**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Сумський державний університет**
**Факультет електроніки та інформаційних технологій**
**Кафедра інформаційних технологій**

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

# КВАЛІФІКАЦІЙНА РОБОТА

**на здобуття освітнього ступеня магістр**

зі спеціальності 122 «Комп'ютерні науки»,

освітньо- професійної програми «Інформаційні технології проектування»

на тему: Реінжиніринг інформаційної системи підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії

Здобувача групи ІТ.м-32 Захарченко Іван Петрович
(шифр групи) (прізвище, ім'я, по батькові)

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

_____ Іван ЗАХАРЧЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник к.т.н., доцент Віра ШЕНДРИК _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

**Суми – 2024**

**Сумський державний університет**
**Факультет** електроніки та інформаційних технологій
**Кафедра** інформаційних технологій
**Спеціальність** 122 «Комп'ютерні науки»
**Освітньо-професійна програма** «Інформаційні технології проектування»

**ЗАТВЕРДЖУЮ**

В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО

«_____» _____ 2024 р.

# ЗАВДАННЯ

**на кваліфікаційну роботу магістра студентові**

***Захарченко Іван Петрович***
**(прізвище, ім'я, по батькові)**

**1 Тема кваліфікаційної роботи** Реінжиніринг інформаційної системи підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії.

затверджена наказом по університету від «11» жовтня 2024 р. № 1044-VI

**2 Термін здачі студентом кваліфікаційної роботи** «6» грудня 2024 р.

**3 Вхідні дані до кваліфікаційної роботи** перелік вимог до реінжинірингу інформаційної системи, доступ до даних та програмного коду наявної інформаційної ситеми підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії.

**4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)** аналіз предметної області, постановка задачі та методи дослідження, моделювання/проєктування об'єкта дослідження, практична реалізація реінжинірингу інформаційної ситеми підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії.

**5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації)** актуальність, постановка задачі, аналіз поточної інформаційної системи, огляд аналогічних інформаційних систем, таблиця порівняння аналогів, вимоги до реінжинірингу, засоби реалізації, структурно-функціональне моделювання, діаграма

класів, моделювання варіантів використання системи, демонстрація роботи інформаційної системи, тестування, висновок.

**6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:**

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

## КАЛЕНДАРНИЙ ПЛАН

| № п/п | Назва етапів кваліфікаційної роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної області | 19.08.24 – 27.08.24 | |
| 2 | Визначення мети та задач дослідження | 28.08.24 – 31.08.24 | |
| 3 | Проєктування об’єкта дослідження | 01.09.24 – 13.09.24 | |
| 4 | Практична реалізація проєкту | 14.09.24 – 22.11.24 | |
| 5 | Оформлення пояснювальної записки | 23.11.24 – 01.12.24 | |
| 6 | Створення презентації | 02.12.24 – 06.12.24 | |
| 7 | Захист дипломного проекту | 12.12.24 | |

Магістрант _____ Іван ЗАХАРЧЕНКО

Керівник роботи _____ к.т.н., доц. Віра ШЕНДРИК

# АНОТАЦІЯ

Тема кваліфікаційної роботи магістра «Реінжиніринг інформаційної системи підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 46 найменувань, додатків. Загальний обсяг роботи − 114 сторінок, у тому числі 74 сторінок основного тексту, 6 сторінок списку використаних джерел, 36 сторінок додатків.

Актуальність роботи зумовлена необхідністю адаптації до зростаючих вимог сучасної енергетики, що включає забезпечення стабільності, автономності та екологічності енергопостачання. Вдосконалення існуючої інформаційної системи підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії дозволить підвищити її гнучкість, масштабованість і надійність, сприяючи ефективному використанню відновлювальних джерел енергії та децентралізації енергетичних процесів, що є особливо важливим в умовах сучасних глобальних та локальних викликів.

Мета роботи полягає у проведенні реінжинірингу інформаційної системи для управління енергетичними мікромережами з відновлювальними джерелами енергії з використанням об'єктно-орієнтованого програмування та патерну MVC. Це сприятиме підвищенню модульності та масштабованості системи, спрощенню її підтримки, а також покращенню структурованості та надійності.

Результатом проведеної роботи є інформаційна система підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії, що забезпечує виконання поставлених задач.

Ключові слова: інформаційна система, вебзастосунок, реінжиніринг, мікромережа.

# ЗМІСТ

# ВСТУП

Використання відновлювальних джерел енергії є надзвичайно важливим інструментом у боротьбі з кліматичними змінами, вплив яких людство відчуває все гостріше з кожним роком. Зростаюча доступність для використання цих технологій робить можливим перехід до екологічно чистих джерел енергії для широких верств населення. Мікромережі з використанням відновлювальних джерел енергії, здатні забезпечувати автономне енергопостачання для окремих підприємств, господарств чи житлових будинків. Вони можуть працювати незалежно від централізованих мереж, що робить енергопостачання більш надійним та стабільним. Крім того, децентралізована система енергопостачання, що не залежить від надходження викопних енергоресурсів, є найкращим рішенням під час надзвичайних ситуацій, що є надзвичайно актуальним для енергетичного сектору України.

Роботи з впровадження використання мікромереж з відновлювальними джерелами енергії вже активно ведуться, зокрема в тому числі в Сумському державному університеті. Для моніторингу та управління цією мікромережою було розроблено веборієнтовану інформаційну систему підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії (ВДЕ) [1]. Проте існуюча розробка потребує реінжинірингу в частині архітектурної побудови, що зробить систему більш гнучкою та надійною.

**Об’єкт дослідження.** Процес реінжинірингу інформаційної системи підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії.

**Предмет дослідження.** Веборієнтована інформаційна система підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії.

**Мета –** забезпечити модульність та масштабованість системи, що дозволить зробити її більш структурованою і надійною, за рахунок реінжинірингу інформаційної системи підтримки управління енергетичними мікромережами з

відновлювальними джерелами енергії шляхом застосування методології об'єктно-орієнтованого програмування.

**Задачі кваліфікаційної роботи магістра:**

- провести літературний огляд предметної області розроблення інформаційних систем для енергетичних мікромереж;
- проаналізувати поточну інформаційну систему підтримки прийняття рішень при управлінні енергетичними мікромережами з відновлюваними джерелами енергії, визначити задачі з реінжинірингу;
- проаналізувати сучасні архітектурні рішення та фреймворки розроблення веборієнтованих інформаційних систем;
- виконати проєктування та моделювання інформаційної системи;
- реалізувати реінжиніринг інформаційної системи та провести її тестування.

**Практичне значення.** Інформаційна система підтримки управління енергетичними мікромережами з ВДЕ у результаті реінжинірингу стала більш структурованою завдяки впровадженню об'єктно-орієнтованого підходу, використання фреймворка Laravel та архітектури MVC. Це забезпечило підвищення модульності та масштабованості системи, спрощення її підтримки, покращення читабельності коду та прискорення процесів розробки й тестування, що в свою чергу позитивно вплинуло на продуктивність системи щодо виконання її завдань з моніторингу і контролю мікромережею.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Аналіз поточного стану веборієнтованих інформаційних систем підтримки та управлінням електромереж

Використання інформаційних систем для підтримки прийняття рішень відіграє важливу роль в розвитку будь-якої галузі. Наприклад, в електроенергетиці такі системи є ключовим інструментом для забезпечення ефективної та надійної роботи мереж. Ці системи допомагають збирати, аналізувати, обробляти та відображати дані, які пов’язані зі станом електричної мережі, поточними показниками енергоспоживання, станом електрообладнання тощо. Крім того, компанії можуть застосовувати отриману інформацію для прогнозування виникнення проблем чи аварій, що підвищує ефективність роботи та зменшує витрати на обслуговування та ремонт.

ВДЕ стають ключовим напрямком розвитку альтернативної енергетики. Енергетичні мікромережі дозволяють більш ефективно використовувати ВДЕ як для промислових, так і для побутових споживачів, які самостійно володіють енергетичними установками. Ці споживачі потребують спеціального програмного забезпечення для моніторингу, прогнозування та управління енергетичними ресурсами, завдяки якому забезпечується прозорість усіх процесів і створюється атмосфера довіри між учасниками ринку. Таким чином, використовуючи інформаційні системи, споживачі можуть контролювати свої енергетичні витрати та оптимізувати використання енергії для максимальної ефективності мікромереж.

Мікромережа – це автономна електрична мережа, яка дозволяє генерувати власну електроенергію та використовувати її, коли вона найбільше потрібна. Впровадження та розвиток автономних мікромереж необхідна складова для розвитку та відновлення українського енергосектору. Вже зараз триває робота над новими

моделями генерації та розподілу енергії, тому розробка та дослідження таких мереж має важливе значення [2].

Реалізація інтерфейсу та доступу до функціоналу інформаційних систем у вигляді вебзастосунків, завдяки своїй гнучкості та доступності відіграє важливу роль, адже дозоляє користувачам отримувати доступ до даних системи з будь-якої точки світу та за допомогою будь-якого обладнання, що має доступ до мережі Інтернет. Такі системи дозволяють швидко реагувати на зміни показників мережі, проводити віддалену діагностику обладнання та керувати ним. Вебтехнології спрощують доступ та обмін даними між різними частинами системи та дозволяють отримати дані в реальному часі через інтерфейс браузера.

Для інформаційних систем підтримки та управління електромережами, крім вебзастосунків, активно використовуються мобільні застосунки, які забезпечують віддалений доступ і моніторинг у реальному часі. Також SCADA-системи дозволяють контролювати й управляти складними промисловими процесами, включно з електромережами, надаючи детальну візуалізацію та швидку реакцію на події в мережі. Варто відмітити, що все частіше застосовується гібридна архітектура вебзастосунку в поєднанні з API для мобільних пристроїв, яка дозволяє інтегрувати дані з різних джерел і забезпечувати безперебійну роботу між вебінтерфейсом і мобільними пристроями для більшої зручності користувачів.

Для того щоб визначити поточний рівень розвитку інформаційних систем управління енергетичними мікромережами з використанням ВДЕ, проведено огляд аналогічних систем, представлених у науковій літературі, які висвітлюють різні підходи до проєктування, функціональності та використання інформаційних систем для управління мікромережами.

У статті [3] описано розробку та результати випробування вебзастосунка в місті Альгінет Іспанія для моніторингу і менеджменту розумними енергомережами районів міста. У вебзастосунку було реалізовано сервіси:

- енергетичний моніторинг;
- енергетичне прогнозування;

- управління;
- енергетична оптимізація;
- виставлення рахунків;
- торгівля електроенергією.

Загальну схему вебзастосунку зображено на рисунку 1.1.

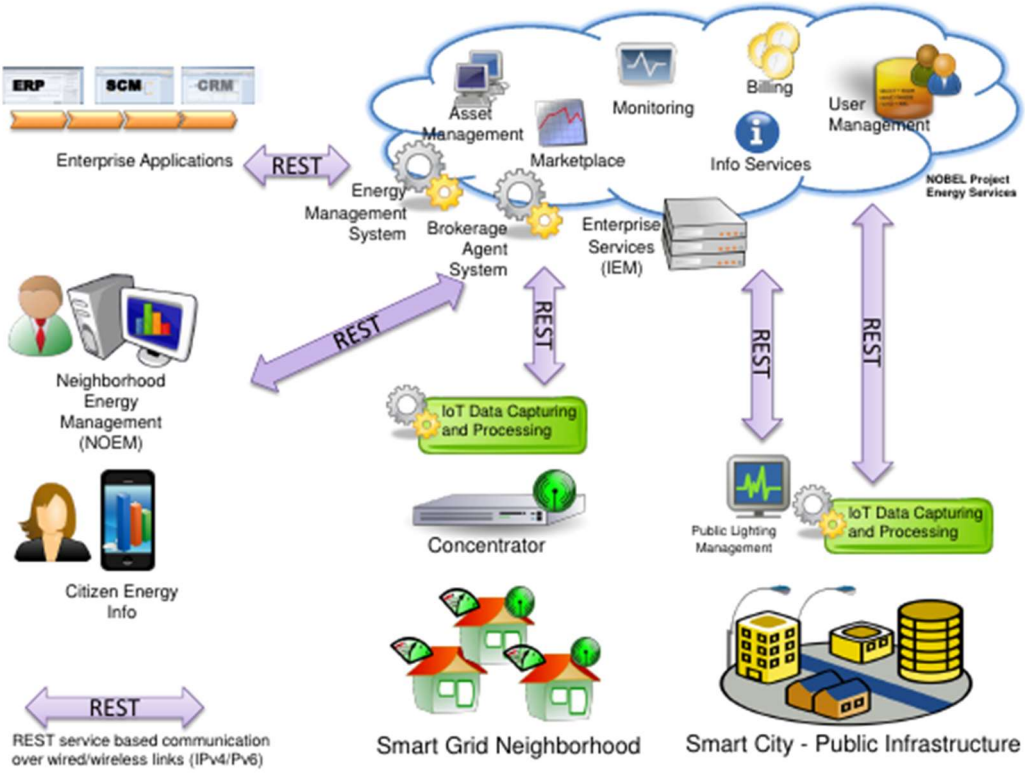


Рисунок 1.1 – Загальна схема вебзастосунку
*Джерело*: [3]

Інтерфейс вебзастосунку розділений на вісім функціональних областей: огляд, моніторинг, управління, прогнозування, брокеринг, оптимізація, виставлення рахунків та спілкування з клієнтами. Більшість вкладок мають бічну панель робочої області. Наприклад, вкладка огляд надає графіки виробництва та споживання енергії. Огляд також може надавати історичні дані використовуючи пошук за початковою та кінцевою датою.

Результатом роботи стало проведення тестування на робочій мережі та збір даних про популярність послух та швидкодію системи загалом. Вебзастосунок NOEM (від «Neighborhood Energy Management») був протестованим локальним

постачальником послуг на реальній мережі у 2013 р. який обслуговував до 5000 абонентів.

Встановлення системи було паралельне до наявної інфраструктури для дослідницьких цілей. Найбільш популярними сервісами виявились купівля та моніторинг електроенергії [3].

Розглянемо ще одне дослідження, описане в статті [4] впроваджено систему управління освітленням в університетському кампусі національному університеті освіти Чанхуа, Тайвань на основі інтернет речей (IOT). Дослідження та впровадження спрямоване на зменшення споживання електроенергії, що в підсумку призведе до економії грошей. Автори[4] не заперечують того, що впровадженням такої ІС поки що є вартісним, але вони стверджують, що з більшою кількістю пристроїв на ринку ситуація зміниться. ІС складається з п’яти рівнів:

- Мережевий рівень з використанням WebAccess (Програмне забезпечення, яке забезпечує вебінтерфейс для взаємодії з пристроями IOT).
- Прикладний рівень з використанням програмного забезпечення та контролерів.
- Управління з використанням віддаленого вводу/виводу.
- Освітлювальне обладнання.
- Рівень сприйняття з використанням датчиків світла.

На рисунку 1.3 наведено знімок інтерфейсу ІС енергоменеджменту, панелі керування освітленням, яка залежить від розкладу на поверсі та рівнем заселеності.

Результати експериментів показали значну економію енергії за рахунок усунення споживання в режимі очікування та/або адаптації поведінки приладів до реальних умов навколишнього середовища. Запропонована ІС управління освітленням в кампусі була успішно завершена і безперебійно працює в режимі онлайн [4].

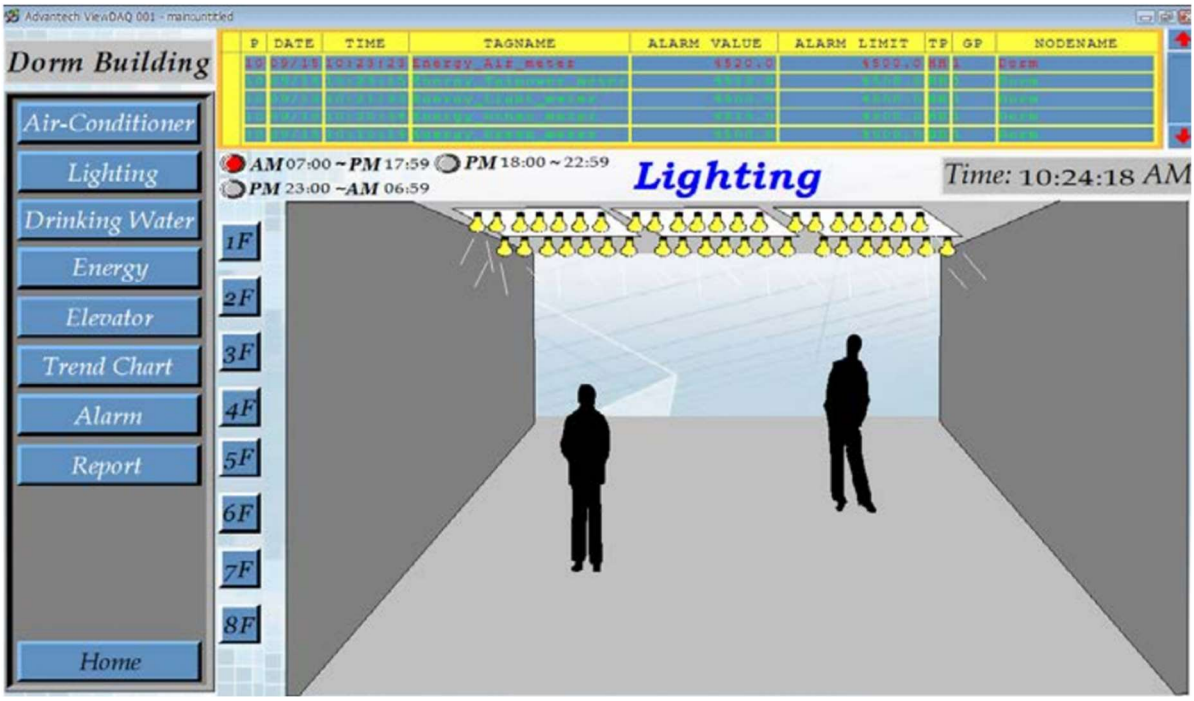


Рисунок 1.2 – Панель інструментів
*Джерело*: [4]

У статті [5] описується використання платформи географічної інформаційної системи (ГІС) в якості тестового проєкту з використанням низьковольтної мережі на базі університету Султна Кабуса в Омані. ГІС – це добре розвинена система онлайн-моніторингу оперативних даних, які вимірюються в різних точках електромережі в режимі реального часу, та відображення всієї мережі на географічній карті. Така система є гарним інструментом для моніторингу, контролю та управління «розумної» мережі, що надає комунальним підприємствам можливість моніторингу та керування мережею з відображенням її розташування на карті. Система візуалізації даних та керування мережею відбувається завдяки використанню вебзастосунку ГІС. Використання вебзастосунку має ряд переваг перед традиційним настільними ГІС завдяки кращій кросплатформеності, можливості доступу з будь-якого браузера великої кількості користувачів та простоті використання, оновлення та масштабування.

Вебзастосунок був розроблений таким чином, щоб бути в першу чергу зручним для користувачів, дозволяє змінювати діапазон видимості, панорамування, можливість вмикати/вимикати різні шари на карті. Користувачі можуть візуалізувати

існуючі розподільні електричні мережі в їх фактичному географічному розташуванні на супутниковій карті. Приклад інтерфейсу вебзастосунку ГІС зображено на рисунку 1.4.

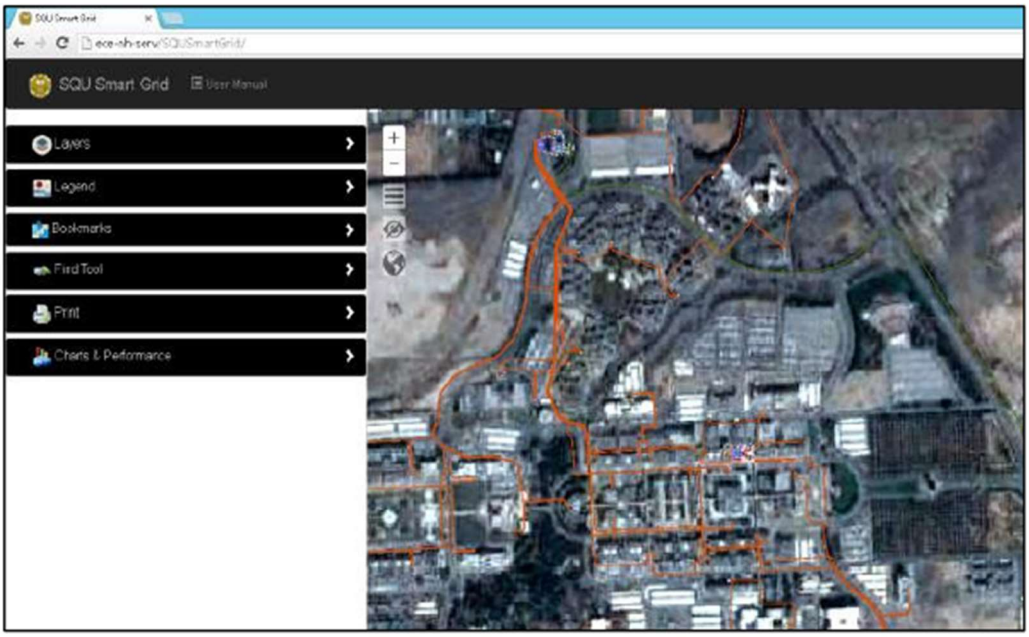


Рисунок 1.3 – Інтерфейс застосунку
*Джерело*: [5]

Однією з ключових функцій вебзастосунку є інструменти відображення графіків та статистики, за допомогою яких відображається потужність фотоелектричних модулів, вітрової турбіни, навантаження та потужність мережі та напруга акумулятора. Всі ці дані зберігаються в базі даних та допомагають проаналізувати, коли і за яких погодних умов краще використовувати потужності певного виду ВДЕ.

Автори статті підкреслюють, що запропонована ГІС може бути легко розширена, а доступність інформації про стан електромережі забезпечить контроль на кожному рівні системи [5].

У наступній розглянутій статті описано процес розробки та розгортання мікромережі на базі лабораторій Каліфорнійського університету в Лос-Анджелесі та Корейського інституту енергетичних досліджень в Теджоні, Південна Корея [6]. Розроблена мікромережа є гнучкою та швидко масштабованою, бо вона підтримує

технологію plug-and-play, призначену для швидкого визначення та конфігурування нових пристроїв у мережі, та використовує ресурсно-орієнтовану архітектуру.

Автори зазначають, що одним з важливих критеріїв для своєї мікромережі вони відмічають її можливість бути гнучкою з точки зору програмного забезпечення та легко адаптуватись. Тому їх програмне забезпечення розроблене за принципом модульної системи. Було створено фреймворк, в якому різноманітні модулі (наприклад, модуль алгоритму планування та модуль зв'язку) можна легко додавати та/або видаляти. Це дозволяє швидко розширяти та вдосконалювати мікромережу. Для моніторингу та керування мікромережею в режимі реального часу використовується вебінтерфейс користувача, як показано на рисунку 1.5.


Рисунок 1.4 – Вебінтерфейс, що показує огляд мікромережі
*Джерело*: [6]

Через вебінтерфейс користувачі можуть отримувати дані про стан мережі, керувати навантаженням, отримувати інтерактивні графіки та таблиці (наприклад, архівні чи прогнозовані дані).

Автори відзначають зручність у користуванні системою та максимальну ефективність завдяки аналізу великої кількості отриманої інформації [6].

У статті [7] автори представляють ефективність своєї стратегії енергоменеджменту для управління мікромережею, яка включає сонячні та вітрові

електростанції, систему зберігання енергії та резервні електричні мережі. За допомогою контроля продуктивності заряджання та розряджання акумуляторів системи зберігання енергії здійснюється ефективне керування системою, що базується на визначенні оптимального навантаження за рахунок доступних відновлювальних джерел енергії та стану заряду акумуляторів. Всі дані вимірювань мікромережі передаються через програмне забезпечення з графічним інтерфейсом користувача для аналізу цих даних. Запропонована стратегія енергоменеджменту використовує передові технології, такі як штучний інтелект, для аналізу даних про споживання та виробництво енергії, стан акумуляторів та інше. Розроблений інтерфейс для моніторингу та керування системою дозволяє користувачам отримати доступ до даних мікромережі з будь-якого місця та в будь-який час, що значно полегшує процес керування та запобігає виникненню критичних помилок системи. Запропонований інтерфейс був розроблений на основі Python, вимірювання показників системи здійснюється за допомогою Matlab/Simulink, а дані зберігаються в MS Excel таблиці. На рисунку 1.6 показано візуалізацію різних показників мікромережі та систему управління енергією (потужність шин змінного та постійного струму, частота рівнів, стан батареї тощо).

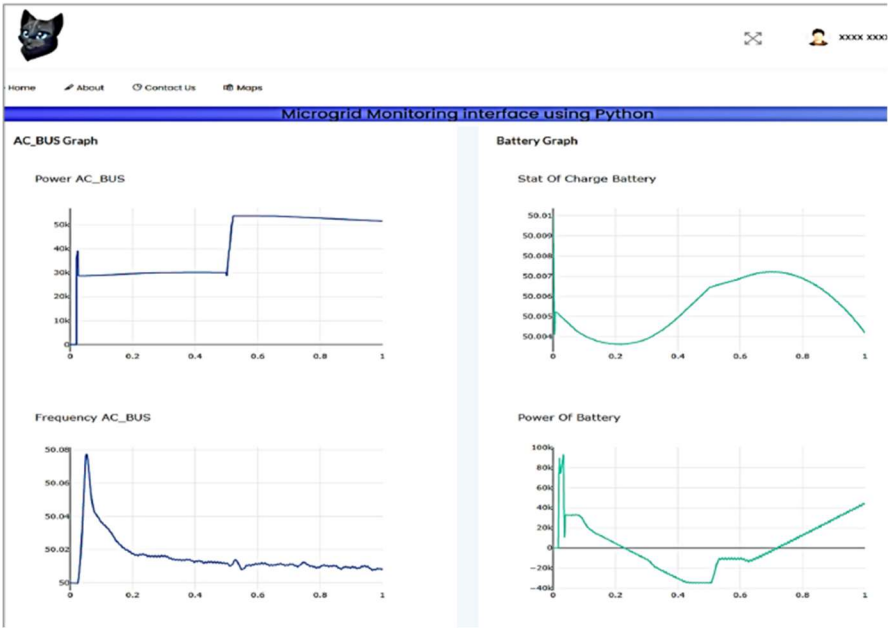


Рисунок 1.5 – Статус мережі
*Джерело*: *[4]*

У запропонованому інтерфейсі можна відслідковувати графічне представлення значень кожного окремого параметра, пов’язаного з роботою мікромережі. Крім того, для оптимізації роботи мережі використовуються погодні дані в реальному часі, такі як швидкість вітру та потужність сонячного випромінювання, що дозволяє підтримувати стабільне електропостачання за будь-яких метеорологічних умов. Загалом автори підкреслюють важливість реалізації зручного інтерфейсу користувачів і важливості онлайн доступу до нього в будь-який момент, що вкотре акцентує увагу на актуальності розробки вебінтерфейсів для моніторингу та керування мікромереж [7].

Автори статті [8] описують свій досвід розробки системи управління енергоспоживанням мікромережі на основі Інтернету речей. Система дозволяє керувати ресурсами енергії з врахуванням змін у навантаженні та генерації енергії. Сама система базується на мікросервісах - тобто системі, розбитій на маленькі незалежні частини. Це включає в себе оптимізатор для планування ресурсів, базу даних, інтерфейс користувачів та програмний інтерфейс для взаємодії з іншими системами. Як зазначають автори, система тестується за допомогою реального симулятора, і використовуються дані мікромережі з університету UNICAMP у Бразилії. До складу мікромережі входять сонячна установка, система зберігання енергії, тепловий генератор та змінні навантаження. Графічний інтерфейс системи надає користувачам можливість зручно використовувати вебдодаток для перегляду та аналізу даних у виді графіків, а також налаштовувати параметри системи та керувати нею. В розробці інтерфейсу використовували фреймворк Angular, що використовує TypeScript як основну мову програмування. На рисунку 1.7 показано інтерфейс користувача, де візуалізовано графіки основних параметрів мікромережі в режимі реального часу, поточний стан акумуляторів системи збереження енергії, активну потужність системи фотоелектричної генерації та діаграму використання різних джерел енергій на поточний момент. Також для користувачів відображається інформація стосовно добового споживання енергії, вартість експлуатації та генерації.


Рисунок 1.6 – Інтерфейс користувача
*Джерело:* [8]

Крім моніторингу система передбачає можливість керування навантаженням та фотоелектричними установками. Також система забезпечує оптимальне планування розподілення енергії на наступну добу. Результати тестування системи оказали її ефективність та здатність масштабувати у подальшому [8].

У статті [9] автори відзначають, що мікромережі є ключовими компонентами для розробки розумних електромереж у майбутньому, проте їхня непостійність створює виклики для керування ними. Тому дослідження та розробка нових методів та підходів керування є дуже важливим кроком для побудови продуктивних систем. З цією метою освітні заклади створюють свої мікромережі, проте часто заклади стикаються з рядом перепон, таких як висока вартість обладнання та технологій, обмеження простору. Через це віртуальні лабораторії стали досить популярними, особливо для роботи з відновлюваною енергією, оскільки дозволяють обійти ці обмеження. Автори у своєму дослідженні описують процес створення веборієнтованої віртуальної лабораторії для мікромереж, яка використовує такі інструменти, як NI LabVIEW, Microsoft .Net Core і MATLAB/Simulink. Розроблені матеріали охоплюють теми сонячної та вітрової енергії, управління акумуляторами та роботу мікромереж, що суттєво підвищило інтерес студентів до теми

відновлюваних джерел енергії та розробки мікромереж з їх використанням. Віртуальна лабораторія стала ефективним інструментом, що дозволив студентам проводити експерименти з мікромережами та відновлювальною енергією. Одним з прикладів роботи віртуальної лабораторії автори зазначають про можливість моніторингу та аналізу виробництва енергії сонячної електростанції потужністю 1кВт в університеті Північного Іллінойсу (США). Дані про напругу, струм та потужність зі станції передаються кожні 5 хвилин до лабораторії, де студенти можуть моніторити та аналізувати вплив сонячної енергії продуктивність мікромережі. Всі дані відображаються у вебінтерфейсі, що є простим та зрозумілим для користувачів, також доступний з будь-якого браузера. Приклад відображення інтерфейсу віртуальної лабораторії показано на рисунку 1.8.

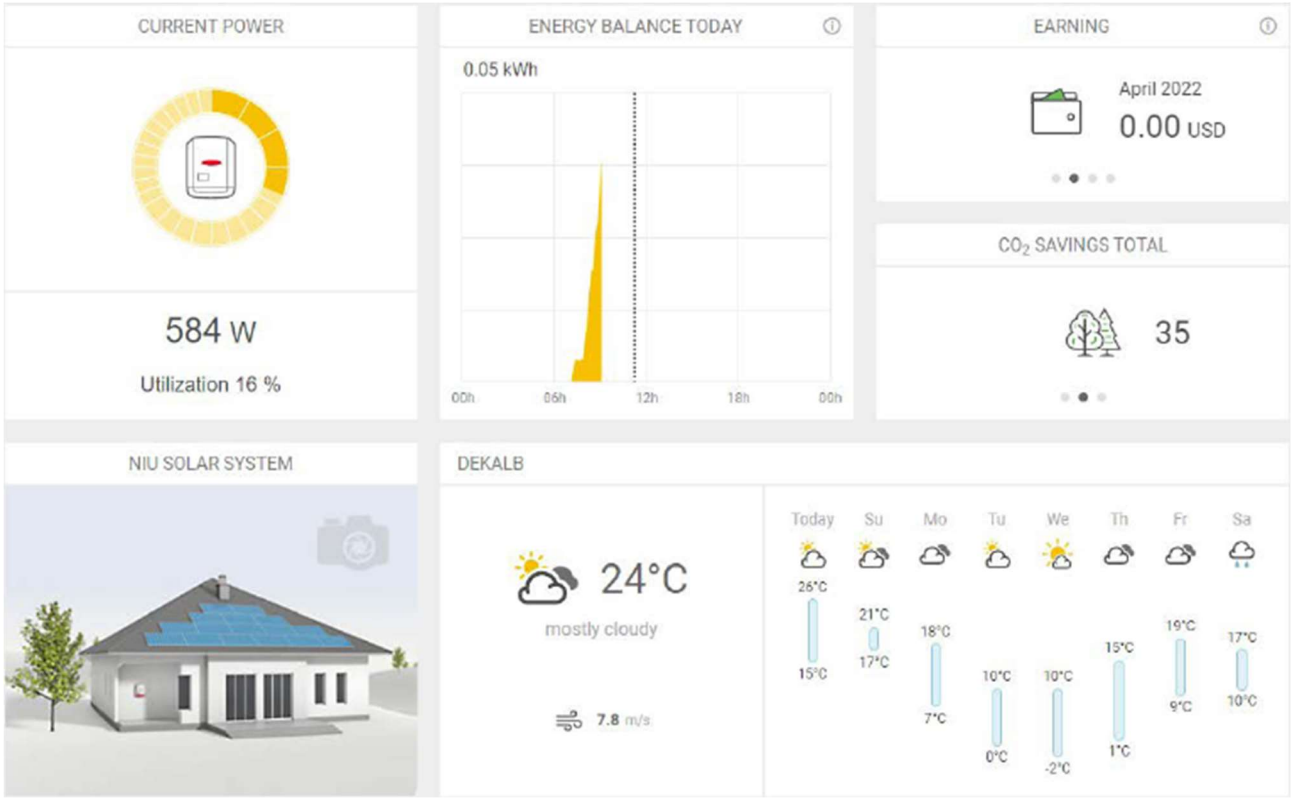


Рисунок 1.7 – Інтерфейс віртуальної лабораторії
*Джерело*: [9]

Результати опитувань показали, що віртуальна лабораторія сприяла кращому засвоєнню інформації про мікромережі та збільшила обізнаність та зацікавленість студентів у відновлювальній енергії. Цьому результату посприяв також і зручний

вебінтерфейс, що вкотре підкреслює важливість розробки інтуїтивно зрозумілого інтерфейсу користувачів [9].

У статті [10] описується, як системи SCADA (англ. Supervisory Control and Data Acquisition) покращую надійність, безпеку та економічну ефективність роботи мікромережі. SCADA виступає зв'язуючим елементом між центральним контролером мікромережі та вебсистемою моніторингу. Програмована на Java, вона забезпечує комунікацію та керування в режимі реального часу, включаючи збір і зберігання даних, збалансування навантаження та обробку команд. Для підвищення стабільності та безпеки мікромережі SCADA виконує обробку даних, захист від збоїв та передачу інструкцій. Усі ці функції були перевірені в реальних умовах експлуатації. Завдяки розробленій вебсистемі моніторингу, користувач може керувати шістьма компонентами мікромережі: трьома показниками навантаження системи (load 1, load 2, load 3), сонячною панеллю, вітрогенератором та акумулятором. Якщо потужності генерації недостатньо, адміністратор може відключити одне з навантажень для збереження енергії. Наприклад, вибравши Load 1 для відключення і натиснувши кнопку "Підтвердити", система передає команду до SCADA через програмний інтерфейс. SCADA передає інструкцію центральному контролеру через модуль UART Ethernet і записує подію у базу даних MySQL. Це забезпечує гнучке управління навантаженнями та зберігання даних про всі зміни в системі. Керування системи здійснюється в режимі реального часу.

Як результат автори відзначають не тільки високу стабільність та легку маштабованість самої SCADA-системи, а також покращення продуктивності завдяки доступності вебінтерфейсу для моніторигну та керування мікромережою [10].

У тексті статті [11] описується розробка недорогої SCADA-системи для гібридної мікромережі на основі відновлюваних джерел енергії (сонячної панелі, вітрогенератора, біогазової установки та акумуляторної батареї). Ця система призначена для лабораторії відновлюваної енергії політехнічного університету Валенсії (Іспанія) і допомагає досліджувати стабільність мікромереж та виробництво енергії. SCADA-система використовує недорогі компоненти, такі як пристрої на

основі Arduino та центральний модуль на Raspberry Pi, які передають дані до локальної бази даних і вебінтерфейсу SCADA. Для комунікації між компонентами застосовуються протоколи TCP/IP, I2C, SPI та бездротові передавачі NRF24L01. Порівняння показало, що ця система на 86% дешевша за стандартні рішення, але забезпечує таку ж надійність для локального та віддаленого управління і збору даних. У тексті статті описується взаємодія користувачів із вебінтерфейсом SCADA для мікромережі, реалізованим на HTML5, JavaScript та PHP і розміщеним на сервері PLESK. Система зберігає дані у хмарній базі даних, яка постійно оновлюється для контролю параметрів мікромережі, а також використовує локальну базу даних для резервного копіювання у разі проблем із мережею. Комунікація з пристроями мікромережі здійснюється через модем 3G або Wi-Fi, а дані записуються кожну секунду та зберігаються у базі MySQL. Проведені експерименти підтвердили надійність системи, ефективність збору даних та комунікації з віддаленою базою даних. Розроблена система має високу універсальність, що дозволяє додавати нові функції та пристрої. Можливість масштабування проєкту відзначають в тому числі й завдяки реалізації системи моніторингу та контролю у вигляді вебінтерфейсу, що суттєво пришвидшує процес інтеграції нових модулів та підсистем до проєкту. У майбутніх проєктах автори статті планують інтегрувати нові технології, такі як IoT, блокчейн та великі дані, для розширення можливостей проєкту [11].

У статті [12] розповідається про проблему енергетичного забезпечення в сільських районах, які стикаються з нестачею електроенергії та отримують основне електрозабезпечення, що використовує в основному екологічно шкідливі невідновлювані джерела енергії (викопне паливо). Щоб вирішити цю проблему, були запропоновано використання мікромереж, які можуть забезпечити енергією такі регіони. З цією метою робоча група досліджувала можливість впровадження ізольованих мікромереж у цих громадах. Однак було виявлено, що для точного аналізу оптимальної конфігурації мікромереж потрібен відповідний програмний інструмент для планування. Тому запропонували розробити зручну та безпечну вебсистему для планування мікромереж. Цей інструмент створено за допомогою

Django та Python, а його функціональність була перевірена користувачами для забезпечення правильності розрахунків і отримання очікуваних результатів. Вебзастосунок розроблено із використанням шаблону проєктування програмного забезпечення Model-View-Controller (MVC). Цей шаблон ділить систему на окремі частини, кожна з яких має конкретну функцію. Система складається з трьох частин: інтерфейс користувача для взаємодії (view), контролер для зв'язку між інтерфейсом і моделлю (controller) та модель для збереження й обробки даних у базі (model). Робочий процес починається з взаємодії користувача, обробки даних у базі та виведення результатів через інтерфейс. Користувачі можуть вказати інформацію про місце розташування, демографію, середній дохід, типи споживачів, поточне паливо, ставки дисконтування та тривалість проєкту. Також користувач може отримати інформацію стосовно вартості установки мікромережі в своєму регіоні, деталі стосовно тарифів та можливостей доходу від продажу електроенергії в мережу, а на вкладці «$CO_2$» вказуються дані про викиди вуглецю відповідно до типу палива, що дозволяє оцінити потенційні зниження викидів (показано на рисунку 1.9). Розрахунки, отримані на основі введених даних, показуються на сторінці «Огляд проєкту», яка включає підсумкові результати фінансового аналізу та прогнозів для мікромережі.

Автори відзначають, що цей інструмент корисний для проєктування й оптимізації мікромереж, особливо для інвесторів із низькими доходами та користувачів без технічних знань. Він має зручний інтерфейс, є доступним як відкрите ПЗ та виступає альтернативою комерційним продуктам (як HOMER або RETScreen). Хоча інструмент не має деяких розширених функцій, наприклад, алгоритмів оптимізації та аналізу чутливості, у майбутньому планується їх додати. Загалом, він є доступним рішенням для розвитку відновлюваної енергії та подолання енергетичних проблем у сільських районах [12].

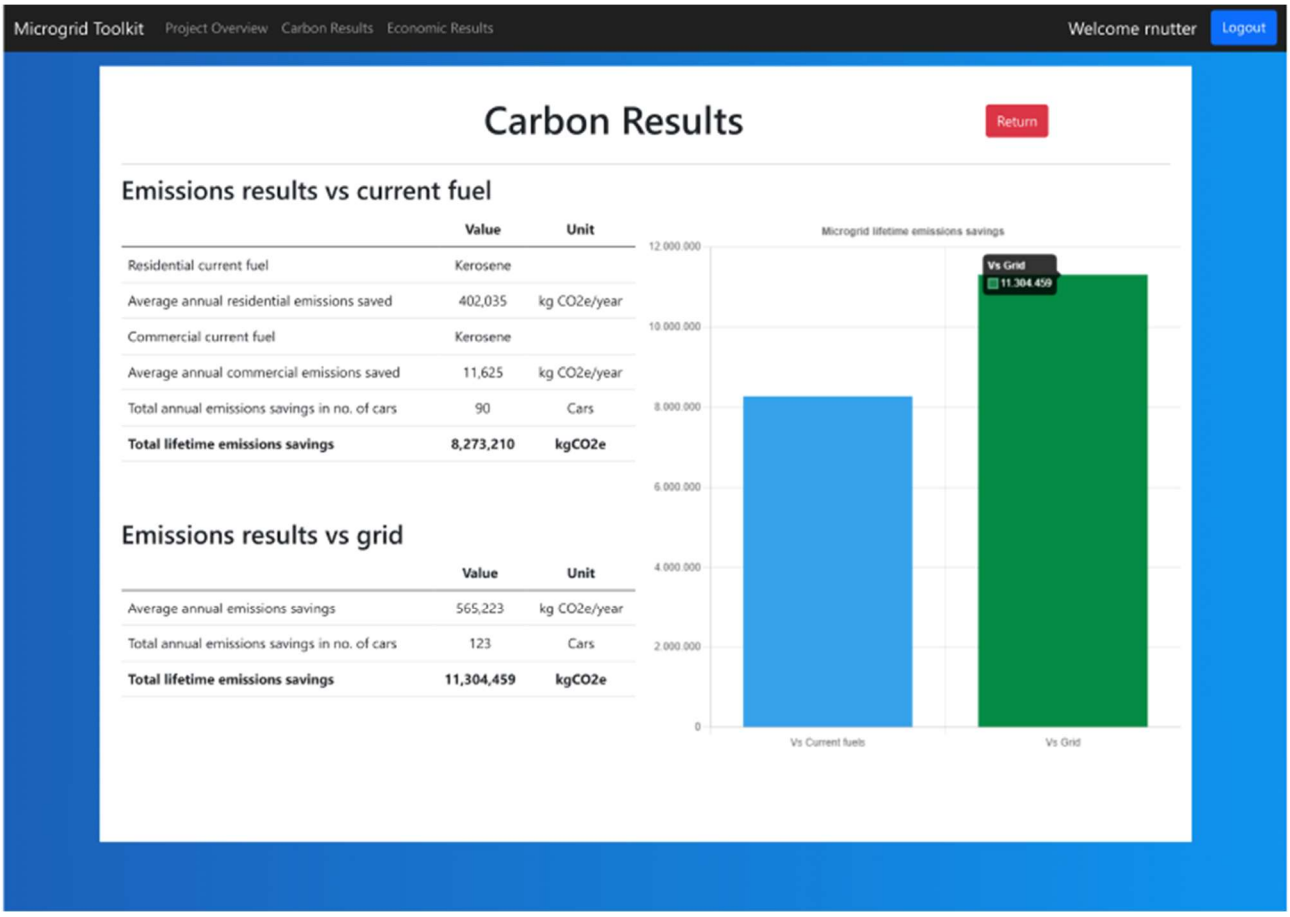


Рисунок *1.8* – Дані про викиди
*Джерело*: [12]

У результаті огляду ряду статей на тему мікромереж та варіантів їх реалізацій встановлено, що досліджувана тематика є актуальною та важливою, існує великий попит на зазначені технології та рішення. Також варто зазначити, що більшість варіантів реалізації проєктів мікромереж з використанням альтернативних джерел енергії використовують вебінтерфейс для доступу до даних та керування системою, бо це значно пришвидшує та спрощує процес масштабування проєкту та можливості додавання нових функцій, є зручним як для користувачів, так і для адміністраторів, також надає можливість доступу до системи з будь-яких пристроїв з доступом до мережі Інтернет.

## 1.2 Аналіз поточної інформаційної системи підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії

Використання систем моніторингу та управління для енергетичних мікромереж створює високий попит на такі системи. У Сумському державному університеті (СумДУ) було реалізовано інформаційну систему управління енергетичними мікромережами з ВДЕ. Основними задачами якої є :

- візуалізація структури мікромережі;
- прогноз енергогенерації та енергоспоживання;
- підтримка прийняття рішень щодо управління та планування гібридних енергетичних систем з використанням відновлюваних джерел енергії (ГЕСВДЕ);
- використання панелі керування експерта;
- моніторинг компонентів мікромережі;
- візуалізація даних щодо згенерованої енергії;
- застосування модуля зворотного зв’язку;
- ведення журналу подій [1,13-15].

Створена ІС підтримки управління енергетичними мікромережами з ВДЕ має ряд позитивних аспектів, які роблять її ефективною та надійною. Дана система дозволяє отримати цілодобовий доступ до даних про стан мікромережі, та реалізована як вебзастосунок, що надає доступ до даних моніторингу та панелі керування віддалено з будь-якого браузера. Інформація про погодні умови отримується з вебсайту за допомогою API інтерфейсу, а використовуючи дані з датчиків, якими обладнана мікромережа, система отримує дані про енергоспоживання. Відображення цих даних в інтерфейсі вебзастосунка показано на рисунку 1.10. Прогнозування погодинного споживання електроенергії здійснюється за допомогою реалізації методу нейромережевого прогнозування з використанням мережі довготривалої пам’яті LSTM, що має перевагу над іншими методами завдяки нечутливості до

тривалих часових перерв [14]. Для побудови короткострокового прогнозу генерації електроенергії від сонячної панелі використовується також штучна нейронна мережа з використанням мережі довготривалої пам'яті LSTM[16].

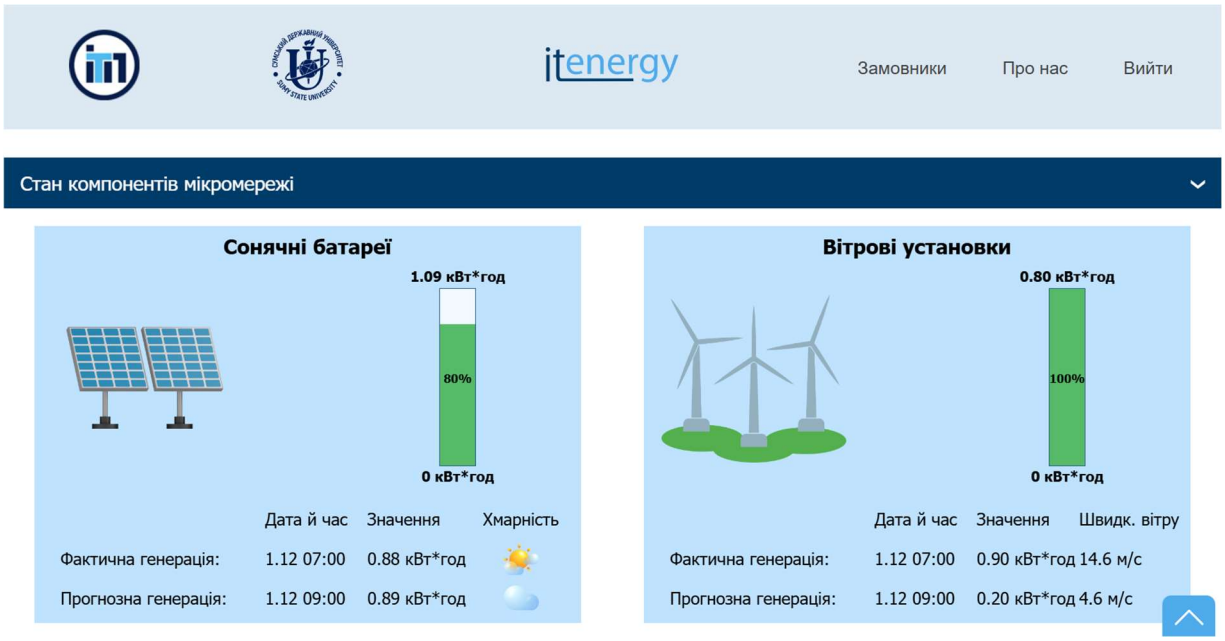


Рисунок 1.9 – Інтерфейс наявної інформаційної системи
*Джерело:* створено автором

Для оперативної обробки досить великих масивів даних в режимі реального часу, забезпечення багатовимірності та багатопоточності обробки цих даних використовується оперативне сховище даних. Такий підхід дозволяє забезпечити інформаційній системі швидку роботу з даними та дозволяє здійснити розмежування використання сховищ даних для аналітики та операції з маніпулювання даними [13].

У даній ІС підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії реалізована функціональна модель збору та зберігання даних, а також математична модель моніторингу даних, які дозволяють забезпечити ефективний моніторинг великої кількості різноманітних даних на різних часових інтервалах. Також інформація про поточний стан енергосистеми та короткострокові прогнози даних зберігаються в оперативній базі даних, що значно пришвидшує доступ до цієї інформації [1].

Проте ІС потребує доопрацювань для підвищення безпеки та ефективності. А також одним з недоліків є архітектура яка побудована без врахування сучасних вимог до проєктування ІС, наприклад, об'єктно-орієнтоване програмування. Використання ООП дозволяє створювати модульний та більш гнучкий код, що значно покращило б структуру коду. А відсутність ООП може призводити до появи заплутаного коду інформаційної системи, коли внесення змін стає все складнішим та підтримка ІС займає більше часу та зусиль.

Ще одним важливим недоліком в інформаційної системи є наявність вразливостей у безпеці, які можуть призвести до викрадення або пошкодження даних. Необхідно провести перевірку ІС для виявлення та усунення потенційних загроз, таких як XSS (міжсайтовий скриптинг), CSRF (підробка міжсайтових запитів), SQL-ін'єкції (виконання шкідливого SQL коду на сервері бази даних). Усунення таких вразливостей дозволить захистити дані користувачів та забезпечити надійну роботу інформаційної системи.

Незначним недоліком, про який варто згадати, є відсутність системи контролю версій. Проте така система відіграє значну роль у процесі командної роботи над проєктом. Без системи контролю версій складно відстежувати зміни, за потреби повертатись до попередніх станів коду та співпрацювати в команді, що призводить до сповільнення роботи над проєктом, ускладнює його розширення та підтримку.

Отже, ІС підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії, яка була розроблена в СумДУ, має ряд своїх переваг та певні недоліки. Проведення реінжинірингу системи дозволить усунути ці недоліки, що зробити систему більш ефективною, безпечною та пришвидшить процес її розвитку.

## 1.3 Огляд аналогічних інформаційних систем підтримки управління енергетичними мікромережами

Проведено аналіз комерційних програмних продуктів, аналогічних до поточної інформаційної системи, для того, щоб визначити їхні можливості та недоліки у сфері підтримки управління енергетичними мікромережами з ВДЕ. Зараз існує багато різноманітних інформаційних систем для моніторингу та підтримки енергетичних мікромереж з ВДЕ. Наприклад, EcoStruxure Power Monitoring Expert від компанії Schneider Electric (Франція). Ця ІС допомагає здійснювати моніторинг електричної мережі, відстежувати споживання енергії та в реальному часі оптимізувати енерговитрати. Вона ідеально підходить для великих систем та підтримує аналіз енергоспоживання з різних джерел енергії. Проте EcoStruxure Power Monitoring Expert є досить складною системою, яка вимагає від користувача експертних знань, та частіше застосовується саме для промислових підприємств [17]. На рисунку 1.10 зображено інтерфейс ІС EcoStruxure Power Monitoring Expert.

Рисунок 1.10 – Інтерфейс ІС EcoStruxure Power Monitoring Expert
*Джерело:* [17]

Також прикладом є ІС Open Energy Monitor, розроблена британською компанією OpenEnergyMonitor, яка за допомогою різноманітних датчиків здатна збирати дані про стан енергомережі та аналізувати енергоспоживання. Доступ до цих даних, аналіз та можливість їх візуалізації здійснюється через вебінтерфейс EmonCMS, що дозволяє ефективно контролювати стан мережі віддалено. Платформа має відкритий код і постійно розвивається, щоб надавати більше можливостей для обробки, реєстрації та візуалізації даних про енергію, температуру та інші дані про навколишнє середовище. Одним з вагомих недоліків даної системи є відсутність офіційної підтримки, тому налаштування та користування може бути складним для новачків [18]. На рисунку 1.11 зображено інтерфейс ІС.

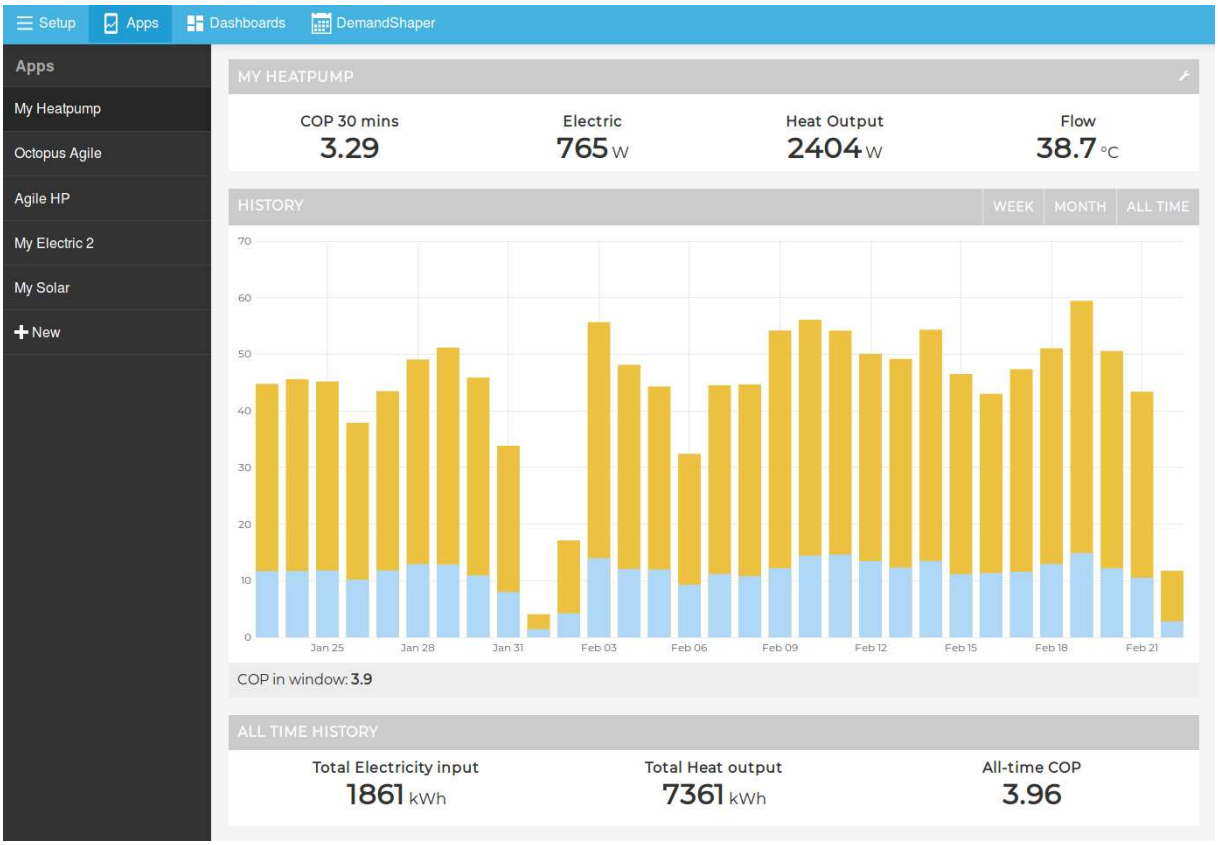


Рисунок 1.11 – Інтерфейс системи
*Джерело:* [18]

Ще одним прикладом вебзастосунка, що надає доступ до даних про споживання енергії в режимі реального часу та за минулі періоди, а також про виробництво сонячної енергії та витрати на електроенергію є Sense Home Energy Monitor. Крім

візуалізації даних про споживання енергії різними приладами в окремі проміжки часу, система надає поради щодо зменшення вуглецевого сліду та можливостей оптимізації споживання. Крім вебзастосунка Sense Home Energy Monitor пропонуються також використання мобільного застосунку (рис. 1.12), який буде оперативно інформувати споживачів про зміни в мережі. До недоліків можна віднести те, що поки інформаційна система працює лише для домашніх мереж та потребує придбання спеціального обладнання [19].

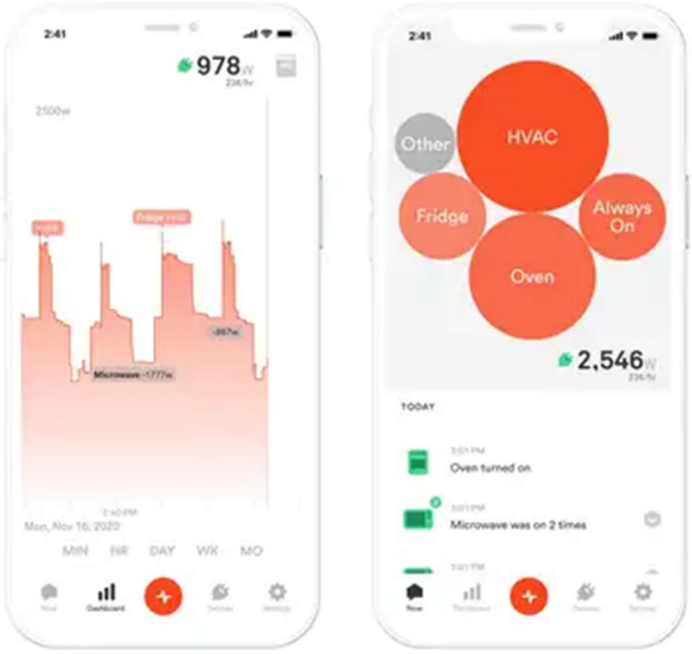


Рисунок 1.12 – Інтерфейс мобільного застосунку
*Джерело:* [19]

До популярних мобільних застосунків відноситься Tesla App, розробник компанія Tesla, Inc. (США). Його перевагами є проста інтеграція з продуктами Tesla, такими як Tesla Solar Roof (сонячні панелі у вигляді покриття даху) та Tesla Powerwall (акумулятор) [20]. Мобільний застосунок має простий та зрозумілий інтерфейс, де можна побачити потужність мікромережі та керувати розподілом енергії. Недоліком є те, що застосунок (рис. 1.13) працює тільки з обладнанням Tesla [20].

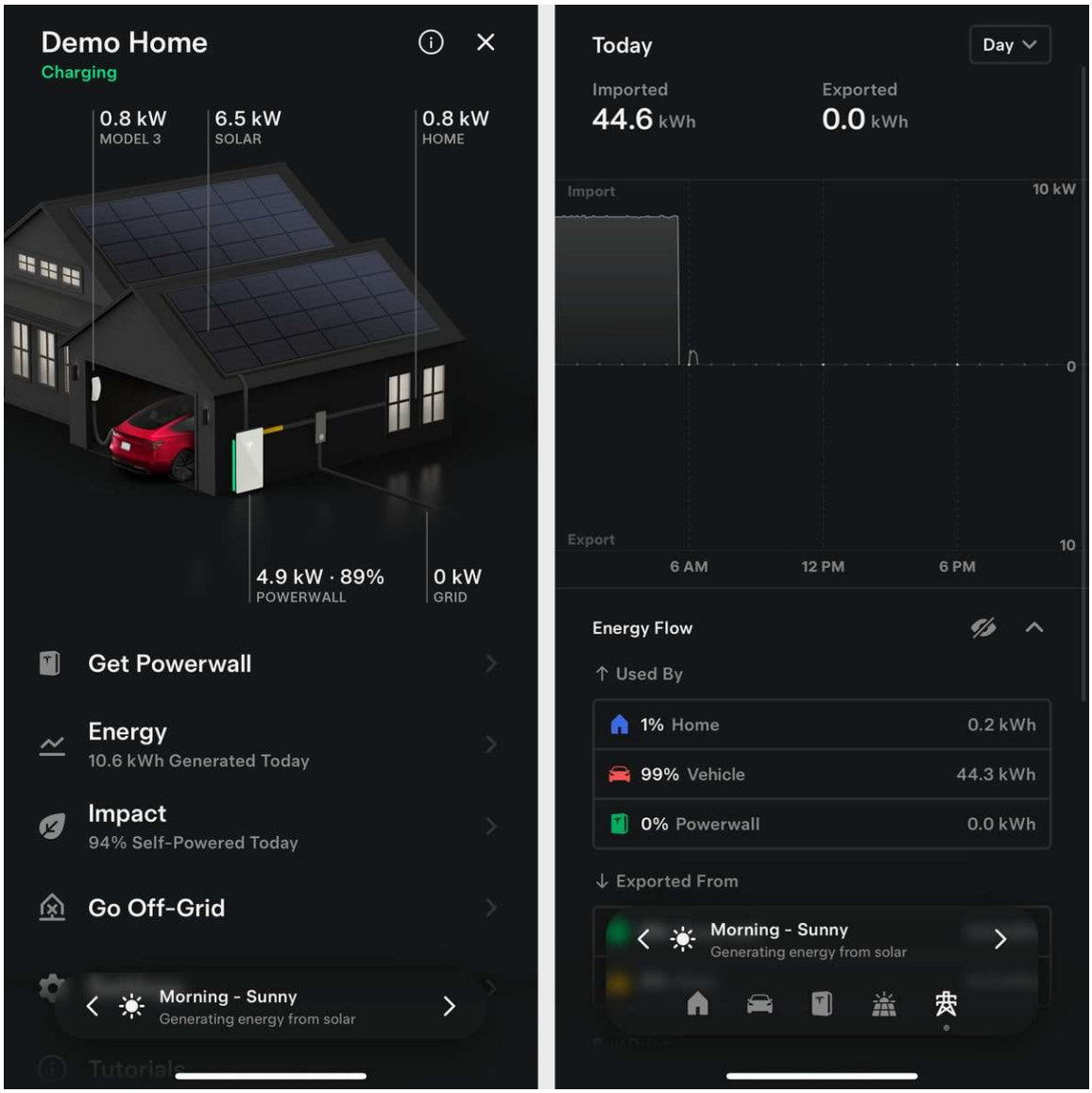


Рисунок 1.13 – Інтерфейс застосунку
*Джерело:* [20]

GridPoint – це система для енергоменеджменту, яка підходить для моніторингу та управління мікромережами з відновлюваними джерелами енергії. Розроблена американською компанією GridPoint, ця платформа надає користувачам доступ до вебінтерфейсу та можливість використовувати мобільний додаток, що дозволяє в реальному часі відстежувати енергоспоживання, керувати обладнанням і оптимізувати використання ВДЕ. Система дозволяє відслідковувати енергоспоживання, управляти піковими навантаженнями та отримувати попередження про критичні ситуації в системах. враховуючи широкий функціонал, GridPoint може бути складною у використанні для нових користувачів і потребує часу

на освоєння, крім того користування за підпискою, що може бути дорогим варіантом для малих підприємств чи домогосподарств, які потребують базових функцій [21]. Інтерфейс застосунку зображено на рисунку 1.14.

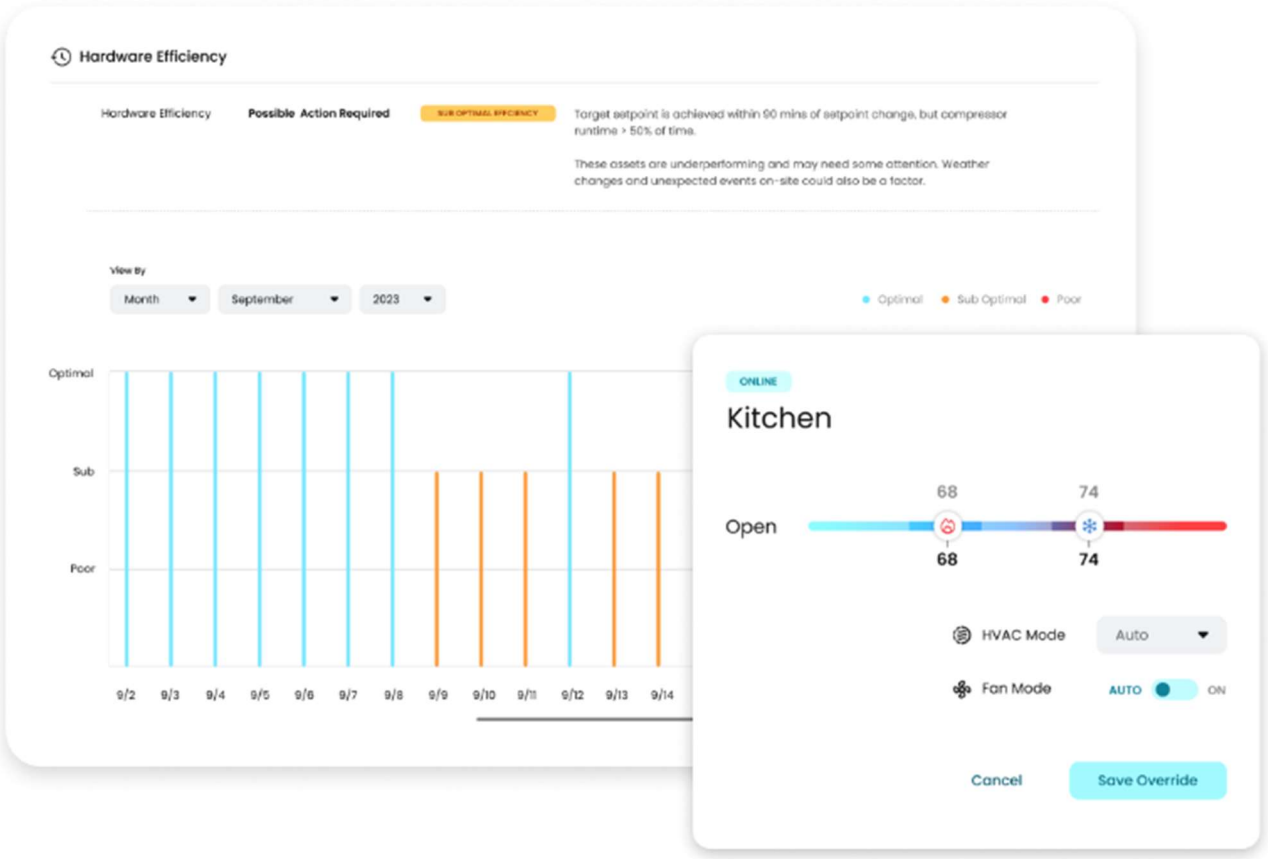


Рисунок 1.14 – Інтерфейс системи
Джерело [21]

Ще одним прикладом мобільного застосунку для аналізу мікромережі з використанням сонячної енергії є Enphase Enlighten, від однойменної компанії з США. У мобільному застосунку користувачі можуть отримати сповіщення про стан системи, провести моніторинг у реальному часі та здійснити відповідні налаштування для ефективної роботи системи. Відмінно підходить для використання в домашніх мікромережах, має простий та зрозумілий інтерфейс (рис. 1.15). Також працює тільки з відповідним обладнанням компанії розробника [22].

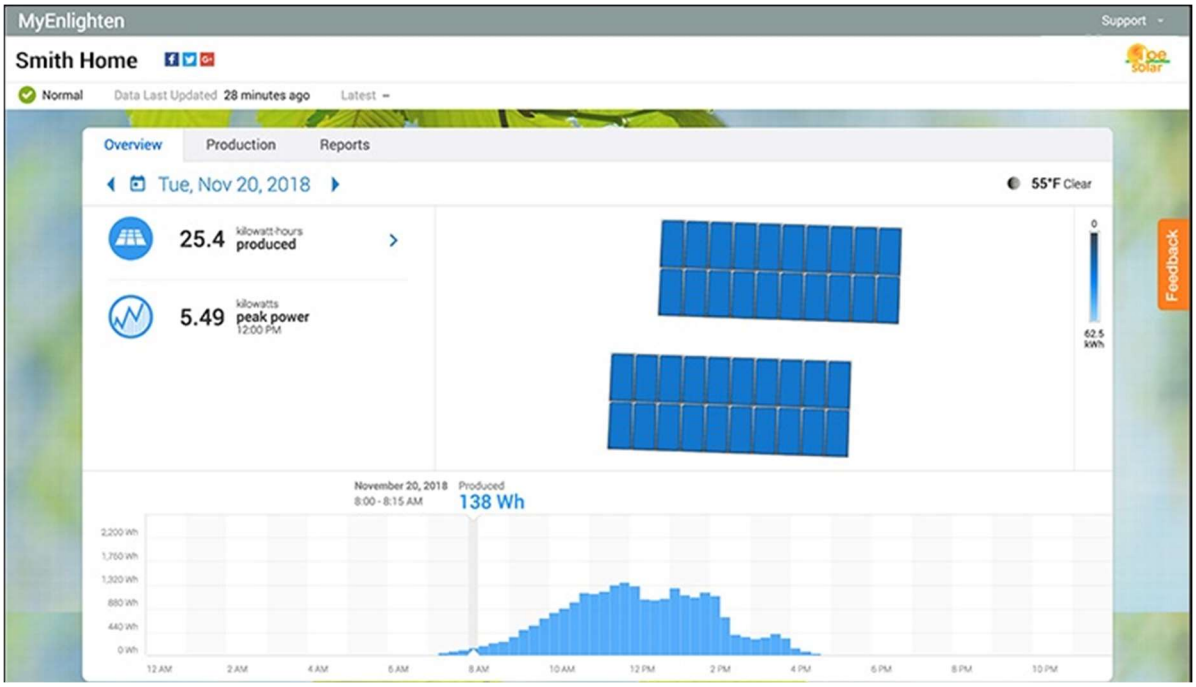


Рисунок 1.15 – Інтерфейс
Джерело:[22]

Прикладом вебзастосунка, що надає багато детальної інформації про стан мікромережі та може підтримувати використання різних видів ВДЕ є ABB Ability™ Energy Manager, від компанії ABB Ltd. (Швейцарія). Ця інформаційна система націлена на використання в промислових установках та потребує професійних навичок налаштування мікромереж. У вебзастосунку (рис. 1.16) крім аналітичної інформації є також і поради стосовно покращення стану мікромережі, попередження про ймовірні помилки та проблеми з обладнанням, що аналізуються на основі великої кількості показників мережі [23].

SolarEdge Go є прикладом реалізації інформаційної системи моніторингу та контролю у вигляді зручного мобільного застосунку. Розроблений компанією SolarEdge Technologies (Ізраїль) він призначений для контролю сонячної енергії та надає детальну інформацію про генерацію електроенергії мікромережею. Як і більшість комерційних розробок, мобільний застосунок SolarEdge Go (рис. 1.17) підтримує взаємодію лише з відповідним обладнанням компанії розробника та має обмежений функціонал для інших видів ВДЕ [24].

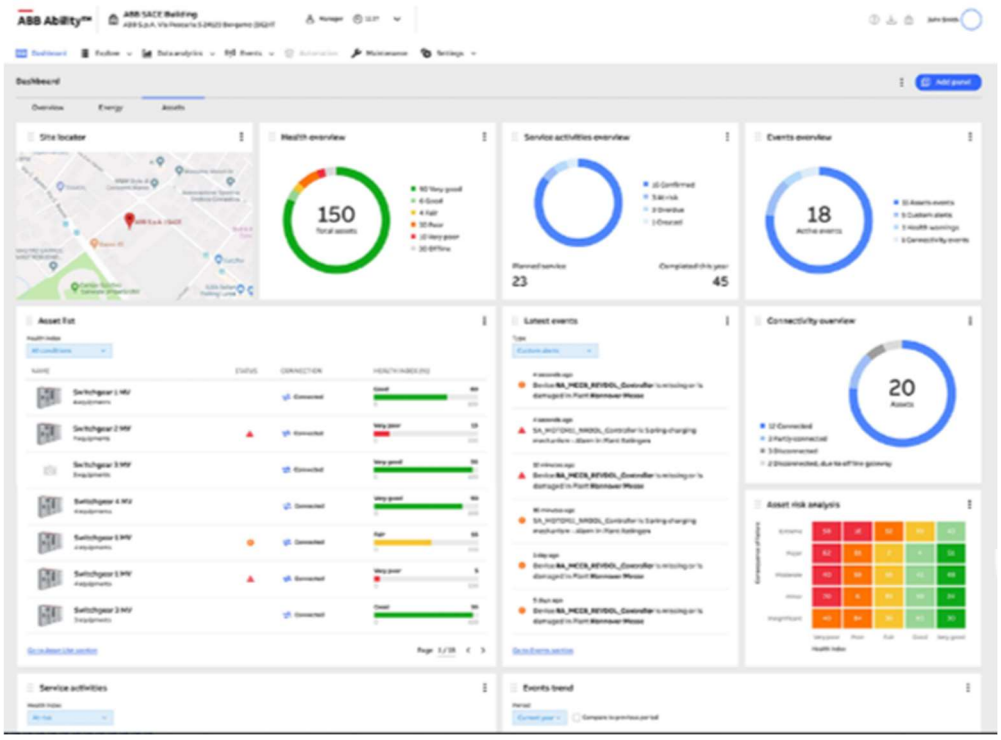


Рисунок 1.16 – Інтерфейс вебзастосунку
*Джерело*: [23]


Рисунок 1.17 – Інтерфейс мобільного застосунку
*Джерело:* [24]

Однією з провідних систем для моніторингу та оптимізації мікромереж, яка допомагає користувачам створювати ефективні системи є HOMER (Hybrid Optimization of Multiple Energy Resources) розроблений компанією HOMER Energy by UL, США. Їх застосунки HOMER Pro і HOMER Grid дозволяють моделювати мікромережі з різними типами відновлюваних джерел енергії, створювати та керувати комбінованими системи і системами зберігання енергії. Крім того, HOMER Grid підтримує тарифні структури та має потужні інструменти для аналізу тарифів. Їх застосунки потрібно встановлювати на комп'ютер (рис. 1.18), також вони досить складні для нових користувачів та вимагають певного рівня підготовки через широкий функціонал [25].

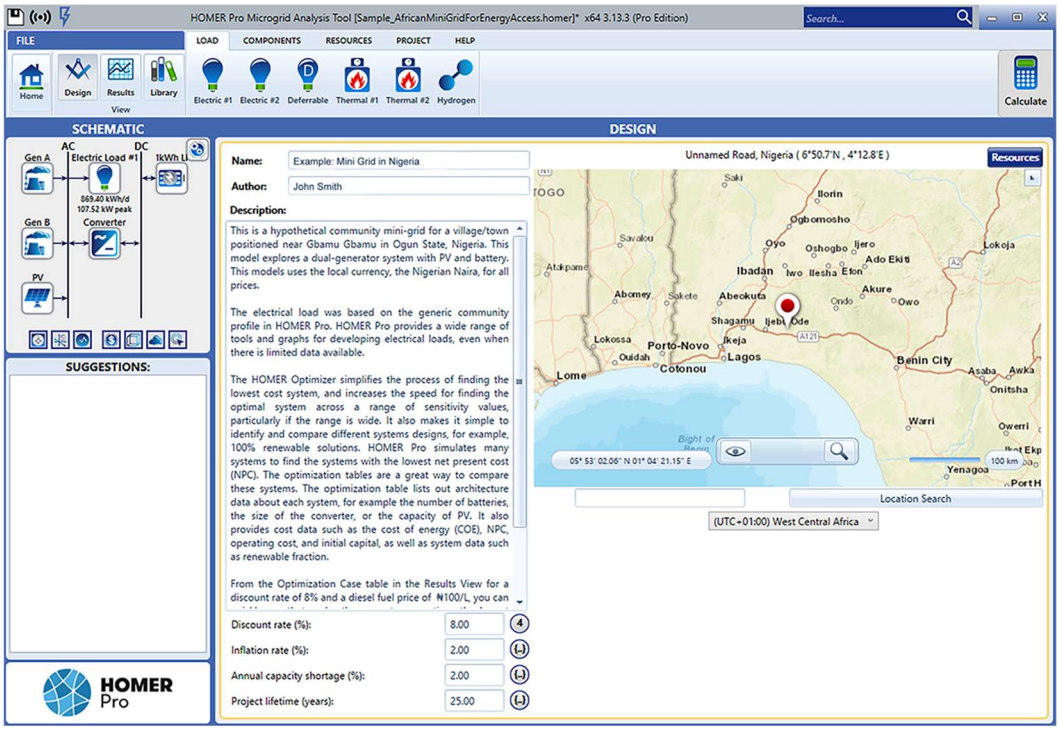


Рисунок 1.18 – Інтерфейс системи
*Джерело:* [25]

Варто згадати про один з найпотужніших інструментів для оцінки економічної ефективності проектів з інтеграції мікромереж з відновлюваною енергетикою – RETScreen. Ця програма розроблена міністерством природних ресурсів Канади, вже перекладена на 36 різних мов та має попит в усьому світі. Крім допомоги з

розрахунком економічних переваг встановлення мікромереж з ВДЕ в підприємствах та приватних домогосподарствах, програма підтримує аналіз продуктивності мережі та можливість її налаштування для отримання більш ефективної роботи. Інтерфейс програми RETScreen зображено на рисунку 1.19.

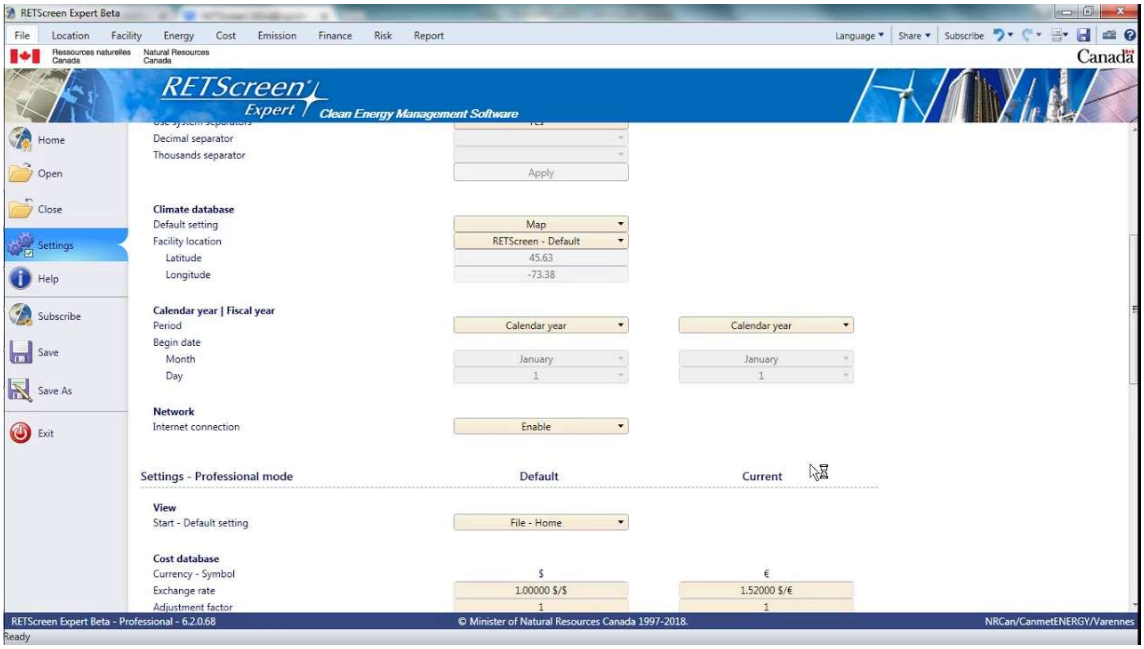


Рисунок 1.19 – Інтерфейс системи
*Джерело*: [26]

Хоча програма має досить широкий функціонал, проте інтеграція нових сторонніх інструментів або модулів є досить складним процесом через відсутність гнучкості програмного забезпечення [26].

Порівняння розглянутих застосунків наведено в таблиці 1.1.

Таблиця 1.1 Порівняння різних типів застосунків для енергоменеджменту

| № | Назва продукту | Задачі/функції | Тип застосунку | Умови використання |
|---|---|---|---|---|
| 1 | EcoStruxure Power Monitoring Expert | Промисловий аналіз енергоспоживання | Веб | Від €500 (ліцензія для бізнесу) |

Продовження Таблиці 1.1.

| № | Назва продукту | Задачі/функції | Тип застосунку | Умови використання |
|---|---|---|---|---|
| 2 | Open Energy Monitor | Моніторинг і налаштування, відкритий код | Мобільний, веб | Безкоштовне; обладнання від £70 |
| 3 | Sense Home Energy Monitor | Моніторинг домашньої енергії | Мобільний | $299 (разова плата за пристрій) |
| 4 | Tesla App | Моніторинг продуктів Tesla | Мобільний | Безкоштовно (з обладнанням Tesla) |
| 5 | GridPoint | Моніторинг і оптимізація енергоспоживання, управління обладнанням | Веб, мобільний | Платна підписка |
| 6 | Enphase Enlighten | Моніторинг і аналіз енергетичних процесів | Мобільний | Безплатно (з обладнанням Enphase) |
| 7 | ABB Ability™ Energy Manager | Промисловий енергетичний менеджмент | Веб | Від $200 на рік |
| 8 | SolarEdge Go | Контроль за сонячною енергією | Мобільний | Безплатно (для користувачів) |
| 9 | HOMER | Моделювання, оптимізація мікромереж | Настільний | Від $595 на рік |

Продовження Таблиці 1.1.

| № | Назва продукту | Задачі/функції | Тип застосунку | Умови використання |
|---|---|---|---|---|
| 10 | RETScreen | Аналіз продуктивності й економічної ефективності | Настільний | Безплатно (платні функції) |

Отже, кількість систем для моніторингу та підтримки енергетичних мікромереж з ВДЕ постійно збільшується, багато з них надають перевагу реалізації у вигляді вебзастосунків, що беззаперечно є зручним рішенням для користувачів, бо надає можливість доступу до даних у будь-який момент та з будь-якого приладу. Крім того, така реалізація дає можливість розробнику масштабувати та розвивати систему, додаючи нові функції та можливості без необхідності постійно оновлювати програмне забезпечення на пристроях користувачів, що є чудовим варіантом для швидкого розвитку проєкту.

# 2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

## 2.1 Мета та задачі дослідження

Наявна інформаційна система підтримки управління енергетичними мікромережами з ВДЕ має архітектуру, яка складається з наступних підсистем та модулів:

- підсистема моніторингу;
- підсистема планування;
- підсистема прогнозування;
- підсистема підтримки прийняття рішень;
- модуль авторизації;
- модуль адміністрування;
- модуль чату [15].

Метою даного дипломного проєкту є реінжиніринг існуючої інформаційної системи із збереженням її підсистем та структури бази даних, забезпечуючи при цьому повну функціональність системи. Проєкт передбачає використання структурного патерну MVC та сучасної об'єктно-орієнтованої парадигми програмування для покращення архітектури та ефективності роботи системи.

Для досягнення визначеної мети необхідно вирішити наступні задачі:

- проаналізувати існуючий код системи для визначення його структури, функціональних елементів та можливих недоліків;
- створити UML-діаграми класів, що візуалізують архітектуру системи й відображають взаємозв’язки між компонентами;
- обрати фреймворк для реалізації ІС;
- описати структуру класів із зазначенням їхніх атрибутів, методів і взаємозв'язків;

- реалізувати архітектуру системи на основі ООП з використанням мови PHP та патерну MVC для структурованого розділення даних, логіки й інтерфейсу;
- протестувати систему для перевірки коректності функціонування та виявлення можливих помилок у реалізації нової архітектури;

## 2.2 Огляд архітектури веборієнтованих інформаційних систем

У сучасному світі все змінюється швидко, а отже ІС повинні мати можливість розширятись та змінюватись, бути конкурентними у своїй сфері. Для швидкої розробки потрібен гнучкий, стабільний та безпечний фреймворк. Реінжиніринг програмного забезпечення – це процес створення нової функціональності або усунення помилок шляхом революційних змін, але з використанням програмного забезпечення, яке вже використовується [27].

Архітектура програмного забезпечення – це високорівневий опис інформаційної системи, що визначає її основні компоненти, їхні взаємозв’язки, а також принципи дизайну. Архітектура забезпечує розподіл функціональності між модулями, вибір технологій, забезпечення продуктивності та безпеки. Важливими аспектами є також масштабованість та можливість підтримки системи в майбутньому [28]

Архітектурні шаблони полегшують розробку та супроводження інформаційних системи. Охарактеризуємо розповсюджені архітектурні патерни програмного забезпечення.

Model-view-controller (MVC) – це архітектурний патерн, який застосовується для розробки програмного забезпечення. MVC зазвичай використовується для реалізації інтерфейсів користувачів (View), даних (Model) та логіки управління (Controller). Патерн підкреслює поділ застосунку між бізнес-логікою та

відображенням. Цей поділ забезпечує кращий розподіл праці та підтримку застосунку. Діаграму взаємодії компонентів патерну зображено на рисунку 2.1.

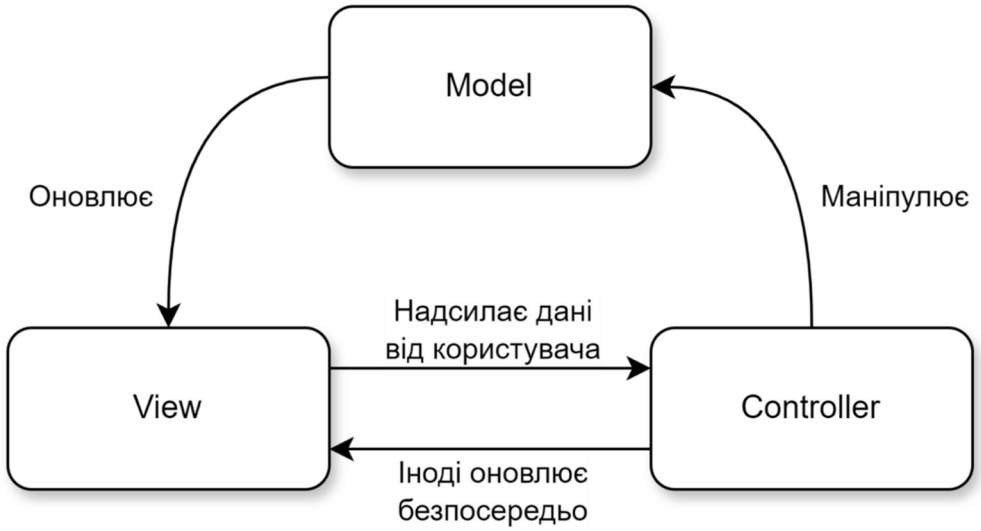


Рисунок 2.1 – Патерн MVC
*Джерело*: побудовано автором на основі статті [29]

Три частини патерну можна пояснити так:

- Модель: Керує даними та бізнес-логікою.
- Представлення: Керує шаблонами та відображенням.
- Контролер: Спрямовує команди до моделі та деталей вигляду [29].

Model-view-presenter (MVP) – це похідний патерн від MVC який розглядали вище і використовується для побудови користувацьких інтерфейсів[30]. У MVP пред’явник (Presenter) виконує функцію «посередника». MVP вся логіка відображення перекладається в зону відповідальності пред’явника (Presenter). Патерн полегшує автоматизоване тестування модулів і покращує розподілення логіки у представленні. Діаграму взаємодії компонентів шаблону зображено на рисунку 2.2.


Рисунок 2.2 – Патерн MVP
*Джерело*: побудовано автором на основі статті [30]

Компоненти патерну можна описати таким чином:

- Модель (Model) – це інтерфейс, що описує дані, які будуть відображатись або над якими буде здійснюватися інша дія в користувацькому інтерфейсі.
- Представлення (Passive View) – це пасивний інтерфейс, який відображає дані й направляє події представнику (presenter) для виконання дій над даними.
- Представник (Presenter) – працює з моделлю (Model) та представленням (View). Представник отримує дані з моделі (Model) і форматує їх для відображення у представленні [30].

Model–view–viewmodel (MVVM) – це архітектурний патерн програмного забезпечення, який забезпечує відокремлення розробки графічного інтерфейсу користувача від розробки внутрішньої логіки (моделі) таким чином, щоб вигляд не залежав від будь-якої конкретної платформи моделі [31]. Діаграму взаємодії компонентів патерну зображено на рисунку 2.3.

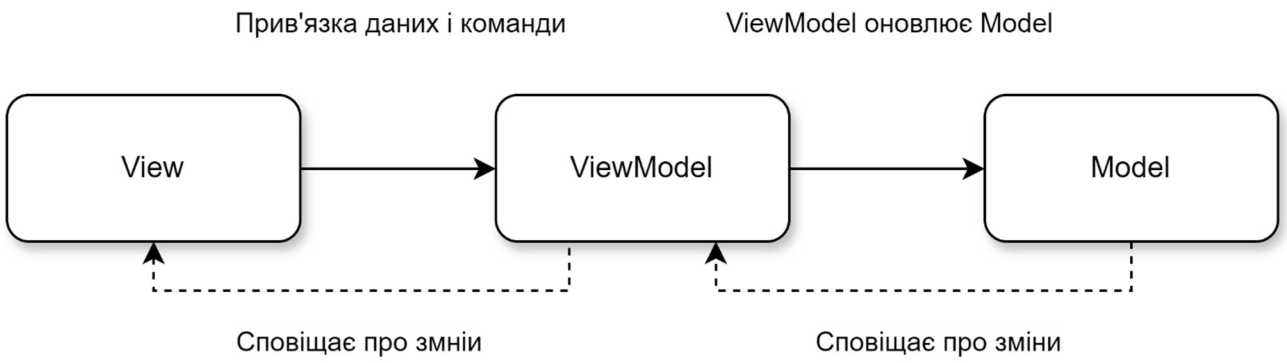


Рисунок 2.3 – Патерн MVVM
Джерело: побудовано автором на основі статті [32]

Описати компоненти патерну MVVM можна так:

- Модель (Model): Відноситься або до моделі домену, яка представляє реальний зміст стану (об’єктно-орієнтований підхід), або до рівня доступу до даних, який представляє зміст (підхід, орієнтований на дані).
- Представлення (View): Керує структурою, макетом і зовнішнім виглядом того, що користувач бачить на екрані.
- Модель представлення (View Model): В абстракції представляє публічні властивості та команди, які зв'язуються з виглядом через з’єднувач (binder). На відміну від MVC або MVP, у патерні MVVM відсутнє пряме посилання на вигляд, замість цього вигляд зв'язується з властивостями моделі представлення для обміну даними. Це вимагає використання технологій прив'язки або генерації коду. Модель представлення також може розглядатися як об'єкт передачі даних [31].

Стаття [33] описує процес проєктування та реалізацію вебзастосунку «Журнал» використовуючи патерн MVC та фреймворк Laravel. Автор статті зазначає, що використання фреймворку прискорює розробку застосунку та допомагає уникнути типових помилок при розробці застосунку. Фреймворк також допомагає скоротити більшість повторюваних і складних задач, а отже абстрагуватись від рутинних завдань, тому розробники можуть писати менше і робити більше з вищою продуктивністю. Використання фреймворку допомагає скоротити витрати на розробку.

Крім того, застосування фреймворку покриває питання безпеки. Це не вимагає від розробників глибокого розуміння кібербезпеки. Існує безліч форм кібератак, Laravel підтримує захист від більшості видів розповсюджених атак, таких як:

- SQL-інєкції;
- підробка міжсайтових запитів (CSRF);
- міжсайтовий скриптинг (XSS).

Також фреймворк слідує архітектурному патерну MVC, що полегшує структуризацію коду для розробки та підтримки проєкта у майбутньому.

У результаті роботи було створено сайт «Журнал», який дозволяє користувачам переглядати та подавати свої статті [33].

У статті [34] автори проаналізували, яку важливу роль патерн MVC відіграє в об’єктно-орієнтованій розробці. Патерн MVC широко застосовується для розробки програмного забезпечення в різних сферах життя людей. Автор статті висвітлив переваги MVC підходу для проєктування застосунків, а саме:

- можливість використання декількох представлень для однієї модулі;
- запобігання тісному зв’язку між об’єктами;
- можливість змінювати зовнішній вигляд застосунку не впливаючи на бізнес-логіку;
- підходить для підтримки застосунку.

Також автор згадав про фреймворки в загальному та відмітив, що вони зазвичай мають вбудований функціонал авторизації, управління сеансами, безпеки, локалізації, автентифікації та маршрутизації. Ці інструменти пришвидшують розробку програмного забезпечення.

У статті також проілюстровано використання архітектурного шаблона проєктування MVC двома мовами програмування C# (використано фреймворк ASP.NET) та Java (використано код мовою програмування Java).

У висновку автор відмічає, що патерн MVC забезпечує гнучкість, надійність, масштабованим та підлягає простій модернізації [34]. Автор статті не згадав про об’єктно-орієнтоване програмування, що було вказано у заголовку статті, тому це

питання залишилось не розкритим, а також було згадано у висновках про недоліки підходу, хоча в тексті самої статті відсутні згадки про них.

Архітектура мікросервісів – це підхід до розробки програмного забезпечення, при якому додаток поділяється на невеликі незалежні сервіси, які можна розробляти, розгортати та масштабувати незалежно. Кожен мікросервіс виконує певну бізнес-функцію і взаємодіє з іншими мікросервісами через чітко визначені API (англ. application programming interface). Чітко визначені API має дуже важливе значення в контексті мікросервісів, він повинен містити чіткі специфікації формату даних, якими обмінюються сервіси, а також правила обробки помилок та виключень. В таблиці № 2.1 наведено переваги та недоліки даного архітектурного стилю.

Таблиця № 2.1– Деякі переваги та недоліки архітектури мікросервісів

| **Переваги** | **Недоліки** |
|---|---|
| Масштабованість | Складність |
| Стійкість | Управління розподіленими системами |
| Швидше виведення на ринок | Накладні витрати на зв'язок |
| Технологічна незалежність | Вищі витрати на розробку та обслуговування |
| Просте обслуговування | Інтеграційні виклики |
| Швидше виведення на ринок | Складність розгортання |

Архітектура мікросервісів дозволяє організаціям розробляти та розгортати сервіси незалежно, з чіткими межами та інтерфейсами. Це може призвести до підвищення спритності, гнучкості та стійкості. Однак, архітектура мікросервісів також створює нові ризики та виклики, такі як підвищена складність, проблеми міжсервісної комунікації, проблеми узгодженості даних, ризики безпеки, прив'язка до постачальника та операційні складнощі [35].

Моноліт це традиційний архітектурний підхід, при якому застосунок будується єдиною кодовою базою, що містить безліч сервісів. Ці сервіси не є незалежно

виконуваними один від одного. Вони взаємодіють з користувачами та сервісами за допомогою різних інтерфейсів. Найбільшою перевагою монолітних застосунків є їх простота у порівнянні мікросервісними застосунками, моноліти легше тестувати, розгортати налаштовувати та розгортати. Однак при збільшенні застосунку всі переваги монолітної архітектури нівелюються, починають з'являтись проблеми з продуктивності, як застосунку, так і команди розробників, часто зміни в одному модулі призводять зміни в іншому, що викликає помилки.

Автори статі [36] дослідили продуктивність застосунків з монолітною та мікросервісною архітектурою, та дійшли висновків, що краще використовувати монолітну архітектуру якщо вам не потрібно підтримувати одночасну велику кількість підключених користувачів. До того ж мікросервісну архітектуру можуть дозволити собі гіганти індустрії з економічного погляду [36].

Отже, для реінжинірингу інформаційної системи підтримки управління енергетичними мікромережами з ВДЕ найбільш доцільним є використання патерну MVC. Цей патерн забезпечує чітке розділення даних, логіки та інтерфейсу користувача, що покращує зручність розробки та масштабованість системи. Крім того, MVC є одним із найпоширеніших підходів у веброзробці, що робить його оптимальним вибором для створення сучасного, гнучкого та ефективного вебзастосунку.

## 2.3 Огляд актуальних бекенд фреймворків

Програмний фреймворк – це готовий до використання комплекс програмних рішень, що включає дизайн, логіку та базову функціональність системи або підсистеми. Відповідно, програмний фреймворк може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти та загалом все, що полегшує створення та

поєднання різних компонентів великого програмного забезпечення чи швидке с творення готового і не обов'язково об'ємного програмного продукту [37].

Порівняємо рейтинг фреймворків зірочками на github:

- Laravel 78 тис.[38];
- Symfony 29.6 тис.[39];
- CodeIgniter 18.3 тис. [40];
- Yii 14.2 тис. [41];
- CakePHP 8.7 тис [42].

Велика кількість зірочок фреймворку Laravel вказує на те, що спільнота розробників активна, а також використовує його на багатьох проєктах, отже актуальність фреймворку не зменшується.

У таблиці 2.2. наведено порівняльні характеристики популярних фреймворків.

Таблиця 2.2 – Порівняння фреймворків

| Фреймворк / Особливості | Laravel | Symfony | CodeIgniter | Yii | CakePHP |
|---|---|---|---|---|---|
| ORM | Eloquent ORM | Doctrine | Active record | Active Record | CakePHP ORM |
| Migration | Так | Так | Так | Так | Так |
| Екосистема | Так | Так | Так | Ні | Так |
| Документація | Зрозуміла. Англійська мова. | Зрозуміла, Англійська мова. | Зрозуміла, Англійська мова. | Зрозуміла. Українська, Англійська мови. | Зрозуміла, Англійська мова |
| Архітектура | MVC | MVC | MVC | MVC | MVC |

Продовження таблиці 2.2.

| Фреймворк / Особливості | Laravel | Symfony | CodeIgniter | Yii | CakePHP |
|---|---|---|---|---|---|
| Безпека | Middleware, CSRF захист, validation, аутентифікація, авторизація, шифрування, хешування | CSRF захист, аутентифікація, контроль доступу (авторизація), хешування шифрування | CSRF захист, validation, хешування | CSRF захист, validation, аутентифікація, авторизація | Middleware, CSRF захист, validation, аутентифікація, авторизація, шифрування, хешування |
| View | Blade (or use optional Livewire, Inertia, Vue, React) | Twig | PHP file | PHP file (or optional Twig, Smarty) | PHP file, View Blocks (or use alternative PHP syntax) |
| Генерування коду | Artisan CLI | Symfony MakerBund | Spark | Gii, Yii CLI | CakePHP Console |

Також дослідимо інформацію щодо популярності запитів в Google в Україні та у світі стосовно вище розглянутих фреймворків. Сервіс Google Trends [43] дозволяє отримати інформацію про запити в Google пошуку. Розглянемо тренди за 2024 рік. На рисунку 2.4 показано діаграму популярності запитів в Україні.

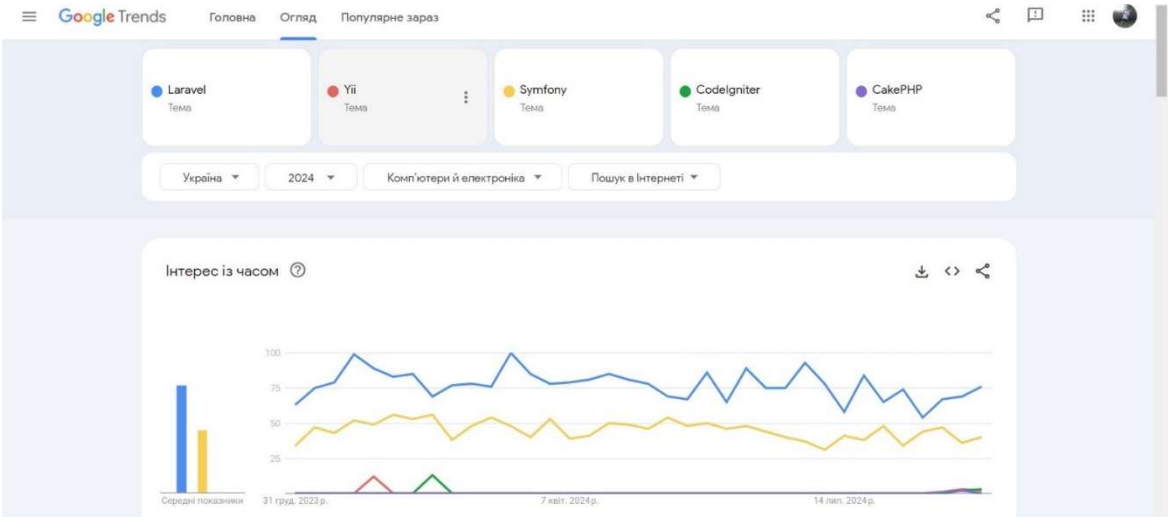


Рисунок 2.4 – Тренди в Україні за 2024 рік
Джерело: [43]

Проаналізувавши діаграму з сервісу Google Trends, можна зробити висновки, що фреймворк Laravel є найпопулярніший серед конкурентів в українському сегменті web-розробників. На рисунку 2.5 показано тренди запитів в усьому світ й він також майже точно збігається з трендами України.

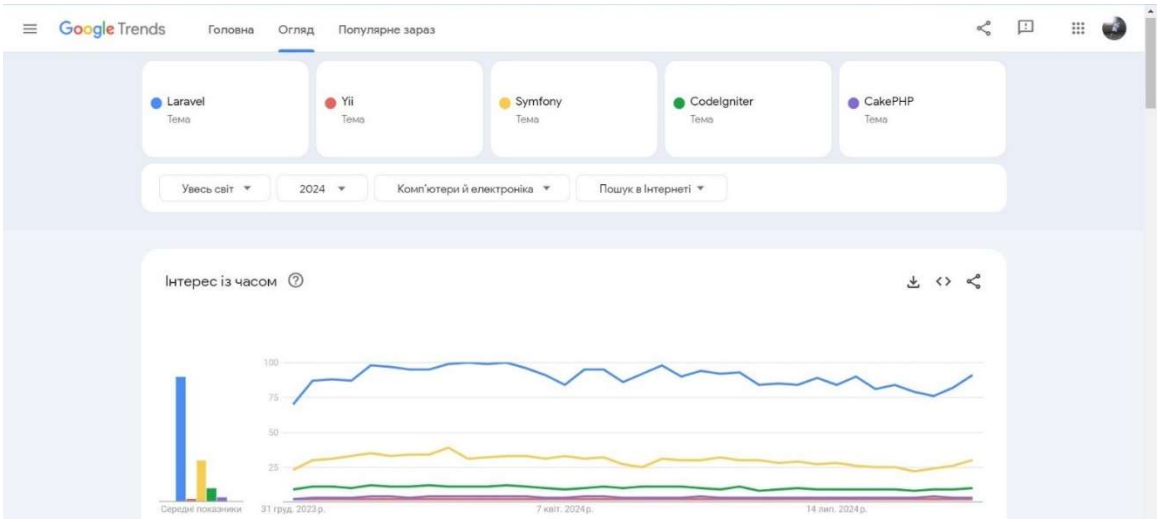


Рисунок 2.5 – Тренди в усьому світі за 2024 рік
Джерело: [43]

Проаналізувавши діаграми з сервісу Google Trends, можна зробити висновок, що фреймворк Laravel є найпопулярніший серед конкурентів як в Україні, так і у світі.

# 3 МОДЕЛЮВАННЯ ПРОЦЕСУ ІНФОРМАЦІЙНОЇ ПІДТРИМКИ СПОЖИВАЧА ПОСЛУГ ВІД ЕНЕРГЕТИЧНИХ МІКРОМЕРЕЖ

## 3.1 Діаграми нотації IDEF0

Для моделювання бізнес-процесів використовують нотацію IDEF0 (Integrated Definition for Function Modelling). Метод функціонального моделювання IDEF0 призначений для моделювання рішень, дій та діяльності організації або системи [44].

Інформаційні потоки зображені у вигляді стрілок, які вказують напрямок. Стрілки дозволяють визначити порядок виконання функцій та виявити взаємозв'язки між процесами. Графічне представлення IDEF0 допомагає чітко візуалізувати та аналізувати складні процеси, виявляти надлишкові операції та підвищувати ефективність роботи.

Контекстна діаграма A-0 процесу інформаційної підтримки споживача послуг від електричних мікромереж у нотації IDEF0 з погляду клієнта який використовує інформаційну систему підтримки управління енергетичними мікромережами з ВДЕ зображено на рисунку 3.1.

На вхід подаються дані про користувача. Інформаційна підтримка послуг надається за допомогою: інформаційної системи, бази даних, API погоди, бази правил, модуля прогнозування та клієнта. Процес обмежується вимогами до функціювання, методики прогнозування, та моделі підтримки прийняття рішень. На виході маємо: дані про мікромережу, дані про стан компонентів мікромережі, дані прогнозу погоди, дані поточного\прогнозованого споживання, дані стану управління мікромережею.


Рисунок 3.1 – Контекстна діаграма інформаційної підтримки надання послуг IDEF0
*Джерело*: побудовано автором

Діаграма декомпозиції у методології IDEF0 є засобом аналізу та моделювання процесів. Вона дозволяє деталізувати та структурувати складні процеси, забезпечуючи чітке розуміння функцій, їх взаємозв’язків та залежностей. В результаті декомпозиції формується ієрархічна структура, що охоплює ту ж саму область, що й вихідний блок. Це спрощує аналіз та управління, надаючи можливість контролювати окремі елементи та рівні системи. В діаграмі (рис 3.2.) описані підпроцеси:

- авторизація;
- вибір адреси розташування;
- вибір пункту управління;
- перегляд даних.

Інформаційна підтримка надання послуг відбувається таким шляхом. Клієнт проходить процедуру авторизації в інформаційній системі. Інформаційна система підтримки управління завантажує адреси користувача. Далі користувач обирає адресу, та пункт меню керування. А потім переглядає дані:

- про мікромережу;
- стан компонентів мікромережі;
- прогнозу погоди;
- поточного\прогнозного прогнозування;
- стан управління мікромережею.

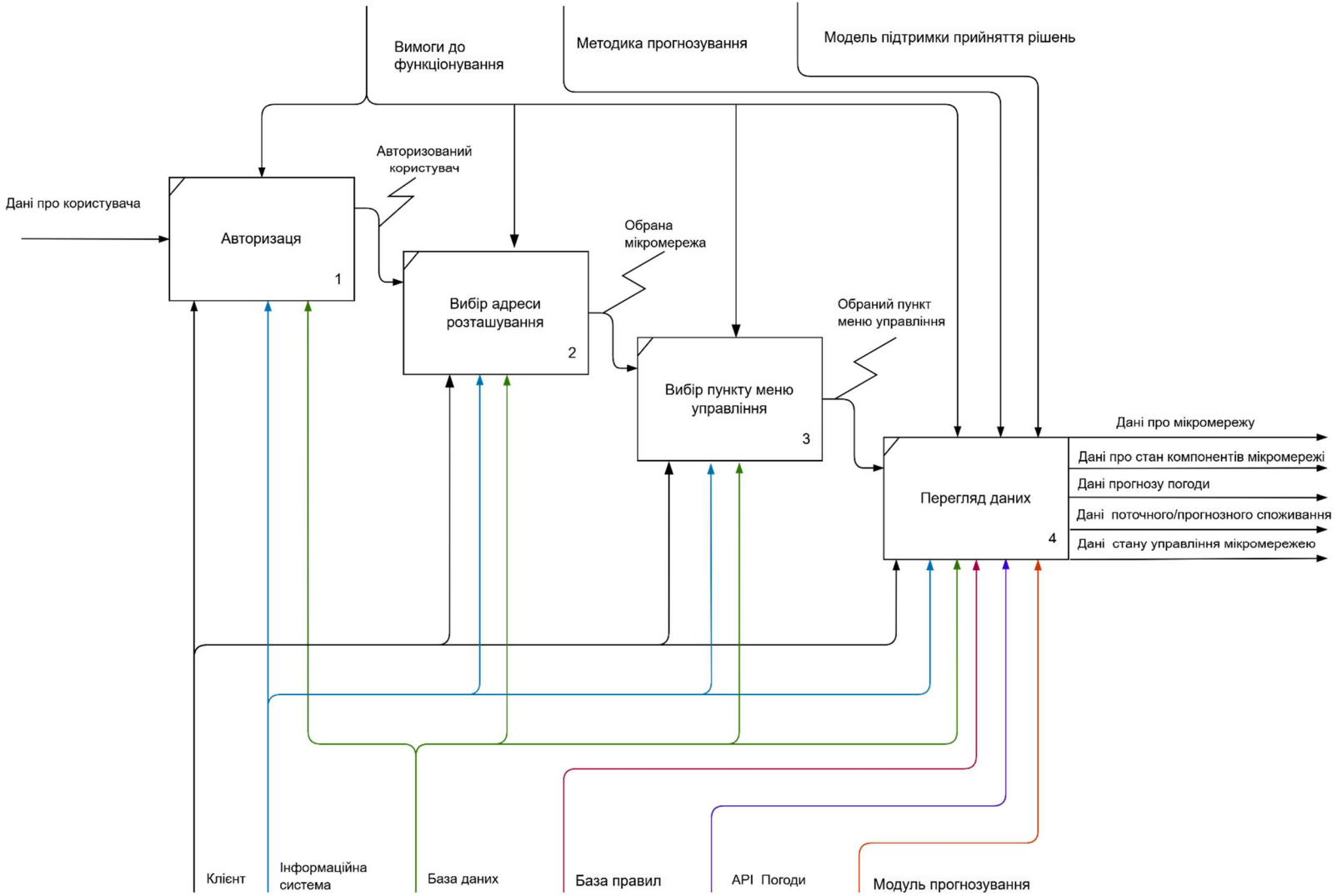


Рисунок 3.2 – Діаграма декомпозиції першого рівня
*Джерело:* побудовано автором

## 3.2 Діаграма Use Case

Діаграма варіантів використання – це поведінковий тип діаграм UML, який часто використовується для аналізу різних систем. Вони дозволяють візуалізувати різні типи ролей у системі і те, як ці ролі взаємодіють із системою [45].

Функціонал системи доступний тільки зареєстрованим користувачам інформаційної системи підтримки управління енергетичними мікромережами з ВДЕ.

Інформаційна система веборієнтована, тому доступ користувачем здійснюється за допомогою браузеру. Після авторизації користувачу доступний вебінтерфейс інформаційної системи підтримки управління енергетичними мікромережами з ВДЕ.

Користувачі інформаційної системи візуалізовані відповідно стандарту як актори:

- Адміністратор – це користувач який здійснює адміністрування інформаційної системи;
- Модератор – це користувач, який керує обліковими записами клієнтів, відповідає за укладання договорів із клієнтами, контролює стан рахунків для оплати послуг, пов’язаних з використанням інформаційної системи підтримки управління енергетичними мікромережами з ВДЕ, перевіряє дійсність укладених договорів та підтверджує реєстрацію клієнтів.
- Оператор – це користувач, який відстежує стан роботи енергетичної мережі для закріплених за ним клієнтів.
- Клієнт – це споживач послуг енергетичних мікромереж, який може володіти кількома об’єктами з встановленим обладнанням (акумуляторами, вітровими станціями, сонячними панелями). Має можливість переглядати структуру мікромережі, її стан, прогноз погоди та статус перемикачів.

Діаграма варіантів використання це візуальне представлення взаємодії між акторами і системою. Вона відображає функціональність або поведінку системи з

погляду користувача. Діаграми варіантів використання відображають функціональні вимоги до системи. А також надають високорівневий огляд функціональності системи, показуючи різні функції або можливості, які вона пропонує, і те, як користувачі або зовнішні системи взаємодіють з нею [45].

На рисунку 3.3 зображено діаграму інформаційної системи підтримки управління енергетичними електромережами з ВДЕ.

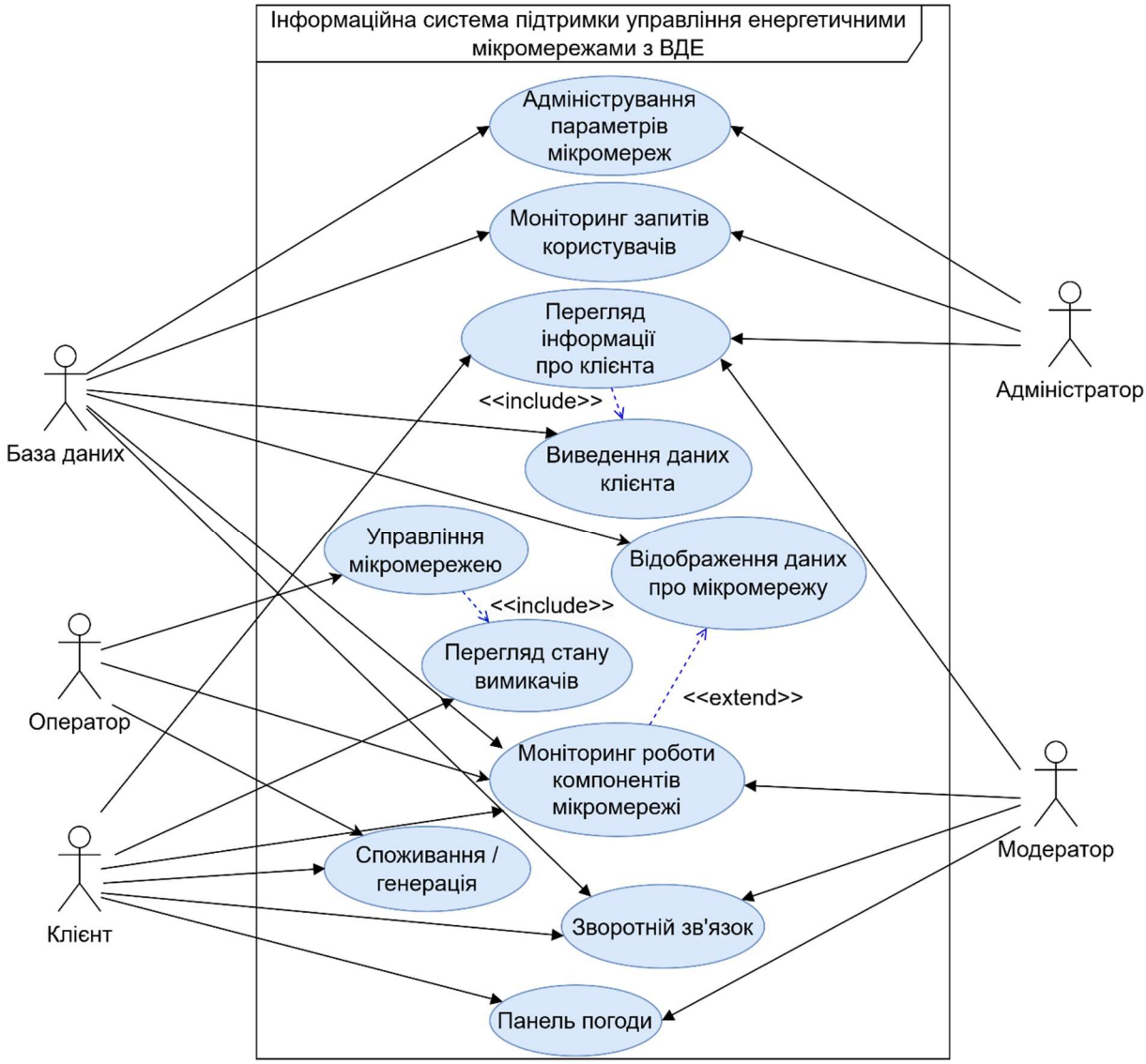


Рисунок 3.3 – Діаграма варіантів використання системи підтримки прийняття рішень в управлінні енергетичними мікромережами

*Джерело :* побудовано автором

## 3.3 Діаграми класів

Спроєктуємо діаграми класів для відображення моделей в інформаційної системи в термінології ООП. Клас (class) – категорія речей, що мають загальні атрибути та операції. Сама діаграма класів являє собою набір статичних, декларативних елементів моделі. Вона дає нам найбільш повне і розгорнуте уявлення про зв'язки в програмному коді, функціональність та інформацію про окремі класи. Додатки генеруються часто саме з діаграми класів [46].

У діаграмі класів представлені класи моделей, які позначаються як сутності (Entity), що означає, що вони відповідають за зв’язок з базою даних та відповідною таблицею в ній. У першій частині моделей зазначені поля, які зберігає клас, в другій частині після горизонтальної лінії методи в класі. А також в діаграмі зображені контролери, які виконують керуванням даними, такими як отримання даних та відправки на відображення та отриманням даних від користувача та обробку цього запиту. Додатково зображено ієрархію класів:

- Сутності наслідуються від класу Model.
- Контролери наслідуються від класу Controller.
- Класи використовують інші класи для реалізації логіки.

Діаграму класів інформаційної системи підтримки управління енергетичними електромережами з ВДЕ зображено на рисунку 3.4.

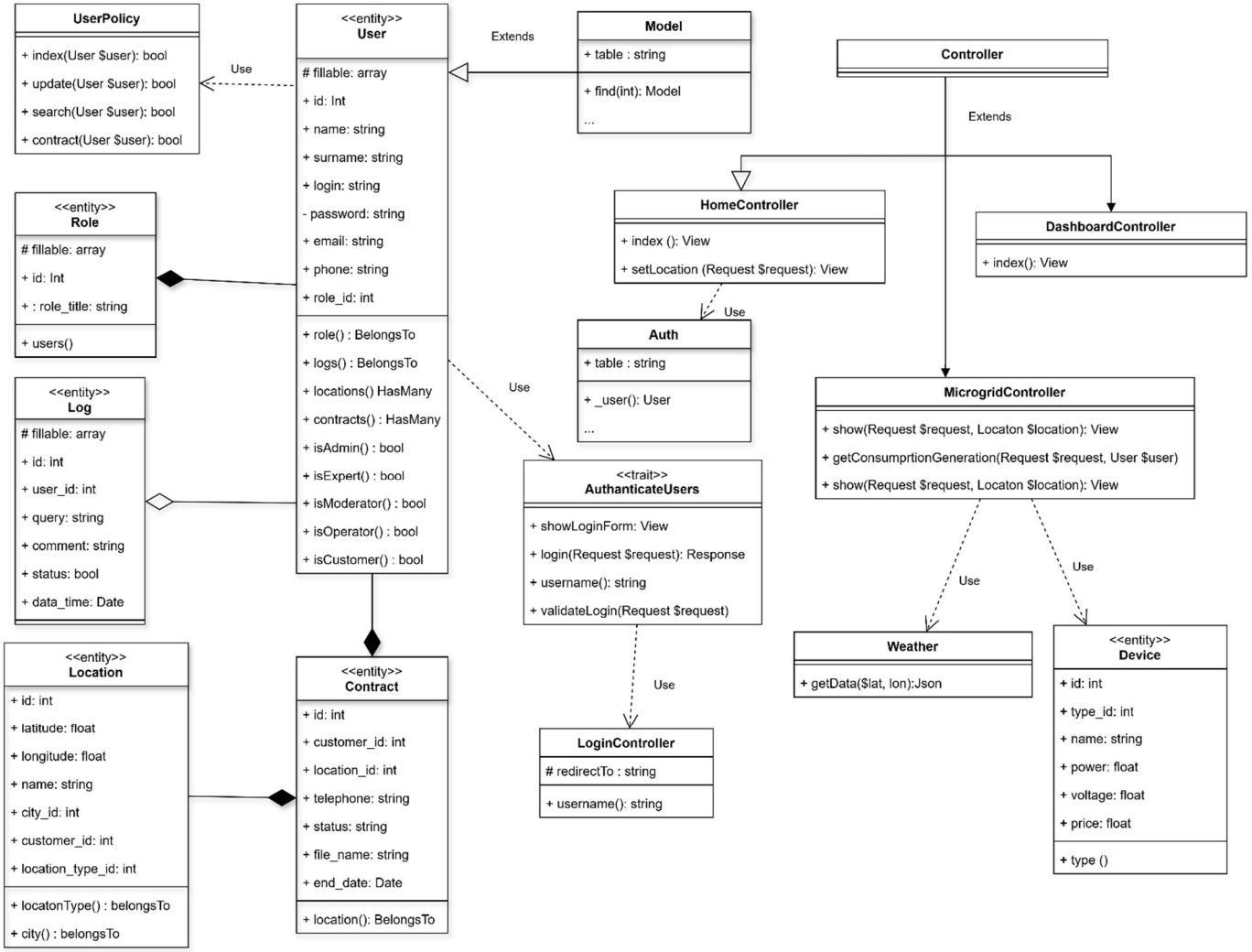


Рисунок 3.4 – Діаграма класів
*Джерело*: Побудовано автором

Таблиця 3.1 Зв’язки та призначення класів у системі управління енергетичними мікромережами

| Клас | Батьківський клас | Дочірній клас | Тип зв’язку | Призначення |
|---|---|---|---|---|
| Controller | — | HomeController, DashboardController, GridController, LoginController | Extends | Базовий клас для контролерів, що забезпечує управління запитами та логікою. |
| User | — | UserPolicy | Use | Сутність, що представляє користувача системи. |
| UserPolicy | User | — | Use | Визначає політики доступу для об’єктів, пов’язаних із користувачем. |
| Auth | — | AuthenticateUsers | Use | Служба аутентифікації для користувачів системи. |
| AuthenticateUsers | Auth | — | Trait | Реалізація механізмів аутентифікації користувачів. |
| HomeController | Controller | — | Extends | Контролер для відображення домашньої сторінки або головного інтерфейсу. |
| DashboardController | Controller | — | Extends | Контролер для управління інформаційною панеллю (dashboard). |
| GridController | Controller | Device, Weather | Use | Контролер для управління енергетичною мікромережею, включаючи пристрої та погоду. |
| LoginController | Controller | — | Extends | Контролер для обробки входу користувачів до системи. |

Продовження таблиці 3.1.

| Клас | Батьківський клас | Дочірній клас | Тип зв’язку | Призначення |
|---|---|---|---|---|
| Log | — | — | Entity | Сутність, що представляє журнал подій системи. |
| Role | — | — | Entity | Сутність, що відповідає за ролі та права доступу користувачів. |
| Location | — | — | Entity | Сутність, що зберігає інформацію про географічне розташування. |
| Contract | — | — | Entity | Сутність, що представляє контракти або угоди, пов’язані з мікромережею. |
| Device | GridController | — | Use | Компонент, що представляє пристрої в енергетичній мікромережі. |
| Weather | GridController | — | Use | Компонент для отримання та обробки погодних даних, які впливають на мережу. |

# 4 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 4.1 Архітектура інформаційної системи - загальна схема

На рисунку 4.1 зображено архітектуру інформаційної системи: основні компоненти, вебінтерфейс, проміжне програмне забезпечення, яке виконує перевірку чи авторизований користувач, контролери, які обробляють запити моделі в якій реалізовується логіка та доступ даних.

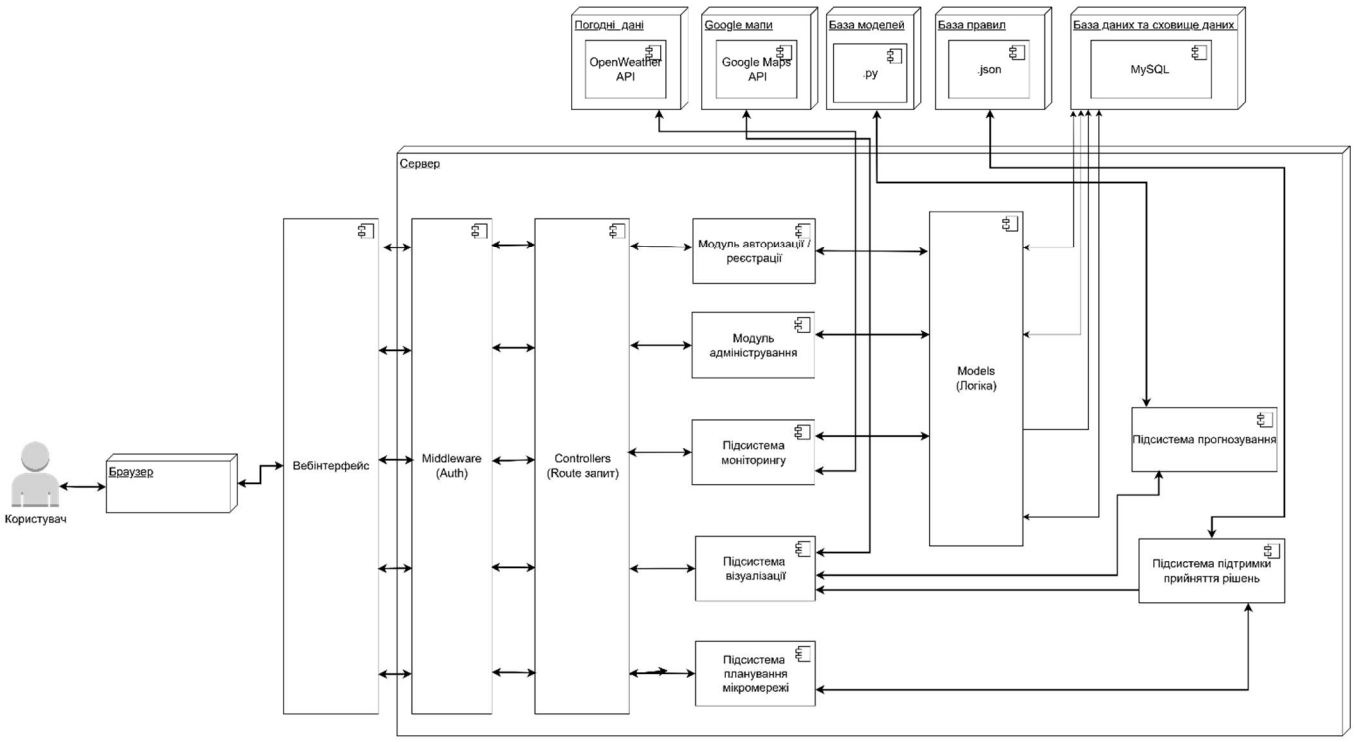


Рисунок 4.1 – Архітектура інформаційної системи
Джерело: Побудовано автором

## 4.2 Реінжиніринг бази даних інформаційної системи

Реінжиніринг бази даних проводиться для відповідності стандартів фреймворку «Laravel». В фреймворку «Laravel» є правила іменування, як і в кожного фреймворку. Зараз сфокусуємось на базі даних, а саме назвах таблиць, первинних та зовнішніх ключів. Приведення до стандартів іменування фреймворку «Laravel» допоможе спростити подальшу розробку та спростить рутинні дії з ручним вказанням назв таблиць бази даних та ключів, чого можна уникнути слідуючи стандартам. Таблиці повинні називатись в множині. Якщо в назві таблиці присутні два слова або більше слова, то використовують стиль написання «snake_case». Для первинних ключів використовують завжди назву «id». Для зовнішніх ключів використовують назву таблиці в однині нижнє підкреслення «id», наприклад, «user_id». На рисунку 4.2. зображено приклад іменувань таблиць та зовнішніх ключів «location_id».

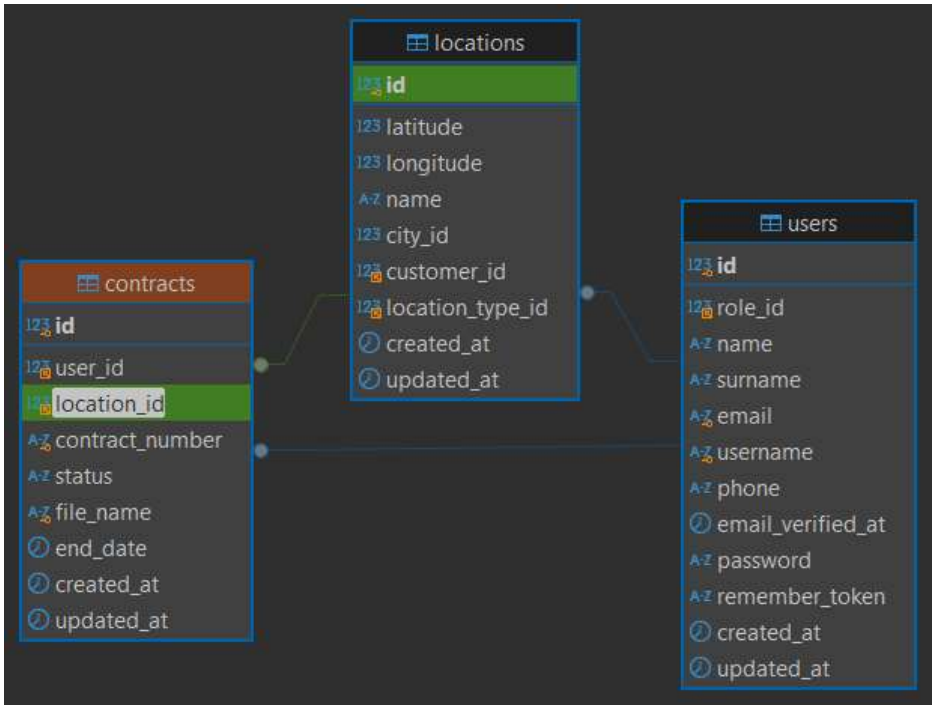


Рисунок 4.2 – Іменування таблиць та ключів
*Джерело*: Побудовано автором

Для відстеження змін в таблицях бази даних використовують міграції, це своєрідна система контролю версій тільки для бази даних, що спрощує спільну підтримку змін в базі даних. Для генерування файлів міграцій використовується консольна команда «artisan», наприклад, «php artisan make:migration create_contacts_table». Але було вирішено піти іншим шляхом та створити міграції разом з моделлю. Модель потрібно називати в однині, як зображено на рисунку 4.3. При такій команді створюється відповідна міграція для таблиці.


Рисунок 4.3 – Створення міграцій
*Джерело*: Побудовано автором

Варто згадати, що міграції можна застосовувати й повертатись на попередні стани. Це забезпечує відповідність бази даних для усіх учасників розробки. Далі потрібно відредагувати створений код у щойно створеному файлі (фрагмент коду 4.1).

Фрагмент коду 4.1 – Файл міграції

```
1. public function up(): void
2. {
3. Schema::create('locations', function (Blueprint $table) {
4. $table->id();
5. $table->float('latitude');
6. $table->float('longitude');
7. $table->string('name');
8. $table->bigInteger('city_id');
9. $table->foreignId('customer_id')->constrained('users');
10. $table->foreignId('location_type_id')->constrained('location_types');
11. $table->timestamps();
12. });
13. }
```

Повний код наведено в додатку Б.

В методі «up» створюємо відповідні поля із зазначенням типів та обмежень для полів таблиці. В рядку 10 (фрагмент коду 4.1) створюється зовнішній ключ

«location_type_id» для таблиці «location_types» назву таблиці можна було не вказувати, так як вона створена по стандарту.

Для застосування міграції потрібно виконати консольну міграцію, виконання міграції зображено на рисунку 4.4. Після виконання міграції створиться відповідна таблиця в базі даних.


Рисунок 4.4 – Застосування міграцій
*Джерело*: Побудовано автором

Далі занесемо дані з попередньої бази даних, за допомогою використання «seed» занесемо дані в щойно створену таблицю. Для створення файлу використано команду «php artisan make:seeder LocationSeeder» (фрагмент коду 4.2).

Фрагмент коду 4.2 – Внесення даних в таблицю «locations»

```
1. DB::table('locations')->insert([
2. ['latitude' => 50.9209, 'longitude' => 34.7977, 'name' => 'Заправна станція', 'city_id' => 1,
   'customer_id' => 4, 'location_type_id' => 4],
3. ['latitude' => 50.9216, 'longitude' => 34.8003, 'name' => 'Приватний будинок', 'city_id' =>
   1, 'customer_id' => 4, 'location_type_id' => 2],
4.  ['latitude' => 46.4163, 'longitude' => 30.711, 'name' => 'Нова локація', 'city_id' => 2,
   'customer_id' => 5, 'location_type_id' => 3],
5. ['latitude' => 50.9048, 'longitude' => 34.8029, 'name' => 'Нова не активна локація',
   'city_id' => 1, 'customer_id' => 6, 'location_type_id' => 4],
6. ]);
```

Повний код наведено в додатку Б.

## 4.3 Створення архітектурних моделей інформаційної системи

Моделі використовуються для зв’язку з відповідної таблиці в базі даних, та логіки пов’язаної з нею. Модель «Location» наслідується від моделі «Model», яка має багато методів для взаємодії з базою даних. В створеній моделі «Location» потрібно

вказати захищене поле «fillable», оскільки за замовчуванням всі властивості в моделі захищенні від масового вставлення даних, потрібно вказати ці поля, щоб мати можливість їх оновлювати. В рядках 5, 7 та 9 (фрагмент коду 4.3) вказані зв’язки один до одного «belongsTo» з моделями які працюють з відповідними таблицями бази даних. В методі «user» рядок 9 довелось вказати зовнішній ключ, тому тут недотримано конвенцію про іменування ключів до відповідних таблиць, але фреймворк «Laravel» дозволяє вказувати свої ключі коли є не відповідність.

Фрагмент коду 4.3 – Файл «Location»

```
1. class Location extends Model
2. {
3.     protected $fillable = ['id', 'latitude', 'longitude', 'name', 'city_id', 'customer_id',
   'location_type_id'];
4.
5.     public function city() { return $this->belongsTo(City::class); }
6.
7.     public function locationType() { return $this->belongsTo(LocationType::class); }
8.
9.     public function user() { return $this->belongsTo(User::class, 'customer_id'); }
```

Повний код наведено в додатку Б.

## 4.4 Реалізація логіки контролерів

В контролері виконується обробка запиту від користувача. Контролери створюються за допомогою консольної команди, наприклад, «php artisan make:controller MicrogridController». Виконання команди зображено на рисунку 4.5.


Рисунок 4.5 – Створення контролеру
*Джерело*: Побудовано автором

Для взаємодії з користувачем потрібно оголосити маршрут для контролера у відповідному файлі «web.php», який знаходиться в каталозі «routes» (фрагмент коду 4.4).

Фрагмент коду 4.4 – Роутинг

```
1. Route::prefix('dashboard')->middleware(['auth'])->group(function () {
2. Route::match(['get', 'post'], '/microgrid/{location}', [MicrogridController::class, 'show'])-
   >name('microgrid.show'); });
```

Повний код наведено в додатку Б.

В першому рядку (фрагмент коду 4.4) вказуємо три функції, які задають префікс роута в адресному рядку, в даному випадку «dashboard». Наступна функція – це middleware, яка перевірятиме, чи користувач увійшов в інформаційній систему, якщо ні, то поверне помилку, що доступ заборонено. Далі вказуємо групу, це означає, що всі маршрути в середині будуть підпорядковуватись всім вище зазначеним правилам.

На другому (фрагмент коду 4.4) рядку зазначаємо, якого типу запит маршрут обробляє. Цей метод обробляє запити типу «GET» та «POST». Наступним параметром вказуємо «url» запиту, в якого є параметр «location». У такому випадку метод в контролері знайде відповідну модель («Location») та дістане дані з таблиці.

В третьому рядку (фрагмент коду 4.4) в масиві вказуємо контролер та метод, який обробить запит та назву маршруту для легшого звернення до маршруту з виду чи переадресації на нього. Далі створимо сам метод «show» в контролері «MicrogridController» (фрагмент коду 4.5).

Фрагмент коду 4.5 – Створення методу «Show»

```
1.    public function show(Request $request, Location $location) {
2.        $request->session()->put('location_id', $location->id);
3.        $user = null;
4.        $customer_id = $request->input('customer_id');
5.        if ($customer_id) { $user['id'] = $customer_id; } else { $user = Auth::user(); }
6.        $locations = Location::forUser($user['id'])->get();
7.        $location->load([ 'locationType:id,title', 'city:id,name,country_id,longitude,latitude',
8.            'city.country:id,name', ]);
9.        $components = $location->getGeneralDeviceConfig($location->id);
10.       $weatherData = new Weather($location->city->longitude, $location->city->latitude);
11.       $weather = $weatherData->getWeather();
12.       return view('dashboard.microgrid.show', compact('location', 'locations', 'components',
    'weather')); }
```

Повний код наведено в додатку Б.

Метод «show» класу «MicrogridController» приймає два параметри «$request» типу «Request» та «$location» типу «Location».

В рядку 2 (фрагмент коду 4.5) кладемо в сесію ідентифікатор локації. На рядку 3 створюємо змінну для користувача, якщо в запиті прийде ідентифікатор користувача, то буде використано його, якщо ні, то отримано ідентифікатор із поточного користувача рядок 5.

В рядку 6 (фрагмент коду 4.5) отримуємо список локацій користувача. В 7 рядку коду завантажуємо поточну локацію. В рядку 9 завантажуємо компоненти електромережі. В рядку 10, 11 отримуємо погоду з утиліти «Weather», яка по переданим координатам отримує погоду з API сервісу погоди.

В рядку 12 передаємо виду «dashboard.microgrid.view» отримані дані за допомогою методу compact, який перетворить дані в масив, якщо це потрібно.

В зазначеному лістингу вище використовується «scope» у рядку 6 «forUser», далі поясними його. Після знаку «::» (статичний виклик) використовуємо «scope», в якому прописали логіку вибірки. В цьому запити робимо вибірку всіх доступних локацій для користувача, щоб мати можливість перемикатись на будь-яку локацію з віджета. В рядку 7 (фрагмент коду 4.6). в методі «with» прописуємо потрібні зв’язки та поля, які потрібно повернути.

Фрагмент коду 4.6.– Метод «scopeForUser»

```
1.    public function scopeForUser($query, $userId) {
2.        return $query->with([
3.                'locationType:id,title',
4.                'city:id,name,country_id',
5.                'city.country:id,name',
6.            ])
7.            ->whereHas('user', function ($query) use ($userId) {
8.                $query->where('customer_id', $userId);
9.            })->select(['id', 'name', 'location_type_id', 'city_id', 'customer_id'])
10.           ->orderBy('id', 'asc');
11.   }
```

Повний код наведено в додатку Б.

## 4.5 Реалізація інтерфейсу користувача

Всі види розташовані в каталозі «resources/views». Розташуємо файл за таким шляхом «resources/views/dashboard/microgrid/show.blade.php». Фреймворк використовує шаблонізатор, тому маємо таке розширення файлу. Нижче в наведеному коду (фрагмент коду 4.6) показано каркас виду в якому багато «HTML».

Фрагмент коду 4.6 – Файл виду «show»

```
1. @extends('layouts.dashboard')
2. @section('title', 'Управління мікромережею')
3. @section('content')
4.  // html
5. @endsection
6. @section('footer-scripts')
7. @vite(['resources/js/execute.js', 'resources/js/chart.js'])
8. @endsection
```

Повний код наведено в додатку Б.

Директива на рядку 1 (фрагмент коду 4.6) означає, що успадкована від базового шаблона, в якому вже є шапка та футер. Директива «@section» – це задані секції в базовому шаблоні, які розміщені у відповідних місцях. Директива «@vite» компілює та підключає файли стилів та «JavaScript». На рисунку 4.7 зображено частину інтерфейсу виду «show». Шапка сторінки підключена із загального шаблону, контент отриманий із контролеру.

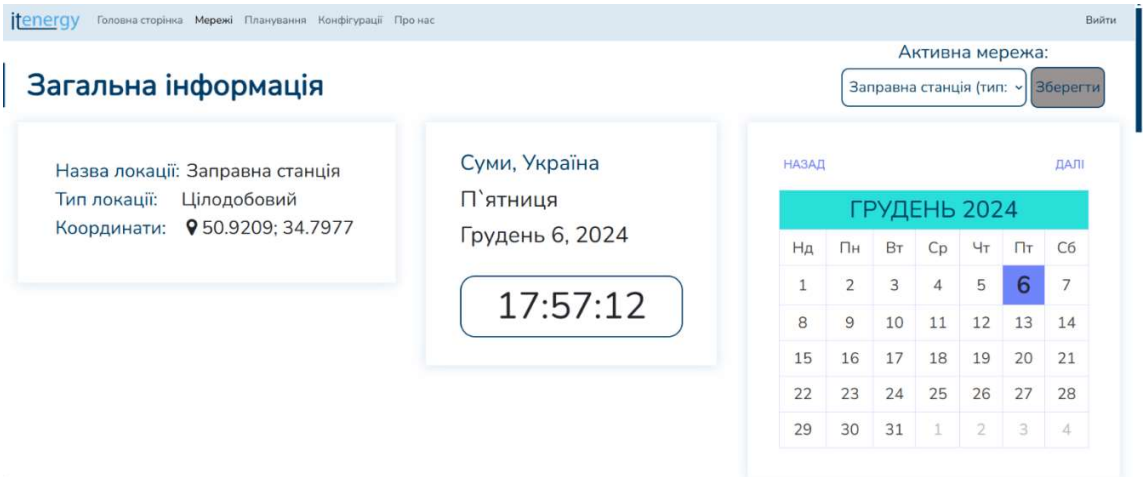


Рисунок 4.6 – Частина інтерфейсу
Джерело: Побудовано автором

## 4.6 Тестування веборієнтованої інформаційної системи

Тестування відіграє ключову роль у створенні будь-якого програмного продукту. В інформаційній системі було виконано тестування в п’ятьох основних напрямах:

- тестування повідомлень при введенні користувачем некоректних даних;
- тестування функціоналу карти параметрів створюваної мережі;
- тестування запита на реєстрацію нового користувача;
- тестування форми зворотного зв'язку;
- тестування коректного отримання даних прогнозу погоди через API.

У результаті відправки форми авторизації з коректними даними користувача перенаправляє до сторінки панелі керування (рис 4.8).

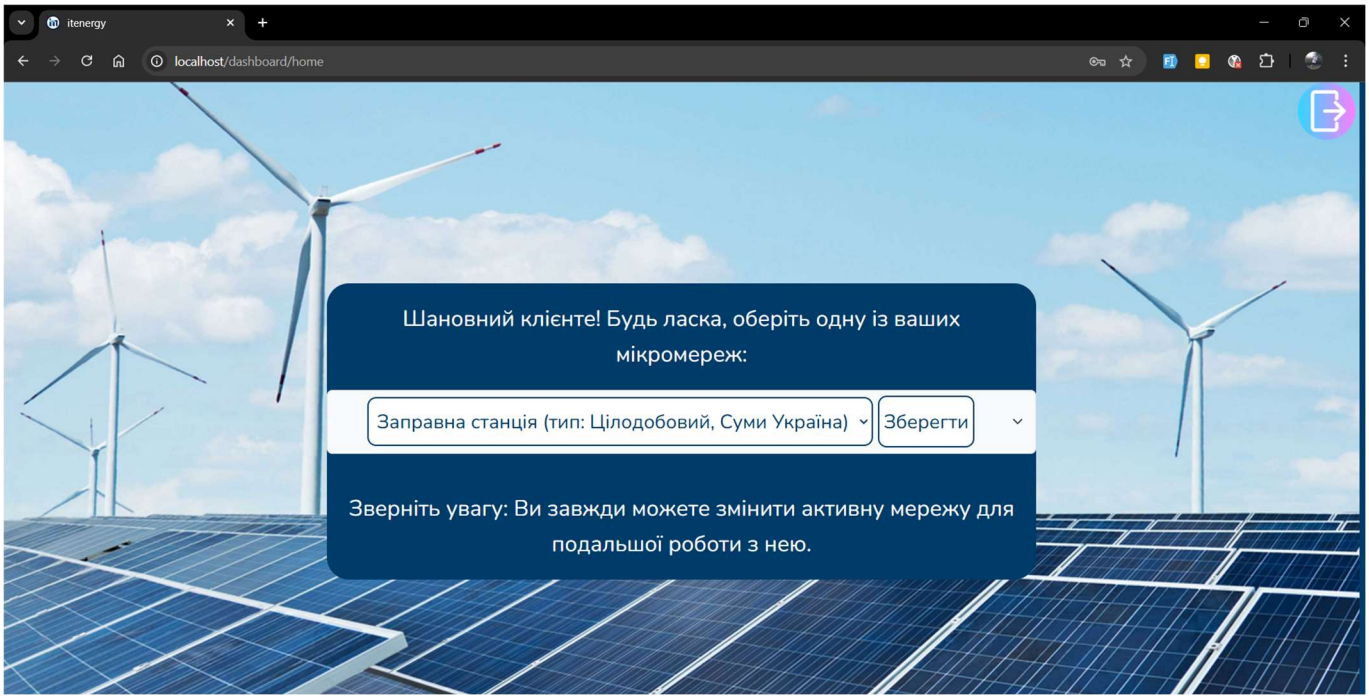


Рисунок 4.7 – Відправка коректних даних
*Джерело*: Побудовано автором

При введені некоректних даних – отримуємо повідомлення про помилку невідповідності даних (рис. 4.9).

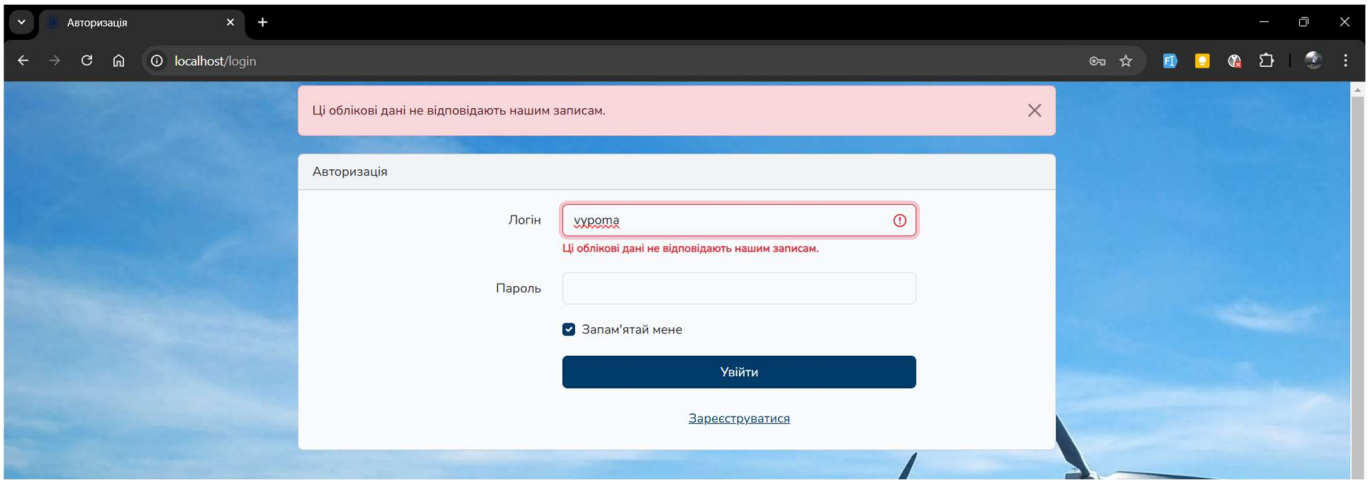


Рисунок 4.8 – Введення некоректних даних
*Джерело*: Побудовано автором

При введені коректних даних клієнта, в якого закінчився контракт, також отримуємо помилку, як і заплановано (рис. 4.10).

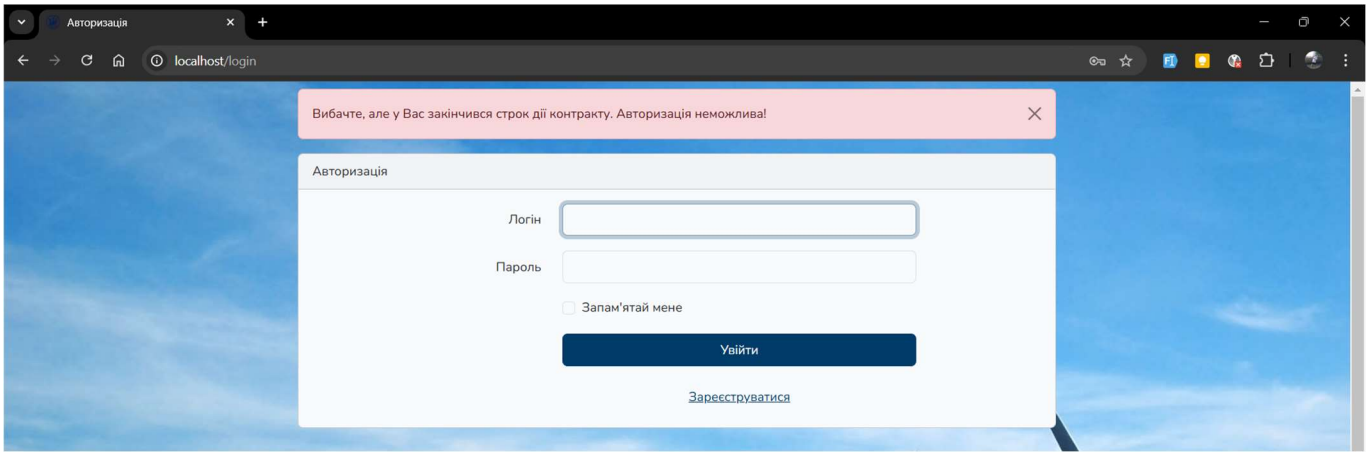


Рисунок 4.9 – Помилка при авторизації після завершення контракту
*Джерело*: Побудовано автором

Заповнивши коректо форму реєстрації нового користувача, отримуємо відповідний запис з даними в базі даних та отримуємо на електронну пошту модератора повідомлення про реєстрацію нового користувача (рис. 4.11).

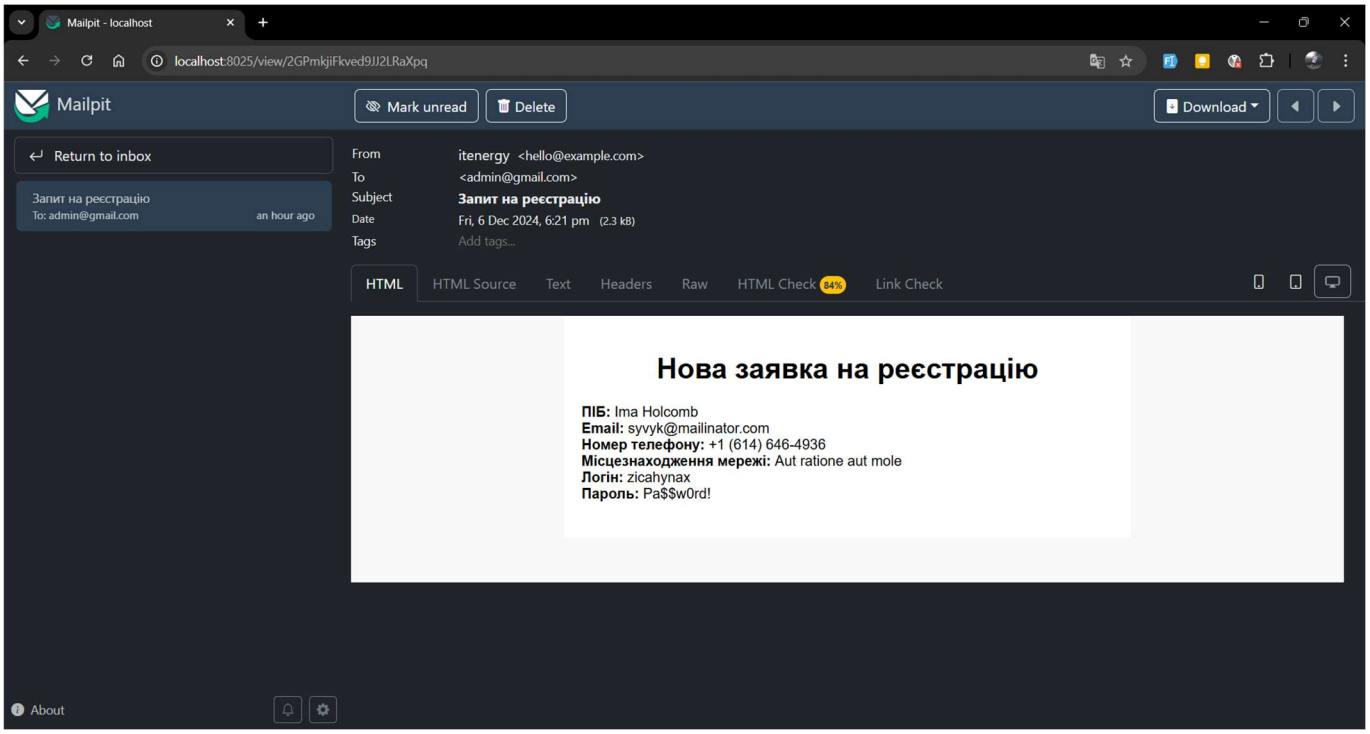


Рисунок 4.10 – Заявка на реєстрацію
*Джерело*: Побудовано автором

При тестуванні функціоналу карти створюваної мережі обираємо довільну площу та отримуємо розрахунки вартості встановлення нової мікромережі (рис.4.12).

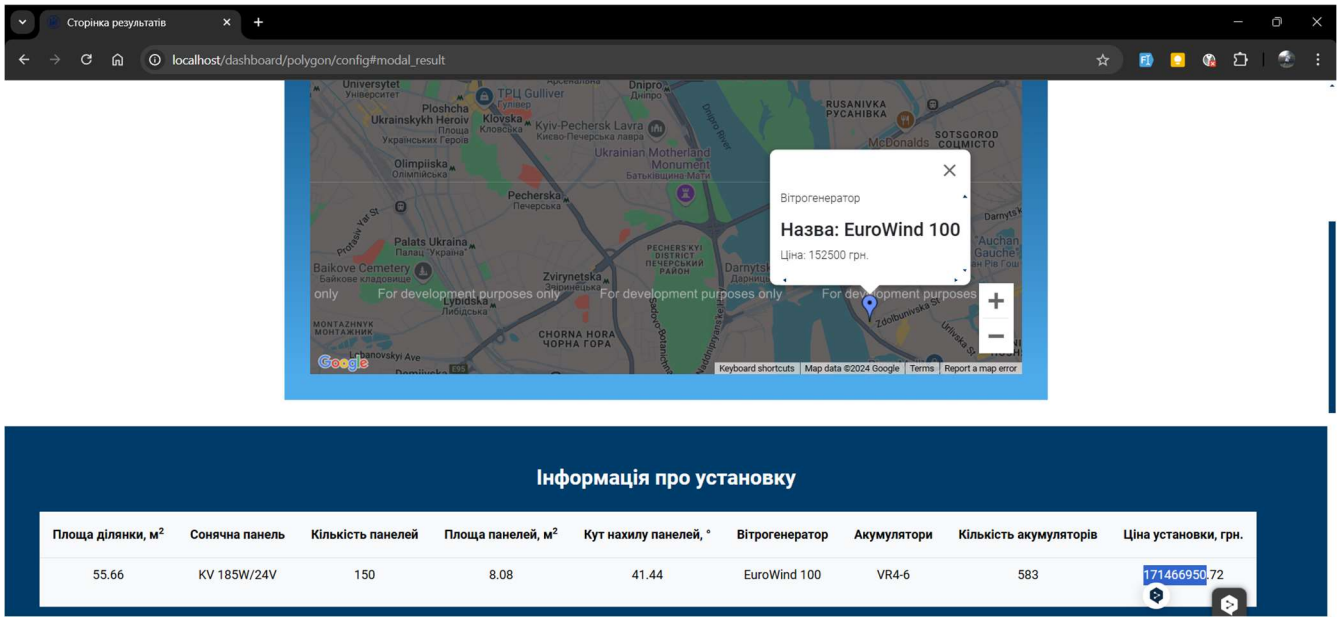


Рисунок 4.11 – Розрахунок вартості
*Джерело*: Побудовано автором

При тестуванні форми зворотного зв'язку, заповнивши всі поля форми, отримуємо відповідне повідомлення звернення на електронну пошту модератора (рис. 4.13).

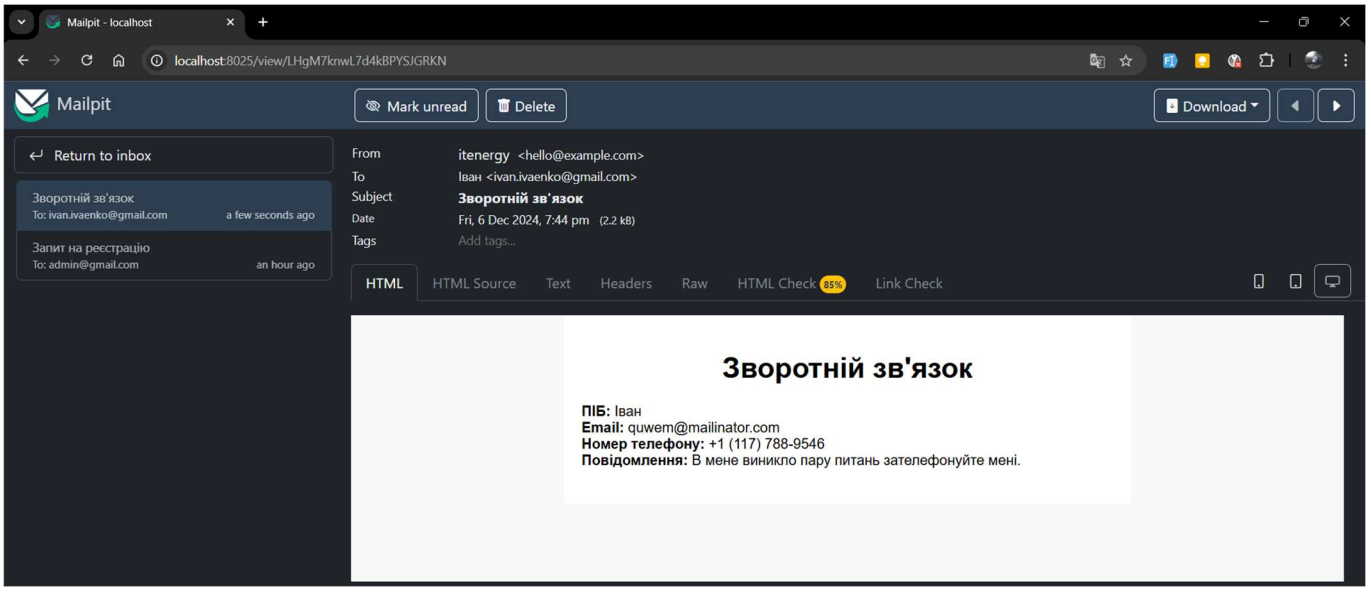


Рисунок 4.12 – Тестування зворотного зв'язку
*Джерело*: Побудовано автором

Система відображає прогноз погоди, отримуючи дані з сервісу «weatherapi» через API. Дані відображаються коректно (рис 4.14).

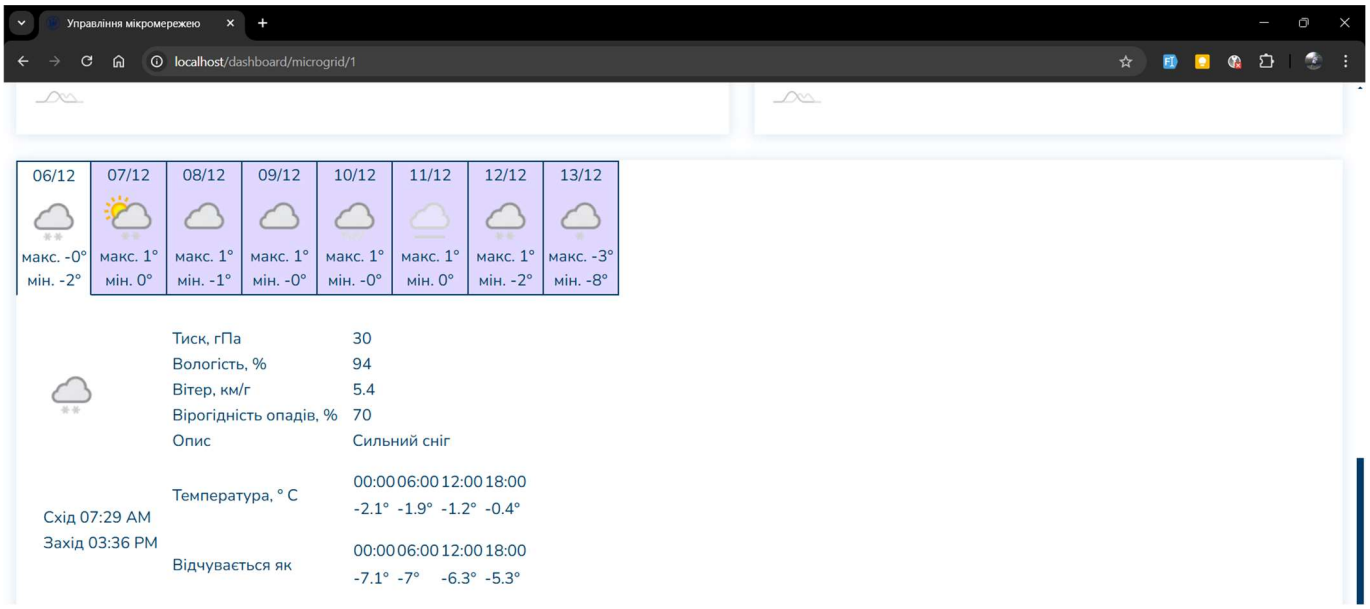


Рисунок 4.13 – Відображення погоди
Джерело: Побудовано автором

# ВИСНОВКИ

У процесі реінжинірингу інформаційної системи підтримки управління енергетичними мікромережами з відновлювальними джерелами енергії було збережено її основні підсистеми та структури, що забезпечило повну функціональність системи. Разом із цим реалізовано вдосконалення архітектури та підвищення ефективності роботи системи за рахунок застосування сучасних технологій і підходів.

На етапі підготовки до реінжинірингу було проведено моделювання процесу інформаційної підтримки споживача послуг від енергетичних мікромереж. У рамках цього моделювання створено діаграми IDEF0, Use Case та діаграму класів, які дозволили детально описати бізнес-процеси, функціональність системи та взаємодію її компонентів. Це забезпечило структурний підхід до реінжинірингу та стало основою для подальшого вдосконалення архітектури системи.

При реінжинірингу:

- використано патерн MVC для чіткого розділення логіки, представлення та обробки даних. Це дозволило спростити підтримку та розширення системи, а також забезпечило більшу прозорість архітектури.
- застосовано об’єктно-орієнтоване програмування, що забезпечило модульність та повторне використання коду. Реалізація ключових підсистем через класи та об’єкти сприяла підвищенню надійності та масштабованості системи.
- використано фреймворк Laravel, який забезпечив швидкий процес розробки завдяки наявності готових інструментів, таких як маршрутизація, робота з базами даних через ORM Eloquent, та підтримка сучасних стандартів безпеки.
- знижено час на виконання рутинних операцій завдяки оптимізованій логіці та ефективній взаємодії з базами даних.

У результаті цих дій:

- покращено архітектуру системи, що забезпечило більшу гнучкість у додаванні нових функцій та полегшило її інтеграцію з іншими компонентами.

Таким чином, завдяки реінжинірингу вдалося не лише зберегти функціональність системи, але й вдосконалити інформаційну систему управління енергетичними мікромережами з відновлювальними джерелами енергії.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


[1] В. Шендрик, Ю. Парфененко, В. Майковський, Д. Юрченко, і С. Шендрик, «ПІДСИСТЕМА ЗБОРУ, ЗБЕРІГАННЯ ТА ВІЗУАЛІЗАЦІЇ ОПЕРАТИВНИХ ДАНИХ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ УПРАВЛІННЯ МІКРОГІД», *Computer systems and information technologies*, вип. 2, с. 69–77, Чер 2022, doi: 10.31891/csit-2022-2-8.

[2] «‘Розумні’ мікромережі – важливий інструмент посилення стійкості енергосистеми | Міністерство енергетики України». Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://mev.gov.ua/novyna/rozumni-mikromerezhi-vazhlyvyy-instrument-posylennya-stiykosti-enerhosystemy

[3] S. Karnouskos, P. G. da Silva, і D. Ilic, «Developing a web application for monitoring and management of Smart Grid neighborhoods», в *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, IEEE, Лип 2013, с. 408–413. doi: 10.1109/INDIN.2013.6622919.

[4] W.-J. Shyr, L.-W. Zeng, C.-K. Lin, C.-M. Lin, і W.-Y. Hsieh, «Application of an Energy Management System via the Internet of Things on a University Campus», *EURASIA Journal of Mathematics, Science and Technology Education*, вип. 14, вип. 5, Лют 2018, doi: 10.12973/ejmste/80790.

[5] A. D. Ashkezari, N. Hosseinzadeh, A. Chebli, і M. Albadi, «Development of an enterprise Geographic Information System (GIS) integrated with smart grid», *Sustainable Energy, Grids and Networks*, вип. 14, с. 25–34, Чер 2018, doi: 10.1016/j.segan.2018.02.001.

[6] E.-K. Lee, W. Shi, R. Gadh, і W. Kim, «Design and Implementation of a Microgrid Energy Management System», *Sustainability*, вип. 8, вип. 11, с. 1143, Лис 2016, doi: 10.3390/su8111143.

[7] Z. Ullah, S. Wang, G. Wu, M. Xiao, J. Lai, і M. R. Elkadeem, «Advanced energy management strategy for microgrid using real-time monitoring interface», *J Energy Storage*, вип. 52, 2022, doi: 10.1016/j.est.2022.104814.

[8] J. A. A. Silva, J. C. López, C. P. Guzman, N. B. Arias, M. J. Rider, і L. C. P. da Silva, «An IoT-based energy management system for AC microgrids with grid and security constraints», *Appl Energy*, вип. 337, 2023, doi: 10.1016/j.apenergy.2023.120904.

[9] L. Guo, N. M. M. Abdul, M. Vengalil, K. Wang, і A. Santuzzi, «Engaging Renewable Energy Education Using a Web-Based Interactive Microgrid Virtual Laboratory», *IEEE Access*, вип. 10, 2022, doi: 10.1109/ACCESS.2022.3181200.

[10] S. Li, B. Jiang, X. Wang, і L. Dong, «Research and Application of a SCADA System for a Microgrid», *Technologies (Basel)*, вип. 5, вип. 2, 2017, doi: 10.3390/technologies5020012.

[11] C. Vargas-Salgado, J. Aguila-Leon, C. Chiñas-Palacios, і E. Hurtado-Perez, «Low-cost web-based Supervisory Control and Data Acquisition system for a microgrid testbed: A case study in design and implementation for academic and research applications», *Heliyon*, вип. 5, вип. 9, 2019, doi: 10.1016/j.heliyon.2019.e02474.

[12] J. F. Polanco *et al.*, «A New Toolkit for Energy Planning for Isolated Microgrids», *IEEE Access*, вип. 11, 2023, doi: 10.1109/ACCESS.2023.3304546.

[13] V. Shendryk, Y. Parfenenko, O. Boiko, S. Shendryk, і Y. Bielka, «Aggregation of multidimensional data for the decision support process for the management of microgrids with renewable energy sources», *Technology audit and production reserves*, вип. 2, вип. 2(64), с. 16–20, Квіт 2022, doi: 10.15587/2706-5448.2022.255957.

[14] Сокрута А. О. і Парфененко Ю. В., «Реалізація інформаційної системи підтримки управління енергетичними мережами з відновлюваними джерелами енергії. Матеріали XIV Всеукраїнської науково-практичної конференції», с. 277–280.

[15] O. Boiko, V. Shendryk, Y. Parfenenko, P. Pavlenko, і A. Titariev, «Information support of stakeholders in the management of energy systems: development and implementation of interfaces», *Eastern-European Journal of Enterprise Technologies*, вип. 6, вип. 8 (126), с. 15–24, Груд 2023, doi: 10.15587/1729-4061.2023.292186.

[16] V. Shendryk, Y. Parfenenko, Y. Kholiavka, P. Pavlenko, O. Shendryk, і L. Bratushka, «Short-term Solar Power Generation Forecasting for Microgrid», в *2022 IEEE 3rd International Conference on System Analysis & Intelligent Computing (SAIC)*, IEEE, Жов 2022, с. 1–5. doi: 10.1109/SAIC57818.2022.9922982.

[17] «EcoStruxure™ Power Monitoring Expert | Schneider Electric Україна». Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://www.se.com/ua/uk/product-range/65404-ecostruxure-power-monitoring-expert/?parent-subcategory-id=4170#overview

[18] «Home | OpenEnergyMonitor». Дата звернення: 09, Листопад 2024. [Online]. Доступний у: https://openenergymonitor.org/

[19] «Product - Sense». Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://sense.com/homes/product/

[20] «Powerwall | Tesla». Дата звернення: 09, Листопад 2024. [Online]. Доступний у: https://www.tesla.com/powerwall

[21] «Building Energy Optimization - GridPoint». Дата звернення: 09, Листопад 2024. [Online]. Доступний у: https://www.gridpoint.com/platform/

[22] «Harness the sun to make, use, save, and sell your own power. | Enphase». Дата звернення: 09, Листопад 2024. [Online]. Доступний у: https://enphase.com/

[23] «ABB Ability Energy Manager - Smart Power Digital Solutions». Дата звернення: 09, Листопад 2024. [Online]. Доступний у: https://new.abb.com/low-voltage/products/smart-power-digital-solutions/abb-ability-energy-manager

[24] «SolarEdge Software Tools - Enhance Your Solar Energy | SolarEdge». Дата звернення: 09, Листопад 2024. [Online]. Доступний у: https://www.solaredge.com/en/products/software-tools

[25] «HOMER - Hybrid Renewable and Distributed Generation System Design Software». Дата звернення: 09, Листопад 2024. [Online]. Доступний у: https://homerenergy.com/

[26] «RETScreen». Дата звернення: 09, Листопад 2024. [Online]. Доступний у: https://natural-resources.canada.ca/maps-tools-and-publications/tools/modelling-tools/retscreen/7465

[27] Sergey Melashich, «Software re-engineering and re-engineering process | Agilie». Дата звернення: 10, Листопад 2024. [Online]. Доступний у: https://agilie.com/blog/what-is-software-reengineering

[28] C. Abid, V. Alizadeh, M. Kessentini, T. do N. Ferreira, і D. Dig, «30 Years of Software Refactoring Research:A Systematic Literature Review», Лип 2020.

[29] «MVC - MDN Web Docs Glossary: Definitions of Web-related terms | MDN». Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://developer.mozilla.org/en-US/docs/Glossary/MVC

[30] «The Model-View-Presenter (MVP) Pattern | Microsoft Learn». Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff649571(v=pandp.10)?redirectedfrom=MSDN

[31] «Patterns - WPF Apps With The Model-View-ViewModel Design Pattern | Microsoft Learn». Дата звернення: 10, Листопад 2024. [Online]. Доступний у: https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-wpf-apps-with-the-model-view-viewmodel-design-pattern

[32] «The MVVM Pattern | Microsoft Learn». Дата звернення: 10, Листопад 2024. [Online]. Доступний у: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/hh848246(v=pandp.10)?redirectedfrom=MSDN

[33] L. A. T. Nguyen, T. S. Huynh, D. T. Tran, і Q. H. Vu, «Design and Implementation of Web Application Based on MVC Laravel Architecture», *European Journal of Electrical Engineering and Computer Science*, вип. 6, вип. 4, 2022, doi: 10.24018/ejece.2022.6.4.448.

[34] R. N. Thakur і U. S. Pandey, «The Role of Model-View Controller in Object Oriented Software Development», *Nepal Journal of Multidisciplinary Research*, вип. 2, вип. 2, 2019, doi: 10.3126/njmr.v2i2.26279.

[35] V. Božić, «Microservices Architecture», Бер 2023, doi: 10.13140/RG.2.2.21902.84802.

[36] G. Blinowski, A. Ojdowska, і A. Przybylek, «Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation», *IEEE Access*, вип. 10, с. 20357–20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[37] S. Partelow, «What is a framework? Understanding their purpose, value, development and use», *J Environ Stud Sci*, вип. 13, вип. 3, с. 510–519, Вер 2023, doi: 10.1007/s13412-023-00833-w.

[38] «GitHub - laravel/laravel: Laravel is a web application framework with expressive, elegant syntax. We’ve already laid the foundation for your next big idea — freeing you to create without sweating the small things.» Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://github.com/laravel/laravel

[39] «GitHub - symfony/symfony: The Symfony PHP framework». Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://github.com/symfony/symfony

[40] «GitHub - bcit-ci/CodeIgniter: Open Source PHP Framework (originally from EllisLab)». Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://github.com/bcit-ci/CodeIgniter

[41] «GitHub - yiisoft/yii2: Yii 2: The Fast, Secure and Professional PHP Framework». Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://github.com/yiisoft/yii2

[42] «GitHub - cakephp/cakephp: CakePHP: The Rapid Development Framework for PHP - Official Repository». Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://github.com/cakephp/cakephp

[43] «Google Тренди». Дата звернення: 17, Жовтень 2024. [Online]. Доступний у: https://trends.google.com/trends/

[44] «IDEF». Дата звернення: 01, Листопад 2024. [Online]. Доступний у: https://www.maxzosim.com/idef/

[45] «Use Case Diagram Tutorial (Guide with Examples) | Creately». Дата звернення: 15, Листопад 2024. [Online]. Доступний у: https://creately.com/guides/use-case-diagram-tutorial/

[46] Марголін Олександр, «Діаграми UML для моделювання процесів і архітектури проекту». Дата звернення: 02, Листопад 2024. [Online]. Доступний у: https://evergreens.com.ua/ua/articles/uml-diagrams.html

# ДОДАТОК А

# ПЛАНУВАННЯ РОБІТ

**SMART – метод деталізації мети проекту.** SMART – метод постановки цілей. Фреймворк SMART дозволяє на етапі цілепокладання сформулювати конкретну і вимірну мету, визначити терміни та необхідні ресурси для її досягнення.

SMART-мета має бути:

– Specific – конкретною;

– Measurable – вимірною;

– Achievable – досяжною;

– Relevant – значимою;

– Time bound – обмеженою в часі.

Результати деталізації мети методом SMART наведено в таблиці А.1.

Таблиця А.1 – Деталізація мети методом SMART

| | |
|---|---|
| Specific (конкретна) | Реінжиніринг інформаційної системи підтримки управління енергетичними мікромережами з ВДЕ. |
| Measurable (вимірювана) | Проведено аналіз та обрано конкретні програмні рішення для використання у роботі (патерн, фреймворк, мова програмування). |
| Achievable (досяжна) | Мета досяжна, оскільки наявні необхідні знання та досвід у застосуванні ООП і архітектурного шаблону MVC, а також потрібні програмні засоби для реалізації. |
| Relevant (значима) | Реінжиніринг інформаційної системи допоможе зробити її більш ефективною, безпечною та пришвидшить процес її розвитку. |

Продовження таблиці А.1.

| | |
|---|---|
| Time-framed (обмежена у часі) | Проєкт має бути завершений відповідно графіку до попереднього захисту дипломних проектів. |

**Планування змісту структури робіт ІТ-проекту (WBS).** Структура декомпозиції робіт (work breakdown structure, WBS) – це інструмент у проєктному менеджменті, який допомагає розподілити проєкт на окремі частини (завдання) для полегшення управління ними та визначення їхньої послідовності. WBS дозволяє зрозуміти структуру проєкту та обсяг робіт, які потрібно виконати для досягнення мети. Завдання, виділені через WBS, розподіляються між членами команди, що сприяє ефективній організації командної роботи. WBS слугує основою для розробки календарного графіка проєкту. На рисунку А.1 наведено WBS діаграму даного проєкту.

**Організаційна структура проекту (OBS).** OBS (Organizational Breakdown Structure) – це інструмент управління проектом, що відображає ієрархічну структуру організації, в якій виконується проєкт. OBS використовується для розподілу робіт, визначених за допомогою WBS, серед членів команди. Вона має таку ж структуру, як WBS, але замість назв робіт містить імена осіб, відповідальних за виконання. OBS сприяє чіткому розподілу ролей і відповідальності, а також покращенню комунікації та координації між учасниками проєкту або підрозділами організації. На рисунку А.2 представлена OBS-діаграма цього проєкту.

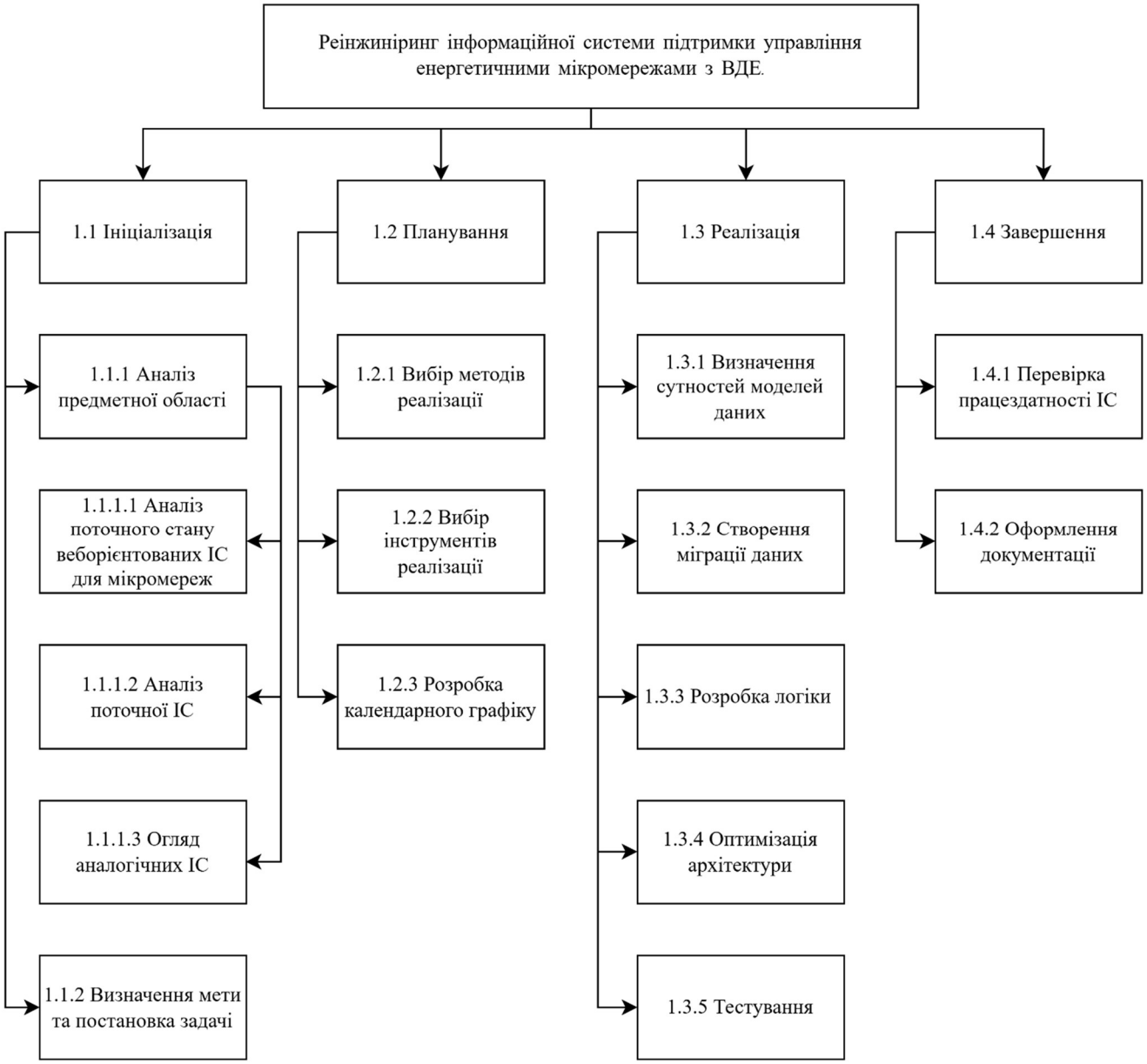


Рисунок А.1 – WBS-структура проєкту

*Джерело*: створено автором

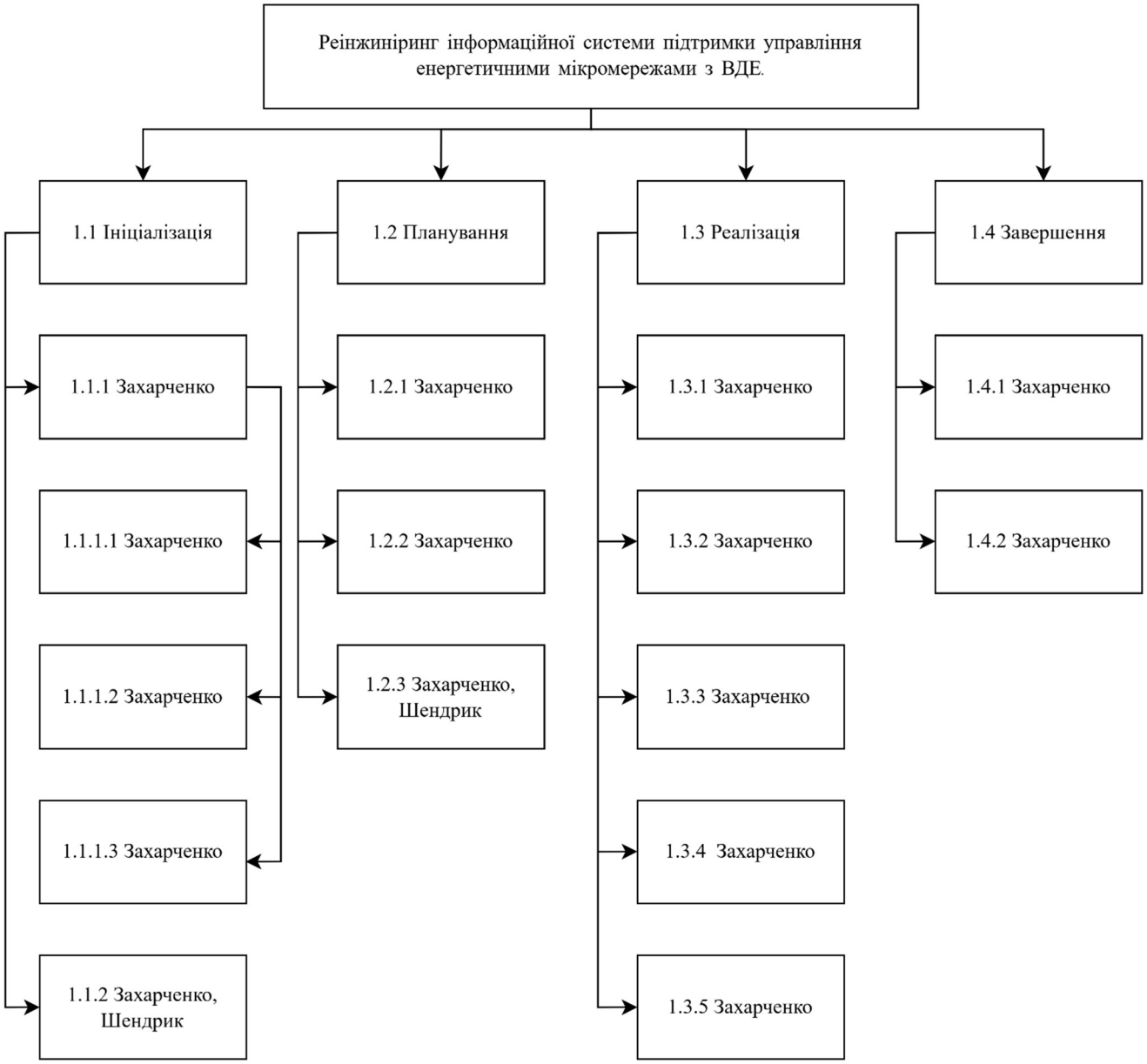


Рисунок А.2 – OBS-структура проєкту

*Джерело*: створено автором

**Створення календарного графіка для виконання ІТ-проекту.** Календарний графік визначає терміни виконання робіт, враховуючи наявні можливості, ресурси та обмеження проєкту, і формується на основі робіт, виділених через WBS. Для його створення та візуалізації застосовується діаграма Ганта, яка складається з горизонтальних смуг, кожна з яких відображає тривалість певної роботи проєкту.

Діаграма Ганта дозволяє відслідковувати послідовність робіт та загальний стан проєкту. Діаграма Ганта, створена в сервісі ClickUp Gantt, наведена на рисунку А.3.

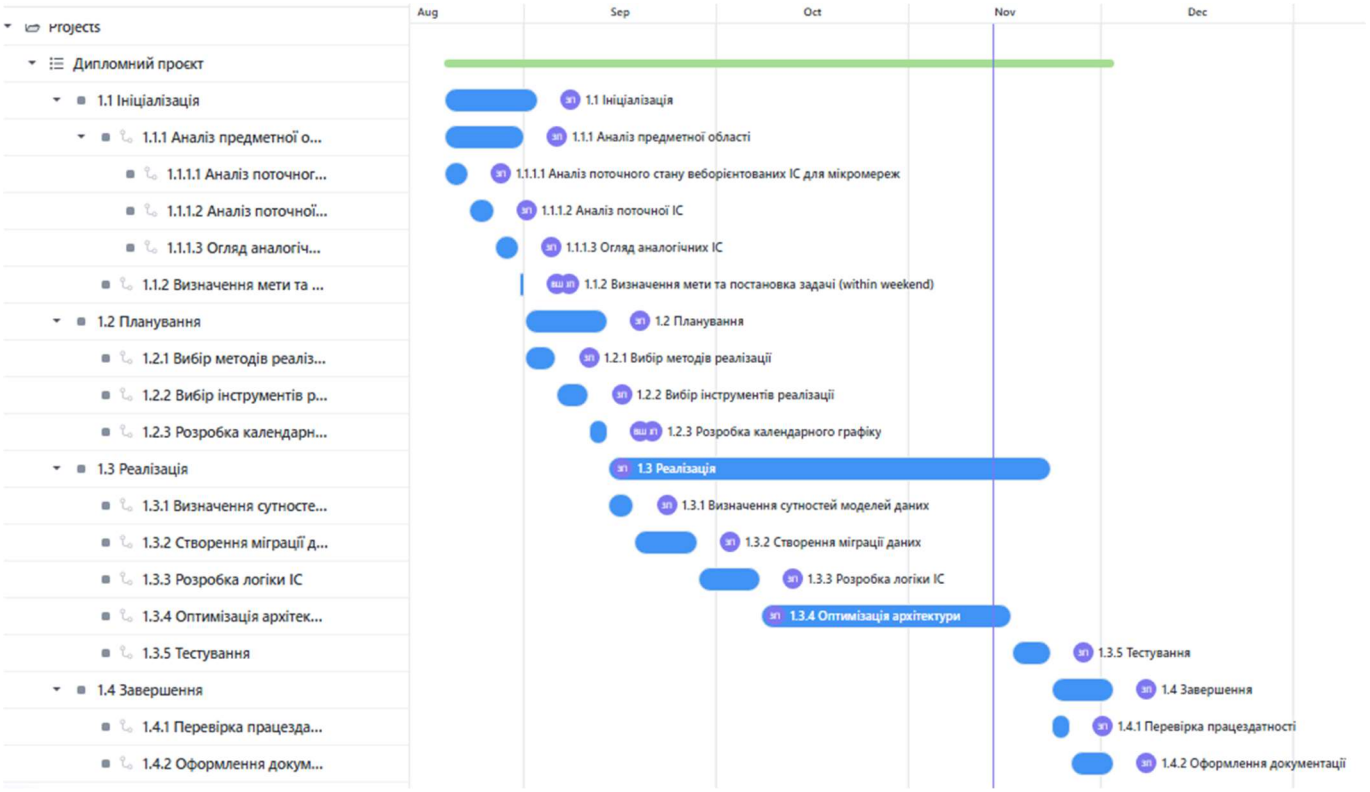


Рисунок А.3 – Діаграма Ганта проєкту

*Джерело*: створено автором

**Управління ризиками.** До основних ризиків, які можуть виникнути в процесі реінжинірингу інформаційної системи підтримки управління енергетичними мікромережами з ВДЕ є:

- недостатнє розуміння поточної архітектури;
- складність інтеграції з наявними компонентами;
- збільшення навантаження під час реалізації проєкту;
- зміна цілей у ході реалізації проєкту;
- людський фактор;
- перебої з електропостачанням;
- зміна строків виконання роботи;
- зростання вимог до проєкту.

Таблиця А.2. Ймовірність виникнення і величина ризику

| № | Ризики | Виникнення | Втрати |
|---|---|---|---|
| 1 | Недостатнє розуміння поточної архітектури | 3 | 5 |
| 2 | Складність інтеграції з наявними компонентами | 3 | 3 |
| 3 | Збільшення навантаження під час реалізації проєкту | 2 | 2 |
| 4 | Зміна цілей у ході реалізації проєкту | 2 | 2 |
| 5 | Людський фактор | 3 | 3 |
| 6 | Перебої з електропостачанням | 4 | 4 |
| 7 | Зміна строків виконання роботи | 2 | 4 |
| 8 | Зростання вимог до проєкту | 3 | 3 |

Матриця «Ймовірність – Втрати»

| Ймовірність | | | | | | | |
|---|---|---|---|---|---|---|---|
| - | | **5** | 5 | 10 | 15 | 20 | 25 |
| Перебої з електропостачанням | | **4** | 4 | 8 | 12 | 16 | 20 |
| Недостатнє розуміння поточної архітектури | | **3** | 3 | 6 | 9 | 12 | 15 |
| Складність інтеграції з наявними компонентами | | | | | | | |
| Людський фактор | | | | | | | |
| Зростання вимог до проєкту | | | | | | | |
| Збільшення навантаження під час реалізації проєкту | | **2** | 2 | 4 | 6 | 8 | 10 |
| Зміна цілей у ході реалізації проєкту | | | | | | | |
| Зміна строків виконання роботи | | | | | | | |
| - | | **1** | 1 | 2 | 3 | 4 | 5 |
| | | | **1** | **2** | **3** | **4** | **5** |
| | | - | Збільшення навантаження під час реалізації проєкту | Зміна цілей у ході реалізації проєкту | Складність інтеграції з наявними компонентами | Людський фактор | Зростання вимог до проєкту | Перебої з електропостачанням | Зміна строків виконання роботи | Недостатнє розуміння поточної архітектури |

Втрати

Аналізуючи ризики за ймовірністю їх виникнення, можемо їх розділити на:

- Ігноровані
    - Відсутні.
- Незначні
    - Збільшення навантаження під час реалізації проєкту.
    - Зміна цілей у ході реалізації проєкту.

- Зміна строків виконання роботи.

- Помірні
    - Недостатнє розуміння поточної архітектури.
    - Складність інтеграції з наявними компонентами.
    - Людський фактор.
    - Зростання вимог до проєкту.
- Істотні
    - Перебої з електропостачанням.
- Критичні
    - Відсутні.

Класифікація ризиків за рівнем впливу:

- Прийнятні
    - Відсутні.
- Виправдані
    - Збільшення навантаження під час реалізації проєкту.
    - Зміна цілей у ході реалізації проєкту.
    - Складність інтеграції з наявними компонентами.
    - Людський фактор.
    - Зростання вимог до проєкту.
    - Перебої з електропостачанням.
    - Зміна строків виконання роботи.
- Неприпустимі
    - Недостатнє розуміння поточної архітектури.

# ДОДАТОК Б

## Лістинг програми

Даний додаток містить лістинги усіх програмних модулів вебзастосунку.

Файл Weather.php

```
<?php

namespace App\Utils;

use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Http;

class Weather
{
    private float $longitude;
    private float $latitude;
    private string $api_key;
    private $weatherData;

    public function __construct($longitude, $latitude)
    {
        $this->longitude = $longitude;
        $this->latitude = $latitude;
        $this->api_key = config('app.wather_api_key');
        $this->getData();
    }

    private function getData()
    {
        $response = Http::get('http://api.weatherapi.com/v1/forecast.json', [
            'key' => $this->api_key,
            'q' => "$this->latitude,$this->longitude",
            'days' => '8',
            'lang' => 'uk',
            'aqi' => 'no',
            'alerts' => 'no'
        ]);

        if ($response->successful()) {
            $this->weatherData = $response->json();
        } else {
            Log::error('Помилка при отриманні даних про погоду.', ['erororr' => $response]);
        }
    }

    public function getDataByDay($day)
    {
```

```
        $response = Http::get('http://api.weatherapi.com/v1/forecast.json', [
            'key' => $this->api_key,
            'q' => "$this->latitude,$this->longitude",
            'days' => '8',
            'lang' => 'uk',
            'aqi' => 'no',
            'alerts' => 'no',
            'dt' => $day,
        ]);

        if ($response->successful()) {
            return $response->json();
        } else {
            Log::error('Помилка при отриманні даних про погоду.', ['erororr' => $response]);
        }
    }

    public function getDetailWeatherbyDay()
    {
        $result = [];
        foreach ($this->getWeather() as $day) {
            $result[] = $this->getDataByDay($day['date']);
        }
        return $result;
    }

    public function getAllWeather()
    {
        return $this->weatherData;
    }

    public function getWeather()
    {
        return $this->weatherData['forecast']['forecastday'];
    }
}
```

Файл CSVHelper.php

```
<?php

namespace App\Utils;

class CSVHelper
{
    /**
     * Функція для запису даних в CSV файл
     *
     * @param  array  $data  Дані для запису
     * @param  string  $filename  Назва файлу
     * @return bool Успішно чи ні
```

```
         */
        public static function dataToCSV(array $data, string $filename): bool
        {
            $f = fopen(storage_path("app/public/csv/{$filename}"), 'w');

            if ($f === false) {
                return false;
            }

            foreach ($data as $row) {
                fputcsv($f, $row);
            }

            fclose($f);

            return true;
        }
    }
```

Файл web.php

```
<?php

use App\Http\Controllers\ApplicationController;
use App\Http\Controllers\CustomerController;
use App\Http\Controllers\DeviceAccumulatorController;
use App\Http\Controllers\DeviceSolarBatteryController;
use App\Http\Controllers\DeviceWindmillController;
use App\Http\Controllers\FeedbackController;
use App\Http\Controllers\FileController;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\MicrogridController;
use App\Http\Controllers\MicrogridParameter;
use App\Http\Controllers\PolygonController;
use Illuminate\Support\Facades\Route;

Route::get('/', fn () => view('welcome'))->name('welcome');
Route::get('/about', fn () => view('about'))->name('about');

Route::get('/application', [ApplicationController::class, 'showApplicationForm'])-
>name('application');
Route::post('/send-email', [ApplicationController::class, 'sendEmail'])-
>name('send.email');

Auth::routes();

Route::prefix('dashboard')->middleware(['auth'])->group(function () {
    Route::get('/home', [HomeController::class, 'index'])->name('home');
    Route::post('/set-locatoin', [HomeController::class, 'setLocation'])->name('set.location');
  Route::get('/customers/search',[CustomerController::class, 'search'])-
>name('customers.search');
    Route::resource('customers', CustomerController::class);
```

```
        Route::get('/microgrid-parameters', [MicrogridParameter::class, 'index'])-
>name('microgrid-parameter.index');
        Route::post('/microgrid-parameters', [MicrogridParameter::class, 'upsert'])-
>name('microgrid-parameter.upsert');
        Route::get('/countries', [MicrogridParameter::class, 'countries'])-
>name('countries.index');
        Route::put('/countries', [MicrogridParameter::class, 'countriesUpdate'])-
>name('countries.update');
        Route::name('devices.')->group(function () {
            Route::get('/solar-batteries/search', [DeviceSolarBatteryController::class, 'search'])-
>name('solar-batteries.search');
            Route::get('/windmills/search', [DeviceWindmillController::class, 'search'])-
>name('windmills.search');
            Route::get('/accumulators/search', [DeviceAccumulatorController::class, 'search'])-
>name('accumulators.search');

            Route::resource('solar-batteries', DeviceSolarBatteryController::class);
            Route::resource('windmills', DeviceWindmillController::class);
            Route::resource('accumulators', DeviceAccumulatorController::class);
        });

        Route::post('/microgrid/generateion', [MicrogridController::class, 'generateion'])-
>name('microgrid.generateion');
        Route::match(['get', 'post'], '/microgrid/{location}', [MicrogridController::class, 'show'])-
>name('microgrid.show');
        Route::post('/microgrid/{location}/load', [MicrogridController::class,
'getConsumptionGeneration'])->name('microgrid.load');

        Route::post('/polygon/store', [PolygonController::class, 'store'])->name('polygon.store');
        Route::get('/polygon', [PolygonController::class, 'create'])->name('polygon.create');
        Route::get('/polygon/config', [PolygonController::class, 'config'])-
>name('polygon.config');
        Route::post('/polygon/data', [PolygonController::class, 'getPolygonData'])-
>name('polygon.data');

        Route::post('/send-feedback', [FeedbackController::class, 'send'])-
>name('feedback.email');
    });
    Route::get('/api/get-file-url/{filename}', [FileController::class, 'getFileUrl']);
```

Файл UserPolicy.php

```
<?php

namespace App\Policies;

use App\Models\User;

final readonly class UserPolicy
{
    /**
     * Determine whether the user can view any models.
```

```
     */
    public function viewAny(User $user): bool
    {

        return $user->isAdmin() || $user->isOperator();
    }

    /**
     * Determine whether the user can view the model.
     */
    public function view(User $user, User $model): bool
    {
        return $user->isAdmin() || $user->isOperator();
    }

    /**
     * Determine whether the user can create models.
     */
    public function create(User $user): bool
    {
        return false;
    }

    /**
     * Determine whether the user can update the model.
     */
    public function update(User $user, User $model): bool
    {
        return false;
    }

    /**
     * Determine whether the user can delete the model.
     */
    public function delete(User $user, User $model): bool
    {
        return false;
    }

    /**
     * Determine whether the user can restore the model.
     */
    public function restore(User $user, User $model): bool
    {
        return false;
    }

    /**
     * Determine whether the user can permanently delete the model.
     */
    public function forceDelete(User $user, User $model): bool
```

```
    {
        return false;
    }

    public function search(User $user): bool
    {
        return $user->isAdmin() || $user->isOperator();
    }
}
```

Файл LoginController.php

```
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\Contract;
use Illuminate\Foundation\Auth\AuthenticatesUsers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class LoginController extends Controller
{
    /*
    |--------------------------------------------------------------------------
    | Login Controller
    |--------------------------------------------------------------------------
    |
    | This controller handles authenticating users for the application and
    | redirecting them to your home screen. The controller uses a trait
    | to conveniently provide its functionality to your applications.
    |
    */

    use AuthenticatesUsers;

    /**
     * Where to redirect users after login.
     *
     * @var string
     */
    protected $redirectTo = '/dashboard/home';

    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('guest')->except('logout');
```

```
        $this->middleware('auth')->only('logout');
    }

    public function username(): string
    {
        return 'username';
    }

    protected function authenticated(Request $request, $user)
    {
        if ($user->isCustomer()) {
            if ($user->contracts->isEmpty()) {
                Auth::logout();

                return redirect()->back()->with('contract-errors', __('auth.no contracts'));
            }

            $hasActiveContract = $user->contracts->contains('status', Contract::ACTIVE);

            if (! $hasActiveContract) {
                Auth::logout();

                return redirect()->back()->with('contract-errors', __('auth.not active contracts'));
            }
        }

        return redirect()->intended($this->redirectPath());
    }
}
```

Файл ApplicationController.php

```
<?php

namespace App\Http\Controllers;

use App\Http\Requests\SendEmailRequest;
use App\Mail\SignUpMessage;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;

class ApplicationController extends Controller
{

    public function showApplicationForm()
    {
        return view('auth.register');
    }

    public function sendEmail(SendEmailRequest $request)
    {
        $validated = $request->validated();
```

```
            Mail::to('admin@gmail.com')->send(new SignUpMessage($validated));

            return redirect()->back()->with('success', __('messages.Application has been sent'));
        }
    }
```

Файл CustomerController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Role;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Gate;

class CustomerController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        Gate::authorize('view', Auth::user());

        $customers = User::with([
            'contracts.location.city.country',
            'contracts.location.locationType'
             ])->select('id', 'name', 'surname', 'username', 'email', 'phone')->where('role_id',
Role::CUSTOMER)->get();

        return view('dashboard.customer.index', compact('customers'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        //
    }
```

```
    /**
     * Display the specified resource.
     */
    public function show(User $user)
    {
        //
    }

    /**
     * Show the form for editing the specified resource.
     */
    public function edit(User $user)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, User $user)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(User $user)
    {
        //
    }

    public function search(Request $request)
    {
        Gate::authorize('search', Auth::user());

        $validated = $request->validate([
            'search' => 'nullable|string|max:255',
        ]);

        $customers = User::with([
            'contracts.location.city.country',
            'contracts.location.locationType'
             ])->select('id',  'name',  'surname',  'username',  'email',  'phone')->where('role_id',
Role::CUSTOMER)
            ->whereRaw("CONCAT(surname, ' ', name) like '%" . $validated['search'] . "%' ")
            ->get();

        return view('dashboard.customer.index', compact('customers'));
    }
```

```
    }
```

Файл DeviceAccumulatorController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Device;
use App\Models\DeviceType;
use Illuminate\Http\Request;
use App\Models\DeviceParameter;

class DeviceAccumulatorController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $devices = Device::query()
            ->accumulator()
            ->paginate();

        $devices = $this->getData($devices);

        return view('dashboard.device.accumulator.index', compact('devices'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        //
    }

    /**
     * Display the specified resource.
     */
    public function show(string $id)
    {
        //
    }
```

```
/**
 * Show the form for editing the specified resource.
 */
public function edit(string $id)
{
    //
}

/**
 * Update the specified resource in storage.
 */
public function update(Request $request, string $id)
{
    //
}

/**
 * Remove the specified resource from storage.
 */
public function destroy(string $id)
{
    //
}

public function search(Request $request)
{
    $validated = $request->validate([
        'search' => 'nullable|string|max:255',
    ]);

    $serch = $validated['search'] ?? false;

    $devices = Device::query()
        ->when($serch, function ($q, string $serch) {
            $q->where('name', 'like', "%{$serch}%");
        })
        ->accumulator()
        ->paginate();

    $devices = $this->getData($devices);

    return view('dashboard.device.accumulator.index', compact('devices'));
}

private function getData($devices)
{
    $deviceIds = $devices->pluck('id');

    $parameters = DeviceParameter::query()
        ->whereIn('device_id', $deviceIds)
        ->join('parameters', 'device_parameters.parameter_id', '=', 'parameters.id')
```

```
            ->select([
                'device_parameters.id as parameter_id',
                'device_parameters.device_id',
                'device_parameters.value',
                'parameters.parameter as parameter_name',
            ])
            ->get()
            ->groupBy('device_id');

        $devices->getCollection()->transform(function ($device) use ($parameters) {
            return [
                'id' => $device->id,
                'name' => $device->name,
                'power' => $device->power,
                'voltage' => $device->voltage,
                'price' => $device->price,
                        'parameters'  =>  $parameters->get($device->id,  collect())->map(function
($parameter) {
                    return [
                        'parameter_id' => $parameter['parameter_id'],
                        'value' => $parameter['value'],
                        'parameter' => $parameter['parameter_name'],
                    ];
                })->toArray(),
            ];
        });

        return $devices;
    }
}
```

Файл DeviceSolarBatteryController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Device;
use App\Models\DeviceParameter;
use App\Models\DeviceType;
use Illuminate\Http\Request;

class DeviceSolarBatteryController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $devices = Device::query()
            ->solarBattey()
            ->paginate();
```

```
        $devices = $this->getData($devices);

        return view('dashboard.device.solar-battery.index', compact('devices'));
    }

    public function search(Request $request)
    {
        $validated = $request->validate([
            'search' => 'nullable|string|max:255',
        ]);

        $serch = $validated['search'] ?? false;

        $devices = Device::query()
            ->when($serch, function ($q, string $serch) {
                $q->where('name', 'like', "%{$serch}%");
            })
            ->solarBattey()
            ->paginate();

        $devices = $this->getData($devices);

        return view('dashboard.device.solar-battery.index', compact('devices'));
    }

    private function getData($devices)
    {
        $deviceIds = $devices->pluck('id');

        $parameters = DeviceParameter::query()
            ->whereIn('device_id', $deviceIds)
            ->join('parameters', 'device_parameters.parameter_id', '=', 'parameters.id')
            ->select([
                'device_parameters.id as parameter_id',
                'device_parameters.device_id',
                'device_parameters.value',
                'parameters.parameter as parameter_name',
            ])
            ->get()
            ->groupBy('device_id');

        $devices->getCollection()->transform(function ($device) use ($parameters) {
            return [
                'id' => $device->id,
                'name' => $device->name,
                'power' => $device->power,
                'voltage' => $device->voltage,
                'price' => $device->price,
                        'parameters'  =>  $parameters->get($device->id,  collect())->map(function
($parameter) {
```

```
                return [
                    'parameter_id' => $parameter['parameter_id'],
                    'value' => $parameter['value'],
                    'parameter' => $parameter['parameter_name'],
                ];
            })->toArray(),
        ];
    });

    return $devices;
  }
}
```

Файл FeedbackController.php

```
<?php

namespace App\Http\Controllers;

use App\Mail\FeedbeackEmail;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;

class FeedbackController extends Controller
{
  public function send(Request $request)
  {
    $data = $request->all();

    Mail::to($request->user())->send(new FeedbeackEmail($data));

    return redirect()->back()->with('success', __('messages.Email has been send'));
  }
}

<?php

namespace App\Http\Controllers;

use App\Models\Location;
use App\Utils\CSVHelper;
use App\Utils\Weather;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;

class MicrogridController extends Controller
{
  public function show(Request $request, Location $location)
  {
    $request->session()->put('location_id', $location->id);

    $user = null;
```

```
            $customer_id = $request->input('customer_id');

            if ($customer_id) {
                $user['id'] = $customer_id;
            } else {
                $user = Auth::user();
            }

            $locations = Location::forUser($user['id'])->get();
            $location->load([
                'locationType:id,title',
                'city:id,name,country_id,longitude,latitude',
                'city.country:id,name',
            ]);
            $components = $location->getGeneralDeviceConfig($location->id);

            $weatherData = new Weather($location->city->longitude, $location->city->latitude);
            $weather = $weatherData->getWeather();

            return view('dashboard.microgrid.show', compact('location', 'locations', 'components',
'weather'));
        }

        public function getConsumptionGeneration(Request $request, Location $location)
        {
            $startDatetime = '2023.12.01 '.date('H:i:s'); //'10:00:00';  //date("Y.m.d H:i:s"); //поточна
дата й час
            $interval_actual = 24 * 7; //пошук на 7 днів назад //інтервал часу в ГОДИНАХ для
пошуку даних в діапазоні [$start_datetime - $interval_actual ; $start_datetime]
            $interval_forecast = 3; //інтервал часу в ГОДИНАХ для пошуку даних в діапазоні
[$start_datetime; $start_datetime + $interval_forecast]
            $data = $location->get_all_components_data($location->id, $startDatetime,
$interval_actual, $interval_forecast);

            return response()->json($data);
        }

        public function generateion(Request $request)
        {
            $location_id = session()->get('location_id');
            $current_datetime = '2023-12-01 00:00:00';
            $data_1d = $this->getActualConsumptionGenerationData($location_id,
$current_datetime, 1);

            if (count($data_1d) == 1) {
                return response()->json('No data');
            } else {
                //прогнозоване споживання на 1 день
                //початкова дата й час
                CSVHelper::dataToCSV($data_1d, 'cons-gen-1_d.csv');
```

```
            //прогнозоване споживання на 3 дні
            $data_3d = $this->getActualConsumptionGenerationData($location_id, $current_datetime, 3);
            CSVHelper::dataToCSV($data_3d, 'cons-gen-3_d.csv');

            //прогнозоване споживання на 7 днів
            $data_7d = $this->getActualConsumptionGenerationData($location_id, $current_datetime, 7);
            CSVHelper::dataToCSV($data_7d, 'cons-gen-7_d.csv');
        }

        return response()->json('ok');
    }

    public function getActualConsumptionGenerationData($locationId, $startDateStr, $intervalInDays)
    {

        $endDateStr = \Carbon\Carbon::parse($startDateStr)->addDays($intervalInDays)->format('Y-m-d');

        $result = DB::table('locations as loc')
            ->join('location_data as locd', 'loc.id', '=', 'locd.location_id')
            ->join('location_devices as ld', 'loc.id', '=', 'ld.location_id')
            ->join('devices as dev', 'ld.device_id', '=', 'dev.id')
            ->join('actual_data as actd', 'actd.device_id', '=', 'ld.id')
            ->join('actual_weather as aw', 'aw.id', '=', 'actd.weather_id')
            ->join('date_time_infos as dti', 'aw.date_time_id', '=', 'dti.id')
            ->where('loc.id', '=', $locationId)
            ->whereBetween('dti.full_date', [$startDateStr, $endDateStr])
            ->whereColumn('locd.date_time_id', '=', 'dti.id')
            ->groupBy('dti.full_date', 'locd.consumption')
            ->orderBy('dti.full_date', 'asc')
            ->select('dti.full_date as date_', 'locd.consumption', DB::raw('SUM(actd.generation) as generation'))
            ->get();

        if ($result->isEmpty()) {
            session()->flash('errMessage', 'Помилка при завантаженні даних про поточне споживання й генерацію енергії');

            return false;
        }

        $data = [['Date', 'Consumption', 'Generation']];

        foreach ($result as $row) {
            $data[] = [
                'date' => $row->date_,
                'consumption' => $row->consumption,
                'generation' => $row->generation,
```

```
                ];
            }

            return $data;
        }
    }
```

Файл MicrogridController.php

```
<?php

namespace App\Http\Controllers;

use App\Models\Location;
use App\Utils\CSVHelper;
use App\Utils\Weather;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;

class MicrogridController extends Controller
{
    public function show(Request $request, Location $location)
    {
        $request->session()->put('location_id', $location->id);

        $user = null;
        $customer_id = $request->input('customer_id');

        if ($customer_id) {
            $user['id'] = $customer_id;
        } else {
            $user = Auth::user();
        }

        $locations = Location::forUser($user['id'])->get();
        $location->load([
            'locationType:id,title',
            'city:id,name,country_id,longitude,latitude',
            'city.country:id,name',
        ]);
        $components = $location->getGeneralDeviceConfig($location->id);

        $weatherData = new Weather($location->city->longitude, $location->city->latitude);
        $weather = $weatherData->getWeather();

        return view('dashboard.microgrid.show', compact('location', 'locations', 'components',
'weather'));
    }

    public function getConsumptionGeneration(Request $request, Location $location)
    {
```

```
            $startDatetime = '2023.12.01 '.date('H:i:s'); //'10:00:00';  //date("Y.m.d H:i:s"); //поточна дата й час
            $interval_actual = 24 * 7; //пошук на 7 днів назад //інтервал часу в ГОДИНАХ для пошуку даних в діапазоні [$start_datetime - $interval_actual ; $start_datetime]
            $interval_forecast = 3; //інтервал часу в ГОДИНАХ для пошуку даних в діапазоні [$start_datetime; $start_datetime + $interval_forecast]
            $data = $location->get_all_components_data($location->id, $startDatetime, $interval_actual, $interval_forecast);

            return response()->json($data);
        }

        public function generateion(Request $request)
        {
            $location_id = session()->get('location_id');
            $current_datetime = '2023-12-01 00:00:00';
            $data_1d = $this->getActualConsumptionGenerationData($location_id, $current_datetime, 1);

            if (count($data_1d) == 1) {
                return response()->json('No data');
            } else {
                //прогнозоване споживання на 1 день
                //початкова дата й час
                CSVHelper::dataToCSV($data_1d, 'cons-gen-1_d.csv');

                //прогнозоване споживання на 3 дні
                $data_3d = $this->getActualConsumptionGenerationData($location_id, $current_datetime, 3);
                CSVHelper::dataToCSV($data_3d, 'cons-gen-3_d.csv');

                //прогнозоване споживання на 7 днів
                $data_7d = $this->getActualConsumptionGenerationData($location_id, $current_datetime, 7);
                CSVHelper::dataToCSV($data_7d, 'cons-gen-7_d.csv');
            }

            return response()->json('ok');
        }

        public function getActualConsumptionGenerationData($locationId, $startDateStr, $intervalInDays)
        {

            $endDateStr = \Carbon\Carbon::parse($startDateStr)->addDays($intervalInDays)->format('Y-m-d');

            $result = DB::table('locations as loc')
                ->join('location_data as locd', 'loc.id', '=', 'locd.location_id')
                ->join('location_devices as ld', 'loc.id', '=', 'ld.location_id')
                ->join('devices as dev', 'ld.device_id', '=', 'dev.id')
```

```
            ->join('actual_data as actd', 'actd.device_id', '=', 'ld.id')
            ->join('actual_weather as aw', 'aw.id', '=', 'actd.weather_id')
            ->join('date_time_infos as dti', 'aw.date_time_id', '=', 'dti.id')
            ->where('loc.id', '=', $locationId)
            ->whereBetween('dti.full_date', [$startDateStr, $endDateStr])
            ->whereColumn('locd.date_time_id', '=', 'dti.id')
            ->groupBy('dti.full_date', 'locd.consumption')
            ->orderBy('dti.full_date', 'asc')
            ->select('dti.full_date as date_', 'locd.consumption', DB::raw('SUM(actd.generation) as generation'))
            ->get();

        if ($result->isEmpty()) {
            session()->flash('errMessage', 'Помилка при завантаженні даних про поточне споживання й генерацію енергії');

            return false;
        }

        $data = [['Date', 'Consumption', 'Generation']];

        foreach ($result as $row) {
            $data[] = [
                'date' => $row->date_,
                'consumption' => $row->consumption,
                'generation' => $row->generation,
            ];
        }

        return $data;
    }
}
```

Файл. Location.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;

class Location extends Model
{
    use HasFactory;

        protected $fillable = ['id', 'latitude', 'longitude', 'name', 'city_id', 'customer_id', 'location_type_id'];

    public function city()
    {
```

```
        return $this->belongsTo(City::class);
    }

    public function locationType()
    {
        return $this->belongsTo(LocationType::class);
    }

    public function user()
    {
        return $this->belongsTo(User::class, 'customer_id');
    }

    public function scopeForUser($query, $userId)
    {
        return $query
            ->with([
                'locationType:id,title',
                'city:id,name,country_id',
                'city.country:id,name',
            ])
            ->whereHas('user', function ($query) use ($userId) {
                $query->where('customer_id', $userId);
            })
            ->select(['id', 'name', 'location_type_id', 'city_id', 'customer_id'])
            ->orderBy('id', 'asc');
    }

    public function getDeviceSummary($locationId)
    {
        $solarAndWindDevices = DB::table('locations')
            ->join('location_devices', 'location_devices.location_id', '=', 'locations.id')
            ->join('devices', 'devices.id', '=', 'location_devices.device_id')
            ->join('device_types', 'devices.type_id', '=', 'device_types.id')
              ->selectRaw('COUNT(device_types.type_title) AS dev_type_cnt, devices.type_id,
device_types.type_title')
            ->where('locations.id', $locationId)
                                    ->whereIn('devices.type_id',     [DeviceType::SOLAR_BATTERY,
DeviceType::WINDMILL])
            ->groupBy('devices.type_id', 'device_types.type_title');

        $batteryDevices = DB::table('locations')
            ->join('accumulators', 'accumulators.location_id', '=', 'locations.id')
            ->join('devices', 'accumulators.device_id', '=', 'devices.id')
            ->join('device_types', 'devices.type_id', '=', 'device_types.id')
              ->selectRaw('COUNT(device_types.type_title) AS dev_type_cnt, devices.type_id,
device_types.type_title')
            ->where('locations.id', $locationId)
            ->groupBy('devices.type_id', 'device_types.type_title');

        return $solarAndWindDevices
```

```
            ->unionAll($batteryDevices)
            ->orderBy('type_id')
            ->get();
    }

    public function getGeneralDeviceConfig($locationId)
    {
        $data = [
            'solar' => $this->getDeviceDataByType($locationId, 1),
            'wind' => $this->getDeviceDataByType($locationId, 2),
            'accum' => $this->getAccumulatorData($locationId),
        ];

        return $data;
    }

    private function getDeviceDataByType($locationId, $typeId)
    {
        return $this->join('location_devices', 'location_devices.location_id', '=', 'locations.id')
            ->join('devices', 'devices.id', '=', 'location_devices.device_id')
            ->join('device_types', 'devices.type_id', '=', 'device_types.id')
            ->where('locations.id', $locationId)
            ->where('devices.type_id', $typeId)
            ->groupBy('devices.id', 'devices.name')
                    ->selectRaw('devices.id AS dev_id, devices.name AS dev_name,
COUNT(devices.name) AS dev_cnt')
            ->get()
            ->toArray();
    }

    private function getAccumulatorData($locationId)
    {
        return $this->join('accumulators', 'accumulators.id', '=', 'locations.id')
            ->join('devices', 'accumulators.device_id', '=', 'devices.id')
            ->join('device_types', 'devices.type_id', '=', 'device_types.id')
            ->where('locations.id', $locationId)
            ->groupBy('devices.id', 'devices.name')
                    ->selectRaw('devices.id AS dev_id, devices.name AS dev_name,
COUNT(devices.name) AS dev_cnt')
            ->get()
            ->toArray();
    }

    public function getActualConsumptionGenerationData($startDate, $intervalInDays)
    {
        return $this->join('location_data', 'locations.id', '=', 'location_data.location_id')
            ->join('location_devices', 'locations.id', '=', 'location_devices.location_id')
            ->join('devices', 'location_devices.device_id', '=', 'devices.id')
            ->join('actual_data', 'actual_data.device_id', '=', 'location_devices.id')
            ->join('actual_weather', 'actual_weather.id', '=', 'actual_data.weather_id')
            ->join('date_time_infos', 'actual_weather.date_time_id', '=', 'date_time_infos.id')
```

```
            ->where('locations.id', $this->id)
            ->whereBetween('date_time_infos.full_date', [
                $startDate,
                now()->parse($startDate)->addDays($intervalInDays),
            ])
            ->whereColumn('location_data.date_time_id', 'date_time_infos.id')
            ->groupBy('date_time_infos.full_date', 'location_data.consumption')
            ->orderBy('date_time_infos.full_date', 'asc')
            ->selectRaw(
                'date_time_infos.full_date AS date_,
                 location_data.consumption,
                 SUM(actual_data.generation) AS generation'
            )
            ->get()
            ->map(function ($row) {
                return [
                    'date' => $row->date_,
                    'consumption' => $row->consumption,
                    'generation' => $row->generation,
                ];
            })
            ->toArray();
    }

    public function get_all_components_data($location_id, $start_datetime, $interval_actual,
$interval_forecast)
    {
        $data[] = $this->getSolarPanelData($location_id, $start_datetime, $interval_actual,
$interval_forecast);
        $data[] = $this->getWindmillData($location_id, $start_datetime, $interval_actual,
$interval_forecast);
        $data[] = $this->getAccumulatorsData($location_id, $start_datetime, $interval_actual);
        $data[] = $this->get_current_system_state_data($location_id, $start_datetime,
$interval_actual);

        return $data;
    }

    public function getSolarPanelActualGeneration($locationId, $startDatetime, $interval)
    {

        // Знаходимо максимальну дату для фільтрації
        $maxDatetime = DB::table('date_time_infos')
            ->join('actual_weather', 'date_time_infos.id', '=', 'actual_weather.date_time_id')
            ->join('actual_data', 'actual_weather.id', '=', 'actual_data.weather_id')
            ->join('location_devices', 'actual_data.device_id', '=', 'location_devices.id')
            ->join('devices', 'location_devices.device_id', '=', 'devices.id')
            ->join('device_types', 'devices.type_id', '=', 'device_types.id')
            ->where('location_devices.location_id', $locationId)
            ->where('device_types.id', 1)
            ->where('device_types.type_title', 'Сонячна батарея')
```

```
            ->whereBetween('date_time_infos.full_date', [
                DB::raw("DATE_ADD('$startDatetime', INTERVAL -$interval HOUR)"),
                $startDatetime,
            ])
            ->orderByDesc('date_time_infos.full_date')
            ->limit(1)
            ->value('date_time_infos.id');

        if (! $maxDatetime) {
            return [];
        }

        $results = DB::table('location_devices as ld')
            ->join('devices as dev', 'ld.device_id', '=', 'dev.id')
            ->join('device_types as dt', 'dev.type_id', '=', 'dt.id')
            ->join('actual_data as ad', 'ad.device_id', '=', 'ld.id')
            ->join('actual_weather as aw', 'ad.weather_id', '=', 'aw.id')
            ->join('date_time_infos as dti', 'aw.date_time_id', '=', 'dti.id')
            ->where('ld.id', $locationId)
            ->where('dt.id', 1)
            ->where('dt.type_title', 'Сонячна батарея')
            ->where('dti.id', $maxDatetime)
            ->selectRaw('
            SUM(ad.generation) AS actual_solar_generation,
            DATE_FORMAT(dti.full_date, "%e.%m %H:%i") AS time,
            aw.clouds
        ')
            ->groupBy('dti.full_date', 'aw.clouds')
            ->get();

        return $results;
    }

    public function getWindmillActualGeneration($locationId, $startDatetime, $interval)
    {
        $subqueryResult = DB::table('location_devices as ld')
            ->join('devices as dev', 'ld.device_id', '=', 'dev.id')
            ->join('device_types as dt', 'dev.type_id', '=', 'dt.id')
            ->join('actual_data as ad', 'ad.device_id', '=', 'ld.id')
            ->join('actual_weather as aw', 'ad.weather_id', '=', 'aw.id')
            ->join('date_time_infos as dti', 'aw.date_time_id', '=', 'dti.id')
            ->where('ld.location_id', $locationId)
            ->where('dt.id', 2)
            ->where('dt.type_title', 'Вітрова установка')
            ->whereBetween('dti.full_date', [
                DB::raw("DATE_ADD('{$startDatetime}', INTERVAL -{$interval} HOUR)"),
                $startDatetime,
            ])
            ->orderByDesc('dti.full_date')
            ->limit(1)
            ->select('dti.id')
```

```
            ->first();

        if ($subqueryResult) {

            $query = DB::table('location_devices as ld')
                ->join('devices as dev', 'ld.device_id', '=', 'dev.id')
                ->join('device_types as dt', 'dev.type_id', '=', 'dt.id')
                ->join('actual_data as ad', 'ad.device_id', '=', 'ld.id')
                ->join('actual_weather as aw', 'ad.weather_id', '=', 'aw.id')
                ->join('date_time_infos as dti', 'aw.date_time_id', '=', 'dti.id')
                ->where('ld.location_id', $locationId)
                ->where('dt.id', 2)
                ->where('dt.type_title', 'Вітрова установка')
                ->where('dti.id', $subqueryResult->id) // Використовуємо конкретний `id`
                ->selectRaw('SUM(ad.generation) AS actual_wind_generation,
                    DATE_FORMAT(dti.full_date, "%e.%m %H:%i") AS time,
                    aw.wind_speed')
                ->groupBy('dti.full_date', 'aw.wind_speed')
                ->get();

            return $query;
        }

        return null;
    }

    public function getAccumStatus($locationId)
    {
        return Accumulator::where('location_id', $locationId)
            ->select('id as accum_id', 'device_id', 'charge', 'discharge', 'convector')
            ->get();
    }

    public function getMaxCapacityLevel($locationId)
    {
        $maxCapacity = DeviceParameter::join('devices', 'device_parameters.device_id', '=',
'devices.id')
            ->join('accumulators', 'devices.id', '=', 'accumulators.device_id')
            ->join('parameters', 'device_parameters.parameter_id', '=', 'parameters.id')
            ->where('accumulators.location_id', $locationId)
            ->where('parameters.id', 16) // Параметр "Ємність"
            ->where('parameters.parameter', 'Ємність (А-год)')
            ->sum('device_parameters.value');
        return $maxCapacity;
    }

    public function getCurrentSystemState($locationId, $startDatetime, $interval)
    {
        $endDatetime = \Carbon\Carbon::createFromFormat('Y.m.d H:i:s', $startDatetime);
        $startDatetimeInterval = $endDatetime->copy()->subHours($interval);
```

```
                return LocationData::join('date_time_infos', 'location_data.date_time_id', '=',
'date_time_infos.id')
            ->where('location_data.location_id', $locationId)
            ->whereBetween('date_time_infos.full_date', [$startDatetimeInterval, $endDatetime])
            ->orderByDesc('date_time_infos.full_date')
            ->select([
                DB::raw("DATE_FORMAT(date_time_infos.full_date, '%e.%m %H:%i') AS time"),
                'location_data.consumption',
                'location_data.fact_sales',
                'location_data.purchase',
            ])
            ->first();
    }

    public function get_current_system_state_data($location_id, $start_datetime, $interval)
    {
        $data['current_system_state_data'] = $this->getCurrentSystemState($location_id,
$start_datetime, $interval);
        if (! $data) {
            return false;
        }

        return $data;
    }

    public function getSolarPanelMaxPower($locationId)
    {
        $maxPower = LocationDevice::query()
            ->join('devices as dev', 'location_devices.device_id', '=', 'dev.id')
            ->join('device_types as dt', 'dev.type_id', '=', 'dt.id')
            ->where('location_devices.location_id', $locationId)
            ->where('dev.type_id', 1)
            ->where('dt.type_title', 'Сонячна батарея')
            ->sum('dev.power');

        return $maxPower ?: 0;
    }

        public function getSolarPanelData($location_id, $start_datetime, $interval_actual,
$interval_forecast)
    {
        $data['actual_solar_generation_data'] =
                        $this->getSolarPanelActualGeneration($location_id, $start_datetime,
$interval_actual)[0] ?? [];

        if (! $data['actual_solar_generation_data']) {
            return false;
        }

        $data['forecast_solar_generation_data'] =
```

```
                $this->getSolarPanelForecastGeneration($location_id,    $start_datetime,
$interval_forecast)[0] ?? [];

        if (! $data['forecast_solar_generation_data']) {
            return false;
        }

        $data['max_power'] = $this->getSolarPanelMaxPower($location_id);
        if (! $data['max_power']) {
            return false;
        }

        return $data;
    }

    public function getSolarPanelForecastGeneration($locationId, $startDatetime, $interval)
    {
        $subquery = DB::table('location_devices as ld')
            ->join('devices as dev', 'ld.device_id', '=', 'dev.id')
            ->join('device_types as dt', 'dev.type_id', '=', 'dt.id')
            ->join('forecast_data as fd', 'fd.device_id', '=', 'ld.id')
            ->join('forecast_weather as fw', 'fd.weather_id', '=', 'fw.id')
            ->join('date_time_infos as dti', 'fw.date_time_id', '=', 'dti.id')
            ->where('ld.location_id', $locationId)
            ->where('dt.id', 1)
            ->where('dt.type_title', 'Сонячна батарея')
            ->whereBetween('dti.full_date', [
                DB::raw("DATE_ADD('{$startDatetime}', INTERVAL -{$interval} HOUR)"),
                $startDatetime,
            ])
            ->orderByDesc('dti.full_date')
            ->select('dti.id')
            ->limit(1);

        $query = DB::table('location_devices as ld')
            ->join('devices as dev', 'ld.device_id', '=', 'dev.id')
            ->join('device_types as dt', 'dev.type_id', '=', 'dt.id')
            ->join('forecast_data as fd', 'fd.device_id', '=', 'ld.id')
            ->join('forecast_weather as fw', 'fd.weather_id', '=', 'fw.id')
            ->join('date_time_infos as dti', 'fw.date_time_id', '=', 'dti.id')
            ->joinSub($subquery, 'latest_dti', function ($join) {
                $join->on('dti.id', '=', 'latest_dti.id');
            })
            ->where('ld.location_id', $locationId)
            ->where('dt.id', 1)
            ->where('dt.type_title', 'Сонячна батарея')
            ->selectRaw('
            SUM(fd.generation) AS forecast_solar_generation,
            DATE_FORMAT(dti.full_date, "%e.%m %H:%i") AS time,
            fw.clouds
        ')
```

```
            ->groupBy('dti.full_date', 'fw.clouds')
            ->get();

        return $query;
    }

     public function getWindmillData($location_id, $start_datetime, $interval_actual,
$interval_forecast)
    {
        $data['actual_wind_generation_data'] =
            $this->getWindmillActualGeneration($location_id, $start_datetime, $interval_actual,
$interval_forecast)[0] ?? [];
        if (! $data['actual_wind_generation_data']) {
            return false;
        }

        $data['forecast_wind_generation_data'] =
                        $this->getWindmillForecastGeneration($location_id, $start_datetime,
$interval_forecast);

        if (! $data['forecast_wind_generation_data']) {
            return false;
        }

        $data['max_power'] = $this->getWindmillMaxPower($location_id);
        if (! $data['max_power']) {
            return false;
        }

        return $data;
    }

    public function getWindmillForecastGeneration($locationId, $startDatetime, $interval)
    {

        $subquery = DB::table('location_devices as ld')
            ->join('devices as dev', 'ld.device_id', '=', 'dev.id')
            ->join('device_types as dt', 'dev.type_id', '=', 'dt.id')
            ->join('forecast_data as fd', 'fd.device_id', '=', 'ld.id')
            ->join('forecast_weather as fw', 'fd.weather_id', '=', 'fw.id')
            ->join('date_time_infos as dti', 'fw.date_time_id', '=', 'dti.id')
            ->where('ld.location_id', $locationId)
            ->where('dt.id', 2)
            ->where('dt.type_title', 'Вітрова установка')
                                        ->whereBetween('dti.full_date',        [$startDatetime,
DB::raw("DATE_ADD('$startDatetime', INTERVAL $interval HOUR)")])
            ->selectRaw('MAX(dti.id) as max_id');
        Log::info('sub', $subquery->get()->toArray());
        $result = DB::table('location_devices as ld')
            ->join('devices as dev', 'ld.device_id', '=', 'dev.id')
            ->join('device_types as dt', 'dev.type_id', '=', 'dt.id')
```

```
            ->join('forecast_data as fd', 'fd.device_id', '=', 'ld.id')
            ->join('forecast_weather as fw', 'fd.weather_id', '=', 'fw.id')
            ->join('date_time_infos as dti', 'fw.date_time_id', '=', 'dti.id')
            ->where('ld.location_id', $locationId)
            ->where('dt.id', 2)
            ->where('dt.type_title', 'Вітрова установка')
            ->whereIn('dti.id', $subquery)
            ->selectRaw('
            SUM(fd.generation) AS forecast_wind_generation,
            DATE_FORMAT(dti.full_date, "%e.%m %H:%i") AS time,
            fw.wind_speed
        ')
            ->groupBy('dti.full_date', 'fw.wind_speed')
            ->first();

        return $result;
    }

    public function getWindmillMaxPower($locationId)
    {
        $maxPower = DB::table('location_devices as ld')
            ->join('devices as dev', 'ld.device_id', '=', 'dev.id')
            ->join('device_types as dt', 'dev.type_id', '=', 'dt.id')
            ->where('ld.location_id', $locationId)
            ->where('dev.type_id', 2)
            ->where('dt.type_title', 'Вітрова установка')
            ->sum('dev.power');

        return $maxPower;
    }

    public function getAccumulatorsData($location_id, $start_datetime, $interval)
    {
        $data['current_capacity_data'] =
            $this->getCurrentCapacity($location_id, $start_datetime, $interval);

        if (empty($data)) {
            return false;
        }

        $data['status'] = $this->getAccumStatus($location_id);
        if (! $data['status']) {
            return false;
        }

        $data['max_capacity'] = $this->getMaxCapacityLevel($location_id);

        Log::info('sql', $data);
        if (!$data['max_capacity']) {
            return false;
        }
```

```
            return $data;
        }

        public function getCurrentCapacity($locationId, $startDatetime, $interval)
        {
            $endDatetime = \Carbon\Carbon::createFromFormat('Y.m.d H:i:s', $startDatetime);
            $startDatetimeInterval = $endDatetime->copy()->subHours($interval);

            $subquery2 = DB::table('date_time_infos as dti')
                ->join('current_capacities as curcap', 'curcap.date_time_id', '=', 'dti.id')
                ->join('accumulators as a', 'curcap.accumulator_id', '=', 'a.id')
                ->join('devices as dev', 'a.device_id', '=', 'dev.id')
                ->join('device_types as dt', 'dev.type_id', '=', 'dt.id')
                ->join('location_devices as ld', 'ld.device_id', '=', 'dev.id')
                ->join('locations as loc', 'loc.id', '=', 'ld.location_id')
                ->where('loc.id', $locationId)
                ->where('dev.id', DB::raw('a.device_id'))
                ->whereBetween('dti.full_date', [
                    DB::raw("DATE_ADD('$startDatetime', INTERVAL -$interval HOUR)"),
                    $startDatetime
                ])
                ->selectRaw('MAX(dti.id) as max_id');

            $subquery1 = DB::table('locations as loc')
                ->join('accumulators as a', 'a.location_id', '=', 'loc.id')
                ->join('location_devices as ld', 'ld.location_id', '=', 'loc.id')
                ->join('devices as dev', 'dev.id', '=', 'ld.device_id')
                ->join('device_types as dt', 'dt.id', '=', 'dev.type_id')
                ->join('current_capacities as curcap', 'curcap.accumulator_id', '=', 'a.id')
                ->join('date_time_infos as dti', 'curcap.date_time_id', '=', 'dti.id')
                ->where('loc.id', $locationId)
                ->where('dev.id', DB::raw('a.device_id'))
                ->where('dti.id', $subquery2)
                ->select('curcap.capacity as acum_curr_capacity', 'dti.full_date as time');

            $result = DB::table(DB::raw("({$subquery1->toSql()}) as tb"))
                ->mergeBindings($subquery1)
                                ->selectRaw('SUM(tb.acum_curr_capacity)    as    current_capacity,
DATE_FORMAT(tb.time, "%e.%m %H:%i") as time')
                ->groupBy('tb.time')
                ->first();

            return json_decode(json_encode($result), true) ?? null;
        }
    }
```