МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ
СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Циклова комісія з інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНИЙ
Голова _____
С.О. Сабанов

ВИПУСКНА РОБОТА МОЛОШОГО СПЕЦІАЛІСТА
РОЗРОБКА ВЕБСЕРВІСУ ТЕЛЕМЕТРІЇ ТА УПРАВЛІННЯ ВІДДАЛЕНОГО ПРИСТРОЮ З ВИКОРИСТАННЯМ REACT ТА PHP

Виконав
ст. гр. ІПЗ – 118к9 _____ Б.М. Михальський

Науковий керівник
ст. викл. _____ К.В. Дереза

Запоріжжя
2022

СТРУКТУРНИЙ ПІДРОЗДІЛ «ФАХОВИЙ КОЛЕДЖ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРАТ «ПВНЗ «ЗІЕІТ»

Циклова комісія з інформаційних технологій

ЗАТВЕРДЖУЮ

Голова

С.О. Сабанов

_____

__.__.__ р.

З А В Д А Н Н Я

НА ВИПУСКНУ РОБОТУ МОЛОДШОГО СПЕЦІАЛІСТА

Студенту гр. ІПЗ – 118к9

Спеціальності: «Інженерія програмного забезпечення»

Михальського Богдана Михайловича

(прізвище, ім’я, по батькові)

1. Тема: Розробка веб сервісу телеметрії та управління віддаленого пристрою з використанням React та PHP

затверджена наказом по коледжу № 09.2-20 від 04.03.22 р.

2. Термін здачі студентом закінченої роботи: __.__.__ р.

3. Перелік питань, що підлягають розробці:

1. Розглянути предметну область
2. Розглянути існуючі аналоги та сформувати задачі проекту
3. Визначити та оглянути стек технологій
4. Визначити структуру проекту
5. Розробити програму
6. Провести тестування
7. Оформити інструкції до розгортки
8. Оформити результати роботи у вигляді звіту

4. Календарний графік підготовки випускної роботи молодшого спеціаліста

№ етапу	Зміст	Терміни виконання	Готовність по графіку (%), підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Формулювання (корегування) теми випускної роботи молодшого спеціаліста та збір практичного матеріалу за темою випускної роботи	17.01.22-26.02.22		
2	І атестація І розділ випускної роботи молодшого спеціаліста	28.03.22-02.04.22		
3	ІІ атестація ІІ розділ випускної роботи молодшого спеціаліста	10.05.22-14.05.22		
4	ІІІ атестація ІІІ розділ випускної роботи молодшого спеціаліста, висновки та рекомендації, додатки, реферат	30.05.22-04.06.22		
5	Перевірка випускної роботи програмою «Антиплагіат»	30.05.22-18.06.22		
6	Доопрацювання випускної роботи молодшого спеціаліста, підготовка презентації, отримання відгуку керівника і рецензії	06.06.22-11.06.22		
7	Попередній захист випускної роботи молодшого	14.06.22-18.06.22		
8	Подача випускної роботи молодшого спеціаліста на кафедру	за 3 дні до захисту		
9	Захист випускної роботи молодшого спеціаліста	20.06.22-25.06.22		

Керівник _____ К.В. Дереза

(підпис) (прізвище та ініціали)

«____» _____ 2022 р.

Завдання отримав до виконання _____ Б.М. Михальський

(підпис студента) (прізвище та ініціали)

«____» _____ 2022 р.

РЕФЕРАТ

Випускна робота молодшого спеціаліста містить * сторінок, * таблиць, * рисунки, * формул, * лістингів, * бібліографічних посилань, * додаток.

Метою роботи є розробка веб сервісу для телеметрії та управління віддаленим пристроєм за допомогою сучасних технологій розробки веб сервісів.

Об’єктом дослідження є застосування сучасних веб та клієнт-серверних технологій для створення веб сервісів телеметрії.

Предметом дослідження є веб сервіс для управління віддаленими пристроями та телеметрії.

Здійснено детальний огляд предметної області та популярних аналогів. Виявлено, що тема телеметрії і віддаленого управління пристроями є актуальною, а розробка веб сервісу для телеметрії і роботи з віддаленими пристроями є доцільною. Проект реалізовано за допомогою таких засобів, як TypeScript, PHP, MySQL, HTML, CSS, JavaScript. Здійснено проектування моделі предметної області, програмування сутностей та алгоритмів за принципами об’єктно-орієнтованого програмування та побудови клієнт-серверної архітектури за сучасними стандартами.

Розроблений програмний продукт є легким у використанні, має привабливий та інтуїтивно зрозумілий дизайн інтерфейсу. Веб сервіс дозволяє легко керувати віддаленими пристроями та отримувати наглядну інформацію про дані пристрою.

ТЕЛЕМЕТРІЯ, КЛІЄНТ, PHP, ВЕБСЕРВІС, СЕРВЕР, TYPESCRIPT, HTML, JAVASCRIPT, CSS, MYSQL, РОЗУМНИЙ ПРИСТРІЙ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Повна назва	Пояснення/переклад
СУБД	Система Управління Базами Даних	Комплекс взаємопов’язаних програм для доступу та управління даними
HTML	HyperText Markup Language	Мова розмітки гіпертексту
CSS	Cascading Style Sheets	Каскадна таблиця стилів
ООП	Об’єктно-Орієнтоване Програмування	Парадигма програмування, основана на взаємодії об’єктів
ПЗ	Програмне Забезпечення	Комплекс програм та компонентів для їх роботи
БД	База Даних	Сукупність організованих даних
SPA	Single Page Application	Веб застосунок, який використовує єдиний HTML документ як оболонку для усіх сторінок і організуючий взаємодію з користувачем через динамічно дованатажуючі HTML, CSS, JS файли
REST	Representational State Transfer	Передача самоописуваного стану
API	Application Programming Interface	Програмний інтерфейс засосунку
JSON	JavaScript Object Notation	Структура даних об’єкту у мові програмування JavaScript

ВСТУП

У повсякденному житті ми кожен день користуємось електронними пристроями. У сучасному світі з кожним днем пристрої удосконалюють, роблячи їх ще більше зручними і функціональними. Для керування пристроєм зазвичай використовуються кнопки, які розташовані на самому приладі, а для індикації стана роботи використовують різні шкали та діоди, доступ до яких, іноді, може бути ускладненим. Однією з тенденцій є створення так званих смарт пристроїв, якими можна керувати видалено, зчитувати дані їх роботи. Для цього використовують веб сервіси, які дозволяють як керувати пристроєм, назначати графік роботи і відображати отримані дані, що робить налаштування і моніторинг роботи пристроїв набагато легшим і ефективнішим .

Для цього створюють веб інтерфейс користувача, який дозволяє легко і швидко підключити пристрій. У цьому інтерфейсі наглядно відображаються дані про роботу пристрою, візуалізуючи інформацію за допомогою графіків, таблиць, діаграм. Цей інтерфейс дозволяє зручно керувати самим пристроєм, змінювати режим чи його стан, створювати розклад роботи пристрою. Також пристрій може надіслати повідомлення про якусь технічну несправність, що може допомогти запобігти нещасним випадкам.

Із вищесказаного випливає, що веб сервіс телеметрії та керування пристроями може значно полегшити процес налагодження, моніторингу стану, керування пристрою, завдяки зручному інтерфейсу користувача.

Метою роботи буде створення веб сервісу для роботи з віддаленими пристроями.

РОЗДІЛ 1
ОГЛАД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність теми

На сьогоднішній день людина використовує різноманітні електропристрої у різних сферах життя. Більшість пристроїв керується власноруч, що вимагає постійного перебування людини поруч із приладом. Також доступ до деяких кнопок чи індикаторів стає складним через розташування пристрою.

Саме для покращення взаємодії з пристроями були створені спеціальні пульти, які посилали команди пристроям, а потім і програмні засоби для керування та отримання показників з пристрою. Одним з таких засобів є «веб додаток». Саме так компанія Dropcam у 2010 році представила камеру відеоспостереження, з можливістю онлайн доступу до камери і збереженням записів на хмарне сховище. На момент початку 2022 року тема є актуальною. Аналітичні дані компанії GreyB щодо ринку смарт пристроїв з 2016 року по 2022 рік, а також прогноз на 2023 і 2024 роки [1] наведено у рисунку 1.1.

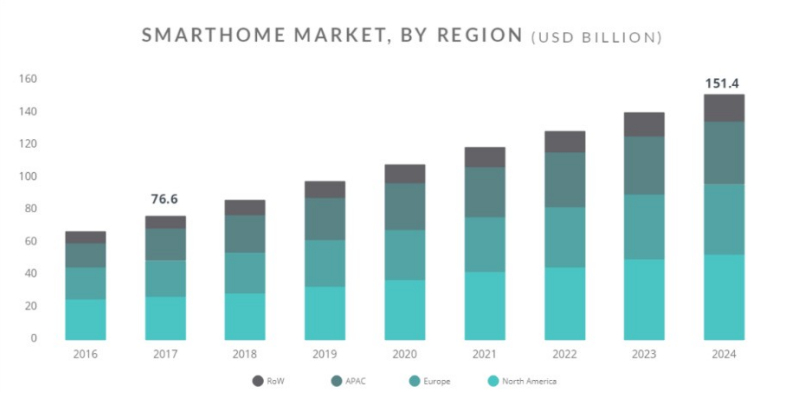

Рисунок 1.1 – Динаміка росту ринка смарт пристроїв згідно аналітичним даним GreyB з 2016 року по 2024 рік

Веб додаток надає можливість керувати смарт пристроєм з будь якого пристроя, який працює на будь якій операційній системі. Так, наприклад, керувати пристроєм можна буде як з телефону чи планшету так і з комп’ютера .

1.2 Веб застосунок

1.2.1 Визначення

Веб застосунок – застосунок, в якому сервером являється веб сервер, а клієнтом виступає браузер. Логіка самого застосунку зосереджується на сервері, а браузер лише завантажує інформацію, відображає її та відсилає наданні користувачем дані [2].

1.2.2 Технічні особливості

Основною перевагою веб застосунку є те, що його функції працюють незалежно від операційної системи пристрою користувача. Таким чином відпадає необхідність писати різні версії під різні операційні системи. Проте різна реалізація певних специфікацій у браузерах та можливість користувача змінювати параметри браузера (змінити шрифт, колір і т.п.) може перешкоджати коректному відображенню інформації [2].

1.2.3 Архітектура веб застосунків

Клієнт посилає запит веб застосунку, який свою чергу виконує певні обчислення, формує веб сторінку й відсилає її клієнту за допомогою HTTP протоколу. Сам веб застосунок може виступати клієнтом інших служб, таких як база даних чи навіть інший веб застосунок.

Для оптимізації і інтерактивності, було створено AJAX підхід розробки застосунків. Він дозволяє відправляти сторінкам веб застосунку запити до веб сервера у фоновому режимі. Завдяки цьому сторінки не перезавантажуються цілком, а лише довантажують дані з сервера, що значно пришвидшує роботу [2].

1.3 Розгляд аналогів

У цьому підрозділі розглядаються аналоги до проекту, що розробляється.

1.3.1 Інтерфейс налаштувань маршрутизатора Tp-link

Веб застосунок який відкривається при запиті на адрес маршрутизатора. Має велику кількість для налаштувань режиму роботу. Приклад інтерфейсу налаштувань маршрутизатора Tp-link наведено на рисунку 1.2.

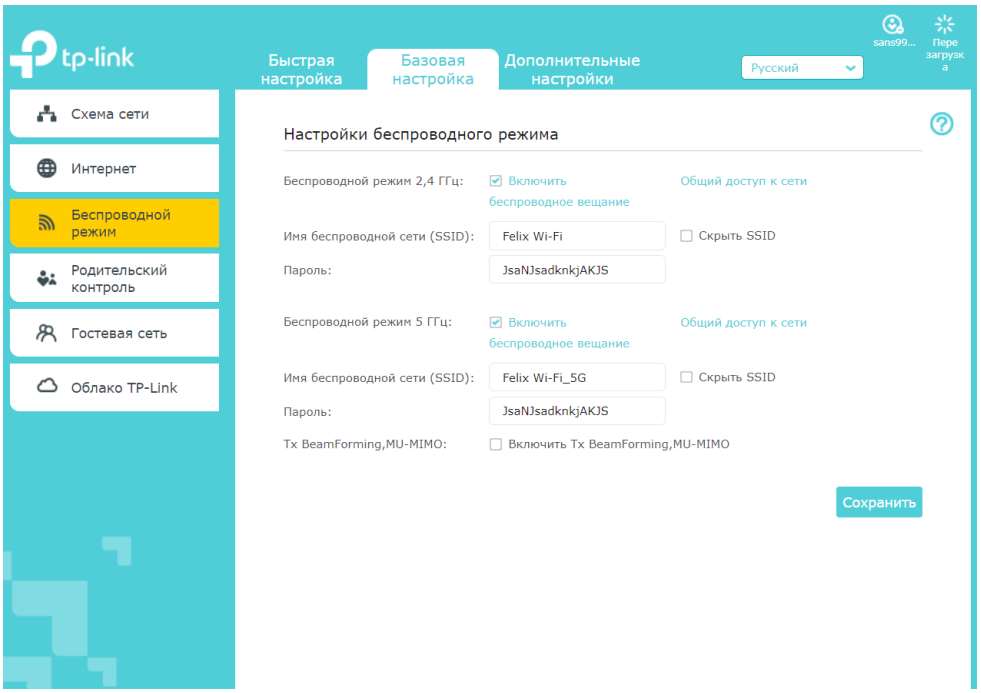

Рисунок 1.2 – Приклад інтерфейсу налаштувань маршрутизатора Tp-link

Переваги:

- велика кількість функцій;
- графічне відображення схеми мережі;
- система інтервального повторення;
- безкоштовність;
- наявність браузерної веб-версії;
- мінімалістичний дизайн.

Недоліки:

- відсутність можливості віддаленого керування пристроєм;
- наявність онлайн ідентифікації користувача, що ускладнює роботу за несправності маршрутизатора.

1.3.2 AMCREST

Застосунок для перегляду відеозображень з пристоїв відеоспостереження . Приклад інтерфейсу AMCREST наведено на рисунку 1.3.

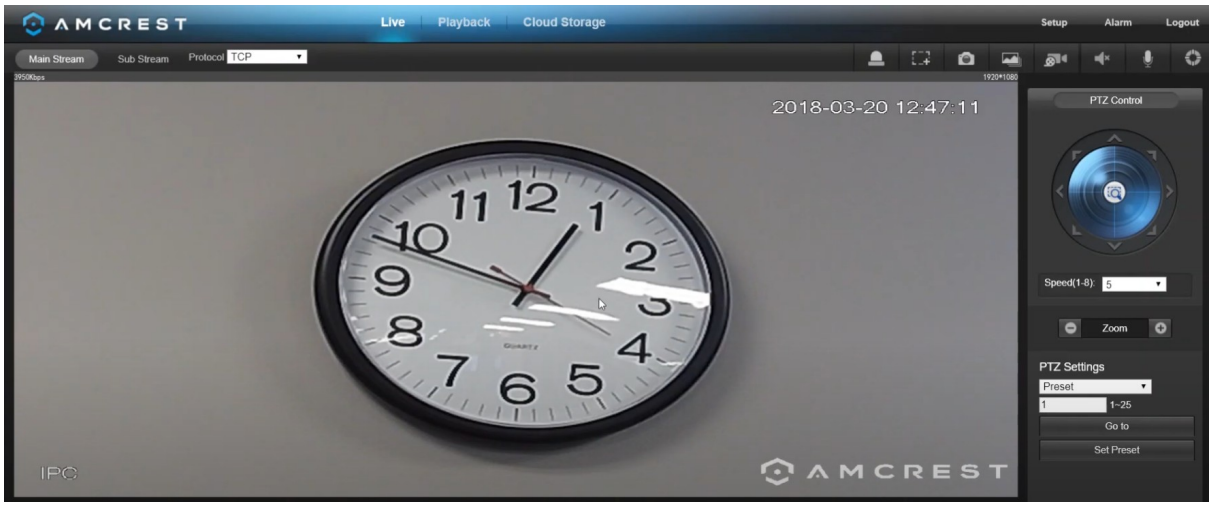

Рисунок 1.3 – Приклад інтерфейсу AMCREST

Переваги:

- перегляд декількох камер;
- можливість збереження відео;
- інструменти керування камерою;
- можливість налаштування камери.

Недоліки:

- відсутність мобільного застосунку.
- відсутність можливості переглядати віддалені пристрої, можна лише в локальній мережі.

1.3.3 SKADA-система

Великий набір програмного забезпечення, спрямованого на створення власного інтерактивного дизайну. Здебільшого

використовується для створення інтерфейсу цілої системи пристроїв. Приклад інтерфейсу створеного у Master SKADA наведено на рисунку 1.4 [3].

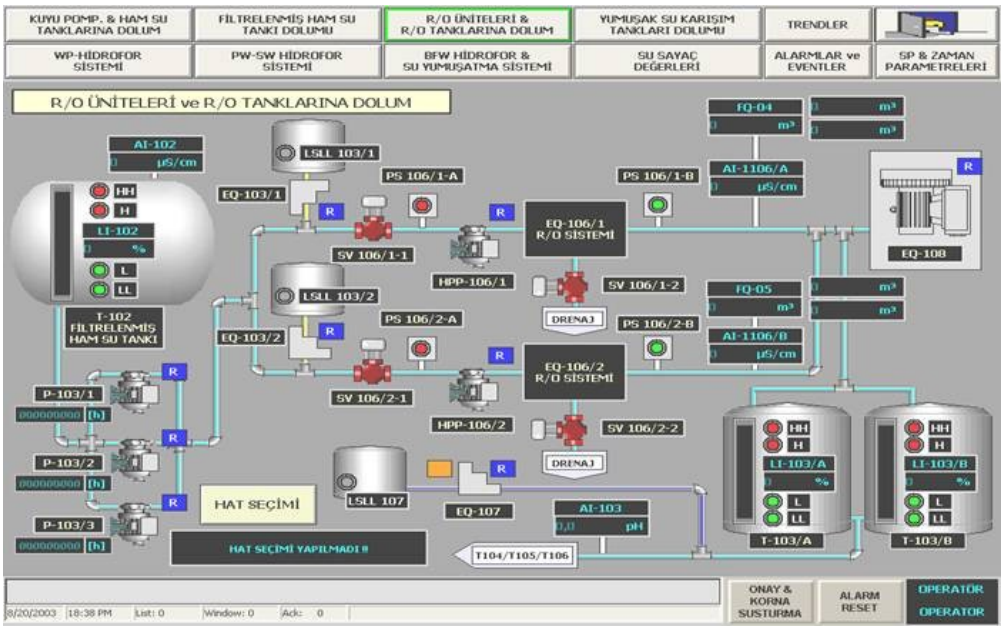

Рисунок 1.4 – Приклад інтерфейсу створеного у Master SKADA

Переваги:

- можливість створення власного інтерфейсу;
- можливість підключення будь якого пристрою;
- можливість створювати мережі пристроїв.

Недоліки:

- програмне забезпечення є платним;
- необхідність навичок під’єднання пристроїв у користувача;
- необхідність навичок програмування у користувача.

1.3.4 Homey

Застосунок що дозволяє групувати пристрої й дистанційно керувати ними. Приклад інтерфейсу Homey наведено на рисунку 1.5 [11].

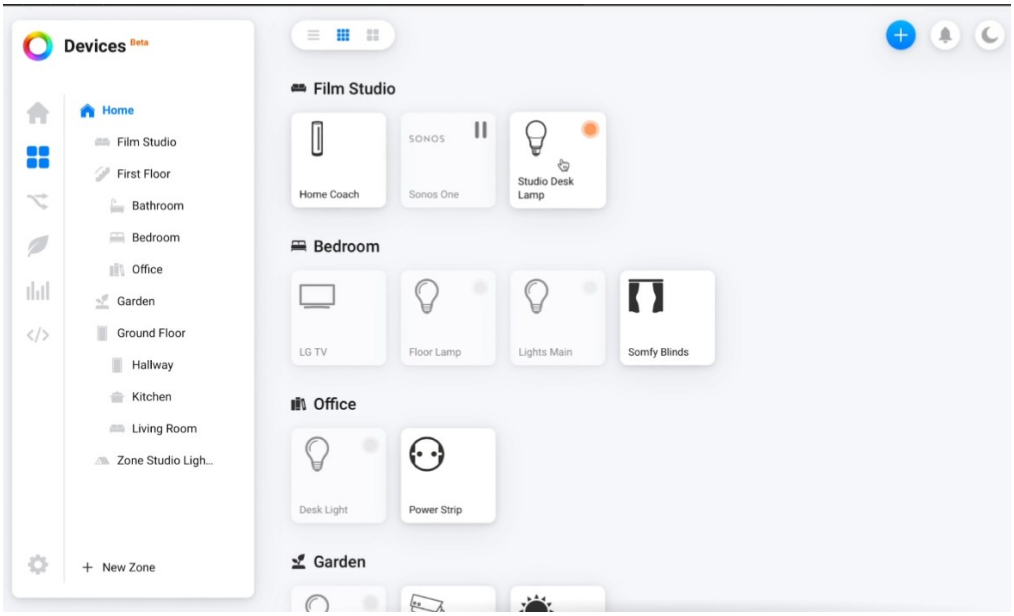

Рисунок 1.5 – Приклад інтерфейсу Homey

Переваги:

- привабливий дизайн;
- розподілення пристроїв за групами;
- можливість керування пристроями віддалено;
- можливість створення власної системи повідомлень;
- наявність версій для Android та IOS;
- наявність веб застосунку;
- синхронізація між пристроями.

Недоліки:

- робота потребує преміум-підписки;
- робота потребує власного пристрою Homey.

1.4 Постановка задачі

Проаналізувавши переваги та недоліки аналогів, був сформований список задач проекту:

- розробити систему користувачів, їх реєстрації та аутентифікації;
- розробити систему пристроїв, їх створення, редагування та підключення;
- розробити систему груп пристроїв;
- розробити систему режиму роботи пристроїв;
- розробити сторінку з візуалізацією даних з пристрою;
- розробити інтуїтивний та привабливий дизайн інтерфейсу;
- розробити програму у вигляді веб застосунку, для охоплення найбільшої кількості пристроїв.

1.5 Висновки за розділом

У даному розділі було проаналізовано актуальність теми: визначено, що тема є актуальною, а розробка програми за цією темою є доцільною.

Було оглянуто предметну область: основні визначення, технічні особливості та архітектура веб застосунку.

Були розглянуті існуючі аналоги, виділено їхні переваги та недоліки.

На основі аналізу аналогів сформовано список задач проекту.

РОЗДІЛ 2
ВИБІР ТА ОБГРУНТУВАННЯ ПРОГРАМНО-АПАРАТНИХ РІШЕНЬ

2.1 Серверна частина

2.1.1 Мова програмування PHP

PHP – високорівнева, інтерпритуєма мова програмування, орієнтована на веб-розробку.

За даними ресурсу W3Techs (рис.2.1) за березень 2022 найпопулярнішою мовою програмування для серверної частини застосунків є PHP [4].

Most popular server-side programming languages

© W3Techs.com	usage	change since 1 March 2022
1. PHP	77.5%	-0.4%
2. ASP.NET	7.7%	-0.2%
3. Ruby	5.9%	-0.1%
4. Java	4.0%	+0.2%
5. Scala	2.7%	+0.3%

percentages of sites

Рисунок 2.1 – статистика популярності мов програмування для серверної частини застосунку

Особливості PHP [5]:

- підтримка ООП;
- автоматичний збірщик сміття;

- підтримка різноманітних СУБД;
- захищеність скриптів;
- можливість конфігурації інтерпритатора;
- підтримка багатьох платформ;
- безліч сторонніх розширень.

Приклад синтаксису найпростішої програми на PHP наведений у лістингу 2.1.

Лістинг 2.1 – найпростіша програма на PHP

```
<?php
        Echo ‘Mihalskiy Bohdan!’
?>
```

Отже, для розробки серверної частини застосунку доцільно використовувати мову програмування PHP.

2.1.2 СУБД MySQL

MySQL – система керування реляційними базами даних з відкритим вихідним кодом [6].

За даними ресурсу Openlogic на момент березня 2022 року (рис. 2.2) двома найпопулярнішими СУБД є PostgreSQL та MySQL [7].

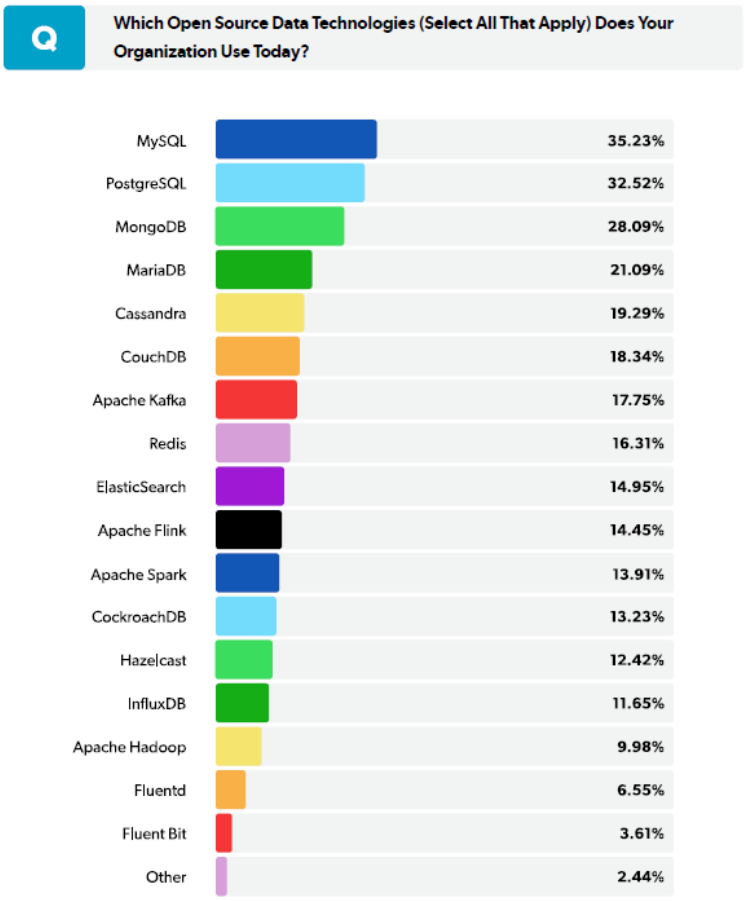

Рисунок 2.2 – статистика популярності СУБД

Особливості MySQL [6]:

- оптимізованість;
- надійність і безпечність роботи БД;
- підтримка великою кількістю засобів розробки ПЗ;
- реляційність БД;
- відкритість вихідного коду;
- безкоштовність використання.

Робота з MySQL підтримується у PHP завдяки розширенню PDO-mysql [8].

Робота з MySQL відбувається через запити стандарту мови SQL. Приклад роботи PHP з MySQL із розширенням PDO-mysql наведено у лістингу 2.2.

Лістинг 2.2 – Приклад роботи PHP з MySQL із розширенням PDO-mysql

```
<?php
    $host = ’mysql:host=localhost;dbname=test_db’;
    $db = new PDO($host,’root’,’12345’);
    $sql = ‘SELECT name, zip from people’;
    foreach($db->query($sql) as $row){
    echo $row[‘username’].” - ”.$row[‘country’].”<br>”;
    }
    $db = null;
?>
```

Отже, у якості СУБД та бази даних даного проекту доцільно вибрати MySQL, а для сумісності роботи з PHP – розширення PDO-mysql [8].

2.1.3 Операційна система Ubuntu

Ubuntu – дистрибутив Linux, заснований на Debian і складається з програмного забезпечення з відкритим вихідним кодом [9].

За даними ресурсу IDC на момент 2017 року (рис. 2.3) найпопулярнішою ОС для серверних рішень була Linux [10].

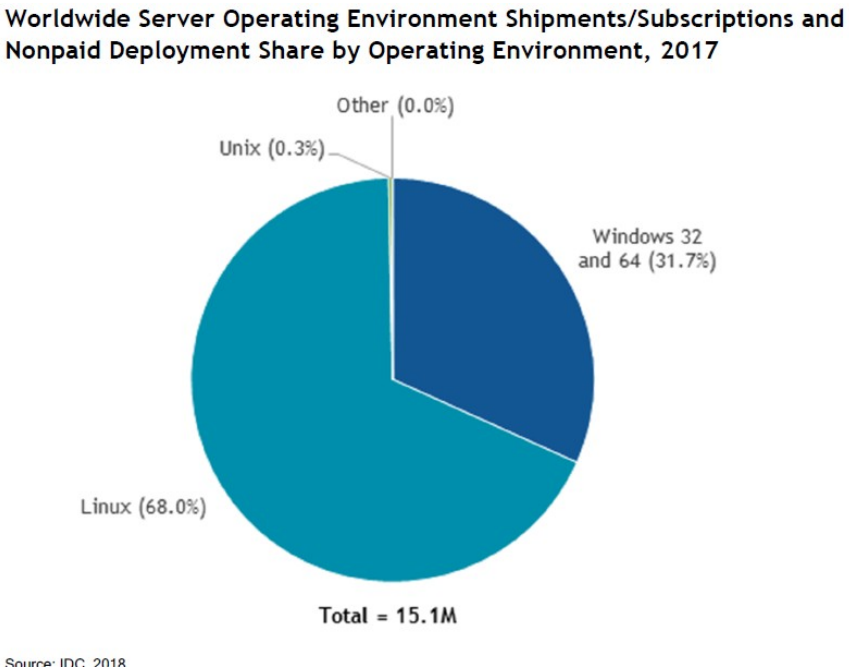

Рисунок 2.3 – статистика популярності серверних ОС

За даними ресурсу Open Stack (рис.2.4) ОС Ubuntu Server має найбільший відсоток ринку серед ОС на базі ядра Linux [11], тому її доцільно вибрати для даного проекту.

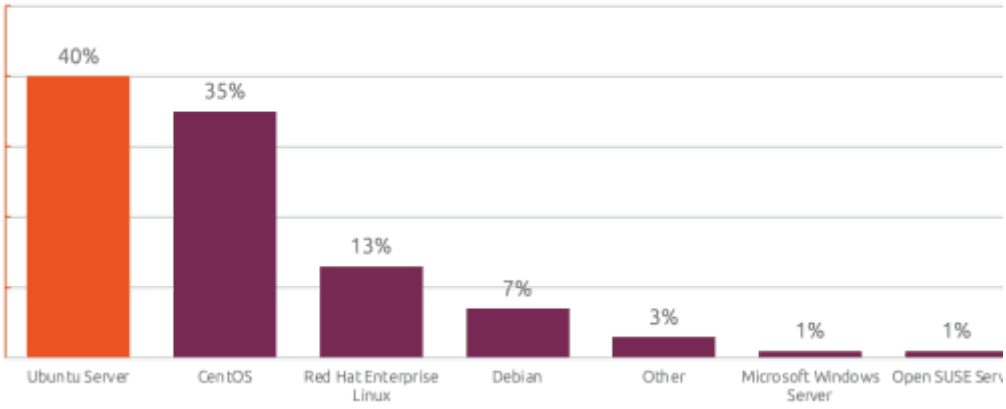

Рисунок 2.4 – статистика популярності ОС на базі ядра Linux

Проте Ubuntu Server має лише командний інтерфейс, на відміну від редакції Ubuntu Desktop[12]. Саме в Ubuntu Desktop присутній сучасний графічний інтерфейс ОС (рис.2.5), що робить розробку проекта

комфортнішою та ефективнішою, завдяки графічній візуалізації та інструментам для тестування.

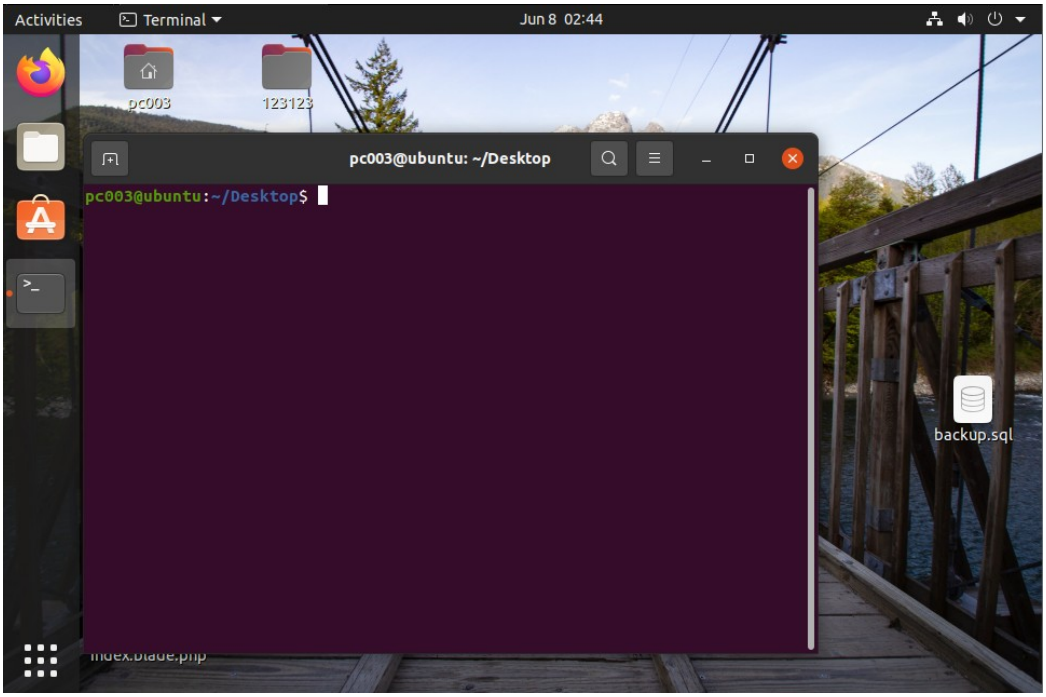

Рисунок 2.5 – Інтерфейс Ubuntu Desktop

Основні особливості Ubuntu [13]:

- захищеність ОС;
- гнучкість налаштування роботи системи, прав користувачів;
- безкоштовність використання системи;
- має велику кількість стандартного ПЗ для моніторингу ресурсів;
- низькі вимоги до апаратного забезпечення.

Отже, для розробки та у якості серверної ОС для проекту доцільно використовувати Ubuntu Desktop.

2.2 Клієнтська частина

2.2.1 Мова розмітки гіпертексту HTML та каскадна таблиця стилів CSS

HTML (HyperText Markup Language) – стандартна мова розмітки для документів, створених для відображення у веб браузері [14].

Елементи HTML є будівельними блоками сторінок HTML. Елементи HTML розмежовуються тегами, записаними за допомогою кутових дужок. Зазвичай теги позначають структурну семантику для тексту, наприклад заголовки, абзаци, посилання, тощо. Інші ж теги (наприклад <img />) виводять зображення чи інтерактивні елементи сторінки, такі як кнопки, поля для тексту. Приклад коду HTML-документу та його відображення у веб браузері показано на рисунку 2.6.

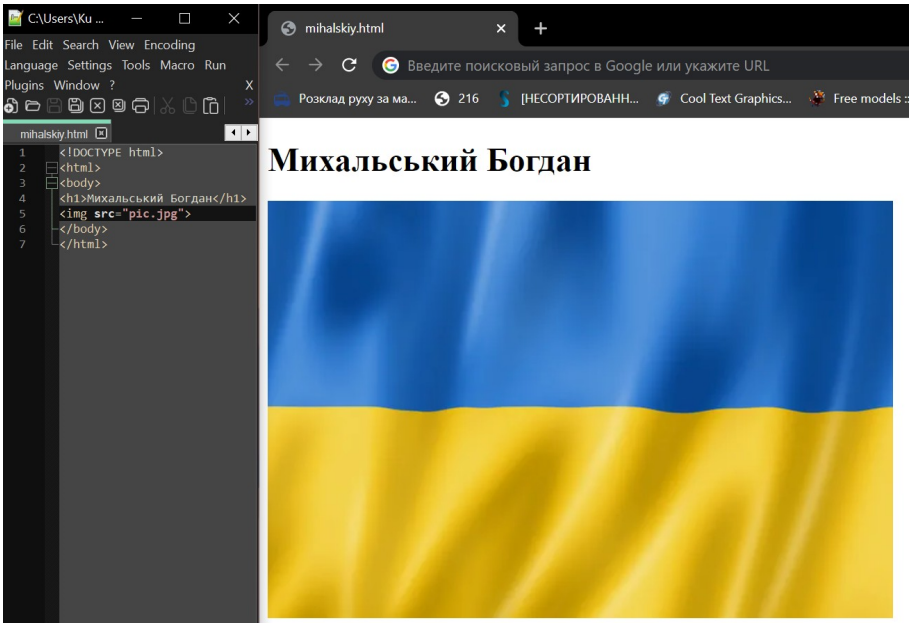

Рисунок 2.6 – приклад відображення HTML-документу

Для опису подання відображення документа, написаного мовою розмітки, використовують каскадну таблицу стилів на мові CSS. Її

використовують для налаштування таких параметрів як: колір, розмір тексту, його шрифт, надання фільтру зображенню, зміна положення певних елементів, створення анімацій, реакцій на натискання на елементи, тощо [15].

CSS можна використовувати напряму у HTML-документі, оголошуючи налаштування окремих елементів за допомогою тегу <style>. Проте здебільшого стилі оголошуються в окремому файлі css, назначаючи певні налаштування відображення окремих елементів та груп елементів. Приклад зміни відображення документу з рисунку 2.6 після присвоєння стилів CSS показано на рисунку 2.7.

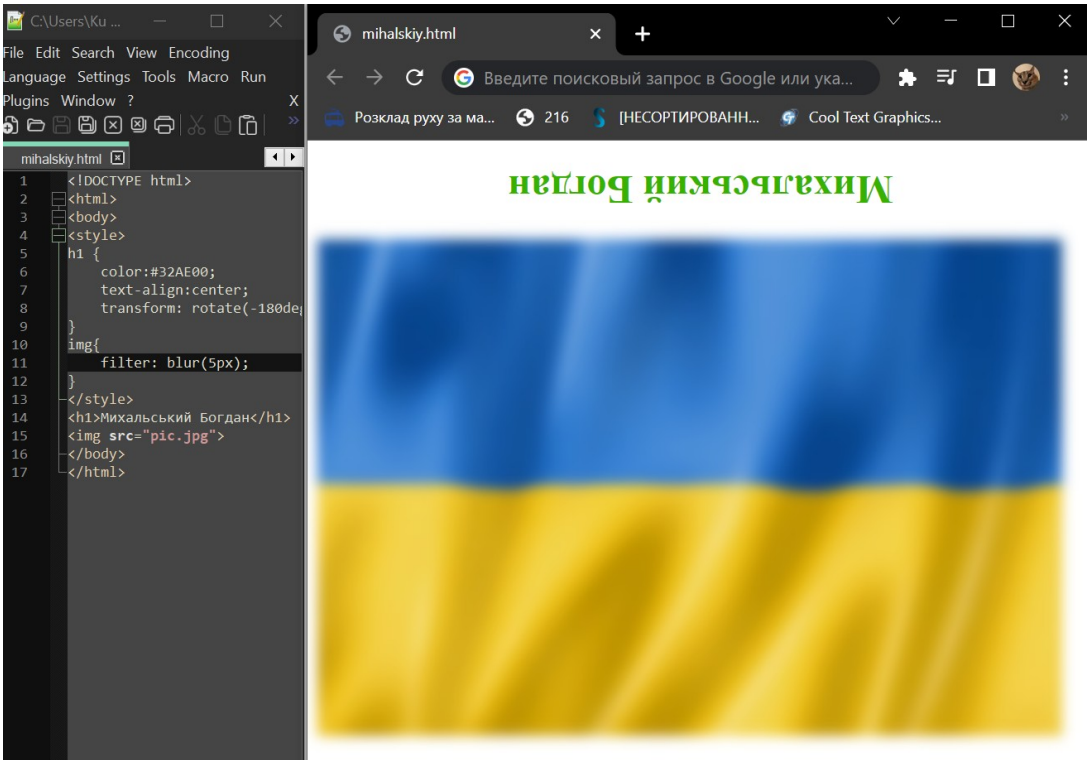

Рисунок 2.7 – приклад присвоєння стилів CSS

Так як клієнтська частина являє собою веб-сайт, для клієнтської частини проекту необхідно використати HTML та CSS.

2.2.2 Мова програмування JavaScript

JavaScript – високорівнева мова програмування. Підтримує об’єктно-орієнтований, функціональний та імперативний стилі. Являється однією із основних технологій створення веб-сторінок, разом з HTML та CSS. Найбільш широке застосування знаходить у браузерах як мова сценаріїв для надання інтерактивності веб-сторінкам. Основні архітектурні риси: динамічна типізація , слабка типізація , автоматичне управління пам'яттю , прототипне програмування , функції як об'єкти першого класу. Також JavaScript підтримує взаємодію із різними HTML компонентами і стандартними структурами даних [16].

Отже виходячи з того, що клієнтська частина веб сервісу являє собою веб-сайт, доцільно використати мову програмування JavaScript для клієнтської частини проекту.

Приклад синтаксису найпростішої програми на JavaScript наведений у лістингу 2.3.

Лістинг 2.3 – Найпростіша програма на JavaScript

```
<script type=”application/javascript”>
    alert(“Mihalskiy Bogdan”)
</script>
```

2.2.3 Мова програмування TypeScript

TypeScript – суворий синтаксичний наднабір JavaScript, який додає до мови необов'язкову статичну типізацію. Він розроблений для розробки великих додатків і трансляції на JavaScript. Оскільки це наднабір JavaScript, існуючі програми JavaScript також є дійсними програмами

TypeScript.. Можна використовувати компілятор TypeScript за замовчуванням або компілятор Babel для перетворення TypeScript у JavaScript [17].

Оскільки TypeScript є синтаксичним наднабіром і робить розробку великих проектів зручніше, цю мову програмування доцільно використати для клієнтської частини проекту.

2.2.4 Бібліотека React

React – безкоштовна бібліотека JavaScript із відкритим вихідним кодом для створення інтерфейсів користувача на основі компонентів інтерфейсу користувача. React можна використовувати як базу для розробки односторінкових , мобільних або серверних додатків із такими фреймворками, як Next.js. Однак React займається лише керуванням станом і відтворенням цього стану в DOM, тому створення додатків React зазвичай вимагає використання додаткових бібліотек для маршрутизації, а також певної функціональності на стороні клієнта [18].

Оскільки клієнтська частина веб сервіса являє собою SPA, а також завдяки тому що React має багатий інструментарій для створення динамічних компонентів веб застосунку, буде доцільно використовувати цю бібліотеку для клієнтської частини проекту.

Приклад синтаксису найпростішого компоненту на React наведений у лістингу 2.4.

Лістинг 2.4 – Найпростіший компонент на React

```
import React from "react";
import Tool from "./Tool";
const Example = () => {
```

```
  return (
    <>
    <div className="app">
    <Tool name="Bogdan" />
  </div>
  </>
  );
};
export default Example;
```

2.3 Середа розробки

2.3.1 Інтегроване середовище розробки WebStorm

WebStorm – сучасне інтегроване середовище розробки. WebStorm забезпечує автодоповнення , аналіз коду, навігацію за кодом , рефакторинг , налагодження , та інтеграцію з системами управління версіями . Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами [19].

Основні функції:

- модифікація файлів .css, .html, .js з одночасним переглядом результатів;
- інструменти розгортання;
- віддалене розгортання за протоколами FTP , SFTP;
- налагодження коду на JavaScript;
- інтеграція з системами керування версіями: Subversion , Git , GitHub, Perforce , Mercurial , CVS;
- кастомізація інтерфейсу;

- система контролю версій;
- підтримка JSDoc, Node.js, React, JSX;
- інтеграція із системами відстеження помилок;
- підтримка плагінів.

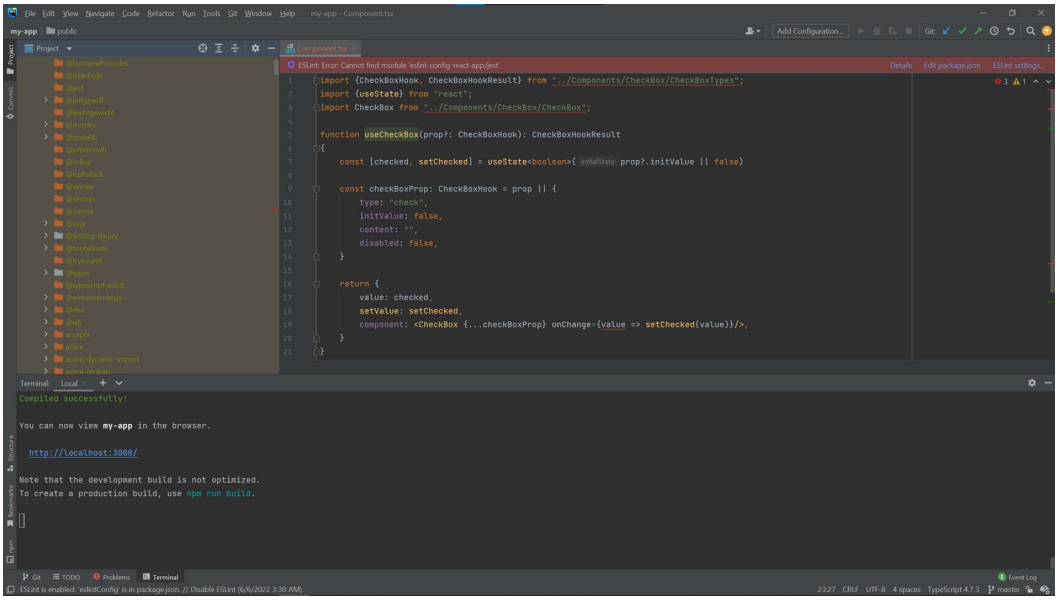

Рисунок 2.8 – інтерфейс інтегрованого середовища розробки WebStorm

Отже, інтегровану середу розробки WebStorm доцільно використати для розробки як клієнтської, так і серверної частини проекту.

2.3.2 Середовище розробки API Postman

Postman – платформа API , на якій розробники можуть проектувати, створювати , тестувати та повторювати свої API [20].

Postman має наступний функціонал:

- зберігання, створення та угрупування API запитів та їх відповідей для тестування;
- візуалізація наявних проблем (проблеми безпеки, помилки запитів);

- створення робочих просторів, з власними змінними й структурою запитів, що дозволяє ефективніше працювати в команді.

Робота із запитами REST API у Postman показана на рисунку 2.14.

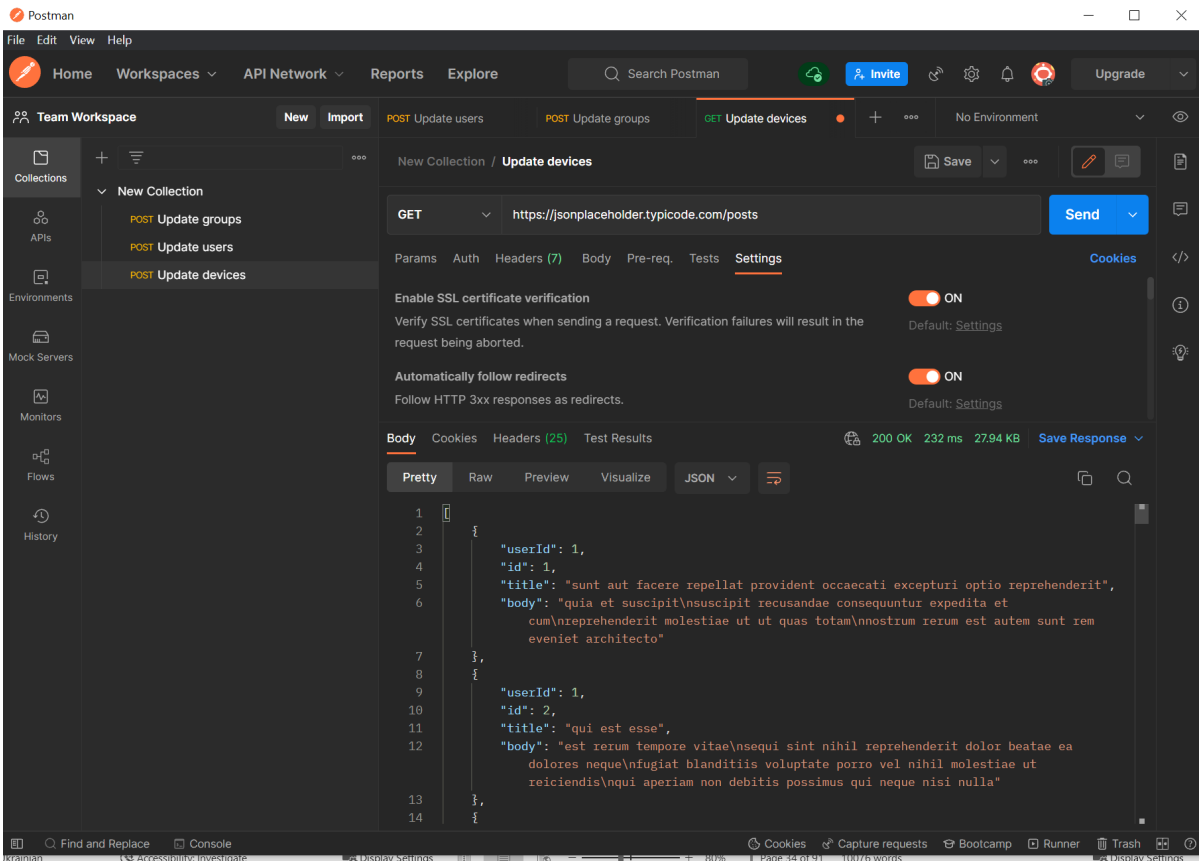

Рисунок 2.14 – робота із запитом REST API у Postman

Із вищесказаного, випливає, що Postman доцільно використати для розробки, тестування та документації REST API проекту, що розробляється.

2.5 Висновки за розділом

У даному розділі було визначено та оглянуто стек технологій для проекту, що розробляється.

Визначено, що:

- у якості серверної ОС буде використана Ubuntu Desktop;
- для розробки серверної частини сервісу буде використана мова програмування PHP;
- у якості СУБД буде використана MySQL;
- для розробки клієнтської частини будуть використані HTML, CSS та мови програмування JavaScript, TypeScript із використанням бібліотеки React;
- інтегровану середу розробки WebStorm буде використано для розробки як серверної, так і клієнтської частини проекту;
- середовище розробки API Postman буде використано для розробки, тестування та використання REST API проекту.

РОЗДІЛ 3
ПРОГРАМНИЙ МОДУЛЬ

3.1 Проектно-архітектурні застосунки UML

UML діаграма прецедентів показана на рисунку 3.1.

Рисунок 3.1 – UML діаграма прецедентів

UML діаграма розгортки веб сервісу показана на рисунку 3.2.

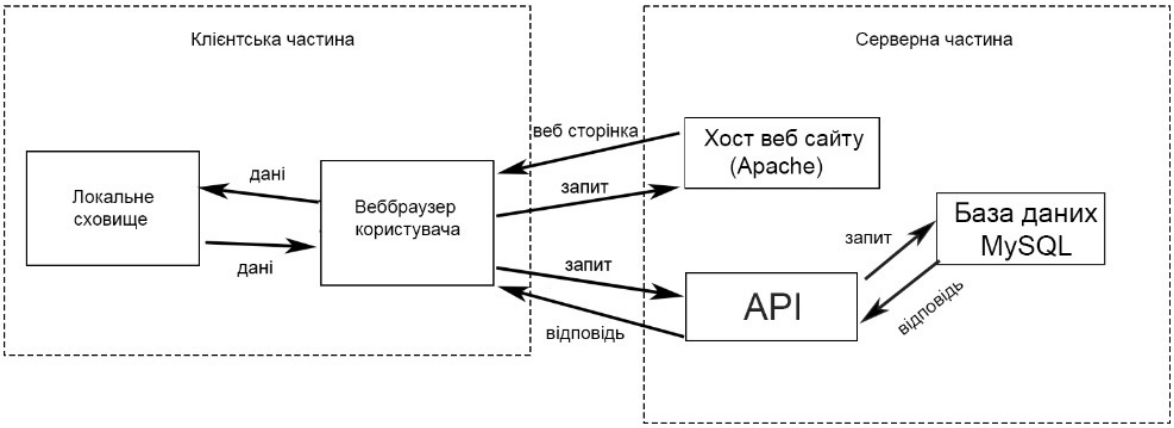

Рисунок 3.2 – UML діаграма розгортки вебсервісу

3.2 Структура проекту

3.2.1 Структура клієнтської частини

Файли клієнтської частини знаходяться у межах 8 основних директоріях. Схема файлової структури показана на рисунку 3.3.

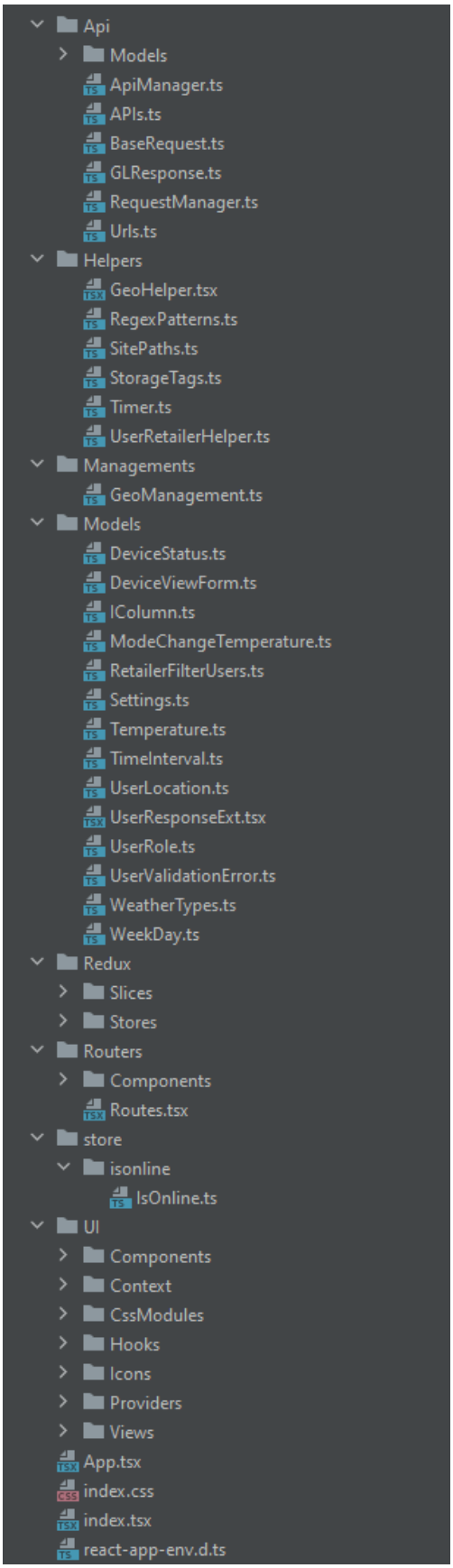

Рисунок 3.5 – Схема файлової структури клієнтської частини

Веб застосунок являє собою SPA, тому App.tsx – основний документ, а навігація між сторінками реалізована завдяки компонентам, які динамічно міняються.

Вхідний документ – «App.tsx», містить у особі основні налаштування маршрутизації веб застосунку. Всі інші компоненти застосунку знаходяться у папці «UI».

У папці «store» знаходяться скрипти для збереження налаштувань користувача.

У папці «Routes» визначено основні розділи веб застосунку:

- «sign-in» (сторінка авторизації);
- «management-users» (сторінка керування користувачами);
- «management-devices» (сторінка керування пристроями);
- «groups» (сторінка керування групами).
- «analytics» (сторінка з статистикою стану пристрою);
- «sign-up» (сторінка для створення нового аккаунту);
- «location» (мапа з пристроями, доступними для перегляду в залежності від ролі);
- «rules» (сторінка для створення правил роботи пристрою);
- «personal-data» (сторінка з інформацією про авторизованого користувача);
- «my-devices» (сторінка для перегляду власних пристроїв користувача);
- «subscriptions» (сторінка для перегляду тарифу користувача);

У папках «Models» і «Redux» визначено допоміжні графічні компоненти, серед яких:

- елемент погоди;
- зображення пристрою;
- бокове меню з налаштуваннями користувача;
- компонент таблиці.

У папках «API», «Helpers» і «Managments» містить компоненти для роботи з API, серед функціоналу яких є:

- отримання зображення флагу користувача;
- взаємодія з основним API, для створення, редагування, відображення і видалення різноманітних елементів проекту;
- створення таймерів для динамічного оновлення інформації;
- взаємодія з додатковими серверами, для отримання інформації з пристроїв.

У файлі «index.css» знаходяться основні стилі веб застосунку.

3.2.2 Структура серверної частини

Файли серверної частини знаходяться в межах однієї директорії. Схема файлової структури показана на рисунку 3.6.

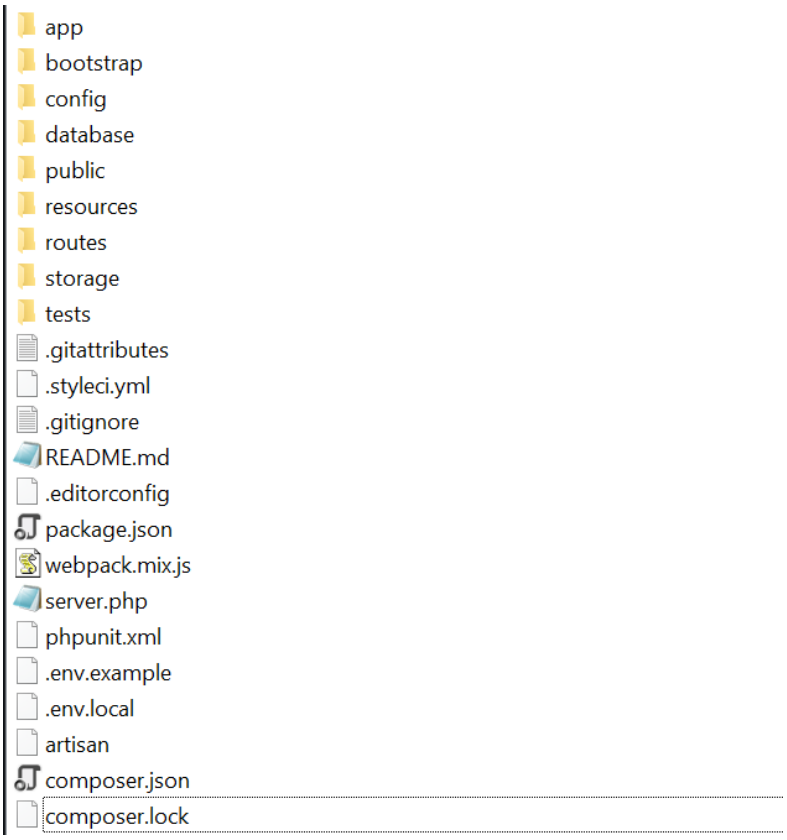

Рисунок 3.6 – Схема файлової структури серверної частини

Файл запуску серверу API – «server.php», у ньому скрипт визначає доступні енд-поінти API. Директорія app містить файли, які описують методи API, серед яких:

- «/login» (метод аутентифікції);
- «/refresh» (оновлює токен користувача);
- «/signup» (метод реєстрації користувача);
- «/avatars» (отримання зображення користувача);
- «/devices» (методи для роботи з пристроями);
- «/groups» (методи для роботи з групами);
- «/geo» (методи для роботи з місцезнаходженням користувача);
- «/messages» (методи для роботи із повідомленнями);
- «/rules» (методи для роботи з правилами);
- «/users» (методи для роботи з користувачами);
- «/plans» (методи для роботи з тарифними планами).

У файлі «.env» визначаються основні налаштування для роботи API, серед яких:

- «APP_KEY» (ключ для генерації токенів авторизації);
- «DB_HOST» (адреса сервера бази даних);
- «DB_USERNAME» (ім’я користувача бази даних);
- «DB_PASSWORD» (пароль користувача бази даних);
- «DB_PORT» (порт сервера бази даних);
- «DB_DATABASE» (назва бази даних).

У діректорії «public» містяться файли клієнтської частини застосунку.

3.2.3 Структура бази даних

База даних складається з 13 таблиць, серед яких:

- «users», що містить інформацію про всіх користувачів;
- «devices», що містить інформацію про всі пристрої;
- «plans», що містить інформацію про всі тарифні плани;
- «groups», що містить інформацію про групи пристроїв;
- «rules», що містить інформацію про всі правила роботи пристроїв.

Поля «ID» у всіх таблицях «users» – первинні ключі, вони зберігають унікальний ідентифікатор користувача у форматі UUID 4.

Поля «id» у таблицях «rules», «users» і «groups» відносяться до полей «rule_id», «user_id», «group_id» у таблиці «devices» відповідно як «один до багатьох». Поле «id» у таблиці «rules» відноситься до поля «rule_id» у таблиці «groups» як «один до багатьох». Поле «id» у таблиці «users» відноситься до поля «user_id» у таблиці «rules» як «один до багатьох». Поле «id» у таблиці «plans» відноситься до поля «plan_id» у таблиці «users» як «один до багатьох».

Графічна схема бази даних представлена на рисунку 3.7.

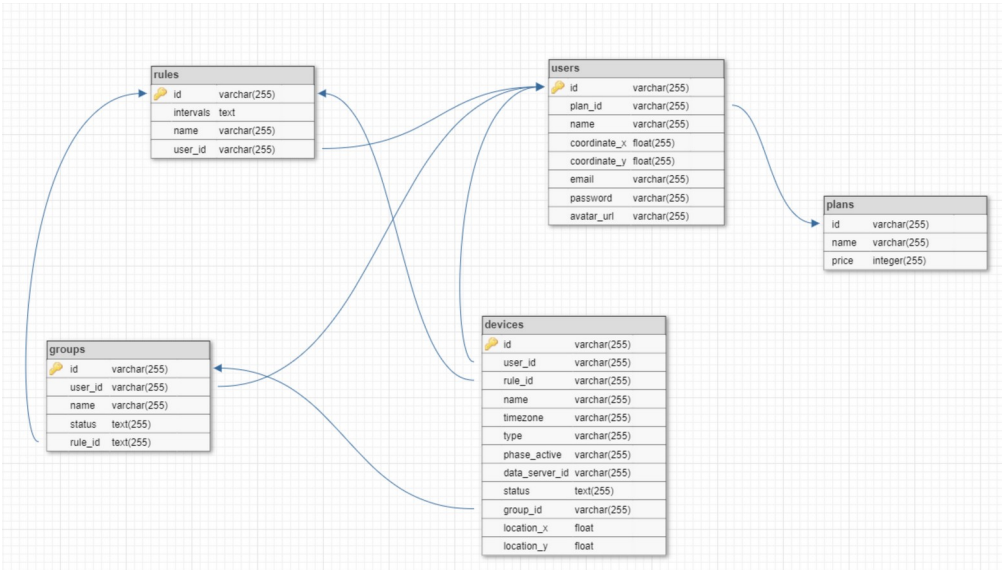

Рисунок 3.7 – Графічна схема бази даних

3.3.1 Розробка клієнтської частини

Клієнтська частина представляє собою SPA веб застосунок. Застосунок динамічно оновлює єдиний html документ, замінюючи різноманітні компоненти. Основний tsx файл, відповідний за зміну компонентів (App.tsx) зображено на рисунку 3.8.

```
<div className={"d-flex flex-row h-100"}>
    {
        auth.isLogin ?
            <div className={"col-auto sidebar p-1"}>
                <Sidebar />
            </div>
        :
            <div className={"col-auto d-none d-lg-block h-100"}>
                <Tagline/>
            </div>
    }
    <div className={"col d-flex flex-column overflow-auto"} >
        {
            auth.isLogin && auth.me ?
                <Navbar className={"position-sticky "}  style={{ overflow:"hidden", borderBottom: "1px solid #DBDBDB"}}>
                    <Navbar.Collapse className="justify-content-between">
                        <img alt={"logo"}
                             className={"ms-2 d-sm-block d-none"}
                             onClick={()=>{}}
                             height={50}
                             width={200}
                             src={LOGO_IMG}/>

                        <img alt={"small logo"}
                             className={"ms-2 d-block d-sm-none"}
                             onClick={()=>{}}
                             height={50}
                             width={25}
                             src={SMALL_LOGO_IMG}/>
                        <SmallProfile/>
                    </Navbar.Collapse>
                </Navbar>
            :
                null
        }
        <Routes/>
    </div>
</div>
```

Рисунок 3.8 – Код файлу App.tsx, що відповідає за зміну компонентів застосунку

На сторінці авторизації застосунок перевіряє наявність/правильність збереженого токену авторизації. Якщо токена немає або закінчився термін придатності – сайт залишається на сторінці авторизації (рис 3.9).

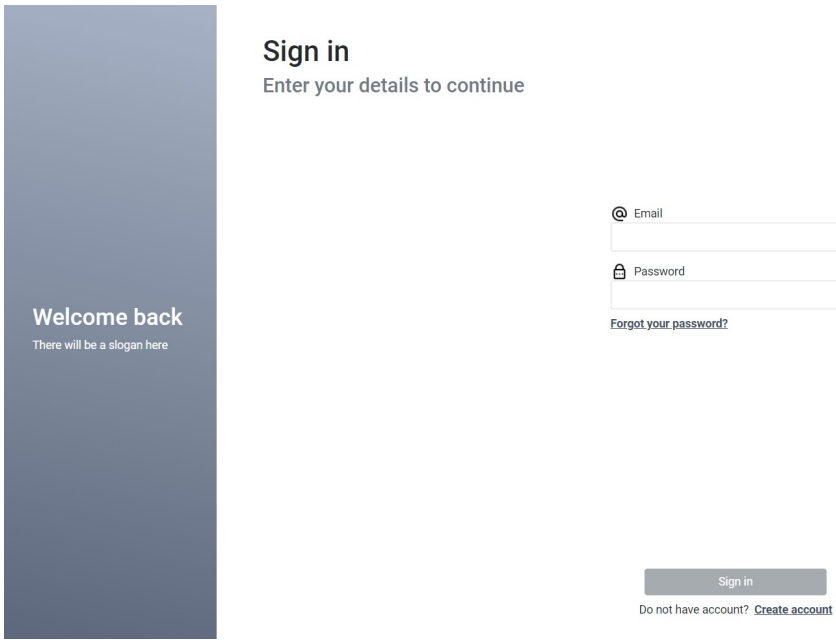

Рисунок 3.9 – Зовнішній вигляд сторінки авторизації

Далі користувач повинен ввести свої дані, або натиснути на кнопку «створити аккаунт». Якщо дані вірні – веб застосунок відкриє розділ з інформацією про користувача, якщо ж дані не правильні – користувача буде оповіщено про помилку (рис. 3.10).

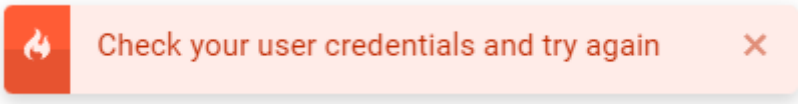

Рисунок 3.10 – Повідомлення про не правильні дані

Якщо ж клієнт натисне на кнопку «створити аккаунт» - відкриється розділ реєстрації нового користувача (рис 3.11).

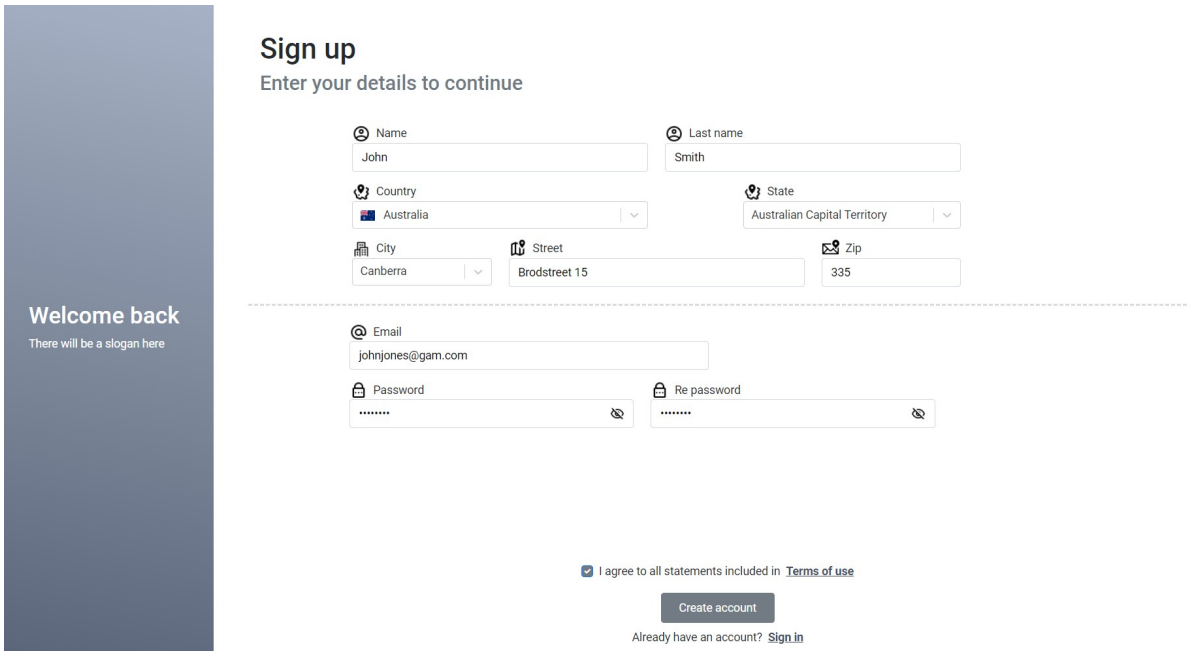

Рисунок 3.11 – Сторінка реєстрації користувача

Після введення даних та натискання кнопки «створити аккаунт», застосунок відправляє запит з даними форми до методу API створеня нового користувача. Відправлені дані показано на рисунку 3.12.

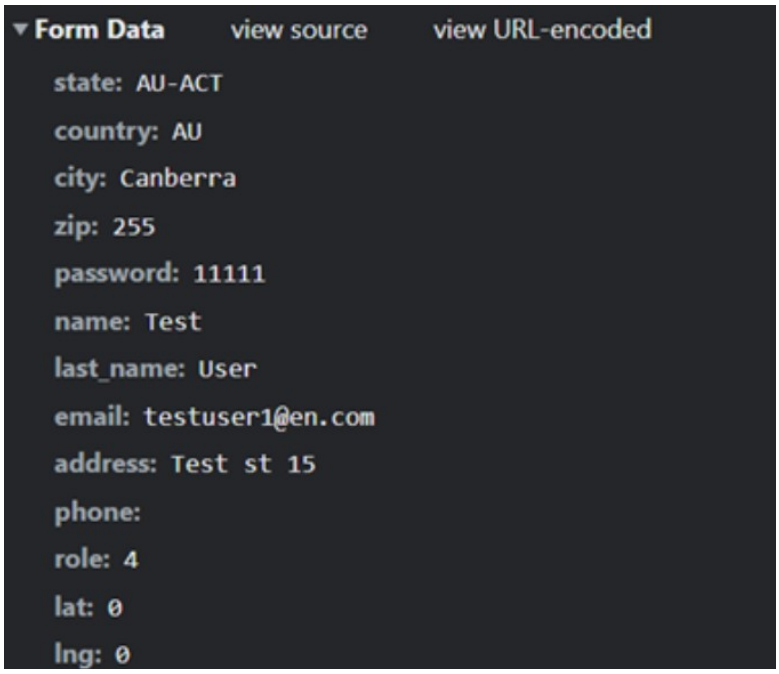

Рисунок 3.12– Дані запиту для створення нового користувача

Після авторизації відкривається сторінка з інформацією про користувача (рис 3.13).

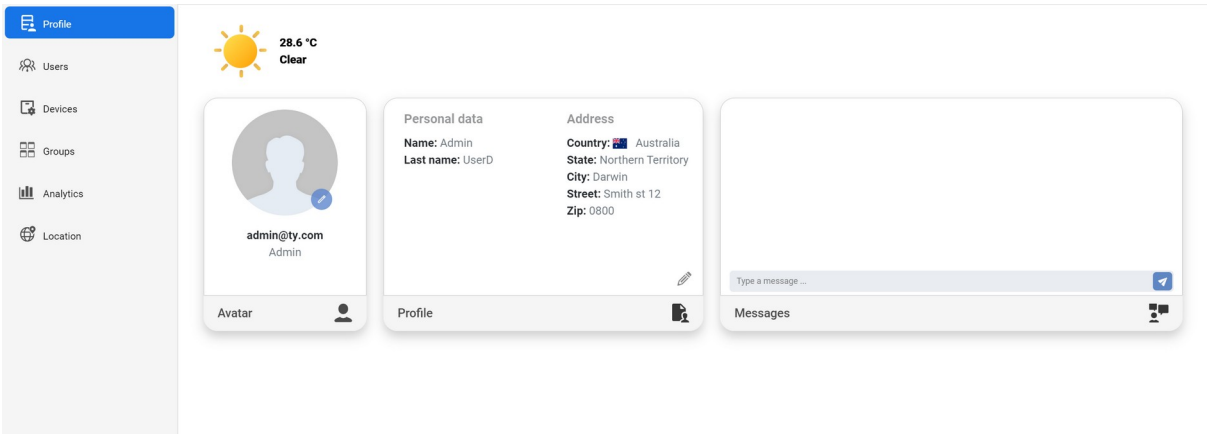

Рисунок 3.13 – Сторінка з інформацією про користувача

При натисканні кнопки з олівцем відкривається меню для вибору зображення користувача, а після вибору зображення – надсилається запит для зміни зображення користувача (рис 3.14).

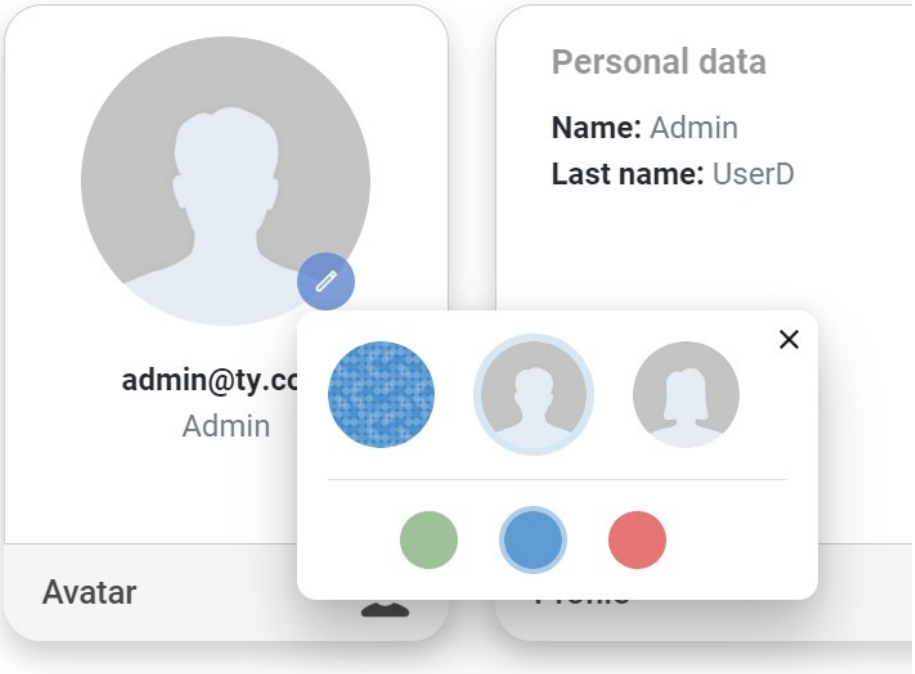

Рисунок 3.14 – Вибір зображення користувача

При натисканні на вкладку «Пристрої» відкривається сторінка з усіма пристроями користувача (рис 3.15), або усіма пристроями в системі (рис 3.16), якщо користувач має адміністраторську роль.

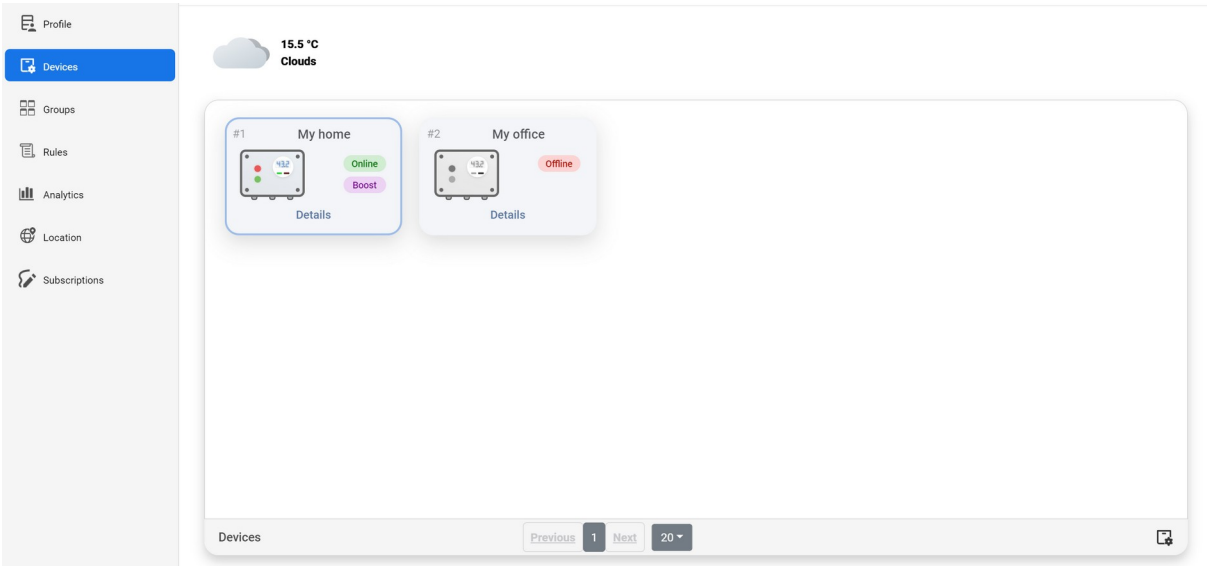

Рисунок 3.15 – Сторінка з пристроями користувача

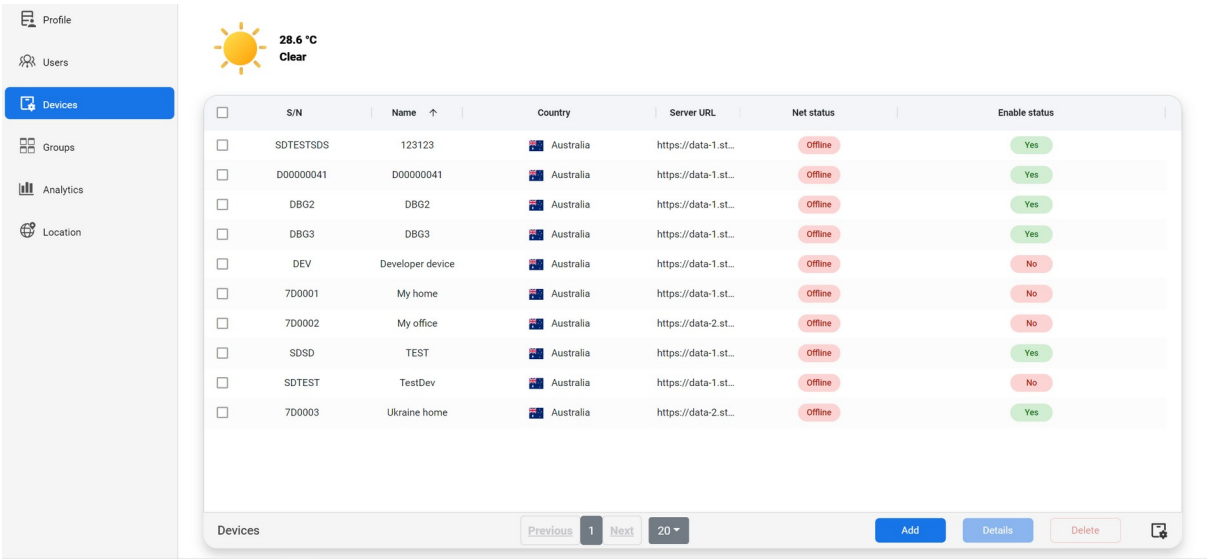

Рисунок 3.16 – Сторінка з усіма пристроями в системі

Якщо роль користувача є адміністраторською, то користувач може додавати (рис 3.17), редагувати та видаляти (рис 3.18) існуючі пристрої в системі. Відповідні запити відсилаються на енд-поінт «/devices».

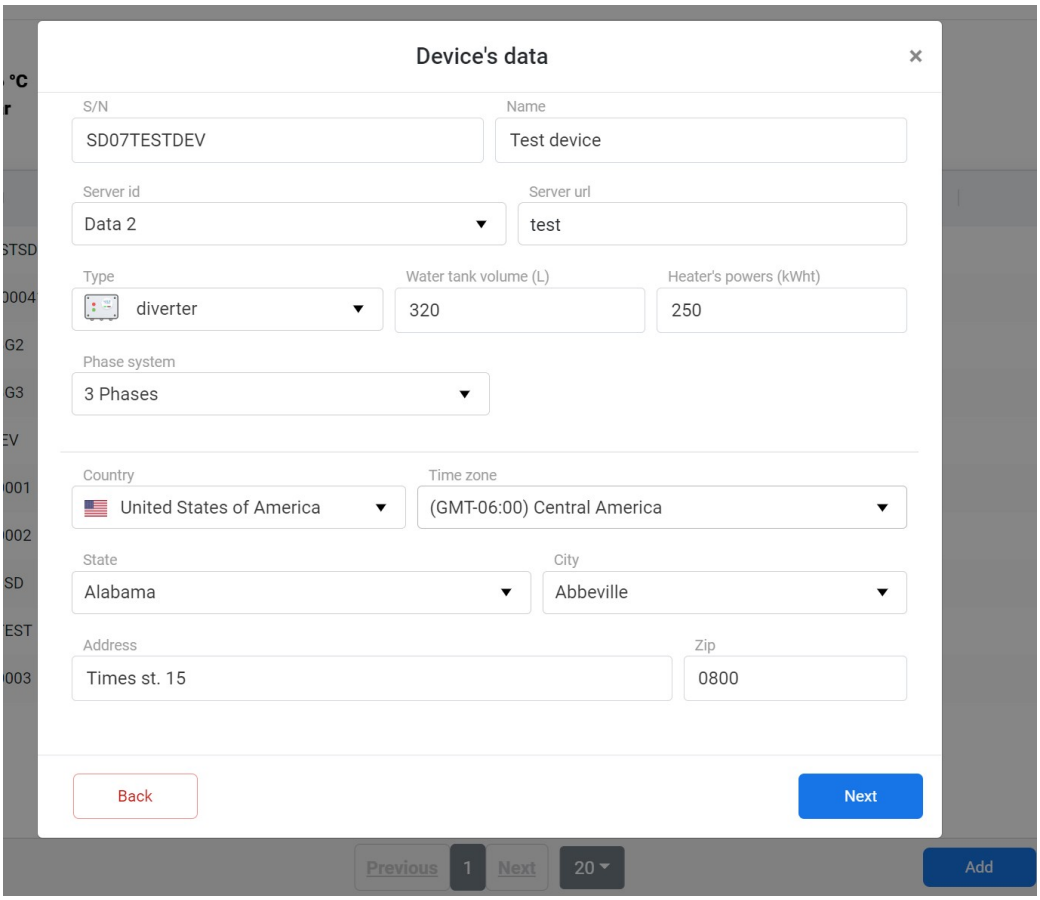

Рисунок 3.17 – Додавання нового пристрою

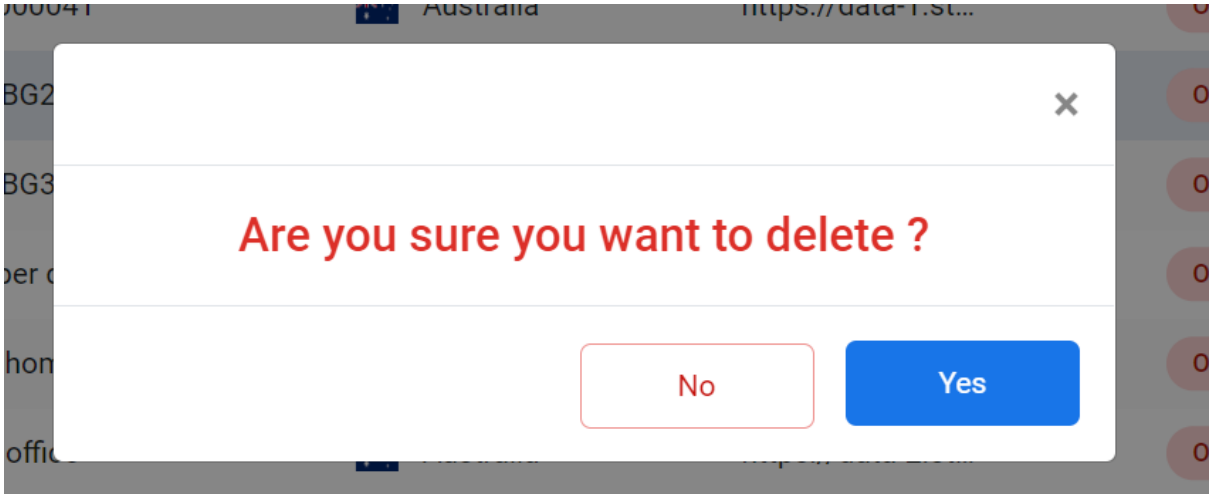

Рисунок 3.18 – Модальне вікно підтвердження видалення пристрою

При натисканні на кнопку «Деталі» відкривається модальне вікно з інформацією про пристрій (рис. 3.19).

При натисканні на перемикач змінюється стан пристрою, в залежності від ролі. Так, наприклад, пристрій може буде вимкнений адміністратором, користувачем чи продавцем. Також тут присутні кнопки для визначення правила роботи пристрою, редагування пристрою, отримання інформації про користувача, продавця, статистику роботи пристрою і місцезнаходження на карті.

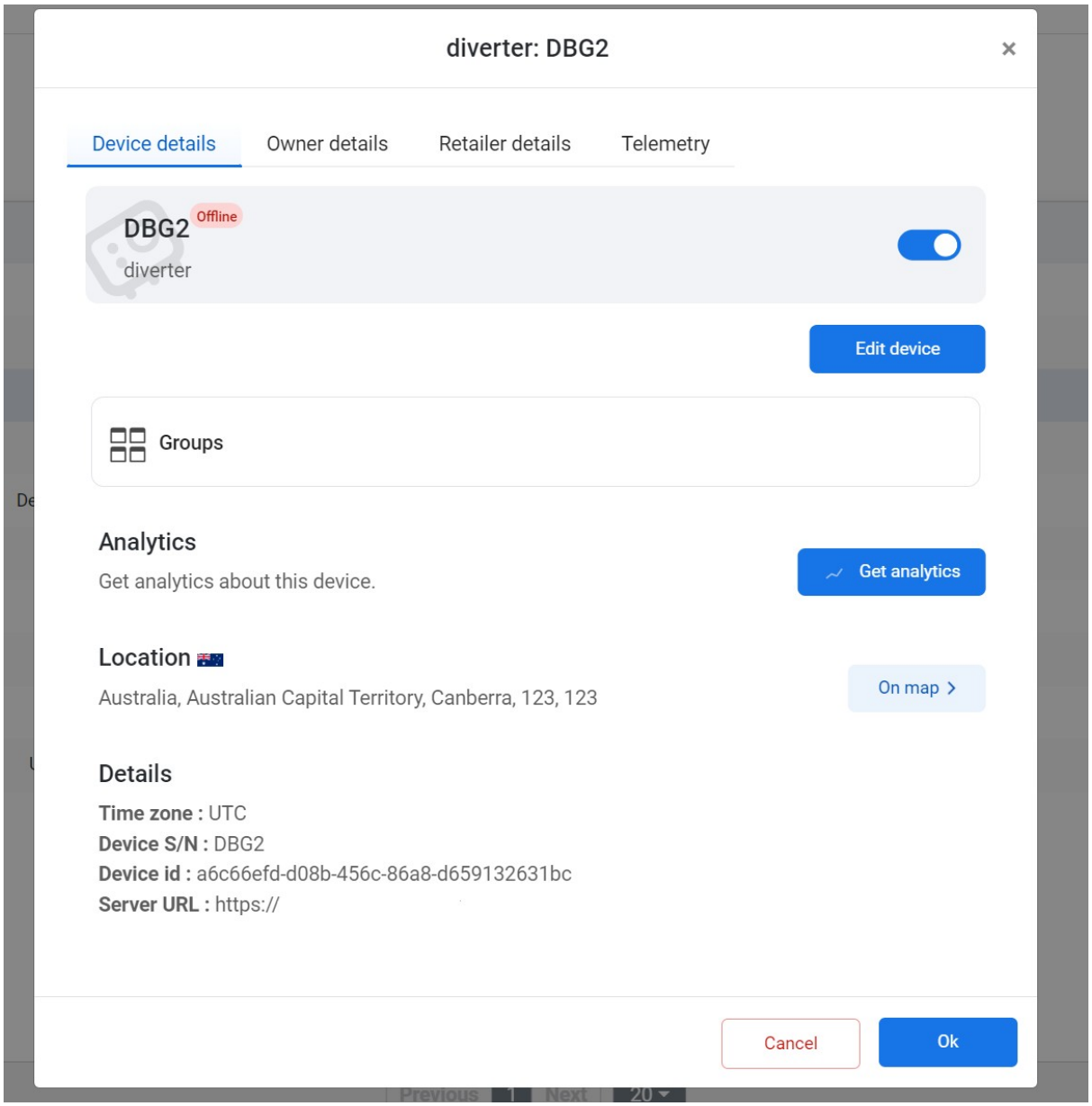

Рисунок 3.19 – Модальне вікно з інформацією про пристрій

Якщо пристрій кимось вимкнено, то у верхньому правому куті з’являється повідомлення з інформацією про роль користувача, який вимкнув пристрій (рис. 3.20).

Рисунок 3.20 – Повідомлення про вимкнення пристрою

При натисканні на розділ «Групи» відкривається сторінка з групами пристроїв користувача (рис. 3.21). При натисканні на бокову кнопку групи відкривається меню для видалення групи.

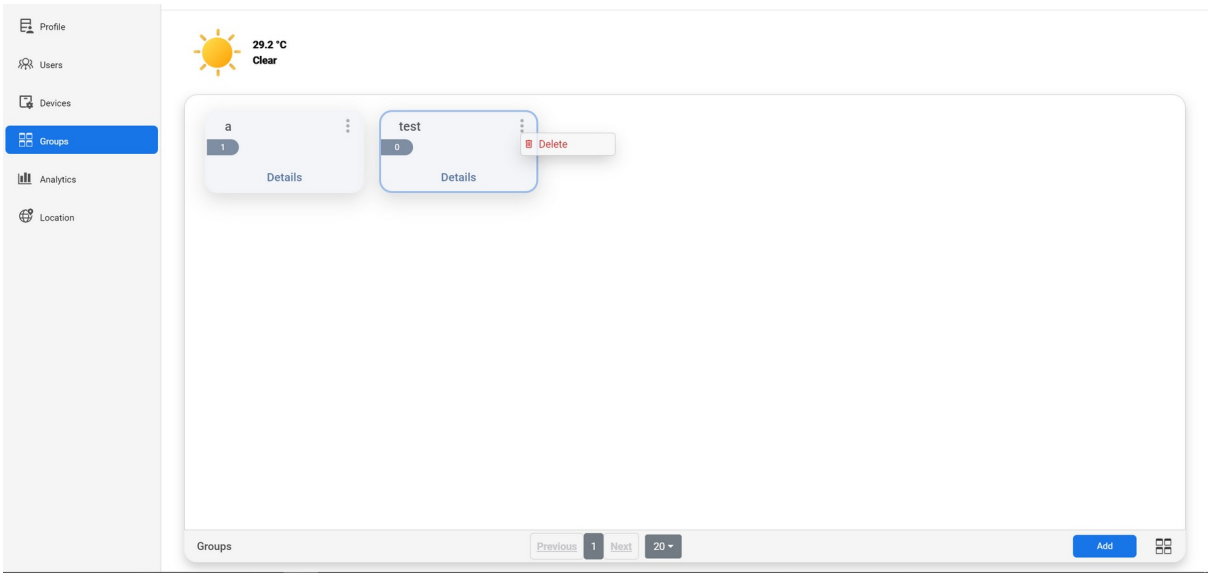

Рисунок 3.21 – Сторінка з групами пристроїв

При натисканні на кнопку «Деталі» відкривається модальне вікно з інформацією про групу пристроїв (рис 3.22). Саме тут можна задати стан

пристроїв в групі, назначити певне правило роботи пристроїв або видалити пристрій з групи.

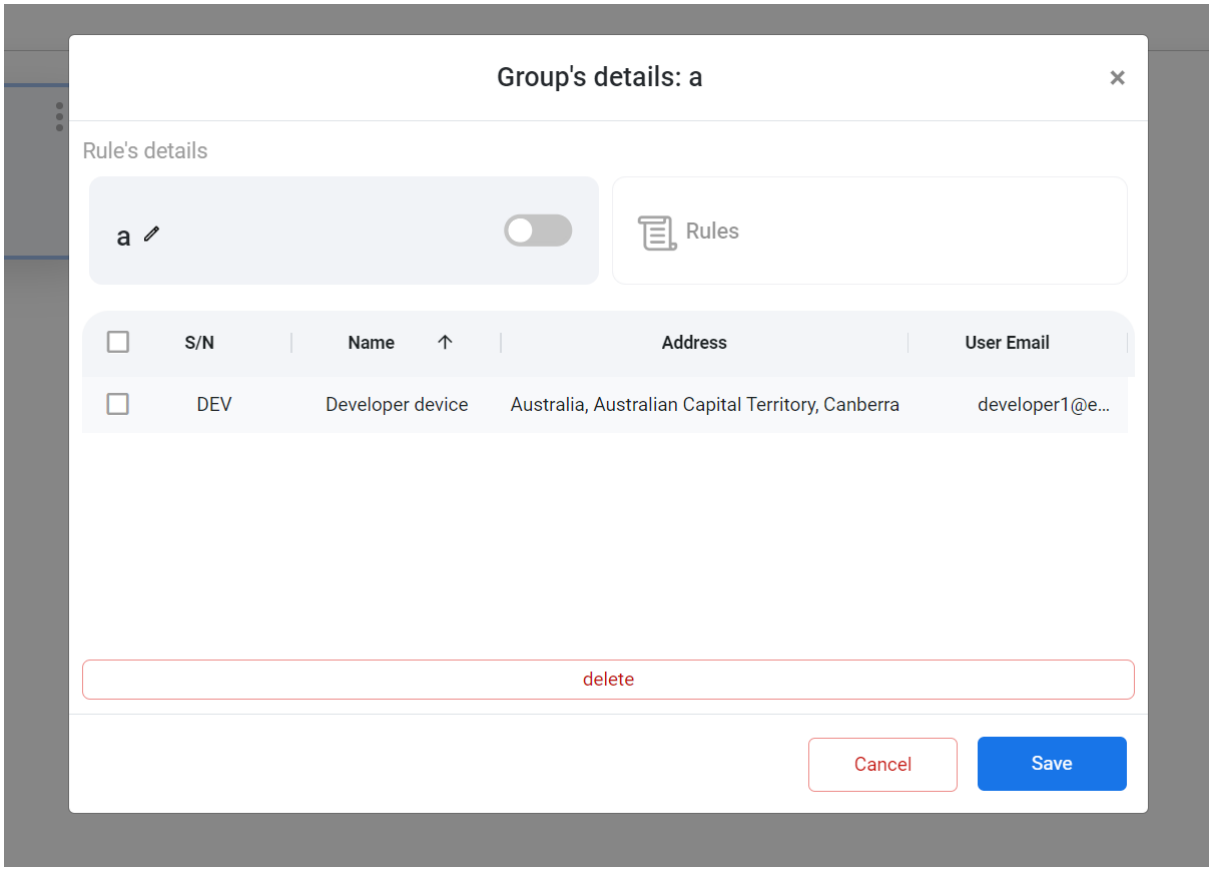

Рисунок 3.22 – Модальне вікно керуванням групою пристроїв

При натисканні на розділ «Місцезнаходження» відкривається мапа з маркерами місцезнаходження пристроїв (рис. 3.23).

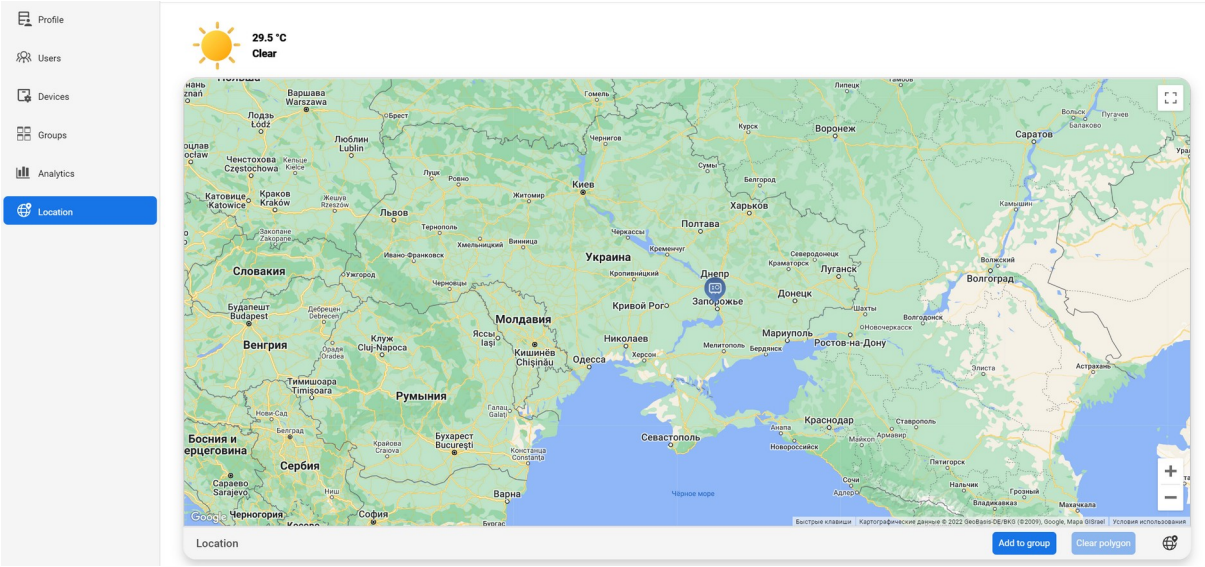

Рисунок 3.23 – Мапа місцезнаходження пристроїв

При натисканні на маркер пристрою з’являється картка з інформацією про пристрій (рис 3.24).

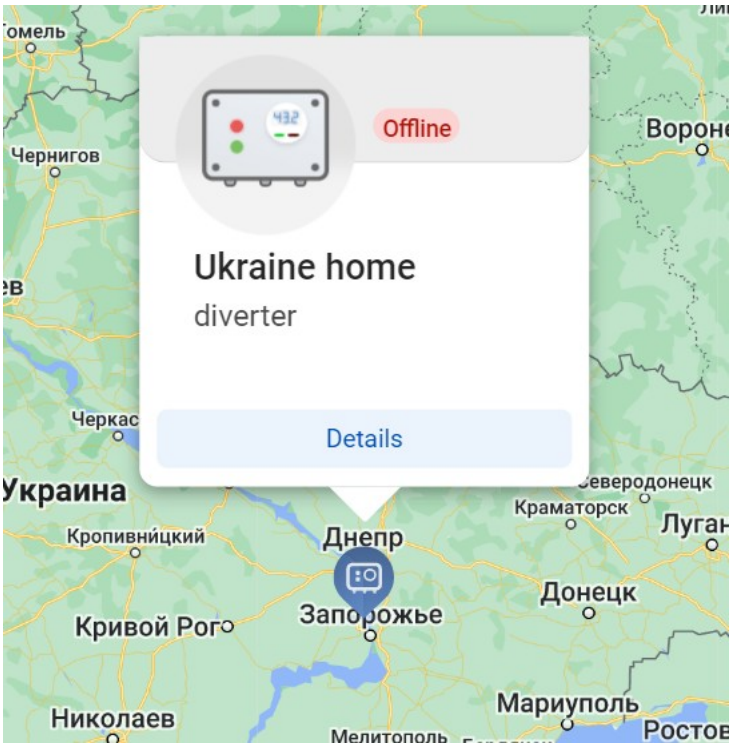

Рисунок 3.24 – Картка з інформацією про пристрій на мапі

При натисканні на кнопку «Деталі» відкривається модальне вікно з інформацією про пристрій (рис 3.19).

При натисненні на розділ «Правила» відкривається сторінка з інформацією про правила роботи пристроїв (рис. 3.25).

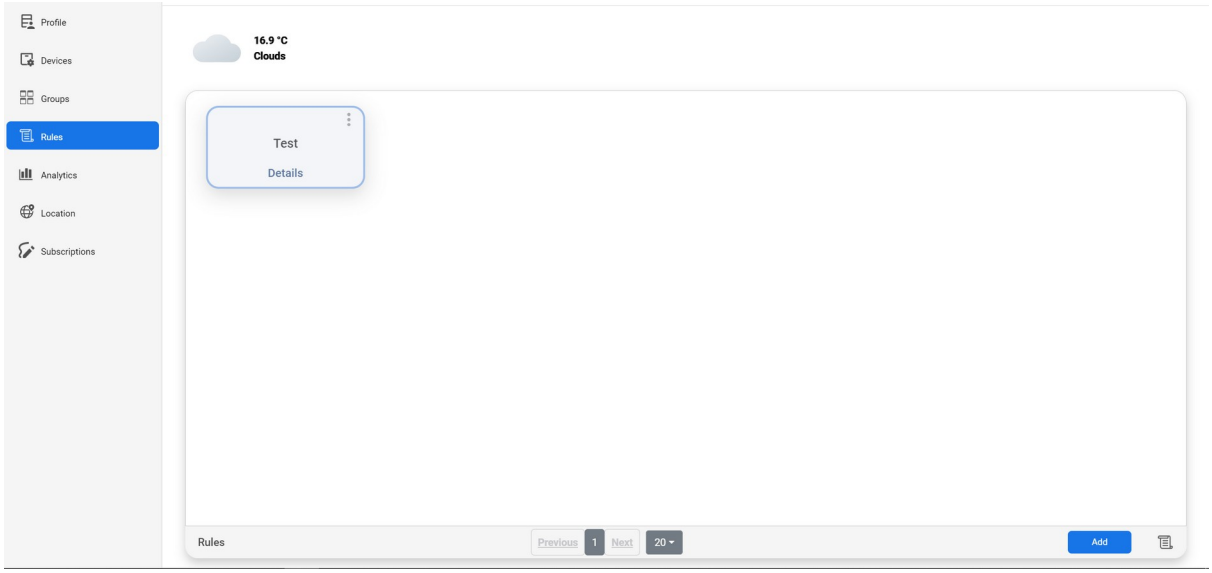

Рисунок 3.25 – Сторінка з правилами роботи пристрою

Саме тут визначається графік й режим роботи приладу. При натисканні на кнопку «Деталі» відкривається модальне вікно з інформацією про правило (рис 3.26).

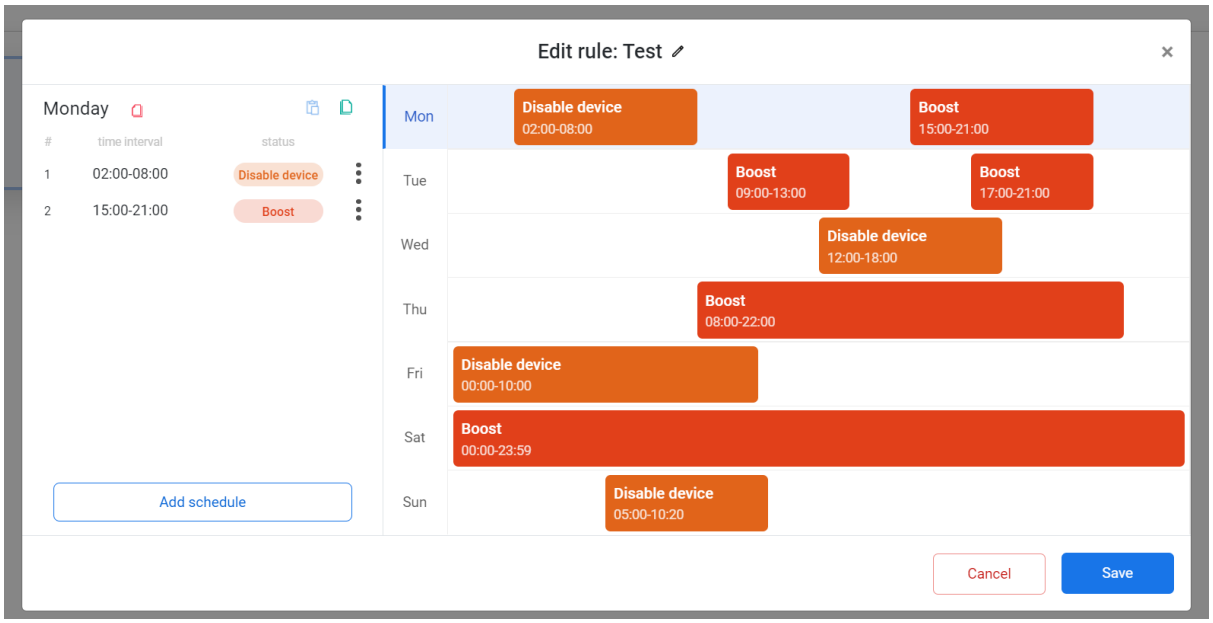

Рисунок 3.26 – Модальне вікно з інформацією про правило

Завдяки натисканню на назви днів тижня, можна переключатись між днями правила. Також у цьому модальному вікні присутні кнопки очищення, копіювання й перенесення графіку дня, створення нового часового проміжку. Код для генерації відображення часових проміжків на основі відповіді з API зображено на рисунку 3.27.

```
const convertToRule = (list:any)=>{

    let arrayOfDays = Object.keys(list);
    let intervalses: any = {}
    let toarr={timeFrom:"",timeTo:"",command:""}
    arrayOfDays.forEach(day => {
        intervalses[day]=[]

        list[day].forEach( (stamp: any) => {

            let apiTime="t".concat((OutputHours(stamp.fromTime).toString()).concat((OutputMinutes(stamp.fromTime)).toString()))

            let time=apiTime.substring(1);
            time=time.substring(0, 2) + ":" + time.substring(2, time.length)
            toarr.timeFrom=time;

            apiTime="t".concat((OutputHours(stamp.toTime).toString()).concat((OutputMinutes(stamp.toTime)).toString()))
            time=apiTime.substring(1);
            time=time.substring(0, 2) + ":" + time.substring(2, time.length)
            toarr.timeTo=time;
            toarr.command=stamp.status.value;
            let teste=intervalses[day]
            teste.push({...toarr})
            intervalses[day]=teste
        })
    })
    console.log("end result", intervalses)
    return {...intervalses}
}
```

Рисунок 3.27 – Код для генерації відображення часових проміжків правила

При натисканні на кнопку «Додати інтервал», відкривається меню (рис. 3.28) для додавання нового часового інтервалу, де користувач вводить часовий період і режим роботи девайсу (відключити або пришвидшений режим).

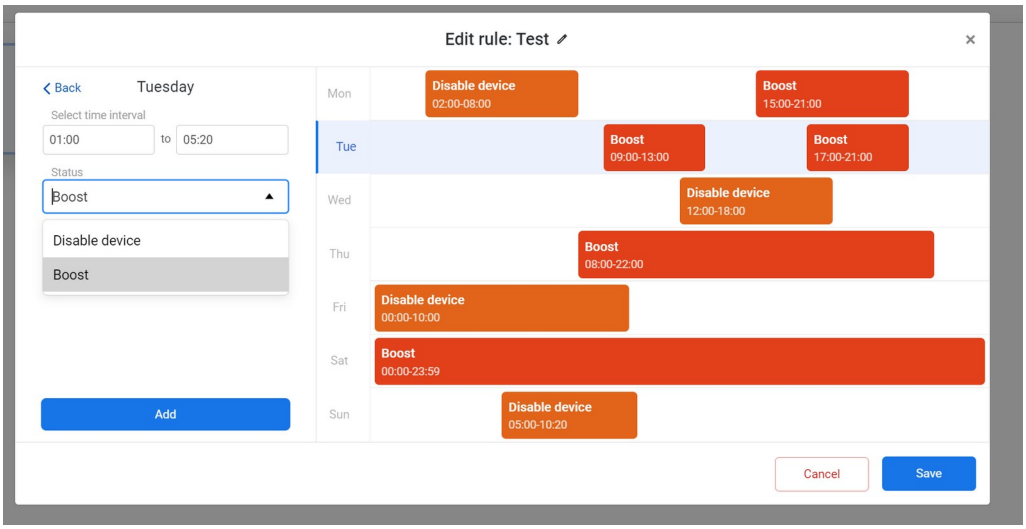

Рисунок 3.28 – Меню додавання нового часового проміжку

При натисканні на розділ «Аналітика» відкриється сторінка з відображенням статистики роботи пристрою (рис. 3.29). Тут можна переглянути статистику роботи пристрою, відфільтрувати дані чи змінити їх представлення (графік або діаграма).

Рисунок 3.29 – сторінка аналітики даних пристрою

3.3.2 Розробка серверної частини

Скрипт маршрутизації API, ініціалізації його работи знаходиться у файлі «server.php» (лістинг 3.1).

Лістинг 3.1 – Код файлу «server.php»

```
<?php
$uri = urldecode(
  parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH)
);
if ($uri !== '/' && file_exists(__DIR__.'/public'.$uri)) {
    return false;
}
require_once __DIR__.'/public/index.php';
```

Енд-поінти API визначаються у різних файлах директорії «app». Основними енд-поінтами являються:

- «/login» для авторизації користувача (POST запит);
- «/refresh» для оновлення токену авторизації користувача (POST запит);
- «/signup» для реєстрації користувача (POST запит);
- «/avatars» для отримання зображень користувачів (GET запит);
- «/devices» для роботи з пристроями (POST, GET, PUT, DELETE запити);

- «/groups» для роботи з групами (POST, GET, PUT, DELETE запити);
- «/geo» для роботи з місцезнаходженням користувача (POST, GET запити);
- «/messages» для роботи із повідомленнями (POST, GET, PUT, DELETE запити);
- «/rules» для роботи з правилами (POST, GET, PUT, DELETE запити);
- «/users» для роботи з користувачами (POST, GET, PUT, DELETE запити);
- «/plans» для роботи з тарифними планами (POST, GET запити).

Приклад запиту для створення нового пристрою показано на рисунку 3.31.

Рисунок 3.31 – Дані для створення нового пристрою (POST запит)

3.4 Тестування веб сервісу

Для тестування веб сервісу було виконано наступний перелік дій:

- було перевірено усі енд поінти API;
- на сторінці реєстрації було введено неправильні дані, дані вже існуючого користувача: при цих умовах на відображалися сповіщення про помилки;
- на сторінці авторизації було не введено певні данні, введено дані неіснуючого користувача та невірний пароль: при цих умовах відображалися сповіщення про помилку, при правильних умовах авторизація проходила коректно;
- для нового користувача було створено нові пристрої;
- в меню профілю користувача було натиснено на кнопку «Вийти з аккаунту», сесія була завершена коректно;
- було перевірено динамічне оновлення даних на сторінці з пристроями користувача, дані оновились правильно;
- на сторінці управління всіма пристроями з аккаунту адміністратора було натиснено на кнопку видалення, з’явилось модальне вікно з підтвердженням: при підтверджені видалення пристрій видалявся, з’являлося повідомлення, за відміни пристрій не було видалено ;
- при створені пристрою не був вказаний часовий пояс пристрою, кнопка продовження була заблокованою;
- у модальному вікні з правилами усі графіки відображались коректно;
- на сторінці аналітики усі діаграми відображались коректно;
- у модальному вікні детальної інформації про пристрій перемикач, кнопки правил та груп відпрацювали коректно;
- при відключені пристрою у верхньому правому куті відображались повідомлення про відключення пристрою певним користувачем;

- на сторінці з групами при натисненні на кнопку «Видалити» після підтвердження група видалялась коректно;
- на сторінці з мапою усі маркери пристроїв відображались коректно;
- на сторінці з користувачами при натисненні на кнопку «Видалити» після підтвердження користувач видалявся коректно ;
- у модальному вікні детальної інформації про групу перемикач і кнопка правил відпрацювали коректно.

Після тестування жодних помилок не виявлено. Тому визначено, що веб сервіс є стабільним та доступним для практичного використання.

3.5 Вимоги до програмного та апаратного забезпечення користувача та сервера

На основі тестування програмного продукту на декількох ПК з різним апаратним забезпеченням , а також на основі мінімальних системних вимог операційних систем визначено мінімальні системні вимоги для сервера та користувача веб застосунку.

Мінімальні системні вимоги для сервера:

- процесор: INTEL Core i7 10700KF або AMD Ryzen 5 3600;
- оперативна пам’ять 8 Гб;
- відеоадаптер роздільної здатності від 1280x720;
- твердотілий накопичувач від 128 Гб;
- наявність монітору;
- наявність миші;
- наявність клавіатури;
- доступ до мережі інтернет.

Мінімальні системні вимоги для користувача веб застосунку:

- процесор: INTEL Core i5 9400F або AMD Ryzen 3 3300X;

- оперативна пам’ять 4 Гб;
- твердотілий накопичувач від 32 Гб;
- відеоадаптер роздільної здатності від 1280x720;
- наявність монітору;
- наявність миші;
- наявність клавіатури;
- доступ до мережі інтернет.

3.6 Інструкція до розгортки веб сервісу

Дана інструкція призначена для ОС Ubuntu Desktop. Всі команди виконуються у терміналі bash від імені суперкористувача (даний режим активується командою «sudo -s»). Команди для терміналу bash від імені суперкористувача у лістингах позначені символом «$».

Для встановлення веб серверу Apache необхідно виконати команди з лістингу 3.2 .

Лістинг 3.2 – Встановлення Apache Web Server

```
$ apt update
$ apt install apache2
$ ufw allow ‘Apache’
$ sudo mkdir /var/www/telemetry.com
$ chown -R $USER:$USER /var/www/telemetry.com
$ -R 755 /var/www/telemetry.com
$ nano /etc/apache2/sites-available/ telemetry.com.conf
```

Далі, для налаштування веб серверу Apache, необхідно використати код конфігурації віртуального хосту з лістингу 3.3.

Лістинг 3.3 – Налаштування веб серверу Apache

```
<VirtualHost *:80>
    ServerAdmin mihalskiy@localhost
    ServerName telemetry.com
    ServerAlias www.telemetry.com
    DocumentRoot /var/www/telemetry.com
</VirtualHost>
```

Для завершення налаштування веб серверу Apache, і запуску віртуального хоста за визначеними параметрами необхідно виконати команди з лістингу 3.3.

Лістинг 3.3 – Запуск віртуального хоста Apache Web Server

```
$ a2ensite telemetry.com.conf
$ a2dissite 000-default.conf
$ apache2ctl configtest
$ systemctl restart apache2
```

У репозиторії на GitHub «telemetry-server» знаходяться усі необхідні файли, які треба розташувати за шляхом «/var/www/» (рис 3.44).

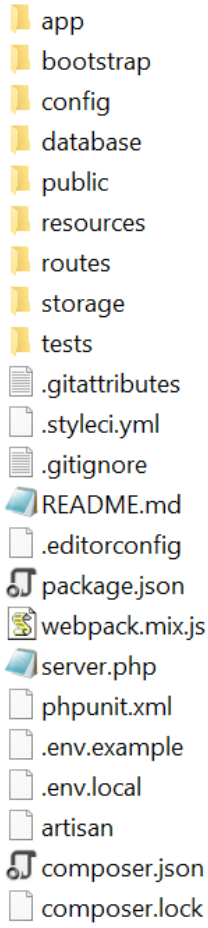

Рисунок 3.32 – файли вебсерверу

Для встановлення СКБД MySQL необхідно виконати команди з лістингу 3.4.

Лістинг 3.4 – Встановлення СКБД MySQL

```
$ apt update
$ apt install mysql-server
$ mysql_secure_installation
```

Для налаштування MySQL необхідно виконати команди з лістингу 3.5 зі своїми параметрами для імені, паролю користувача, назви бази

даних. Команди для командного інтерфейсу MySQL (викликається командою «sql») у лістингах позначені символом «>».

Лістинг 3.5 – Налаштування MySQL

```
$ mysql
> SELECT user,authentication_string,plugin,host FROM
mysql.user; ALTER USER 'root'@'localhost' IDENTIFIED WITH
caching_sha2_password BY 'password';
> FLUSH PRIVILEGES;
> exit
$ mysql -u root -p
> CREATE USER 'username'@'localhost' IDENTIFIED BY
'password';
> GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost'
WITH GRANT OPTION;
> CREATE DATABASE db;
> exit
```

Далі для налаштування веб сервісу необхідно заповнити файл «.env.local» із своїми налаштуваннями, а саме вказати свої данні в полях, указаних у лістингу 3.6.

Лістинг 3.6 – Обов’язкові налаштування у файлі «.env.local»

```
APP_URL= server_url
DB_CONNECTION= mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE= db_name
DB_USERNAME= db_username
```

DB_PASSWORD= db_password

Для активації веб сервісу, створення криптографічних ключів для генерації токенів і створення таблиць в базі даних, необхідно в каталозі з файлами веб сервісу в терміналі bash виконати команди, указані у лістингу 3.7.

Лістинг 3.7 – Команди для активації веб сервісу

```
composer install
php artisan migrate
php artisan passport:install
php artisan db:seed
```

3.7 Висновки за розділом

У даному розділі було:

- побудовано UML діаграми;
- визначено файлову структуру проекту;
- визначено архітектуру проекту;
- розроблено проект;
- оформлено процес й деталі розробки;
- протестовано проект, визначено його стабільність;
- визначено вимоги до апаратного забезпечення користувача та сервера;
- оформлено інструкцію до розгортки веб сервісу на сервері.

ВИСНОВОК

У повсякденному житті людина використовує різноманітні електропристрої у різних сферах життя. Проте для керування ними необхідно находитись поруч із пристроєм, що може викликати певні труднощі. Для того щоб керувати пристроєм і дізнатись його стан дистанційно розробляються спеціальні сервіси телеметрії для віддаленого доступу до пристрою.

Метою роботи було створення веб сервісу для дистанційної роботи з приладами, переглядом їх стану.

У ході виконання випускної роботи було:

- проаналізовано актуальність теми;
- оглянуто предметну область, а саме основні визначення, поняття, технічні та архітектурі особливості веб застосунків;
- розглянуто існуючі аналоги, їх переваги й недоліки;
- сформовано список задач проекту;
- оглянуто стек технологій для проекту;
- побудовано UML діаграми;
- визначено архітектуру проекту;
- визначено файлову структуру проекту;
- розроблено проект;
- оформлено процес розробки та реалізації;
- протестовано проект, визначено його стабільність;
- визначено вимоги до апаратного забезпечення користувача та сервера;
- оформлено інструкцію до розгортки веб сервісу на сервері;
- оформлено звіт за результатами виконання.

Результатом виконання випускної роботи молодшого спеціаліста є веб сервіс телеметрії та управління віддаленими пристроями .

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.https://www.greyb.com/smart-home-market/

2.https://uk.wikipedia.org/wiki

3 https://eldeprocess.ru/texnologii/avtomatization-process/skada-sistemy/

4. https://w3techs.com/

5. https://en.wikipedia.org/wiki/PHP

6. https://en.wikipedia.org/wiki/MySQL

7. https://www.openlogic.com/resources/2022-open-source-report

8. https://www.php.net/manual/ru/ref.pdo-mysql.php

9. https://en.wikipedia.org/wiki/Ubuntu

10. https://www.redhat.com/fr/blog/red-hat-continues-lead-linux-server-market

11. https://www.openstack.org/analytics/

12. https://ubuntu.com/download/desktop

13. https://hyperhost.ua/info/ru/ubuntu-opisanie-os-aktualnyie-versii-plyus

14. https://en.wikipedia.org/wiki/HTML

15. https://en.wikipedia.org/wiki/CSS

16. https://ru.wikipedia.org/wiki/JavaScript

17. https://en.wikipedia.org/wiki/TypeScript

18. https://en.wikipedia.org/wiki/React_(JavaScript_library)

19. https://ru.wikipedia.org/wiki/WebStorm

20. https://en.wikipedia.org/wiki/Postman_(software)