НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО
В.о. завідувача кафедри
_____Олександр КОВАЛЬ
«_____»_____2023р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»
спеціальності 121 Інженерія програмного забезпечення

на тему: «Веб-додаток візуалізації даних лабораторного стенду сонячної панелі»

Виконала:
студентка IV курсу, групи ТІ-91
Соболь Валерія Ігорівна _____
(підпис)

Керівник:
проф. каф. ІПЗЕ, д.т.н Федорова Н.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент:

_____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Київ – 2023

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

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ (підпис)
«____»_____202_р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Соболь Валерії Ігорівні_____
(прізвище, ім'я, по батькові)

1. Тема роботи
«Веб-додаток візуалізації даних лабораторного стенду сонячної панелі» керівник роботи Федорова Наталія Володимирівна д.т.н, професор кафедри ІПЗЕ_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "29" травня 2023 року №_2039-с___
2. Строк подання студентом роботи _12.06.2023_____
3. Вихідні дані до роботи: візуалізація даних лабораторного стенду сонячної панелі у вигляді графіків та таблиць для подальшого аналізу
_____
4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити): описати лабораторний стенд, розробити програмне забезпечення для візуалізації даних, проаналізовати отримані дані та визначено ефективність роботи сонячної панелі в різних умовах._____
_____
5. Перелік ілюстративного матеріалу: Склад сонячної панелі, Принцип роботи елементу сонячної панелі, Лабораторний стенд сонячної панелі, Epsolar MT50 мониторинговий дисплей, Схема підключення лабораторного стенду сонячної панелі, Діаграма розгортання, Діаграма прецедентів, UML діаграма класу пристрою, Перша сторінка веб-додатку, Сторінка виводу даних за проміжок часу, Таблиця даних, Сторінка виводу даних в реальному часі, Наявні пристрої для підключення, Форма додавання нових пристроїв

6. Дата видачі завдання «30» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	30.09.2022	виконано
2	Дослідження предметної області	02.10.2022 – 20.02.2023	виконано
3	Постановка вимог до проєктування системи	21.02.2023 – 28.02.2023	виконано
4	Дослідження існуючих рішень	29.02.2023 – 16.04.2023	виконано
5	Розробка програмного продукту	17.04.2023 – 10.05.2023	виконано
6	Тестування	11.05.2023 – 21.05.2023	виконано
7	Захист програмного продукту	17.05.2023	виконано
8	Оформлення дипломної роботи	22.05.2023 – 04.06.2023	виконано
9	Передзахист	5.06.2023	виконано
10	Захист	19.06.2023	виконано

Студент _____ Валерія Соболь
(підпис) (ім'я, прізвище)

Керівник роботи _____ Наталія Федорова
(підпис) (ім'я, прізвище)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Робота містить 53 сторінки, 18 рисунків, 1 додаток та 17 посилань.

Метою роботи є розробка програмного забезпечення для візуалізації даних лабораторного стенду сонячної панелі, що дозволить отримати графічну та статистичну інформацію про ефективність роботи панелі, а також провести аналіз отриманих даних.

Для досягнення поставленої мети виконано наступні завдання:

- описано лабораторний стенду та методи збору даних про роботу сонячної панелі;
- розроблено програмне забезпечення для візуалізації даних;
- проаналізовано отримані дані та визначено ефективність роботи сонячної панелі в різних умовах.

Розроблено програмну систему, що надає змогу користувачеві отримувати дані з сонячної панелі у режимі реального часу, візуалізувати їх у вигляді графіків та таблиць

Практичне значення одержаних результатів полягає в отриманні інструменту, що дозволить отримувати та аналізувати інформацію про ефективність сонячної панелі, дозволить під’єднання широкого спектру сонячних панелей за допомогою створення додаткових розширюваних модулів системи та уніфікований користувацький інтерфейс, що надає інформацію у табличному вигляді, та у вигляді графіків для оцінки.

Ключові слова: сонячна панель, збір даних, візуалізація даних, лабораторний стенд, мікросервіси, високонавантажена система.

ABSTRACT

Structure and scope of the thesis. The work contains 53 pages, 18 figures, 1 appendice and 17 references.

The purpose of the work is to develop software for data visualization of the solar panel laboratory stand, which will allow to obtain graphic and statistical information about the efficiency of the panel, as well as to analyze the received data.

To achieve the set goal, the following tasks were completed:

- the laboratory stand and methods of collecting data on the operation of the solar panel are described;
- software for data visualization was developed;
- the obtained data were analyzed and the efficiency of the solar panel was determined in different conditions.

A software system has been developed that allows the user to receive data from the solar panel in real time, visualize them in the form of graphs and tables

The practical value of the obtained results is to obtain a tool that will allow obtaining and analyzing information about the efficiency of a solar panel, will allow the connection of a wide range of solar panels by creating additional expandable system modules, and a unified user interface that provides information in tabular form and in the form of graphs for evaluations

Keywords: solar panel, data collection, data visualization, laboratory bench, microservices, highly loaded system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Сервер – в залежності від контексту, може бути як фізичний компонент – спеціалізований комп'ютер з операційною системою налаштованою під запуск спеціалізованого програмного продукту для обробки даних/запитів, роботи із базами даних, так і сам спеціалізований програмний продукт.

Клієнт – частина програмного застосунку, з якою взаємодіє кінцевий користувач.

Контейнер – віртуально відділений процес із програмним продуктом, що виконує певні дії та запущений через платформу для контейнеризації

Контейнеризація – процес створення контейнеру – віртуального середовища для запуску певного спеціалізованого програмного продукту для запуску на платформі контейнеризації.

Платформа контейнеризації – спеціалізоване програмне забезпечення, що може бути встановлено на фізичному сервері, чи простому комп'ютері, для запуску великої кількості контейнерів (окремого програмного продукту)

Мікросервіс – одиниця програмного серверного продукту, що за архітектурою є однією складовою системи та розроблена з ціллю виконувати одну певну задачу. Запуск декількох однакових мікросервісів дозволяє горизонтальне масштабування цих одиниць задля збільшення продуктивності всього програмного застосунку.

Горизонтальне масштабування – архітектурний термін, що описує спосіб запуску програмного продукту способом запуску n-ної кількості одного і того ж мікросервісу без конфліктів між собою і дозволяє збільшити пропускну здатність запитів та кількість задач, що виконуються одночасно у самій системі.

Дебаггер – програма, що йде разом із компілятором/інтерпретатором мови програмування для відлагодження коду, що дозволяє маніпулювати внутрішнім станом програми під час її роботи задля пошуку та виправлення помилок.

ВСТУП

Зважаючи на активний розвиток використання відновлювальних джерел енергії, сонячна енергія є однією з найбільш перспективних галузей. Ця енергія відносно дешева та безпечна для довкілля, тому вона стає все більш популярною.

Сонячна енергія вже забезпечує електричною енергією тисячі будівель та домогосподарств по всьому світу.

Важливо враховувати, що останнім часом в Україні спостерігається активний розвиток використання сонячної енергії. Сонячна енергія стає все більш ефективною та доступною для використання, а це сприяє зменшенню залежності України від імпорту енергоресурсів та зниженню витрат на оплату електроенергії для населення та промисловості.

Крім того, Україна має великий потенціал для використання сонячної енергії, оскільки має достатню кількість сонячних днів та розвинуту сонячну енергетику, що сприяє збільшенню кількості сонячних електростанцій та розвитку відповідної інфраструктури.

Отже, розвиток сонячної енергетики в Україні є важливим напрямом для забезпечення сталого економічного розвитку країни та зменшення негативного впливу на навколишнє середовище. Використання сонячної енергії може стати ключовим чинником для забезпечення енергетичної незалежності України та підвищення якості життя громадян.

Одним з ключових елементів сонячної енергетики є сонячні панелі, які перетворюють сонячну енергію на електричну. Проте, для ефективного використання сонячних панелей необхідно мати змогу аналізувати та візуалізувати дані, отримані з них.

Отримані результати досліджень представлені у вигляді графіків та діаграм, які дозволять зрозуміти ефективність роботи сонячної панелі та провести аналіз її роботи в різних умовах. Крім того, в результаті досліджень можуть бути зроблені

рекомендації щодо поліпшення ефективності роботи сонячної панелі, які будуть корисні для виробників та споживачів.

У зв'язку з високою актуальністю даної теми та перспективністю використання сонячної енергії, результати досліджень, проведених у рамках даної бакалаврської дипломної роботи, можуть бути корисними для науковців, енергетиків, виробників та споживачів сонячних панелей.

У процесі виконання роботи будуть використані такі методи досліджень, як статистичний аналіз даних, візуалізація даних, аналіз даних та моделювання системи сонячної панелі.

Отже, дана бакалаврська дипломна робота є актуальною та перспективною, оскільки сонячна енергія стає все популярнішою та ефективнішою альтернативою традиційним джерелам енергії.

1 МЕТА, ПОСТАНОВКА ЗАДАЧІ ТА ОСНОВНІ ЗАВДАННЯ

Метою даної бакалаврської дипломної роботи є розробка програмного забезпечення для візуалізації даних лабораторного стенду сонячної панелі, що дозволить отримати графічну та статистичну інформацію про ефективність роботи панелі, а також провести аналіз отриманих даних.

У процесі розробки програмного забезпечення будуть використані сучасні методи візуалізації даних, такі як графіки, діаграми, інфографіки тощо. Також будуть враховані можливості збору даних, що отримуються з лабораторного стенду.

У роботі будуть використані сучасні програмні інструменти, такі як мова програмування Python, бібліотека Chart.js для візуалізації даних, та Flask – фреймворк для створення веб-додатків, що дозволяє створити інтерактивний інтерфейс для користувачів.

У роботі будуть вирішені такі завдання:

- опис лабораторного стенду та збір даних про роботу сонячної панелі;
- розробка програмного забезпечення для візуалізації даних;
- аналіз отриманих даних та визначення ефективності роботи сонячної панелі в різних умовах.

Нижче описані вимоги до складових частин програмного продукту.

Серверна частина повинна відповідати наступним вимогам:

- серверна частина повинна отримувати та оброблювати дані про вироблення енергії сонячної панелі;
- серверна частина має зберігати оброблені дані у базу даних;
- серверна частина має надавати змогу надсилати дані у реальному часі у вигляду потоку даних.

Клієнтська частина має відповідати наступним вимогам:

- містити інформацію про дані, що збережені у базі даних для перегляду історичних даних за певний період;
- містити спосіб перегляду інформації про дані у реальному часі;
- містити блок із пристроями для моделювання сценаріїв під'єднання пристроїв у реальному часі;
- має надавати змогу візуалізації даних у вигляді графіків та таблиць.

База даних має відповідати наступним вимогам

- для виконання поставленої задачі має бути реляційна база даних;
- база даних має бути структурованою;
- база даних має підтримувати цілісність даних та реплікацію на випадки непередбачуваних ситуацій і збереження даних;
- база даних має бути упакована в контейнер Docker.

Висновки до розділу 1

У першому розділі було описано мету даної бакалаврської дипломної роботи, проаналізовано та описано основні вимоги до складових частин програмного продукту. Коротко описано про початкові інструменти для розробки, виділено та описано задачі, які потрібно розв'язати під час написання роботи.

2 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ, МЕТОДІВ, МОДЕЛЕЙ, ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Основним елементом для проведення дослідження та тестування виступає лабораторний стенд сонячної панелі, який складається з сонячної панелі Epsolar, акумуляторної батареї та моніторингового дисплею Epsolar MT50.

До апаратних частин моніторингу виробниками створені додатки для перегляду статистики, проте зазвичай мають недолік – вони прив’язані до конкретного апаратного модулю збору статистики.

Розглянемо деякі з таких програмних додатків.

2.1 Аналіз існуючих систем

На ринку застосунків представлено багато альтернатив серед додатків моніторингу роботи сонячної панелі, що дозволяють власникам сонячних електростанцій контролювати ефективність роботи своїх систем в режимі реального часу. Такі додатки зазвичай містять інформацію про вироблену електроенергію, відсоток використання панелей та їх ефективність. Деякі з додатків також можуть включати функції аналізу погодних умов, які дозволяють користувачам отримувати прогнози змін виробленої електроенергії в залежності від погодних умов.

2.1.1 MySolarEdge Monitoring App

На рисунку 2.1 зображений додаток для моніторингу MySolarEdge забезпечує розширену функцію моніторингу продуктивності фотоелектричної системи. Це також допомагає гарантувати врожайність шляхом негайного виявлення несправностей. Він також надає сповіщення на рівнях модулів, системних і рядкових рівнів.

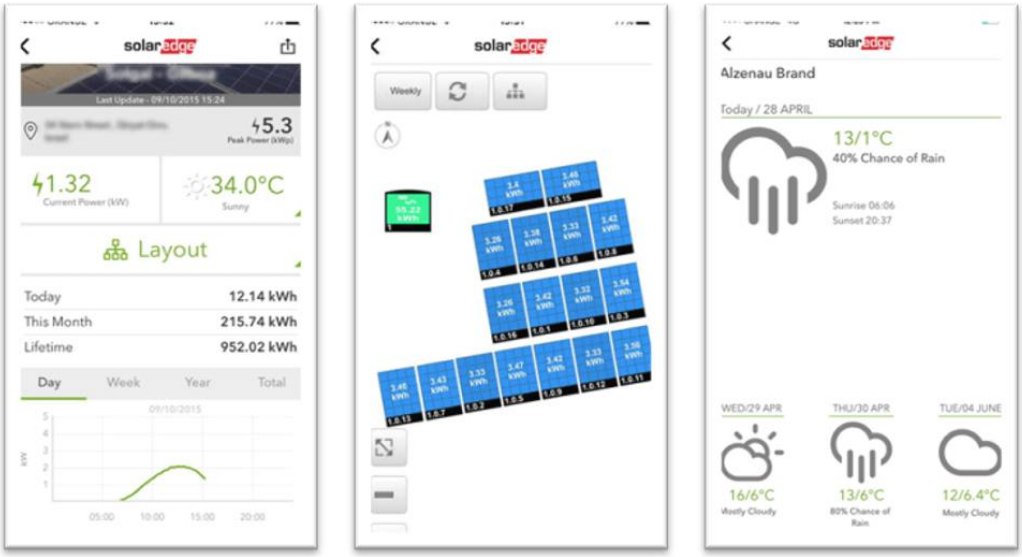

Рисунок 2.1 – Додаток для моніторингу MySolarEdge

Додаток для моніторингу MySolarEdge забезпечує розширену функцію моніторингу продуктивності фотоелектричної системи. Це також допомагає гарантувати врожайність шляхом негайного виявлення несправностей. Він також надає сповіщення на рівнях модулів, системних і рядкових рівнів.

Для передачі даних сонячної панелі використовуються бездротові технології. Це також стосується даних оптимізатора потужності, що надходять до сонячного інвертора. Передавачі та датчики моніторингу підключаються до оптимізатора потужності та сонячного інвертора. Вони також вимірюють дані, які передаються через звичайні лінії живлення.

Програма доступна на таких пристроях, як планшети та комп’ютери з доступом до Інтернету.

Переваги:

- автоматичні сповіщення про будь-які системні проблеми;
- забезпечує повну видимість функціональності сонячної системи;
- допомагає виявити надмірне споживання енергії.

Недоліки:

- немає можливості контролювати поведінку зарядки.

Особливості:

- вимірювання поточної енергії системи;
- моніторинг вироблення енергії в місяцях, днях, а також протягом життя;
- відображення графіка потужності;
- список сайтів і зображення;
- дані про погоду;
- індивідуальна конфігурація налаштувань користувача.

2.1.2 Enphase Enlighten App

На рисунку 2.2 зображений додаток для моніторингу Enphase Enlighten App

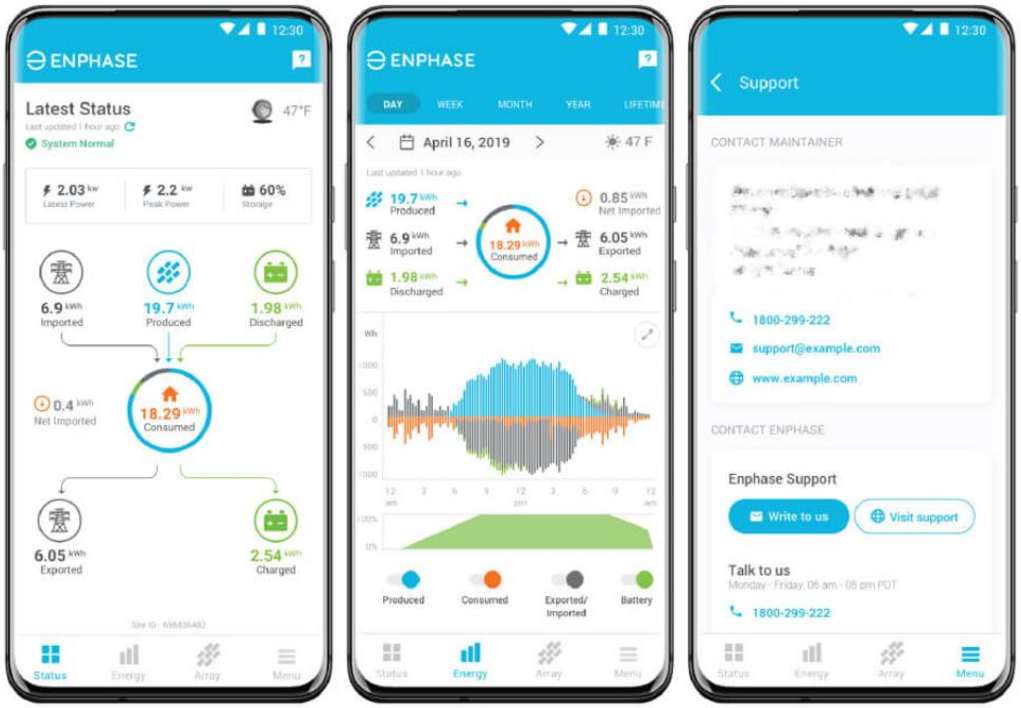

Рисунок 2.2 – Додаток для моніторингу Enphase Enlighten App

Додаток дозволяє занурюватися в важливі деталі стану та продуктивності системи.

Додаток також надає простий огляд споживання або зберігання енергії. Головний розділ цього додатка для моніторингу сонячної енергії фактично є веб-сайтом. Це означає, що додаток доступний будь-де де є доступ до мережі інтернету

Нещодавно запущений мобільний додаток допоможе вам відстежувати все це на ходу. Це дозволяє будь-якому власнику системи зробити правильний вибір для споживання енергії.

Плюси:

- перевірка споживання та зберігання енергії;
- індивідуальний контроль панелі для виявлення несправностей;
- дозволяє перевірити працездатність системи.

Недоліки:

- можливо, програма не зможе надати поточні дані.

Особливості:

- легка перевірка справності та продуктивності системи;
- аналіз виробництва енергії за день, годину або місяць;
- аналіз даних про погоду (попередній);
- звіти про виробництво та використання енергії.

2.1.3 Energy Monitoring & Analysis App (EMA)

Для користувачів продукції мікроінвертора APS існує додаток для моніторингу Energy Monitoring & Analysis App (EMA) (рис. 2.3). Додаток EMA дозволяє користувачам перевіряти продуктивність системи в реальному часі.

Рисунок 2.3 – Додаток для моніторингу EMA

Додаток також надає власникам системи доступ до відстеження продуктивності в реальному часі. Він також надає доступ до графічного представлення характеристик окремих сонячних панелей. Додаток дозволяє відстежувати вихідні дані системи.

Також є можливість розрахувати економію енергії або економію навколишнього середовища. Ці категорії включають дерева, галони бензину та викиди CO2.

Переваги:

- перемикання між різними функціями моніторингу;
- відстежуйте енергоспоживання за місяць, день, рік або протягом життя;
- легко доступна інформація про фотоелектричну систему.

Недоліки:

- проблеми з входом через певні помилки.

Особливості:

- детальне відстеження стека;

- розширені повідомлення про помилки;
- екологічна економія;
- розрахунок енергозбереження.

2.1.4 Fronius Solar.Web App

На рисунку 2.4 зображений додаток Fronius Solar.Web App, який використовується власниками сонячних систем і доступен у вигляді веб- та мобільної програми. За допомогою цього додатка можна відстежувати видачу енергії системою.

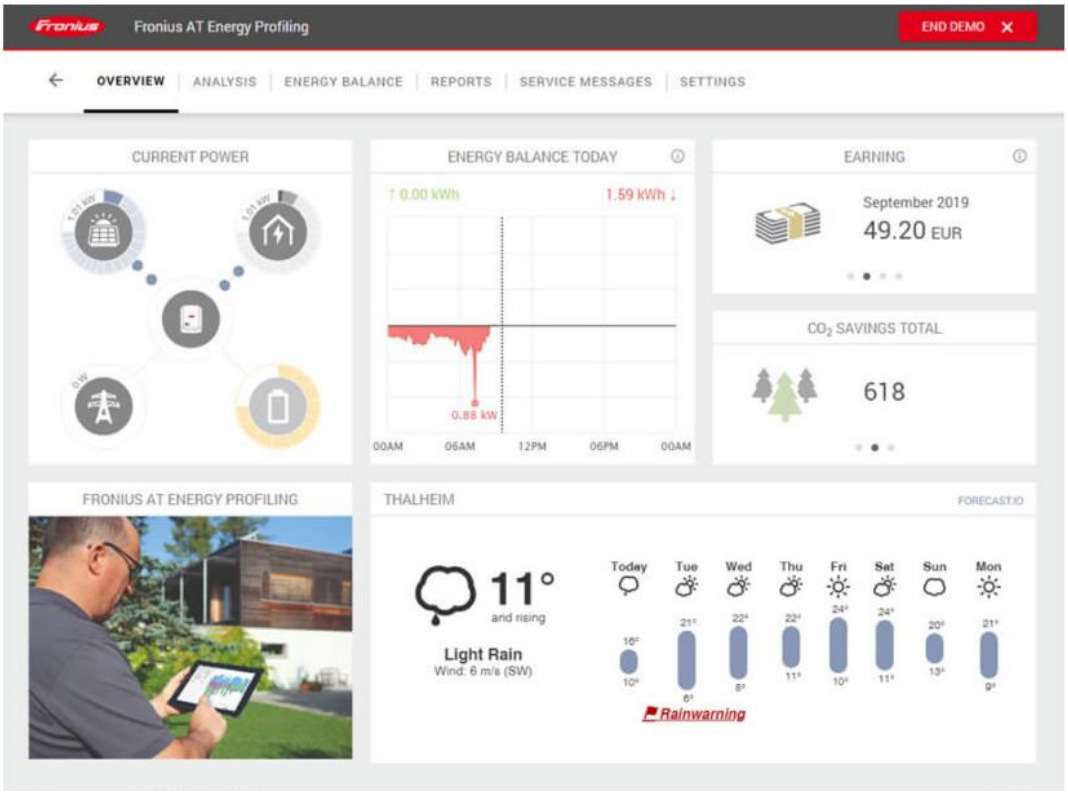

Рисунок 2.4 – Додаток Fronius Solar.Web App

Також є доступ до знімків виходу енергії за будь-який необхідний період часу та доступ до таких показників сталого розвитку, як висаджені дерева, заощадження CO2 і заощаджені гроші.

Переваги:

- сумісність з годинниками Apple;

- формування звіту про прибутковість у відведені терміни;
- легкий доступ до показників.

Недоліки:

- перегляд графіка може мати збої.

Особливості:

- доступний темний режим;
- відображення поточного значення та кривої енергії;
- аналіз та оцінка економії CO2 та виходу енергії;
- інтуїтивно зрозуміле управління;
- історія даних про використання енергії.

2.2 Опис предметної області

Тема дипломної роботи – візуалізація даних лабораторного стенду сонячної панелі, належить до предметної області сонячної енергетики. Сонячна енергетика є галуззю відновлювальної енергетики, яка використовує сонячну радіацію для виробництва електричної та теплової енергії. В Україні, як і в багатьох країнах світу, за останні кілька років спостерігається швидкий розвиток сонячної енергетики.

Основними елементами систем сонячної енергетики є сонячні панелі, інвертори, системи зберігання енергії та системи моніторингу. Сонячні панелі збирають сонячну радіацію та перетворюють її в електричну енергію. Інвертори забезпечують перетворення постійного струму, який генерується сонячними панелями, у змінний струм, який може бути використаний в електричних мережах. Системи зберігання енергії дозволяють зберігати електричну енергію для використання в нічний час або в періоди недостатньої сонячної радіації.

Системи моніторингу є важливим елементом сонячної енергетики, оскільки дозволяють відстежувати роботу систем та отримувати дані про їхню ефективність та продуктивність. Лабораторні стенди сонячної енергетики є важливим

інструментом для вивчення роботи сонячних систем та проведення експериментів з їх покращення.

Однією з ключових проблем сонячної енергетики є ефективне використ ання сонячного випромінювання для отримання електричної енергії. Тому важливо розробляти та вдосконалювати методи дослідження та аналізу роботи сонячних електростанцій та панелей, які дозволяють підвищувати їх ефективність.

Однією з ключових проблем сонячної енергетики є ефективне використання отриманої електричної енергії. Це досягається за рахунок розробки та використання новітніх технологій, що дозволяють підвищувати ефективність роботи сонячних електростанцій та панелей. До таких технологій належить, зокрема, використання високоефективних сонячних батарей, які забезпечують високу конверсію сонячної енергії в електричну, а також застосування різноманітних систем моніторингу та контролю роботи сонячних панелей.

В цьому контексті візуалізація даних лабораторного стенду сонячної панелі стає важливим елементом досліджень у галузі сонячної енергетики. Вона дозволяє відображати різноманітну інформацію про роботу сонячних електростанцій та панелей у зручному для сприйняття форматі, що дозволяє дослідникам більш детально вивчати процеси, що відбуваються в цих системах. Така візуалізація може бути використана для аналізу роботи окремих елементів сонячної електростанції або для оцінки її загальної ефективності.

2.2.1 Опис сонячної панелі

Основною частиною апаратної частини є модуль сонячної панелі, який встановлено у навчальній лабораторії.

Тож, сонячна панель, або фотовольтаїчний модуль, також відомий як сонячна панель, є спеціальною структурою, що складається з набору зв'язаних між собою фотоелектричних комірок. Кожна з цих комірок, також називаних селами, виготовлена з напівпровідника, наприклад кремнію, який є найпоширенішим

матеріалом для створення сонячних панелей, оскільки демонструє найвищу ефективність в даний час. На рисунку 2.5 зображено склад сонячної панелі.

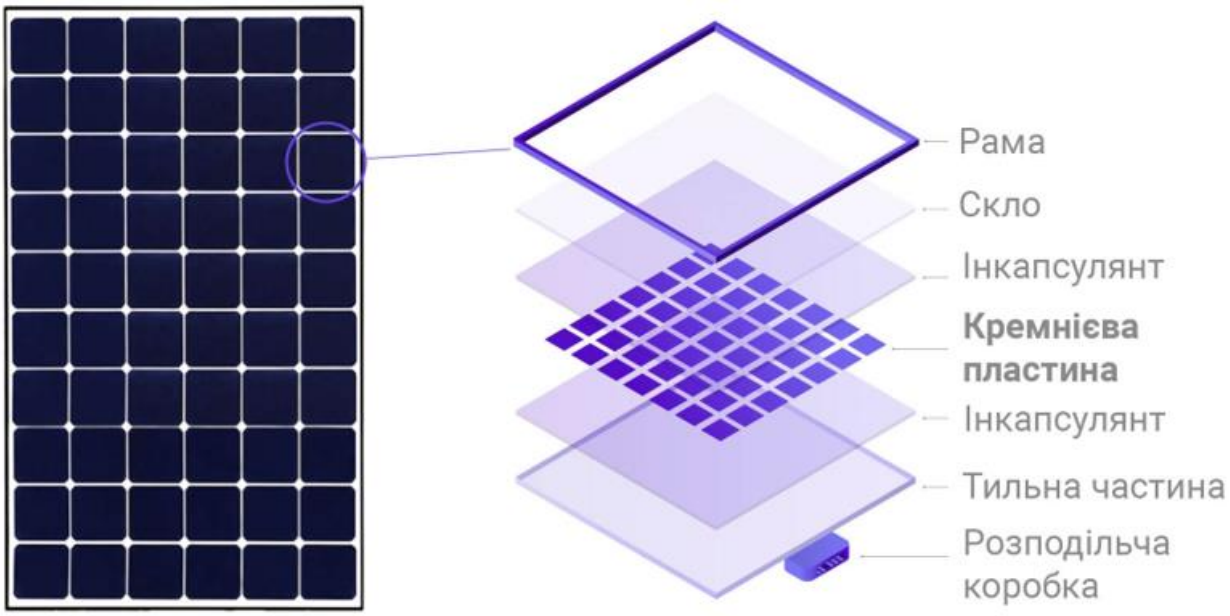

Рисунок 2.5 – Склад сонячної панелі

Коли сонячні промені падають на напівпровідник, він нагрівається і частково поглинає енергію, випромінювану світлом. Фотони відштовхують електрони від атомів напівпровідника, створюючи вільні електрони, які отримують заряд.

Замість однорідних шматочків кремнію кожна комірка складається з двох шарів напівпровідників. Однак кремній сам по собі не є хорошим провідником і тому його недостатньо для створення повного електричного поля. Для формування позитивних і негативних зарядів на шарі кремнію використовуються сторонні домішки.

Верхній шар кремнію насичений фосфором, який приносить додаткові негативно заряджені електрони. Цей шар називається «тип n» (від «негатив»). З іншого боку, нижній шар насичений бором, який зменшує кількість електронів і створює позитивний заряд. Цей шар називається «p-тип». Таким чином, між шарами кремнію утворюється електричне поле. Коли фотони сонячного світла стикаються з вільними електронами, електричне поле відштовхує їх від

кремнієвого переходу, створюючи електричний струм. Цей провідник називається «P-N».

Таким чином утворюється електричний заряд. Які практичні переваги цього? Кожна батарея оточена струмопровідними контактами та провідними пластинами, які направляють отриманий струм до системи. На рисунку 2.6 зображено принцип роботи елементу сонячної панелі.

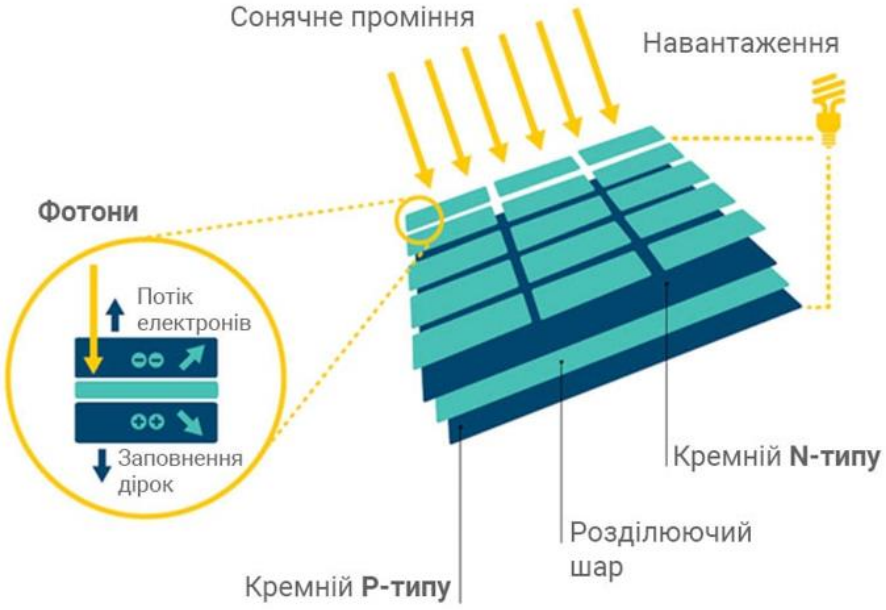

Рисунок 2.6 – Принцип роботи елементу сонячної панелі

Існує кілька факторів, що впливають на генерацію електроенергії сонячними панелями. Один з них - погодні умови, зокрема інтенсивність сонячної інсоляції. Для досягнення максимального виробітку, потрібна висока сонячна активність. Проте, занадто сильне нагрівання панелей може призвести до втрати потужності. Лабораторні дослідження показали, що при нагріванні панелі на 1 градус Цельсія вище номінальної температури (45 градусів Цельсія), втрати потужності становлять приблизно 0.4%. Це призводить до зниження продуктивності фотомодулів у зимовий період, коли рівень сонячної інсоляції нижчий.

Потужність сонячної панелі можна легко визначити. Якщо панель підключити до резисторного навантаження R, то в колі виникне струм I, величина якого залежить від роботи панелі. Потужність P обчислюється за формулою (2.1).

$$P = I * U, \quad (2.1)$$

де I – сила струму;

U – напруга.

Сонячні панелі працюють в системі шляхом взаємного з'єднання. Існують два способи підключення: послідовне та паралельне. При послідовному з'єднанні збільшується напруга на виході, а при паралельному - струм. Іноді комбінують обидва способи, щоб збільшити як напругу, так і потужність. Проте варто враховувати, що частіше застосовується послідовне підключення. Важливо підбирати панелі з однаковими характеристиками напруги та струму, що впливає на ефективність та загальну продуктивність системи [17].

2.2.2 Лабораторний стенд

Наданий лабораторний стенд (рис. 2.7) складається з установки Hanplast Solar SW Premium Plus.

Рисунок 2.7 – Лабораторний стенд сонячної панелі

Характеристики сонячної панелі:

- номінальна потужність – 275 Вт;
- заміряна потужність – 275.9 Вт;
- струм короткого замикання – 9.2 А;
- максимальний точковий струм – 8.7 А;
- товщина скла – 3.2мм.

Hanplast Solar SW Premium Plus - це сонячна панель високої якості, яка пропонує ефективне використання сонячної енергії для виробництва електрики. Вона володіє превосходними технічними характеристиками і надійними компонентами, що роблять її ідеальним вибором для встановлення на даху будинку або інших приміщень.

Сонячна панель Hanplast Solar SW Premium Plus виготовлена з високоякісних монокристалічних сонячних клітин, які забезпечують високу ефективність перетворення сонячної енергії у електрику. Її дизайн забезпечує максимальний збір сонячного світла, що дозволяє отримати більше енергії навіть за умов обмеженої площі.

SW Premium Plus має потужність 275 ватт і високий коефіцієнт перетворення, що дозволяє отримувати більше електроенергії зі зібраного сонячного випромінювання. Вона також володіє високою стійкістю до механічних навантажень і впливу неблагоприятних погодних умов, що забезпечує тривалу роботу панелі протягом довгого часу.

Панель Hanplast Solar SW Premium Plus має компактну конструкцію і легку вагу, що спрощує її транспортування та встановлення. Вона має також високий рівень безпеки, включаючи захист від перенапруги, короткого замикання і надмірного навантаження, що робить її безпечним рішенням для використання в домашніх умовах.

Завдяки своїм технічним можливостям та надійності, Hanplast Solar SW Premium Plus є ідеальним вибором для виробництва чистої електроенергії в будь-яких умовах. Вона може бути використана як в домашніх умовах, так і в

комерційних або промислових об'єктах, допомагаючи знизити витрати на електрику і покращити енергетичну ефективність.

Отже, Hanplast Solar SW Premium Plus - це надійне, ефективне та стильне рішення для виробництва електрики з сонячної енергії.

2.2.3 Апаратний монітор EPSolar MT50

Також для моніторингу використовується апаратний модуль-монітор EPSolar MT50. На рисунку 2.8 зображений Epsolar MT50 – це дисплей, що використовується для керування та моніторингу сонячних систем з використанням контролерів зарядки Epsolar. Цей пристрій забезпечує зручний та інтуїтивно зрозумілий спосіб керування та налаштування сонячної системи з будь-якого місця.

Рисунок 2.8 – Epsolar MT50 мониторинговий дисплей

Серед позитивних рис Epsolar MT50 варто відзначити його легкість у використанні. Завдяки зручному інтерфейсу користувач може легко налаштувати систему зарядки та переглянути інформацію про поточний стан системи. Крім того, дисплей має світлодіодне підсвічування, що дозволяє користуватися ним в будь-який час доби.

Однак, серед негативних рис Epsolar MT50 можна відзначити його обмежену функціональність. Хоча дисплей забезпечує базовий моніторинг та керування сонячною системою, він не має додаткових функцій, які можуть бути корисними для деяких користувачів. Крім того, дисплей працює лише з контролерами зарядки виробництва Epsolar, що може бути обмеженням для користувачів, які використовують інші контролери.

Узагалі, Epsolar MT50 є корисним та зручним дисплеєм для керування та моніторингу сонячної системи. Його легкість використання та зручний інтерфейс дозволяють швидко налаштувати систему та отримувати необхідну інформацію про її стан. Однак, варто враховувати його обмежену функціональність та обмеження у використанні з контролерами зарядки виробництва Epsolar.

Загальна схема підключення лабораторного стенду сонячної панелі зображена на рисунку 2.9.

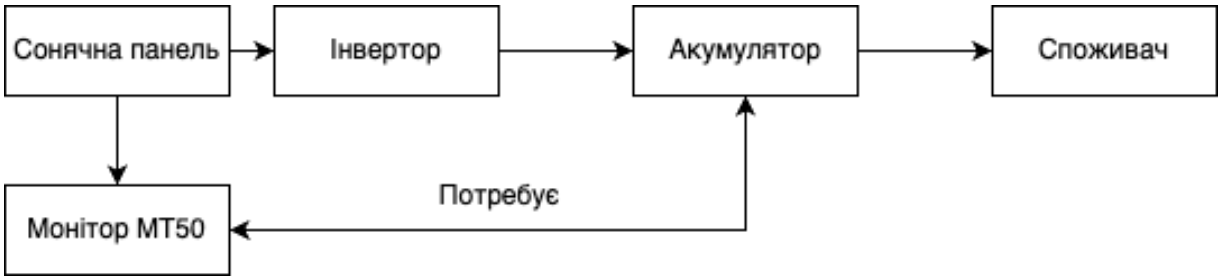

Рисунок 2.9 – Схема підключення лабораторного стенду сонячної панелі

Так як споживач підключається до акумулятору, то споживач повинен відповідати по напрузі до акумулятору. У нашому випадку – 12 вольт. Якщо є потреба у підключенні інших елементів, між акумулятором та споживачем треба підключати перетворювач до цільової напруги.

Висновки до розділу 2

У другому розділі було розглянуто провідні аналоги на ринку, проведено аналіз та описано їх переваги і недоліки. Було описано предметну область, тему

дипломної роботи, розглянуто розвиток даної теми в рамках ринку. Роглянуто проблематику та потребу в інструментах візуалізації даних отриманих з подібних установок.

Описано основні принципи роботи сонячних панелей, розглянуто лабораторний стенд сонячної панелі та наведено інформації про його складові та їх характеристики.

3 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі буде наведено та описано використані технології під час розробки даного програмного продукту, а також їх ключові переваги, через які вибір на користь використання було віддано їм.

3.1 Мова програмування Python

Python — це мова програмування з високою інтерпретацією, яка стала однією з найпопулярніших мов для програмістів у різних галузях.

Однією з головних характеристик мови Python є її простота. Синтаксис Python дуже простий для розуміння, що робить його ідеальним для початківців. Крім того, Python має багато стандартних бібліотек, які можуть виконувати багато завдань одночасно.

Python підтримує багато парадигм програмування, включаючи процедурне, об’єктно-орієнтоване та функціональне програмування. Це дозволяє програмістам використовувати найкращий метод для власних проектів [15].

Завдяки великій кількості сторонніх бібліотек, таких як NumPy, Pandas, SciPy тощо, Python також підтримує автоматизацію таких завдань, як збір даних з Інтернету, обробка даних і наукові обчислення.

Крім того, Python є кросплатформною мовою програмування, яка дозволяє розробникам написати програму один раз і запустити її на будь-якій платформі. Загалом Python є потужною, але простою мовою програмування, яка дозволяє розробникам швидко писати якісний код і вирішувати широкий спектр завдань [13].

3.2 Фреймоворк Flask

Flask є одним із найпопулярніших мікрофреймворків веб-розробки на мові програмування Python. Flask дуже простий у використанні та вимагає мінімальної конфігурації сервера. Це дозволяє розробникам швидко створювати веб-додатки з мінімальними зусиллями та часом [15].

Ось деякі з переваг Flask.

Легкий і простий у використанні. Flask — це легкий мікрофреймворк з мінімальним набором функцій. Він має чистий і стислий синтаксис, який дозволяє розробникам швидко створювати веб-додатки.

Гнучкий і масштабований. Flask дозволяє розробникам легко розширювати його функціональність за допомогою різноманітних розширень і плагінів. Це робить Flask дуже гнучким і придатним для будь-якого типу веб-додатків.

Інтеграція з іншими інструментами. Flask підтримує різні інструменти, такі як SQLAlchemy, WTForms, Jinja2 тощо, що дозволяє розробникам створювати програми з потужними та надійними функціями.

Широкий діапазон можливостей розгортання. Flask можна легко розгорнути на різних платформах, таких як Heroku, Google App Engine, Amazon Web Services тощо.

Відкрите джерело. Flask є відкритим кодом, що дозволяє розробникам змінювати код і допомагати розвивати проект.

Загалом, Flask є чудовим вибором для розробки веб-додатків з мінімальними зусиллями та максимальною ефективністю [3].

3.3 Бібліотека Pandas

Pandas — бібліотека для роботи з табличними даними на мові програмування Python. З його допомогою можна легко і швидко читати, записувати і обробляти дані в різних форматах, включаючи CSV, Excel, SQL, JSON, HTML і т.д.

Нижче описані основні переваги Pandas.

Простий у використанні. Pandas надає простий і зрозумілий API для роботи з табличними даними. Це дозволяє користувачам легко створювати, трансформувати, зберігати, фільтрувати, групувати, комбінувати та агрегувати дані [15].

Швидкість дії. Pandas забезпечує швидку та ефективну обробку даних. Його інтеграція з бібліотеками NumPy і SciPy дозволяє використовувати векторизовані операції, які можуть значно прискорити обробку даних.

Гнучкість. Pandas надає різноманітні функції обробки даних, які дозволяють легко налаштувати обробку даних відповідно до потреб. Користувачі можуть працювати з даними за допомогою простих функцій і методів, а також більш складних функцій, таких як об’єднання та злиття таблиць.

Широкий спектр функцій. Pandas надає багато функцій для аналізу та обробки даних, включаючи статистичний аналіз, візуалізацію даних, обробку відсутніх значень тощо. [10].

Коротше кажучи, Pandas — це потужний і універсальний інструмент на мові програмування Python для роботи з табличними даними. За допомогою Pandas користувачі можуть забезпечити ефективну та точну обробку даних відповідно до своїх потреб.

3.4 Платформа Node.js

Node.js – це потужна та інноваційна платформа, яка змінює ландшафт розробки веб-додатків. Його унікальність полягає в тому, що він поєднує в собі два основних компоненти: серверну частину JavaScript та потужний двигун V8 від Google. Ця комбінація дає розробникам незрівнянну швидкодію та ефективність.

Node.js забезпечує змогу виконувати JavaScript на сервері, що раніше було неможливо. Це означає, що розробники можуть використовувати одну мову

програмування як для інтерфейсу, так і для серверу. Це спрощує розробку, обслуговування та масштабування веб-додатків.

Одним із найважливіших аспектів Node.js є його необхідність циклу подій. Замість традиційного синхронного підходу Node.js пропонує модель асинхронного програмування, що дозволяє обробляти багато запитів одночасно без блокування виконання. Це інноваційний підхід, який дозволяє створювати високопродуктивні додатки зі збільшеною масштабованістю.

Node.js також має велику кількість модулів та пакетів, доступних через менеджер пакетів npm. Ця величезна екосистема розширює можливості Node.js, дозволяючи розробникам використовувати готові рішення та зручні інструменти для будь-яких завдань [9].

Багатопотоковість також є однією з головних переваг Node.js. Використовуючи модульну архітектуру та пули потоків, розробники можуть ефективно використовувати ресурси сервера та обробляти багато запитів одночасно. Це особливо корисно для додатків обміну інформації у реальному часі, таких як чати, стрімінгові сервіси або системи миттєвих оновлень.

Ще однією особливістю Node.js є його легковаговість та швидкодія. Завдяки оптимізації виконання JavaScript-коду за допомогою двигуна V8, Node.js забезпечує вражаючу продуктивність, яка вирізняє його серед інших серверних технологій.

Node.js активно підтримується та оновлюється великою спільнотою розробників. Це означає, що розробники можуть легко знаходити допомогу, документацію та рішення для своїх проектів. Спільнота Node.js постійно працює над вдосконаленням та розширенням функціональності цієї платформи, що робить її ще більш привабливою для розробників.

Завдяки всім цим особливостям, Node.js став популярним вибором для розробки сучасних веб-додатків. Він відкриває безліч можливостей для швидкого, ефективного та масштабованого програмування на сервері. Незважаючи на

складність веб-сайту або додатку, Node.js допомагає забезпечити високу продуктивність та надійність програми [4].

3.5 Бібліотека Chart.js

Chart.js — це бібліотека JavaScript для відображення даних у вигляді діаграм і таблиць. Це дозволяє легко створювати динамічні та інтерактивні діаграми на веб-сторінках, надаючи користувачам простіший і зрозуміліший спосіб відображення даних [1].

Нижче наведено декілька переваг Chart.js.

Простота використання. Chart.js має простий і зрозумілий API, який дозволяє швидко створювати діаграми з мінімальними зусиллями. Він також має велику кількість документації та підтримується великою спільнотою розробників.

Інтерактивність. Chart.js дозволяє додавати різноманітні функції взаємодії з графіками, такі як масштабування, перетягування, приближення та інші. Це дозволяє користувачам докладніше аналізувати дані.

Можливість візуалізації різноманітних типів даних. Chart.js підтримує відображення різноманітних типів даних, таких як лінійні графіки, стовпчасті графіки, кругові діаграми, гістограми та інші. Це дозволяє відображати дані в максимально зрозумілому для користувача форматі.

Адаптивність. Chart.js має вбудовану функцію для адаптації діаграм до різних розмірів екрану, що забезпечує зручний і читабельний перегляд на будь-якому пристрої.

Налаштування. Chart.js дозволяє налаштовувати вигляд графіків, включаючи кольори, шрифти та інші елементи, що дозволяє створювати графіки, які відповідають індивідуальним потребам та вимогам користувачів.

Підтримка різних платформ. Chart.js підтримується на різних платформах, включаючи комп’ютерні та мобільні, що дозволяє використовувати його в різноманітних проектах.

Загалом, Chart.js – це потужна та проста у використанні бібліотека, яка дозволяє швидко та ефективно візуалізувати дані на веб-сторінках. Вона має багато переваг, що роблять її популярним вибором для візуалізації даних на різноманітних проектах [4].

3.6 Текстовий формат даних JSON

JSON (JavaScript Object Notation) — це легкий формат обміну даними загального призначення, який заслужено став фаворитом розробників у всьому світі. Завдяки своїй простоті та зручності читання JSON став незамінним інструментом для передачі даних між різними системами та платформами.

JSON використовується в багатьох галузях, від веб-розробки та мобільних додатків до обробки даних у великих корпоративних системах. Його популярність полягає в його простоті та гнучкості. JSON представляє дані у вигляді пар ключ-значення, що дозволяє зберігати структуровану інформацію в зручному для розуміння форматі [16].

Однією з найважливіших властивостей JSON є те, що він не залежить від мов програмування. Це означає, що дані JSON можна легко створювати та обробляти за допомогою будь-якої мови програмування, яка підтримує використання рядків та об’єктів.

JSON також надає можливість вкладати дані одне в одне, створюючи складні структури. Надає можливість зберігати списки значень і об’єктів за допомогою масивів для групування пов’язаних даних. Це забезпечує велику свободу в організації та моделюванні даних.

Крім того, JSON підтримує різні типи даних, включаючи рядки, числа, логічні значення, масиви та об'єкти. Це дозволяє чітко відображати різні типи інформації та легко з ними взаємодіяти.

JSON надає можливість безпосереднього кодування та декодування даних у різних форматах, таких як рядки, масиви байтів або об’єкти. Це робить його

ідеальним інструментом для передачі даних через мережу або зберігання даних у файловій системі.

Ще одна перевага JSON полягає в тому, що його можна легко використовувати. Багато мов програмування забезпечують вбудовані функції кодування та декодування JSON, які полегшують роботу з даними у форматі JSON. Крім того, багато інструментів і служб підтримують використання JSON, що дозволяє легко й ефективно виконувати такі операції, як перетворення, перевірка та обробка даних JSON [4].

Також важливим компонентом є гнучкість JSON щодо розширень і додаткових функцій. JSON підтримує розширення з додатковими полями або вбудованими об’єктами, що дозволяє додавати власні властивості та функції до документів JSON.

Узагальнюючи, JSON — це потужний і універсальний формат обміну даними, який спрощує передачу, зберігання та обробку інформації. Він забезпечує простоту, гнучкість та незалежність від мови, роблячи його незамінним інструментом для розробників у всіх галузях програмування.

3.7 Протокол передачі даних HTTP

Зв’язок між клієнтськими програмами та серверами в Інтернеті використовує протокол передачі гіпертексту (HTTP), один із основних протоколів для обміну даними між вузлами у великій мережі.

HTTP визначає правила та формати, які керують взаємодією між клієнтами та серверами. Протокол підтримує передачу гіпертекстових ресурсів, таких як веб-сторінки, зображення, відео та інші мультимедійні елементи, від одного вузла до іншого.

HTTP базується на моделі клієнт-сервер, де клієнтська програма (зазвичай веб-браузер) надсилає запит серверу, що містить відповідний ресурс. Запити складаються з повідомлень HTTP, які відповідають методам (таким як GET, POST,

PUT, DELETE), шляхам до ресурсів та іншим метаданим, які повідомляють серверу, що робити [14].

При отриманні запиту сервер обробляє його і генерує відповідь у вигляді HTTP-статусу, що вказує на результат операції, і іншого тіла відповіді, що містить передані дані. Відповідь повертається клієнту, який може обробити та відобразити отриману інформацію.

HTTP використовує фундаментальні принципи, такі як відсутність стану та постійне з'єднання, щоб забезпечити ефективний і масштабований зв'язок між клієнтами та серверами. Протокол також підтримує використання шифрування HTTPS для забезпечення безпеки передачі даних [6].

HTTP став основою для розвитку сучасного Інтернету та розширення взаємодії між користувачами та веб-сервісами. Це дозволяє нам отримувати доступ до багатьох ресурсів, спілкуватися, обмінюватися даними та насолоджуватися багатим вмістом, який пропонує Інтернет.

3.8 Система збірки Parcel

Широко прийнята в багатьох галузях, система збірки Parcel є потужним інструментом, призначеним для ефективного керування залежностями та пакетування ресурсів у веб-розробці. Система забезпечує зручний і надійний спосіб створення та доставки веб-додатків, забезпечуючи високу продуктивність і швидку розробку.

Система побудови Parcel визначається як програмне забезпечення, розроблене з використанням сучасних технологій і принципів, що дозволяє автоматизувати процес пакування та створення файлів проекту. Завдяки інтуїтивно зрозумілому інтерфейсу, який легко настроюється, розробники можуть ефективно керувати залежностями та збирати ресурси оптимізованим способом для доставки веб-додатків [11].

Однією з ключових переваг системи Parcel є її здатність автоматично визначати та вирішувати залежності між файлами, спрощуючи процес розробки та забезпечуючи стабільність інтеграції. Також система має високу продуктивність за рахунок алгоритму паралельної компіляції та розподілених обчислень. Це дозволяє пришвидшити час налаштування проекту та забезпечити швидкий час виконання під час розробки.

Ще однією перевагою системи складання Parcel є її гнучкість і масштабованість. Розробники можуть використовувати різні плагіни та налаштування, щоб налаштувати систему відповідно до своїх потреб. Це дозволяє налаштувати процес складання та упаковки веб-додатків відповідно до конкретних вимог вашого проекту, забезпечуючи високу якість і продуктивність.

Ототожнюючи, система складання Parcel є інноваційним і надійним інструментом у світі веб-розробки. Його здатність автоматизувати процес пакування, керувати залежностями та швидкістю створення робить його обов'язковим інструментом для розробників, які прагнуть досягти високої продуктивності та ефективності своїх проектів [15].

3.9 Платформа Apache Kafka

Apache Kafka — потужна та масштабована платформа розподіленої обробки повідомлень. Він призначений для обробки великих потоків даних і підтримує високу продуктивність і масштабованість [5].

Нижче наведено деякі переваги Apache Kafka.

Висока продуктивність. Apache Kafka може обробляти великі обсяги даних і забезпечувати високу продуктивність. Це робить його ідеальним для обробки потоків даних у реальному часі.

Масштабованість. Apache Kafka легко масштабується, що дозволяє збільшити його потужність і продуктивність шляхом додавання нових вузлів до кластера.

Надійність. Apache Kafka забезпечує надійний обмін повідомленнями, зменшує втрати даних і покращує стабільність системи.

Гнучкість. Apache Kafka можна використовувати для різних випадків використання, включаючи потокове передавання даних у реальному часі, зберігання журналів і обробку подій.

Відкритий код. Apache Kafka має відкритий код і підтримується великою спільнотою розробників, що дозволяє швидко розвивати та вдосконалювати платформу.

Загалом Apache Kafka — це потужна та надійна структура обробки потоків даних. Він має багато переваг і може використовуватися в різних випадках.

3.10 Меседжинг, як спосіб передачі даних між застосунками

У сучасному світі меседжинг між різними програмами став невід'ємною частиною розробки програмного забезпечення. У цьому сенсі меседжинг – передача повідомлень як ефективний спосіб обміну повідомленнями між різними компонентами забезпечує безперебійне спілкування.

Візуалізуючи цю концепцію в програмуванні, меседжинг можна порівняти з брокером, який забезпечує обмін повідомленнями між різними компонентами програмної системи. Цей посередник допомагає кожному компоненту безпосередньо взаємодіяти з іншими компонентами, створюючи віртуальний канал зв'язку [6].

Використовуючи меседжинг, розробники можуть встановлювати зв'язки між різними службами, дозволяючи їм передавати повідомлення, ініціювати події та відповідати на них. Такий підхід забезпечує високу модульність і незалежність компонентів, оскільки кожен з них може взаємодіяти через стандартизовані інтерфейси, тобто обмінюватися повідомленнями.

Базуючись на загальноприйнятих принципах асинхронності, меседжинг дозволяє надсилати й отримувати повідомлення в будь-який час, не чекаючи

негайної відповіді. Це забезпечує гнучкість і швидкість взаємодії компонентів, дозволяючи їм виконувати завдання незалежно, в той час як обмін даними відбувається у фоновому режимі.

Багатопотоковість і масштабованість є іншими перевагами меседжингу. Завдяки паралельній обробці повідомлень компоненти можуть працювати одночасно, виконуючи різні завдання, не блокуючи один одного. Крім того, важливою властивістю передачі повідомлень є її здатність легко розширюватися, дозволяючи включати нові компоненти у систему без серйозних змін у існуючій архітектурі.

Тому меседжинг є ключовим механізмом у програмуванні, головним завданням якого є забезпечення ефективного зв’язку та передачі даних між різними компонентами програмного забезпечення. Використання меседжингу дозволяє створювати віртуальні канали зв’язку, щоб забезпечити модульність, незалежність і гнучкість компонентів. Загалом меседжинг відкриває нові можливості для створення складних програмних систем, забезпечуючи їх ефективну та гнучку взаємодію.

3.11 Реляційна PostgreSQL

PostgreSQL – це потужна реляційна система керування базами даних (RDBMS), яка пропонує багато функцій та можливостей, що забезпечують високу продуктивність та надійність баз даних.

Нижче наведено декілька переваг PostgreSQL.

Висока надійність. PostgreSQL забезпечує високий рівень надійності та стійкості баз даних завдяки механізмам відновлення даних та підтримці транзакцій. Це дозволяє зберігати цінні дані без ризику втрати.

Розширюваність. PostgreSQL має широкий спектр можливостей для розширення та настройки баз даних. Вона підтримує різноманітні розширення, такі

як модулі, які додають підтримку для нових типів даних та інші додаткові функції, що дозволяють адаптувати базу даних до індивідуальних потреб користувачів.

Підтримка SQL. PostgreSQL підтримує мову SQL з розширеними можливостями, що дозволяє здійснювати складні запити та аналізувати великі обсяги даних. Вона також має підтримку NoSQL даних та гнучкий JSON тип даних.

Відкритий код та безкоштовність. PostgreSQL є відкритим та безкоштовним програмним забезпеченням, що дозволяє користувачам вільно користуватися та розповсюджувати його безкоштовно. Крім того, вона має велику спільноту розробників, яка допомагає у підтримці та розвитку бази даних.

Підтримка реплікації. PostgreSQL має вбудовану підтримку реплікації, що дозволяє створювати декілька копій баз даних для резервного копіювання та збільшення продуктивності. Це особливо корисно для додатків з високою доступністю та вимогами до продуктивності.

Безпека. PostgreSQL має вбудовану систему автентифікації та авторизації, що забезпечує високий рівень безпеки баз даних. Вона підтримує різні рівні доступу до даних та можливість налаштування шифрування даних для додаткової захисту.

Підтримка GIS-даних. PostgreSQL має вбудовану підтримку геопросторових даних (GIS), що дозволяє зберігати та аналізувати дані про географічне положення, наприклад, дані про місцезнаходження та картографічні дані.

Узагальнюючи, PostgreSQL є потужною та надійною системою керування базами даних з багатьма вбудованими функціями та можливостями, що забезпечують високу продуктивність та безпеку. Вона також є відкритою та безкоштовною, що робить її привабливим вибором для багатьох користувачів та розробників програмного забезпечення [12].

3.12 Платформа Docker

Docker Desktop — це комп’ютерна програма для розробки та розгортання програм у контейнерах Docker на локальному комп’ютері. Вона доступна для

операційних систем Windows і macOS і дозволяє створити ізольоване середовище для програм, що працюють у контейнерах, не впливаючи на роботу інших програм на комп’ютері [2].

Ось деякі з переваг Docker Desktop.

Простота використання. Docker Desktop має простий та інтуїтивно зрозумілий інтерфейс користувача, який дозволяє швидко налаштувати та запустити контейнери Docker. Крім того, Docker Desktop містить інструменти для моніторингу та керування контейнерами.

Ізоляція. Docker Desktop забезпечує ізольоване середовище для програм, що працюють у контейнерах, що запобігає конфліктам між різними програмами та забезпечує безпеку та стабільність системи.

Портативність. Контейнери Docker переносяться між середовищами, що полегшує переміщення програм між розробкою, тестуванням і виробництвом. Docker Desktop підтримує створення контейнерів, які можна легко перенести на інші системи, що прискорює процес розгортання програми.

Масштабованість. Docker Desktop дозволяє масштабувати програми, додаючи або видаляючи контейнери, забезпечуючи гнучкість і ефективність використання ресурсів.

Економія ресурсів. Docker Desktop дозволяє використовувати менше системних ресурсів, оскільки контейнери забезпечують ізоляцію програм і використовують спільні системні бібліотеки, заощаджуючи багато обчислювальних ресурсів.

Підтримка. Docker Desktop має широку підтримку спільноти та розширену документацію, тож надається можливість швидко знайти відповіді на будь-які запитання щодо використання Docker. Крім того, Docker Desktop пропонує інтеграцію з такими популярними інструментами розробки та розгортання програм, як Git, Jenkins, Kubernetes тощо.

Зменшення ризиків помилок. Docker Desktop дозволяє створити однакове середовище для розробки, тестування та виробництва додатків. Це зменшує ризик помилок, які можуть виникнути через різні середовища розробки.

Підтримка крос-платформи. Docker Desktop дозволяє розгортати програми на різних операційних системах і архітектурах, що дозволяє забезпечити сумісність і масштабованість рішень.

Загалом Docker Desktop — це потужний інструмент розробки та розгортання додатків, який забезпечує швидкий і безпечний процес розгортання, ефективне використання ресурсів і гнучку конфігурацію середовища [7].

Висновки до розділу 3

У третьому розділі було розглянуто інструменти, що були використані під час розробки програмного продукту, а саме: основну мову програмування, що було обрано для написання даної системи, описано її переваги у вирішенні даного типу задач. Проаналізовано та описано вибір бібліотек для обробки даних, фреймворку для побудови веб серверу з огляду на його переваги та простоту роботи і підтримки. Розглянуто та обґрунтовано використання платформи для обміну повідомленнями між частинами програмного продукту. Описано та розглянуто платформу для контейнеризації Docker desktop, її переваги та потреба у використанні при написанні програмного продукту. Обґрунтовано вибір бази даних, пояснено її переваги та потреба для вирішення задачі зі збереження оброблених даних. Описано основні бібліотеки для візуалізації інформації на клієнтській частині для мови програмування JavaScript.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

У цьому розділі буде описана програмна реалізація, яка передбачає отримання даних з сонячної панелі, їх обробку та візуалізацію на клієнті.

Розділ міститиме опис алгоритмів, використаних для отримання даних з сонячної панелі, їх обробки та зберігання на платформі Apache Kafka, а також аналіз та візуалізацію даних на клієнті. Крім того, будуть розглянуті деталі програмної реалізації, такі як структура проекту, використані паттерни проектування, методи тестування та інші аспекти, що впливають на ефективність та надійність роботи програмного продукту.

4.1 Архітектура системи

Архітектурою системи обрано мікросервісний підхід, через його переваги у написанні відказостійких та гнучких систем.

Мікросервісний підхід є досить популярним у світі програмної інженерії та розробки програмного забезпечення. Його ідея полягає у тому, щоб розділити монолітний додаток на більш дрібні компоненти - мікросервіси, кожен з яких відповідає за свою частину функціональності. Кожен мікросервіс може бути розроблений та розгорнутий незалежно від інших, що дозволяє забезпечити більш гнучку та швидку розробку, розгортання та масштабування додатку [8].

На рисунку 4.1 зображено діаграму розгортання системи.

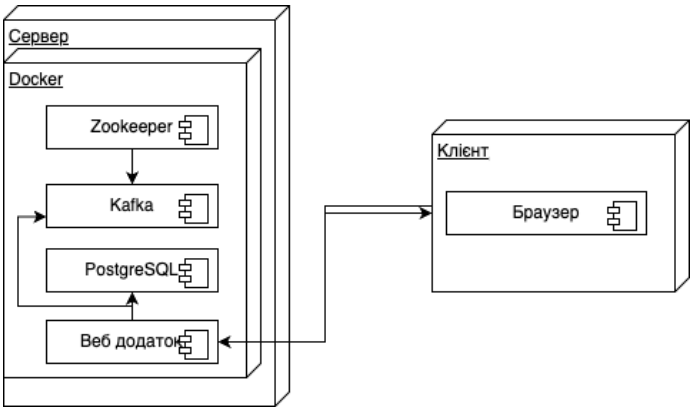

Рисунок 4.1 – Діаграма розгортання

У випадку з описаним проектом, мікросервісний підхід був обраний через його складність та розмір. Обраний підхід дозволяє розділити проект на менші компоненти, кожен з яких відповідає за свою частину функціональності. Так, компонент, який отримує дані з сонячної панелі, може бути розроблений та розгорнутий незалежно від компонента, який відповідає за їх обробку та збереження на платформі Apache Kafka. Крім того, мікросервісний підхід дозволяє забезпечити більш гнучку та швидку розгортання та масштабування окремих компонентів системи.

На рисунку 4.2 представлена діаграма прецедентів системи.

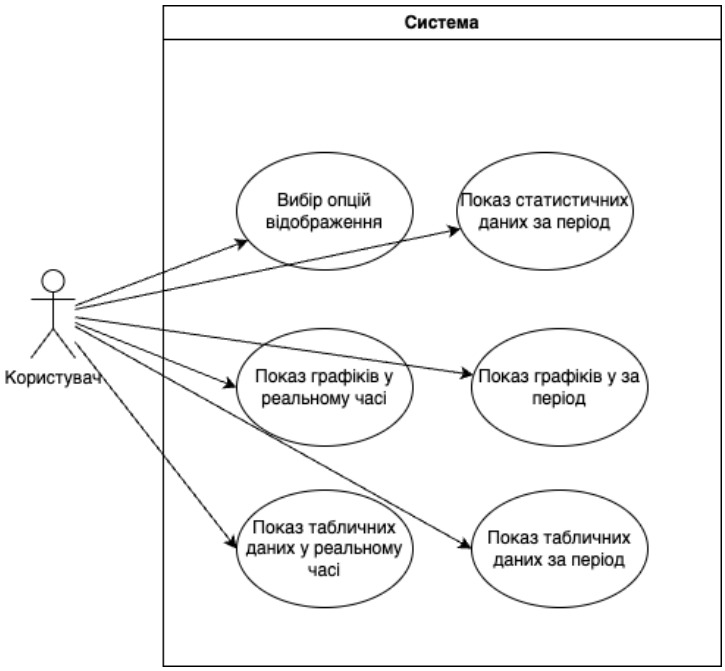

Рисунок 4.2 – Діаграма прецедентів

Діаграма прецедентів описує можливі варіанти взаємодії користувача із програмним продуктом.

4.2 Обробка даних у реальному часі

Отримання даних з сонячної панелі здійснюється через сервер, що оброблює ці дані та надсилає їх на Apache Kafka. Оброблені дані зберігаються у топіку Kafka та очікують, що їх забере звідти другий сервер.

Якщо другий сервер запущено, він одразу же буде збирати ці дані до своєї пам’яті та зберігатиме, доки клієнт не відкриє сторінку та не запустить систему відображення даних. Цей відхід дозволяє працювати компонентам асинхронно, і якщо один з них вийде з ладу та перезапуститься, вся система продовжить працювати у штатному режимі.

Другий сервер, що збирає та підготовує дані, віддає їх у форматі JSON, щоб клієнтська частина могла з ними взаємодіяти досить просто. Клієнт, за допомогою коду із використанням бібліотеки ChartJS візуалізує у вигляді графіків дані, що надійдуть.

4.3 Обробка даних за попередній проміжок часу

Обробка даних за проміжок часу дуже схожа на обробку в реальному часі, тільки першопочаткові дані, що збираються, зберігаються у базу даних. У нашому випадку в PostgreSQL, і другий сервер має також доступ до цих даних.

Користувачеві лише треба зробити запит на сторінці з параметрами про дату початку та дату кінця, звідки треба брати дані, сервер обробить цей запит, та надасть дані за період, а клієнт, використовуючи ту ж саму частину із Chart JS побудує та відобразить вже частину із даними за попередній період часу.

4.4 Обробка даних під’єднаних пристроїв

До лабораторного стенду який складається з сонячної панелі, акумулятору та дисплею-монітору можна під’єднувати пристрої, як до звичайного електричного кола та живити їх з енергії, яку виробила сонячна панель.

Сонячна панель підключена до акумулятора, який має вихідну напругу 12 вольт, тобто це означає що споживачі, без модифікації кола, мають теж споживати 12 вольт.

Користувач може на сторінці моніторингу даних у реальному часі вмикати пристрої за замовчуванням, такі як: акумулятор, вентилятор та лампу, і дивитись, як ці пристрої впливатимуть на показання струму з лабораторного стенду.

Обробка даних винесена на клієнтську частину з огляду на мікросервісну архітектуру. Серверна частина відповідає за сприймання та обробку даних про вироблення сонячної енергії з панелі, зберігає інформацію та видає її користувачеві, у випадку з історичними даними.

Тому обробка показників споживання під'єднаних пристроїв відбувається в реальному часі з отриманих даних серверної частини. Це також дозволяє розвантажити сервер від таких обчислень та оброблені дані використовувати напряму у графічній та табличній інформації клієнту, не звертаючись до шляхів передачі даних між клієнтом та сервером. На рисунку 4.3 зображено UML діаграма класу пристрою.

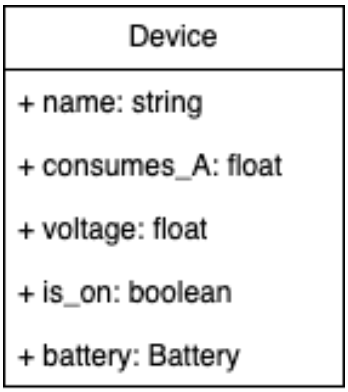

Рисунок 4.3 – UML діаграма класу пристрою

Всі пристрої, що створюють через файл конфігурації, або ж напраму з клієнтської частини повинні відповідати наступній діаграмі та мати назву пристрою, силу струму у амперах, напругу у вольтах.

Далі частина обробки цих пристроїв проставить посилання на об'єкт акумулятору, щоб код, який прораховує споживання міг взаємодіяти із показниками батареї, на випадок, якщо потужності сонячної панелі не вистачить для живлення якоїсь частини пристроїв.

Алгоритм обробки також передбачає врахування в обробку споживання тільки тих пристроїв, що увімкнені на даний момент часу на клієнтській частини. Для цього є прапорець у вигляді поля типу Boolean у кожного пристрою – is_on.

При додаванні пристрою через клієнтську частину – веб інтерфейс, отримані дані з модальної форми передаються у метод обробник, що створить об’єкт типу Device, заповнить його та додасть до загального словнику об’єктів, що зараз можуть враховуватись при обчисленні. Також під час цієї дії, на веб сторінку додасться блок із даними про пристрій та перемикач, що відповідатиме за зміну прапорцю is_on, зміну стану цього поля об’єкту відслідковує EventListener, що дає зможу зреагувати на будь-які зміни цього об’єкту.

Оновлення інформації відбувається періодично, тобто кожен раз, коли сервер надішле інформацію про вироблення енергії сонячної панелі на kafka, а клієнт в свою чергу зчитає та почне оброблювати ці дані.

Так можна вносити всі потрібні змінити, а як тільки свіжа інформація буде доступна клієнтові, він проведе необхідні обчислення та відобразить поточний стан виробленої та спожитої енергії у режимі реального часу.

Висновки до розділу 4

У четвертому розділі було розглянуто питання архітектури системи, вирішено завдання з вибору архітектури для всієї системи, описано взаємодію окремих компонентів системи, створено та описано діаграму прецедентів, описано діаграму розгортання та загального вигляду системи на фізичному та програмному рівні.

Також у підрозділах описано три ключових процеси у системі, а саме – отримання та обробка даних у реальному часі із подальшою візуалізацією, отримання та візуалізація історичних даних і взаємодія цих даних у реальному часі із пристроями, що під’єднано до системи.

5 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

В даному розділі дипломної роботи буде розглянуто питання, пов'язані зі встановленням та використанням програмної системи з боку користувача. Розглянуті будуть системні вимоги, необхідні для встановлення та коректної роботи системи, а також процес інсталяції та конфігурації.

Окрім цього, буде описано процес взаємодії користувача з веб-додатком програмної системи. Розглянуто будуть основні функції та можливості веб-інтерфейсу, процес авторизації та навігації по сторінках, а також введення та редагування даних.

5.1 Підготовча робота та інсталяції

Для початку роботи із додатком, потрібно встановити Docker Desktop чи Docker клієнт на цільову систему.

Після встановлення потрібно відкрити проект через термінал.

Наступним кроком, потрібно запустити наступну команду – docker compose build для початку процесу збірки контейнерів та завантаження потрібних залежностей.

Після того, як процес побудови буде закінчено, потрібно ввести docker compose up

Скрипти описані у файлі docker-compose.yml містять усі необхідні інструкції для роботи системи.

Останнім кроком потрібно зайти на http://localhost:1234 та система відкриється.

5.2. Взаємодія з веб-додатком

Після встановлення веб-додатку необхідно запустити його і перейти за посиланням. При переході за посиланням відкриється сторінка (рис. 5.1) користувач може обрати одну з двох опцій: вивід даних у реальному часі (Real-time data) або ж за якийсь період часу (Historical data).

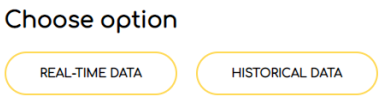

Рисунок 5.1 – Перша сторінка веб-додатку

При виборі опції Historical data з’являється сторінка з порожніми графіками (рис. 5.2), для запуску виводу даних необхідно обрати проміжок часу та натиснути Submit. Для очищення графіків треба натиснути на кнопку Clear chart.

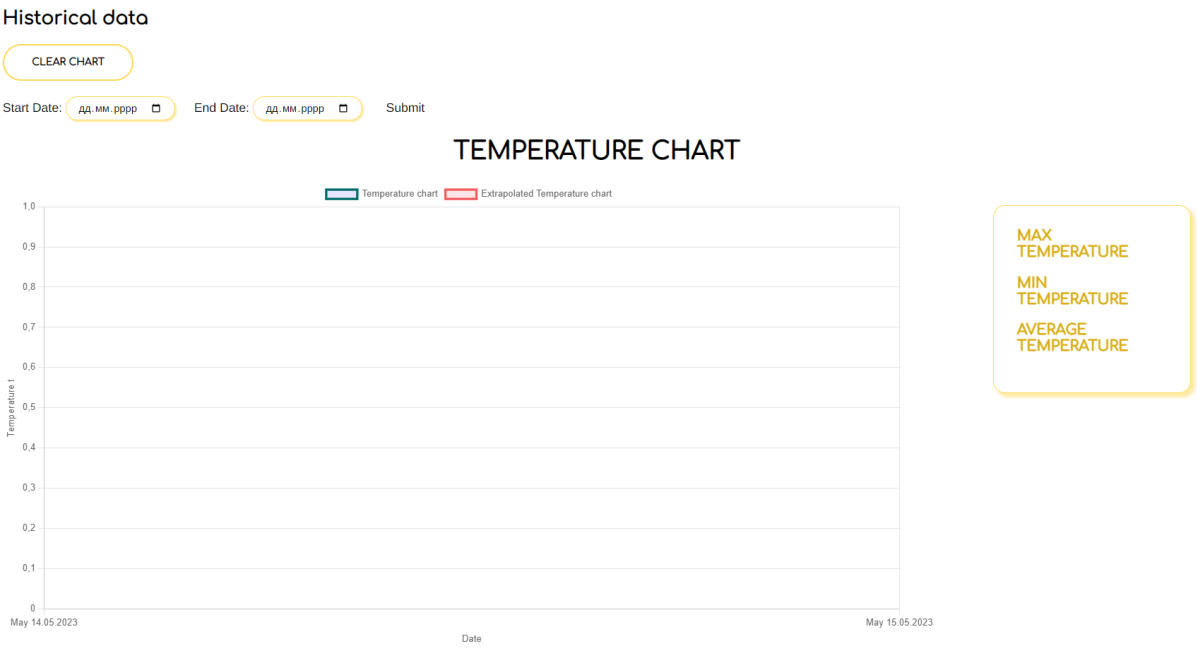

Рисунок 5.2 – Сторінка виводу даних за проміжок часу

Внизу сторінки знаходиться таблиця з даними (рис. 5.3), яку можна показати натиснувши на кнопку Show table та приховати натиснувши на кнопку Hide table.

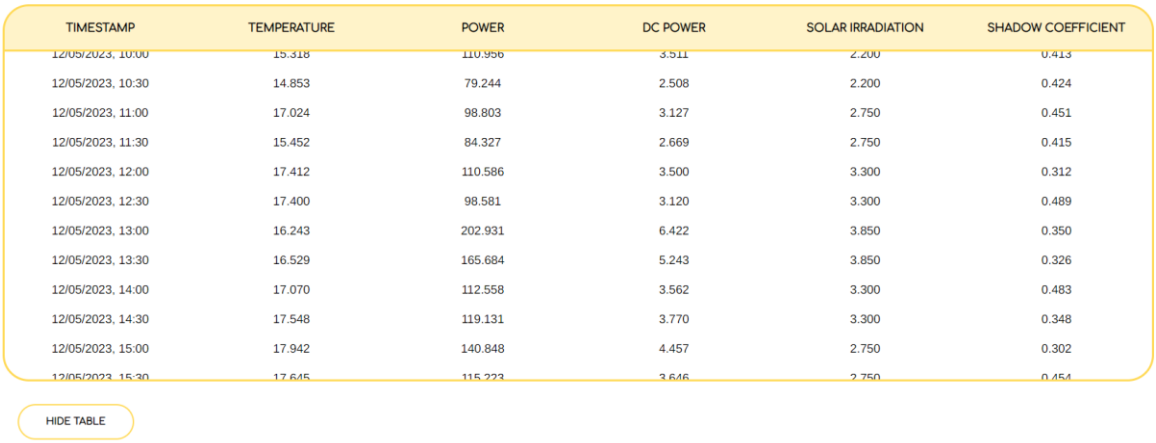

Рисунок 5.3 – Таблиця даних

При виборі опції Real-time data з’являється сторінка з порожніми графіками (рис. 5.4), для запуску виводу даних необхідно натиснути на кнопку Start/Stop. Для очистки графіків треба натиснути кнопку Clear chart. Для зупинки виводу даних необхідно знов натиснути на кнопку Start/Stop.

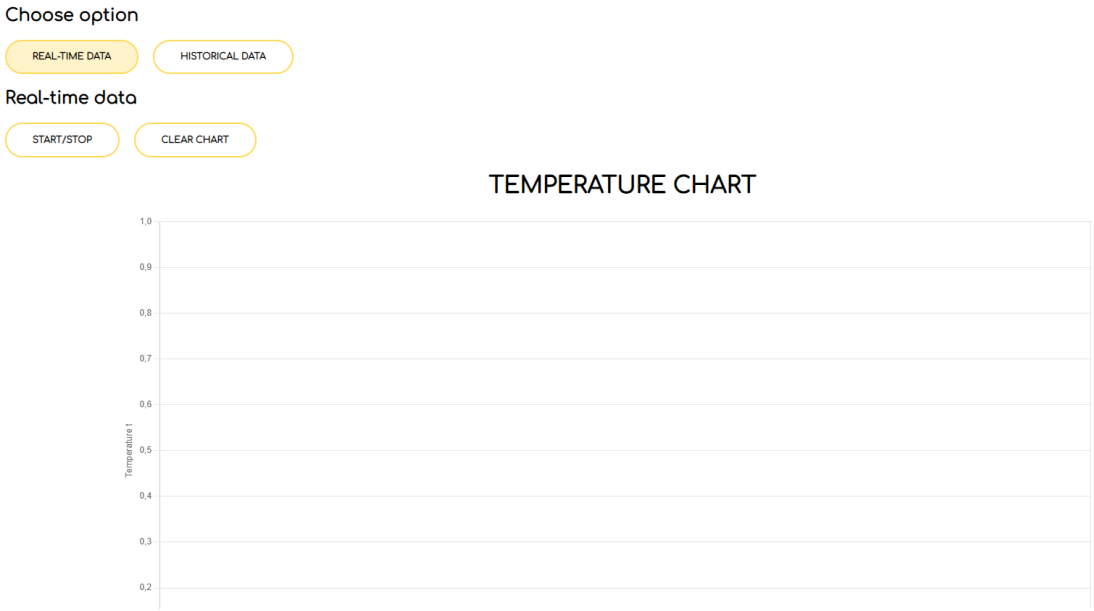

Рисунок 5.4 – Сторінка виводу даних в реальному часі

На даній сторінці також присутня можливість під’єднування пристроїв до сонячної панелі.

На рисунку 5.5 зображено блок з наявними пристроями які можна під’єднати. Враховуючи особливості сонячної панелі пристрої не можуть бути доданими без ввімкнення акумулятора. Для того щоб увімкнути пристрій необхідно натиснути на перемикач.

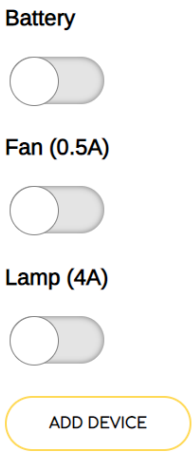

Рисунок 5.5 – Наявні пристрої для підключення

Для того щоб додавати власні пристрої необхідно натиснути кнопку Add device, після чого відкривається форма (рис. 5.6) в якій необхідно ввести назву пристрою та силу струму, яку споживає пристрій.

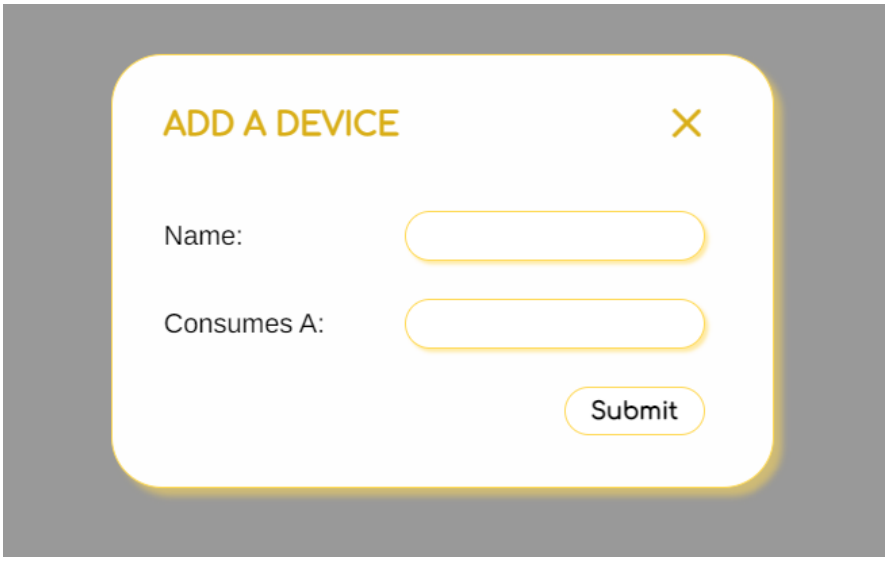

Рисунок 5.6 – Форма додавання нових пристроїв

Висновки до розділу 5

У п’ятому розділі було розглянуто кроки встановлення та налаштування програмного продукту для користування потенційним користувачем. Було описано підготовчі кроки, у яких описано все необхідне для запуску додатку. Також описано роботу користувача і його взаємодію із додатком. Розглянуто можливості розробленого програмного продукту.

ВИСНОВКИ

У процесі написання роботи було проаналізовано наявну літературу за темою, проаналізовано існуючі аналоги систем та додатки на ринку, виділено їх переваги та недоліки. Розглянуто та обрано технології, мову програмування та фреймворки для написання програмного продукту. Обрано основну архітектуру системи, розроблено систему з відповідністю до архітектури. Описано підготовчі кроки та процес інсталяції і запуску програмного продукту. Описано взаємодію користувача з системою.

Програмний продукт розроблений під час написання цієї роботи має вирішує поставлені задачі, відповідає поставленим вимогам.

Результатом цієї роботи є система, що складається з двох серверів та клієнтської частини, що допомагає обробити та візуалізувати дані лабораторного стенду сонячної панелі.

В подальшому ця система дозволить створити комплекс моніторингу всіх датчиків лабораторного стенду, та також дозволить отримувати, оброблювати і візуалізувати дані від великого набору сенсорів у єдиній оболонці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chart.js documentation: веб-сайт. URL: https://www.chartjs.org/docs/latest/ (дата звернення: 25.05.2023)
2. Docker documentation: веб-сайт. URL: https://docs.docker.com/ (дата звернення: 24.05.2023)
3. Flask documentation: веб-сайт. URL: https://flask.palletsprojects.com/en/2.3.x/ (дата звернення: 26.05.2023)
4. Haverbeke M. Eloquent Javascript. San Francisco : No Starch Press, 2009. 224 p.
5. Kafka documentation: веб-сайт. URL: https://kafka.apache.org/documentation/ (дата звернення: 26.05.2023)
6. Kshemkalyani A. D. Distributed computing: principles, algorithms, and systems. Cambridge : Cambridge University Press, 2008. 736 p.
7. Matthias K., Kane S. P. Docker: Up & Running: Shipping Reliable Containers in Production. O'Reilly Media, 2018. 352 p.
8. Newman S. Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, Incorporated, 2020. 250 p.
9. Node.js documentation: веб-сайт. URL: https://nodejs.org/en/docs (дата звернення: 24.05.2023)
10. Pandas documentation: веб-сайт. URL: https://pandas.pydata.org/docs/ (дата звернення: 25.05.2023)
11. Parcel documentation: веб-сайт. URL: https://parceljs.org/docs/ (дата звернення: 25.05.2023)
12. PostgreSQl documentation: веб-сайт. URL: https://www.postgresql.org/docs/ (дата звернення: 24.05.2023)
13. Python documentation: веб-сайт. URL: https://docs.python.org/3/ (дата звернення: 25.05.2023)

14. Shiflett C. HTTP developer's handbook. Indianapolis, Ind : Sams, 2003. 282 p.

15. Slatkin B. Effective Python: 90 Specific Ways to Write Better Python. Addison-Wesley Professional, 2019. 480 p.

16. Smith B. Beginning JSON. Apress, 2015. 324 p.

17. Мисак Й. С. Сонячна енергетика: теорія та практика / Й. С. Мисак, О. Т. Возняк, О. С. Дацько, С. П. Шаповал. — Львів: вид-во Львів. політехніки, 2014. — 340 с.

ДОДАТОК А

Веб-додаток візуалізації даних лабораторного стенду сонячної панелі

Програмний код

Аркушів 9

2023

```
import threading
class CustomThread(threading.Thread):
    def __init__(self, target):
        threading.Thread.__init__(self)
        self.stop_event = threading.Event()
        self._target = target
    def run(self):
        while not self.stop_event.is_set():
            self._target()
            print("Run")
            print(self.stop_event.is_set())
    def stop(self):
        self.stop_event.set()
        print(self.stop_event.is_set())

import json
import logging
import time
from confluent_kafka import Producer
from data import generator
class SolarDataProducer:
    def __init__(self) -> None:
        logging.basicConfig(format='%(asctime)s %(message)s',
                            datefmt='%Y-%m-%d %H:%M:%S',
                            filename='../producer.log',
                            filemode='w')
        self.logger = logging.getLogger()
        self.logger.setLevel(logging.INFO)
        ######################
        print('Kafka Producer has been initiated...')
        self.producer = Producer({'bootstrap.servers': 'localhost:29092'})
    def receipt(self, err, msg):
        if err is not None:
            print('Error: {}'.format(err))
        else:
            message = 'Produced message on topic {} with value of {}\n'.format(msg.topic(), msg.value().decode('utf-8'))
            self.logger.info(message)
            print(message)
    def produce_in_date_range(self, start_date: str, end_date: str):
        solar_data = generator.generate_data(start_date, end_date)
        for data in solar_data:
            message = json.dumps(data.to_json())
            self.producer.produce('solar-data', message.encode('utf-8'), callback=self.receipt)
            self.producer.flush()
    def produce_real_time(self):
        data = generator.generate_one()
        message = json.dumps(data.to_json())
        self.producer.poll(1)
        self.producer.produce('solar-data_rt', message.encode('utf-8'), callback=self.receipt)
        self.producer.flush()
        time.sleep(5)

import json
import signal
from threading import Event
from time import sleep
from flask import Flask, jsonify
from flask_cors import CORS
from flask_kafka import FlaskKafka
```

```
app = Flask(__name__)
CORS(app)
data = []
data_period = []
INTERRUPT_EVENT = Event()
bus = FlaskKafka(INTERRUPT_EVENT,
                 bootstrap_servers=",".join(["localhost:29092"]),
                 group_id="python-consumer"
                 )
def listen_kill_server():
    signal.signal(signal.SIGTERM, bus.interrupted_process)
    signal.signal(signal.SIGINT, bus.interrupted_process)
    # signal.signal(signal.SIGQUIT, bus.interrupted_process)
    # signal.signal(signal.SIGHUP, bus.interrupted_process)
@bus.handle('solar-data_rt')
def test_topic_handler(msg):
    global data
    data_s = json.loads(msg.value)
    data.append(data_s)
    print("consumed {} from solar-data_rt".format(data_s))
    print(data_s)
@bus.handle('solar-data')
def test_topic_handler(msg):
    global data
    data_s = json.loads(msg.value)
    data_period.append(data_s)
    print("consumed {} from solar-data".format(data_s))
    print(data_s)
@app.route('/api/data')
def get_data():
    global data_period
    len1 = len(data_period)
    sleep(1)
    len2 = len(data_period)
    while len1 != len2:
        len1 = len(data_period)
        sleep(2)
        len2 = len(data_period)
    result = data_period
    data_period = []
    return jsonify(result)
@app.route('/api/data_rt')
def get_data_rt():
    global data
    result = data
    data = []
    return jsonify(result)
if __name__ == '__main__':
    bus.run()
    listen_kill_server()
    app.run(debug=False, port=5004)

from sqlalchemy import Column, Integer, Float, DateTime, UniqueConstraint
from sqlalchemy.ext.declarative import declarative_base
from data.solar_data import SolarData
Base = declarative_base()
class SolarDataModel(Base):
    __tablename__ = 'solar_data'
    id = Column(Integer, primary_key=True)
    timestamp = Column(DateTime, unique=True)
```

```
    temperature = Column(Float)
    power = Column(Float)
    dc_power = Column(Float)
    solar_irradiation = Column(Float)
    shadow_coefficient = Column(Float)
    __table_args__ = (UniqueConstraint('timestamp', name='_timestamp_uc'),)
    def from_solar_data_object(self, solar_data: SolarData):
        self.timestamp = solar_data.timestamp
        self.temperature = solar_data.temperature
        self.power = solar_data.power
        self.dc_power = solar_data.dc_power
        self.solar_irradiation = solar_data.solar_irradiation
        self.shadow_coefficient = solar_data.shadow_coefficient
```

```
import axios from "axios";
export default axios
import {Chart, registerables} from 'chart.js';
import axios from "./api";
import $ from 'jquery'
import 'chartjs-adapter-luxon';
import * as config from "./chart_config";
import {Battery, Device} from "./devices_config"
Chart.register(...registerables);
let rtTempChart = new Chart(document.getElementById('data-rt-temp-chart'), config.rt_temp_config);
let rtPowerChart = new Chart(document.getElementById('data-rt-power-chart'), config.rt_power_config);
let rtDcPowerChart = new Chart(document.getElementById('data-rt-dc-power-chart'), config.rt_dc_power_config);
let rtCurrentChart = new Chart(document.getElementById('data-rt-current-chart'), config.rt_current_config);
let devices = {
    "battery_switch": new Battery()
}
function addClearRTChartHandler() {
    let clearButton = document.getElementById('clearRTBtn');
    clearButton.addEventListener('click', () => {
        rtTempChart.data.datasets.forEach((dataset) => {
            dataset.data = [];
        });
        rtTempChart.update();
        rtPowerChart.data.datasets.forEach((dataset) => {
            dataset.data = [];
        });
        rtPowerChart.update();
        rtDcPowerChart.data.datasets.forEach((dataset) => {
            dataset.data = [];
        });
        rtDcPowerChart.update();
        const parentElement = document.getElementById('table-body-rt');
        while (parentElement.childNodes.length > 1) {
            parentElement.removeChild(parentElement.lastChild);
        }
    });
}
function parseAndFormatISODate(isoString) {
    const date = new Date(isoString);
    const formatter = new Intl.DateTimeFormat('en-GB', {
        day: '2-digit',
        month: '2-digit',
        year: 'numeric',
        hour: '2-digit',
        minute: '2-digit',
        hour12: false
```

```
    });
    return formatter.format(date);
}
function insertTableRow(jsonData, tableId) {
    let tableBody = document.getElementById(tableId);
    let newRow = tableBody.insertRow();
    let order = ['timestamp', 'temperature', 'power', 'dc_power', 'current', 'solar_irradiation', 'shadow_coefficient']
    for (const key of order) {
        if (jsonData.hasOwnProperty(key)) {
            let newCell = newRow.insertCell();
            let newValue = document.createElement("span");
            switch (key) {
                case "timestamp":
                    newValue.innerHTML = parseAndFormatISODate(jsonData[key]);
                    break
                case "power":
                    newValue.innerHTML = jsonData[key].toFixed(3);
                    break
                case "dc_power":
                    newValue.innerHTML = jsonData[key].toFixed(3);
                    break
                case "current":
                    newValue.innerHTML = jsonData[key].toFixed(3);
                    break
                case "shadow_coefficient":
                    newValue.innerHTML = jsonData[key].toFixed(3);
                    break
                case "temperature":
                    newValue.innerHTML = jsonData[key].toFixed(3);
                    break
                case "solar_irradiation":
                    newValue.innerHTML = jsonData[key].toFixed(3);
                    break
                default:
                    newValue.innerHTML = jsonData[key];
            }
            newCell.appendChild(newValue);
        }
    }
}
let alertShown = false;
devices['fan_switch'] = new Device("Fan", 0.5, 12, devices['battery_switch']);
devices['lamp_switch'] = new Device("Lamp", 4, 12, devices['battery_switch']);
function drawRtCharts() {
    axios.get("http://127.0.0.1:5004/api/data_rt").then(response => {
        response.data.forEach(elem => {
            console.log(response.data);
            /* Main logic - init all default devices in js code
            * call update for every device
            * call update for the battery and get the final measures for the current and voltage
            * update the charts and the percentages accordingly
            * */
            if (rtTempChart !== null) {
                let dc_power = 20.3;
                let dc_power_change = 0;
                let current = 0;
                let batteryCharge = $('#battery_charge');
                let battery = devices['battery_switch'];
                batteryCharge.text('Battery (charge:' + devices.battery_switch.chargeStatus.toFixed(1) + '%)');
                for (const device of Object.values(devices)) {
```

```
                if (device.is_on) {
                    current += device.update();
                    dc_power_change += current / 6;
                }
            }
            if (dc_power_change > 0) {
                dc_power = 12 - dc_power_change;
            }
            if (current < 8.7 && battery.is_on) {
                let batteryCompensation = 0;
                for (const device of Object.values(devices)) {
                    if (device.is_on && device.name !== 'Battery') {
                        batteryCompensation += (device.consumes_A / 2);
                    }
                }
                battery.chargeStatus += batteryCompensation;
            }
            if (current > 8.7) {
                current = 8.7;
                if (!alertShown) {
                    alert("Solar panel at it's peak, adding current draw from battery");
                    alertShown = true;
                }
            }
            addDataRt(rtTempChart, elem['timestamp'], elem['temperature']);
            addDataRt(rtPowerChart, elem['timestamp'], Math.floor(elem['power']));
            addDataRt(rtDcPowerChart, elem['timestamp'], dc_power);
            addDataRt(rtCurrentChart, elem['timestamp'], current);
            elem['dc_power'] = dc_power;
            elem['current'] = current;
            insertTableRow(elem, "table-body-rt");
        }
    })
});
}
function addDataHistorical(chart, label, data) {
    let dateFromData = new Date(label);
    let dateNow = new Date();
    chart.data.labels.push(label);
    if (dateFromData < dateNow) {
        chart.data.datasets[0].data.push(data); // add to first dataset (same color)
        chart.data.datasets[1].data.push(data);
    } else {
        chart.data.datasets[1].data.push(data); // add to second dataset (different color)
    }
    chart.update();
}
function addDataRt(chart, label, data) {
    chart.data.labels.push(label);
    chart.data.datasets.forEach((dataset) => {
        dataset.data.push(data);
    });
    chart.update();
}
setInterval(drawRtCharts, 7000)
// Period data
let tempChart = new Chart(document.getElementById('data-temp-chart'), config.temp_config);
let powerChart = new Chart(document.getElementById('data-power-chart'), config.power_config);
let dcPowerChart = new Chart(document.getElementById('data-dc-power-chart'), config.dc_power_config);
function addClearHistoryChartHandler() {
```

```
    let clearButton = document.getElementById('clearHistoryBtn');
    clearButton.addEventListener('click', () => {
        tempChart.data.datasets.forEach((dataset) => {
            dataset.data = [];
        });
        tempChart.update();
        powerChart.data.datasets.forEach((dataset) => {
            dataset.data = [];
        });
        powerChart.update();
        dcPowerChart.data.datasets.forEach((dataset) => {
            dataset.data = [];
        });
        dcPowerChart.update();
        const parentElement = document.getElementById('table-body');
        while (parentElement.childNodes.length > 1) {
            parentElement.removeChild(parentElement.lastChild);
        }
        document.getElementById("maxTemp").innerHTML = "";
        document.getElementById("minTemp").innerHTML = "";
        document.getElementById("avgTemp").innerHTML = "";
        document.getElementById("powerHoursMZ").innerHTML = "";
        document.getElementById("powerHoursAvg").innerHTML = "";
        document.getElementById("powerHoursMax").innerHTML = "";
        document.getElementById("dc_powerHoursMZ").innerHTML = "";
        document.getElementById("dc_powerHoursAvg").innerHTML = "";
        document.getElementById("dc_powerHoursMax").innerHTML = "";
    });
}
function drawHistoryCharts(data) {
    let fullData = data.data;
    document.getElementById("maxTemp").innerHTML = fullData.maxTemp.toFixed(3);
    document.getElementById("minTemp").innerHTML = fullData.minTemp.toFixed(3);
    document.getElementById("avgTemp").innerHTML = fullData.avgTemp.toFixed(3);
    let powerFromMZ = new Date(fullData.powerHoursMZ[0]);
    let powerToMZ = new Date(fullData.powerHoursMZ[1]);
    document.getElementById("powerHoursMZ").innerHTML = powerFromMZ.getUTCHours() + ":" +
powerFromMZ.getUTCMinutes().toString().padStart(2, '0') + " - " + powerToMZ.getUTCHours() + ":" +
powerToMZ.getUTCMinutes().toString().padStart(2, '0');
    document.getElementById("powerHoursAvg").innerHTML = fullData.powerHoursAvg.toFixed(3);
    document.getElementById("powerHoursMax").innerHTML = fullData.powerHoursMax.toFixed(3);
    let dc_powerFromMZ = new Date(fullData.dc_powerHoursMZ[0]);
    let dc_powerToMZ = new Date(fullData.dc_powerHoursMZ[1]);
    document.getElementById("dc_powerHoursMZ").innerHTML = dc_powerFromMZ.getUTCHours() + ":" +
dc_powerFromMZ.getUTCMinutes().toString().padStart(2, '0') + " - " + dc_powerToMZ.getUTCHours() + ":" +
dc_powerToMZ.getUTCMinutes().toString().padStart(2, '0');
    document.getElementById("dc_powerHoursAvg").innerHTML = fullData.dc_powerHoursAvg.toFixed(3);
    document.getElementById("dc_powerHoursMax").innerHTML = fullData.dc_powerHoursMax.toFixed(3);
    fullData.data.forEach(data => {
        addDataHistorical(tempChart, data['timestamp'], Math.floor(data['temperature']))
        addDataHistorical(powerChart, data['timestamp'], Math.floor(data['power']))
        addDataHistorical(dcPowerChart, data['timestamp'], Math.floor(data['dc_power']))
        insertTableRow(data, "table-body")
    });
}
function addSubmitButtonHandlers() {
    $('.date-form').on('submit', function (event) {
        event.preventDefault();
        let form = $('#' + event.currentTarget.id); // Getting the info about from what form submit came
        const startDateObj = new Date(form.find('.start-date').val());
```

```
        const endDateObj = new Date(form.find('.end-date').val());
        if (endDateObj <= startDateObj) {
            alert('End date must be later than start date');
            return;
        }
        const startDate = startDateObj.toISOString().slice(0, 19).replace('T', ' ');
        const endDate = endDateObj.toISOString().slice(0, 19).replace('T', ' ');
        const diffTime = endDateObj.getTime() - startDateObj.getTime();
        const diffDays = Math.floor(diffTime / (1000 * 60 * 60 * 24));
        switch (event.currentTarget.id) {
            case "datePicker":
                axios.get('http://127.0.0.1:5001/api/get_period?startDate=' + startDate + '&endDate=' + endDate + "&diff=" +
diffDays).then(response => {
                    console.log(response)
                    drawHistoryCharts(response);
                }).catch(e => console.log(e))
            case "datePickerDb":
                axios.get('http://127.0.0.1:5001/api/fill_db?startDate=' + startDate + '&endDate=' + endDate).then(response => {
                    console.log(response)
                }).catch(e => console.log(e))
        }
    });
}
function addDevicesModalFormHandler() {
    $('#show-modal').on('click', function (event) {
        event.preventDefault();
        $('#myModal').show();
    });
    // Add event listener to form submission
    $('#myForm').on('submit', function (event) {
        event.preventDefault(); // Prevent default form submission behavior
        // Retrieve input values
        let name = $('#name').val();
        let consumesA = $('#consumes_A').val();
        console.log(name);
        console.log(consumesA);
        // Generate HTML part using input values
        let device_id = name + '_switch';
        let newDeviceHtml = '<div class="device">\n' +
            '            <h3 class="'info_title">' + name + ' (' + parseFloat(consumesA) + 'A)</h3>\n' +
            '            <input id="'' + name + '_switch" type="checkbox">\n' +
            '        </div>'
        // Append HTML part to the container
        $('#checkboxes').append(newDeviceHtml);
        // Close the modal
        $('#myModal').hide();
        devices[device_id] = new Device(name, parseFloat(consumesA), 12, devices['battery_switch']);
        $('#' + device_id).on('click', function () {
            let isChecked = $(this).is(':checked');
            // Check the checkbox only if a certain condition is met
            if (devices['battery_switch'].is_on) {
                $(this).prop('checked', isChecked);
            } else {
                // Uncheck the checkbox if the condition is not met
                $(this).prop('checked', false);
            }
        });
        $('#' + device_id).on('change', function () {
            let isChecked = $(this).is(':checked');
            if (isChecked) {
```

```
                devices[device_id].turn_on();
            } else {
                devices[device_id].turn_off();
            }
        });
    });
}
function initDefaultDevices() {
    for (const device_id in devices) {
        if (device_id !== 'battery_switch') {
            $('#' + device_id).on('click', function () {
                let isChecked = $(this).is(':checked');
                // Check the checkbox only if a certain condition is met
                if (devices['battery_switch'].is_on) {
                    $(this).prop('checked', isChecked);
                } else {
                    // Uncheck the checkbox if the condition is not met
                    $(this).prop('checked', false);
                }
            });
        }
        $('#' + device_id).on('change', function () {
            let isChecked = $(this).is(':checked');
            if (isChecked) {
                devices[device_id].turn_on();
            } else {
                devices[device_id].turn_off();
            }
        });
    }
}
```