ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ЗЕЛЕНОЇ ЕНЕРГЕТИКИ ДЛЯ ЕФЕКТИВНОГО БАЛАНСУВАННЯ ЕНЕРГОСИСТЕМИ»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми «Інформаційні системи та технології»
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Денис ПАРШУТКІН

_____ _____

(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: Здобувач вищої освіти гр. ІСДМ-62

Денис ПАРШУТКІН
Ім'я, ПРІЗВИЩЕ

Керівник: Сергій СІМЧЕНКО
Доцент каф. ВММФ, кандидат фізико-математичних наук
Ім'я, ПРІЗВИЩЕ

Рецензент: _____
Ім'я, ПРІЗВИЩЕ

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інформаційні системи та технології

Ступінь вищої освіти «Магістр»

Спеціальність 126 Інформаційні системи та технології

Освітньо-професійна програма – «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІСТ
_____ К. П. Сторчак
“____” _____2023 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Паршуткіну Денису Олександровичу
(прізвище, ім’я, по батькові)

1.Тема кваліфікаційної роботи Розробка системи моніторингу зеленої енергетики для ефективного балансування енергосистеми

керівник кваліфікаційної роботи Сергій СІМЧЕНКО, доцент каф. ВММФ
(Ім’я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «_____»_____20__ р. № ____

2. Строк подання кваліфікаційної роботи «___»_____20__ р.

3. Вихідні дані до кваліфікаційної роботи: система моніторингу зеленої енергетики для ефективного балансування енергосистеми

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз стану та розвитку зеленої енергетики.
2. Визначення програмно-технічного комплексу системи моніторингу зеленої енергетики
3. Розробка системи моніторингу зеленої енергетики для ефективного балансування енергосистеми

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «___»_____20__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури.	09.10.2023-22.10.2023	виконав
2	Аналіз стану зеленої енергетики. Визначення перспектив розвитку.	23.10.2023-29.10.2023	виконав
3	Аналіз існуючих систем моніторингу	30.10.2023-05.11.2023	виконав
4	Визначення програмно-технічного комплексу системи моніторингу зеленої енергетики	06.11.2023-19.11.2023	виконав
5	Розробка системи моніторингу зеленої енергетики	20.11.2023-10.12.2023	виконав
6	Тестування системи моніторингу зеленої енергетики	11.12.2023-17.12.2023	виконав
7	Формування висновків розробленої системи. Оформлення кваліфікаційної роботи.	18.12.2023-22.12.2023	виконав
8	Розробка демонстраційних матеріалів (Презентація)	23.12.2023-27.12.2023	виконав

Здобувач(ка) вищої освіти _____ Денис ПАРШУТКІН
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Сергій СІМЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 89 стор., 55 рис., 6 табл., 63 джерел, 3 додатки.

Об'єкт дослідження – процес розробки системи моніторингу зеленої енергетики.

Предмет дослідження – система моніторингу зеленої енергетики.

Мета роботи – розробити систему моніторингу зеленої енергетики для ефективного балансування енергосистеми.

Короткий зміст роботи:

В кваліфікаційній роботі визначено важливість зеленої енергетики, розвиток та потреба в інноваціях відновлюваних джерел енергії.

Досліджено та створено технології збору, передачі, зберігання та візуалізації даних з датчиків.

Розроблено систему моніторингу енергетики для ефективного балансування енергосистеми. Створено простий, інтерактивний та захищений інтерфейс візуалізації.

КЛЮЧОВІ СЛОВА: ЗЕЛЕНА ЕНЕРГЕТИКА, МОНІТОРИНГ, ВІЗУАЛІЗАЦІЯ, MQTT, TSDB, PROMETHEUS, GRAFANA, SPRING BOOT.

ABSTRACT

The text part of the master's qualification work: 89 pages, 55 pictures, 6 tables, 63 sources, 3 appendices.

The purpose of the work – to develop a green energy monitoring system for efficient balancing of the power system.

Object of research – the process of developing a green energy monitoring system.

The subject of research – a green energy monitoring system.

Summary of work:

The qualification work identifies the importance of green energy, the development and need for innovation of renewable energy sources.

The technologies for collecting, transmitting, storing and visualizing data from sensors were researched and created.

An energy monitoring system was developed for efficient balancing of the power system. A simple, interactive and secure visualization interface is created.

KEYWORDS: GREEN ENERGY, MONITORING, VISUALIZATION, MQTT, TSDB, PROMETHEUS, GRAFANA, SPRING BOOT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ВДЕ – Відновлюванні джерела енергії

ГВт – Гігават

EMS (Energy Management System) – Система енергоменеджменту

ПЗ – Програмне забезпечення

TSDB (Time series database) – База даних часових рядів

MQTT (Message Queuing Telemetry Transport) – Телеметричний транспорт черги повідомлень

IDE (Integrated Development Environment) – Інтегроване середовище розробки

IoT (Internet of Things) – Інтернет речей

БД – База даних

HTTP (HyperText Transfer Protocol) – Протокол передачі гіпертексту

HTML (HyperText Markup Language) – Мова гіпертекстової розмітки

CSS (Cascading Style Sheets) – Каскадні таблиці стилів

ВСТУП

Сьогодні постійне зростання попиту на електроенергію та проблеми, пов’язані зі зменшенням традиційних джерел енергії та зміною клімату, вимагають шукати нові ефективні стратегії зменшення викидів та забезпечення стабільності енергосистем. Це призводить до зростання частки зеленої енергетики в енергосистемі по всьому світу, яка базується на використанні відновлюваних джерел енергії.

Починаючи з кінця XX століття зростають інвестиції та капіталовкладення в розвиток альтернативної енергетики на світовому ринку. За останні роки збільшується щорічний приріст виробництва сонячної енергії, вітрової енергії на інших джерел енергії. В Україні також відбувається активний розвиток відновлюваних джерел енергії, що підтримується державною ініціативою. Розвиток енергетичної системи та укладені угоди з міжнародними партнерами щодо зменшення викидів парникових газів стимулює державу розробляти шляхи та можливості збільшення частки зеленої енергетики. Спостерігається інтенсивна розбудова відновлюваних джерел, які інтегровано в електроенергетичну систему на рівні розподільних електричних мереж. Це призводить до поступового переходу від централізованої моделі електропостачання, заснованої на потужних теплових і ядерних електростанціях, до комбінованої моделі, коли частина електроенергії генерується розподіленими джерелами.

Після початку повномасштабної війни в Україні та постійними обстрілами критичної інфраструктури з’являються нові виклики стабілізації роботи електромережі. Зростає потреба балансування електромережі як через постійний попит населення України, так через необхідність адаптації до змін у структурі та навантаженні системи внаслідок руйнувань, відключення та тимчасової втрати частини енергетичної інфраструктури. Тому для забезпечення стабілізації та ефективного балансування енергосистеми необхідно впроваджувати інноваційні технології для постійного контролю та моніторингу енергетики, включаючи зелену енергетику.

Актуальність теми. У сучасному світі енергетичні виклики та постійне зростання попиту на електроенергію вимагають нових та інноваційних підходів для роботи з енергією. Зростання виробництва енергії з відновлюваних джерел потребують системи нагляду та моніторингу. Це може розв'язання проблеми контролю нестабільності виробництва енергетики, ефективного балансування та впровадження гнучкості енергосистеми, покращення ефективності використання енергетичних ресурсів під час швидкого зростання частки зеленої енергетики у світі. Після початку війни в Україні створення системи моніторингу зеленої енергетики може бути корисною під час можливих руйнувань інфраструктури для постійного спостереження та швидкого реагування.

Мета і завдання дослідження. Мета дослідження полягає в розробці системи моніторингу зеленої енергетики для забезпечення ефективного балансування енергосистеми. Щоб досягти поставленої мети, потрібно проаналізувати стан та розвиток зеленої енергетики, сучасні системи моніторингу енергетики, створити систему моніторингу зеленої енергетики та провести дослідження та аналіз залучених інструментів та технологій.

Об'єкт дослідження. Об'єктом дослідження є процес створення системи моніторингу зеленої енергетики.

Предмет дослідження. Предметом дослідження є система моніторингу зеленої енергетики для ефективного балансування енергосистеми.

Методи дослідження. Методи алгоритмізації та програмування, методи вимірювання параметрів енергосистеми.

Наукова новизна одержаних результатів. Розроблена система використовує інтегрований підхід, поєднуючи різні методи та технології для комплексного інтерактивного моніторингу зеленої енергетики та ефективного балансування енергетичної системи. Вона концентрується на отриманні показників із датчиків, їх опрацюванні та візуалізації даних за допомогою безпечного та інтерактивного інтерфейсу.

Практичне значення одержаних результатів. Розроблена система може бути інтегрована з наявними енергетичними системами, використовуватися в

енергетичних компаніях, дистриб’юторах, промислових підприємствах, уряді, господарстві, дослідницьких інститутах.

Апробація результатів магістерської роботи та публікації:

Паршуткін Д.О., Теплюк О.В., Шапкін В.А. «Аналіз сучасних технологій розширення реляційних СУБД для роботи з часовими рядами». Наукова стаття у загальногалузевому науково-виробничому журналі «Зв’язок», м.Київ – випуск листопад-грудень №6, 2023.

1. АНАЛІЗ СТАНУ ЗЕЛЕНОЇ ЕНЕРГЕТИКИ ТА ОПИС СУЧАСНИХ СИСТЕМ МОНІТОРИНГУ ЕНЕРГОСИСТЕМ

1.1 Аналіз зеленої енергетики

Зелена, або відновлювана енергія – це енергія, яка отримана з природних відновлюваних джерел енергії (ВДЕ) та поповнюється з більшою швидкістю, ніж споживається. ВДЕ повсюди навколо нас.

Викопне паливо, таке як вугілля, нафта та газ, є невідновлюваними енергетичними ресурсами, для формування яких потрібні мільйони років. Викопне паливо, спалюване для виробництва енергії, спричиняє викиди шкідливих парникових газів, вуглекислого газу. Виробництво відновлюваної енергії створює набагато менші викиди, ніж спалювання викопного палива. Тому перехід від використання викопного палива, від якого наразі йде основна частина викидів, на постійне використання відновлюваної енергії є ключем у боротьбі з кліматичною кризою [1].

Все більший попит на енергію у світі та збільшення населення призводить до стрімкого використання енергетичних ресурсів, зокрема викопного палива (вугілля, нафта та газу). Це призвело до виникнення численних проблем, таких як виснаження резервів викопного палива, ефект парникового газу та інші екологічні труднощі. Геополітичні та військові конфлікти викликають непостійність цін на пальне. Такі проблеми створюють нестабільні умови, які можуть потенційно призвести до загрози для суспільства.

Всупереч цьому, відновлювані джерела енергії стають важливою альтернативою та єдиною відповіддю проблеми, що зростають. У 2012 році вони складали 22% від загального світового виробництва енергії, що було малоймовірним роками раніше. Надійне енергозабезпечення має важливе значення для економіки, забезпечуючи опалення, освітлення, роботу промислового

обладнання, транспорту та багато іншого.

ВДЕ суттєво зменшують викиди парникових газів порівняно з традиційними джерелами. Оскільки вони отримують енергію з постійних природних потоків у нашому оточенні, вони можуть бути стійким рішенням. Для того, щоб відновлювана енергія була дійсно стійкою, важливо, щоб вона була необмеженою і нешкідливою для навколишнього середовища [2]. Впровадження ВДЕ в енергетичному, тепловому та транспортному секторах є одним з основних факторів, що сприяють утриманню зростання середньої глобальної температури нижче 1,5°C. Відновлювані види палива та відновлюване тепло сприяють значному скороченню викидів транспорту та промисловості.

ВДЕ охоплюють гідроенергію, сонячну енергію, енергію вітру, геотермальну енергію, біоенергію, енергію океану (припливів і хвиль) тощо.

1.1.1 Гідроенергетика

Гідроенергетика – це галузь відновлювальної енергетики, що включає сукупність великих природних та штучних підсистем, які слугують для перетворення кінетичної енергії води в електроенергію.

Гідроенергоресурси є відновлювальним та найбільш екологічно чистим джерелом енергії. Їх використання дозволяє зменшити викиди парникових газів в атмосферу. Гідроелектростанції відзначаються великою мобільністю, що дозволяє їм за необхідності швидко збільшувати виробіток електроенергії, особливо в період пікового навантаження. Крім того, вони забезпечують надійність енергопостачання навіть в аварійних ситуаціях [3].

За останні роки відбувається зростання глобальної потужності гідроенергетики, виробництва чистої енергії та гідроакумуляції. Глобальна встановлена потужність гідроенергетики зросла на 26 ГВт до 1360 ГВт у 2021 році. З гідроенергії вироблено 4250 ТВт чистої електроенергії, що в 1,5 раза перевищує

всю кількість споживаної електроенергії в Європейському Союзі. До енергетичної мережі було додано 4,7 ГВт гідроакумулювальної гідроенергії, що втричі більше, ніж у 2020 році [4].

Гідроелектростанції виробляють електроенергію, використовуючи потік води як джерело енергії. Зазвичай такі станції розташовані на річках або поблизу водних джерел. Виробництво електроенергії залежить від об'єму потоку води, висоти та напору від однієї точки до іншої. Шляхом використання гребель на річках можна штучно створювати перепади рівня води. Чим сильніший потік і напір, тим більше електроенергії можна згенерувати. На рисунку 1.1 показано роботу гідроелектростанції [5].

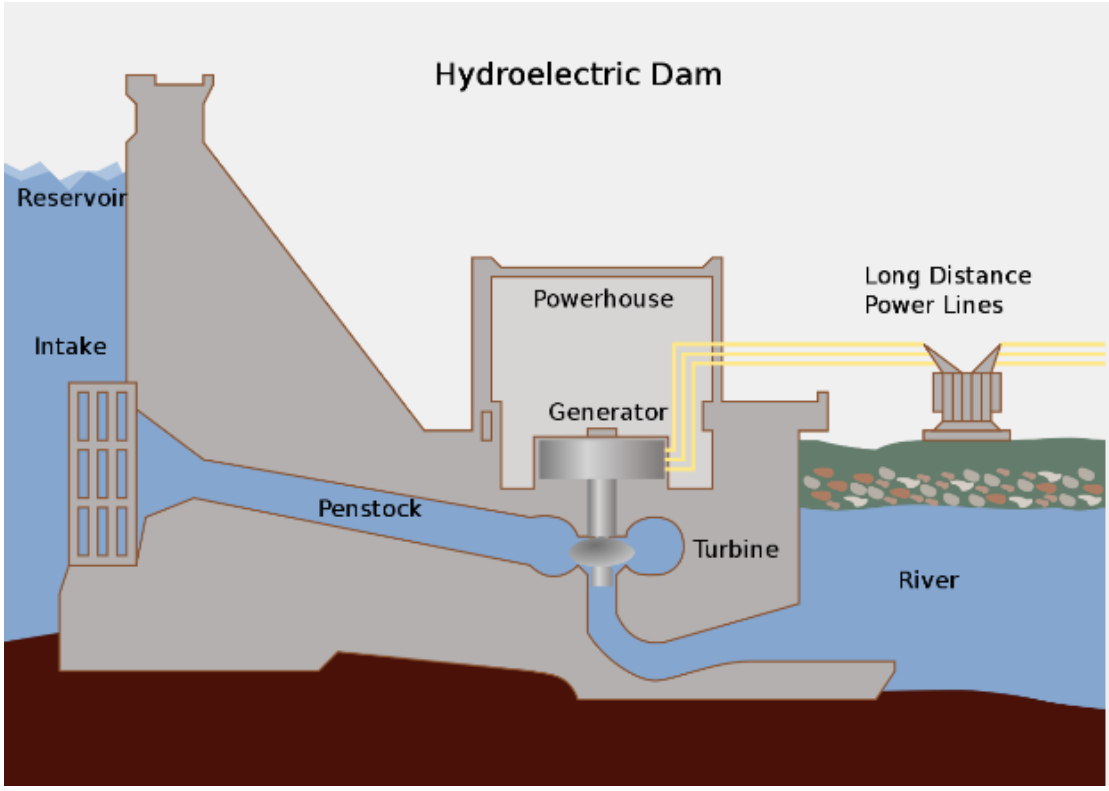

Рисунок 1.1 – Принцип роботи гідроелектростанції

Вода під впливом сили тяжіння, переливається з верхнього рівня в нижній через спеціальні протоки, де розташовані гідротурбіни. На гідроелектростанцію вода направляється через трубу, відому як гідроагрегат, потім подається на лопаті турбіни. Лопаті цих турбін рухаються під дією водяного потоку, створюючи

механічну енергію. Турбіна обертається під впливом водного потоку, передаючи енергію генератору, який згодом виробляє електроенергію. Більшість гідроелектростанцій, річкових систем та гідроакумулювальних електростанцій працюють за цим принципом [5].

Україна має величезні водні ресурси – 94,1 млрд. кубометрів та використовує їх на 50%. Встановлена потужність ГЕС і ГАЕС в Об'єднані енергетичній системі України складає 7350 МВт. [3]

Для розвитку гідроенергетики України необхідна модернізація та технічне удосконалення гідровузлів. Заміну застарілого фізичного обладнання слід проводити на високотехнологічному рівні, використовуючи автоматизацію та комп'ютеризацію.

1.1.2 Сонячна енергетика

Сонячна енергія – це кількість довгохвильової сонячної радіації, яке досягає землі й викликає систематичні зміни в теплообміні та температурі протягом якогось часового періоду [7].

Існують два основні методи перетворення сонячної енергії в електричну: термодинамічний та фотоелектричний.

У термодинамічному методі використовується схема теплових установок, де електрична енергія виробляється завдяки сонячному випромінюванню. Воно концентрується, замінюючи теплоту від згорання палива. Цей процес використовує традиційні теплові елементи.

Фотоелектричний метод базується на внутрішньому фотоефекті в напівпровідниках. Внутрішній фотоефект виникає внаслідок поглиблення світлового кванта, що призводить до вивільнення електронів і дірок. Завдяки різниці у розташуванні вивільнених електронів і дірок у внутрішній структурі батареї утворюється електричне поле, яке приводить до виникнення напруги між

двома сторонами напівпровідника. Через внутрішню напругу електрони та дірки рухаються в протилежних напрямках, утворюючи електричний струм [8]. На рисунку 1.2 можна побачити принцип роботи сонячної батареї [9]. Сучасні сонячні батареї використовують пластини з кремнію різних типів.

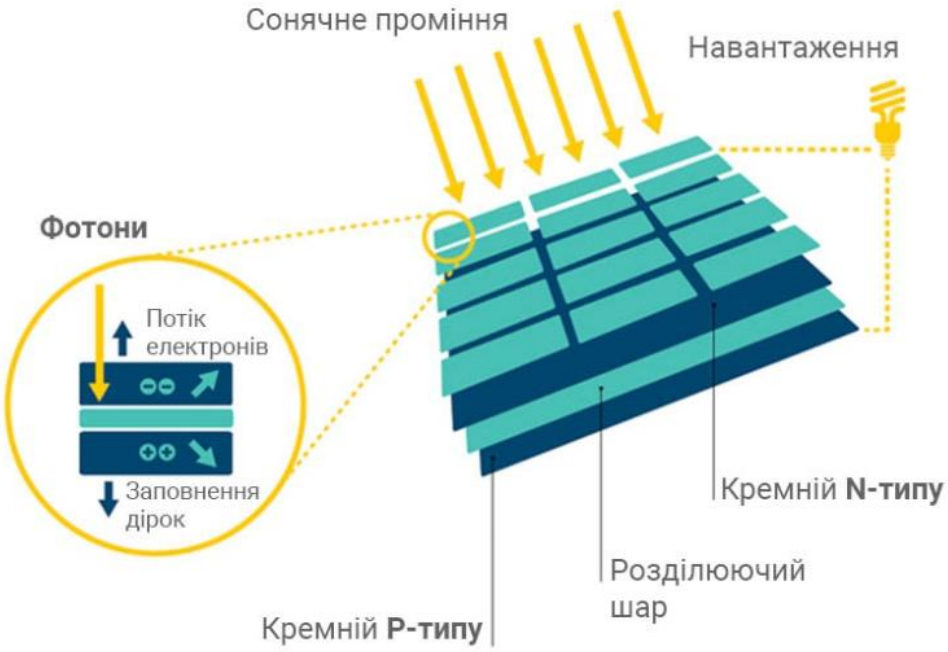

Рисунок 1.2 – Принцип роботи сонячної батареї

В останній час розробляються та впроваджуються все більше нових інновацій та технологій в сонячній енергетиці. Найбільш перспективною розробкою є впровадження та використання наноматеріалів, що підвищують ефективність сонячних елементів. Інтеграція штучного інтелекту дає змогу оптимізувати та краще налаштувати сонячні батареї для збільшення продуктивності. Розвиток сучасних акумуляторів та батареї допомагає вирішити проблему нестачі ємності зберігання, дозволяючи зберігати та використовувати надлишок сонячної енергії за потреби. Цей технічний напрямок швидко розвивається, що з кожним роком покращує технологію отримання сонячної енергії.

Розвиток сонячної енергетики обумовлений такими факторами як технологічний прогрес, зміни в державній політиці та вартості інших джерел

енергії. Хоча точні прогнози майбутнього складно зробити, загалом очікується зростання ролі сонячної енергетики у глобальному енергетичному балансі. Спостерігається кілька тенденцій, що свідчать про продовження росту сонячної енергетики. Наприклад, стрімке зниження вартості сонячних панелей та поліпшення їхньої ефективності роблять сонячну енергію більш конкурентоспроможною на фоні невідновлюваних джерел. Національні та міжнародні ініціативи також сприяють впровадженню сонячної енергетики, установлюючи цілі та стимули для підприємств.

Розвитку сонячної енергетики 2023 року виділяється зниженням вартості сонячних панелей, підвищення їхньої ефективності, зростання комунальних проєктів та впровадження сонячної енергії на підприємствах. Ринок дахових сонячних електростанцій також прогресує, а збільшення обсягів зберігання сонячної енергії відображає зростання її використання та потребу в ефективних засобах зберігання. Спостерігаються досягнення у розробці більш ефективних сонячних панелей та використанні нових матеріалів, що сприятиме подальшому розвитку цієї галузі [10].

Загалом, сонячна енергетика продовжить зростати завдяки технологічному прогресу, зниженню вартості та підтримці з боку політики. Проте конкретний темп розвитку буде залежати від багатьох факторів, у тому числі відповідних підходів влади.

Сонячна енергетика є новою та швидкозростаючою галуззю в Україні, яка представляє понад 5% загального виробництва електроенергії на кінець 2021 року [11]. Південні регіони країни є найбільш сприятливими для розташування сонячних електростанцій. Вторгнення агресора та початок повномасштабної війни суттєво вплинув на розвиток галузі. Для її відновлення потрібно впроваджувати додаткове фінансування на покращення інфраструктури, залучення інвестицій для будівництва нових сонячних електростанцій та надання пільгових умов для компаній галузі.

1.1.3 Вітроенергетика

Вітроенергетика – це галузь енергетики, що перетворює кінетичну енергію вітру в електричну енергію. Енергія вітру є одним із найбільш доступних та простих у використанні джерел відновлювальної енергії, яке застосовувалося людством вже тисячі років. Ця форма енергії може бути використана у різних галузях, таких як рух вітрильних суден, обертання млинів для перемелювання зерна або створення електроенергії за допомогою вітрогенераторів [12].

Вітрогенератори працюють за простим принципом: вітер передає кінетичну енергію на ротор, який перетворює цю енергію у механічну. Далі за допомогою спеціального пристрою ця механічна енергія перетворюється в електричну, що дозволяє отримувати електроенергію безоплатно. Потужність вітряних електростанцій може варіюватися від 5 до 4500 кВт. Та виробляти електроенергію при низькій вітровій швидкості [13].

Вітроенергетика є однією з найбільш розвинених галузей альтернативної енергетики. Енергія встановлених вітрогенераторів у світі перевищила 450 ГВт. Це дозволяє виробляти понад 3% всієї електроенергії на планеті. Прогнози вказують на те, що до 2025 року вітроенергетика може забезпечити близько 12% світового виробництва електроенергії. У деяких країнах частка вітрогенерації в електросистемі досягає значних результатів: 42% в Данії, 20% в Іспанії, та 19% в Німеччині та Ірландії. Загалом в ЄС вітрові електростанції забезпечують 7,5% виробництва електроенергії [12]. Більше одного мільйона працівників зайнято у сфері вітроенергетики.

Основні переваги використання енергії вітру включають поновленість та безкоштовність вітру. Також важливо відзначити, що використання енергії вітру не завдає значної шкоди навколишньому середовищу. Проте недоліками є нестабільність вітрових потоків, нерівномірний розподіл вітрового потенціалу та обмеження по мінімальній та максимальній швидкості вітру.

1.1.4 Геотермальна енергія

Геотермальна енергія являє собою тепло планети Земля й знаходиться буквально всередині неї.

Через те, що кора землі приймає теплоту розжарених надр, які труться одне об одне, як наслідок розпадаються радіоактивні елементи і виникають хімічні реакції. Постійний потік тепла з них прямує до поверхні Землі. Така енергія виходить назовні у вигляді термальних вод, пароводяних сумішей і природного пару. Люди можуть використовувати це тепло для вироблення електроенергії та теплопостачання [8].

Геотермальна енергія має дуже великі запаси. Її активними користувачами є такі країни як Угорщина, Італія, Мексика, Нова Зеландія, США, Японія. В Ісландії завдяки геотермальній енергії забезпечується навіть 26,5% вироблення електроенергії. Столицю Рейк'явік геотермальна енергія забезпечує теплом, і в 1943 р. там було зроблено 32 свердловини глибиною від 440 до 2400 м. Завдяки цим свердловинам підіймалася вода до поверхні з температурою від 60 до 130°С. Сьогодні працюють лише дев'ять з них.

Ресурси геотермальної енергії також розташовані й в Україні, та вони доволі значні. В Закарпатській, Миколаївській, Одеській, Херсонській областях представлені родовища геотермальних вод, які є придатними до використання. Найперспективнішими для використання є Карпатський регіон, менш значні в Полтавській, Харківській, Сумській і Чернігівській областях [14].

1.1.5 Біоенергетика

Біоенергетикою є галуззю енергетики, яка була виявлена та заснована на використанні біопалива, тобто палива, яке було отримано та вироблено з

органічних матеріалів. Прикладом таких матеріалів є рослини та залишки від них, відходи й побічні продукти, які можна використовувати, щоб виробляти енергію в біоенергетиці [15].

Біопаливо виробляється з цієї біомаси, яка є одним з найдавніших джерел енергії. Біомаса складається з органічних речовин, які виникають в рослинах через фотосинтез і можуть використовуватись, щоб отримати енергію. Вони включають усі види рослинності, такі як рослинні відходи сільського господарства і побутові відходи.

Біомаса відіграє дуже важливу роль в енергобалансах промислово розвинених країн, наприклад, і США – 4%, і Данія – 6%, Канада – 7%, Австрія – 14%, та Швеція – 16% загального споживання первинних енергоресурсів даних країн.

Взагалі існують декілька найпоширеніших методи використання біомаси в біоенергетиці. Перший метод – фізичний, який являє собою пряме спалювання. Наступний метод є хімічним, який включає піроліз, газифікацію, виробництво спиртів і масел для отримання моторного палива. Останній метод – мікробіологічний метод, який використовує анаеробну ферментацію з утворенням метану.

Біоенергетика являє собою одним із напрямків розвитку відновлюваних джерел енергії для України, проте розвиток є дуже повільним і досі відстає від Європи. Сьогодні частка біомаси у валовому кінцевому енергоспоживанні становить 1,78% [16].

1.2 Розвиток та інновації зеленої енергетики

Зелена енергетика – це ВДЕ, які ніколи не закінчаться у природі, наприклад, вода, вітер і сонце. Вони не впливають на довкілля, тому що не виробляють газів і ніяк не сприяють потеплінню. Для того, щоб температура довкілля не зростала вище за 2°C, треба зменшити середні викиди вуглекислого газу на 40-60%.

Зараз відбувається швидке зростання популярності зеленої енергетики через страх перед екологічними проблемами та подорожчання інших видів енергії. На рисунку 1.3 показано графік динаміки росту зеленої енергетики у світі [17].

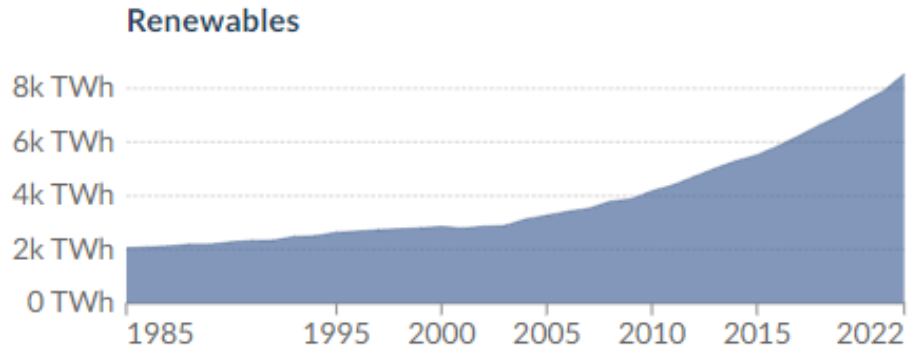

Рисунок 1.3 – Динаміка зростання зеленої енергетики у світі

Як видно з графіка, починаючи з 2000 року відбувається швидке зростання видобутку зеленої енергетики. Це можна пояснити збільшенням популяції населення світу, а тому збільшення попиту на енергетичні ресурси.

Найбільше енергії отримується через гідростанції, тому що в середині XX століття відбувалися значні інвестиції у проєкти, які пов’язані з гідроенергією. На рисунку 1.4 показано динаміку зростання різних видів ВДЕ [18].

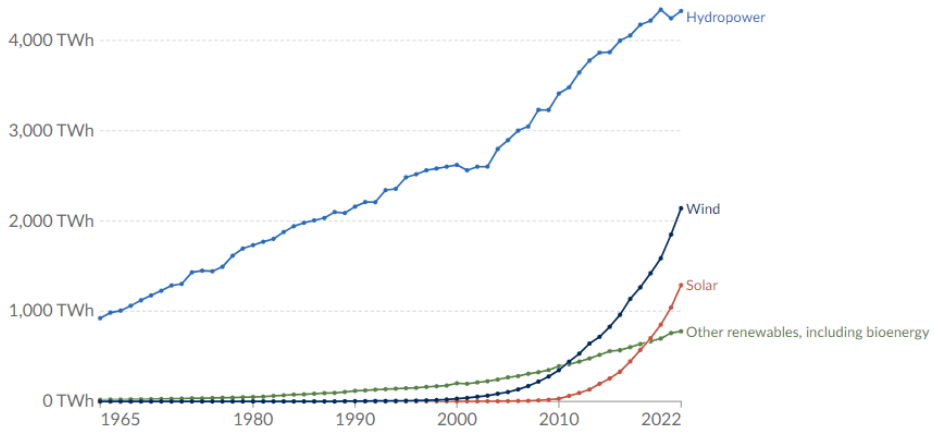

Рисунок 1.4 – Динаміка зростання виробництва різних видів ВДЕ

З початку XXI століття відбувається стрімке зростання виробництва й інших видів ВДЕ, таких як енергія вітру, сонця, біоенергія та інші.

Багато компаній та підприємств, які займаються виробництвом та розповсюдженням зеленої енергетики, використовують аналітику та моніторинг даних. У більшості випадків системи моніторингу має описовий характер і не дозволяє отримувати інформацію в режимі реального часу. Застарілі системи аналітики та моніторингу з часом втрачають свою ефективність, оскільки станції збільшуються в розмірах, мережі стають складнішими, а до систем ВДЕ приєднується все більше нових станцій та учасників. Тому лідери ринку енергетики активно шукають нові технологічні рішення та інновації.

Хмарні обчислення та Інтернет речей забезпечують збір метрик, зв'язок та обмін даними в масштабах всієї організації. Отриманні показники (напруги, температури, вібрації тощо) від різноманітного обладнання можуть бути легко збережені та відправленні для подальшого аналізу.

Алгоритми аналітики та великих даних дають можливість оперувати даними з різних джерел та на їх основі робити висновки в режимі реального часу, що допоможе запобіганню простою та несправності обладнання.

Інтернет речей та аналітика великих даних допомагають операторам вітрової енергетики збільшувати вихідну потужність вітрових турбін, зменшуючи при цьому витрати на експлуатацію та обслуговування. Технології цифровізації дозволяють створювати віртуальні копії фізичних систем, активів та процесів, що збільшує стабільність та надійність системи. Цифрове відтворювання інфраструктуру вітроелектростанції дозволяє системі вимірювати зовнішні дані, робити звіти про погоду, обслуговування та продуктивність аналогічних моделей вітрових турбін для покращення прогнозів та аналітики.

Сонячні енергетичні системи можуть бути оснащені датчиками IoT. Вони передають показники продуктивності в режимі реального часу на одну центральну панель управління, що дозволяє на ранній стадії виявляти проблеми в точці їх виникнення. Це дозволить вжити превентивних заходів від можливих несправностей та збоїв. Спільні енергетичні операції, оптимізація налаштування контуру, управління аварійними сигналами, цифрові операції, обслуговування та управління активами стають можливими для використання операторами сонячних

електростанцій завдяки впровадженню IoT та інших хмарних технологій моніторингу енергосистеми.

Цифровування гідроелектростанцій та впровадження можливостей IoT можуть надати операторам гідроенергетики практичну інформацію на основі даних від датчиків та обладнання. Це підвищить ефективності інфраструктури, мінімалізує простої, що розширить можливостей гідроенергетики. Передові компанії в галузі гідроенергетики зараз використовують комбінацію програмного та апаратного забезпечення для точного вимірювання та налаштування продуктивності своїх станцій. Впровадження машинного навчання та глибокого навчання може покращити прогнозування виробництва гідроенергії та оптимізувати експлуатацію обладнання.

Нове ПЗ для біоенергетики дозволить автоматизувати та оптимізувати загальні процеси роботи, такі як обробка палива, очищення димових газів, продуктивність парових турбін та розподіл електроенергії. Аналітичні калькулятори біомаси можуть оцінювати придатність різних типів сировини та розраховувати оптимальне завантаження біогазової установки. Також це дасть визначати оптимальний розмір сховища на основі наявного гною або енергетичних культур [19].

Отже, зростання популярності зеленої енергетики дало стрибок впровадженню нових інноваційних технологій для енергосистем. Більшість підприємств та установ все ще працює зі застарілим обладнанням та ПЗ. Тому впровадження сучасних систем IoT та хмарного моніторингу дозволять оптимізувати енергетичну інфраструктуру.

1.3 Зелена енергетика в Україні

Зелена енергетика в Україні відзначається значними досягненнями протягом останніх років. Так з 2018 року ця галузь почала стрімко зростати з кожним роком

все більше і більше.

В Україні дуже хороші умови для зеленої енергетики. За останні роки в Україні було зроблено кращі та вигідні умови на ринку енергетики. І тому в зеленій енергетиці України є значний потенціал для росту та розвитку [20].

Також впроваджуються в Україні інструменти стимулювання розвитку зеленої енергетики, такі як «зелені» тарифи, податкові та митні пільги. «Зелений» тариф застосовується для виробників ВДЕ, у яких держава закуповує електроенергію на вигідних умовах, що компенсує частину затрат. Податкові та митні пільги дозволяють виробникам звільнитися від оподаткування з імпорту установок, обладнання або комплектуючих, якщо вони застосовуються платником податку для власного виробництва або ідентичні товари з аналогічними показниками не виробляються в Україні [21]. Такі підходи роблять зелену енергетику вигіднішою.

Після початку повномасштабної війни в Україні впали показники видобутку зеленої енергетики. Це пов’язано з втратою частини об’єктів ВДЕ через окупацію, активні військові дії та руйнування. Так зелена енергетика до війни мала потужність близько 10 ГВт та інвестицій у понад 12 млрд доларів. Після 2022 року видобуток та інвестиції значно впали.

На півдні України розташовувалися близько 75% вітрових та 50% сонячних станцій. Їх прийшлося вивести з експлуатації через окупацію та неможливість забезпечення підтримки. Руйнування Каховської гідроелектростанції призвело до втрати гідроенергетичної інфраструктури.

Через війну частка ВДЕ України становить 5-6%, що значно менше від 13,4% довоєнного періоду [22]. Також значно впав заробіток зеленої енергетики, що зменшує можливості провадження інновацій.

Технології та ПЗ енергетичних установ, особливо державних, використовують застарілі системи аналітики та моніторингу, які не мають можливості оперувати даними в режимі реального часу. Тому питання впровадження сучасних систем моніторингу дуже гостре.

1.4 Балансування енергосистеми

Балансування енергосистеми забезпечує відповідність між споживанням та виробництвом енергії в будь-який момент часу в загальній енергетичній системі. Балансування енергетики допомагає стежити за частотою мережі, підвищуючи або зменшуючи подачу енергії в залежності від споживання.

Скорочення кількості вугільних електростанцій та інтеграція зеленої енергетики ускладнюють балансування енергосистеми. ВДЕ можуть швидко змінюватися в залежності від погодних умов та стану навколишнього середовища [23]. Це потребує гнучкі способи балансування мережі та постійний моніторинг енергосистеми.

Впроваджується багато методів та технологій для балансування енергомережі з інтегрованими ВДЕ. Технологія «розумних мереж» може оптимізувати попит та пропозицію електроенергії, зменшити втрати та перевантаження, підвищити безпеку та стійкість мережі. Вона включає «розумні» лічильники, датчики, інвертори, перемикачі та ПЗ. Ці пристрої та програми дозволяють постійно контролювати виробництво та подачу енергії без втручання людини, що набагато пришвидшує роботу над балансуванням мережі.

Гнучке виробництво дозволяє електростанціям швидко та ефективно регулювати свою потужність відповідно до коливань елементів мережі. Реагування на попит дозволяє контролювати споживання під час пікової частоти, регулюючи напругу та використання зеленої енергетики.

Інновації та дослідження у сфері зеленої енергетики можуть допомогти підвищити продуктивність та ефективність ВДЕ за допомогою розробки нових інструментів та сервісів тестування, контролю та моніторингу енергосистеми [24]. Це дозволить вирішити проблеми балансування енергетичної системи та швидко реагувати на можливий дисбаланс мережі.

1.5 Моніторинг енергетики

Енергетичний моніторинг – це важливий інструмент для захисту, контролю та оптимізації всієї системи електропостачання в режимі реального часу. ПЗ для моніторингу надає інформацію, яка допомагає розрізнити неефективність в енергетиці, що є критичним для оптимізації та розв'язання енергетичних проблем бізнесу, підприємства, міста, країни [25].

Точне вимірювання, визначення проблем, розпізнавання неефективності обладнання, контроль реального часу допомагають вимірювання використання енергії та швидко реагувати на незвичайні ситуації. Моніторинг енергосистеми є ключовим для вирішення завдань оцінки безпеки, аналізу непередбачених ситуацій, відновлення контролю та оптимальної експлуатації.

Основні процеси моніторингу включають визначення топології електричного з’єднання мережі, аналіз вимірювальної зв’язності та оцінку миттєвих потоків навантаження в мережі. Головними завданнями контролю енергетики є централізований збір, обробка та відображення загальносистемних даних у реальному часі. Засіб збору даних використовує мікропроцесорні віддалені станції для накопичення і передачі інформації про стан вимикачів та ізоляторів. Потім ці дані обробляються та організовуються в централізованій базі даних [26].

Для розв'язання задач збору та обробки даних використовують програмне забезпечення, яке містить в собі функції управління системою та визначення топології мережі, спостереження за мережею, моделювання мережі та оцінку стану.

Процес визначення топології охоплює формування мережевих груп та вузлів для динамічної та статичної інформації. Спостереження за мережею використовує реєстратори даних, сенсори та технології зв’язку для збору та передачі різних параметрів мережі. Оператори можуть відстежувати стан напруги, струму, активної потужності, а також виявляти будь-які аномалії чи несправності. Оцінка стану мережі проводить аналіз та інтерпретацію зібраних даних для визначення ефективності та безпеки мережі. Застосування методів штучного інтелекту,

математичних моделей та алгоритмів дозволяє визначити можливі ризики, виявити слабкі місця та розробити стратегії для підвищення продуктивності.

Структура електричних мереж може змінюватись залежно від потреб і установ, які споживають електроенергію. Виділяють три основні сегменти споживачів:

- високовольтне підключення для великих об’єктів;
- середньовольтне для об’єктів з середньою потужністю;
- низьковольтне для менших виробництв або комерційних об’єктів.

Мета моніторингу електричних мереж включає забезпечення безперервності постачання, ефективне управління, контроль якості електроенергії та оптимізації витрат. Сегментація споживачів за різними рівнями напруги (висока, середня, низька) враховує особливості підключення та потреби об’єктів.

Для забезпечення безперервності важливо враховувати вимоги різних галузей та підприємств. Високотехнологічні рішення та розподіл навантаження дозволяють уникнути втрат виробництва чи несправності обладнання під час перебоїв.

Якість електроенергії стає все важливішим через появу нових технологій та генераторів гармоніки. Аналіз гармоній та інших параметрів дозволяє вчасно реагувати на проблеми та забезпечувати якість електропостачання. Оптимізація витрат впливає на тарифні плани дистриб’юторів. Централізоване управління та моніторинг дозволяють зменшувати витрати та ефективно використовувати енергію. Використання реєстраторів даних в електричних мережах сприяє моніторингу, технічному обслуговуванню та управлінню системою. Вони виконують функції відображення тривоги, їх обробку та аналіз, а також надають інтелектуальні інформаційні можливості [27]. Розміщення цих технологій в різних вузлах мережі (електростанції, розподільники мережі, промислові об’єкти) забезпечує комплексний моніторинг та управління електричною напругою для різних споживачів енергії.

Моніторинг енергії є дуже важливим елементом для досягнення ефективності балансування енергосистеми, особливо в умовах сучасних енергетичних викликів.

Результатом такого комплексного підходу є система моніторингу енергосистеми, яка надає динамічну інформацію про стан елементів мережі та дозволяє виконувати різноманітні заходи з управління та оптимізації енергоспоживання.

1.6 Опис наявних комплексних систем моніторингу енергетики

Комплексні системи моніторингу енергетики відстежують та аналізують параметри електроенергетичних мереж для ефективного контролю та оптимізації енергопостачання. Ці системи забезпечують постійний моніторинг та автоматичне реагування на зміни, сприяють підвищенню надійності та продуктивності енергетичних систем.

1.6.1 ETAP EMS

ETAP EMS (Energy Management System) – це інтелектуальна система управління енергоспоживанням від компанії ETAP відповідає за оптимізацію споживання енергії, покращення використання системи, підвищення надійності, прогнозування продуктивності електричної системи та зменшення витрат.

Основні складові системи енергомоніторингу та управляння EMS:

1) Система автоматичного керування генерацією (AGC) відповідає за нагляд та регулювання різних рівнів виробництва;
2) Економічна диспетчерська (Economic Dispatch) розподіляє змінний попит на генерацію енергосистеми серед керованих генераторних установок;
3) Значок прогнозування навантаження (Load Forecasting Icon) прогнозує та відстежує завантаження системи на основі алгоритмів, які адаптивно корелюють вхідні змінні (погодні умови, навантаження та інше);

4) Блок підтримки (Unit Commitment) мінімізує витрати операцій та збільшує термін служби енергоблока;
5) Розклад перепадів (Interchange Scheduling) керує графіками електричних транзакцій та диспетчеризація торгів, що виникають в результаті купівлі-продажу енергії;
6) Управління резервами (Reserve Management) відстежує робочу потужність системи та динамічно розраховує баланс між генерацією та прогнозованим навантаженням.

Програмне забезпечення (ПЗ) системи управління енергоспоживанням ETAP EMS користується даними в режимі реального часу. Система моніторить та аналізує параметри частоти, фактичної генерації, потоки навантаження на лініях зв'язку та стан контролерів блоків станції [28]. Це дозволяє забезпечити зміни в системі. В таблиці 1.1 показано основні частини та програмні рішення базового пакета системи ETAP EMS.

Таблиця 1.1 – Програмні рішення ETAP EMS

Назва блоку	Програмні рішення
Потік навантаження (Load Flow), Змінний струм (AC)	Power Flow, 3-Phase & 1-Phase Systems, Voltage Drop, Result Analyzer
Кабельні системи IEEE, ICEA, NEC	Cable Ampacity & Sizing – IEEE, ICEA, NEC
Кабельний стандарт IEC 60502 & 60364	Cable Capacity, Sizing & Shock Protection – IEC 60502 & IEC 60364
ВДЕ	Wind Turbine Generator, PV Array – PV Pro, Controls, Libraries, Profiles
Схеми панелей	Schedules, Calculations, 3-Phase, 1-Phase, NEC, IEC
Базовий модуль	Multi-Dimension Digital-Twin Platform, Revisions, Libraries, Wizards

ПЗ управління енергоспоживанням ETAP EMS покладає на себе обов’язок підтримки частоти в системі розподілу електроенергії та збереження потужності ліній електропостачання на рівні, що наближається до запланованого. Система активно використовує регулювання потужності генераторів для вирівнювання коливань навантаження та оптимізації роботи генераторних установок, забезпечуючи ефективну та стабільну електроенергетичну систему.

Основними проблеми ETAP EMS є складність використання, оскільки внутрішні конфігурації та налаштування інтерфейсу є досить складними, та залежність від технічних засобів. Вартість імплементації висока для невеликих підприємств, тому що охоплює витрати на обладнання, ПЗ та навчання персоналу. Проблеми з безпекою можуть бути критичними через недостатній рівень захисту на всіх рівнях системи. Також система не працює з датчиками та сенсорами на пряму, а використовує сервіси зберігання даних.

1.6.2 Inavitas Platform

Inavitas Platform від компанії inavitas є передовим рішенням для енергетичного моніторингу, що дозволяє відстежувати та контролювати енергію та бути на крок попереду [29]. Inavitas є лідером на ринку комунальних послуг із понад 20 клієнтами-розповсюджувачами. Контролює понад 1,3 ГВт відновлюваних джерел енергії та має великі філії в Австралії, Індії, Туреччині та Європі.

Метою Inavitas є підвищення ефективності енергетичних систем та зменшення витрат на енергію для комерційних, малих і середніх підприємств за допомогою безперервного моніторингу, попередньої аналітики, розумних попереджень, звітів та дистанційного керування навантаженнями. Також система дозволяє порівнювати тарифи, плани, навантаження. На рисунку 1.5 показано роботу системи Inavitas [30].

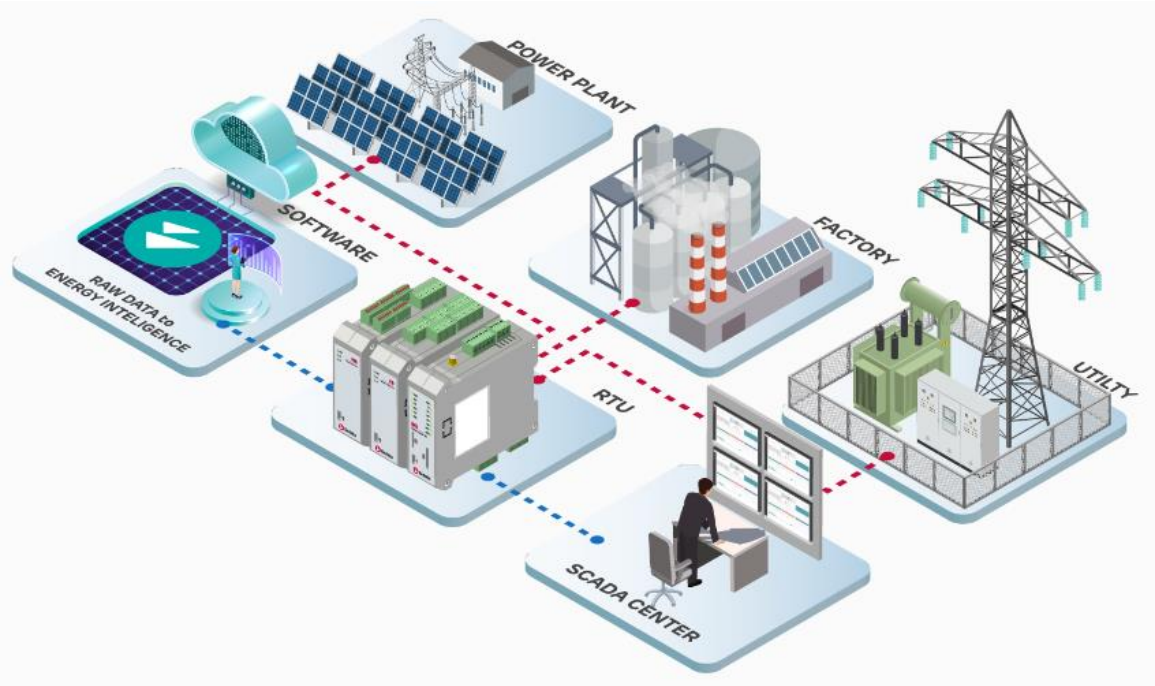

Рисунок 1.5 – Робота платформи Inavitas

Програмний пакет SCADA (Supervisory Control And Data Acquisition) забезпечує роботу системи збирання в реальному часі, обробку, відображення та архівування інформації енергомережі. Система управління мобільною робочою силою M-WFM працює повністю на карті з оперативними даними про мережу. Платформа моніторингу та аналізу якості електроенергії повністю відповідає стандартам IEC та IEEE, та повністю незалежна від постачальника. Колекція додатків DMS виконують ефективний та надійний моніторинг та управління всією розподільчою мережею.

Платформа має інтелектуальні пристрої та інтелектуальне програмне забезпечення inavitas – це економічно ефективні рішення для підвищення ефективності та відмовостійкості. Модуль inavitas Outage Management System покращує стан системи та працює з перебоями в електропостачанні. Розумний моніторинг та управління покращує хостингову потужність мережі. Інтеграція в режимі реального часу з географічною інформаційною системою (GIS), клієнтською інформаційною системою (CIS) та системою управління персоналом (WFM) [30].

Мінусами Inavitas Platform є висока складність та заплутаність системи

моніторингу, потреба у вдосконаленні для виробничих підприємств, дуже незручний інтерфейс звітності та проблеми з інтеграцією різних джерел даних. Головною проблемою є стабілізація платформи Inavitas, тому що деякі функції починають ламатись та працювати неправильно.

1.6.3 Quickbase

Quickbase – це комплексне рішення для моніторингу та управління енергоспоживанням. Воно легко налаштовується та швидко масштабує операції енергосистеми. Quickbase працює з бізнесом та підприємствами, відстежує партнерів та конкурентів, спрощує складні енергетичні процеси в режимі реального часу. Система моніторингу та управляння є високу продуктивність, об’єднує розрізнені системи та має безліч рішень для різних етапів управління енергоспоживанням.

Quickbase виділяється низькою ціною для такого роду послуг і функцій. ПЗ є надзвичайно гнучким та має багато можливостей для налаштування. Команда підтримки Quickbase швидко реагує та проводить онлайн тренінги, де показує як вирішувати базові проблеми, як працюють різні інструмент [31]. Генерація звітів, фільтрація результатів, функція пошуку дуже прості та адаптивні, що допомагає динамічній роботі.

Quickbase має багато рішень для управління проєктами, відстеження та розподілу ресурсів, організації критичних процесів та робочих потоків, адміністрування закладених політик і процедур, інтеграція та встановлення продуктів та послуг.

Недоліками Quickbase є неможливість інтеграції достатньої кількості додатків та інструментів, обмеження на кількість додаткових послуг. Проблеми з надійністю та безпекою є значними. Існує складність розпізнавання помилок та можливості втрати даних. Складний та неінтуїтивний інтерфейс викликає

проблеми з розумінням всіх інструментів. Quickbase має дуже перенасичену різними сервісами, можливостями кастомізації, непотрібними метриками, значками та функціями платформу, що ускладнює швидкий моніторинг та аналіз даних.

Quickbase не дуже підходить для малих бізнесів та підприємств, що пов’язані з моніторингом та управлінням енергетикою.

1.6.4 EnergyCAP

EnergyCAP – це провідне програмне забезпечення у сфері енергетики та сталого розвитку. ПЗ відстежує та аналізує використання енергії, витрати та викиди вуглецю, надаючи комплексну звітність та аналіз для прийняття обґрунтованих рішень. Завдяки надійному управлінню рахунками за комунальні послуги та функції Bill CAPture ефективно проводиться аналіз використання комунальних послуг, витрат та викидів.

EnergyCAP візуалізує щомісячні дані про комунальні послуги та забезпечує порівняльний аналіз складових енергосистеми, аналітику, моніторинг та оцінку проєктів, а також безперешкодну інтеграцію з ENERGY STAR Portfolio Manager, який зберігає та курує інформацію користувача. Серед користувачів EnergyCAP можна виділити постачальників комунальних послуг, урядові структури, комерційні установи та вищі навчальні заклади.

Це комплексне рішення дозволяє менеджерам у сфері енергетики та сталого розвитку ідентифікувати неефективність об’єктів, переглядати рейтинги ENERGY STAR, оцінювати вплив проєктів з енергозбереження та здійснювати інші функції моніторингу енергосистем. Крім того, в складі EnergyCAP існує інформаційна панель, де менеджери можуть обмінюватися даними та відстежувати їх динаміку. На рисунку 1.6 показано інтерфейс EnergyCAP з інтерактивними графіками та візуалізацією даних [32].

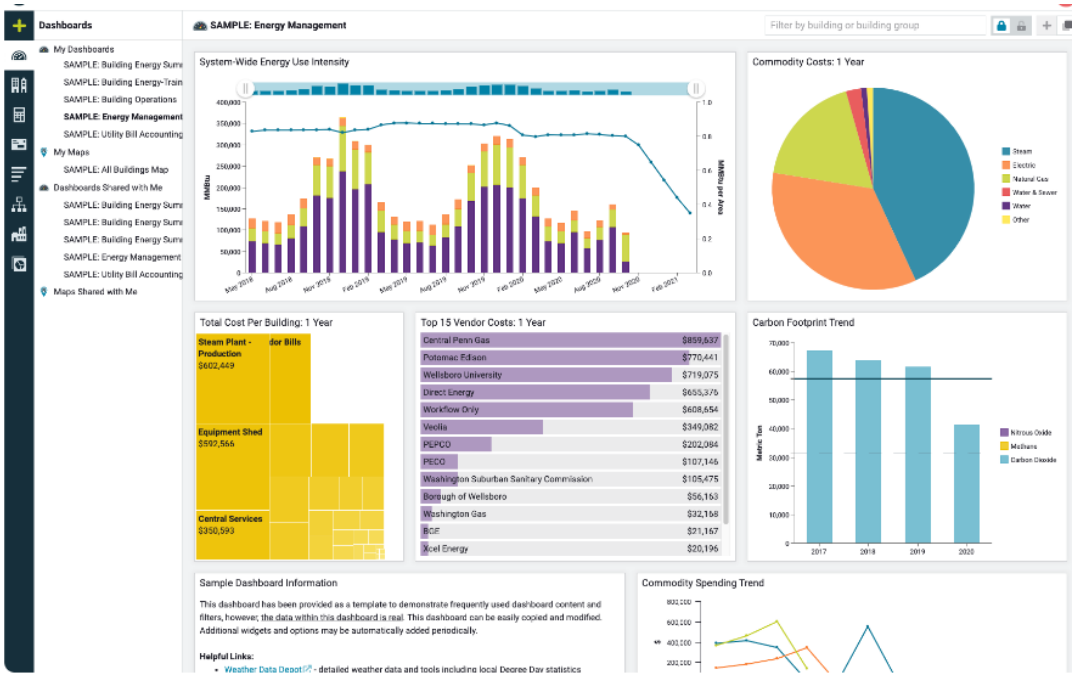

Рисунок 1.6 – Інтерфейс EnergyCAP

EnergyCAP спрощує процес обліку комунальних платежів, зменшуючи необхідність ручного введення даних, використання складних електронних таблиць та інших застарілих методів. Це рішення забезпечує організаціям автоматизований доступ до введення рахунків, звітів та інструментів для проведення аудитів.

Оплата за використання EnergyCAP здійснюється на основі лічильника, включаючи ліцензійну плату, яка розраховується щорічно. Доступні індивідуальні пакети враховують потреби кожної організації. Можливість вибору між локальними та хмарними рішеннями дозволяє вибрати оптимальний варіант відповідно до вимог та вподобань.

EnergyCAP дозволяє вимірювати продуктивність на уніфікованій основі, автоматично нормалізувати дані з урахуванням погодних змін та заносити їх час. Інтелектуальні алгоритми можуть приймати обґрунтовані рішення на основі даних та фокусуватися на важливих проєктах. EnergyCAP містить такі функції як нормалізація, календарізація, порівняльний аналіз та визначення інтенсивності використання енергії.

Користувач має можливість отримувати інформацію на потрібному рівні,

деталізувати дані на рівнях регіону, підрозділу, будівлі, лічильника та сублічильники, та відстежувати потрібні метрики.

Зручна можливість відстежувати успіх енергетичної програми дозволяє миттєво оцінювати результати енергозберігаючих заходів. EnergyCAP допомагає встановлювати базові показники, вимірювати успіх програми, показує економію та ділитися результатами.

Інструмент порівняльного аналізу EnergyCAP Utility Management дозволяє порівнювати ефективність будівель та лічильників, враховуючи витрати, використання енергії, піковий попит та інші показники. Можна використовувати автоматичні або ручні групи, користувацькі будівельні показники для організації та порівняння даних [33].

Функція відстеження проєктів допомагає звітувати про економію енергії та витрати проєкту з енергоменеджменту, надаючи деталізовану інформацію про типи проєктів, дати їх реалізації, вартість та очікувану/фактичну економію. Функція автоматично відображає проєкти на графіку EnergyCAP «Інтенсивність використання енергії», щоб легко спостерігати за їх впливом на споживання енергії.

Проблемою EnergyCAP є застаріла та неінтуїтивна електронна таблиця, через що доводиться використовувати сторонні додатки. Інтеграція зі сторонніми постачальниками занадто складна і займає досить багато часу. Інтерфейс сайту зроблений на Flash, що є застарілою технологією, від якої більшість компаній-розробників ПЗ відмовились. Дуже велика кількістю звітів і налаштувань фільтрів ускладнює швидке налаштування.

2. ОПИС ТЕХНОЛОГІЙ СИСТЕМИ МОНІТОРИНГУ ЗЕЛЕНОЇ ЕНЕРГЕТИКИ

Програмний комплекс системи моніторингу зеленої енергетики для балансування енергосистеми включає наступні компоненти:

- Протокол MQTT. Використовується для забезпечення комунікації між датчиками та іншими пристроями системи. Дозволяє передавати дані в реальному часі з датчиків до системи моніторингу;
- Eclipse Mosquitto. MQTT брокер, який обробляє та маршрутизує повідомлення. Забезпечує основну інфраструктуру для обміну даними через MQTT;
- MQTT-клієнт Eclipse Paho Java Client. Використовується для розробки MQTT-клієнтів мовою програмування Java та середовищі розробки Eclipse. Дозволяє взаємодіяти з MQTT брокером та емулювати запити від датчиків;
- Maven. Інструмент для управління залежностями та побудови проєктів мовою програмування Java. Забезпечує структуру проєкту та легке управління залежностями;
- Node-RED. Візуальна програмна оболонка для побудови потоків обробки даних. Дозволяє легко налаштовувати обробку та передачу даних в системі моніторингу;
- Prometheus. Система моніторингу та зберігання метрик. Збирає, зберігає та дозволяє аналізувати дані про зелену енергетику.
- TSDB – база даних часових рядів. База даних призначена для зберігання часових рядів. Використовується для ефективного зберігання та опрацювання метрик часових рядів;
- Grafana. Інструмент візуалізації та створення графіків на основі даних з Prometheus. Надає зручну візуалізацію для створення та перегляду графіків та панелей моніторингу;

- Spring Boot. Фреймворк для розробки вебдодатків мовою програмування Java. Використовується для реалізації RESTful API та створення захищеного вебінтерфейсу;
- HTML, CSS, JavaScript. Технології для створення вебінтерфейсу. Використовуються для створення користувацького інтерфейсу для візуалізації даних та взаємодії з системою моніторингу.

2.1 Протокол MQTT

2.1.1 Опис протоколу MQTT

MQTT (Message Queuing Telemetry Transport) – це протокол обміну даними, реалізований за принципом видавець/підписник (pub/sub). Вперше його опублікували Енді Стенфорд-Кларк з IBM та Арлен Ніппер з Cirrus Link у 1999 році. Основною метою було забезпечити комунікацію між пристроями в мережах із нестабільним підключенням чи обмеженою пропускною здатністю. Його основними характеристиками є легкість, відкритість та простота реалізації, що робить його ідеальним для використання в умовах контексту «машина-машина» (M2M) та «Інтернету речей» (IoT), де обмежений обсяг коду та пропускна здатність є ключовими параметрами.

MQTT є протоколом обміну повідомленнями, спроєктованим для простої реалізації, переважно на стороні клієнта. Він використовується для забезпечення зв’язку між пристроями в умовах обмеженої потужності. Протокол базується на принципах публікації/підписки, де відправників повідомлення називають видавцями, а отримувачів – підписниками.

Основні поняття протоколу включають брокера, який виступає посередником між видавцями та підписниками, та взаємодію між ними за допомогою MQTT.

Видавці відправляють дані, підписники отримують їх через брокера. Цей протокол є ідеальним для обміну повідомленнями в умовах обмеженої потужності, а також для мереж із низькою пропускною здатністю, великою затримкою або нестабільною роботою [34]. Тому в режимі публікації/підписки клієнти не спілкуються безпосередньо один з одним, а взаємодіють через брокера. Цей підхід робить MQTT ідеальним для IoT пристроїв та інфраструктури.

2.1.2 Функціональність MQTT

Основні властивості протоколу MQTT:

- Асинхронність, яка дозволяє асинхронний обмін повідомленнями між клієнтом та брокером;
- Компактність повідомлень, яка використовує ефективний формат даних для передачі повідомлень у компактній формі;
- Робота в умовах нестабільного зв’язку, яка використовує протокол роботи в умовах непостійного зв’язку на лінії передачі даних;
- Підтримка різних рівнів якості обслуговування (QoS), яка надає можливість вибору рівня гарантованої якості доставлення повідомлень.
- Легка інтеграція нових пристроїв, яка робить простіше вставку та інтеграцію нових пристроїв у мережу.

Протокол MQTT працює на рівні додатків через TCP/IP. Обмін повідомленнями відбувається між клієнтом, який може бути видавцем або підписником, та брокером повідомлень. На рисунку 2.1 показано схему взаємодії між видавцем (publisher), брокером (broker) і підписником (publisher) [35].

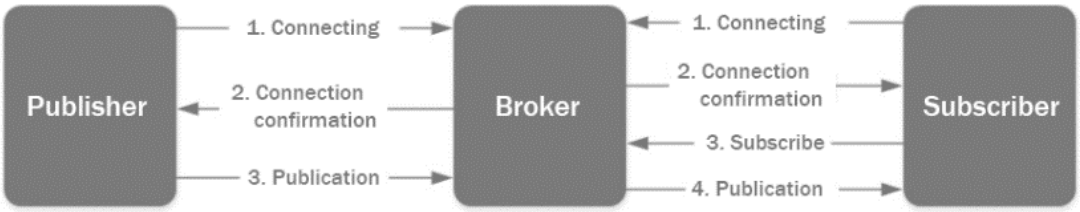

Рисунок 2.1 – Взаємодія видавця, брокера і підписника

Видавець надсилає повідомлення з конкретною темою брокеру MQTT. Підписники можуть отримувати різні дані від різних видавців, залежно від їхньої підписки на відповідні теми.

Протокол використовує різні типи повідомлень, такі як Connect (встановлення з'єднання), Disconnect (від'єднання), Publish (опублікування даних), Subscribe (підписка на тему) та Unsubscribe (відписка від теми), для взаємодії між пристроями та брокером.

Структура повідомлення MQTT включає наступні компоненти:

- Фіксований заголовок;
- Змінний заголовок для певних типів повідомлень;
- Дані.

Така структура робить повідомлення зрозумілим та легшим.

2.1.3 Використання протоколу MQTT

Використання MQTT протоколу може бути в ситуаціях, коли необхідно забезпечити ефективний обмін повідомленнями в умовах обмежених ресурсів, вбудованих систем з обмеженою обчислювальною потужністю та пам'яттю, або системи, що працюють в умовах ненадійних мереж. Цей протокол забезпечує надійні функції обміну повідомленнями, які є критичними для взаємодії з

віддаленими системами та пристроями, при цьому зберігаючи мінімальну частину пропускної здатності мережі.

Комунальні компанії встановлює «розумні» лічильники з використанням MQTT в будинках клієнтів для дистанційного контролю за споживанням електроенергії. Це дозволяє компанії мінімізувати навантаження мережі передачі даних та раціоналізувати витрати під час виробництво електроенергії. Також використовується віртуальна електростанція (VPP), яка дозволяє прогнозувати та знижувати загальний попит на електроенергію, регулювати електроприлади в установах споживачів.

Сфера соціальних мереж вимагає ефективний механізм для надсилання повідомлень. Оскільки наявний метод є надійним, але повільним, через що відбуваються затримки. Для подолання цих обмежень розробники почали використовувати протокол MQTT. Це дозволило забезпечити постійне з’єднання з серверами обміну повідомленнями, маршрутизувати повідомлення через системи управління. В результаті компанії досягли миттєвого доставлення повідомлень за кілька сотень мілісекунд, замість кількох секунд.

Медична організація використовує MQTT для домашнього моніторингу кардіостимуляторів. Це рішення дозволяє вести моніторинг пацієнтів після виписки з лікарні, підвищити ефективність оглядів та відповідність новим стандартам збору даних [36]. Також протокол використовується для віддаленого відстеження життєво важливих показників пацієнтів з різними групами ризику.

Використання MQTT можливе у торгівлі для надсилання інформації про продажі та отримання оновлень цін через повільні мережі. MQTT може бути використаний для автоматизованого збору в реальному часі діагностичної інформації з автомобільних компонентів під час профілактичного обслуговування автомобілів. Важливим протокол є для відстеження навколишнього середовища за допомогою дистанційних датчиків, та забезпечення негайного аналізу даних в реальному часі.

Ці сценарії демонструють успішне впровадження MQTT в різних областях, від охорони здоров’я до енергетики та комунальних послуг.

2.2 Eclipse

Eclipse – це інтегроване середовище розробки (IDE) для Java. Також це інструментальний фреймворк, що використовується для розробки різноманітних інструментів, які не обов'язково пов'язані з програмуванням.

Платформа Eclipse має розширюваність завдяки реалізації OSGi R4, що дозволяє створювати плагіни, які можуть надавати різноманітні функції для будь-яких потреб.

Функціональні можливості платформи Eclipse включають:

- Редактори, спеціалізовані для конкретних мов та представлень;
- Підтримка процесу рефакторингу;
- Вбудовані інструменти для модульного тестування та налагодження;
- Інкрементальна компіляція та збірка коду;
- Підтримка командної розробки;
- Вбудована підтримка CVS (Concurrent Versions System).

Eclipse має різні фреймворком для розробки на інших мовах програмування, таких як Java, C/C++, PHP та інші.

Спільнота Eclipse активно використовує можливості фреймворку для створення інтегрованих середовищ розробки та різноманітних інструментів. Існують різноманітні проєкти та підпроєкти, спрямовані на створення загальновживаних фреймворків та API для подальшого використання в програмному забезпеченні.

Екосистема Eclipse включає понад 140 компаній-членів, що надають ресурси для проєктів та використовують їх результати для комерційних програмних продуктів. Eclipse Foundation, створена у 2004 році, відповідає за управління та розвиток проєктів з відкритим вихідним кодом, надаючи IT-інфраструктуру для команд розробників [37].

Eclipse використовує плагіни для реалізації своєї функціональності за допомогою системи виконання, що базується на Equinox, яке реалізує специфікацію OSGi. Всі компоненти Eclipse реалізовані у вигляді плагінів, що дозволяє інтегрувати різноманітні додатки. Eclipse надає плагіни для різноманітних можливостей, включаючи підтримку мови моделювання UML, DB Explorer та інші. Eclipse SDK включає засоби розробки Eclipse Java (JDT), що дозволяє використовувати рефакторинг та аналіз коду.

2.3 Eclipse Mosquitto

Eclipse Mosquitto – це брокер повідомлень з відкритим вихідним кодом, який реалізує протокол MQTT. Mosquitto важить небагато, тому підходить для використання на всіх пристроях, від малопотужних одноплатних комп'ютерів до повноцінних серверів. Протокол MQTT легко здійснює обмін повідомленнями за моделлю публікації/підписки. На рисунку 2.2 схематично зображено можливості брокеру з'єднуватися за допомогою концепції публікація (publish), що відповідає сенсору, та підписка (subscribe), що є кінцевим користувачем [38].

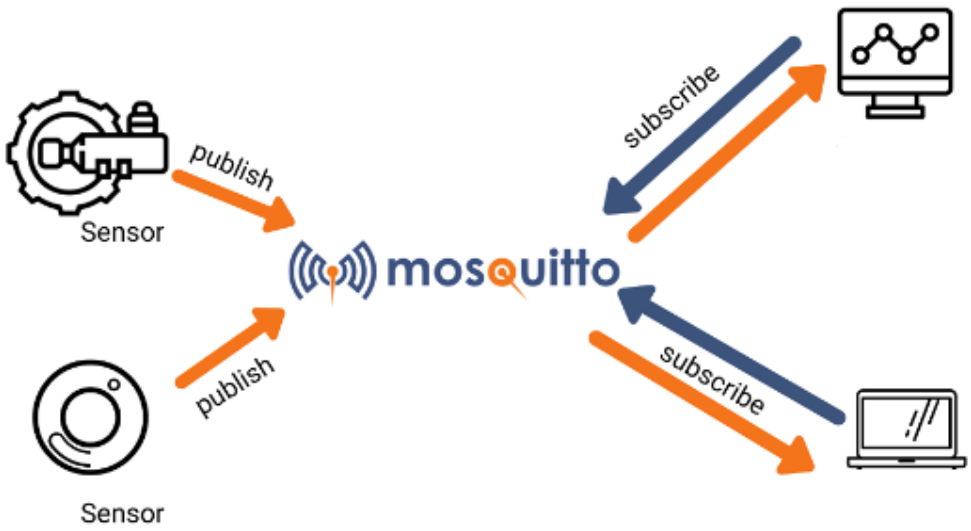

Рисунок 2.2 – Схема роботи Mosquitto за моделлю публікації/підписки

Це робить MQTT придатним для обміну повідомленнями в Інтернеті речей, зокрема з малопотужними датчиками або портативними пристроями, такими як смартфони, вбудовані комп'ютери або мікроконтролери [39]. Mosquitto є частиною Eclipse Foundation та містить бібліотеки для реалізації клієнтів.

Eclipse Mosquitto надає просту серверну реалізацію протоколу MQTT, тому сенсори, датчиків та виконавчих механізмів, що використовують MQTT для обміну повідомленнями, мають можливість працювати на невеликих та енергоефективних пристроях. Mosquitto може бути використаний на вбудованих пристроях або мікропроцесорів, до яких вони підключені.

У реалізації Mosquitto виконуваний файл має розмір приблизно 120 кБ та використовує приблизно 3 МБ оперативної пам'яті при наявності 1000 підключених клієнтів. Тестування засвідчили успіх з 100 000 підключеними клієнтами при помірній швидкості обміну повідомленнями.

Mosquitto не обмежується прийняттям з'єднань від клієнтських додатків MQTT, а може підключатися до інших MQTT-серверів, які мають інші екземпляри Mosquitto. Ця технологія можлива за допомогою налаштування конфігурації мостів. Це дає можливість будувати мережі MQTT-серверів і передавати повідомлення з будь-якого місця в мережі до будь-якого іншого.

Mosquitto є ключовим компонентом проєкту Eclipse Streamsheets, який надає простий інтерфейс, схожий на електронну таблицю в реальному часі. Це дозволяє обробляти вхідні дані з MQTT, OPC-UA, REST та інших протоколів для створення інформаційних панелей або управління процесами [40].

Однією з важливих характеристик Mosquitto MQTT є його легка конструкція. Mosquitto вимагає мінімальних системних ресурсів, що робить його ідеальним для обмежених пристроїв, таких як датчики та мікроконтролери. Брокер добре працює в умовах обмеженого мережевого підключення та може працювати на різних операційних системах, включаючи Linux, Windows, macOS, Raspberry Pi та інші вбудовані системи. Мостові з'єднання дозволяють брокерам Mosquitto з'єднуватися з іншими MQTT-брокерами, об'єднуючи дані з різних мереж.

Функцію зберігання повідомлень дає можливість уникнення втрати даних при тимчасовому відключенні пристроїв.

Недоліки та обмеження Mosquitto MQTT [41]:

- Обмежена інтеграція даних, що ускладнює з'єднання з інструментами обробки даних, реляційними або нереляційними базами даних;
- Обмежена підтримка хмарних технологій, що не дозволяє користуватися хмарними сервісами AWS, Azure, Google Cloud та інші;
- Відсутність вбудованого графічного вебінтерфейсу, що ускладнює керування брокерами та темами MQTT через Інтернет;
- Обмежені функції безпеки, що унеможливлює використання розширених механізмів безпеки та контролю доступу на основі ролей (RBAC);
- Відсутність вбудованих функцій кластеризації або резервування, що може ускладнити забезпечення високої доступності;
- Механізм обмеженої стійкості та збереження даних, що може бути непридатним для великомасштабних розгортань IoT.

Проте недоліків Mosquitto MQTT в рази менше за переваг.

2.4 Eclipse Paho Java Client

Eclipse Paho Java Client являє собою бібліотеку MQTT для Java на платформі Eclipse, розроблену для створення додатків. Проєкти виконуються на JVM або інших платформах, які є сумісними з Java. Цей клієнт надає два API: MqttAsyncClient та MqttClient. MqttAsyncClient пропонує повністю асинхронний підхід, де про завершення дій сповіщається через зареєстровані зворотні виклики. MqttClient є синхронною обгорткою для MqttAsyncClient, де функції відображаються синхронно.

Проєкт Paho створений для надання надійних реалізацій відкритих і стандартних протоколів обміну повідомленнями з відкритим вихідним кодом. Цей проєкт призначений для розвитку нових, наявних та майбутніх додатків для обміну даними між машинами (M2M) та Інтернетом речей (IoT) [42].

Paho може враховувати фізичні та вартісні обмеження, що властиві пристроям. Його завдання включає забезпечення ефективного рівня з'єднання між пристроями та додатками з метою масштабування ПЗ та росту різних можливостей для додатків в Інтернеті або на підприємствах.

Особливості Eclipse Paho Java Client:

- Підтримка MQTT 3.1, MQTT 3.1.1 та MQTT 5.0;
- Використання LWT (Last Will and Testament);
- Підтримка SSL/TLS;
- Збереження повідомлень;
- Автоматичне підключення після втрати з'єднання;
- Буферизація офлайн-повідомлень;
- Підтримка WebSocket;
- Стандартна підтримка TCP;
- Деблокуючий та блокуючий API;
- Висока доступність.

Тому бібліотека Eclipse Paho Java Client часто застосовується у розробці додатків для взаємодії між пристроями в інтернеті речей та машинному спілкуванні.

2.5 Maven

2.5.1 Опис Maven

Maven є широко використовуваним інструментом від вебсервісу Apache для автоматизації процесу збірки в основному для Java проєктів. Maven визначає процедуру збірки програми та визначає її залежності. За допомогою конвертацій цей інструмент дозволяє описати проєкт у форматі XML, включаючи інформацію про його залежності від інших зовнішніх модулів та компонентів, порядок збірки, каталоги та обов'язкові плагіни. З часу випуску версії Maven 1.0 у 2004 році, цей інструмент завоював популярність і на сьогодні є невід'ємною частиною багатьох відкритих та комерційних проєктів [43].

Під час створення проєкту Maven використовується для виконання конкретних завдань, таких як компіляція коду та його пакування. Однією з його особливостей є динамічне завантаження бібліотек Java та плагінів Maven з репозиторіїв, таких як Центральний репозиторій Maven, і їх локальне збереження.

Файли об'єктної моделі проєкту (POM) визначають конфігураційну інформацію, залежності, вихідний каталог, плагіни, цілі тощо. Maven використовує ці файли для створення проєкту, читаючи їх при виконанні заданих команд.

Залежності – це зовнішні бібліотеки Java, необхідні для проєкту. Також вони є каталогами, де зберігаються запаковані JAR-файли. Maven шукає залежності у локальному сховищі, а якщо їх там немає, завантажує з центрального сховища Maven.

Профілі збірки дозволяють збирати проєкт з різними конфігураціями. Додавши профілі до POM-файлу, можна активувати різні конфігурації для збірки проєкту [44]. Плагіни збірки використовуються для виконання конкретних завдань, які потрібні користувачу. Maven має стандартні плагіни, а також можливість реалізації власних плагінів мовою програмування Java.

2.5.2 Архітектура та життєвий цикл Maven

Maven є потужним інструментом для створення артефактів java-проєктів, використовуючи конфігураційний файл «pom.xml». У Apache Maven життєвий цикл збірки визначається наперед і включає набір цілей, які виконуються в певному порядку для збірки та управління проєктом. Кожен етап представляє різні життєві цикли.

Чистий життєвий цикл Maven очищає всі залежності, які раніше зберігалися в локальному сховищі та використовувалися для інших проєктів.

Життєвий цикл за замовчуванням включає стандартний набір команд, які визначають послідовність дій [44].

В таблиці 2.1 подано набір команд життєвого циклу за замовчуванням та їх призначення.

Таблиця 2.1 – Команди життєвого циклу за замовчуванням Maven

Назва команди	Призначення
validate	Перевіряє конфігурацію проєкту
compile	Компілює вихідний код у байт-код
test	Запускає тести для проєкту
package	Пакує скомпільований код і ресурси в артефакт (наприклад, JAR, WAR)
install	Встановлює артефакт до локального сховища
deploy	Копіює артефакт до віддаленого сховища

Ці життєві цикли можна викликати за допомогою різних команд Maven, наприклад, «mvn clean:clean» або «mvn compile». Також можна виконати весь життєвий цикл за допомогою команди «mvn clean install».

Архітектура Maven включає плагіни, ресурси, залежності, файли коду та

взаємодію між репозиторіями. На рисунку 2.3 показано архітектуру Apache Maven [45].

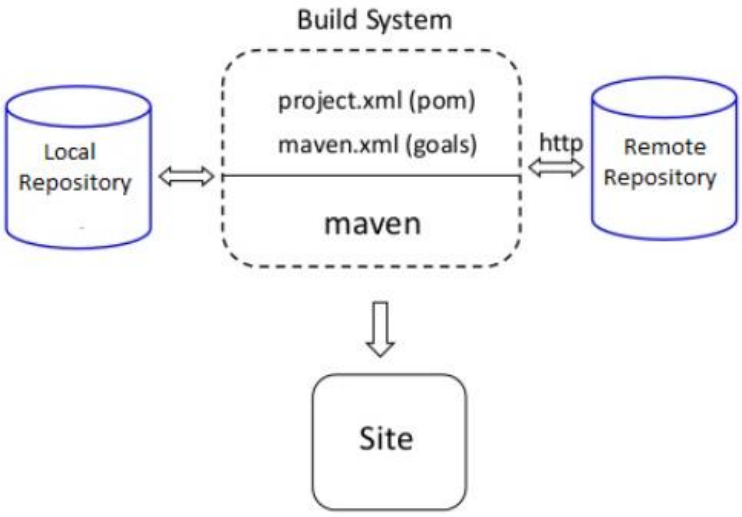

Рисунок 2.3 – Архітектура Apache Maven

Maven взаємодіє з файлом «pom.xml», де визначаються залежності. Він завантажує ці залежності до локального репозиторію (local repository) з центрального або віддаленого репозиторію (remote repository). Maven виконує життєві цикли, фази, цілі та плагіни, які визначені у файлі «pom.xml».

Maven є незамінним інструментом для ряду завдань у розробці проєктів. Використовуючи Maven, можна легко здійснювати збірку проєкту, додавати бібліотеки та інші залежності, отримувати вичерпну інформацію про проєкт, лог-файли та звіти про юніт-тести. Також Maven виявляється надзвичайно корисним при оновленні центрального репозиторію JAR та інших залежностей. Можна створювати різноманітні проєкти з різними вихідними типами (JAR, WAR тощо), інтегрувати їх з системами контролю версій (Subversion, Git).

Maven допомагає ефективно управляти життєвим циклом збірки проєкту, забезпечувати стандартну структуру проєкту та швидкий пошук необхідних файлів. Не менш важливою є його підтримка багатомодульних проєктів, що дозволяє ефективно працювати над кількома пов’язаними проєктами та управляти їхніми залежностями.

2.6 Node-RED

Node-RED є вебпрограмою потокового програмування, яка розроблена компанією IBM Emerging Technology Services. Зараз входить до складу команди OpenJS Foundation. Потокове програмування започатковане в 1970-х роках двадцятого століття. Воно описує метод проєктування комп'ютерних програм та новий архітектурний підхід, що є відмінним від традиційних структурних методів проєктування. У цьому методі проєктування дані обробляються незалежними елементами та взаємодіють один з одним та зі зовнішніми сервісами за допомогою системи обміну. У Node-RED такі елементи є вузлами, що виконують запрограмовану дію з отриманими даними та передають їх далі. Ця модель добре піддається візуальному представленню, тому є доступною для різних користувачів.

Група IBM Emerging Technology Services почала створювати Node-RED у 2013 році як побічний проєкт. Початково інструмент був розроблений для візуалізації та маніпулювання даними між темами MQTT, проте швидко став більш загальним інструментом для роботи з потоками даних. З вересня того року його виклали у відкритий доступ, а у 2019 році він став частиною OpenJS Foundation.

Node-RED складається з середовища виконання на основі Node.js. Запускається Node-RED на локальному комп'ютері або віддаленому сервері, доступ можна отримати через веббраузер. Для цього потрібно ввести назву або номер сервера. На локальних комп'ютерах найчастіше це сервер під номером 1880.

У вебінтерфейсі користувачі можуть перетягувати вузли із загальної панелі в робочу область і з'єднують їх між собою. Так утворюються потоки даних. Одним натисканням миші створений додаток розгортається для виконання в середовищі Node.js [46]. Кожен вузол можна розширювати, додаючи до нього нові вузли.

Можливості Node-RED включають браузерний редактор потоків, який дозволяє легко налаштовувати потоки за допомогою бібліотек та налаштувань вузлів. Вбудовані бібліотеки мають корисні функції, шаблони та потоки для подальшого використання. У репозиторії Node налічується понад 2 000 модулів та

бібліотек, тому можна легко розширити функціонал вузлів, додавши нові можливості. Потоки можуть бути швидко розгорнуті для виконання за допомогою одного клацання миші. Крім того, можна створювати функції, методи, маніпуляції з даними мовою програмування JavaScript у текстовому редакторі. Це дає можливість налаштовувати вузли та управляти потоками. Самі потоки можна легко експортувати як файли JSON. Тому Node-RED є ідеальним для роботи на доступних пристроях і в хмарному середовищі [47].

2.7 Prometheus

2.7.1 Опис Prometheus

Prometheus є відкритою системою моніторингу, що заснована на метриках, від команди Cloud Native Computing Foundation, та написана мовою програмування Go. Ця система є простою та ефективною для зберігання та аналізу метрик з отриманих даних від баз даних (БД), різних програм та датчиків. Prometheus має модель даних (data model) і мову запитів (query language), які здійснюють повний аналіз роботи додатків та інфраструктури проєкту.

Prometheus реєструє та зберігає свої метрики у вигляді часових рядів. Інформація про метрики містить час їх фіксації, а також, по можливості, додаткові пари ключ-значення, які називаються мітками (label). Мітки використовуються для відокремлення отриманих даних. Також ця система моніторингу використовує досить простий текстовий формат для передачі метрик, що полегшує взаємодію з іншими моніторинговими системами. Модель даних і мова запитів PromQL дозволяють агрегувати дані за різними параметрами [48].

Prometheus підтримує багато мов програмування (Go, Java, C#, Python, Ruby та інших) та середовищ виконання (JVM, .Net, Node.js та інші). Завдяки широкій

можливості експортування Prometheus також взаємодіє з різними системами, БД, базами даних часових рядів (TSDB), сервісами, включаючи Kubernetes, Docker, HAProxy, MySQL, PostgreSQL, InfluxDB, Kafka.

Система Prometheus легко інтегрується в чинні проєкти та додається до інших інструментів з управління конфігурацією, що дозволяє оптимізувати наявні сервіси та ПЗ.

2.7.2 Функціональність та ахітектура Prometheus

Prometheus – дуже потужний інструмент для моніторингу систем, зберігання часових рядів та метрик даних. Він має багато функцій та особливостей, розроблених для збільшення функціонала системи [49]. У таблиці 2.2 показані функції Prometheus та їх характеристика.

Таблиця 2.2 – Функції Prometheus

Функція	Опис функції
Багатовимірна модель даних	Дозволяє управляти даними різними способами, щоб отримати уявлення про продуктивність і стан системи
Збір часових рядів	Збирає дані часових рядів за допомогою pull-моделі через HTTP
Зберігання часових рядів	Всі зібрані метрики зберігаються в TSDB для зручного запиту та аналізу історії даних
Переміщення даних часових рядів	Підтримує передачу даних користувацьких метрик до Prometheus через проміжний шлюз

Автоматичне виявлення цілей моніторингу	Автоматично виявляє і відстежує нові сервіси в міру їх додавання в систему
Візуалізація	Підключає базовий графічний інтерфейс та інтеграція з популярними інструментами візуалізації
Система оповіщення	Запускає оповіщення на основі певних значень метрик або шаблонів, реагує на системні проблеми
Інтеграція	Встановлює просте з’єднання з іншими інструментами та платформами

Ці функції роблять Prometheus потужним і дуже ефективним інструментом моніторингу для хмарних технологій.

Метрики Prometheus використовуються для моніторингу хмарної інфраструктури та відстежування утворюваних проблем. Вони дуже добре сповіщають про стан всієї системи [50]. У таблиці 2.3 наведено опис чотирьох основних типів метрик системи Prometheus: Лічильника, Показника, Зведення та Гістограми.

Таблиця 2.3 – Типи метрик Prometheus

Назва метрики	Опис
Лічильник (Counter)	кумулятивна метрика, яка лише збільшується або обнуляється. Використовується для відстеження величин, які зростають з часом.
Показник (Gauge)	метрика, що представляє одне числове значення, яке може коливатися. Використовується для вимірювання таких величин, як температура або поточне використання пам’яті

Зведення (Summary)	метрика, яка відображає розмір і кількість подій за певний проміжок часу. Використовується для обчислення середніх значень або для вимірювання затримки запитів
Гістограма (Histogram)	метрика, яка робить вибірку та рахунок спостережень. Використовується для розподілу даних, обчислення співвідношення у виборці або тривалості запиті

Ці чотири типи метрик важливі для моніторингу інфраструктури та для розуміння того, що відбувається в конкретному середовищі.

Архітектура Prometheus побудована на основі клієнт-сервер. Сервер збирає та зберігає дані метрик. Клієнти або експортери відповідають за збір та представлення даних метрик. Для збору даних метрик потрібно розгорнути експортери, які збирають дані з додатків, БД або серверів. Збір відбувається через проміжний пуш-шлюз для короткотривалих завдань [51]. На рисунку 2.4 показано схему архітектури Prometheus.

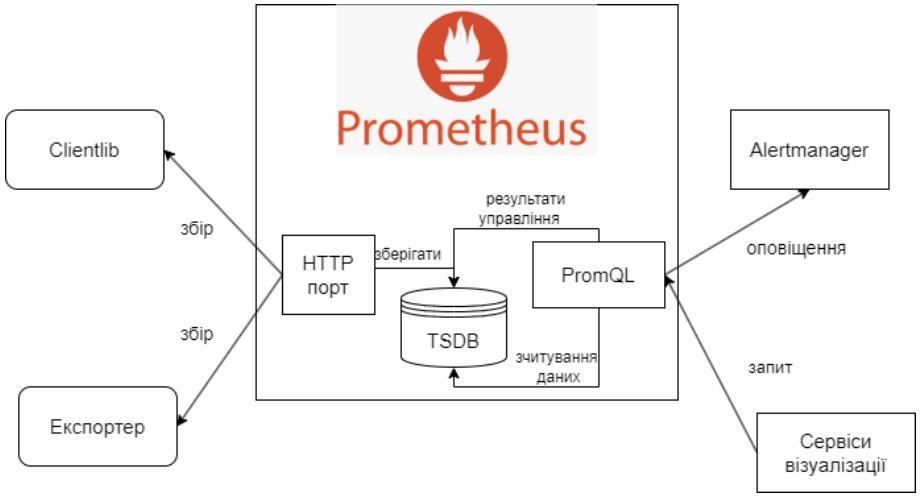

Рисунок 2.4 – Архітектура Prometheus

Prometheus збирає дані метрик за допомогою експортерів або клієнтських

бібліотек (clientlib) через пуш-шлюз або HTTP-порт, якщо дані отримані від зовнішніх серверів. Prometheus зберігає дані локально і може виконувати агрегацію та запис нових даних часових ряди з наявних даних. Мова запитів PromQL допомагає виконувати управління над цими даними, робити виміри, встановлювати мітки, обчислювати середнє значення, суму і відсоток даних метрик. Також за допомогою PromQL відбувається генерація сповіщення, які будуть оброблятися системою оповіщення (Alertmanager). Для візуалізації зібраних даних можна використовувати стороні сервіси візуалізації, такі як Grafana, PromLens, Web UI та інші.

Prometheus часто використовується для моніторингу інфраструктури, різних додатків, Інтернету речей (IoT), безпеки, напруги в енергосистемах та бізнес-показників.

2.8 TSDB

У наш час сенсори, датчики, лічильники, механізми та різні прилади, які підключені до Інтернету, створюють величезний потік даних. IoT і хмарні системи стали викликом для багатьох систем обробки та зберігання даних. Реляційні та нереляційні бази даних не можуть впоратися з таким великим обсягом даних часових рядів, тому системах, які працюють з даними часових рядів.

TSDB або база даних часових рядів використовується як система керування базами даних для зберігання, обробки та аналізу даних часових рядів. Дані часових рядів являють собою послідовність параметрів, які мають мітку часу та індексуються або перераховуються за цією міткою.

TSDB спеціально оптимізовано для швидкого приймання, затримки запитів і стиснення даних часових рядів. Також бази даних часових рядів включають спеціальні аналітичні функції та функції керування даними для простого інтегрування та розробки програм [52].

Бази даних часових рядів використовувалися у фінансовій та переробній промисловості, проте зараз вони стають популярними завдяки стрімкому зростанню підключений пристроїв до Інтернету та розвитку Інтернету речей.

Завдяки можливості зберігати велику кількість показників і аналізувати дані в режимі реального часу TSDB використовують у моніторингу додатків та ПЗ, які пов'язані з контролю та аналізу даних. Також TSDB є корисними під час моніторингу активності мереж та ефективному зберіганню даних з високою деталізацією, що використовується у кібербезпеці та системах контролю мереж [53].

Зараз існує багато баз даних часових рядів через зростання потоку даних часових рядів. На рисунку 2.5 показано графік зростання п'яти наявних найпопулярніших баз даних часових рядів [54].

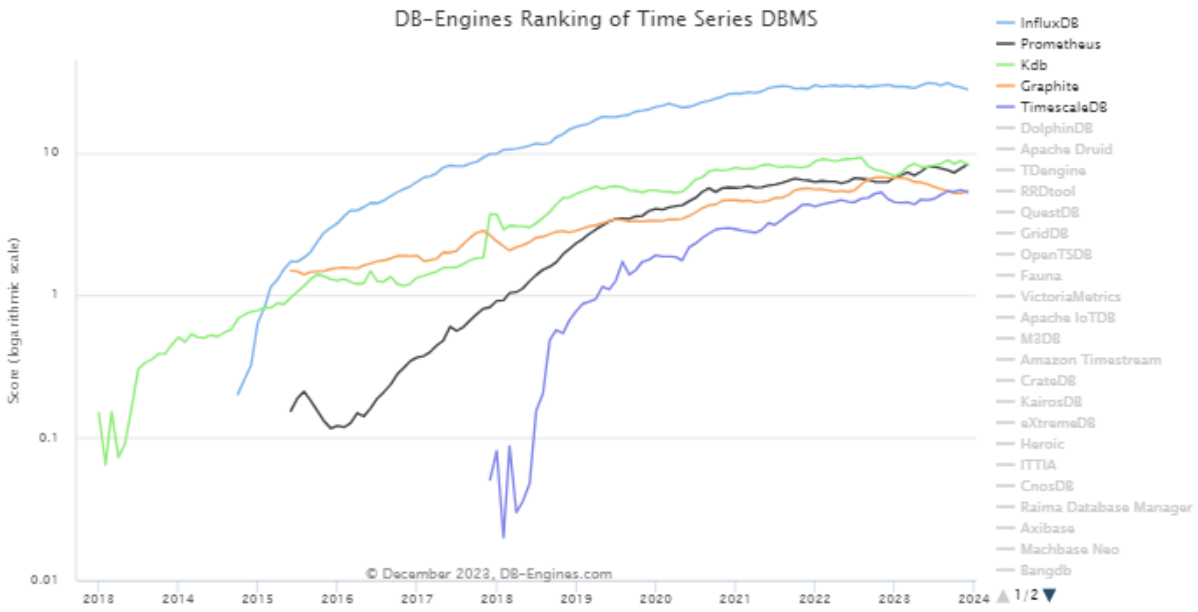

Рисунок 2.5 – Графік популярності наявних TSDB

InfluxDB, Prometheus, Kdb, Graphite, TimescaleDB зараз найпопулярніші бази даних часових рядів. InfluxDB та Prometheus дозволяють працювати з даними під час високого навантаження. TimescaleDB розширює можливості реляційної БД та може працювати в інфраструктурі PostgreSQL.

2.9 Grafana

Grafana – це інтерактивне ПЗ для візуалізації даних з відкритим кодом, яке розроблено командою Grafana Labs. Така платформа створена для перегляду даних за допомогою діаграм, графіків або інших інструментів візуалізації, об’єднаних в інформаційну панель для кращої інтерпретації та розуміння. Воно може запускатися на різних оперативних системах, таких як Windows, MacOS, Gnu або Linux. Підтримує інтеграцію з багатьма ПЗ та сервісами, наприклад, такими як MySQL, PostgreSQL, Graphite, Elasticsearch, OpenTSDB, Prometheus, InfluxDB та інші [55].

За допомогою Grafana можна надсилати запити, встановлювати сповіщення, моніторити показники, вибирати потрібні інтервали та періоди часу для легшого аналізу даних, визначенню тенденції та невідповідностей. Завдяки цьому процеси роботи з даними стають ефективнішими. Тому багато компаній, які працюють з великими потоками даних та потребують швидкий аналіз та формування висновків, використовують сервіс Grafana. Це такі сайти та компанії як Uber, Amazon, PayPal, Stack Overflow та інші.

Інформаційні панелі Grafana надають статистичну інформацію з отриманих даних, які можуть бути зібрані із різних джерел. Такі інформаційні панелі забезпечують плавну візуалізацію та переміщення отриманого аналізу між підрозділами та членами команди для швидкого розуміння та розв'язання проблем [56]. Це дозволяє співпрацювати та більш детально досліджувати дані та їх наслідки.

За допомогою плагінів Grafana та зручного API можна відстежувати дані в режимі реального часу. Зручне API дозволяє підключатися до наявних джерел даних та отримувати показники з різних сервісів, платформ та пристроїв.

Інтерфейс Grafana дуже зручний. Він дозволяє налаштовувати, редагувати, перетворювати та кастомізувати графіки, діаграми та інформаційні панелі. Візуалізація дозволяє узагальнювати, об’єднувати та перетворювати дані для

різних потреб аналізу та моніторингу. Система сповіщення доповнює інтерфейс завдяки можливості створювати, консолідувати та контролювати повідомлення.

Ці функції зробили Grafana дуже потужним та ефективним інструментом для аналізу та моніторингу показників для різних інформаційних систем.

2.10 Spring Boot

Spring Boot - це розширення Spring Framework, яке дозволяє швидко підключати функції програми, вебсервери, БД, сервіси безпеки тощо.

2.10.1 Spring Framework

Spring Framework забезпечує комплексну модель програмування та конфігурації для сучасних корпоративних програм на основі Java на різних платформах [57]. Головною складовою Spring є інфраструктурна підтримка на рівні додатків, тобто Spring зосереджується на внутрішніх процесах корпоративних додатків та загальній бізнес-логіці роботи системи.

Використання Spring Framework полягає у створенні програми або ПЗ мовою програмування Java, що сповідує принцип мінімального впливу на програму. У таблиці 2.4 наведено основні складові системи Spring Framework, які були започатковані зі самого зародження Spring [58].

Таблиця 2.4 – Основні складові Spring Framework

Назва	Опис
Spring Core	Основа Spring, яка містить контейнер Bean і допоміжні утиліти.
Spring Context	Інтерфейс, що надає інформацію про конфігурацію програми.
Spring DAO	Інфраструктура транзакцій, підключення до бази даних Java (JDBC) та підтримка об’єктів доступу до даних (DAO).
Spring ORM	Підтримка зовнішніх систем, таких як Hibernate, iBATIS і Java Data Objects (JDO).
Spring AOP	Сумісність з аспектно-орієнтованим програмуванням (AOP).
Spring Web	Основний функціонал інтеграції, підтримка сервлетів та веборієнтований контекст програми.
Spring Web MVC	Вебплатформа Model-View-Controller (MVC) для побудови вебпрограм.

Дані складові закривають усі потреби користувачів Spring Framework для ефективного написання, підключення, оптимізації та підтримки програм, які написані на Java.

Інверсія управління (IOC) та ін'єкція залежності (DI) є основними функціями Spring Framework для створення ПЗ.

Інверсія управління полягає у тому, що під час розробки ПЗ передається керування об’єктами або частинами програми до контейнера або фреймворку. Це дозволяє відокремлювати виконання завдання від реалізації, впроваджувати перемикання між різними реалізаціями, збільшувати модульність програми та полегшувати тестування.

Ін’єкція залежностей – це такий шаблон, який можна використовувати для

реалізації інверсії управління, де інвертований елемент керування встановлює залежності об’єкта. Це дає можливість з’єднувати об’єкти з іншими об’єктами без написання коду вручну, а за допомогою Spring Framework [59].

Ці функції пришвидшують написання та підтримку програм та ПЗ, перекладаючи частину обов’язків на Spring Framework.

2.10.2 Опис Spring Boot

Spring Boot – це мікрофреймворк з відкритим кодом від Pivotal для скорочення коду та легшого запуску програм. Він розширює Spring Framework та надає більші можливості для налаштування ПЗ. Spring Boot має вбудовані інструменти автоконфігурації та пакети для створення та оптимізації програми, які базуються на розробках сторонніх команд та власних налаштувань. Мікрофреймворк дозволяє уникати помилок під час написання коду, оскільки зменшує шаблонний код.

Spring Boot відрізняється від Spring Framework тим, що Spring Boot автоматично налаштовує усі залежності, які необхідні для запуску програми, використовується для проєктування REST API, надає конфігурацію для залежностей, скорочуючи шаблонний код та містить оперативну інформацію програми.

Перевагами Spring Boot є автоконфігурація, яка завантажує всі потрібні залежності для запуску програми та зменшує потребу в шаблонному коді, направлений підхід, що автоматично завантажує саме необхідні бібліотеки відповідно до потреб програми, наявність початкових додатків, які пришвидшують ініціалізацію програми, та можливість створювати автономні програми, які не залежать від зовнішніх вебсерверів [60].

Отже, Spring Boot є найкращим помічником для швидкого та ефективного написання вебсервісів та додатків.

2.11 HTML

HTML (HyperText Markup Language) – стандартизована мова гіпертекстової розмітки документів для побудови та перегляду вебсторінок в Інтернет браузері. Мова гіпертекстової розмітки являє собою мова тексту, яка може керувати відображення комп'ютерних програм.

HTML використовується як основна мова Інтернету та найпоширеніша мова, яка використовується у вебдизайні або в розробці. Усі вебсторінки написані мовою HTML. HTML визначає вигляд зображення, мультимедії та текст у веббраузерах. Ця мова містить елементи для з'єднання документів та виготовлення інтерактивних вебдокументів. Остання версія HTML5 використовується приблизно 90 відсотками вебсайтів.

HTML використовує теги для визначення простої структури тексту. Елементи та теги визначаються символами «<» та «>». Одні теги, такі як «<body>», « » тощо, розміщують текст, інші теги в середині себе. Теги, наприклад «<p>», «
», змінюють текст або структуру тексту. Теги, такі як «<img», «<a>» та інші, додають об'єкти (зображення, гіперпосилання, відео тощо) безпосередньо на вебсторінку. Інші теги, наприклад «<head>», надають інформацію про тіло документа. HTML документ починається з тегу «<html>».

Конструкції HTML дозволяють вставляти різні об'єкти, такі як зображення, відео та інші інтерактивні елементи. HTML позначає структурну семантику тексту, формує таблиці, заголовки, абзаци, списки, посилання, цитати та інші елементи. Під час створення сторінки HTML, розмітка не відображається користувачу, а тільки готовий текст, зображення та відео. Веббраузер сам знає як організувати та відображати текст та інші елементи [61].

HTML пропонує швидший і надійніший підхід до веброзробки. А з мовами CSS, JavaScript стає дуже потужним інструментом для створення вебпрограм та вебінтерфейсів.

2.12 CSS

CSS (Cascading Style Sheet) або каскадна таблиця стилів (CSS) використовуються для керування зовнішнім виглядом елементів на вебсторінці. CSS користується правилами стилю, щоб показувати браузеру які саме стилі вибрати до певних елементів.

Ідея CSS була запропонована Хокон Віум Лі у 1994 році, через кілька років після створення. CSS створювався, щоб відокремити стиль від вмісту вебсторінки за допомогою параметрів кольорів, макета, типографіки та інших [62].

CSS може доповнювати HTML безпосередньо в середині тегів HTML-документа або через окремий CSS-документ. Він управляє тегами HTML або деякою кількістю тегів одночасно, додаючи потрібні форми, кольори, відступи, інтервали, розміри, стилі та багато іншого.

CSS є презентацією та зовнішнім виглядом вебсторінки. Разом з HTML та JavaScript створюється інтерактивна сторінка браузера, наповнена кольором, формами та стилем.

2.13 JavaScript

Мова JavaScript була розроблена Бренданом Ейхом в середині 1990-х років для додання інтерактивності та гнучкості вебсайтів та використання у веббраузерах. Спочатку вона використовувалася для створення та налаштування форм ведення та використання числових символів. Це дозволило перевіряти введені символи та обробляти помилки. Це дало початок популярності JavaScript.

Назва JavaScript містить слово «Java», проте майже не має з цією мовою програмування нічого спільного. Використання частинки «Java»в назві було для звернення уваги, тому що мова Java була вже дуже популярна на той момент [63].

JavaScript на стороні клієнта або сервера використовує базовий синтаксис, який дозволяє писати власне ПЗ та програми. На стороні клієнта скрипт JavaScript найчастіше включений в HTML-документ або має посилання на нього. Це робиться, щоб код міг бути інтерпретований браузером та вебсторінка була динамічною та могла взаємодіяти з користувачем.

JavaScript можна використовувати для відстеження подій, таких як натискання кнопок, пересування мишкою, взаємодія з об’єктами. Це підвищує інтерактивність, додає постійних зворотний зв'язок з користувачем, вводить різні програмні інтерфейси та зменшує взаємодію з сервером, що економить серверний трафік та зменшує навантаження на сервер.

Мова програмування JavaScript дозволяє створювати змінні, які будуть використовуватися для маніпуляції даними та використовує основні типи даних, такі як числові значення, логічні значення, рядки символів, масиви та об'єкти. Написаний код додається в скрипт та взаємодіє з HTML та CSS, що дозволяє створювати цікаві та сучасні вебсайти та вебінтерфейси.

3. РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ЗЕЛЕНОЇ ЕНЕРГЕТИКИ ДЛЯ ЕФЕКТИВНОГО БАЛАНСУВАННЯ ЕНЕРГОСИСТЕМИ

3.1 Опис системи моніторингу зеленої енергетики для ефективного балансування енергосистеми

Система моніторингу зеленої енергетики для ефективного балансування енергосистеми створюється для збору параметрів зі станцій зеленої енергетики, їх аналіз та візуалізацію, щоб легше відстежувати та керувати енергосистемою загалом.

Головні завдання для майбутньої системи моніторингу зеленої енергетики для ефективного балансування енергосистеми:

1) Збирати параметри напруги або інші дані від датчиків, сенсорів або вузлів енергетичних станцій зеленої енергетики (сонячних електростанцій, вітрових електрогенераторів, гідроелектростанцій, геотермальних станцій, станцій біоенергетики та інших);
2) Обробляти та маршрутизувати отримані дані далі по інфраструктурі системи моніторингу;
3) Збирати та зберігати отримані параметри та метрики;
4) Будувати графіки, розклад, діаграми, моделі на основі отриманих параметрів;
5) Мати сучасний інтерфейс, який повинен бути не перенасичений деталями та налаштуваннями, легкий у розумінні та простий у користуванні;
6) Мати надійний захист від хакерських атак та неправомірного доступу до інтерфейсу та даних, який базується на засадах авторизації та автентифікації.

На рисунку 3.1 показану загальну схему роботи системи моніторингу зеленої енергетики.

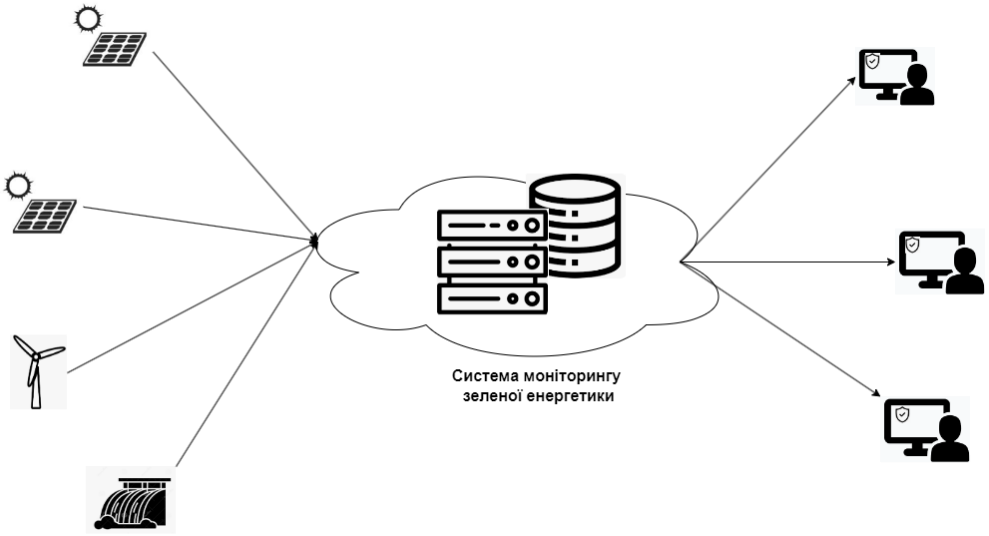

Рисунок 3.1 – Загальна схема роботи системи моніторингу зеленої енергетики

Як видно, система моніторингу зеленої енергетики збирає дані з датчиків енергетичних станцій зеленої енергетики. Потім передає їх на віддалені сервера, де зберігає їх, обробляє та створює візуалізацію. Користувачі можуть віддалено користуватися системою, моніторити параметри та аналізувати дані.

На рисунку 3.2 показано більш детальну схему роботи системи моніторингу зеленої енергетики з використанням технологій з розділу 2.

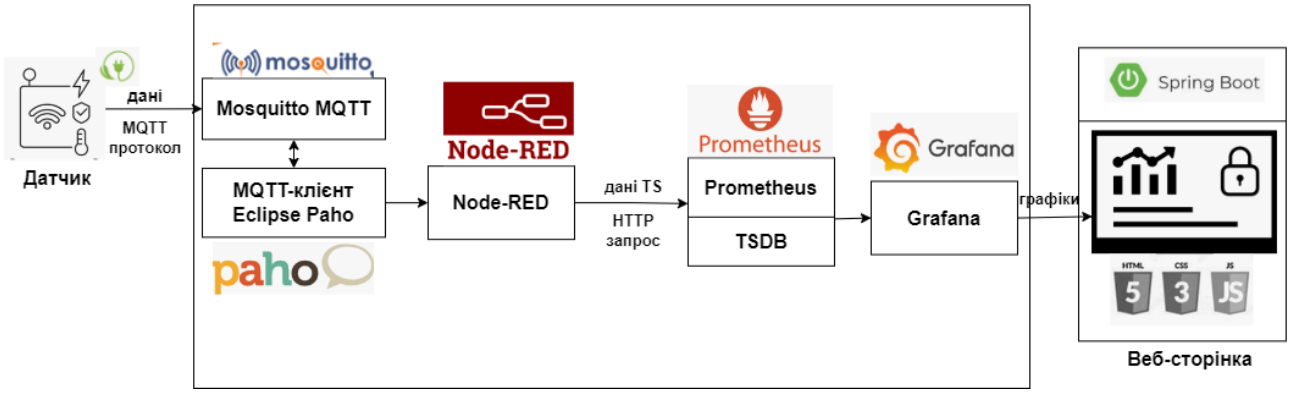

Рисунок 3.2 – Схема роботи системи моніторингу зеленої енергетики зі стеком технологій

Дані з датчика через протокол MQTT обробляються брокером Eclipse Mosquitto. MQTT-клієнт Eclipse Paho взаємодіє з MQTT брокером та емулює запити від датчиків. Node-RED створює потік даних та передає через HTTP запит в Prometheus, який збирає метрики та виконує роль TSDB. Grafana візуалізує дані та створює графіки. Spring Boot запускає вебсервер та його бібліотеки допомагають захистити доступ за допомогою авторизації. Вебсторінка відображає графіки в режимі реального часу та інтерактивний користувацький інтерфейс за допомогою мов програмування HTML, CSS, JavaScript.

Розробка виконувалась на локальному комп’ютері з характеристиками:

- Процесор: Intel™ Core™ i5-4440 CPU @ 3.10GHz   3.10 GHz;
- Відеокарта: NVIDIA GeForce GT 740;
- Операційна система: Windows 10 Home;
- Версія операційної системи: 22H2;
- Тип системи: 64-bit operating system, x64-based processor.

3.2 Встановлення та налаштування Mosquitto

Інсталятор Eclipse Mosquitto для Windows потрібно завантажити з офіційного сайту за адресою «https://mosquitto.org/». Запуск MQTT-брокера відбувається через команду «mosquitto –v», яку потрібно ввести в командний рядок Windows. На рисунку 3.3 видно результат запуску Mosquitto в командний рядок.

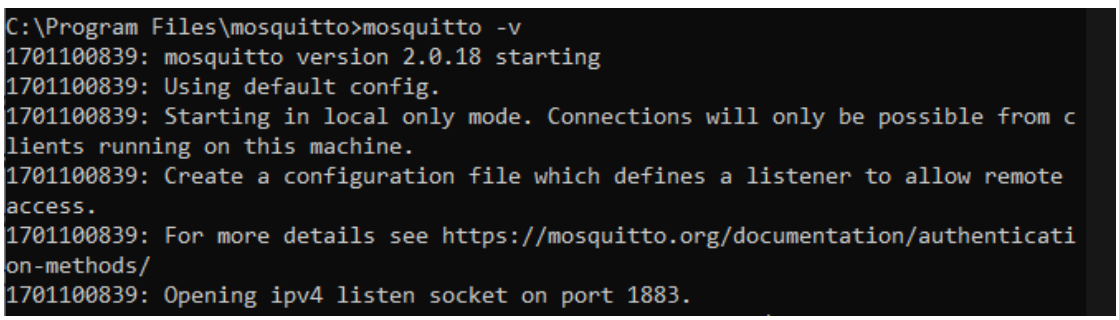

Рисунок 3.2 – Запуск Mosquitto в командному рядку Windows

З рисунка видно, що Mosquitto запущений і слухає порт 1883.

Далі потрібно перевірити статус служби Mosquitto. Для цього в командний рядок вводиться команда «sc query mosquitto». На рисунку 3.3 видно роботу цієї команди.

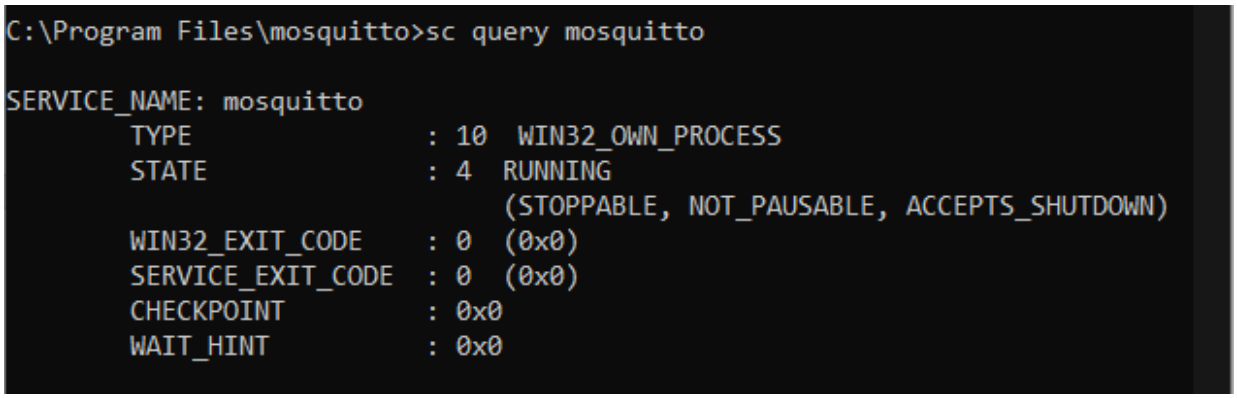

Рисунок 3.3 – Перевірка статус служби Mosquitto

Статус служби показаний як «RUNNING», тобто «ЗАПУЩЕННИЙ». Це значить, що служба Mosquitto активна.

Для перевірки порту 1883 виконується команда «netstat -an | find "1883"». На рисунку 3.4 показано роботу команди.

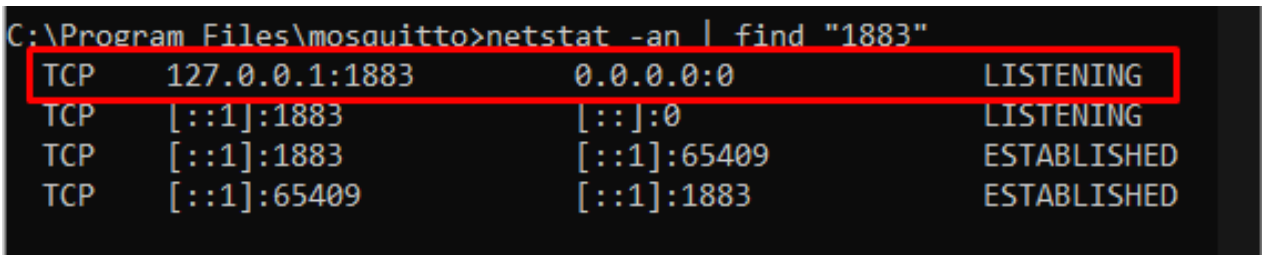

Рисунок 3.4 – Перевірка порту 1883

Напроти «TCP 127.0.0.1:1883» видно напис «LISTENING». Це означає, що брокер Mosquitto слухає порт 1883, тому може отримувати дані від датчиків.

3.3 Налаштування Eclipse Paho Java Client та емуляція роботи датчиків

Для налаштування MQTT-клієнта Eclipse Paho потрібно створити проєкт в середовище розробки Eclipse. На рисунку 3.5 показано створення проєкту.

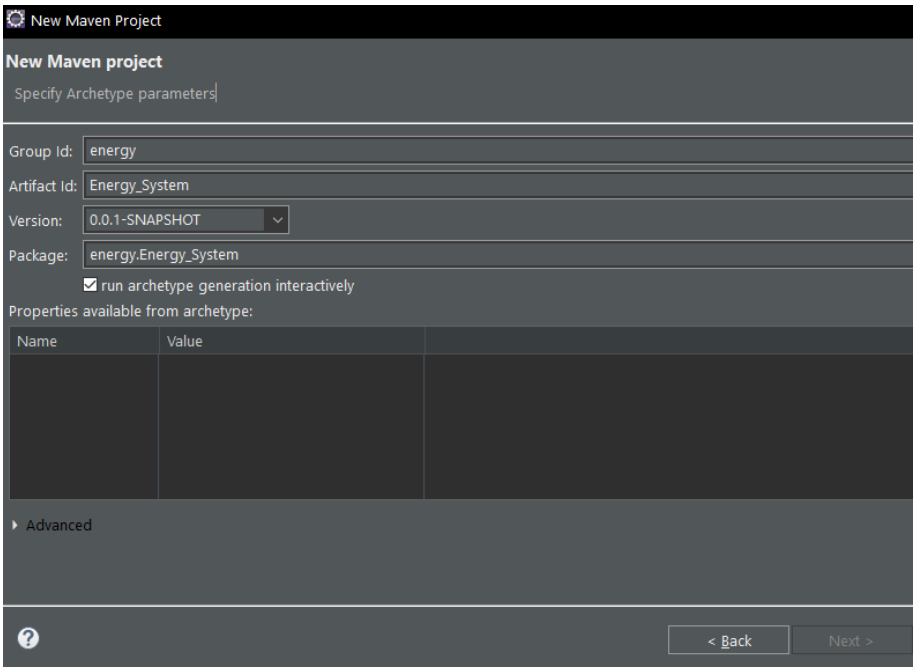

Рисунок 3.5 – Створення проєкту Eclipse Paho

Назва проєкту «Energy_System». Будується та складається проєкт за допомогою інструменту Maven.

Після створення проєкту потрібно додати в файл «pom.xml» залежність «org.eclipse.paho» для підключення Eclipse Paho Java Client. На рисунку 3.6 показана створена залежність в файлі «pom.xml».

```
20
21  <dependencies>
22    <dependency>
23        <groupId>org.eclipse.paho</groupId>
24        <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
25        <version>1.2.5</version>
26    </dependency>
27  </dependencies>
28
```

Рисунок 3.6 – Додання залежності в файл «pom.xml»

Залежність міститься між двома скриптами «<dependencies>» та містить назву групи, версію та назву, що відповідає розміщенню пакетів бібліотеки Eclipse Paho. Maven автоматично завантажує бібліотеку та оновлює проєкт.

Потім потрібно створити Java клас для підключення до MQTT брокера та створення видимості роботи датчиків. У створеному класі спочатку потрібно позначити порт, який прослуховує MQTT брокер, та назви датчиків. На рисунку 3.7 показано позначення порту та назву датчиків.

```
public class App
{

    public static void main(String[] args) {
        String broker = "tcp://localhost:1883";
        String clientId = "SensorEmulator";
        String topic = "sensor/data";

        //позначаємо датчики станцій зеленої енргетики
        String hydroStationTopic = "hydro_station_sensor/data";
        String solarStationTopic = "solar_station_sensor/data";
        String windStationTopic = "wind_station_sensor/data";
        String geothermalStationTopic = "geothermal_station_sensor/data";
        String bioStationTopic = "bio_station_sensor/data";
```

Рисунок 3.7 – Позначення порту MQTT брокера та датчиків

Назви датчиків відповідають за гідро-, сонячну, вітрову, геотермальну та біо-станції. Вони потрібні для формування теми (топіку) повідомлення.

Далі потрібно під‘єднатися до MQTT брокера за допомогою класу «MqttClient», куди передається позначений порт. Клас «MqttClient» відповідає за підключення до брокера, доставлення повідомлень через мережу та перезавантаження клієнта.

Потім потрібно згенерувати повідомлення з показниками від кожної станції та відправити через встановлений порт. Відправлення повідомлення відбувається за допомогою методу «client.publish()». На рисунку 3.8 показано підключення до MQTT брокера та генерація повідомлення від гідростанції для прикладу.

```
//емулюємо запити від датчиків
try {
    System.out.println("Connecting to broker: " + broker);
    MqttClient client = new MqttClient(broker, clientId);
    client.connect();

    while (true) {
        String message = " ";

        //HydroStation
        message = sensorData(80, 100);
        MqttMessage mqttMessageHydro = new MqttMessage(message.getBytes());
        client.publish(hydroStationTopic, mqttMessageHydro);
        System.out.println("Sent: "+ hydroStationTopic + " " + message);

        //SolarStation
```

Рисунок 3.7 – Підключення до MQTT брокера та генерація повідомлення

Після генерації повідомлення з показниками від кожної станції потрібно додати «Thread.sleep(5000)» для емуляції очікування 5 секунд перед наступним посиланням даних. Повний код програми показано у додатку А.

Після запуску програми у консолі почнуть відображатися створені повідомлення від станцій. На малюнку 3.8 показано роботу консолі.

```
Markers  Properties  Servers  Data Source Explorer  Snippets
<terminated> App [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (
Sent: hydro_station_sensor/data 89.63356247329935
Sent: solar_station_sensor/data 58.56192825671555
Sent: wind_station_sensor/data 58.659880290325475
Sent: geothermal_station_sensor/data 21.24225270713866
Sent: bio_station_sensor/data 2.2912645486519945
Sent: hydro_station_sensor/data 80.20180134178014
Sent: solar_station_sensor/data 55.5671221114988
Sent: wind_station_sensor/data 52.68351947620321
Sent: geothermal_station_sensor/data 13.989562470771398
Sent: bio_station_sensor/data 17.620360680249902
Sent: hydro_station_sensor/data 86.19743944839718
Sent: solar_station_sensor/data 58.92496763377156
Sent: wind_station_sensor/data 63.309060519038425
Sent: geothermal_station_sensor/data 18.521966027629553
```

Рисунок 3.8 – Робота консолі програми Eclipse Paho Java Client

Генерація повідомлень від датчиків відбувається кожні 5 секунд.

3.4 Встановлення Node-RED та створення потоку

Для конвертації даних з MQTT через HTTP порт до Prometheus потрібно використовувати спеціальний інструмент Node-RED. Для цього потрібно встановити Node.js та завантажити Node-RED за допомогою команди «npm install -g --unsafe-perm node-red».

Для запуску Node-RED в командний рядок Windows вводиться команда «node-red». На рисунку 3.9 показано роботу команди «node-red».

```
C:\Users\ddenc>node-red
11 Dec 17:56:36 - [info]

Welcome to Node-RED
===================

11 Dec 17:56:36 - [info] Node-RED version: v3.1.0
11 Dec 17:56:36 - [info] Node.js  version: v18.18.0
11 Dec 17:56:36 - [info] Windows_NT 10.0.19045 x64 LE
11 Dec 17:56:38 - [info] Loading palette nodes
11 Dec 17:56:42 - [info] Settings file  : C:\Users\ddenc\.node-red\settings.js
11 Dec 17:56:42 - [info] Context store  : 'default' [module=memory]
11 Dec 17:56:42 - [info] User directory : \Users\ddenc\.node-red
11 Dec 17:56:42 - [warn] Projects disabled : editorTheme.projects.enabled=false
11 Dec 17:56:42 - [info] Flows file     : \Users\ddenc\.node-red\flows.json
11 Dec 17:56:42 - [warn]

---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------

11 Dec 17:56:42 - [info] Server now running at http://127.0.0.1:1880/
11 Dec 17:56:42 - [info] Starting flows
```

Рисунок 3.9 – Запуск Node-RED в командному рядку Windows

Після цього, доступ до інтерфейсу Node-RED отримано за адресою «http://localhost:1880».

Інтерфейс Node-RED складається із палітри вузлів ліворуч, робочої області посередині та інформативних панелей праворуч. На рисунку 3.10 показано запущений інтерфейс Node-RED за адресою «http://localhost:1880» в інтернет браузері.

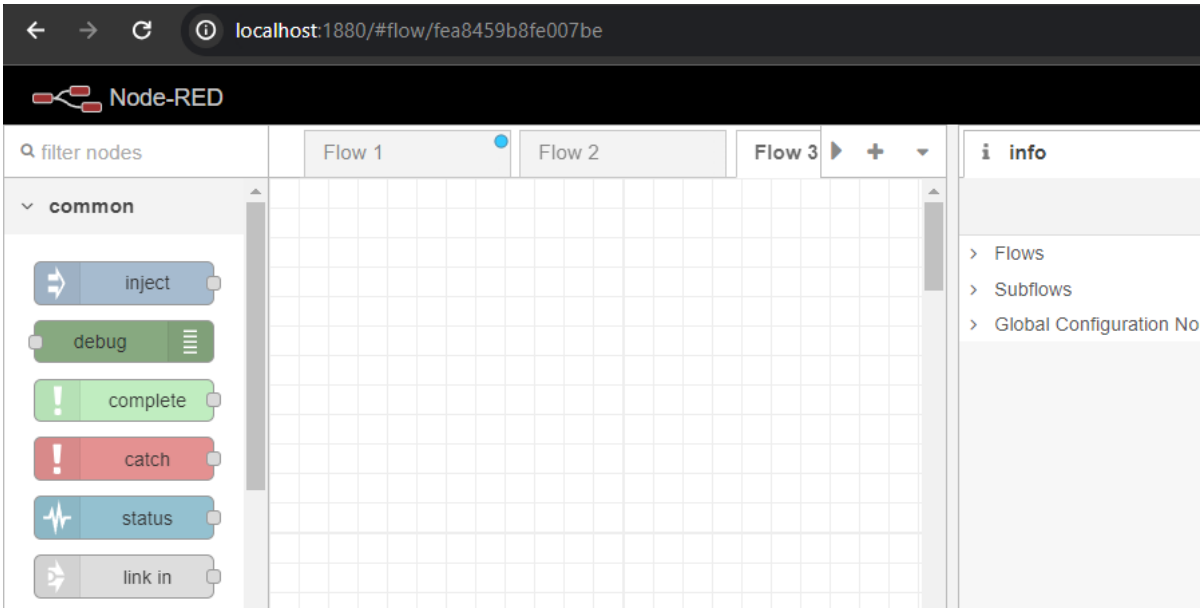

Рисунок 3.10 – Початковий інтерфейс Node-RED в інтернет браузері

Для створення потоку даних потрібно встановити бібліотеки «node-red-contrib-mqtt-broker» та «node-red-contrib-prometheus-exporter» в панелі «Nodes». Вони дадуть доступ до вузлів, які потрібні для налаштування потоку повідомлення. Вузли потрібно перенести з палітри вузлів до робочої області та з'єднати між собою.

Вузол «mqtt in» підключається до брокера MQTT і підписується на повідомлення із зазначеної теми (топіка). У ньому вказується порт MQTT, що для нашої системи відповідає сервер «http://localhost:1883», та назва топіку повідомлення від датчика.

Вузол «function» дії для повідомлень, що надходять на вузол за допомогою JavaScript-функції. Він допомагає прикріпити мітку та сформувати правильне представлення повідомлення.

Вузол «prometheus out» передає повідомлення до Prometheus. У ньому

вказується назва метрики, що відповідає назві датчика.

Вузол «debug» відображає вибрані властивості повідомлення на вкладці бічної панелі й журнал виконання. Він використовується для відстеження роботи потоку та тестування.

Для формування потоку повідомлення потрібно з'єднати вказані вузли разом та налаштувати їх. На рисунку 3.11 показано приклад сформованого потоку для гідростанції.

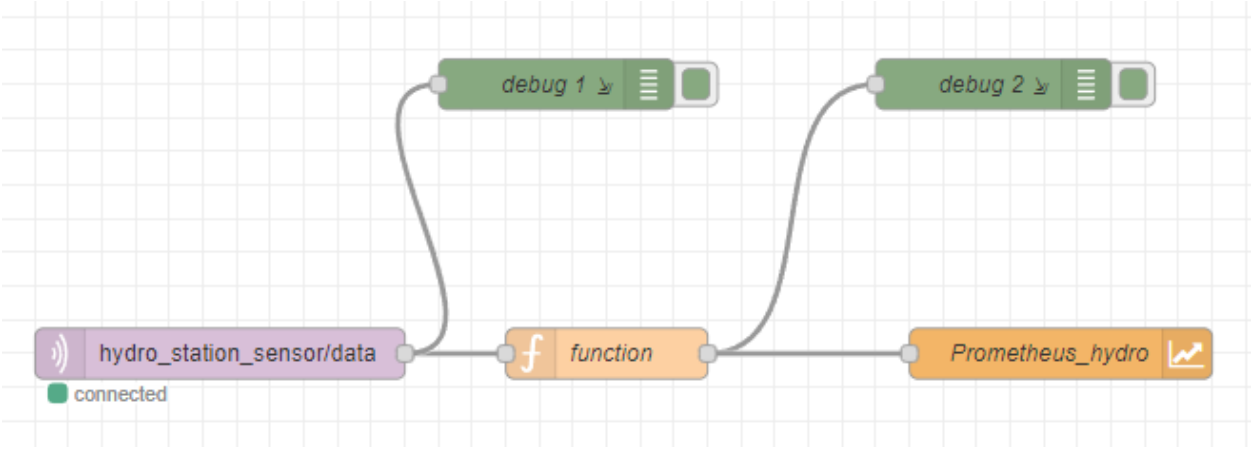

Рисунок 3.11 – Створений потік повідомлень у Node-RED для гідростанції

Індикатор «connected» означає, що вузол «mqtt in» з'єднаний з портом MQTT та може зчитувати з нього повідомлення.

Потрібно створити та налаштувати потоки повідомлень для кожної станції зеленої енергетики, вказуючи потрібні топіки та метрики.

Для запуску роботи потоків потрібно натиснути на червону кнопку «Deploy», яка знаходиться праворуч зліва на навігаційній панелі сторінки. Вона оновить інтерфейс та запустить потоки повідомлень. Якщо запуск та з'єднання зі зовнішніми сервісами пройшло успішно, то висвітлиться блок з написом «Sucsessfully deployed».

На рисунку 3.12 показано запуск потоків повідомлень для всіх станцій зеленої енергетики.

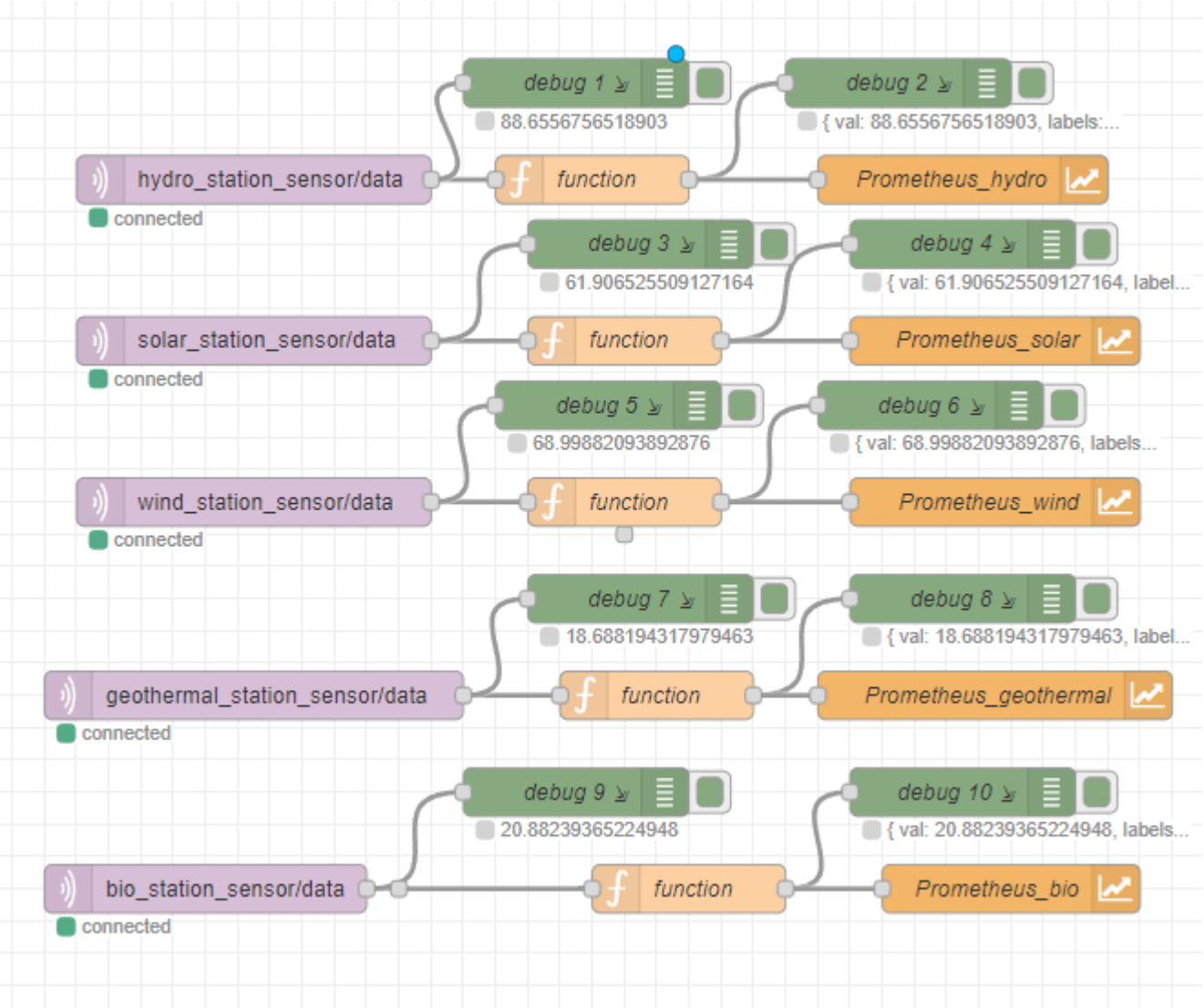

Рисунок 3.12 – Робота потоків повідомлень для станцій зеленої енергетики у Node-RED

Під вузлом «debug» відображаються показники для кожної станції.

3.5 Встановлення та налаштування Prometheus

Для встановлення Prometheus потрібно перейти на офіційний сайт Prometheus за адресою «https://prometheus.io/» і завантажте останню версію для Windows. Завантажений архів потрібно розпакувати. У головній директорії треба змінити

вміст конфігураційного файлу «prometheus.yml», вказавши з’єднання з Node-RED. Для цього потрібно у тег «static_configs: - targets:» додати порт з’єднання з Node-RED «['localhost:1880']». На рисунку 3.13 показано зміни конфігураційного файлу «prometheus.yml».

```
# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  - job_name: 'node-red'
    static_configs:
      - targets: ['localhost:1880']
```

Рисунок 3.13 – Зміни конфігураційного файлу «prometheus.yml»

Для запуску Prometheus потрібно перейти до головної директорії та ввести у командний рядок Windows команду «prometheus.exe --config.file=prometheus.yml». На рисунку 3.14 показано запуск Prometheus у командному рядку Windows.

```
C:\Diplom\prometheus\prometheus-2.48.0.windows-amd64>prometheus.exe --config.file=prometheus.yml
ts=2023-12-11T18:20:19.981Z caller=main.go:539 level=info msg="No time or size retention was set so usin
g the default time retention" duration=15d
ts=2023-12-11T18:20:19.983Z caller=main.go:583 level=info msg="Starting Prometheus Server" mode=server v
ersion="(version=2.48.0, branch=HEAD, revision=6d80b30990bc297d95b5c844e118c4011fad8054)"
ts=2023-12-11T18:20:19.983Z caller=main.go:588 level=info build_context="(go=go1.21.4, platform=windows/
amd64, user=root@755dcbe41edf, date=20231116-04:38:48, tags=builtinassets,stringlabels)"
ts=2023-12-11T18:20:19.983Z caller=main.go:589 level=info host_details=(windows)
ts=2023-12-11T18:20:19.983Z caller=main.go:590 level=info fd_limits=N/A
ts=2023-12-11T18:20:19.983Z caller=main.go:591 level=info vm_limits=N/A
ts=2023-12-11T18:20:20.006Z caller=web.go:566 level=info component=web msg="Start listening for connecti
ons" address=0.0.0.0:9090
ts=2023-12-11T18:20:20.012Z caller=main.go:1024 level=info msg="Starting TSDB ..."
ts=2023-12-11T18:20:20.016Z caller=tls_config.go:274 level=info component=web msg="Listening on" address
=[::]:9090
ts=2023-12-11T18:20:20.018Z caller=tls_config.go:277 level=info component=web msg="TLS is disabled." htt
p2=false address=[::]:9090
ts=2023-12-11T18:20:20.035Z caller=repair.go:56 level=info component=tsdb msg="Found healthy block" mint
=1701103848745 maxt=1701122400000 ulid=01HGB74AA0WWR9Q5X6Y2JMHEM4
ts=2023-12-11T18:20:20.049Z caller=repair.go:56 level=info component=tsdb msg="Found healthy block" mint
=1701167264481 maxt=1701194400000 ulid=01HGC2K3H40MJJN2CG7YE1BTYD
```

Рисунок 3.14 – Запуск Prometheus у командному рядку Windows

Prometheus працює за адресою «http://localhost:9090». Щоб відкрити вебінтерфейс Prometheus, потрібно перейти у веббраузер і ввести дану адресу.

На рисунку 3.15 можна побачити запущений вебінтерфейс Prometheus у веббраузері.

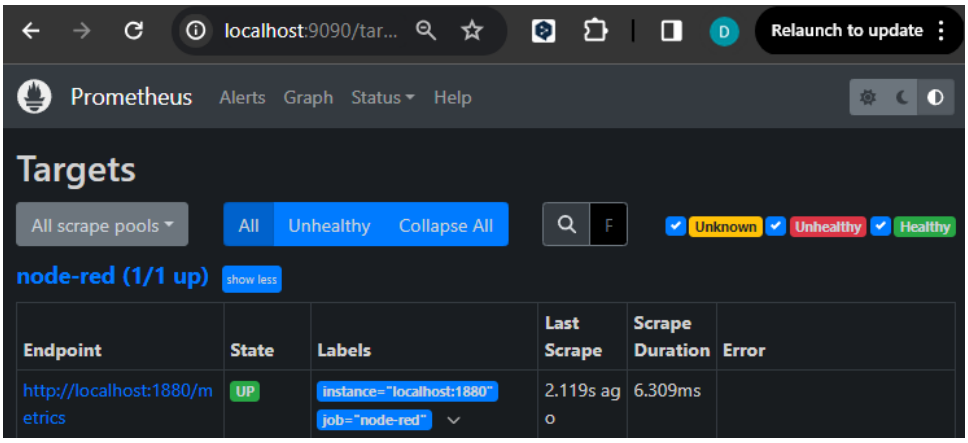

Рисунок 3.15 – Вебінтерфейс Prometheus

У вкладці «Targets» сервер, який підключений до потоку Node-RED має статус «UP». Отже, Prometheus працює і може збирати та зберігати метрики від станцій зеленої енергетики. Метрики зберігаються у вигляді часових рядів, тобто містять дані та час створення.

3.6 Встановлення та налаштування Grafana

Спершу потрібно встановити середовище візуалізації Grafana на Windows через офіційний сайт Grafana Installation. Доступ до інтерфейсу Grafana буде за адресою «http://localhost:3000/». На рисунку 3.16 показано інтерфейс Grafana у веббраузері.

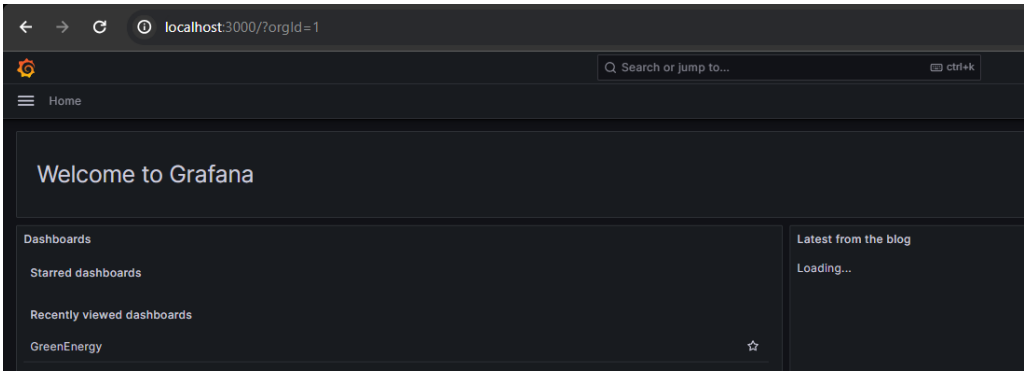

Рисунок 3.16 – Інтерфейс Grafana у веббраузері

Для налаштування з'єднання між Grafana та Prometheus потрібно в Grafana додати джерело даних Prometheus. Для цього потрібно перейти до «Configuration», «Data Sources», потім натиснути «Add your first data source», вибрати «Prometheus» серед джерел даних та встановити URL «http://localhost:9090/» до Prometheus.

Для створення графіків потрібно перейти на вкладку «Create», «Dashboard», додати нову панель графіку (Add Panel), вибрати «Prometheus» та встановити запит до Prometheus у розділі «Metric».

Для гідростанції потрібно вибрати метрику «hydro_station_sensor_data». На рисунку 3.17 показано панель створення графіку для гідростанції.

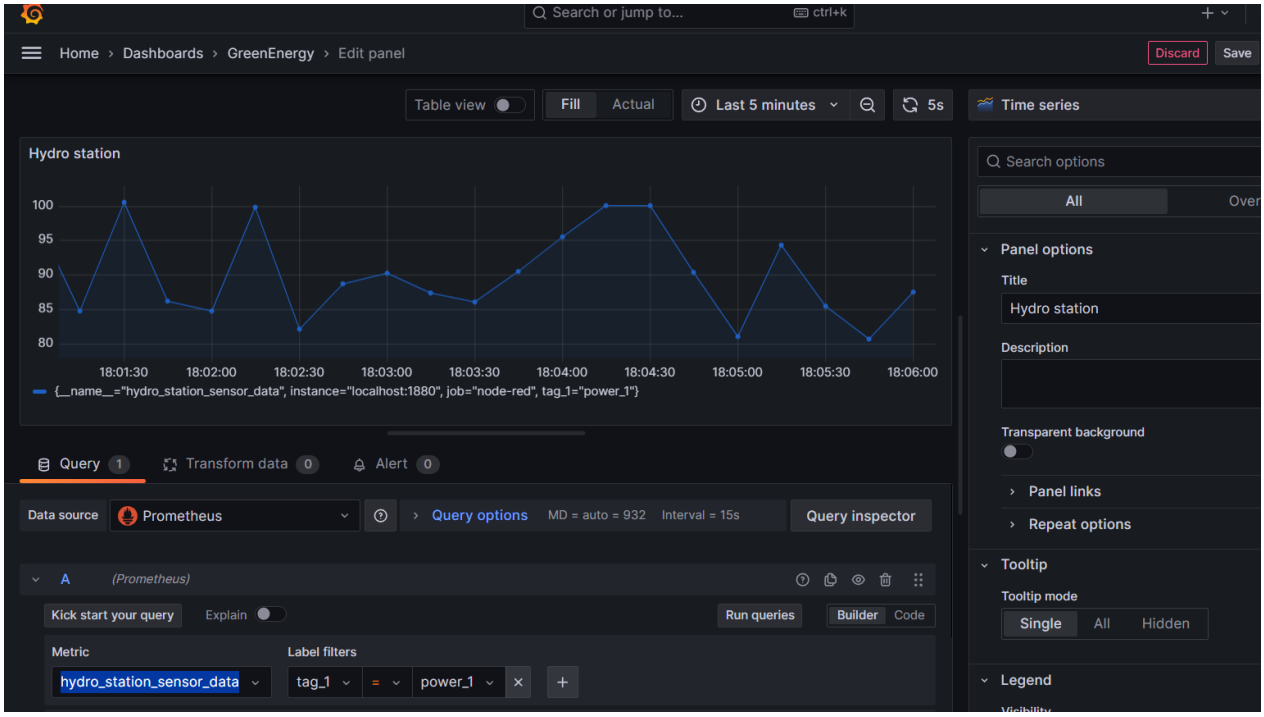

Рисунок 3.17 – Графік для гідростанції в середовище Grafana

На графіку можна змінювати стиль, ширину, наповненість, колір та іншу кастомізацію. Середовище Grafana відрізняється своєю можливістю покращення візуалізації отриманих даних. Кожна метрика містить параметри та дату.

Для створення спільного графіка зі всіма станціями зеленої енергетики потрібно створити новий графік та додати додаткові метрики з Prometheus у розділі «Metric». На рисунку 3.18 показано спільний графік зі всіма станціями зеленої

енергетики у панелі створення графіку.

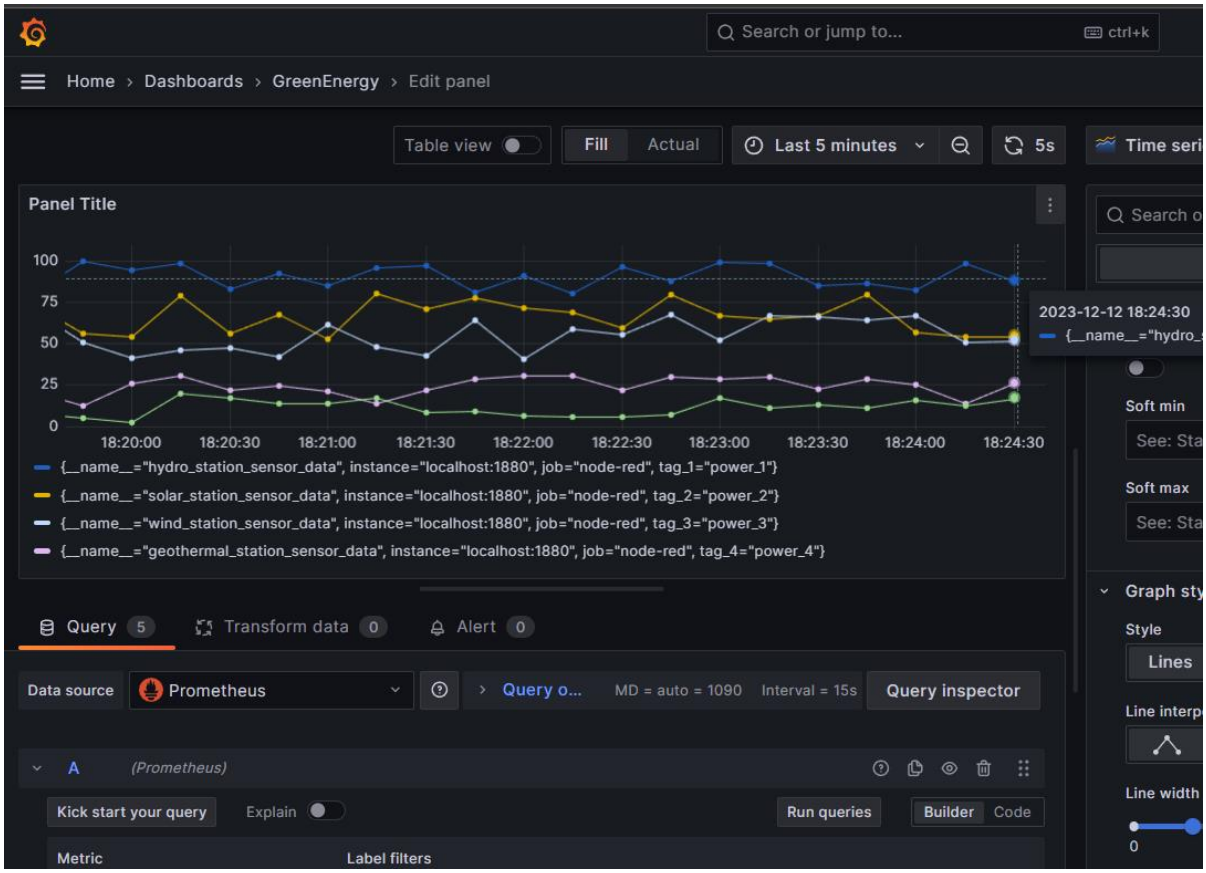

Рисунок 3.18 – Спільний графік для станцій в середовище Grafana

Кожна станція має свій колір. При наведенні на метрику виводиться час, дані та інша характеристика.

У Grafana можна вказати відрізок часу, який потрібно виводити, вказавши початковий та кінцевий час. Також можна позначити період відстежування даних (5 секунд, 30 секунд, 1 хвилина, 15 хвилин та інші) та час оновлення.

Потрібно створити графіки для кожної станції зеленої енергетики, змінивши колір та вказавши назву станції.

На рисунку 3.19 показано графіки для кожної станції зеленої енергетики та спільний графік у панелі «Dashboard».

Рисунок 3.19 – Створені графіки для кожної станції в середовище Grafana

Графіки оновлюються в режимі реального часу кожні 5 секунд. Кожний графік має свій колір та містить дані від станцій зеленої енергетики.

3.7 Налаштування Spring Boot та засобів захисту вебінтерфейсу

Для реалізації серверної частини вебінтерфейсу системи моніторингу зеленої енергетики використовується Spring Boot та бібліотеки, які містять потрібні функції. Спочатку потрібно створити Spring проєкт «GreenEnergy» на основі

збирача проєктів Maven та внести залежності у файл «pom.xml», щоб Maven автоматично завантажив їх до проєкту. На рисунку 3.20 показано залежності у «pom.xml».

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
</dependencies>
```

Рисунок 3.20 – Залежності проєкту

Бібліотека «spring-boot-starter-thymeleaf» використовується для обслуговування HTML у вебдодатках, щоб зменшити написання коду. Бібліотека «spring-boot-starter-web» використовується додавання в проєкт усіх бібліотеки, необхідні для створення вебпрограми. Бібліотека «spring-boot-starter-security» для додання захисту та можливості авторизації та автентифікації.

Клас «GreenEnergyApplication» є головним та з нього починається запуск програми. На рисунку 3.21 показано клас GreenEnergyApplication.

```
@SpringBootApplication
public class GreenEnergyApplication {

    public static void main(String[] args) { SpringApplication.run(GreenEnergyApplication.class, args); }

}
```

Рисунок 3.21 – Клас «GreenEnergyApplication»

Анотація «@SpringBootApplication» показує Spring Boot з якого файлу слід

запускати програму.

Для створення обробки помилок потрібно створити файли з текстом, який буде з’являтися під час помилок. На рисунку 3.22 показано код файлу «error.html», який написаний мовою HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Error</title>
</head>
<body>
    <h1>Error, something went wrong!</h1>
</body>
</html>
```

Рисунок 3.22 – Код файлу «error.html»

Текст «Error, something went wrong!» повинен буди з’являтися у разі виникнення помилок.

На рисунку 3.23 показано HTML код файлу «404.html».

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Not found</title>
</head>
<body>
    <h1>Page not found!</h1>
</body>
</html>
```

Рисунок 3.23 – Код файлу «404.html»

Текст «Page not found!» з’явиться, якщо користувач неправильно введе назву

вебсторінки в адресному рядку веббраузера.

Для того, щоб браузер знав про обробку помилок програмою, потрібно ввести команди «server.error.whitelabel.enabled=false» та «server.error.path=/error» в файл «application.properties», який відповідає за налаштування проєкту. Ці команди дозволять перекласти відповідальність обробки помилок на Spring Boot.

Далі потрібно створити спеціальний контролер, який буде регулювати обробку помилок. Контролер призначений для опрацювання запитів від клієнта і повернення результатів. На рисунку 3.24 показано контролер «WebErrorController», який відповідальний за обробку помилок.

```
@Controller
public class WebErrorController implements ErrorController {
    @RequestMapping("/error")
    public String handError(HttpServletRequest request){
        Object status = request.getAttribute(RequestDispatcher.ERROR_STATUS_CODE);
        //Перевірка на статус 404
        if(status != null &&
                Integer.valueOf(status.toString()) == HttpStatus.NOT_FOUND.value()){
            return "404";
        }
        return "error";
    }
}
```

Рисунок 3.24 – Контролер «WebErrorController»

Анотація «@Controller» показує Spring Boot, що цей клас контролер. Анотація «@RequestMapping("/error")» показує, що цей контролер працює зі запитом для помилок. Якщо помилка від сервера або користувача, тоді браузер завантажить файл «error.html». А якщо помилка, коли користувач неправильно введе назву вебсторінки, тоді браузер покаже файл «404.html».

Захист інформації та забезпечення безпечного доступу до вебінтерфейсу важлива частина розробки програми. Для цього потрібно створити окремий клас, який буде відповідати за захист додатка, авторизація та автентифікація системи. Клас називається «WebSecurityConfig», де за допомогою інтерфейсу

«UserDetailsService» реалізовується захищений доступ до вебсторінки та пересилання на сторінку авторизації. На рисунку 3.25 показано частина коду класу «WebSecurityConfig».

Рисунок 3.25 – Частина коду класу «WebSecurityConfig»

Анотація «@Configuration» показує Spring Boot, що це конфігурація проєкту, тобто його налаштування. Анотація «@EnableWebSecurity» позначає, що ця конфігурація відповідає за захист інформації. Анотація «@Bean» використовується для видимості об’єкта всій програмі.

Для першочергового доступу в методі «userDetailsService()» описується користувач «admin» з доступом адміністратора та такими ж логіном та паролем. Для додання нових користувачів можна під'єднати БД. Пароль шифрується в методі «passwordEncoder()» за допомогою кодування «BCryptPasswordEncoder», яке засновано на хешуванні даних. Повний код класу «WebSecurityConfig» знаходиться в додатку Б.

Коли користувач захоче перейти на вебсторінку, тоді вебдодаток перенаправить його на сторінку авторизації, де буде потрібно ввести логін та

пароль. Якщо введений логін або пароль не збігаються, тоді доступ до вебінтерфейсу буде заблоковано.

На рисунку 3.26 показано форму авторизації проєкта, створена бібліотекою безпеки Spring Boot.

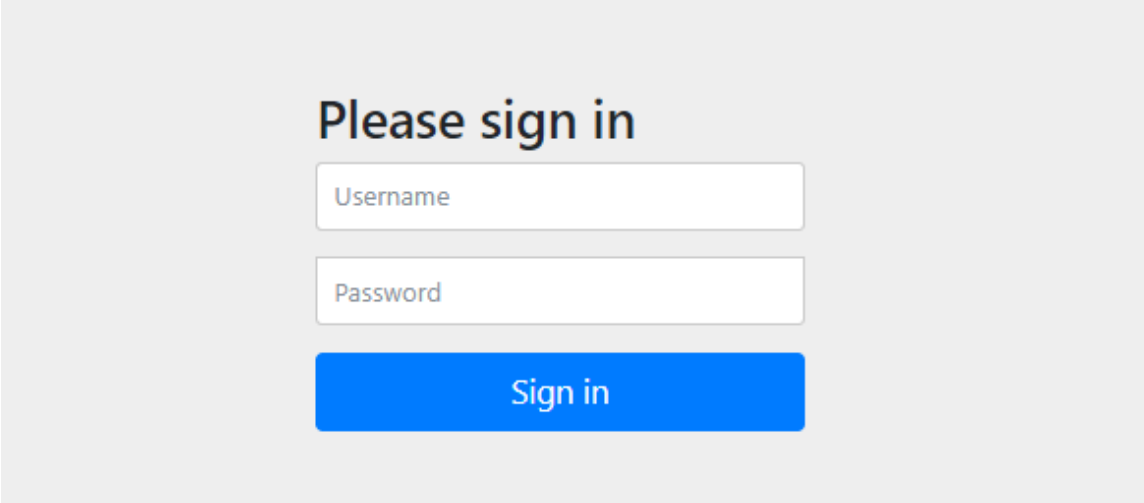

Рисунок 3.26 – Форма авторизації

У рядку «Username» потрібно ввести ім'я користувача (логін), а в «Password» – пароль. Якщо все введено правильно, тоді доступ буде дозволено і вебдодаток перенаправить користувача до вебінтерфейсу.

Для налаштування запиту від користувача та формування адресного рядка вебінтерфейсу потрібно створити контролер «GreenEnergyController». На рисунку 3.27 показано реалізацію контролера «GreenEnergyController».

```
8   @Controller
9   @RequestMapping("/greenEnergy")
10  public class GreenEnergyController {
11      @GetMapping
12      public String getIndex(){
13          return "index";
14      }
15  }
```

Рисунок 3.27 – Контролер «GreenEnergyController»

Анотація «@RequestMapping("/greenEnergy")» опрацьовує адресний рядок «/greenEnergy», де буде розгорнуто вебінтерфейс. Анотація «@GetMapping» працює на посилання результату під час запиту. Контролер «GreenEnergyController» перенаправляє запит до файлу «index.html», де реалізовано вебінтерфейс.

Spring Boot дає можливість запускати вебдодаток на локальному сервері. На рисунку 3.28 показано запуск програми за допомогою Spring Boot.

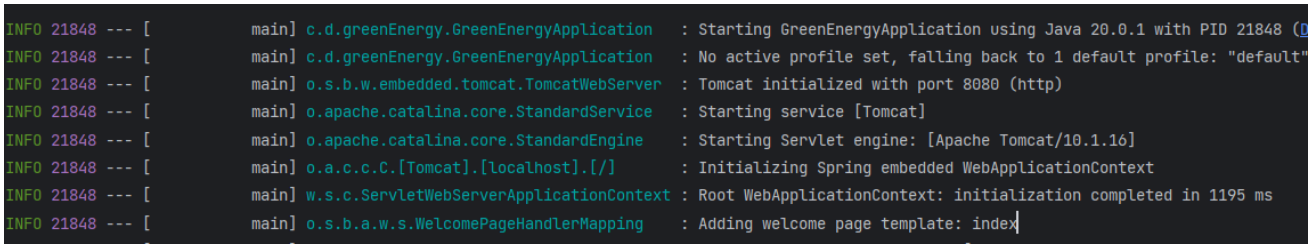

Рисунок 3.28 – Запуск програми за допомогою Spring Boot

Вебінтерфейс тепер доступний на порту 8080 за адресним рядком «http://localhost:8080/».

3.8 Розробка користувацького вебінтерфейсу системи

Користувацький інтерфейс системи повинен бути простим та досить інтерактивним. Для його реалізації знадобляться мови програмування HTML, CSS та JavaScript.

Спочатку за допомогою HTML потрібно зробити розмітку сторінки, розбити її на секції та додати навігаційну панель. Навігаційна панель створюється за допомогою тегу «<nav>» для відокремлення, тегу «<img>» для логотипа та тегів «<a>» для посилань на частини сторінки або інші ресурси. Секції позначаються тегами « ». Потрібно реалізувати дві основні секції: перша для окремих

графіків для кожної станції зеленої енергетики, друга для загального графіка. Додавання графіків з інструмента Grafana відбувається за допомогою тегу «<iframe>», де потрібно вказати посилання на конкретний графік. Повний код програми знаходиться в додатку В.

Для додання кольорів, стилів, розташування та розмірів потрібно скористатися CSS. Стилі описуються в тегу «<style>» HTML сторінки. Опис стилів для кожної секції, об’єктів та елементів наведено в додатку В.

Для інтерактивності вебінтерфейсу потрібно скористатися мовою програмування JavaScript. Вона має багато можливостей для побудови сучасних та інтерактивних вебсторінок. Наприклад, можна додати можливість показувати графік під час наведення курсором на значок конкретної станції. Це робиться для того, щоб зменшити перенасичення інтерфейсу різними деталями та зробити його простим у використанні. Для цього потрібно скористатися командою «'mouseover'». На рисунку 3.29 показано реалізації події при наведенні курсора.

```
130    // Додаємо обробник події при наведенні курсору
131    stationElement.addEventListener('mouseover', () => {
132        // Створюємо контейнер для графіка
133        const graphContainer = document.createElement('div');
134        graphContainer.className = 'graph-container';
135
136        // Створюємо iframe із графіком Grafana
137        const iframe = document.createElement('iframe');
138        iframe.src = station.grafanaUrl;
139        iframe.width = "450";
140        iframe.height = "200";
```

Рисунок 3.29 – Реалізація події при наведенні курсора

Коли курсор миші буде наведено на значок станції зеленої енергетики, то спрацює скрипт «'mouseover'», який створить елемент «iframe», де відображається графік з Grafana.

У скрипті JavaScript також вказується посилання на графіки з Grafana для

кожної станції, їх розміри та значки. Всі рисунки та значки знаходяться в папці «images» каталогу проєкта. Потрібно посилання на рисунки для їх використання. Код JavaScript знаходиться між тегами «<script>» на HTML розмітці. Повний код скрипту наведено в додатку В.

На рисунку 3.30 наведено реалізацію вебінтерфейсу системи моніторингу зеленої енергетики для ефективного балансування енергосистеми.

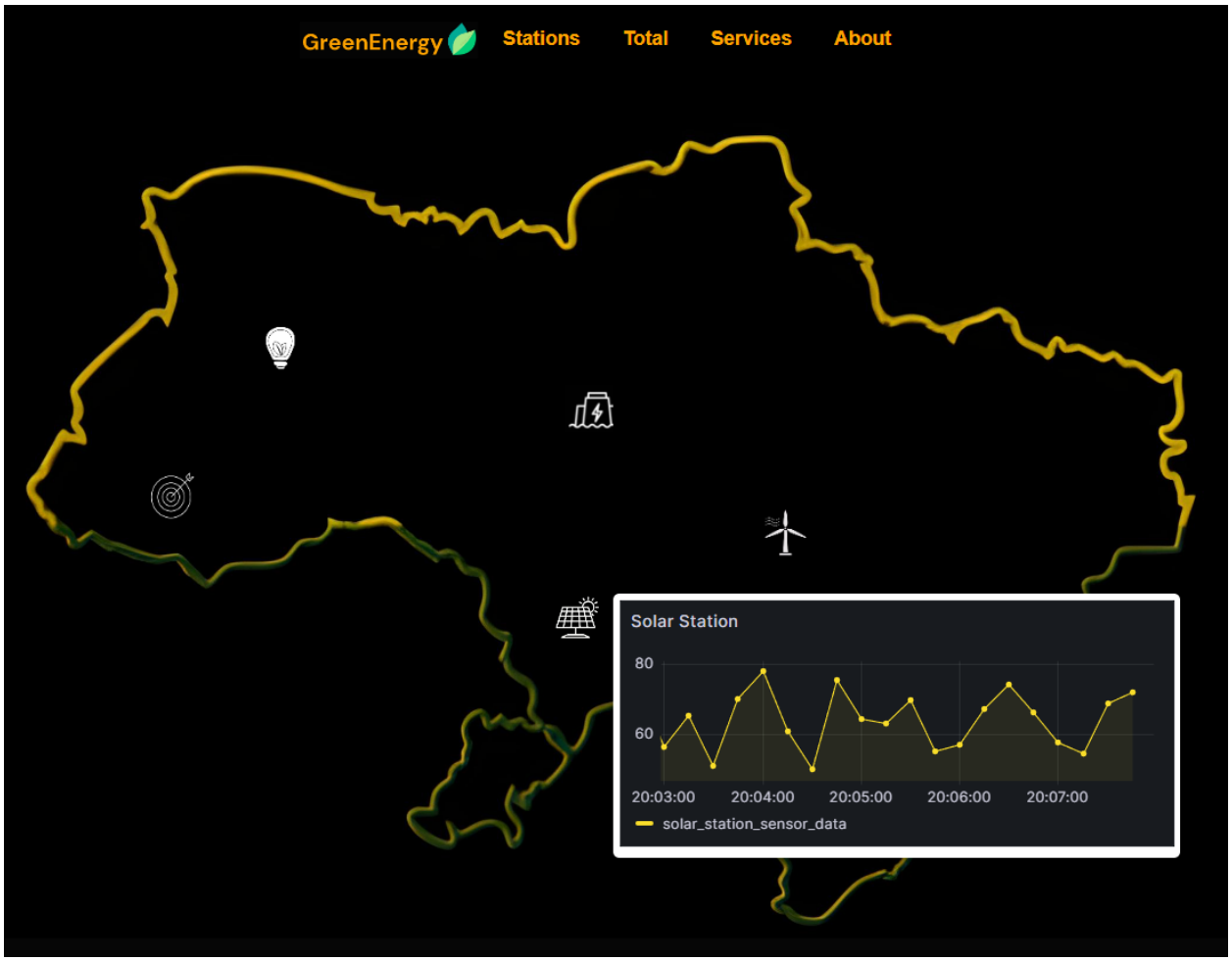

Рисунок 3.30 – Реалізація вебінтерфейсу

Якщо навести курсор миші на значок станції, тоді поряд з’явиться графік з Grafana із показниками датчика в режимі реального часу. Це працює з кожною станцією. На рисунку було наведено на значок сонячної станції, тому праворуч відображається графік датчика сонячної станції.

Зверху реалізовано навігаційну панель з логотипом «GreenEnergy» та

посиланнями на основну секцію зі всіма станціями, на секцію зі загальним графіком, різні сервіси та ресурси.

Для відображення загального графіка, де показано роботу всіх датчиків одночасно для візуалізації всієї енергосистеми, створено секцію «Total». На рисунку 3.31 показано загальний графік та секцію «Total».

Рисунок 3.31 – Загальний графік в секції «Total»

Графік демонструє показники датчиків станцій зеленої енергетики в реальному часі.

Отже, користувацький інтерфейс системи моніторингу зеленої енергетики відповідає критеріям простоти інтерактивності.

3.9 Тестування

Тестування системи моніторингу зеленої енергетики для ефективного балансування енергосистеми важливий аспект розробки. Воно допомагає перевірити надійність системи та знайти її слабкі місця.

3.9.1 Тестування Mosquitto

Щоб перевірити правильність роботи MQTT брокеру Mosquitto, потрібно скористатися командами «mosquitto_sub» та «mosquitto_pub».

Команда «mosquitto_pub» відповідає за публікацію повідомлення. Наприклад, команда «mosquitto_pub -t "sensor/topic" -m "1000"» відправляє повідомлення на MQTT порт з темою «sensor/topic» та показником «1000». На рисунку 3.32 показано роботу даної команди.

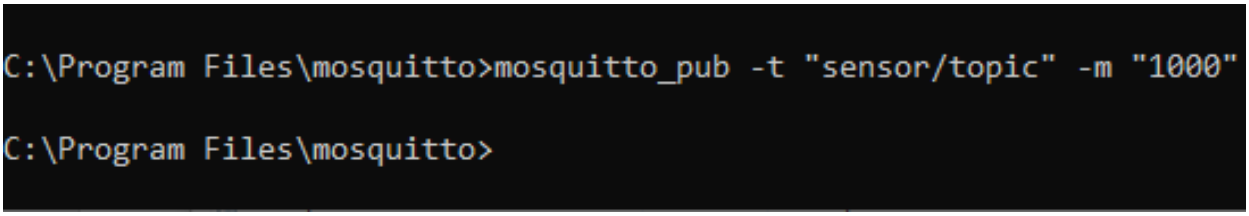

Рисунок 3.32 – Посилання повідомлення в Mosquitto

Для зчитування повідомлення використовується команда «mosquitto_sub», яка відповідає за підписку на тему. На рисунку 3.33 показано роботу команди «mosquitto_sub».

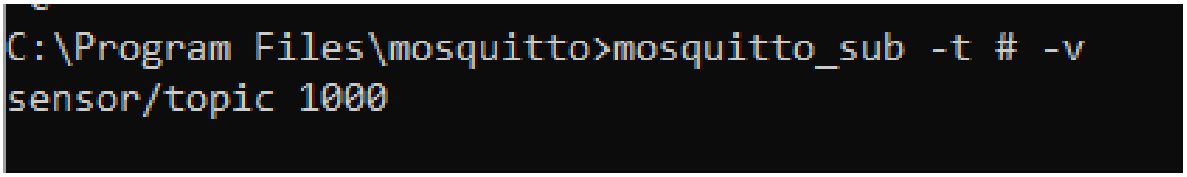

Рисунок 3.33 – Підписка на повідомлення в Mosquitto

Тег «-t» відповідає за підписку на топік (тему), а «-v» – за зчитування повідомлення повністю. Тепер Mosquitto може відстежувати всі повідомлення на порту 1883. Отже, брокер Mosquitto працює правильно.

3.9.2 Тестування Eclipse Paho Java Client

Для тестування MQTT-клієнта Eclipse Paho Java Client потрібно перевірити з'єднання з MQTT брокером та порівняти згенеровані повідомлення з повідомленнями на порту 1883, який відстежує брокер. На рисунку 3.34 показані згенеровані повідомлення за допомогою MQTT-клієнта Eclipse Paho в консолі середовища розробки Eclipse.

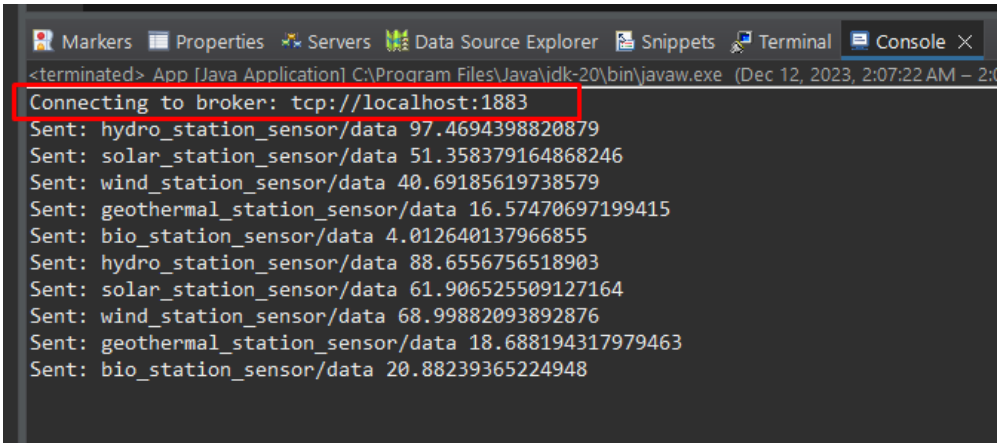

Рисунок 3.34 – Повідомлення Eclipse Paho

З рисунка видно, що MQTT-клієнт приєднався до брокера та згенеровані повідомлення з різними параметрами. На рисунку 3.35 показано підписку до порту 1883 за допомогою команди «mosquitto_sub» інструмента Mosquitto.

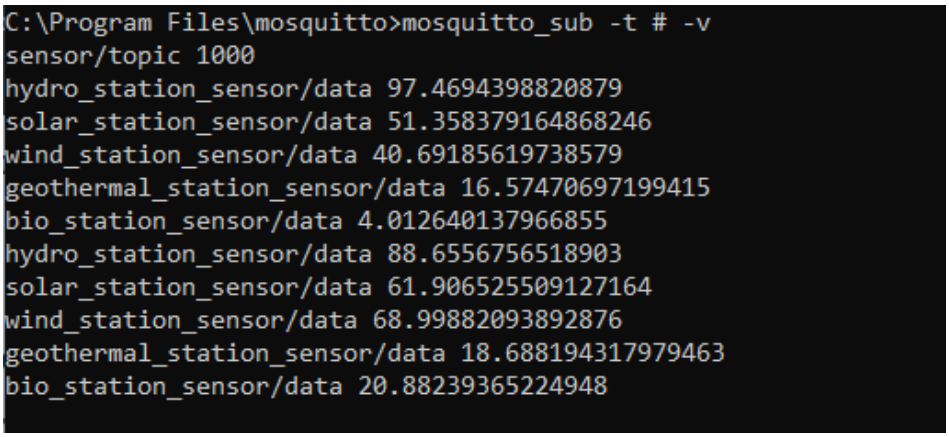

Рисунок 3.35 – Тестування повідомлень Eclipse Paho

Теми повідомлення та дані збігаються. Отже, MQTT-клієнта Eclipse Paho Java Client працює правильно.

3.9.3 Тестування потоку Node-RED

Для тестування потоку Node-RED потрібно порівняти дані, отримані від MQTT-клієнта Eclipse Paho, з даними, які надійшли до потоку Node-RED. Це можна зробити за допомогою вузла «debug».

На рисунку 3.36 видно останні повідомлення, відстежані MQTT брокером на порту 1883.

```
hydro_station_sensor/data 88.6556756518903
solar_station_sensor/data 61.906525509127164
wind_station_sensor/data 68.99882093892876
geothermal_station_sensor/data 18.688194317979463
bio_station_sensor/data 20.88239365224948
```

Рисунок 3.36 – Останні повідомлення

Вузол «debug» відображає дані потоку, на які підписано вузол «mqtt in». На рисунку 3.37 показано потік повідомлень для гідростанції.

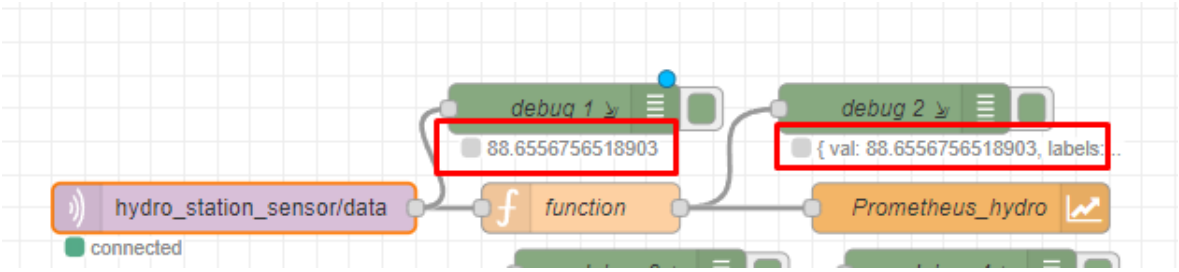

Рисунок 3.37 – Потік даних для гідростанції Node-RED

Значення показників в потоці Node-RED збігається зі значенням останніх повідомлень.

Вузол «debug» також формує журнал виконання, де відображаються показники потоку. На рисунку 3.38 показано журнал виконання потоків Node-RED.

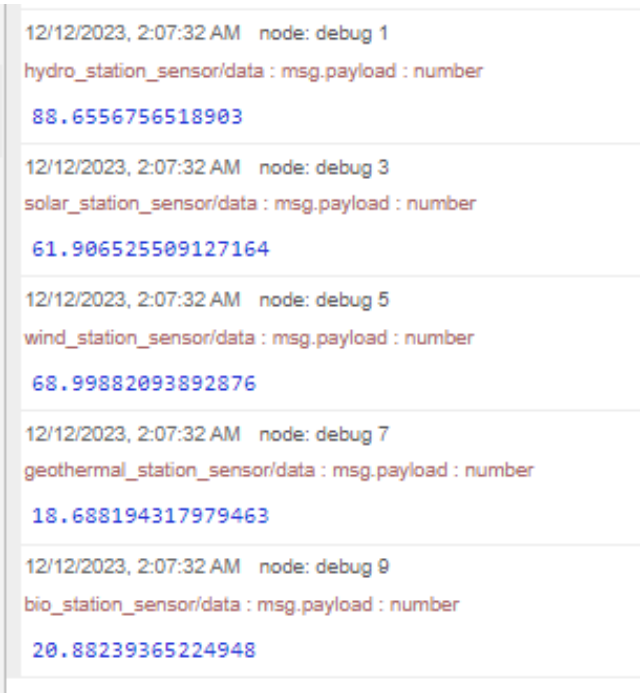

Рисунок 3.38 – Журнал виконання потоків Node-RED

Значення показників відповідають останнім показникам кожного датчика станцій зеленої енергетики. Отже, потік даних Node-RED працює правильно.

3.9.4 Тестування Prometheus та Grafana

Для тестування Prometheus та Grafana потрібно порівняти початкові дані з даними, які надійшли до цих інструментів.

Для прикладу взято останні показники гідростанції з інтервалом у 15 секунд та відрізком часу трохи понад дві хвилин. У таблиці 3.1 наведено перелік показників датчика гідростанції з інтервалом у 15 секунд та відрізком часу з 19:54:00 12.12.2023 до 19:56:15 12.12.2023.

Таблиця 3.1 – Показники гідростанції з інтервалом у 15 секунд та відрізком часу з 19:54:00 12.12.2023 до 19:56:15 12.12.2023

Номер	Показник
1	94.08311342
2	90.59315539
3	89.78113147
4	86.26447589
5	97.26721183
6	83.22276385
7	81.68390391
8	82.07553198
9	90.84333977
10	82.41288043

В Prometheus потрібно перейти в панель «Graph», де можна побачити дані у вигляді графіка. Потім потрібно вибрати метрики для гідростанції та встановити відрізок часу з 19:54:00 12.12.2023 до 19:56:15 12.12.2023. На рисунку 3.39 показано графік показників гідростанції у визначений період часу в середовище Prometheus.

Рисунок 3.39 – Показники гідростанції в середовище Prometheus

Як видно з рисунка, дані в Prometheus збігаються з початковими даними від датчика. Отже, Prometheus працює правильно.

У середовище Grafana з графіка можна отримати таблицю даних. На рисунку 3.40 показана таблиця показників гідростанції у визначений період часу з інтервалом у 15 секунд в середовище Grafana.

Time	{__name__="hydro_station_sensor_data",
2023-12-12 19:54:00	94.1
2023-12-12 19:54:15	90.6
2023-12-12 19:54:30	89.8
2023-12-12 19:54:45	86.3
2023-12-12 19:55:00	97.3
2023-12-12 19:55:15	83.2
2023-12-12 19:55:30	81.7
2023-12-12 19:55:45	82.1
2023-12-12 19:56:00	90.8
2023-12-12 19:56:15	82.4

Рисунок 3.40 – Таблиця показників гідростанції в середовище Grafana

Дані збігаються з початковими даними від датчика гідростанції. Показники округлені до десятків.

Отже, середовище Grafana працює правильно.

3.9.5 Тестування Spring Boot та вебінтерфейсу

Spring Boot реалізовує безпеку доступу до вебсторінки та обробку помилок. Для тестування авторизації потрібно перевірити її надійність та безпеку системи загалом.

Спочатку треба перевірити під час правильного вводу імені користувача та паролю. Для цього потрібно перейти за посиланням

«http://localhost:8080/greenEnergy». Далі у формі авторизації треба ввести логін та пароль «admin». Браузер завантажує інтерфейс. Все працює як потрібно.

Для перевірки на надійність потрібно ввести недійсний логін або пароль. На рисунку 3.41 показано роботу авторизації під час спроби ввести недійсний логін та пароль.

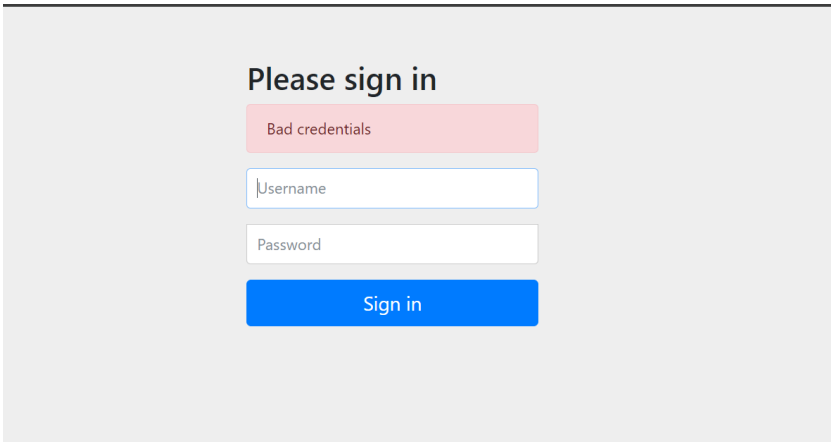

Рисунок 3.41 – Форма авторизації під час спроби ввести недійсний логін та пароль

Додаток блокує доступ до вебінтерфейсу системи та висвітлюється напис «Bad credentials», який значить, що були введені неправильні облікові дані. Авторизація працює правильно та забезпечує безпеку системи.

Для перевірки обробки помилок спочатку треба ввести неправильну електронну адресу в адресному рядку браузера. На рисунку 3.42 показано результат адреси «http://localhost:8080/NotTrue», що не існує.

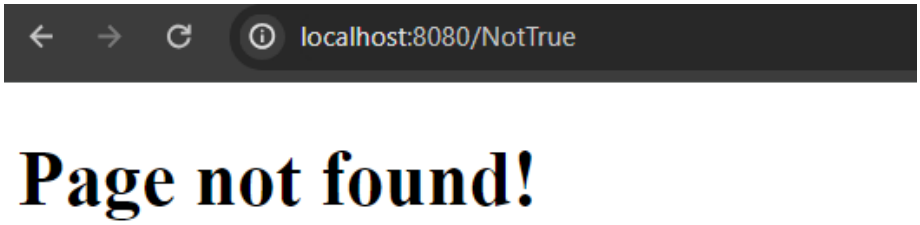

Рисунок 3.42 – Обробка помилки «Not Found»

Під час спроби ввести неправильну адресу браузер виводить текст «Page not found!».

Отже, Spring Boot, система безпеки та обробка помилок працює правильно.

Для тестування вебінтерфейсу спочатку потрібно перевірити консоль розробника на наявність помилок. Для цього на вебсторінці потрібно натиснути на клавішу «F12» і перейти на вкладку «Console». На рисунку 3.43 показана вкладка «Console».

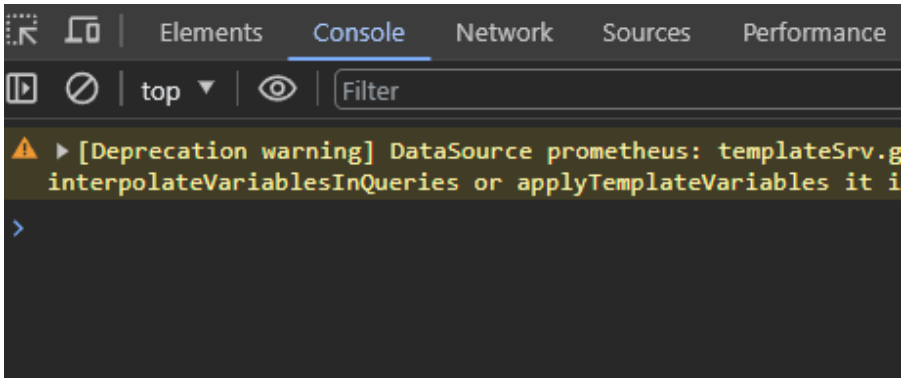

Рисунок 3.43 – Вкладка «Console»

У вкладці «Console» помилок немає.

Далі потрібно перейти на вкладку «Network» для перевірки мережевих запитів, зроблених вебсторінкою. На рисунку 3.44 показано вкладку «Network».

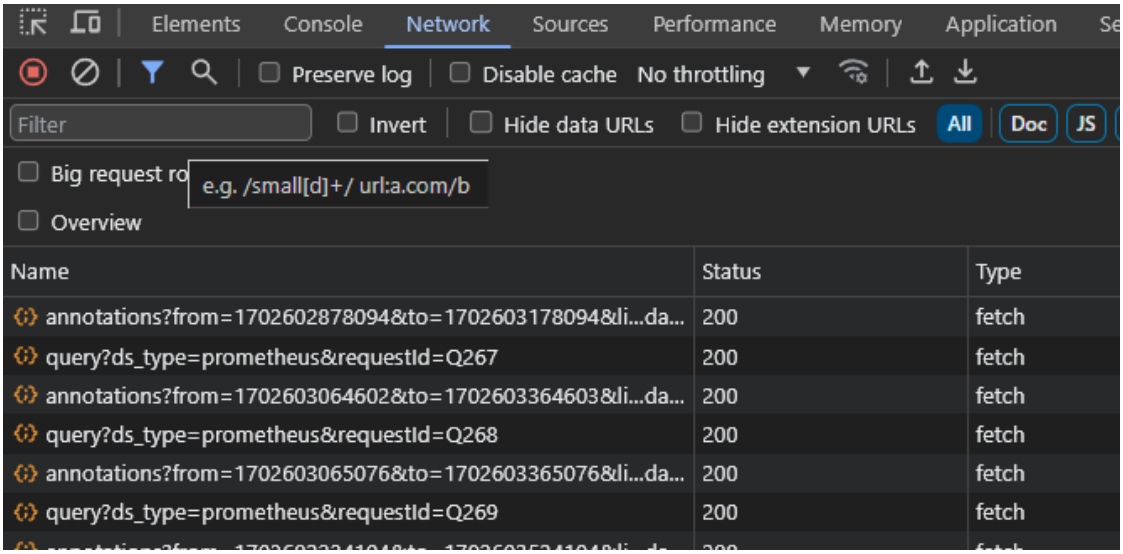

Рисунок 3.44 – Вкладка «Network»

У вкладці «Network» помилок не виявлено. Більшість мережевих запитів стосується з’єднання з Grafana. Їх статус «200», що демонструє успішність відповіді від сервісу Grafana. Тому можна зробити висновок, що графіки датчиків станцій зеленої енергетики працюють правильно. Отже, користувацький вебінтерфейс працює правильно та надає вірну інформацію.

Система зеленої енергетики для ефективного балансування енергосистеми пройшла тестування, кожні її складові працюють правильно та надають вірну інформацію.

ВИСНОВКИ

У кваліфікаційній магістерській роботі проаналізований стан зеленої енергетики в Україні та світі. Розглянуто види зеленої енергії, такі як гідроенергія, сонячна енергія, енергія вітру, геотермальна енергія та біоенергія.

Досліджено перспективи розвитку зеленої енергетики та впровадження нових інноваційних технологій в галузь ВДЕ, таких як системи IoT, хмарні обчислення, машинне навчання та інтерактивні системи моніторингу показників в режимі реального часу.

Розглянуто можливості моніторингу енергосистеми та проаналізовано наявні комплексні системи моніторингу енергетики, такі як ETAP EMS, Inavitas Platform, Quickbase та EnergyCAP. Також визначено їх переваги та недоліки.

Здійснений процес вибору та дослідження інструментів та технологій для розробки системи моніторингу зеленої енергетики.

Описано процеси налаштування протоколу MQTT для збору метрик з датчиків, зберігання показників через потік Node-RED в базі даних часових рядів Prometheus, створення графіків сервісом Grafana та побудови інтерактивного та захищеного користувацького інтерфейсу, який відображає моніторинг даних в режимі реального часу.

Проведено тестування та діагностику кожної частини системи.

Результатом є розроблена система моніторингу зеленої енергетики для ефективного балансування енергосистеми, яка може збирати показники датчиків станцій, опрацьовувати, направляти, зберігати та візуалізувати отримані дані.

ПЕРЕЛІК ПОСИЛАНЬ

1. What is renewable energy? [Електронний ресурс] / Режим доступу: https://www.sutori.com/en/story/a-brief-history-of-air-quality-sensors--jMm2LKEQZDWmXvsDMv2MP5LV – (дата звернення 13.10.2023). – Назва з екрана.
2. Phebe Asantewaa Owusu, Samuel Asumadu-Sarkodie, A review of renewable energy sources, sustainability issues and climate change mitigation. – Cogent Engineering, 2016. – 4 с.
3. Гідроенергетика [Електронний ресурс] / Режим доступу: https://uhe.gov.ua/diyalnist/gidroenergetika – (дата звернення 14.10.2023). – Назва з екрана.
4. Alex Campbell, Debbie Gray, 2022 Hydropower Status Report. – London: International Hydropower Association, 2022. – 7-14 с.
5. Schematic diagram of Hydroelectric power plant [Електронний ресурс] / Режим доступу: https://commons.wikimedia.org/wiki/File:Hydroelectric_dam.svg – (дата звернення 16.10.2023).
6. How Hydropower Works [Електронний ресурс] / Режим доступу: https://www.energy.gov/eere/water/types-hydropower-plants – (дата звернення 16.10.2023). – Назва з екрана.
7. Лановенко О.Г., Остапішина О.О., СЛОВНИК – ДОВІДНИК З ЕКОЛОГІЇ: Навчально-методичний посібник / О. Г. Лановенко, О. О. Остапішина. – Херсон: ПП Вишемирський В.С., 2013. – 162 с.
8. Бурячок Т.О., Дубовськой С.В., Плачков І.В., Електроенергетика та охорона навколишнього середовища. Функціонування енергетики в сучасному світі. – Енергетика: історія, сучасність і майбутнє, 2013. – 14с, 25с.
9. Як працює сонячна батарея? [Електронний ресурс] / Режим доступу: https://sun-energy.com.ua/image/catalog/statti/yak_pracuye_panel/yak_pracuye_soniachna_batareya-min.jpg – (дата звернення 20.10.2023).
10. The Future Of Solar Energy: Predictions For 2024 and Beyond [Електронний ресурс] / Режим доступу: https://arka360.com/ros/future-of-solar-energy-predictions-for-2023/ – (дата звернення 20.10.2023). – Назва з екрана.
11. Галузь сонячної енергетики в Україні [Електронний ресурс] / Режим доступу: https://www.ueex.com.ua/presscenter/news/galuz-sonyachnoi-energetiki-v-ukraini/ – (дата звернення 20.10.2023). – Назва з екрана.
12. Енергія вітру [Електронний ресурс] / Режим доступу: https://eenergy.com.ua/tag/wind-energy/ – (дата звернення 14.10.2023). – Назва з екрана.
13. Принцип роботи вітрогенератора [Електронний ресурс] / Режим доступу: https://vencon.ua/ua/articles/printsip-raboty-vetrogeneratora – (дата

звернення 14.10.2023). – Назва з екрана.

14. Геотермальна енергетика в Україні: вода чи надра? [Електронний ресурс] / Режим доступу: https://uz.ligazakon.ua/ua/magazine_article/EA014534 – (дата звернення 25.10.2023). – Назва з екрана.

15. Біоенергетика [Електронний ресурс] / Режим доступу: https://saee.gov.ua/uk/ae/bioenergy – (дата звернення 26.10.2023). – Назва з екрана.

16. Відновлювана енергетика та системи розосередженої генерації [Електронний ресурс] / Режим доступу: https://ep.kpi.ua/uk/node/24 – (дата звернення 26.10.2023). – Назва з екрана.

17. Electricity production from fossil fuels, nuclear and renewables,World [Електронний ресурс] / Режим доступу: https://ourworldindata.org/grapher/elec-fossil-nuclear-renewables?facet=metric – (дата звернення 26.10.2023).

18. Modern renewable energy generation by source, World [Електронний ресурс] / Режим доступу: https://ourworldindata.org/grapher/modern-renewable-prod?facet=none – (дата звернення 26.10.2023).

19. The Future of Renewable Energy: IT Solutions by Industry [Електронний ресурс] / Режим доступу: https://www.infopulse.com/blog/the-future-of-renewable-energy-it-solutions-by-industry – (дата звернення 26.10.2023). – Назва з екрана.

20. Зелена енергетика в Україні. Що відбувається? [Електронний ресурс] / Режим доступу: https://greenenergy.rbc.ua/– (дата звернення 27.10.2023)

21. “Зелена” енергетика [Електронний ресурс] / Режим доступу: https://ukraineinvest.gov.ua/incentives/green-energy/– (дата звернення 27.10.2023). – Назва з екрана.

22. Наскільки постраждала через війну "зелена" енергетика в Україні [Електронний ресурс] / Режим доступу: https://texty.org.ua/fragments/109722/naskilky-postrazhdala-cherez-vijnu-zelena-enerhetyka-v-ukrayini/– (дата звернення 27.10.2023). – Назва з екрана.

23. What is grid balancing? And how does it help the climate? [Електронний ресурс] / Режим доступу: https://www.trueenergy.io/blog/what-is-grid-balancing/– (дата звернення 28.10.2023). – Назва з екрана.

24. How do you balance renewable energy grid integration with other objectives? What is grid balancing? And how does it help the climate? [Електронний ресурс] / Режим доступу: https://www.linkedin.com/advice/1/how-do-you-balance-renewable-energy-grid-integration– (дата звернення 28.10.2023). – Назва з екрана.

25. Why is Energy Monitoring Important? [Електронний ресурс] / Режим доступу: https://harksys.com/solutions/energy-monitoring/why-is-energy-monitoring-so-important/ – (дата звернення 29.10.2023). – Назва з екрана.

26. S. Saly, K. Signer, A. Sullivan, Power System Monitoring. – Brown Boveri & Co., Ltd., Baden, Switzerland. – 189 с.

27. MONITORING OF ELECTRICAL NETWORKS [Електронний ресурс] / Режим доступу: https://supervision-clever.fr/monitoring-of-electrical-networks/ – (дата звернення 29.10.2023). – Назва з екрана.

28. ETAP [Електронний ресурс] / Режим доступу: https://etap.com/ – (дата звернення 30.10.2023).

29. Real-Time Energy Intelligence Platform for Businesses [Електронний ресурс] / Режим доступу: https://info.gartnerdigitalmarkets.com/inavitas-gdm-lp/?category=energy-management&utm_source=capterra – (дата звернення 30.10.2023). – Назва з екрана.

30. Solutions for energy monitoring and management from inavitas Energy Intelligence [Електронний ресурс] / Режим доступу: https://www.inavitas.com/ – (дата звернення 30.10.2023).

31. Quickbase [Електронний ресурс] / Режим доступу: https://www.quickbase.com/ – (дата звернення 31.10.2023).

32. About EnergyCAP [Електронний ресурс] / Режим доступу: https://www.softwareadvice.com/cafm/energycap-profile/ – (дата звернення 31.10.2023). – Назва з екрана.

33. EnergyCAP [Електронний ресурс] / Режим доступу: https://www.energycap.com/ – (дата звернення 31.10.2023).

34. Dhairya Parikh, Raspberry Pi and MQTT Essentials. – Packt Publishing Ltd., UK, – 2022. – 4 с.

35. MQTT Protocol Description [Електронний ресурс] / Режим доступу: https://support.smart-maic.com/en/knowledge-bases/2/articles/41-mqtt-protocol-description – (дата звернення 01.11.2023).

36. Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry. – International Technical Support Organization, – 2012. – 14 с.

37. Eclipse Foundation [Електронний ресурс] / Режим доступу: https://www.eclipse.org/ – (дата звернення 02.11.2023).

38. Cedalo - Eclipse Mosquitto MQTT Broker [Електронний ресурс] / Режим доступу: https://developer.community.boschrexroth.com/t5/image/serverpage/image-id/13339i02334993E38BBD81/image-size/original?v=v2&px=-1 – (дата звернення 31.10.2023).

39. Mosquitto [Електронний ресурс] / Режим доступу: https://mosquitto.org/ – (дата звернення 03.11.2023).

40. Eclipse Mosquitto [Електронний ресурс] / Режим доступу: https://projects.eclipse.org/projects/iot.mosquitto – (дата звернення 03.11.2023). – Назва з екрана.

41. Mosquitto MQTT Broker: Pros/Cons, Tutorial, and a Modern Alternative [Електронний ресурс] / Режим доступу: https://www.emqx.com/en/blog/mosquitto-mqtt-broker-pros-cons-tutorial-and-modern-alternatives – (дата звернення 03.11.2023). – Назва з екрана.

42. Eclipse Paho Java Client [Електронний ресурс] / Режим доступу: https://eclipse.dev/paho/index.php?page=clients/java/index.php – (дата звернення 04.11.2023). – Назва з екрана.

43. Raghuram Bharathan, Apache Maven Cookbook. – Packt Publishing, – 2015.

– 1 с.

44. Apache Maven [Електронний ресурс] / Режим доступу: https://www.geeksforgeeks.org/apache-maven/ – (дата звернення 043.11.2023). – Назва з екрана.

45. Maven Architecture [Електронний ресурс] / Режим доступу: https://www.w3schools.blog/wp-content/uploads/2018/03/maven-architecture2.png – (дата звернення 04.11.2023).

46. Node-RED [Електронний ресурс] / Режим доступу: https://nodered.org/about/ – (дата звернення 05.11.2023).

47. Node-RED - free open source tool for wiring things together [Електронний ресурс] / Режим доступу: https://evocean.com/products/node-red/ – (дата звернення 04.11.2023). – Назва з екрана.

48. Brian Brazil, Prometheus: Up & Running. – Robust Perception Ltd, USA – 2018. – 3 с.

49. What is Prometheus? [Електронний ресурс] / Режим доступу: https://prometheus.io/docs/introduction/overview/ – (дата звернення 06.11.2023). – Назва з екрана.

50. What is Prometheus: What is Prometheus, how to use it and challenges of scaling Prometheus [Електронний ресурс] / Режим доступу: https://last9.io/blog/what-is-prometheus/ – (дата звернення 06.11.2023). – Назва з екрана.

51. A Comprehensive Guide to Prometheus Monitoring [Електронний ресурс] / Режим доступу: https://www.weave.works/blog/a-comprehensive-guide-to-prometheus-monitoring – (дата звернення 06.11.2023). – Назва з екрана.

52. What Is a Time Series Database and Why Do I Need One? [Електронний ресурс] / Режим доступу: https://tdengine.com/what-is-a-time-series-database/ – (дата звернення 20.11.2023). – Назва з екрана.

53. Relational Databases vs Time Series Databasescc [Електронний ресурс] / Режим доступу: https://www.influxdata.com/blog/relational-databases-vs-time-series-databases/ – (дата звернення 20.11.2023). – Назва з екрана.

54. DB-Engines Ranking - Trend of Time Series DBMS Popularity [Електронний ресурс] / Режим доступу: https://db-engines.com/en/ranking_trend/time+series+dbms – (дата звернення 20.11.2023).

55. Grafana, програмне забезпечення з відкритим кодом для аналізу та моніторингу [Електронний ресурс] / Режим доступу: https://ubunlog.com/uk/нагляд-за-аналізом-програмного-забезпечення-grafana/ – (дата звернення 21.11.2023). – Назва з екрана.

56. What is Grafana? [Електронний ресурс] / Режим доступу: https://www.redhat.com/en/topics/data-services/what-is-grafana – (дата звернення 21.11.2023). – Назва з екрана.

57. Spring Framework [Електронний ресурс] / Режим доступу: https://spring.io/projects/spring-framework – (дата звернення 22.11.2023). – Назва з екрана.

58. Iuliana Cosmina, Rob Harrop, Chris Schaefer, Clarence Ho, Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools. – Venice, Florida, USA, – 2017. – 2 с.

59. Loredana Crusoveanu, Intro to Inversion of Control and Dependency Injection with Spring, 2023 [Електронний ресурс] / Режим доступу: https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring – (дата звернення 22.11.2023). – Назва з екрана.

60. Devlin Basilan Duldulao, Seiji Ralph Villafranca, Spring Boot and Angular. – Packt Publishing Ltd., Birmingham, UK, – 2022. – 4 с.

61. Sufyan bin Uzayr, HTML: The Ultimate Guide. – 4 Park Square, Milton Park, Abingdon, Oxon, – 2023. – 7 с.

62. Martine Dowden, Michael Gearon, Tiny CSS Projects. – Manning Publications Co., Shelter Island, NY, USA, – 2023. – 3 с.

63. Eric Sarrion, JavaScript from Frontend to Backend. – Packt Publishing, Birmingham, UK, – 2022. – 4 с.

ДОДАТОК А

Код програми MQTT-клієнта Eclipse Paho Java Client

```
package energy.Energy_System;

import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;


public class App
{

        public static void main(String[] args) {
    String broker = "tcp://localhost:1883";
    String clientId = "SensorEmulator";
    String topic = "sensor/data";

    //позначаємо датчики станцій зеленої енргетики
    String hydroStationTopic = "hydro_station_sensor/data";
    String solarStationTopic = "solar_station_sensor/data";
    String windStationTopic = "wind_station_sensor/data";
    String geothermalStationTopic = "geothermal_station_sensor/data";
    String bioStationTopic = "bio_station_sensor/data";

    //емулюємо запити від датчиків
    try {
        System.out.println("Connecting to broker: " + broker);
       MqttClient client = new MqttClient(broker, clientId);
       client.connect();

       while (true) {
           String message = " ";

           //HydroStation
           message = sensorData(80, 100);
          MqttMessage mqttMessageHydro = new MqttMessage(message.getBytes());
          client.publish(hydroStationTopic, mqttMessageHydro);
          System.out.println("Sent: "+ hydroStationTopic + " " + message);

          //SolarStation
          message = sensorData(50, 80);
          MqttMessage mqttMessageSolar = new MqttMessage(message.getBytes());
```

```
            client.publish(solarStationTopic, mqttMessageSolar);
            System.out.println("Sent: "+ solarStationTopic + " " + message);

            //WindStation
            message = sensorData(40, 70);
            MqttMessage mqttMessageWind = new MqttMessage(message.getBytes());
            client.publish(windStationTopic, mqttMessageWind);
            System.out.println("Sent: "+ windStationTopic + " " + message);

            //GeothermalStation
            message = sensorData(10, 30);
            MqttMessage mqttMessageGeothermal = new
MqttMessage(message.getBytes());
            client.publish(geothermalStationTopic, mqttMessageGeothermal);
            System.out.println("Sent: "+ geothermalStationTopic + " " + message);

            //BioStation
            message = sensorData(2, 20);
            MqttMessage mqttMessageBio = new MqttMessage(message.getBytes());
            client.publish(bioStationTopic, mqttMessageBio);
            System.out.println("Sent: "+ bioStationTopic + " " + message);

            // Очікування 5 секунд перед наступною відправкою даних
            Thread.sleep(5000);

        }

    } catch (MqttException | InterruptedException e) {
        e.printStackTrace();
    }
  }

        //Генерація даних для вузлів станцій
        private static String sensorData(int min, int max) {
         int range = (max - min) + 1;
            return String.valueOf((Math.random() * range) + min);
        }

}
```

ДОДАТОК Б

Код класу «WebSecurityConfig»

```
package com.dden.greenEnergy.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig {

   @Bean
   public UserDetailsService userDetailsService(){
      return new InMemoryUserDetailsManager(
            User.builder()
                  .username("admin")
                  .password(passwordEncoder()
                        .encode("admin"))
                  .roles("ADMIN")
                  .build()
      );
   }

   @Bean
   public PasswordEncoder passwordEncoder(){
      return new BCryptPasswordEncoder(12);
   }
}
```

ДОДАТОК В

Повний код візуальної частини вебсторінки (файл «index.html»)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Energy Monitoring System</title>
    <!--Стилі об'єктів та елементів-->
    <style>
        body {
            margin: 0;
            padding: 0;
            background-color: #000;
            color: #fff;
            font-family: Arial, sans-serif;
        }
        nav {
            background-color: #000;
            padding: 15px;
            text-align: center;
        }
        nav img {
            max-height: 50px;
            vertical-align: middle;
        }
        nav a {
            color: #FFA500;
            text-decoration: none;
            margin: 0 15px;
            font-weight: bold;
        }
        nav a:hover {
            color: #fff;
        }
        #map {
            width: 100%;
            height: 700px;
            background-image: url('images/zyro-image.png');
            background-size: contain;
            background-repeat: no-repeat;
            background-position: center;
```

```
            position: relative;
        }
        .station {
            width: 40px;
            height: 40px;
            background-size: cover;
            position: absolute;
            cursor: pointer;
        }
        .graph-container {
            position: absolute;
            top: 0;
            left: 100%;
            margin-left: 10px;
            display: none;
            background-color: #fff;
            padding: 5px;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
        }
        #total-section {
            text-align: center;
            padding: 40px;
            background-color: #1212125b;
            color: #FFA500;
        }
        #cop-section {
            text-align: left;
            padding: 40px;
            background-color: #0000005b;
            color: #ffa600;
        }
    </style>
</head>

<body>
<!--Навігаційна панель-->
    <nav>
        <img src="images/your-logo.jpg" alt="Logo" height="30px">
        <a href="#">Stations</a>
        <a href="#">Total</a>
        <a href="#">Services</a>
        <a href="#">About</a>
    </nav>
```

```
    <div id="map">

    </div>

    <div class="graph-container">
    </div>
<!--Секція загального графіка-->
    <div id="total-section">
        <h2>Total</h2>
        <iframe src="http://localhost:3000/d-solo/bb259cc3-d818-4a2c-ab9c-3a0127c39c69/greenenergy?orgId=1&refresh=auto&theme=dark&panelId=6" width="950" height="350" frameborder="0"></iframe>
    </div>

    <div id="cop-section">
        <p>GreenEnergy by Denys Parshutkin</p>
    </div>
<!--JavaScript скрипт-->
    <script>
        // Позначаємо графіки
        const stationCoordinates = [
            { left: 720, top: 250, grafanaUrl: "http://localhost:3000/d-solo/bb259cc3-d818-4a2c-ab9c-3a0127c39c69/greenenergy?orgId=1&refresh=auto&theme=dark&panelId=1", icon: "images/hidro.jpg" },
            { left: 710, top: 420, grafanaUrl: "http://localhost:3000/d-solo/bb259cc3-d818-4a2c-ab9c-3a0127c39c69/greenenergy?orgId=1&refresh=auto&theme=dark&panelId=2", icon: "images/solar.jpg" },
            { left: 880, top: 350, grafanaUrl: "http://localhost:3000/d-solo/bb259cc3-d818-4a2c-ab9c-3a0127c39c69/greenenergy?orgId=1&refresh=auto&theme=dark&panelId=3", icon: "images/wind.jpg" },
            { left: 380, top: 320, grafanaUrl: "http://localhost:3000/d-solo/bb259cc3-d818-4a2c-ab9c-3a0127c39c69/greenenergy?orgId=1&refresh=auto&theme=dark&panelId=4", icon: "images/geo.jpg" },
            { left: 470, top: 200, grafanaUrl: "http://localhost:3000/d-solo/bb259cc3-d818-4a2c-ab9c-3a0127c39c69/greenenergy?orgId=1&refresh=auto&theme=dark&panelId=5", icon: "images/bio.jpg" },
        ];

        const mapContainer = document.getElementById('map');
```

```
        stationCoordinates.forEach(station => {
            const stationElement = document.createElement('div');
            stationElement.className = 'station';
            stationElement.style.left = `${station.left}px`;
            stationElement.style.top = `${station.top}px`;
            stationElement.style.backgroundImage = `url(${station.icon})`;

            // Додаємо обробник події при наведенні курсору
            stationElement.addEventListener('mouseover', () => {
                // Створюємо контейнер для графіка
                const graphContainer = document.createElement('div');
                graphContainer.className = 'graph-container';

                // Створюємо iframe із графіком Grafana
                const iframe = document.createElement('iframe');
                iframe.src = station.grafanaUrl;
                iframe.width = "450";
                iframe.height = "200";
                iframe.frameBorder = "0";

                // Додаємо iframe у контейнер для графіка
                graphContainer.appendChild(iframe);

                // Показуємо контейнер
                graphContainer.style.display = 'block';

                // Показуємо контейнер поряд зі станцією
                graphContainer.style.top = `${station.top}px`;
                graphContainer.style.left = `${station.left + 40}px`;

                // Додаємо контейнер до карти
                mapContainer.appendChild(graphContainer);
            });

            // Додаємо обробник події під час відходу курсору
            stationElement.addEventListener('mouseout', () => {
                // Видаляємо всі контейнери з графіками
                const graphContainers = document.querySelectorAll('.graph-container');
                graphContainers.forEach(container => container.remove());
            });
            mapContainer.appendChild(stationElement);
        });
    </script>
</body>
</html>
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

Державний університет інформаційно-комунікаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

“РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ЗЕЛЕНОЇ ЕНЕРГЕТИКИ ДЛЯ ЕФЕКТИВНОГО БАЛАНСУВАННЯ ЕНЕРГОСИСТЕМИ”

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Виконав: здобувач вищої освіти гр. ІСДМ-62
Денис ПАРШУТКІН
Керівник: доцент каф. ВММФ
Сергій СІМЧЕНКО

Київ - 2023

- ***Актуальність теми:*** необхідність розв’язання проблеми контролю нестабільності виробництва енергетики, ефективного балансування та впровадження гнучкості енергосистеми, покращення ефективності використання енергетичних ресурсів під час швидкого зростання частки зеленої енергетики у світі. Після початку війни в Україні створення системи моніторингу зеленої енергетики може бути корисною під час можливих руйнувань інфраструктури для постійного спостереження та швидкого реагування.
- ***Об`єкт дослідження:*** процес створення системи моніторингу зеленої енергетики.
- ***Предмет дослідження:*** система моніторингу зеленої енергетики.
- ***Мета дослідження:*** розробка системи моніторингу зеленої енергетики для забезпечення ефективного балансування енергосистеми.
- ***Завдання дослідження:***
  - здійснити аналіз стану та розвитку зеленої енергетики;
  - дослідити та проаналізувати наявні системи моніторингу енергетики;
  - вибрати та описати технології для побудови системи моніторингу зеленої енергетики;
  - розробити систему моніторингу зеленої енергетики для ефективного балансування енергосистеми.

2

Зелена енергетика

Відновлюванні джерела енергії охоплюють гідроенергію, сонячну енергію, енергію вітру, геотермальну енергію, біоенергію та інші.

Рисунок 3.1 – Зелена енергетика

Розвиток зеленої енергетики

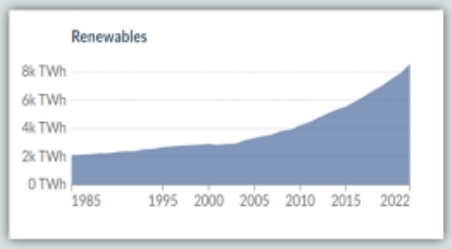

Рисунок 4.1 – Загальний графік зростання ВДЕ

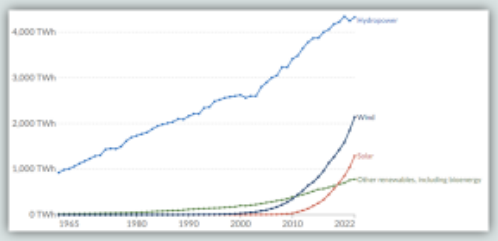

Рисунок 4.2 – Графік зростання окремих видів зеленої енергії

Важливість балансування енергосистеми

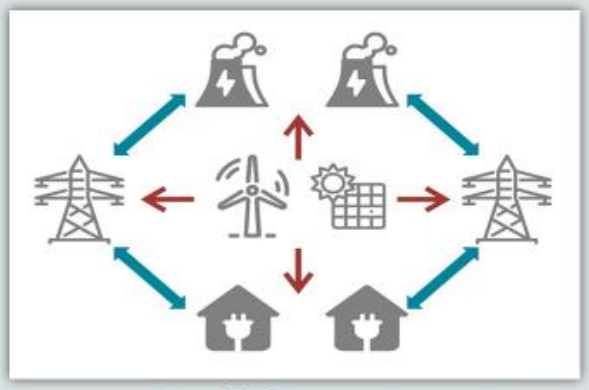

Рисунок 5.1 – Балансування енергосистеми

Огляд технологій

- Протокол MQTT
- MQTT брокер Eclipse Mosquitto
- MQTT-клієнт Eclipse Paho Java Client
- Maven
- Node-RED
- Prometheus.TSDB
- Grafana
- Spring Boot
- HTML, CSS, JavaScript

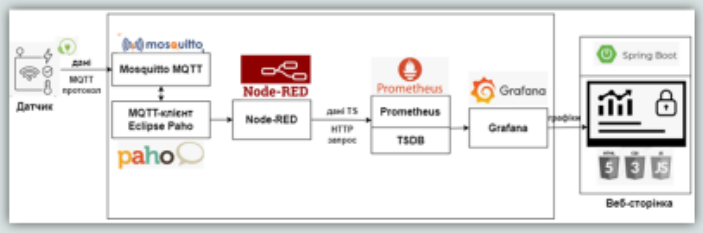

Рисунок 6.1 – Схема роботи системи моніторингу зеленої енергетики зі стеком технологій

Розробка системи моніторингу зеленої енергетики для ефективного балансування енергосистеми

Рисунок 7.1 – Загальна схема роботи системи моніторингу зеленої енергетики

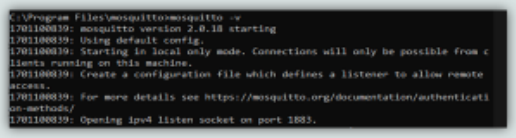

Рисунок 8.1 – Запуск Mosquitto в командному рядку Windows

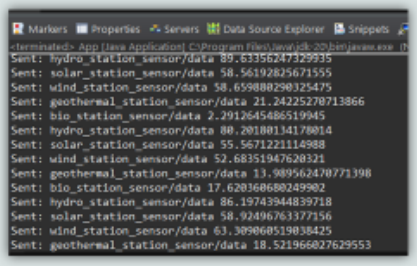

Рисунок 8.2 – Робота консолі програми Eclipse Paho Java Client

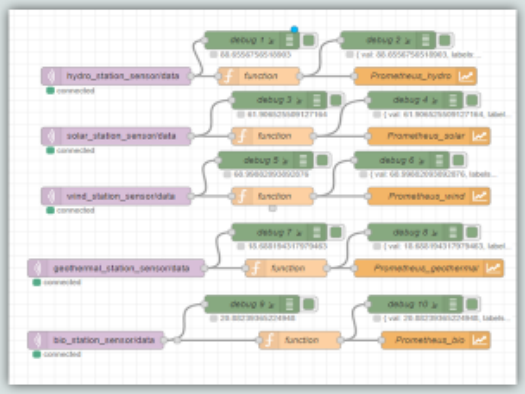

Рисунок 8.3 – Робота потоків повідомлень для станцій зеленої енергетики у Node-RED

Рисунок 9.1 – Графіки роботи для кожної станції в середовище Grafana

```
@Controller
public class WebErrorController implements ErrorController {
    @RequestMapping("/error")
    public String handleError(HttpServletRequest request){
        Object status = request.getAttribute(RequestDispatcher.ERROR_STATUS_CODE);
        //Recognizing the status code
        if(status != null &&
                Integer.valueOf(status.toString()) == HttpStatus.NOT_FOUND.value()){
            return "404";
        }
        return "error";
    }
}
```

Рисунок 9.2 – Контролер опрацювання помилок

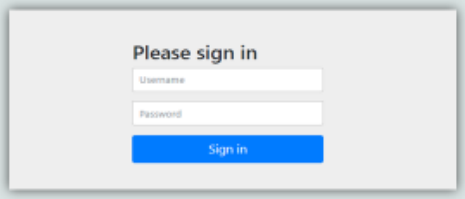

Рисунок 9.3 – Форма авторизації

Рисунок 10.1 – Вебінтерфейс системи

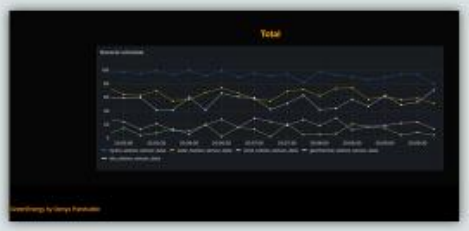

Рисунок 10.2 – Загальний графік в секції «Total»

Тестування

```
C:\Program Files\mosquitto>mosquitto_pub -t "sensor/topic" -m "1000"

C:\Program Files\mosquitto>
```

Рисунок 11.1 – Посилання повідомлення в Mosquitto

```
C:\Program Files\mosquitto>mosquitto_sub -t # -v
sensor/topic 1000
```

Рисунок 11.2 – Підписка на повідомлення в Mosquitto

Рисунок 11.3 – Показники станції в середовище Prometheus

Рисунок 11.4 – Форма авторизації під час спроби ввести недійсний логін та пароль

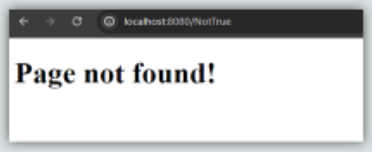

Рисунок 11.5 – Обробка помилки «Not Found»

ВИСНОВКИ

- У кваліфікаційній магістерській роботі досліджені перспективи розвитку зеленої енергетики та системи моніторингу.
- Проаналізовані наявні комплексні системи моніторингу енергетики, знайдено їх переваги та недоліки.
- Здійснений процес планування вибору та дослідження інструментів та технологій для розробки системи моніторингу зеленої енергетики.
- Розроблена система моніторингу зеленої енергетики для ефективного балансування енергосистеми базується на використанні протоколу MQTT для збору метрик з датчиків, які опрацьовуються за допомогою Prometheus, побудові графіків сервісом Grafana та створені інтерактивного користувацького інтерфейсу.

Апробація результатів дослідження:

Паршуткін Д.О., Теплюк О.В., Шапкін В.А. «Аналіз сучасних технологій розширення реляційних СУБД для роботи з часовими рядами». Наукова стаття у загальногалузевому науково-виробничому журналі «Зв'язок», м.Київ – випуск листопад-грудень №6, 2023.

ДЯКУЮ ЗА УВАГУ!