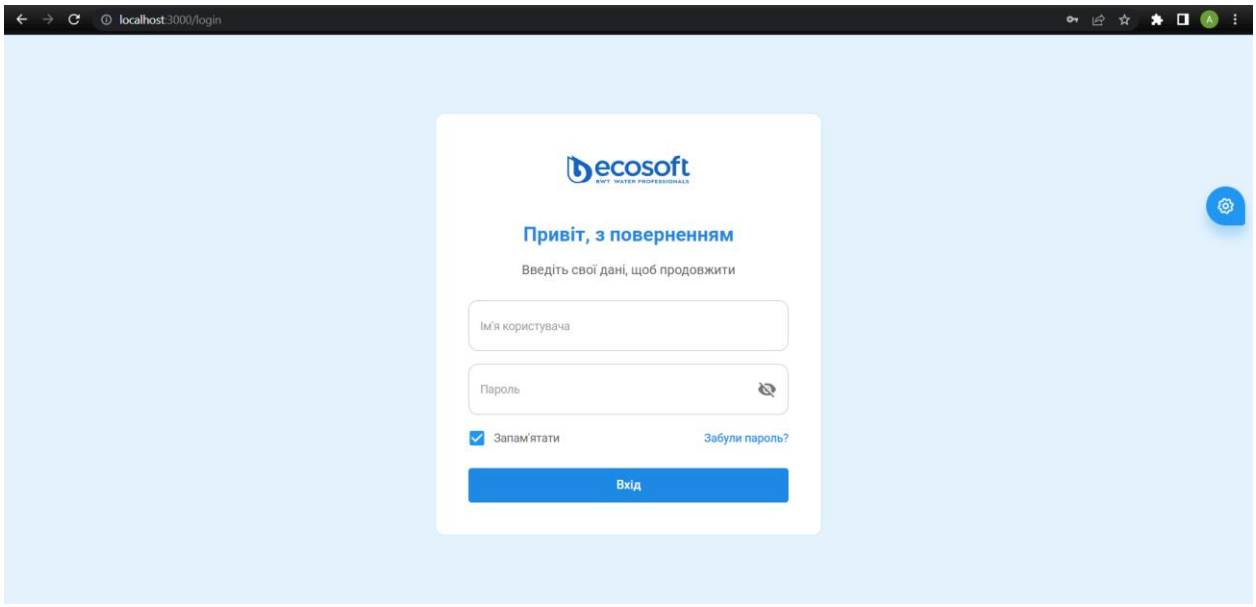


Рисунок 3.1 – Сторінка логіну

Щоб увійти в застосунок, користувачеві необхідно ввести ці дані. Після введення коректних даних користувач переходить на головну сторінку застосунку – сторінку з дашкордом (рисунок 3.2).

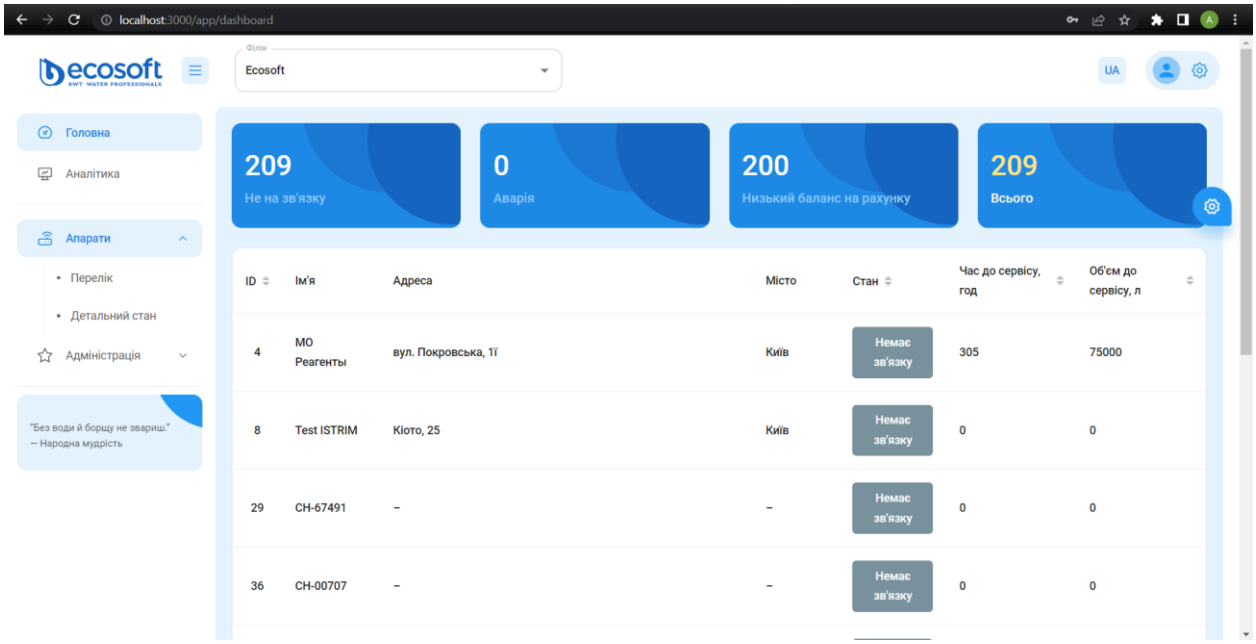


Рисунок 3.2 – Головна сторінка застосунку

Щоб перейти на сторінку з аналітикою (рисунок 3.3), необхідно натиснути вкладку «Аналітика» у боковому меню.

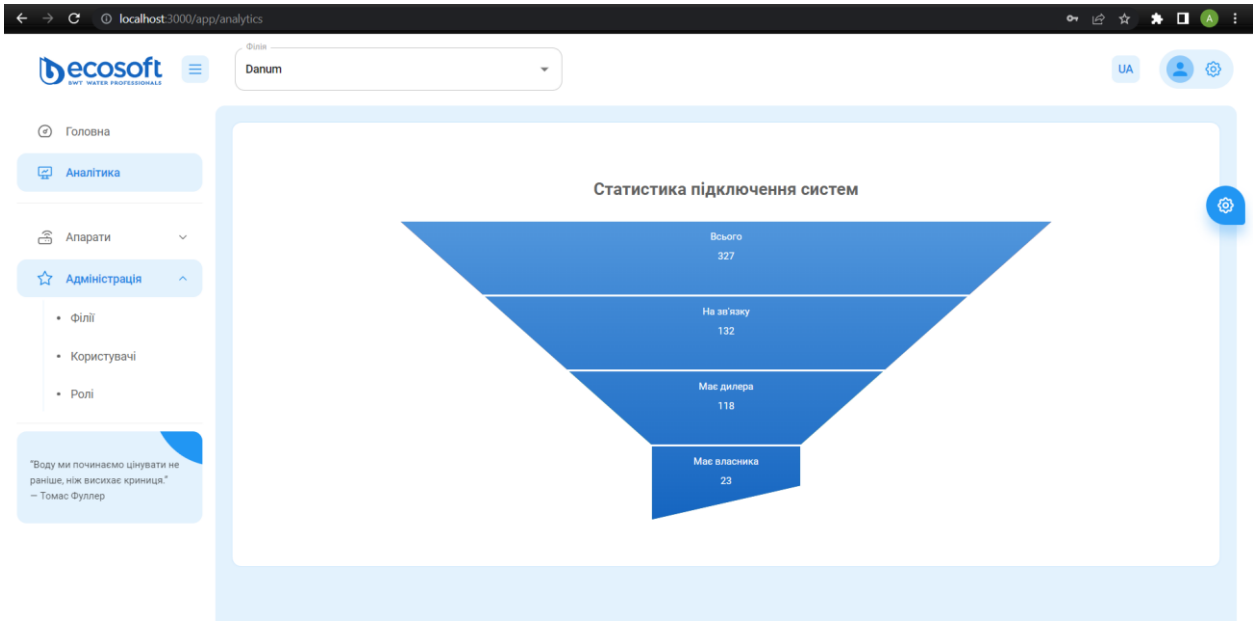


Рисунок 3.3 – Сторінка з аналітикою

Щоб перейти на сторінку зі списком апаратів поточної філії (рисунок 3.4), необхідно натиснути вкладку «Апарати» в боковому меню й вибирати дочірню вкладку «Перелік».

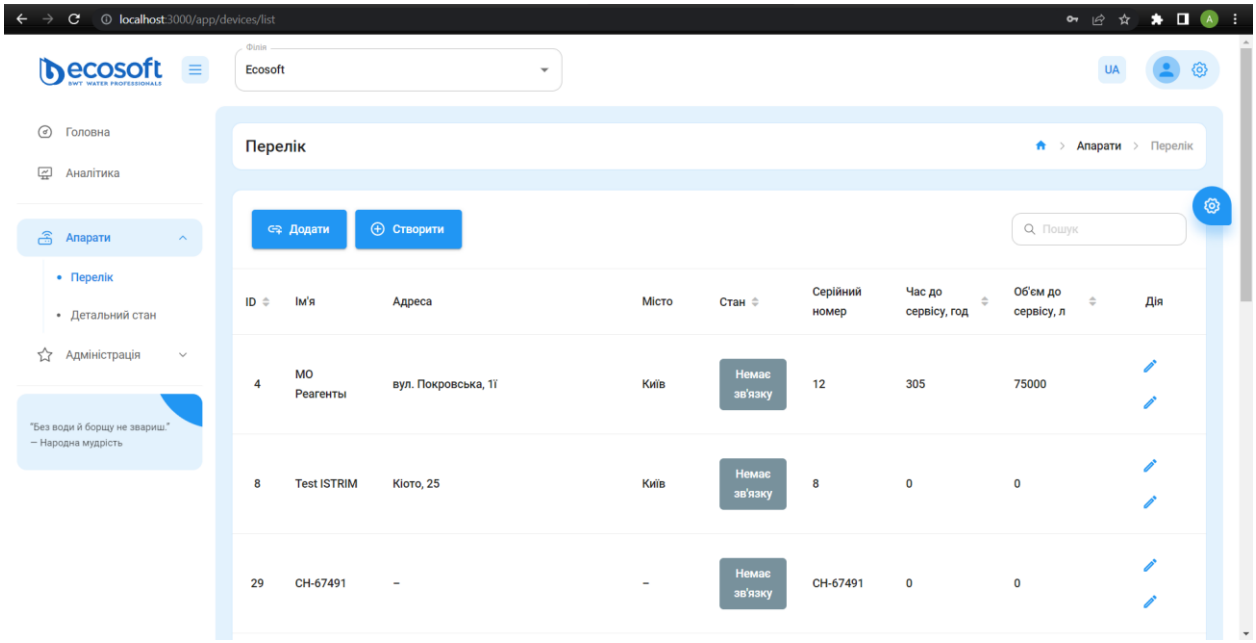


Рисунок 3.4 – Сторінка зі списком апаратів

Щоб створити апарат, потрібно натиснути кнопку «Створити» на сторінці зі списком апаратів. З'явиться модальне вікно з формою (рисунок 3.5). Користувачеві треба ввести дані в поля й натиснути кнопку «Зберегти».

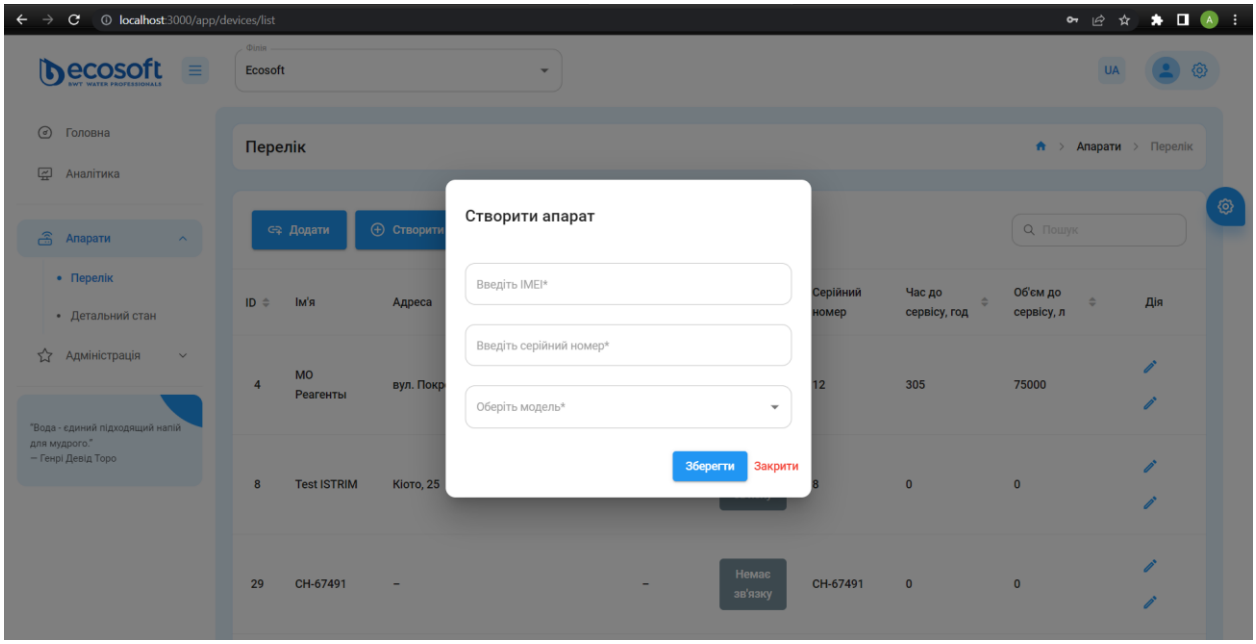


Рисунок 3.5 – Модальне вікно з формою для створення апарата

Щоб додати апарат, потрібно натиснути кнопку «Додати» на сторінці зі списком апаратів. З'явиться модальне вікно з формою (рисунок 3.6). Користувачеві треба ввести дані в поля й натиснути кнопку «Зберегти».

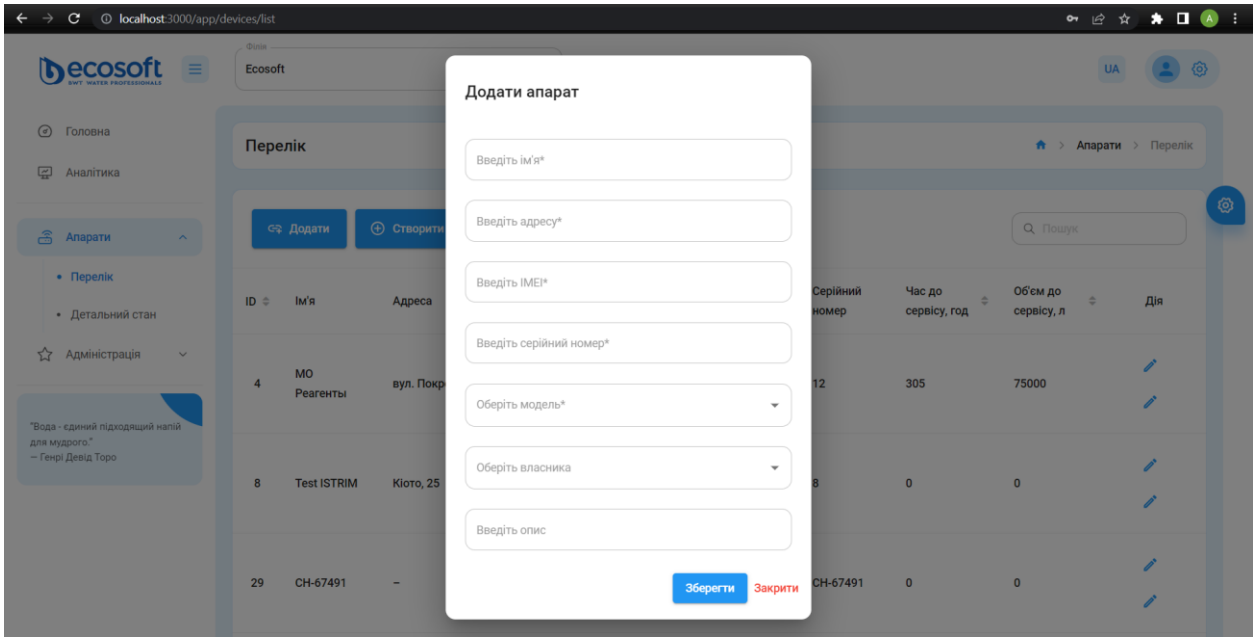


Рисунок 3.6 – Модальне вікно з формою для додавання апарата

Щоб оновити дані про апарат, потрібно натиснути іконку в колонці «Дія» у таблиці на сторінці зі списком апаратів. З’явиться модальне вікно з формою (рисунок 3.7). Користувачеві треба змінити принаймні одне поле й натиснути кнопку «Зберегти».

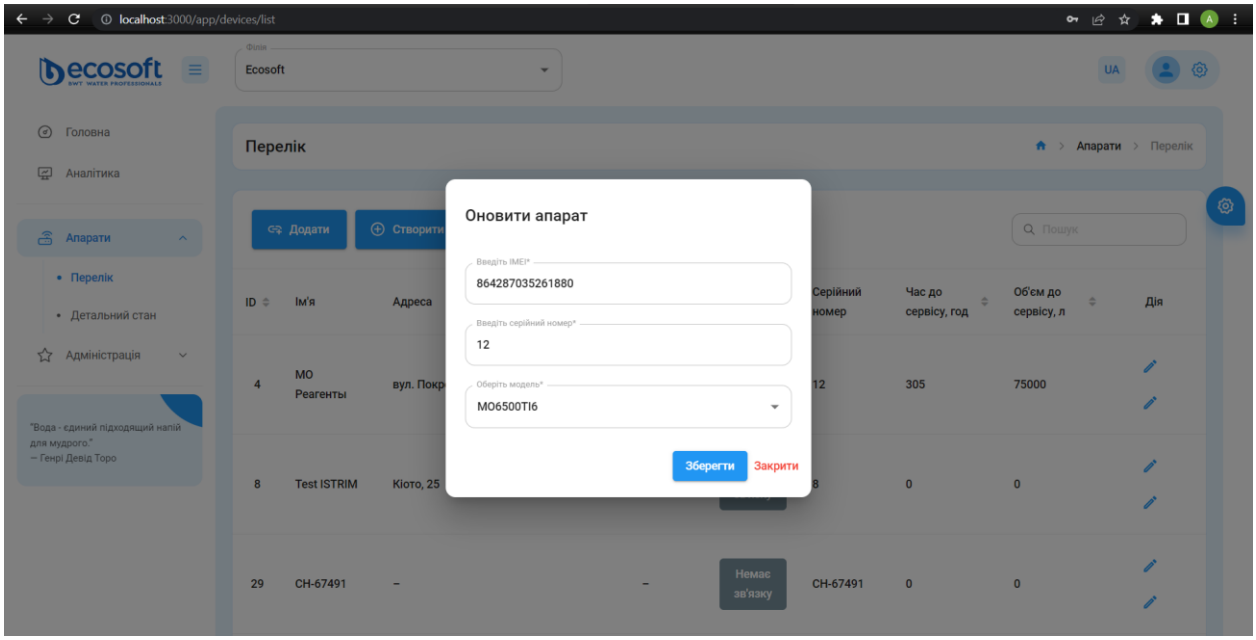


Рисунок 3.7 – Модальне вікно з формою для оновлення даних про апарат

Щоб знайти апарати за ключовими словами, треба в полі з заповнювачем «Пошук» вводити текст (рисунок 3.8). На кожне натискання кнопки користувачеві відображатиметься результат.

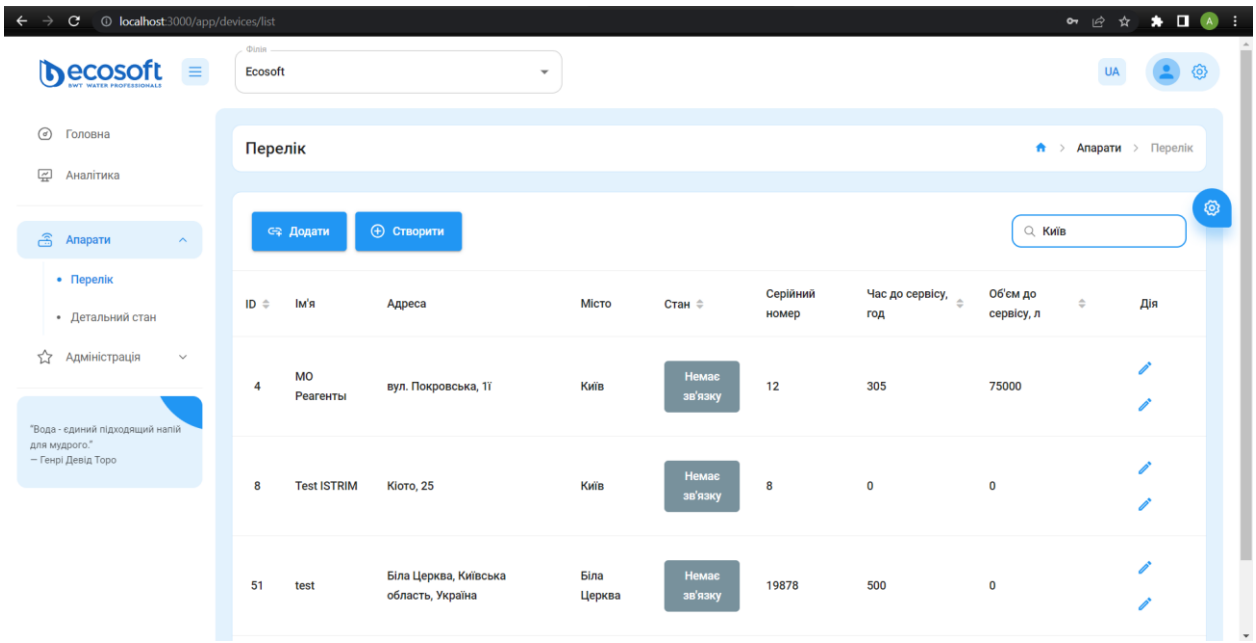


Рисунок 3.8 – Пошук апаратів за ключовими словами

Щоб перейти на сторінку з детальним станом апарата поточної філії (рисунок 3.9), необхідно натиснути вкладку «Апарати» в боковому меню й вибирати дочірню вкладку «Детальний стан».

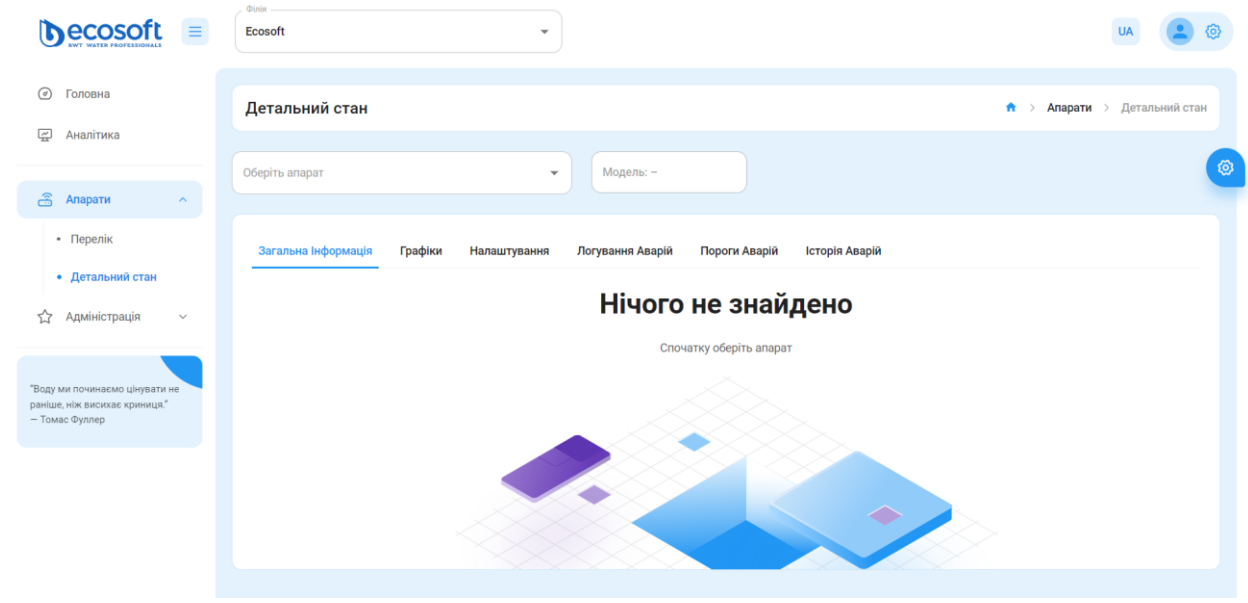


Рисунок 3.9 – Сторінка з детальним станом апарата

Щоб відобразити загальну інформацію про апарат (рисунок 3.10), потрібно натиснути поле «Оберіть апарат» та в списку, що випадає, обирати апарат.

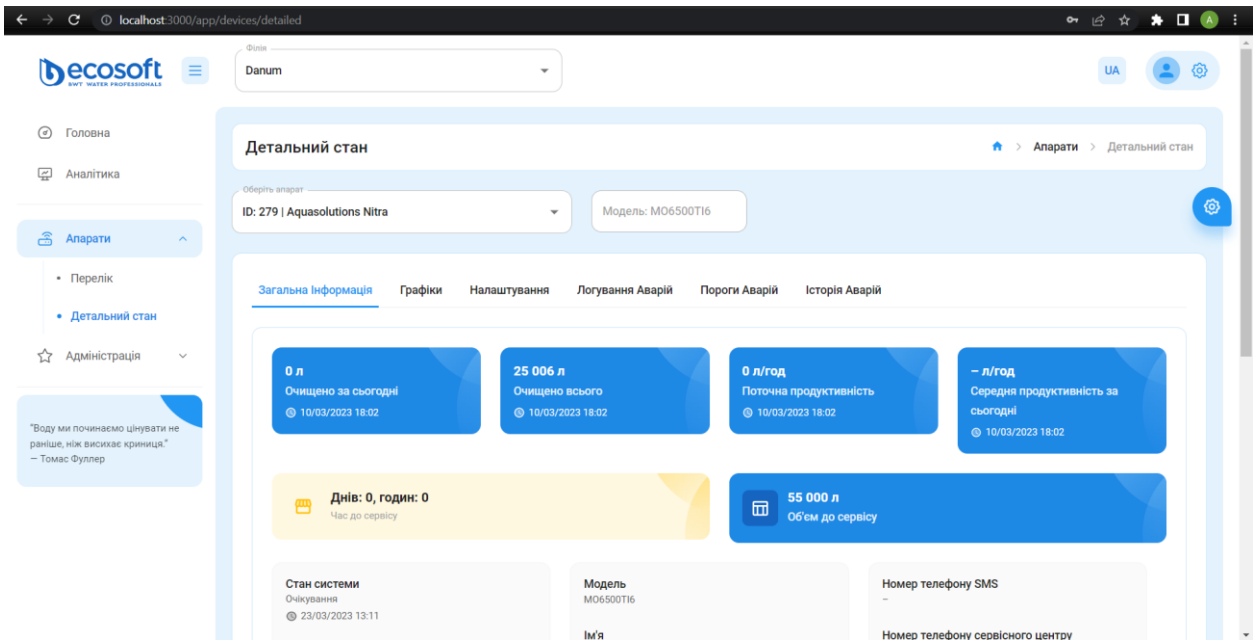


Рисунок 3.10 – Відображення загальної інформації про обраний апарат

Щоб відобразити статистичну інформацію про апарат у вигляді графіків, потрібно натиснути на таб «Графіки» на сторінці з детальним станом про апарат. Там необхідно вибрати проміжок дат, за який відображатиметься статистика. Користувач може це зробити, натиснувши на кнопку «Оберіть дату». Відкриється компонент для вибору дат (рисунок 3.11). Далі треба натиснути кнопку «Прийняти». Після цього дані відобразяться на графках (рисунок 3.12).

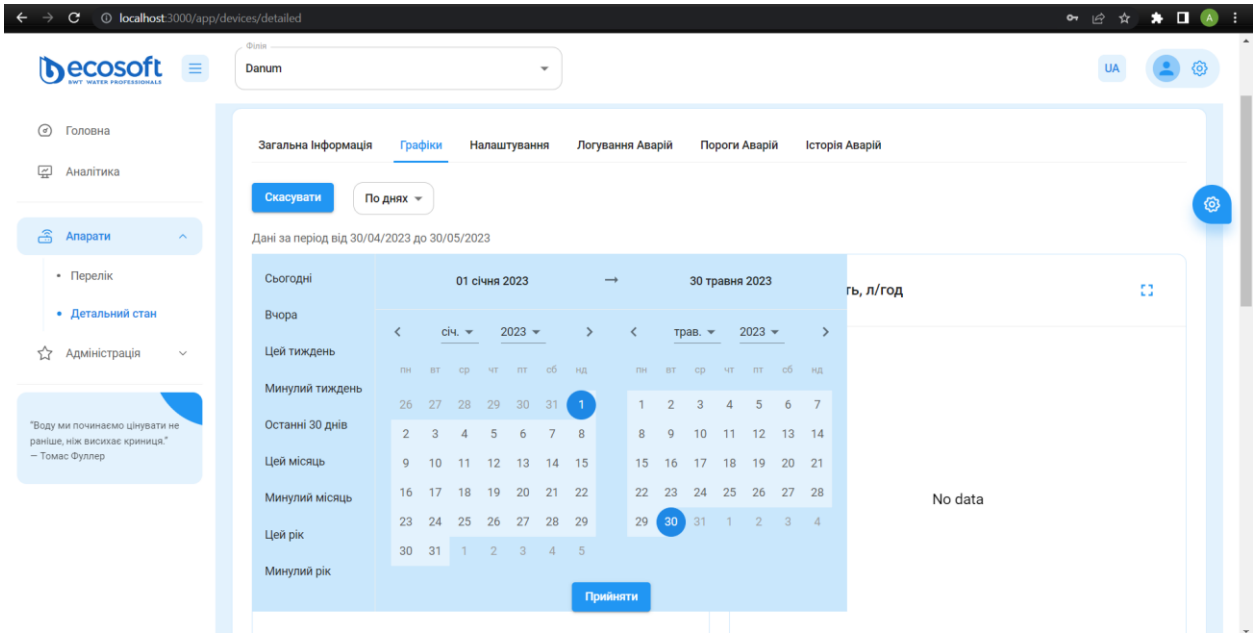


Рисунок 3.11 – Вибір проміжку дат

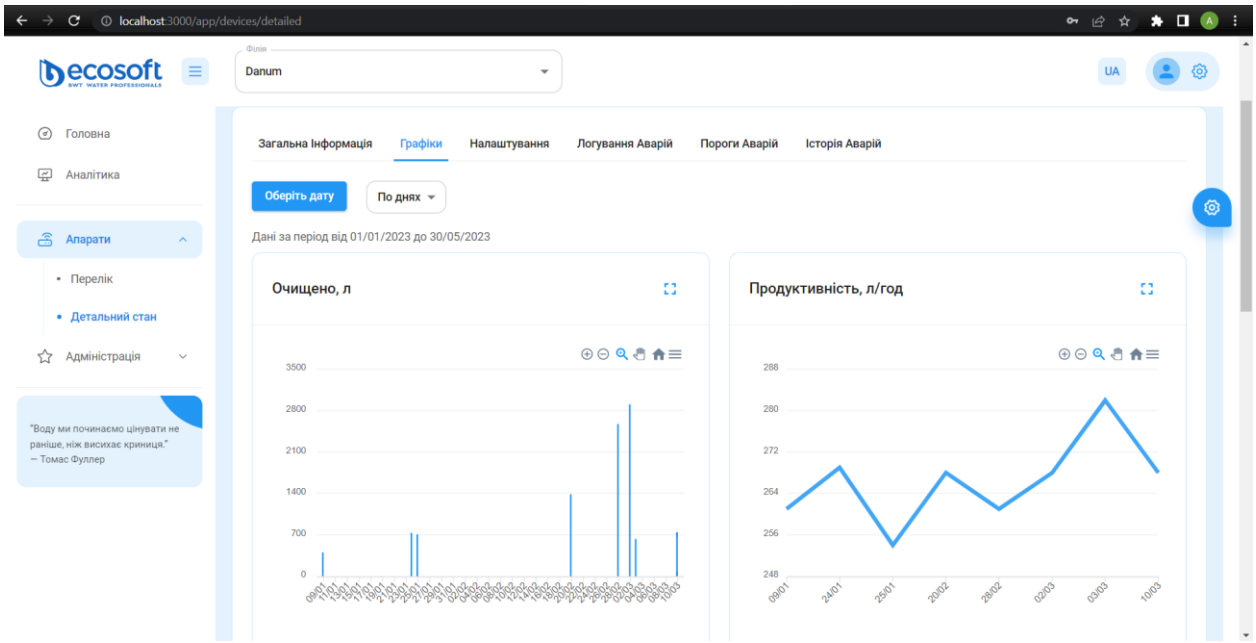


Рисунок 3.12 – Відображення статистичних даних апарата у вигляді графіків

Щоб перевірити доступні налаштування апарат (рисунок 3.13), потрібно натиснути на таб «Налаштування» на сторінці з детальним станом про апарат.

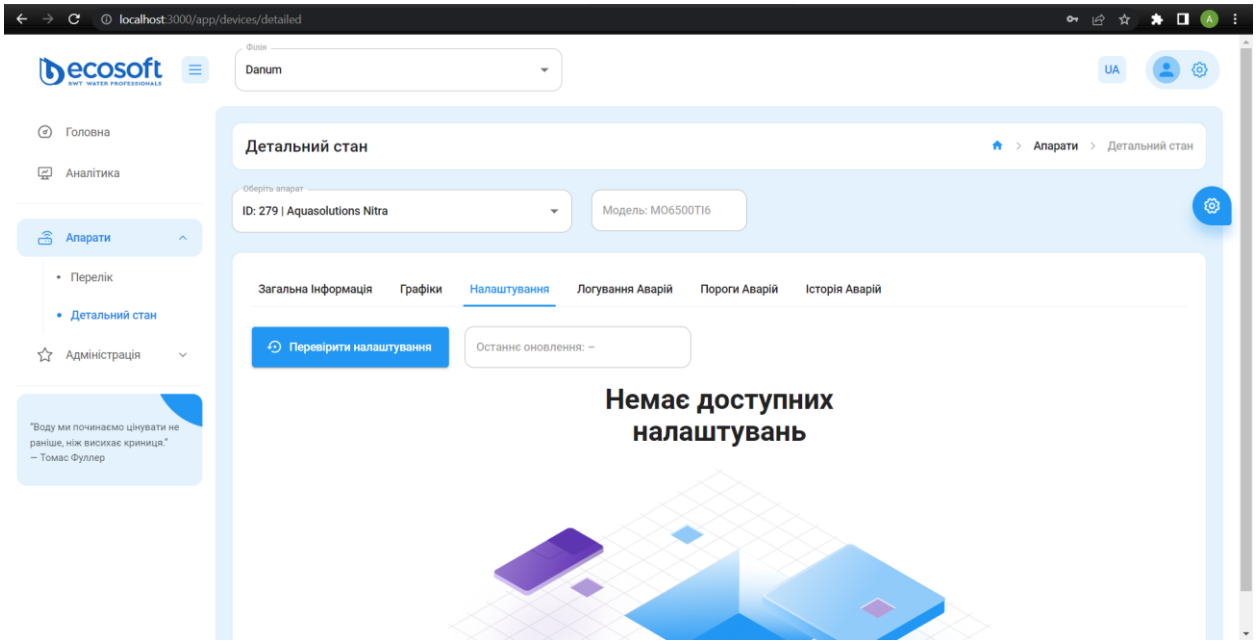


Рисунок 3.13 – Відображення налаштувань апарата

Щоб переглянути список аварій і те, чи вони увімкнені для логування вибраного апарата (рисунок 3.14), потрібно натиснути на таб «Логування Аварій» на сторінці з детальним станом про апарат.

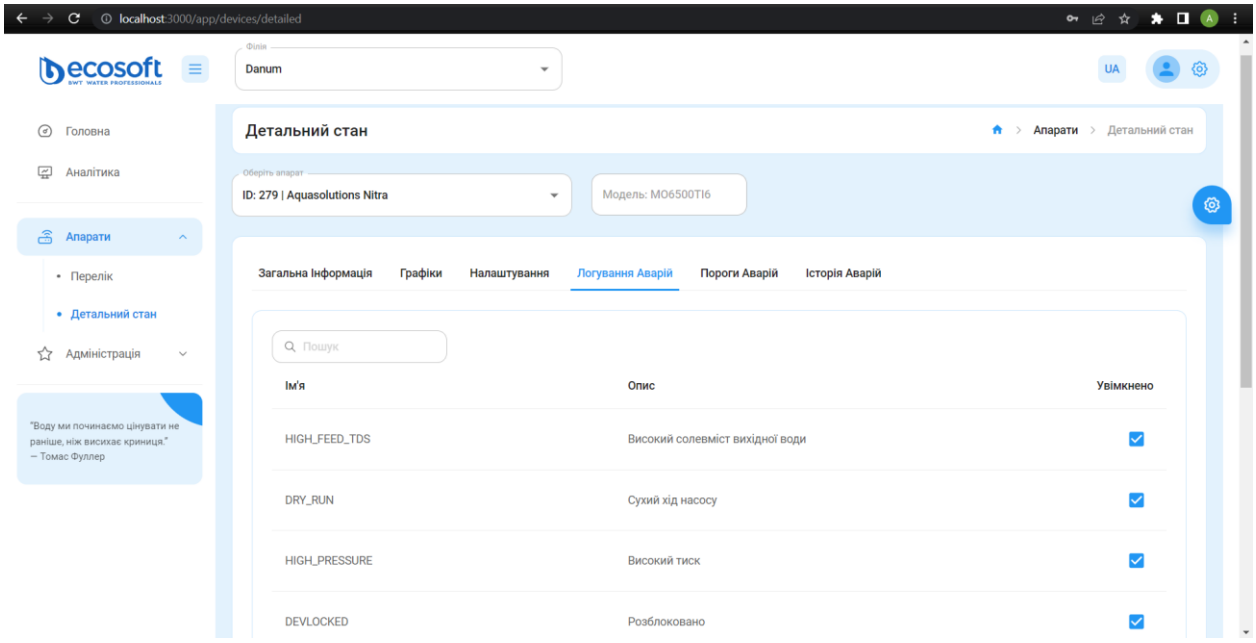


Рисунок 3.14 – Відображення списку можливих аварій апарата

Щоб переглянути список порогів аварій і те, чи вони увімкнені для вибраного апарата (рисунок 3.15), потрібно натиснути на таб «Пороги Аварій» на сторінці з детальним станом про апарат.

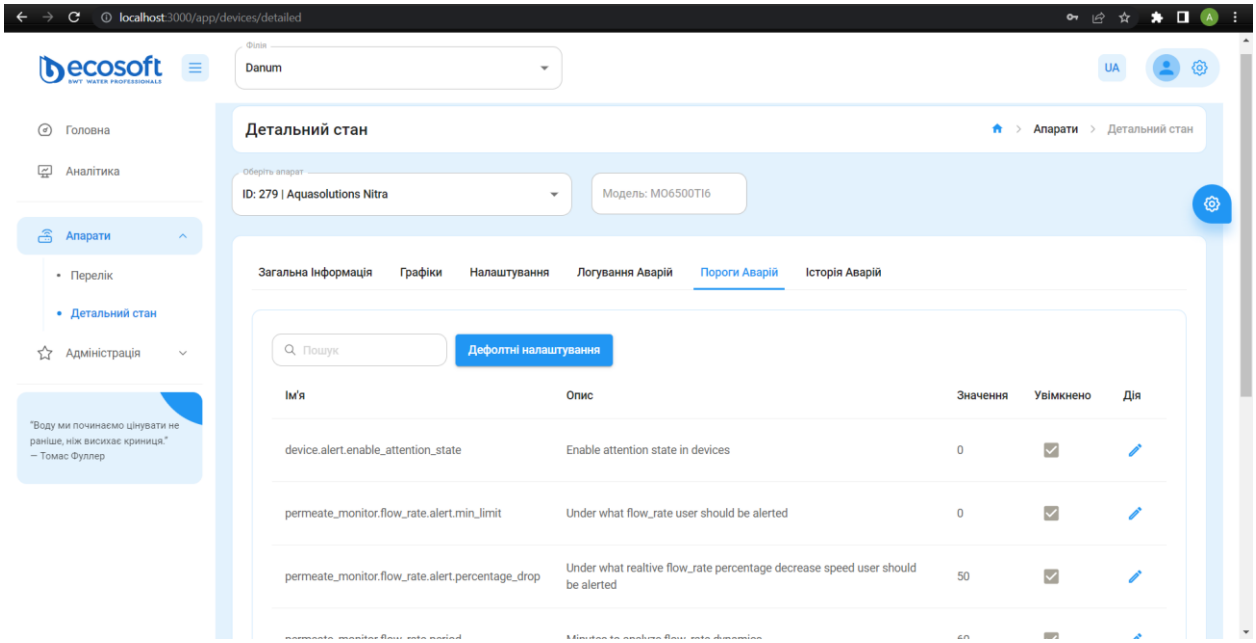


Рисунок 3.15 – Відображення списку порогів аварій

Щоб переглянути історію аварій апарата, потрібно натиснути таб «Історія Аварій» на сторінці з детальним станом про апарат. Далі натиснути кнопку «Оберіть дату». З'явиться компонент для вибору проміжку дат, список аварій за який має відобразитися в таблиці. Натискається кнопка «Прийняти». Відображається список аварій даного апарату (рисунок 3.16).

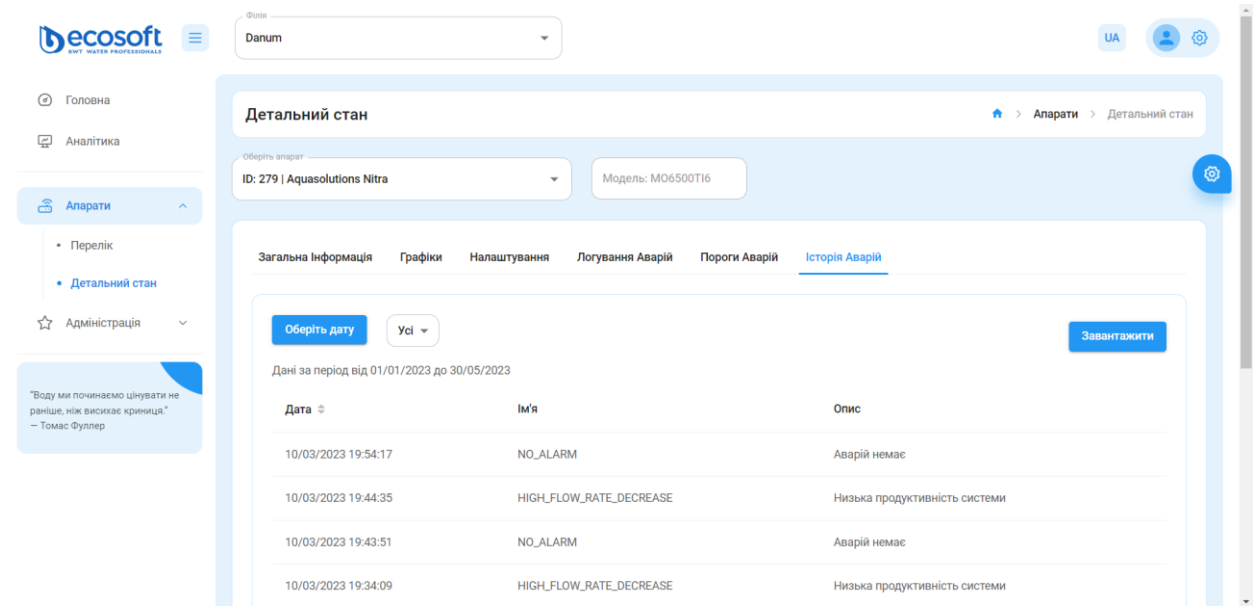
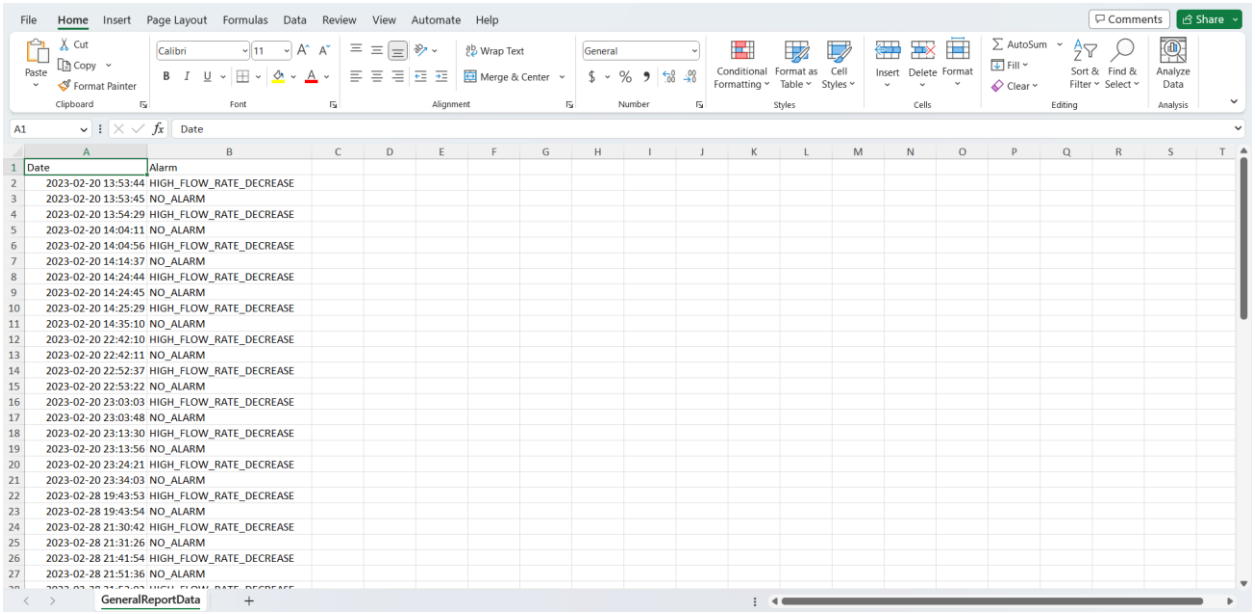


Рисунок 3.16 – Відображення списку аварій, які були в апарата

Щоб завантажити історію аварій апарата в Excel файл, потрібно натиснути кнопку «Завантажити» над таблицею з історією апаратів на сторінці з детальним станом про апарат. Завантажиться Excel файл (рисунок 3.17).



Date	Alarm
2023-02-20 13:53:44	HIGH_FLOW_RATE_DECREASE
2023-02-20 13:53:45	NO_ALARM
2023-02-20 13:54:29	HIGH_FLOW_RATE_DECREASE
2023-02-20 14:04:11	NO_ALARM
2023-02-20 14:04:56	HIGH_FLOW_RATE_DECREASE
2023-02-20 14:14:37	NO_ALARM
2023-02-20 14:24:44	HIGH_FLOW_RATE_DECREASE
2023-02-20 14:24:45	NO_ALARM
2023-02-20 14:25:29	HIGH_FLOW_RATE_DECREASE
2023-02-20 14:35:10	NO_ALARM
2023-02-20 22:42:10	HIGH_FLOW_RATE_DECREASE
2023-02-20 22:42:11	NO_ALARM
2023-02-20 22:52:37	HIGH_FLOW_RATE_DECREASE
2023-02-20 22:53:22	NO_ALARM
2023-02-20 23:03:03	HIGH_FLOW_RATE_DECREASE
2023-02-20 23:03:48	NO_ALARM
2023-02-20 23:13:30	HIGH_FLOW_RATE_DECREASE
2023-02-20 23:13:56	NO_ALARM
2023-02-20 23:24:21	HIGH_FLOW_RATE_DECREASE
2023-02-20 23:34:03	NO_ALARM
2023-02-28 19:43:53	HIGH_FLOW_RATE_DECREASE
2023-02-28 19:43:54	NO_ALARM
2023-02-28 21:30:42	HIGH_FLOW_RATE_DECREASE
2023-02-28 21:31:26	NO_ALARM
2023-02-28 21:41:54	HIGH_FLOW_RATE_DECREASE
2023-02-28 21:51:36	NO_ALARM

Рисунок 3.17 – Завантажений Excel файл із відображенням історії аварій

Щоб перейти на сторінку зі списком філій (рисунок 3.18), необхідно натиснути вкладку «Адміністрація» в боковому меню й вибирати дочірню вкладку «Філії».

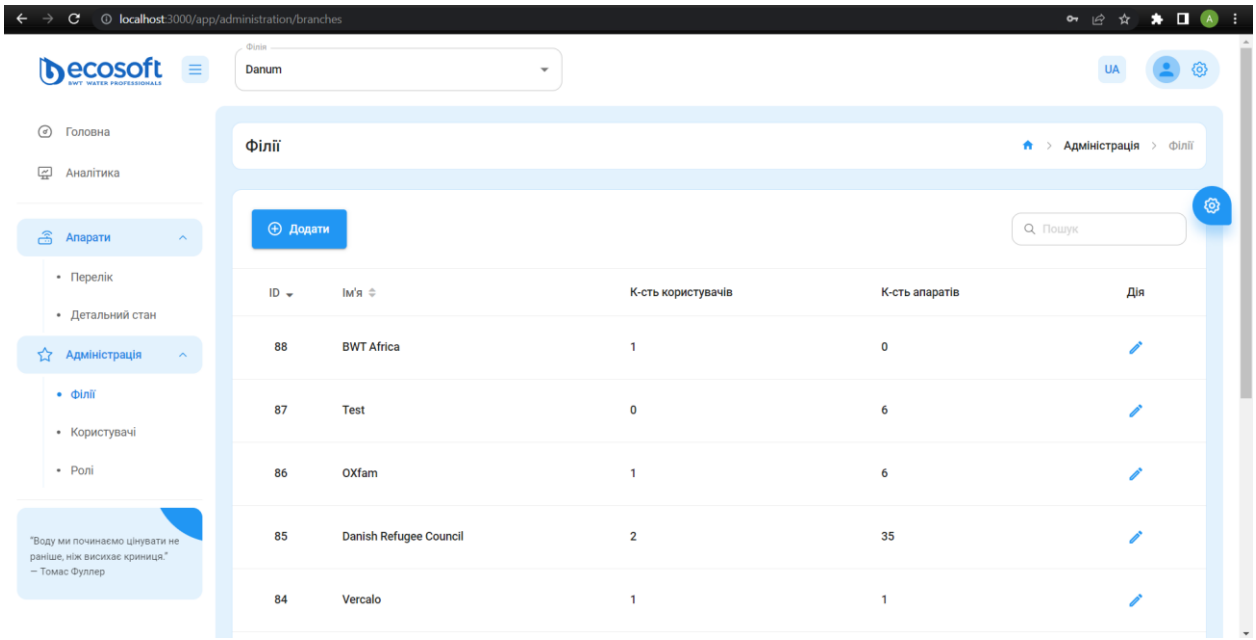


Рисунок 3.18 – Сторінка зі списком філій

Щоб додати філію, потрібно натиснути кнопку «Додати» на сторінці зі списком філій. З’явиться модальне вікно з формою (рисунок 3.19). Користувачеві треба ввести дані в поле й натиснути кнопку «Зберегти».

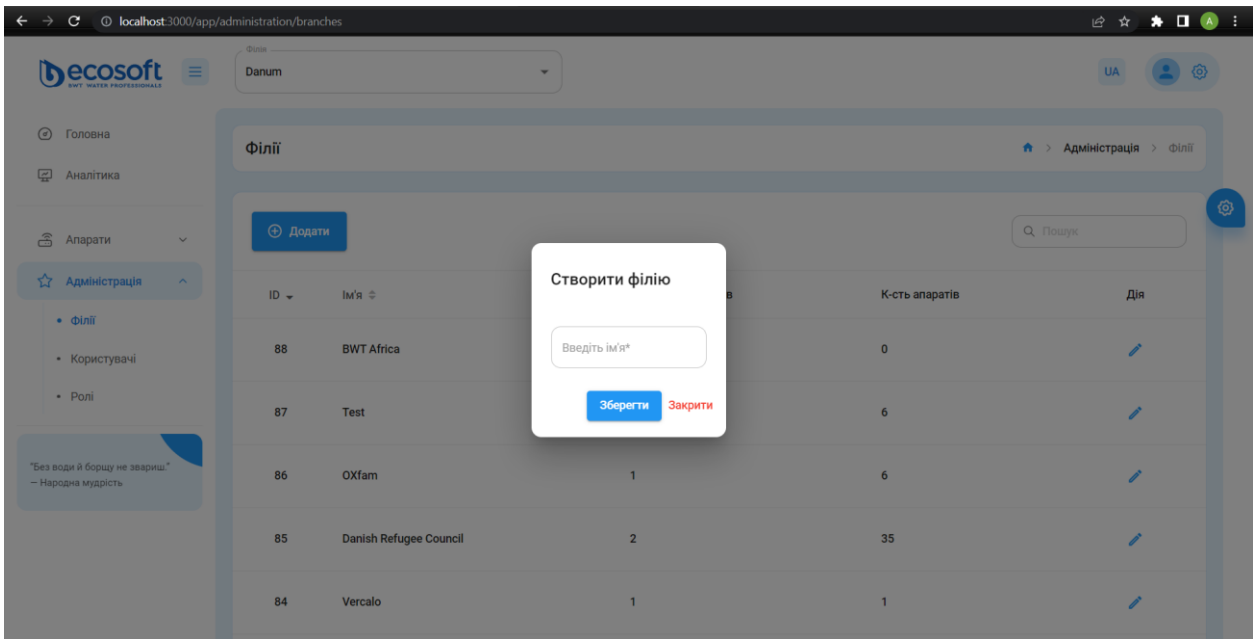


Рисунок 3.19 – Модальне вікно з формою для додавання філії

Щоб оновити дані про філію, потрібно натиснути іконку в колонці «Дія» у таблиці на сторінці зі списком філій. З’явиться модальне вікно з формою (рисунок 3.20). Користувачеві треба змінити значення поля й натиснути кнопку «Зберегти».

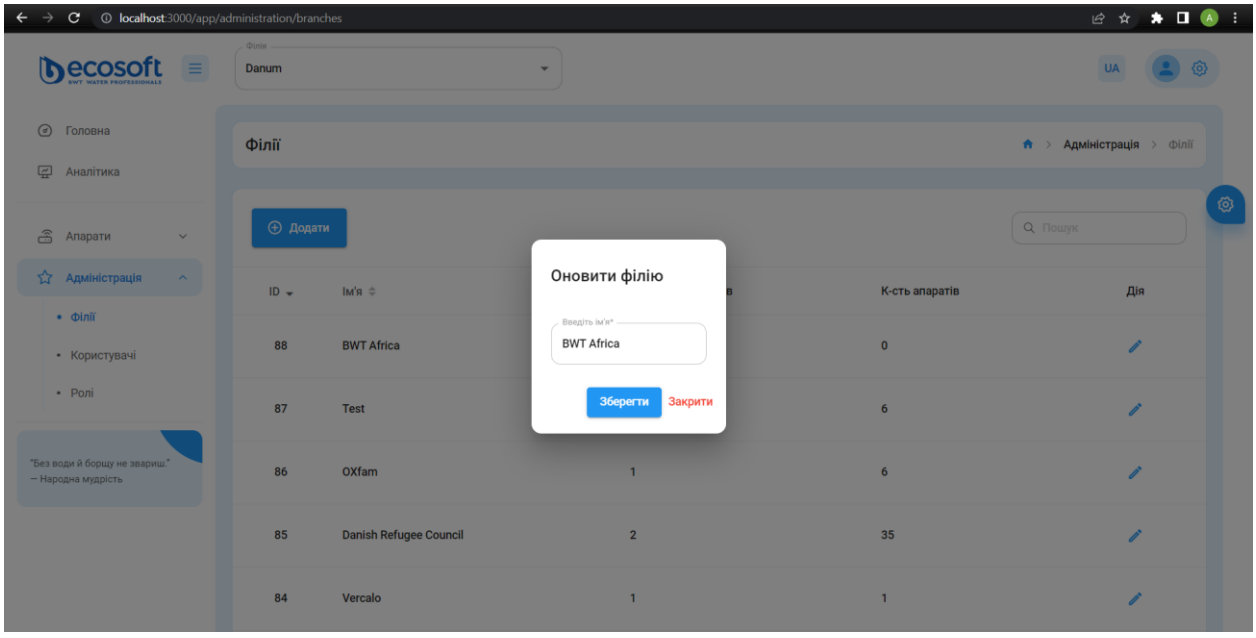


Рисунок 3.20 – Модальне вікно з формою для оновлення даних про філію

Щоб перейти на сторінку зі списком користувачів (рисунок 3.21), необхідно натиснути вкладку «Адміністрація» в боковому меню й вибирати дочірню вкладку «Користувачі».

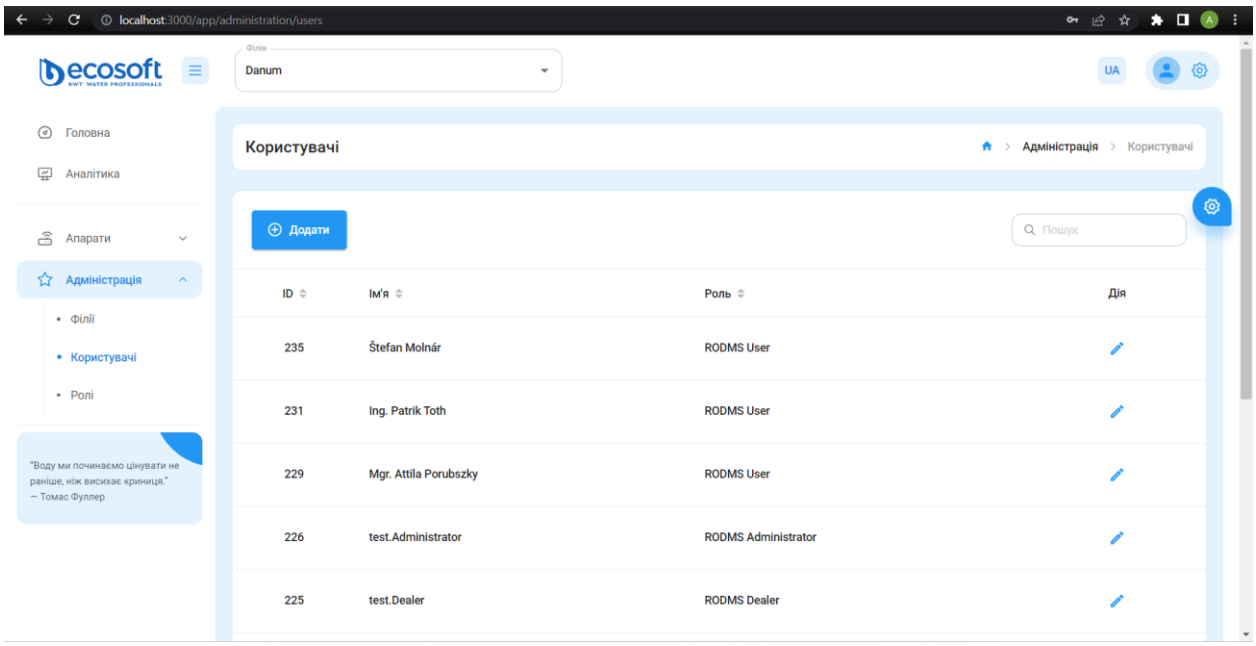


Рисунок 3.21 – Сторінка зі списком користувачів

Щоб додати філію, потрібно натиснути кнопку «Додати» на сторінці зі списком користувачів. З'явиться модальне вікно з формою (рисунок 3.22). Користувачеві треба ввести дані в поле й натиснути кнопку «Зберегти».

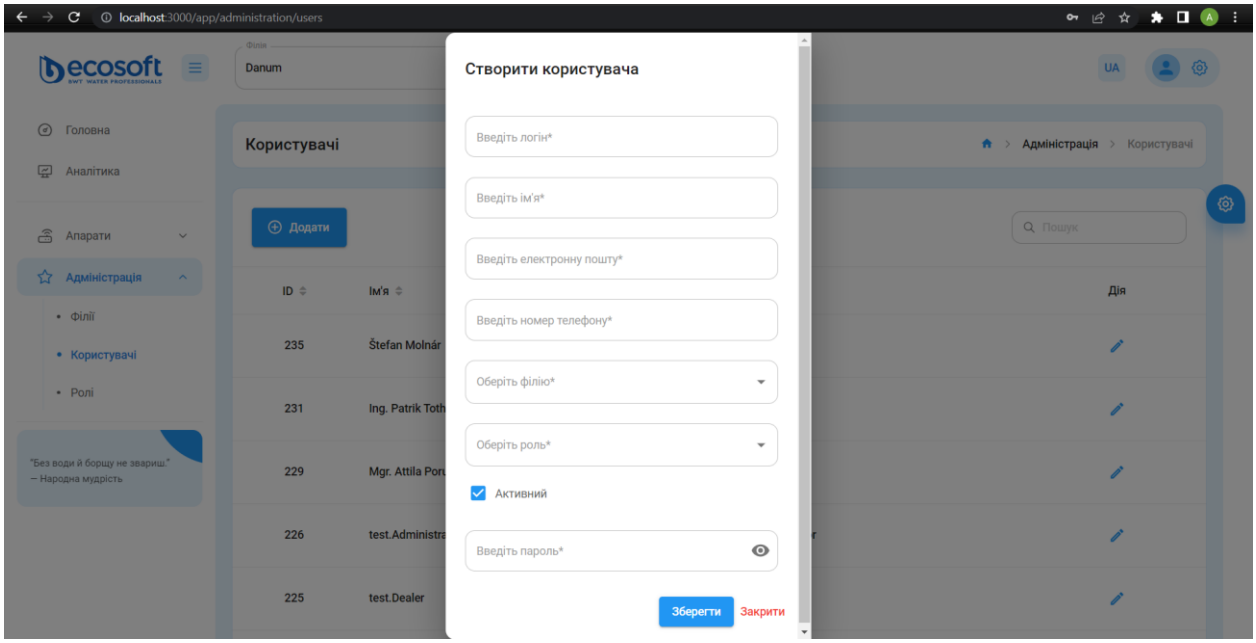


Рисунок 3.22 – Модальне вікно з формою для додавання користувача

Щоб оновити дані про користувача, потрібно натиснути іконку в колонці «Дія» у таблиці на сторінці зі списком користувачів. З’явиться модальне вікно з формою (рисунок 3.23). Користувачеві треба змінити значення принаймні одного поля й натиснути кнопку «Зберегти».

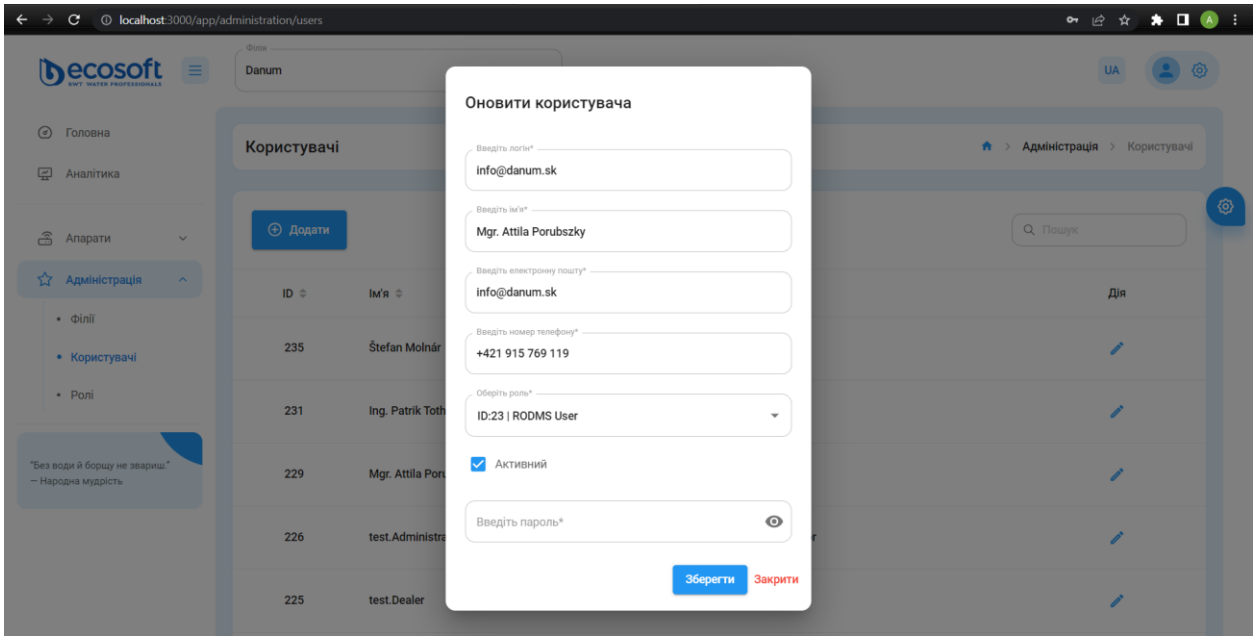


Рисунок 3.23 – Модальне вікно з формою для оновлення даних про користувача

Щоб перейти на сторінку зі списком ролей (рисунок 3.24), необхідно натиснути вкладку «Адміністрація» в боковому меню й вибирати дочірню вкладку «Ролі».

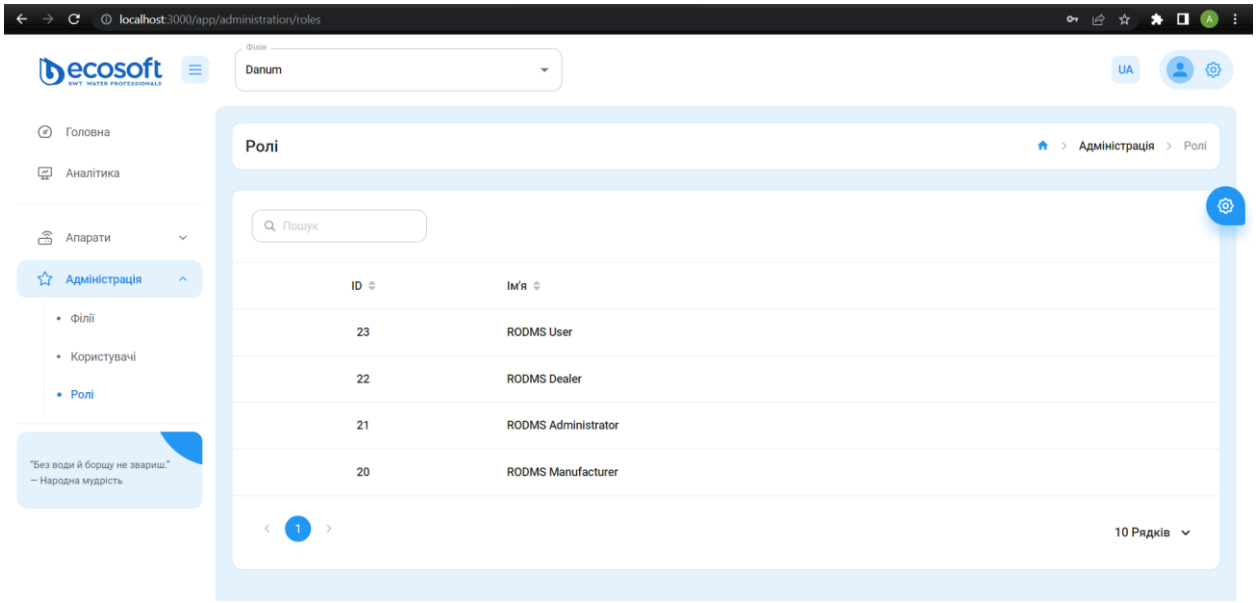


Рисунок 3.24 – Сторінка зі списком ролей

Щоб змінити колір інтерфейсу, необхідно натиснути на кнопку у формі краплі, що знаходиться на будь-якій сторінці справа. З'явиться бокове меню (рисунок 3.25). Для зміни кольору сторінок, треба натиснути на невибраний чекбокс у картці «Режим». Колір сторінки зміниться на протилежний (рисунок 3.26).

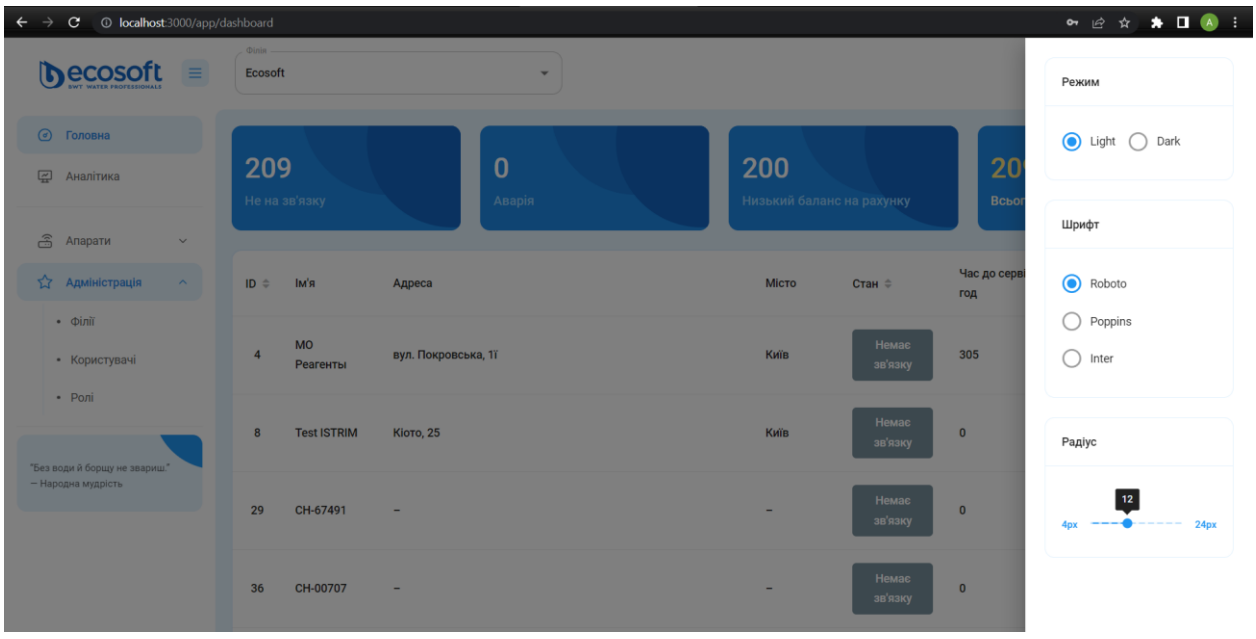


Рисунок 3.25 – Бокове меню для зміни вигляду сторінок

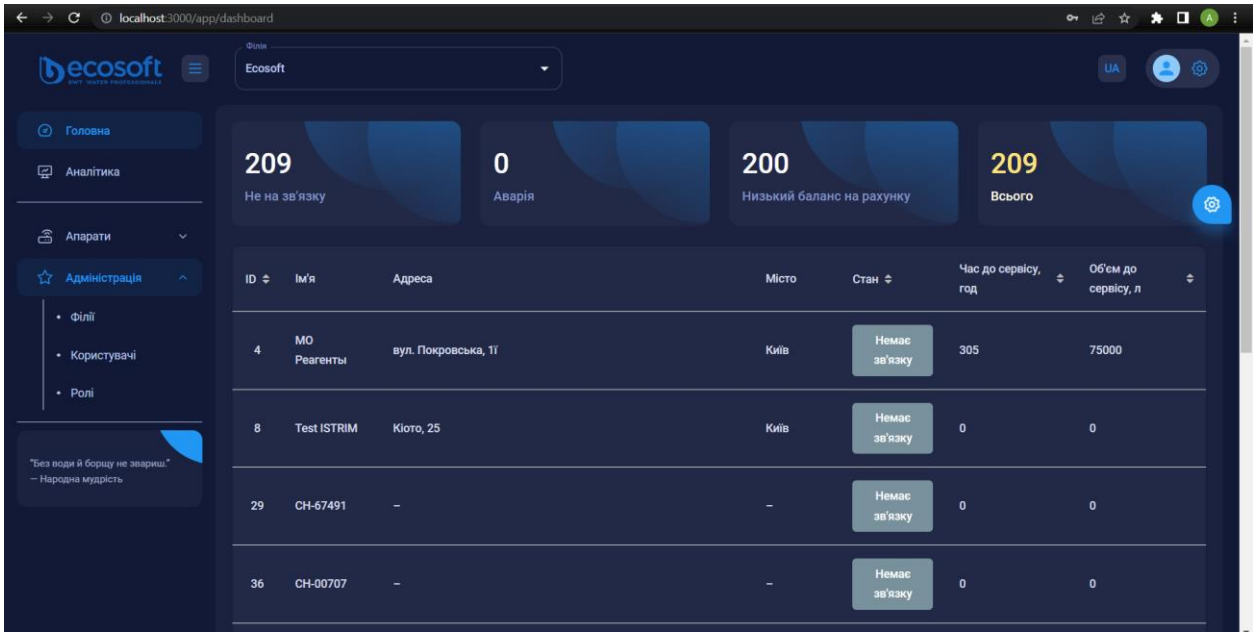


Рисунок 3.26 – Темний інтерфейс головної сторінки

Щоб змінити шрифт інтерфейсу, необхідно натиснути на кнопку у формі краплі, що знаходиться на будь-якій сторінці справа. З'явиться бокове меню. Для зміни шрифту, що використовується на сторінках, треба натиснути на невибраний чекбокс у картці «Шрифт». Колір сторінки зміниться на вибраний (рисунок 3.27).

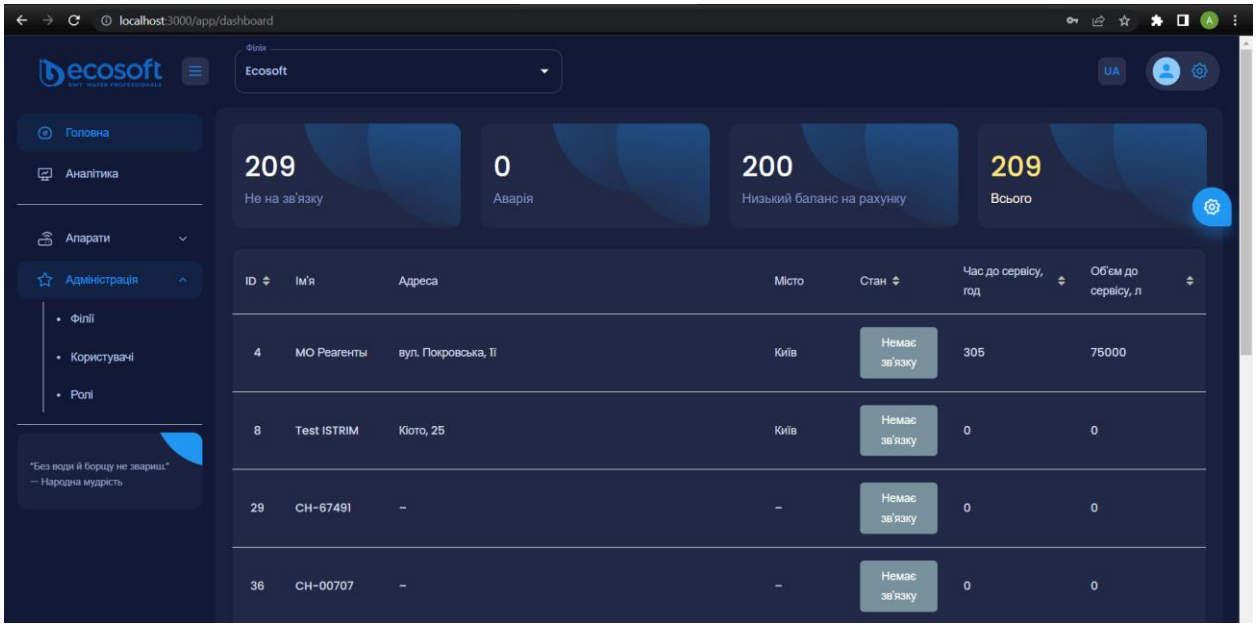


Рисунок 3.27 – Головна сторінка зі шрифтом Poppins

Щоб змінити мову інтерфейсу, необхідно натиснути на кнопку в хедері з кодом поточної мови. З’явиться випадаючий список. Для зміни мови, що використовується на сторінках, треба натиснути на невибраний списку. Мова сторінки зміниться на іншу (рисунок 3.28).

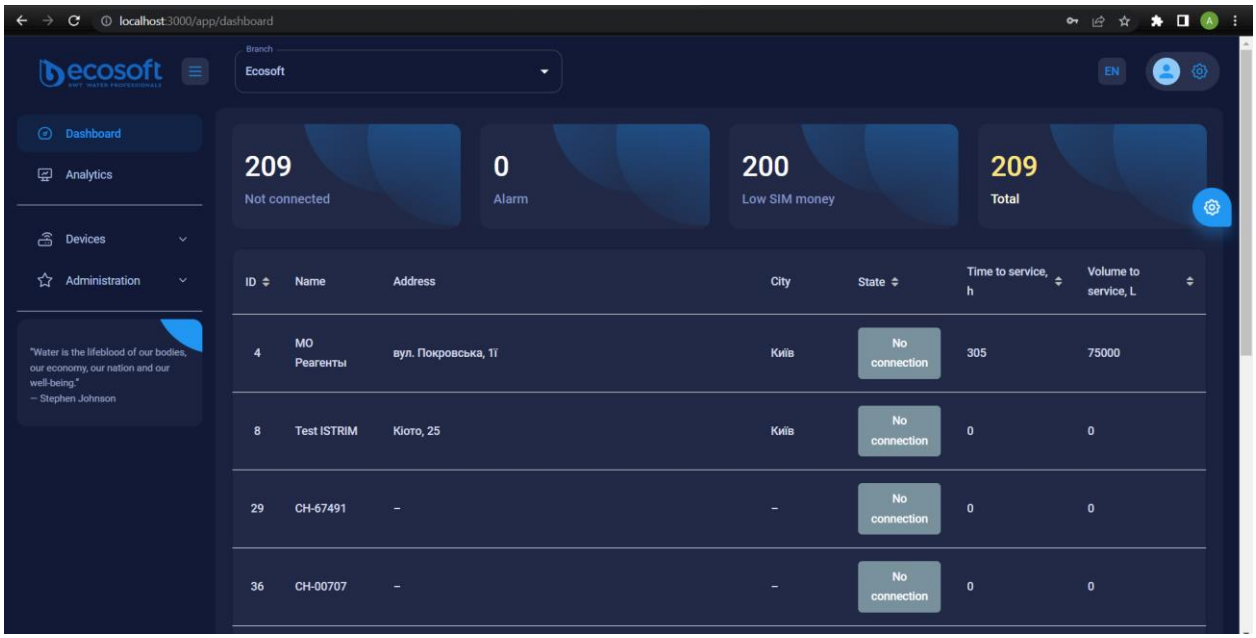


Рисунок 3.28 – Головна сторінка з інтерфейсом англійською мовою

Щоб змінити персональні дані, необхідно натиснути на кнопку в хедері з іконкою користувача та у вікні, що з’явиться, натиснути «Налаштування акаунту». З’явиться форма для оновлення профілю (рисунок 3.29). Користувачеві необхідно змінити принаймні одне поле та натиснути кнопку «Оновити Профіль».

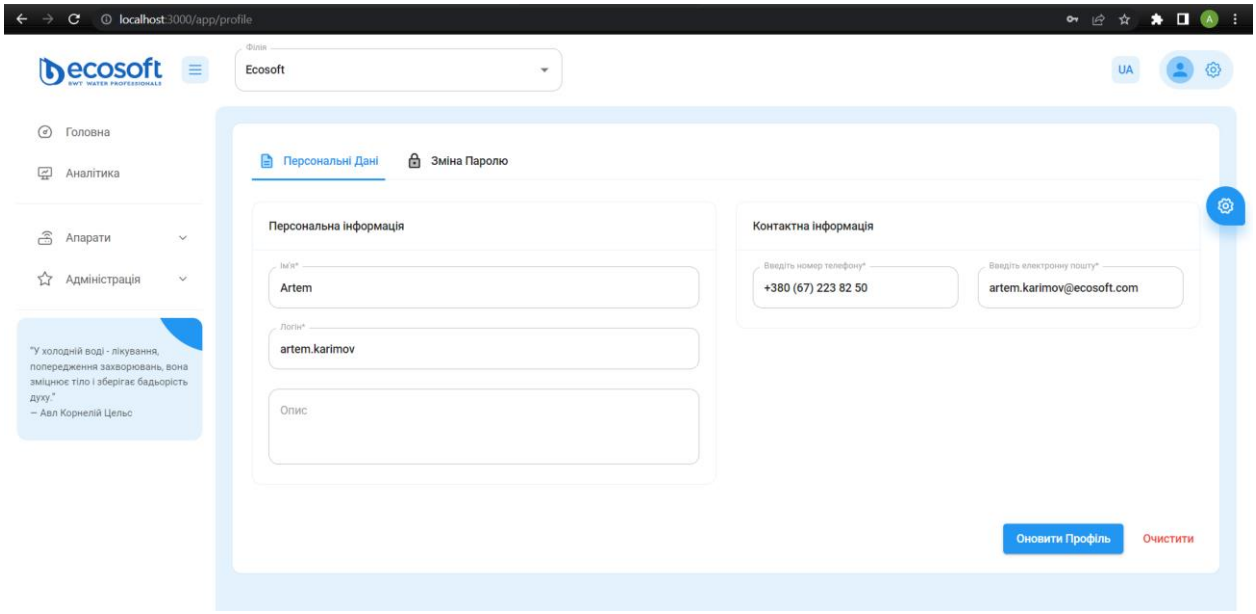


Рисунок 3.29 – Сторінка з формою для оновлення профілю

Щоб змінити пароль, необхідно натиснути на кнопку в хедері з іконкою користувача та у вікні, що з’явиться, натиснути «Налаштування акаунту» і перейти на таб «Зміна Паролю». З’явиться форма для зміни пароля (рисунок 3.30). Користувачеві необхідно заповнити всі поля та натиснути кнопку «Зміна Паролю».

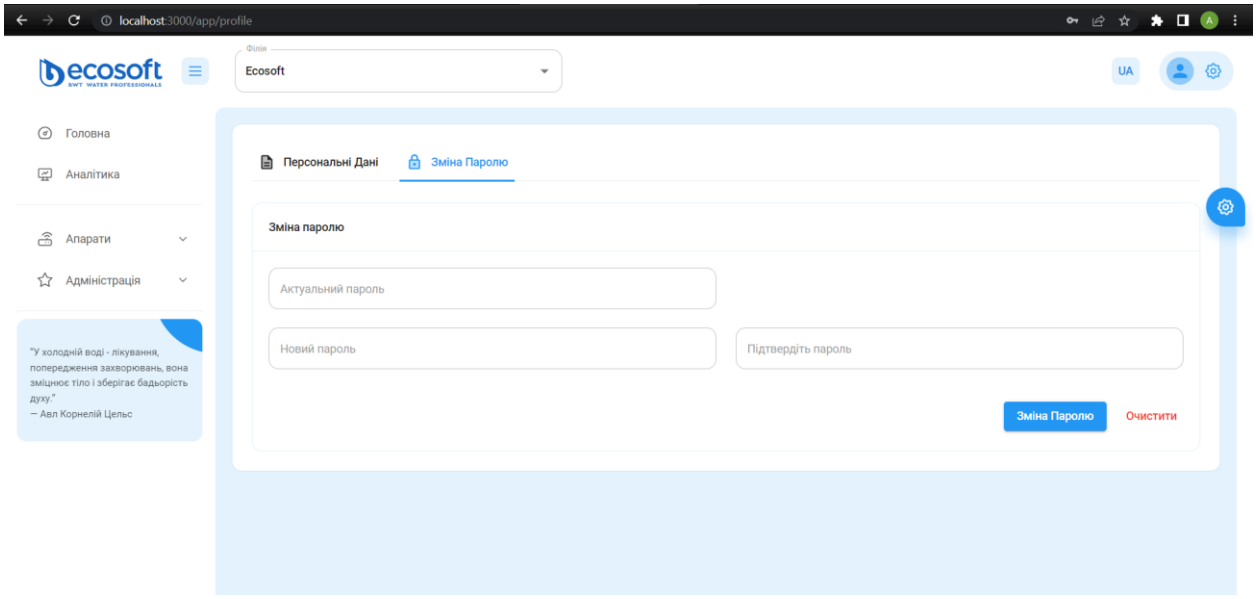


Рисунок 3.30 – Сторінка з формою для зміни пароля

Щоб вийти із застосунку, необхідно натиснути на кнопку в хедері з іконкою користувача та у вікні, що з’явиться, натиснути «Вихід» (рисунок 3.31). Користувач перенаправиться на сторінку логіну.

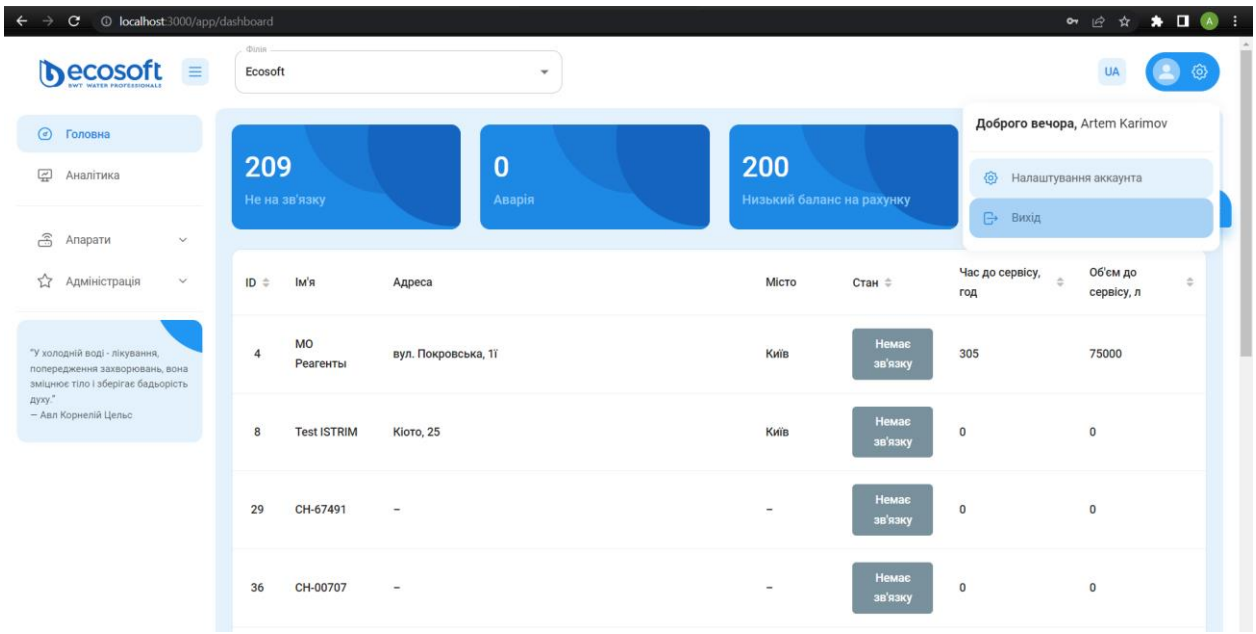


Рисунок 3.31 – Вихід із застосунку

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ ВІДДАЛЕНОГО КЕРУВАННЯ ТА МОНІТОРИНГУ РОБОТИ СИСТЕМ ОЧИЩЕННЯ ВОДИ

Графічний матеріал

КПІ.ІП-9619.045440.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

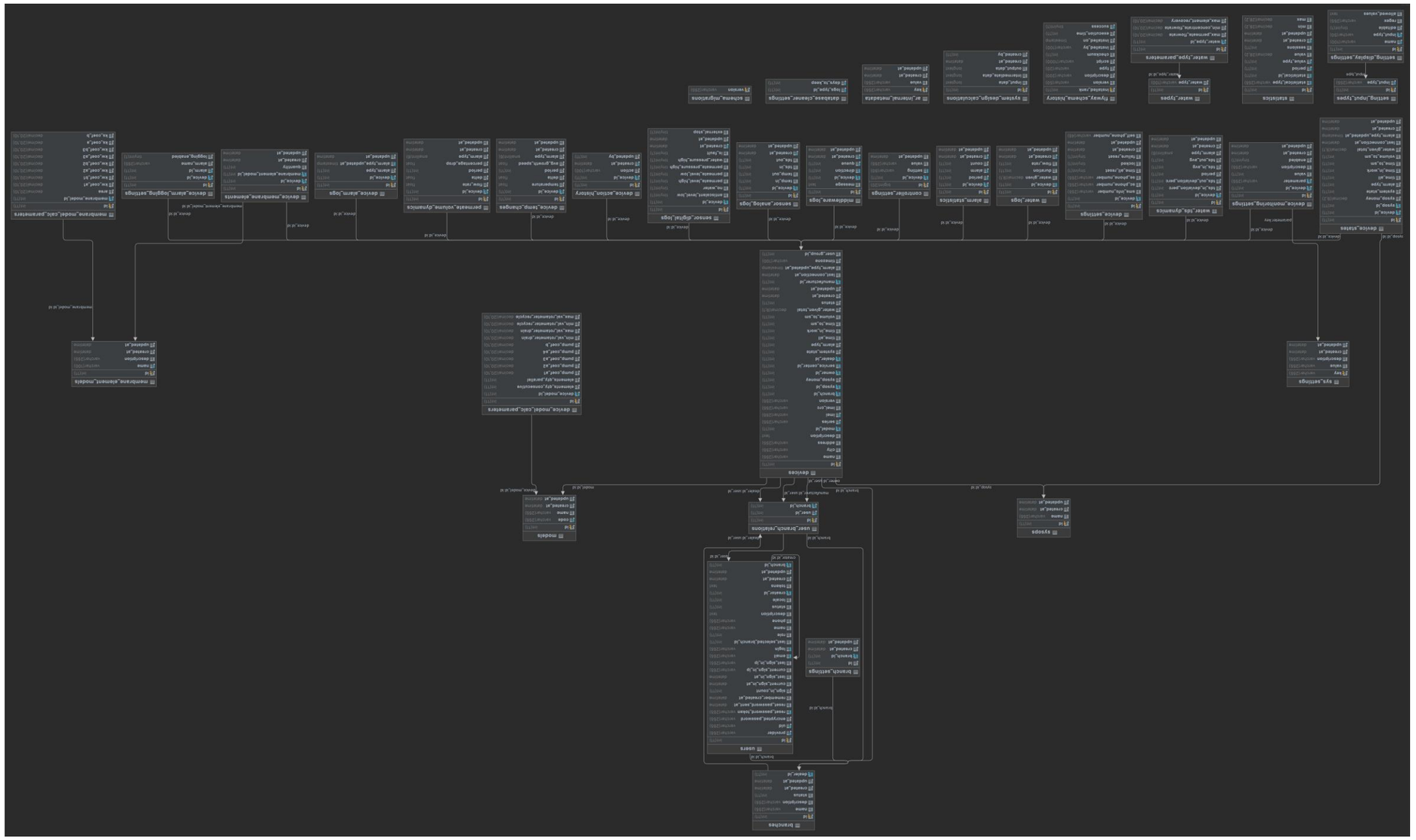
Виконавець:
_____ Артем КАРІМОВ

Київ – 2023

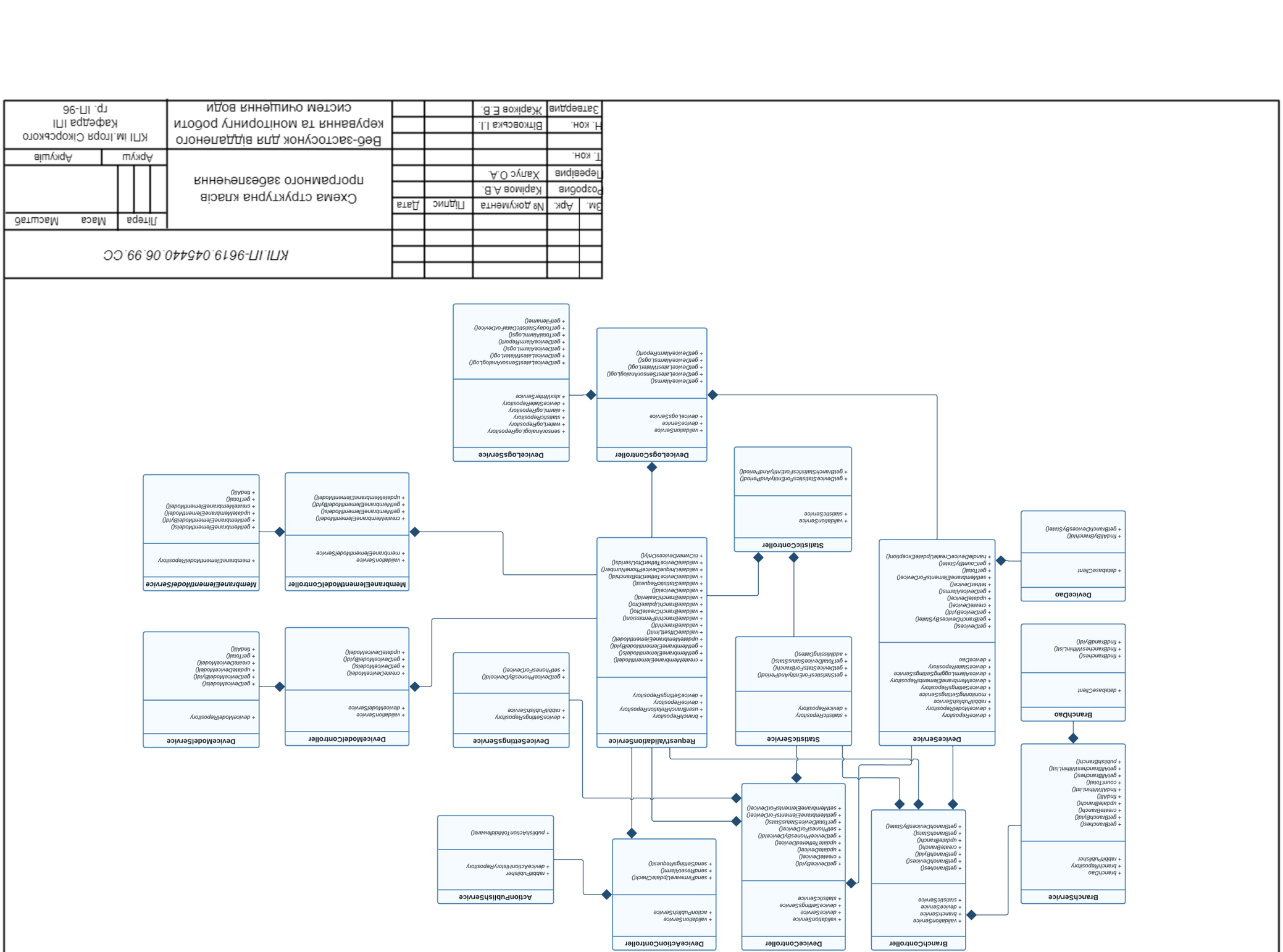
Зм.	Арк.	№ докумeнта	Пiдпис	Дата

КПІ.ІП-9619.045440.06.99.СБД

Розробив	Карпов А.В.			Схема бази даних	Літера	Маса	Масштаб
Перевірив	Хачус О.А.						
Т. кон.					Аркуш		Аркушів
Н. кон.	Бітковська І.І.			Веб-застосунок для віддаленого керування та моніторингу роботи системи очищення води	КПІ ім.Ігоря Сікорського кафедра ІПІ гр. ІП-96		
Затвердив	Жарікова Е.В.						

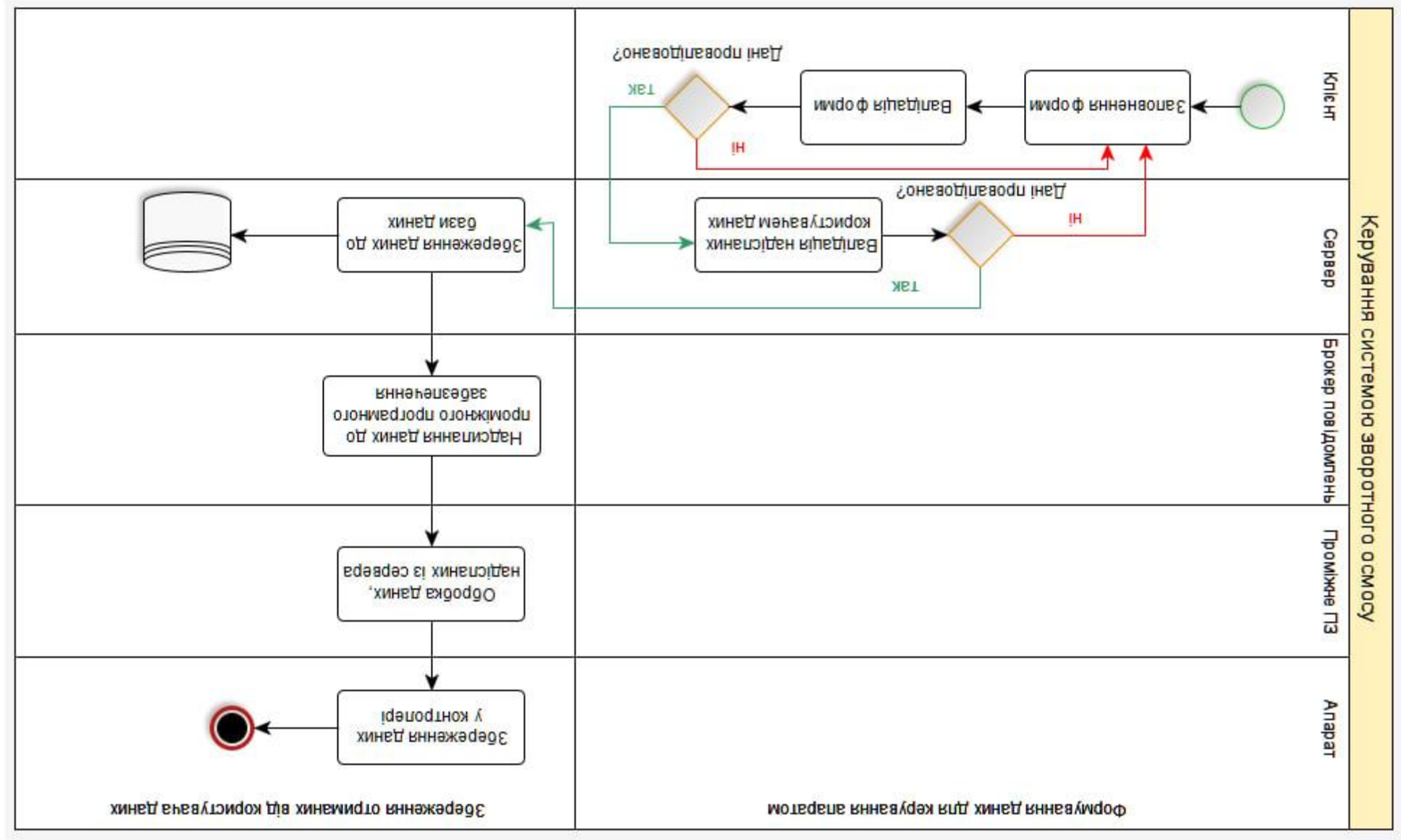


КПІ.ІП-9619.045440.06.99.СБД



КПІ.ІП-9619.045440.06.99.СС

Зм.	Арк.	№ докумeнта	Підпис	Дата	КПІ.ІП-9619.045440.06.99.СС		
Розробив		Кармов А.В.			Схема структура класів програмного забезпечення	Літера	Маса Масштаб
Перевірив		Харус О.А.					
Т. кон.						Аркуш	Аркушів
Н. кон.		Бітковська І.І.			Веб-застосунок для віддаленого керування та моніторингу роботи систем очищення води	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-96	
Затвердив		Жарікова Е.В.					



Зм.	Арк.	№ документа	Підпис	Дата	Схема бізнес-процесу керування системою зворотного осмосу	Літера	Маса	Масштаб
Розробив		Карімов А.В.						
Перевірив		Харус О.А.						
Т. контр.						Аркуш	Аркушів	
Н. контр.		Бітковська І.І.			Веб-застосунок для відділеного керування та моніторингу роботи систем очищення води	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІП-96		
Затвердив		Жарікова Е.В.						

КПІ.ІП-9619.045440.06.99.СБП