Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)
Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система моніторингу зарядних станцій електромобілів.
Розробник клієнтської частини
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-10

Горкун Д.О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Русакова Н.Є.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Горкуну Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система моніторингу зарядних станцій електромобілів. Розробник клієнтської частини
Затверджена наказом по університету від 20.05. 2024р. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 17.06.2024
3. Вихідні дані до роботи Розробити клієнтську частину веб-застосунку для управління депо електротранспорту з метою контролю витрат потужності та оптимізації споживання енергії зарядними станціями, використовуючи мову програмування TypeScript та інструменти Angular й Ionic.
4. Перелік питань, що потрібно опрацювати в роботі Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, впровадження програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	17.04.2024	*виконано*
2	Створення специфікації ПЗ	01.05.2024	*виконано*
3	Проектування ПЗ	02.05.2024	*виконано*
4	Розробка ПЗ	05.06.2024	*виконано*
5	Тестування ПЗ	07.06.2024	*виконано*
6	Оформлення пояснювальної записки	10.06.2024	*виконано*
7	Підготовка презентації та доповіді	11.06.2024	*виконано*
8	Попередній захист	12.06.2024	*виконано*
9	Нормоконтроль, рецензування	12.06.2024	*виконано*
10	Здача роботи у електронний архів	14.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	16.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024р.

Студент (ка) _____ Горкун Д.О .
(підпис)

Керівник роботи _____ доц. кафедри ПІ Русакова Н.Є.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 129 стор., 40 рис., 4 табл., 13 джерел.

ВЕБ-СИСТЕМА, ЗАРЯДНІ СТАНЦІЇ, ЕЛЕКТРОМОБІЛІ, МОНІТОРИНГ, СПОЖИВАННЯ ЕЛЕКТРОЕНЕРГІЇ, ANGULAR, OCPP

Об'єктом розробки є програмна система для централізованого управління та моніторингу мережі зарядних станцій для електромобілів.

Мета роботи – створити веб-додаток із зручним інтерфейсом для ефективного управління зарядною інфраструктурою, що забезпечить оптимізацію витрат електроенергії, моніторинг та аналітику процесів зарядки.

Використані методи та інструменти: середовище WebStorm, Angular 17, Ionic 8, Bootstrap 5, Jest, протокол OCPP 1.6.

У результаті роботи реалізовано клієнтську частину веб-системи, що надає власникам та операторам мереж зарядних станцій функціонал для гнучкого налаштування профілів споживання електроенергії, моніторингу стану зарядок і аналізу статистичних даних. Створений продукт відповідає вимогам безпеки ISO/IEC 27001, забезпечує інтеграцію через відкриті API та високу доступність сервісу.

Результати роботи можуть застосовуватись операторами зарядної інфраструктури для електромобілів, муніципалітетами та енергетичними компаніями.

Розроблене програмне рішення дозволяє оптимізувати витрати електроенергії на зарядку електротранспорту, скоротити експлуатаційні витрати та підвищити ефективність використання зарядної інфраструктури.

WEB SYSTEM, CHARGING STATIONS, ELECTRIC VEHICLES, MONITORING, ENERGY CONSUMPTION, ANGULAR, OCPP

The subject of the development is a software system for centralized management and monitoring a network of charging stations for electric vehicles.

The work aims to create a web application with a user-friendly interface for efficient management of the charging infrastructure, which will ensure the optimization of energy consumption, monitoring, and analysis of charging processes.

Methods and tools used: WebStorm environment, Angular 17, Ionic 8, Bootstrap 5, Jest, OCPP 1.6 protocol.

As a result of the work, the client part of the web system has been implemented, providing owners and operators of charging station networks with functionality for flexible configuration of energy consumption profiles, monitoring charging status, and analyzing statistical data. The created product meets the security requirements of ISO/IEC 27001, ensures integration through open APIs, and high service availability.

The results of the work can be used by operators of charging infrastructure for electric vehicles, municipalities, and energy companies.

The developed software solution allows for the optimization of energy costs for charging electric vehicles, reducing operational costs, and increasing the efficiency of using the charging infrastructure.

Я, Горкун Дмитро Олександрович, студент гр. ПЗПІ-20-10, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система мониторингу зарядних станцій електробілей. Розробник клієнтської частини», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АЗС – Автозаправна станція

BPMN – Нотація моделювання бізнес-процесів (Business Process Model and Notation)

ЕТЗ – Електричні транспортні засоби

МЕА – Міжнародне енергетичне агентство

OCPP – Протокол відкритого з'єднання для зарядної інфраструктури (Open Charge Point Protocol)

OCPI – Протокол обміну інформацією для зарядної інфраструктури (Open Charge Point Interface)

ПЗ – Програмне забезпечення

RBAC – Рольовий контроль доступу (Role-Based Access Control)

ТЗ – Технічне завдання

TDD – Розробка через тестування (Test-Driven Development)

UML – Уніфікована мова моделювання (Unified Modeling Language)

CRUD – Створення, читання, оновлення, видалення (Create, Read, Update, Delete)

CSV – Значення, розділені комами (Comma-Separated Values)

HTTPS – Протокол захищеного гіпертекстового перенесення (Hypertext Transfer Protocol Secure)

ISO – Міжнародна організація зі стандартизації (International Organization for Standardization)

JSON – Запис об'єктних нотацій JavaScript (JavaScript Object Notation)

API – Прикладний програмний інтерфейс (Application Programming Interface)

XSS – Міжсайтовий скриптинг (Cross-Site Scripting)

CSRF – Міжсайтова підробка запиту (Cross-Site Request Forgery)

ВСТУП

Темою даної кваліфікаційної роботи є розробка програмної системи моніторингу зарядних станцій електромобілів. Зростаюча популярність електричного транспорту та необхідність ефективного управління зарядною інфраструктурою обумовлюють актуальність створення комплексного рішення для моніторингу та контролю зарядних станцій.

Мета роботи полягає у створенні програмної системи, що складається з повноцінного веб-застосунку, для комплексного управління зарядною інфраструктурою електротранспорту, та мобільного застосунку для водіїв. Ключовими завданнями є забезпечення можливостей віддаленого моніторингу та контролю зарядних станцій, оптимізації процесів зарядки, надання аналітичних даних для прийняття обґрунтованих рішень, а також вдосконалення управління персоналом та мережею депо.

Розроблене програмне забезпечення спрямоване на підвищення ефективності використання електроенергії, зниження експлуатаційних витрат та поліпшення досвіду як водіїв електромобілів, так і операторів зарядних мереж. Завдяки можливості віддаленого моніторингу та контролю зарядних станцій, система дозволить оперативно реагувати на виникаючі проблеми та забезпечить безперебійну роботу зарядної інфраструктури. Оптимізація процесів зарядки сприятиме раціональному використанню електроенергії та зменшенню витрат на експлуатацію. Аналітичні дані, отримані із системи, допоможуть приймати обґрунтовані рішення щодо розширення мережі зарядних станцій та покращення обслуговування клієнтів.

Отже, розробка програмної системи моніторингу зарядних станцій електромобілів є актуальним завданням, вирішення якого сприятиме прискоренню розвитку електротранспорту та зарядної інфраструктури в цілому.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

1.1.1 Статистичні дані

Глобальний ринок електромобілів продовжує стрімко зростати. Згідно з даними МЕА [1], продажі нових електромобілів у 2023 році зросли на більше ніж 100% порівняно з 2020 роком і сягнули 14 мільйонів одиниць, що становить 18% від загального ринку нових автомобілів1. Найбільші ринки – Китай (60% від глобальних продажів), Європа (більше ніж 20% від загальних продажів) та США (8% від загальних продажів)

Експерти прогнозують, що частка електромобілів у загальних продажах нових автомобілів у світі зросте до 35% до 2030 року. До 2030 року очікується, що глобальний парк електромобілів досягне значної кількості.

Паралельно зі зростанням кількості електромобілів, активно розвивається і мережа публічних зарядних станцій. За даними МЕА, станом на кінець 2021 року у світі налічувалося близько 16 мільйонів публічних зарядних точок. Очікується, що до 2030 року їх кількість може зрости щонайменше у 10 разів – до 160-200 мільйонів одиниць.

1.1.2 Ключові учасники та їхні потреби

Система управління зарядними станціями для ЕТЗ передбачає взаємодію трьох основних категорій учасників:

– власники/оператори зарядних станцій. Вони зацікавлені в отриманні оперативної інформації про стан і завантаженість своїх станцій, можливості віддаленого управління, аналітиці для оптимізації роботи;
– власники/користувачі ЕТЗ. Для них важливо мати актуальну інформацію про доступність, режими роботи та вартість зарядки на конкретних станціях, можливість планування маршрутів з урахуванням необхідності підзарядки;
– державні/муніципальні органи влади. Вони зацікавлені в отриманні агрегованих даних про використання зарядної інфраструктури для

ефективного планування її розвитку.

1.1.3 Існуючі рішення та технології

На ринку програмного забезпечення для моніторингу та управління зарядними станціями електромобілів представлено кілька відомих рішень (див. табл. 1.1).

Таблиця 1.1 – Порівняння існуючих рішень (таблиця виконана самостійно)

Назва	Опис	Ключові можливості	Цільова аудиторія	Додатково
GreenFlux [2]	Голландська компанія, що пропонує хмарну платформу для управління зарядною інфраструктурою	Збір даних з мережі, контроль доступу та оплати, аналітика, інтеграція з зовнішніми системами	Оператори мереж зарядних станцій, муніципалітети, енергокомпанії	Стандарти OCPP та OCPI для інтеграції обладнання різних виробників
Ampecp [3]	Універсальне рішення для заряджання базується в Америці.	Моніторинг стану станцій, віддалене управління, автоматизація тарифікації, аналітика	Оператори мереж зарядних станції, комерційні паркінги, мережі АЗС	Пропонує гнучку масштабування та інтеграцію з платіжними системами
Driivz [4]	Ізраїльська компанія, що надає хмарну платформу для управління зарядною інфраструктурою електромобілів	Моніторинг, централізоване управління станціями, тарифізкація	Оператори мереж зарядних станцій, електромережеві компанії, прокат електроавто	OCPP 1.6 і 2.0.1, звітування та аналітика

Кінець таблиці 1.1

ChargeLab [5]	Канадський стартап з аналогічним конкурентам набором функцій	Доступ та оплата, моніторинг, аналітика	Власники мереж зарядних станцій, АЗС та муніципалітети	Відкриті стандарти та API для інтеграції

Бачимо, що на ринку вже представлено низку програмних рішень для моніторингу та управління зарядними станціями ЕТЗ. Вони, як правило, включають функції збору даних про стан та режими роботи станцій, віддаленого управління, надання інформації про доступність і бронювання, а також аналітику та звітність.

Технологічно такі системи зазвичай реалізуються на основі відкритих протоколів OCPP [6], які передають дані до централізованої серверної інфраструктури. Платформа, у свою чергу, надає доступ до даних через веб-інтерфейс або мобільні додатки.

Незважаючи на наявність подібних рішень, ринок все ще знаходиться на етапі становлення. Існують певні проблеми та обмеження, такі як недостатня інтегрованість різних систем, складність масштабування, обмежені можливості аналітики та прогнозування, відсутність уніфікованих стандартів обміну даними. Тому існує потреба у розробці більш гнучких, функціонально розширених систем управління зарядною інфраструктурою ЕТЗ.

1.1.4 Виявлення та вирішення проблем

Відповідно до наданого контексту проекту, основні проблеми, які необхідно вирішити при розробці системи управління зарядними станціями для ЕТЗ, можна сформулювати таким чином:

- забезпечення збору, обробки та збереження даних про стан і режими роботи зарядних станцій;
- надання користувачам (власникам ЕТЗ, операторам станцій, владним органам) зручного доступу до актуальної інформації;
- реалізація гнучкого механізму обмеження споживання електроенергії;

– реалізація можливостей віддаленого моніторингу та управління зарядними станціями;
– проведення аналітики на основі накопичених даних.

Для вирішення цих проблем пропонується розробити комплексне програмне рішення, яке включатиме такі ключові компоненти:

– емулятор зарядних станцій;
– серверна частина додатку;
– веб-застосунок для користування системою;
– мобільний додаток для взаємодії з зарядними станціями.

Реалізація цих компонентів дозволить задовольнити потреби всіх ключових учасників – власників ЕТЗ, операторів станцій та державних органів – та вирішити наявні проблеми у сфері моніторингу та управління зарядною інфраструктурою.

1.2 Постановка завдання

1.2.1 Функціональність продукту

Метою розробки системи управління зарядними станціями для електричних транспортних засобів є створення комплексного програмного рішення, яке забезпечить:

– можливості управління та моніторингу зарядних станцій для електромобілів, керування автопарком під час процесу зарядки;
– надання можливості обрання тарифу електроенергії;
– можливість додавання зарядних станцій до системи;
– надання статистичних даних для зарядних станцій, їх експорт у SVG, CSV формат;
– відображення поточного стану зарядних станцій в реальному часі;
– менеджмент працівників та мережею депо.

Реалізація зазначених функціональних можливостей дозволить вирішити ключові проблеми, пов'язані з управлінням зарядною інфраструктурою ЕТЗ, та задовольнити потреби всіх зацікавлених сторін - власників ЕТЗ, операторів станцій та державних органів.

Невід’ємною частиною окреслення функцій додатку є опис частини функціональності, котру продукт не покриває:

– інтеграція зі сторонніми програмами, окрім основних функціональних можливостей, таких як інтеграція з сервісом споживання електроенергії в регіоні;
– функціональні можливості, окрім керування зарядкою електотранспорту, такі як планування маршрутів або загальна допомога з технічного обслуговування транспорту.

1.2.2 Мета та цілі

Вищеподаний набір функцій додатку покриває визначений відповідно до мети продукту. Окреслимо ключові цілі проєкту які визначають мету:

Для досягнення цієї мети визначено наступні ключові цілі проекту:

а) підвищення ефективності управління та моніторингу зарядної інфраструктури:
   1) можливість додавання нових зарядних станцій до системи;
   2) відображення поточного стану зарядних станцій в реальному часі;
   3) забезпечення віддаленого контролю та управління роботою станцій, включаючи доступ та тарифікацію;

б) оптимізація процесів зарядки електромобілів:
   1) надання водіям можливості вибору тарифів на електроенергію;
   2) збір статистичних даних щодо процесів зарядки (час, кількість заряджених авто тощо);
   3) інтеграція з системами обліку споживання електроенергії;

в) надання аналітики та звітності для прийняття обґрунтованих рішень:
   1) генерація статистичних звітів по зарядних станціях у форматах SVG, CSV;
   2) проведення глибокого аналізу завантаженості та ефективності використання інфраструктури;

г) вдосконалення управління персоналом та мережею депо:

1) менеджмент працівників, відповідальних за обслуговування зарядної інфраструктури;
2) моніторинг та контроль діяльності депо з електромобілями.

Реалізація цих ключових цілей дозволить створити комплексну систему управління зарядними станціями, яка забезпечить ефективне функціонування зарядної інфраструктури, поліпшить досвід водіїв електромобілів та сприятиме прийняттю обґрунтованих управлінських рішень.

Слід зазначити, що моя роль в розробці продукту – розробник клієнтської частини. Відповідно в мої обов’язки входять:

– архітектура клієнтської частини системи;
– створення прототипу додатку;
– підготовка дизайну системи;
– розробка повноцінного веб-додатку;
– розробка мобільного додатку з мінімально необхідними функціями;
– впровадження розробленого продукту.

1.2.3 Основні обмеження

Розробка даного продукту підлягає таки загальним обмеженням:

а) сумісність із протоколами:
   1) система повинна підтримувати стандарт OCPP 1.6 для взаємодії із зарядними станціями різних виробників;
   2) додаток має забезпечувати успішну авторизацію та обмін даними із зарядними станціями через протокол OCPP;

б) безпека даних:
   1) відповіднять стандарту безпеки даних ISO/IEC 27001 [7] для захисту конфіденційної інформації;
   2) система повинна використовувати шифрування даних при передачі та зберіганні, щоб запобігти несанкціонованому доступу;
   3) доступ користувачів до системи повинен контролюватися за допомогою роле-базованого контролю доступу (RBAC);

в) доступність сервісу:

1) користувачі маються доступ до системи 99% часу протягом року;
2) середній час відновлення роботи після критичного збою не повинен перевищувати 4 години;

г) інтеграція із сторонніми системами:

1) продукт повинен забезпечувати API для реалізації кастомізованих клієнтських рішень;
2) формат даних API повинен бути стандартизованим (JSON) для полегшення інтеграції.

Основна інформація щодо:

– цінностей;
– відносин з клієнтами;
– політики монетизації;
– структури витрат.

Була структурована використовуючи бізнес-модель Canva (див. Додаток А).

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги

Базуючись на цілях проєкту, функціях та обмеженнях був сформований список функціональних вимог до продукту (див. табл. 2.1).

Таблиця 2.1 – Функціональні вимоги продукту (таблиця виконана самостійно)

Функція	Опис	Вхідні дані	Обробка	Вихідні дані	Обробка помилок
Управління інформацією / створення депо	Надання власникам мереж депо можливості керувати інформацією / створювати свої депо	Назва, адреса, контактна інформація, обмеження енергії	Додавання, редагування та видалення інформації про депо	Відображення актуальних змін в секціях депо	Повідомлення у вигляді попапу під час керування інформацією депо
Створення співробітника	Надання адміністрації депо можливості створювати користувачів (співробітників/адміністраторів) для управління мережами зарядних станцій	Дані для авторизації, персональна інформація	Перевірка унікальності пошти, валідація полів згідно о RFC 5322, валідація складності пароля	Повідомлення про успішне створення користувача	Повідомлення про помилку зі сторони серверу

Продовження таблиці 2.1

Запрошення існуючого співробітника в депо	Реалізувати можливість надсилати запрошення наявним користувачам в системі приєднатися до депо	Електронна пошта співробітника, роль в депо, OCPP тег	Пошук користувача за електронною поштою, валідація полів згідно RFC 5322	Повідомлення про успішно надіслане запрошення	Повідомлення про помилку зі сторони серверу.
Пошук депо	Надання власнику мережі можливості шукати депо	Ключові слова (назва, адреса)	Наочне фільтрування списку, обробка порожнього списку	Відображення списку депо, котрі відповідають зазначеним ключовим словам	Повідомлення про помилку під час пошуку
Управління стану заряду станцій	Забезпечення інструментів для управління роботи станцій	Ідентифікатор зарядної станцій, команда управління (почати/зупинити зарядку)	Моніторинг змін в реальному часі	Відображення поточного статусу заряду	Інформування про невдало виконану команду
Моніторинг стану депо в реальному часі	Надати можливість слідкувати за станом депо/зарядними станціями в реальному часі	Ідентифікатор об’єкта моніторингу	Встановлення комунікації в реальному часі зі станціями через сервер	Актуальне відображення змін стану зарядних станцій/депо, на картках станцій так і на основній діаграмі споживання енергії	Інформування про невдалу спробу встановити контакт зі станціями

Продовження таблиці 2.1

Налаштування споживання електроенергії в рамках депо	Можливість глобального налаштування обмежень споживання енергії в заданих часових інтервалах депо	Обмеження депо, зарядних станцій, часових інтервалів	Перевірка перетенів часових інтервалів використання електроенергії. Перевірка сумарного споживання відносно обмежень депо. Валідація глобальних часових обмежень налаштувань	Налаштування обмеження споживання енергії. Оновлення графіку головної сторінки.	Модальне вікно з деталями помилки. Інформування користувача про невірно задані значення в процесі налаштування
Статистика та звітність	Збір та надання статистичних даних для аналізу ефективності роботи зарядних станцій	Період для збору статистики	Генерація графічного відображення, формування звітів у форматах Excel, CSV	Відображення статистичних даних у вигляді графіків, доступний до завантаження звіт	Інформування про помилки використовуючи модальне вікно
Створення профілів зарядки	Конфігурація індивідуальних та групових профілів для контролю потужності зарядних станцій	Параметри витрат (максимальна потужність, години витрат, одиниці виміру)	Перевірка суми групової потужності, обробка часових інтервалів, перевірка текстових полів	Параметри потужності для зазначеної зарядної станції	Модальне вікно з деталями помилки та шляхами вирішення

Кінець таблиці 2.1

Управління резервацією зарядних станцій	Організація системи резервації для ефективного використання станцій	Ідентифікатори (OCPP Tag, розетка, зарядна станція), додаткові деталі резервації (назва, опис)	Отримання списку розеток відповідно до обраної зардяної станції, валідація текстових полів	Створення блоку резервації в календарному представленні	Модальне вікно з інформацією про помилку
Зміна мови додатку	Надані інструменту для зміни мови додатку	Бажана мова інтерфейсу	Пошук наявного перекладу	Зміна інтерфейсу сторінки відповідно до обраної мови	Використання мови за замовчуванням, в разі відсутності бажаної мови відображення

Детальний опис функціональних вимог надає чітке уявлення про важливий функціонал додатку, що, в свою чергу, вплине на якість кінцевого продукту.

2.2 Нефункціональні вимоги

Відповідно до моєї ролі у проєкті – розробник клієнтської частини, був сформований перелік нефункціональних вимог, слідування яким забезпечує реалізацію надійного, зрозумілого, безпечного та сучасного продукту.

а) сумісність з протоколами:
   1) відповідність OCPP 1.6;
   2) обробка версій протоколу;

б) безпека даних:
   1) контроль доступу (RBAC);
   2) використання HTTPS та Angular Security;

в) доступність сервісу:

1) обробка помилок та інформування користувача;
2) адаптивний інтерфейс;
3) впровадження з використанням CDN;

г) продуктивність:
1) ефективна обробка запитів з використанням останніх версій HTTP2/3;
2) оптимізація продуктивності;
3) тестування продуктивності;
4) використання віртуального DOM;
5) кешування;
6) мініфікація коду;
7) ліниве завантаження сторінок;
8) мінімізація використання поліфілів;
9) gzip;

д) супровід:
1) документація;
2) модульна архітектура;
3) контроль версій (Git).

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проектування ПЗ

Перед початком реалізації клієнтської частини, після проведеного аналізу та поставленої задачі, був визначений основний набір функцій для акторів системи.

3.1.1 Use Case діаграма

Зобразімо необхідний, відповідно до бізнес потреб, функціонал продукту використовуючи Use Case діаграму (див. Додаток Д).

3.1.2 Діаграма діяльності

Активність співробітника системи зображена у вигляді діаграми діяльності (див. Додаток Ж). Дана діаграма слугує підґрунтям для подальшого планування архітектурних рішень клієнтської частини.

Важливо зазначити, що діаграма спроектована саме для web-версії продукту зі сторони адміністратора, власника та співробітника депо, адже дана частина включає в себе найбільшу кількість вузлів та кроків.

3.2 BPMN діаграма

Основна цінність та ідея додатку полягає в гнучкому управлінні витратами електроенергії. Відповідно процес обробки запитів від бізнесу та конфігурація конекторів відповідно до вимог і є основним бізнес процесом. Використовуючи BPMN нотацію зобразімо цей процес (див. рис. 3.1).

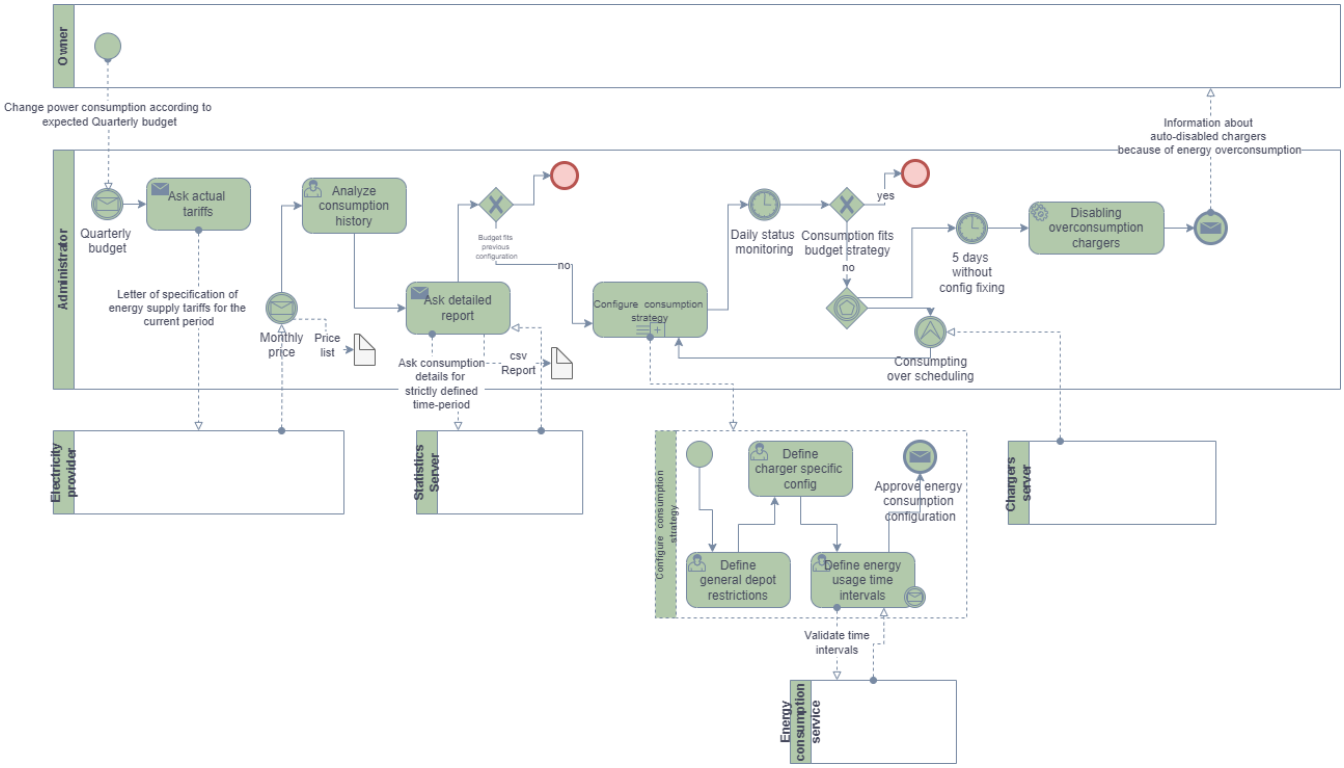


Рисунок 3.1 – Процес оновлення конфігурації споживання енергії описана нотацією BPMN (рисунок виконано самостійно)

3.3 Проектування архітектури ПЗ

У сфері розробки веб-додатків, особливо у комплексних системах управління автопарками електромобілів, важливо застосовувати гнучкі та масштабовані архітектурні підходи. З огляду на критичність ефективного споживання електроенергії для бізнесів, що працюють з електротранспортом, розробка зручного додатку, який автоматизує управління споживанням електроенергії, є ключовою. Такий додаток має забезпечувати швидкість, стабільність та інтуїтивність у використанні.

3.3.1 Вибір архітектури

Для досягнення цих цілей було обрано Onion архітектуру [8], яка, хоч і не є новинкою у серверній розробці, проте її застосування у клієнтській частині є менш поширеним. Ця архітектура структурує додаток у послідовні рівні відповідальності, забезпечуючи слабке зв’язування та високу адаптивність компонентів.

3.3.2 Асинхронне виконання операцій в Angular

Варто зазначити, що Angular та інші сучасні фреймворки підтримують асинхронний код, що робить Onion архітектуру особливо привабливою для реалізації. В результаті користувач отримає швидкий відгук системи на дії, а розробники ізольований в рамках доменів код, що полегшить підтримку проєкту.

3.3.3 Взаємодія ключових компонентів

Слідуючи обраному архітектурному рішенню важливо окреслити критичні аспекти, які потребуються найбільшої уваги в процесі реалізації:

– легкі компоненти без бізнес-логіки;
– управління станом системи;
– уникнення недоцільного розміщення бізнес-логіки поза відповідними сервісами.

Відповідно можна виділити абстрактні частини, котрі повторюються в кожному домені системи, і є обов’язковими для вище окреслених аспектів:

– компоненти, які відображаються користувачеві;
– клієнти для спілкування з API;
– стан як єдине джерело істини;
– домен, що містить бізнес-логіку.

Управління потоками даних, так звана «Оркестрація», прихована за методами фасаду, що зменшує шари представлення та бізнес логіки в рамках домену. Компоненти отримують чіткий контракт зі сторони фасаду, що дає змогу змогу розробнику незалежно масштабувати необхідні частини.

У домені шлюзи виступають як інтерфейси, що використовують механізм Dependency Injection фреймворку Angular. Це дозволяє динамічно замінювати реалізації інтерфейсів, наприклад, на сервіси під час тестування. Бізнес-логіка виконується доменом і є незалежною від модулів представлення, клієнта і стану (див. Додаток Е).

3.4 Приклад найцікавішого алгоритму

3.4.1 Формування вимог до алгоритму

Згідно основної мети, яка і є головною цінністю продукту зазначеною в минулих розділах і проілюстрована на рисунку 3.1, важливим кроком є створення зрозумілого процесу взаємодії користувача з інтерфейсом для налаштування плану споживання електроенергії (див. рис. 3.2). З метою успішного виконання даної задачі – спроектуймо алгоритм побудови часових інтервалів споживання електроенергії.

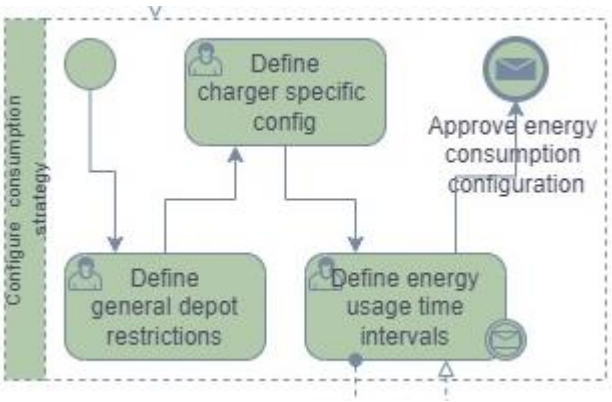


Рисунок 3.2 – Процес налаштування плану споживання енергії (рисунок виконано самостійно)

Перш за все, виділимо функціональні вимоги для алгоритму роботи з часовими інтервалами (див. табл. 3.1).

Таблиця 3.1 – Функціональні вимоги для алгоритму управлінням часовими інтервалами споживання електроенергії (таблиця виконана самостійно)

Функція	Опис	Вхідні дані	Обробка	Вихідні дані	Обробка помилок
Значення за замовчуванням	За замовчуванням наявність одного часового інтервалу в межах загальних налаштувань депо	Обмеження споживання енергії депо в kW, початок та кінець дії плану споживання	Створення єдиного інтервалу з точністю до секунд максимально допустимою протяжністю згідно налаштувань депо. Кількість використання енергії якого дорівнює максимально допустимому значенню для депо	Інтервал з максимальними допустимими значеннями для депо	Відображення відповідних валідаційних помилок для текстових полів
Створення нового інтервалу	У відповідній частині форми занести значення в межах конфігурації депо та підтвердити зміни	Час початку, кінця та к-ть доступної до використання енергії в даному часовому проміжку	Новостворений інтервал має найвищий пріорітет, що призводить до необхідності змінити існуючі часові інтервали. За потреби інтервали будуть: видалені, розбиті на частини, доповнені	Оновлений список з інтервалами, де новостворений інтервал знаходиться у повному обсязі (не піддається редагуванню, розбиттю під час створення)	Якщо початок або кінець плану споживання електроенергії виходить за рамки вказані для депо – план оновлюється згідно налаштувань депо

Продовження таблиці 3.1

Редагування існуючого інтервалу	У списку наявних інтервалів використовуючи відповідні поля користувач змінює налаштування існуючого часового інтервалу	Список оновлених часових інтервалів	Відредагований інтервал позначається як новостворений і для нього дійсні вимоги «Створення нового інтервалу»	Аналогічно вимогам «Створення нового інтервалу»	Аналогічно обробці помилок для «Створення нового інтервалу»
Видалення існуючого інтервалу	Користувач має змогу натиснути на відповідну кнопку видалення, якщо плані містить більше 1го часового проміжку	Список без обраного на видалення інтервалу	Пропущений часовий інтервал заповнюється максимально допустимим значенням використаної енергії. Час початку – кінець минулого інтервалу, або час початку дії плану. Час закінчення – час початку наступного інтервалу, або час кінця дії плану. Час початку/кінця дії плану використовується за умови, якщо попереднього/наступного інтервалу не існує відповідно.	Оновлений список інтервалів заповнений з максимальними доступними депо значеннями використання енергії.	Аналогічно обробці помилок для «Створення нового інтервалу»

Кінець таблиці 3.1

Об’єднання інтервалів	При умові, що дотичні за часом інтервали мають однакове значення споживання енергії – вони мають бути об’єднані в один	Дотичні інтервали з однаковим значенням споживання енергії	Інтервали об’єднуються в один інтервал, зі збереженням протяжності і єдиним значенням споживання енергії	Єдиний інтервал, де час початку – час початку дії першого інтервалу, а час кінця – час кінця другого інтервалу, за умови, що інтервали були відсортовані за часом початку від меншого до більшого	Аналогічно обробці помилок для «Створення нового інтервалу»

Слід наголосити, що план використання:

– не має містити часових проміжків без конфігурації;
– новостворений інтервал вставляється у план цілком, а в разі перетину, перетнуті плани модифікуються надаючи перевагу доданому.

3.4.2 Проектування

В процесі створення даного алгоритму була проведена презентація, з метою командного обговорення. Результатом обговорення є частина алгоритму зображена на рисунку 3.3.

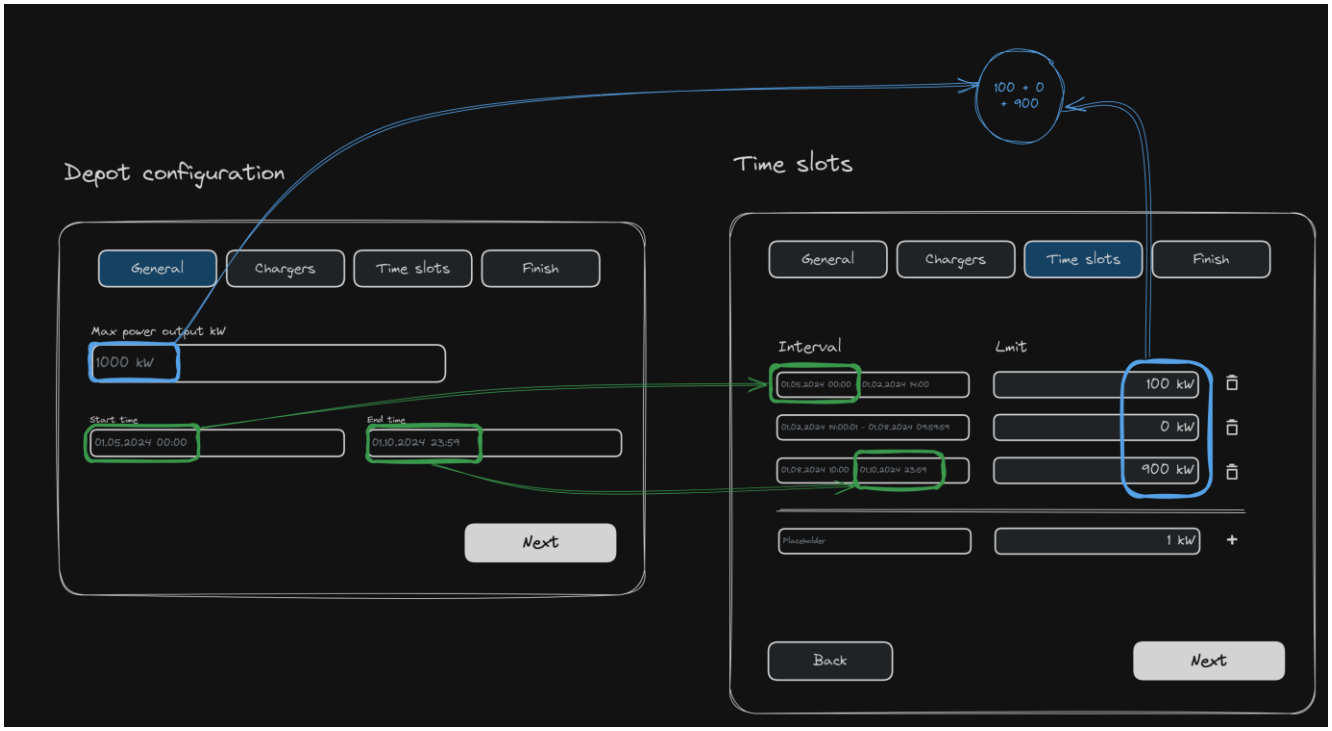


Рисунок 3.3 – Спрощений прототип створення часових інтервалів (рисунок виконано самостійно)

Базуючись на визначеній поведінці алгоритму, яка була сформована на основі функціональних вимогах був підготований алгоритм роботи з часовими інтервалами (див. рис. 3.4).

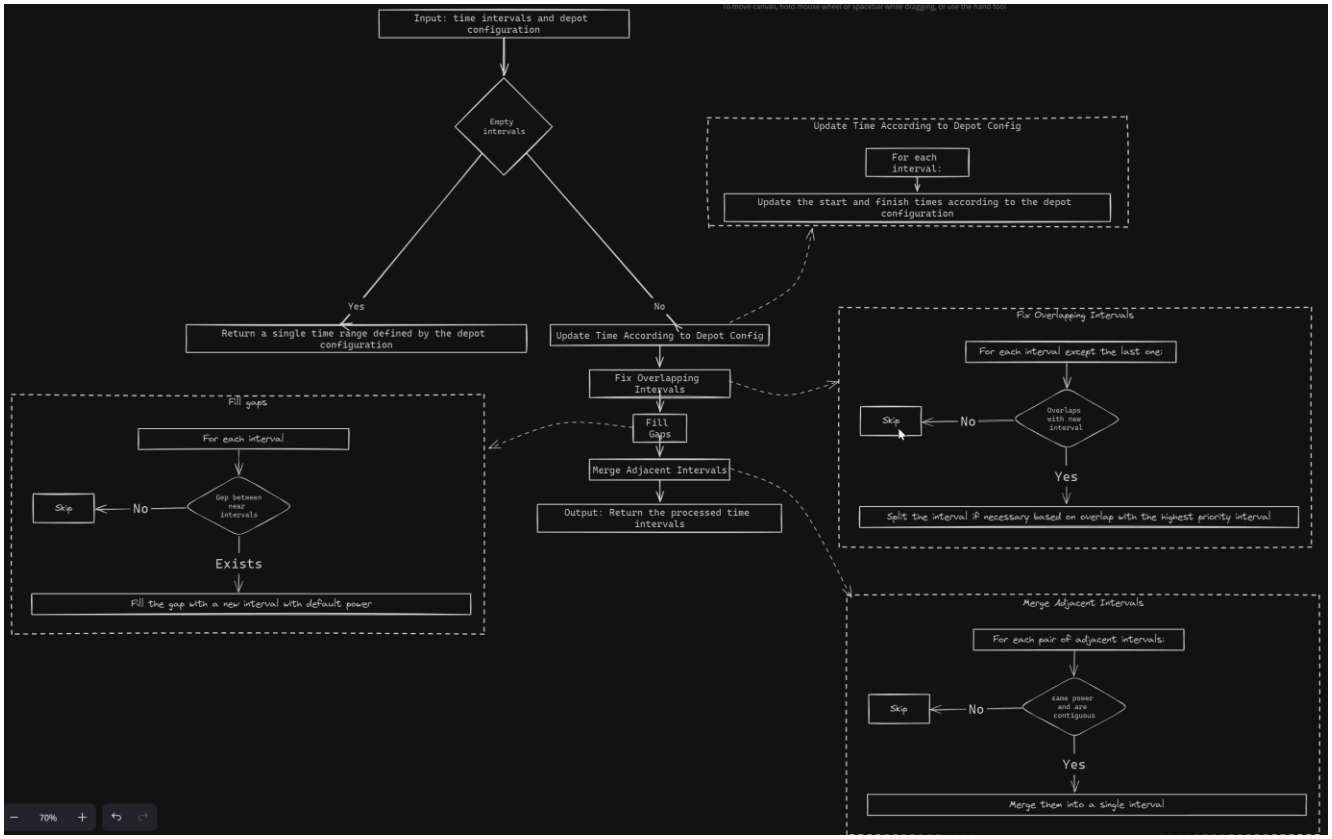


Рисунок 3.4 – Алгоритм формування часових інтервалів для процесу налаштування споживання електроенергії (рисунок виконано самостійно)

Реалізація даного функціоналу не є тривіальним алгоритмом, тож імплементація виконується згідно TDD (Test-Driven Development) [9] підходу. Тест кейси для сервісу винесені в Додаток Б.

Зазначу, що з моєї сторони проводилося unit тестування ключового функціоналу додатку, для чого використовувався фреймворк Jest (див. рис. 3.5).

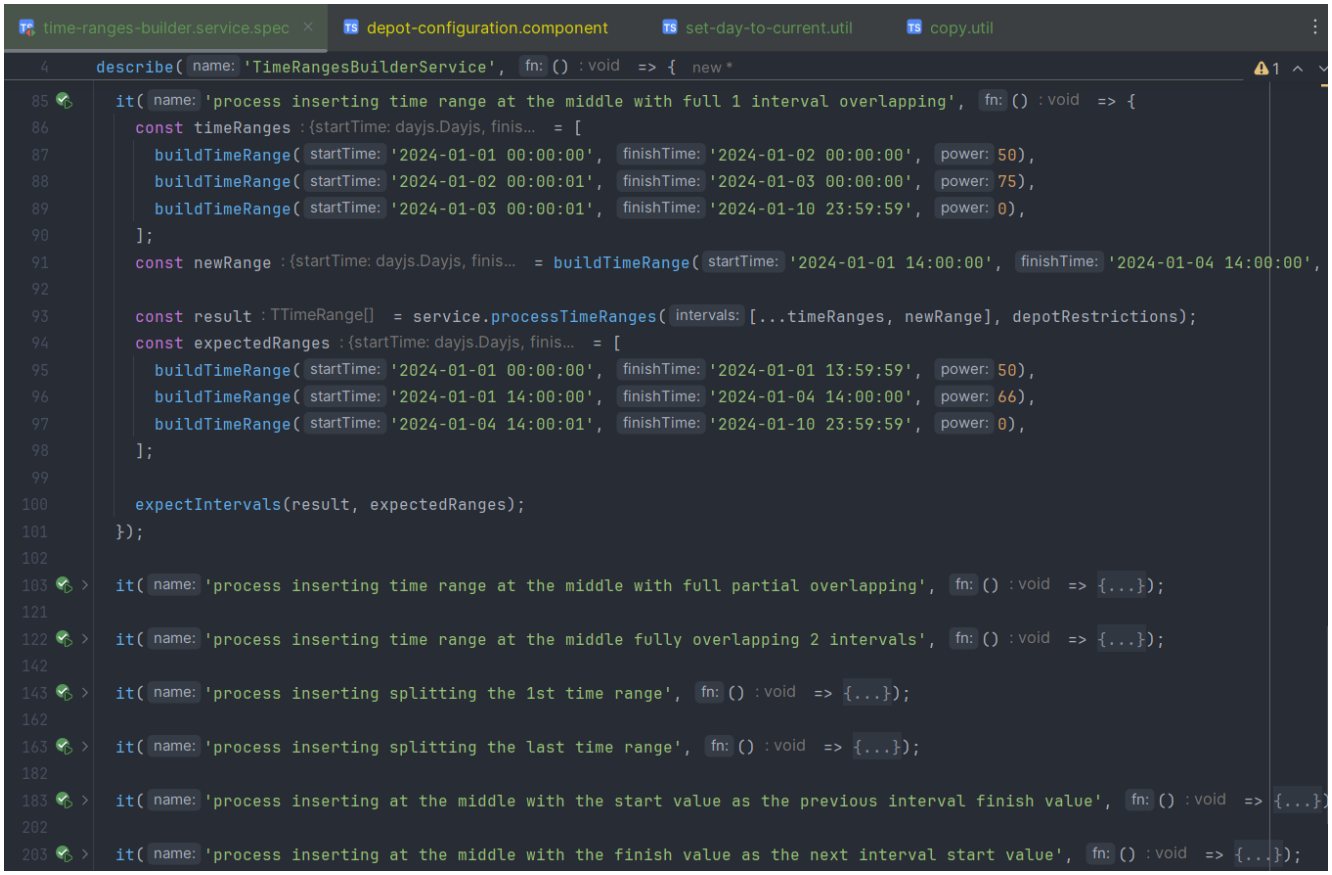


Рисунок 3.5 – Тести для алгоритму створення інтервалів споживання енергії (рисунок виконано самостійно)

Цей підхід забезпечив надійність та стабільність критично важливого алгоритму системи.

3.4.3 Реалізація

Згідно описаному алгоритму та відповідним тест-кейсам був підготований сервіс (детальніше див. Додаток В) для обробки інтервалів згідно описаним функціональним вимогам. Структура сервісу наведена на рисунку 3.6.

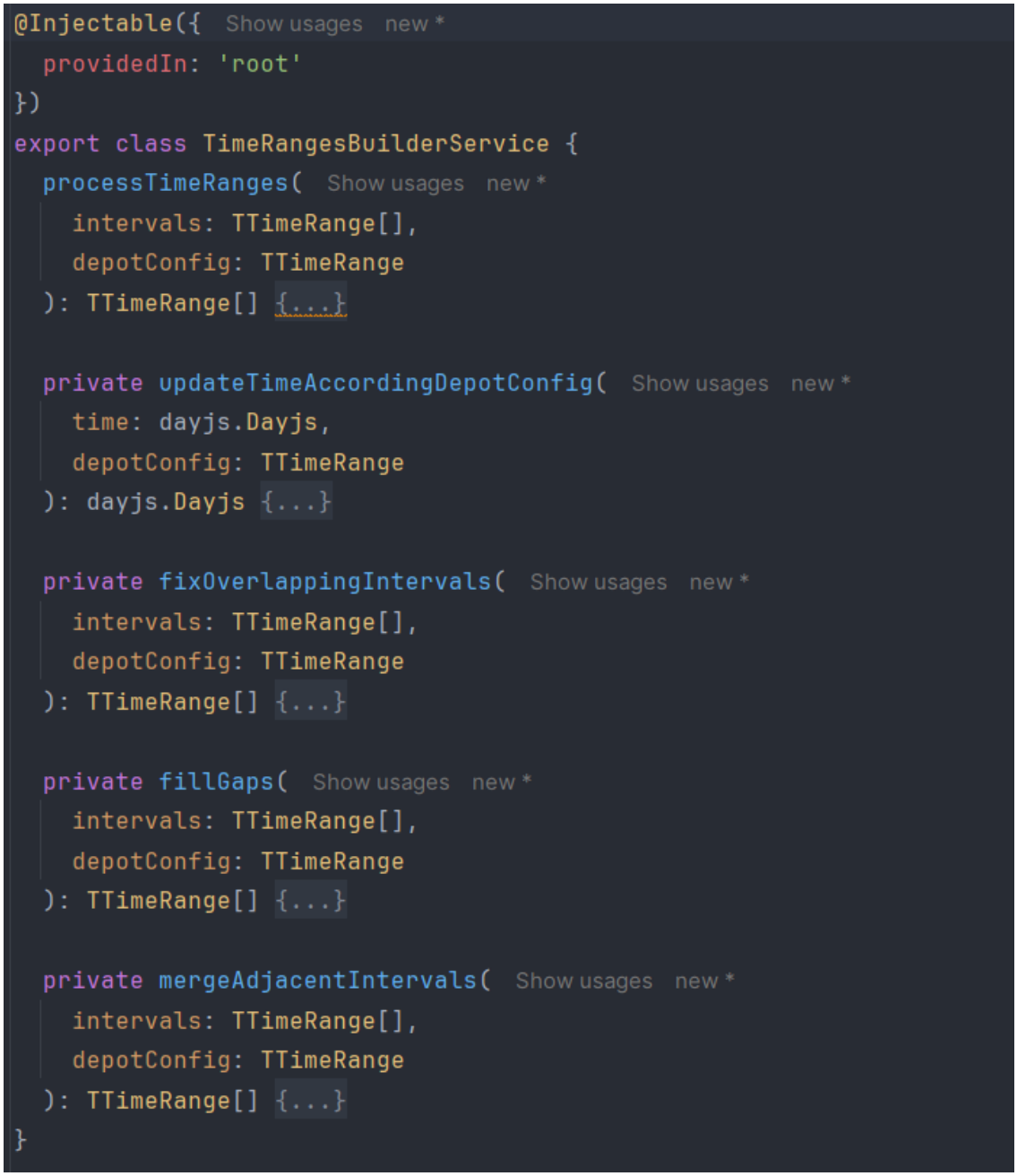


Рисунок 3.6 – Структура сервісу для роботи з інтервалами споживання електроенергії (рисунок виконано самостійно)

Переконаймося, що алгоритм працює згідно тестів, підготованих заздалегідь (див. рис. 3.7).

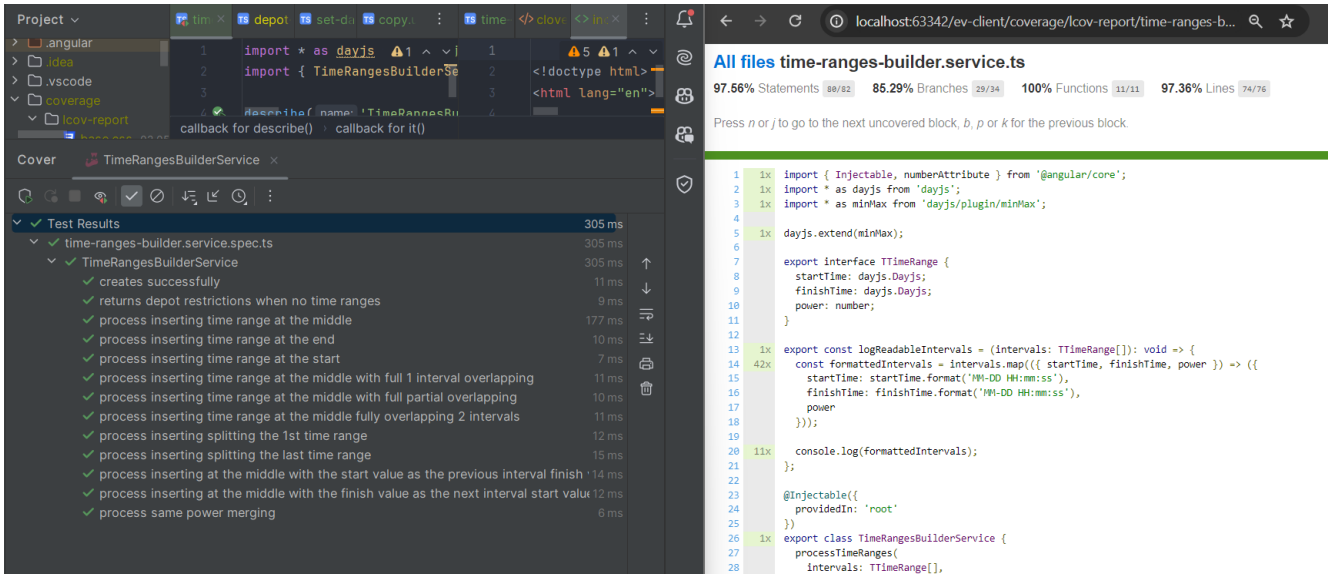


Рисунок 3.7 – Результат проходження тестів алгоритмом для роботи з часовими інтервалами (рисунок виконано самостійно)

В результатів було розроблено алгоритм покритий тестами на 97.5% відповідно до зазначених функціональних вимог системи.

3.5 Прототипування додатку

Як було зазначено вище, моя роль в розробці даного продукту – розробник клієнтської частини. Беручи до уваги обмежену кількість людських ресурсів в команді в мої обов’язки входить і створення прототипу майбутнього додатку.

Слід зазначити, що з метою розробки якісного продукту, робота над дизайном була обмежена створенням прототипу, адже відповідальність за рішення в інтерфейсі лежала на мені. Основною метою варфрейму (див. рис. 3.8) було узгодити з командою реалізацію критично необхідних функції продукту за мінімально можливий, для якісного результату, час.

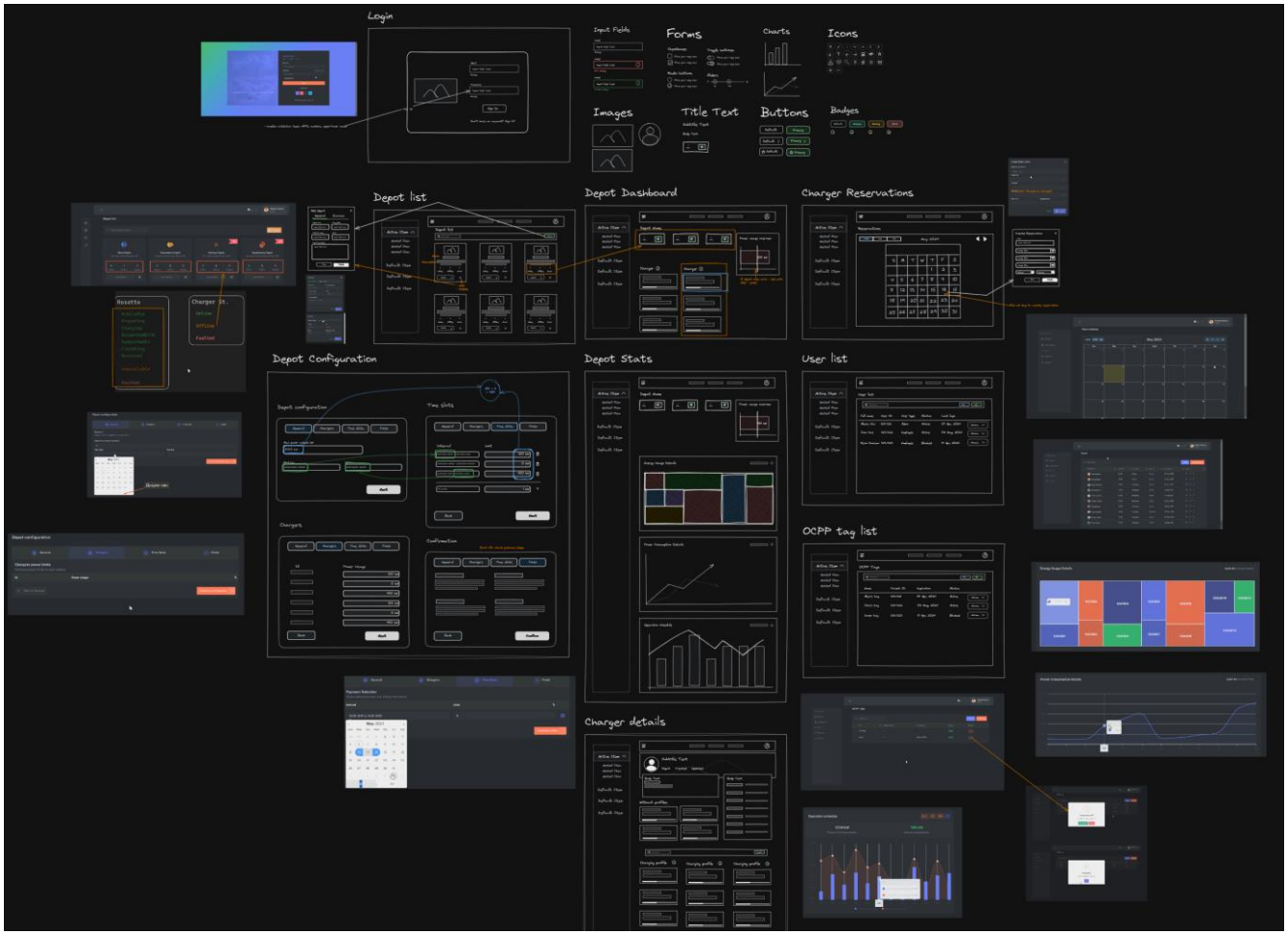


Рисунок 3.8 – Прототип системи для управління витрат електроенергії в депо електротранспорту (рисунок виконано самостійно)

Зазначимо, що під час розробки прототипу були підготовані основні необхідні для функціонування продукту. Другорядні сторінки, модальні вікна, підказки та інші не ключові компоненти, узгодження поведінки яких не потребувало участі команди, не були проілюстровані прототипами.

Після авторизації в системі, власнику мережі, необхідна можливість ефективно взаємодіяти з депо. Беручи до уваги важливість даного процесу – розроблена сторінка з загальним списком усіх підконтрольних організацій (див. рис. 3.9).

В рамках даного списку користувач має змогу:

– фільтрувати;
– створювати;
– редагувати;

– видаляти;
– контролювати поточний стан усіх депо одночасно.

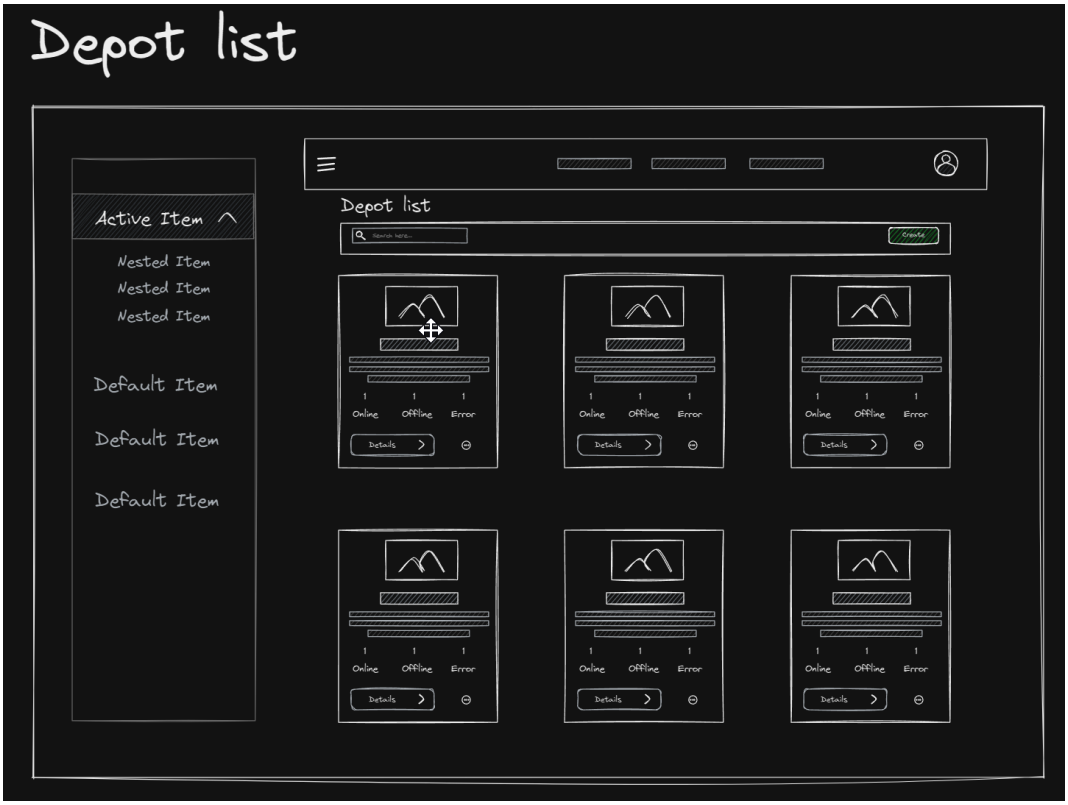


Рисунок 3.9 – Сторінка зі списком депо доступний власнику мережі (рисунок виконано самостійно)

Більш детальна інформація (див. рис. 3.9), як про зарядні станції, так і про кожну підключену розетку доступна і співробітникам депо (див. рис. 3.10). На даній сторінці адміністрація:

– контролює процес споживання електроенергії в реальному часі;
– у разі помилок – ідентифікує проблемну станцію;
– керує процесом зарядки електротранспорту.

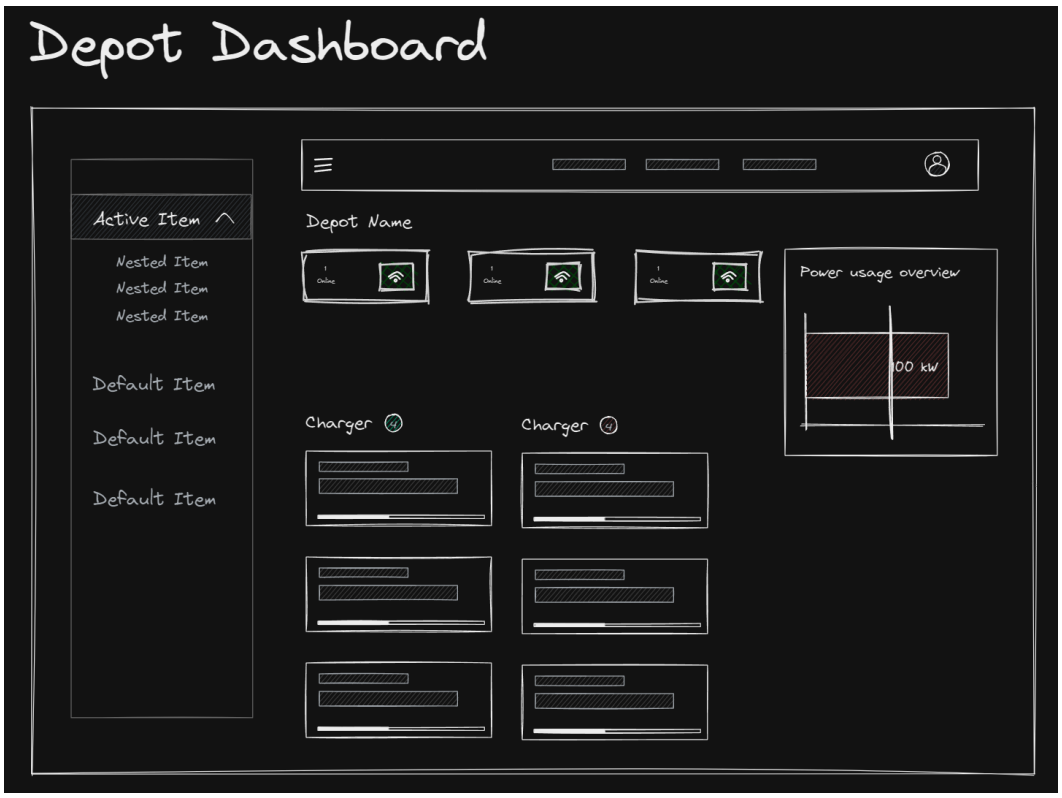


Рисунок 3.10 – Головна сторінка депо з інформацією про стан станцій в реальному часі (рисунок виконано самостійно)

Статистична інформація про споживання електроенергії в минулому дає змогу аналізувати великі об’єми даних і налаштовувати майбутні плани споживання більш гнучко відповідно до плану споживання та виділеного бюджету. Командою була виділена задача – реалізувати необхідний для детального аналізу механізм відображення даних. В результаті, було підготовано три візуальних представлення:

– treemap з інформацією про індивідуальне споживання еоектроенергії кожною станцією окремо;
– лінійний графік загального енергоспоживання;
– комбінований графік для відображення запланованого та актуального графіку роботи депо.

Вищенаведені представлення дають змогу гнучко обирати часові періоди, які необхідні користувачу системи, та в шидко експортувати як в CSV так і Excel файли. Слід зазначити, що графічна інформація представлена в межах однієї

сторінки, що дає змогу ефективно порівнювати та аналізувати дані щодо роботи станцій (див. рис. 3.11).

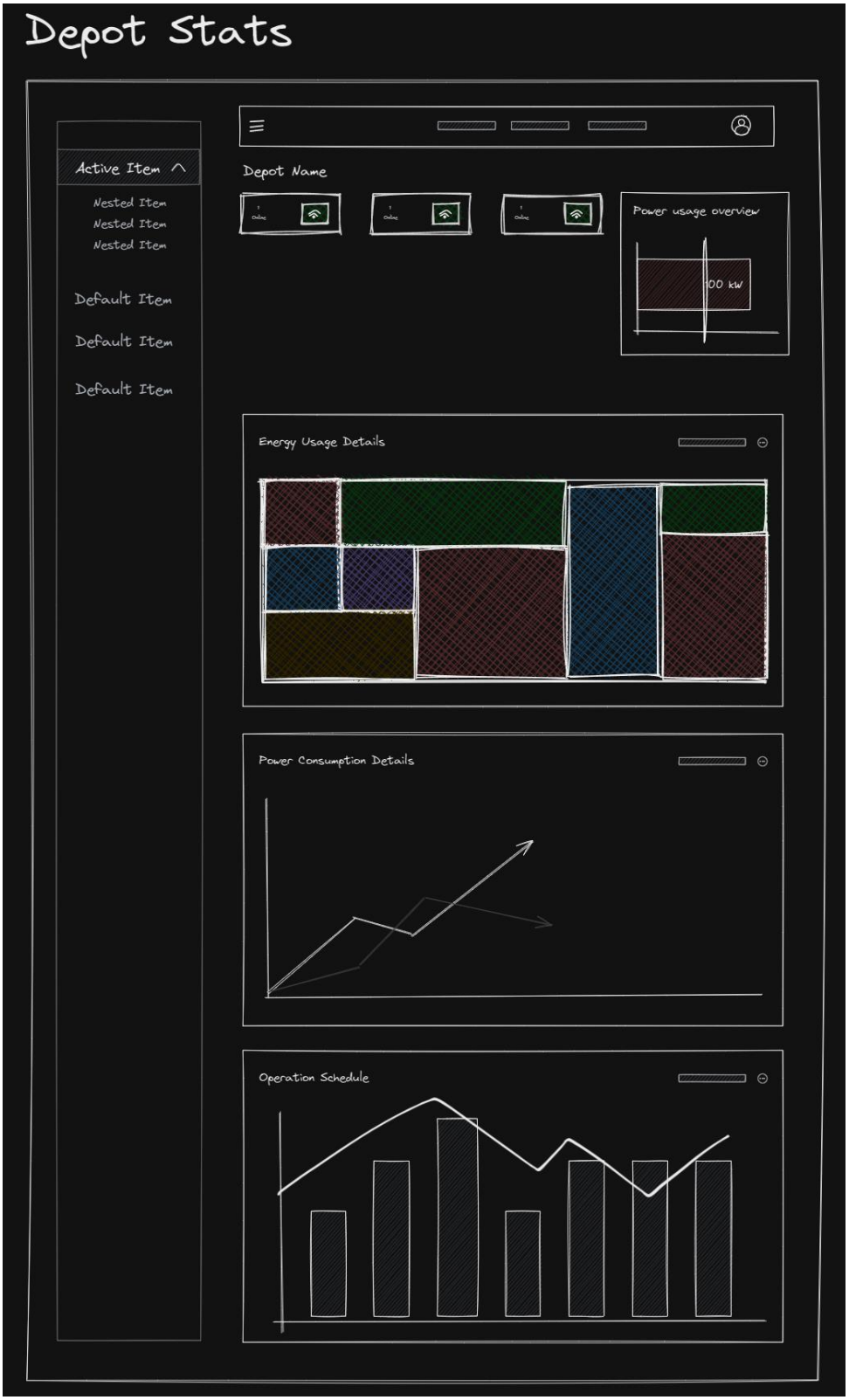


Рисунок 3.11 – Статистика і звіти про споживання електроенергії зарядними станціями (рисунок виконано самостійно)

Не менш важливою частиною в процесі роботи з депо електротранспорту є бронювання часових рамок для використання станції. Даний функціонал включає в себе можливість взаємодіяти з інтерфейсом календаря для створення, редагування та видалення записів про резервації (див. рис. 3.12).

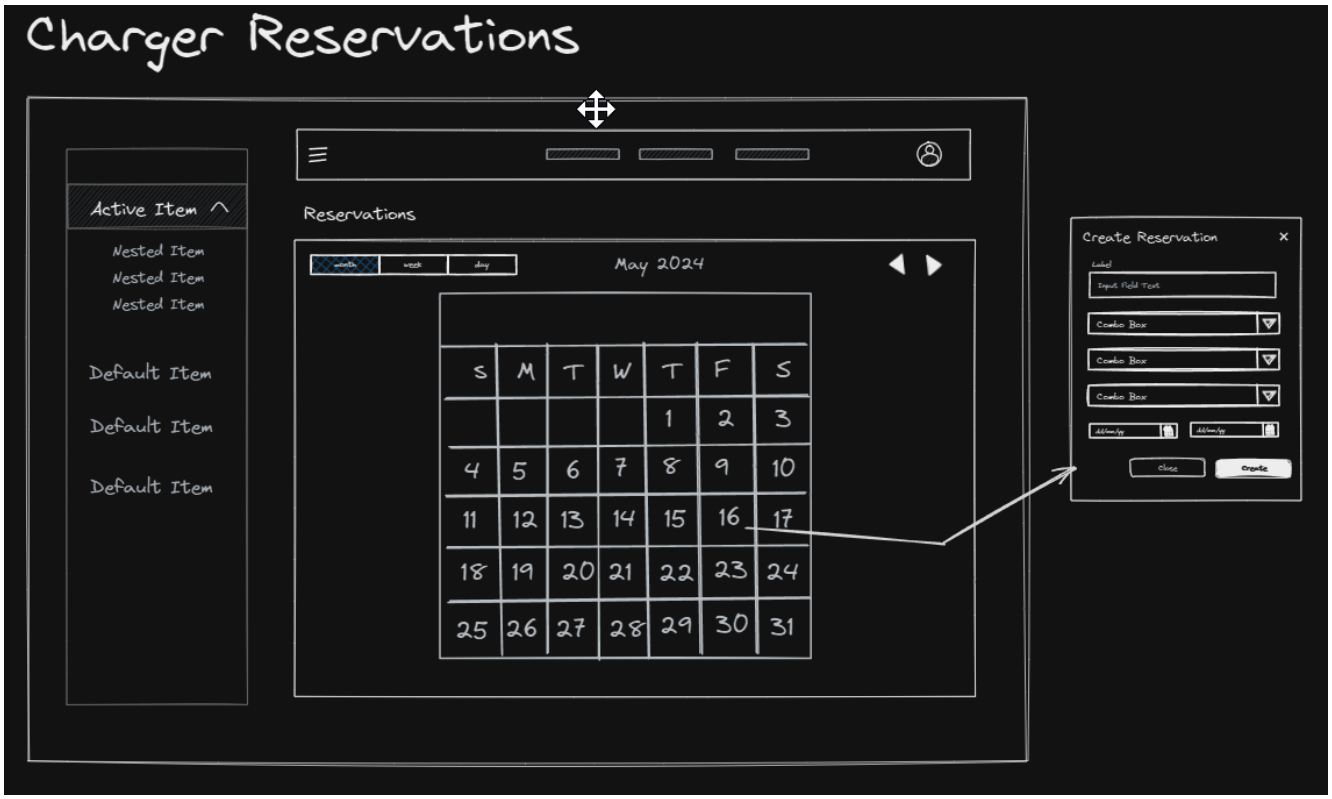


Рисунок 3.12 – Сторінка резервації зарядних станцій (рисунок виконано самостійно)

Слід визнати, що будь-яка система адміністрування має на меті організувати зручну роботу з великими об’ємами даних. В нашій до даних даного типу можна віднести: списки користувачів та осрр теги. Для візуального представлення даних сутностей було прийнято рішення скористатися табличним представленням, що дасть змогу фільтрувати, сортувати, додавати, редагувати та видаляти дані в декілька кліків. Реалізація даного представлення є тривіальною задачею, проте сторінки прототипів також були розроблені (див. рис. 3.13).

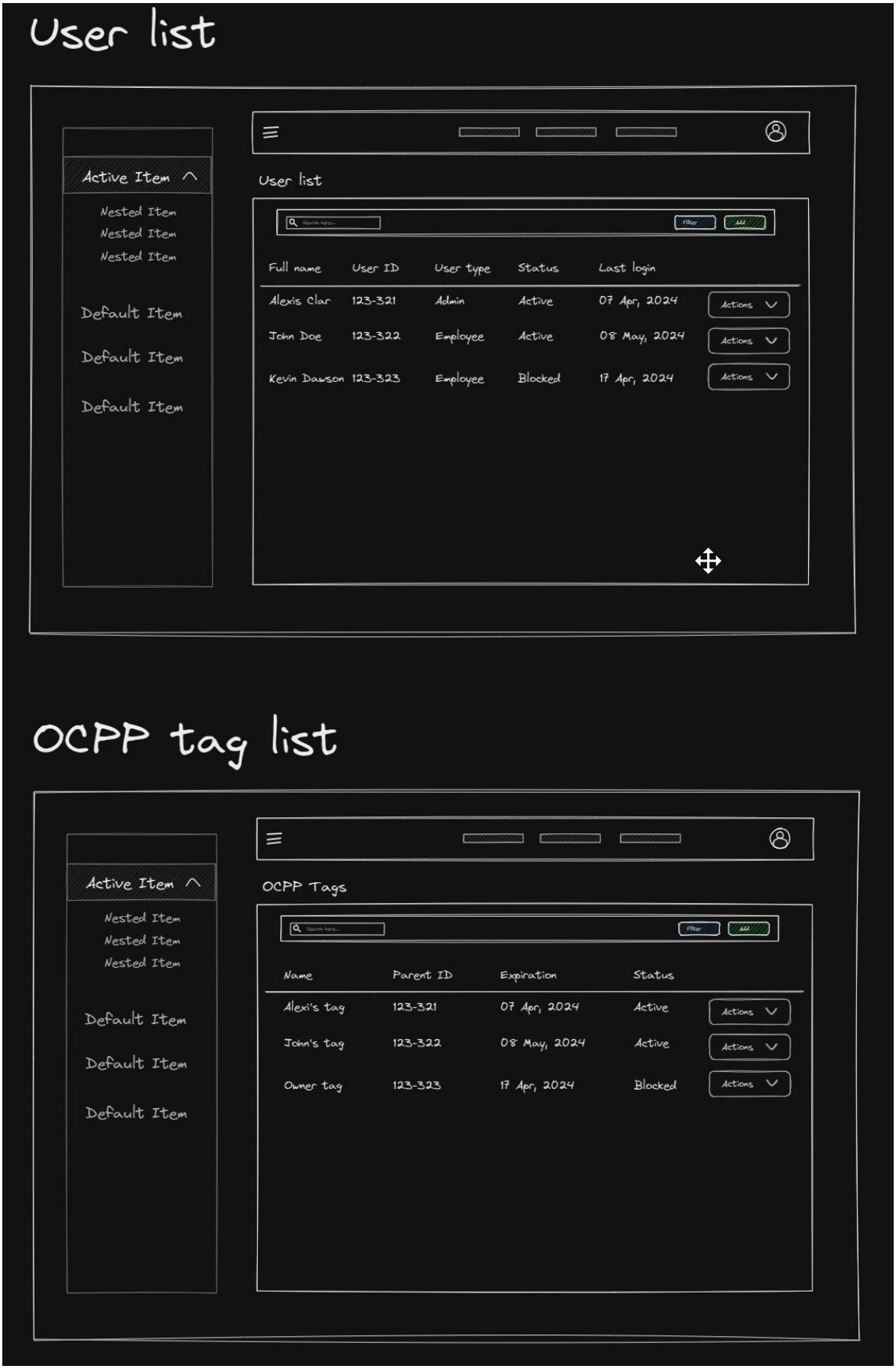


Рисунок 3.13 – Прототип табличного представлення сутностей (рисунок виконано самостійно)

Звичайно, не можемо оминути прототип налаштувань, частини якого вже були продемонстровані на рисунку 3.3 та описані згідно BPMN нотації на рисунку 3.2. Налаштування плану споживання електроенергії, яке складається з чотирьох кроків дозволяє користувачу організувати довгостроковий план, згідно якого станції системи будуть використовувати електроенергію (див. рис. 3.14).

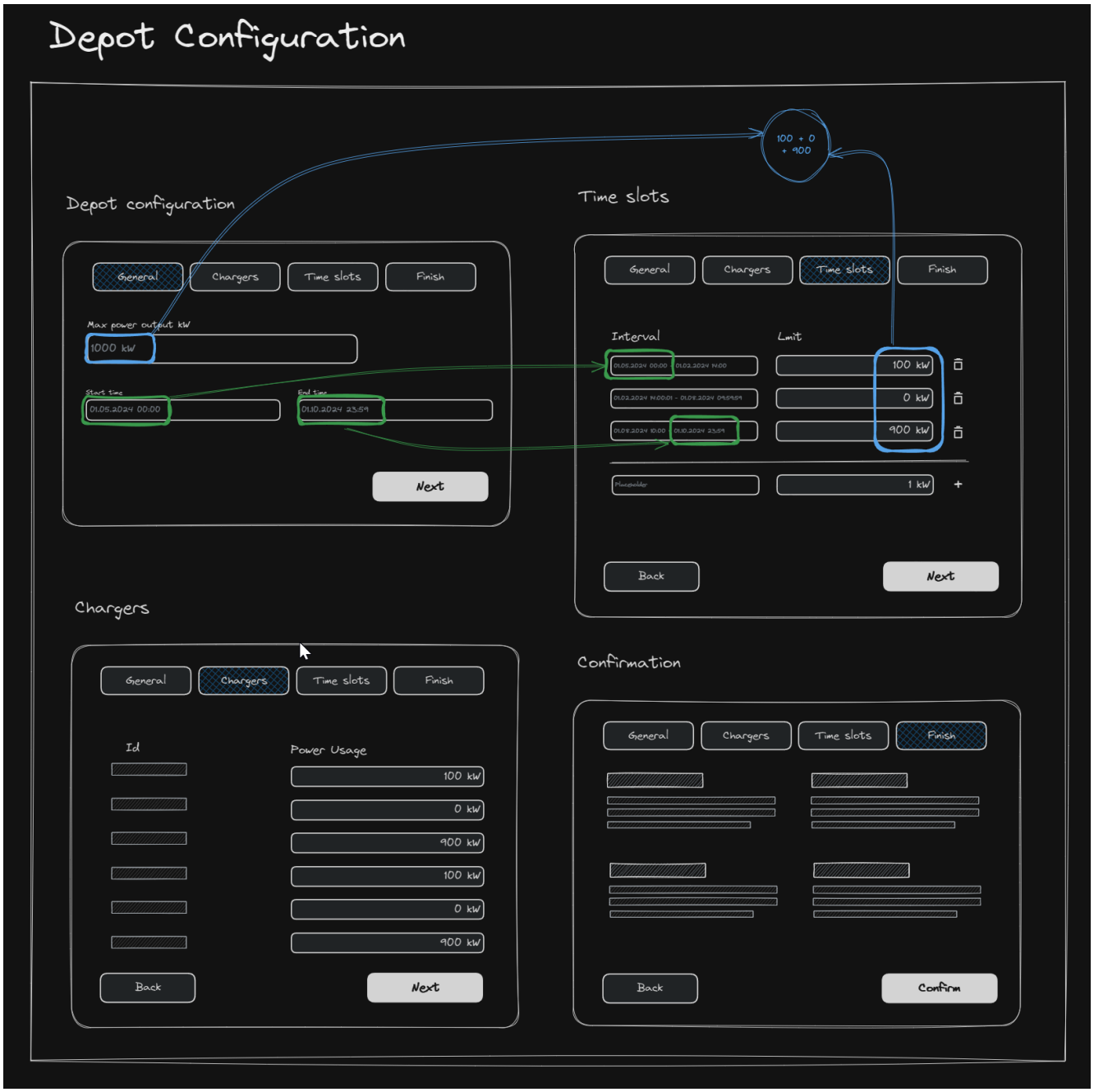


Рисунок 3.14 – Прототип налаштування плану споживання електроенергії (рисунок виконано самостійно)

Розробка вищенаведених прототипів використовувалась командою протягом повного життєвого циклу продукту, що допомогло в формуванні функціональних вимог, виявленні критично важливих функцій додатку та, безпосередньо, реалізації продукту.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Управління обліковими записами

4.1.1 Реєстрація та запрошення користувачів

Реєстрація в системі доступна для всіх ролей (див. рис. 4.1), окрім супер адміністратора, адже даний користувач, вноситься в систему при купівлі продукту.

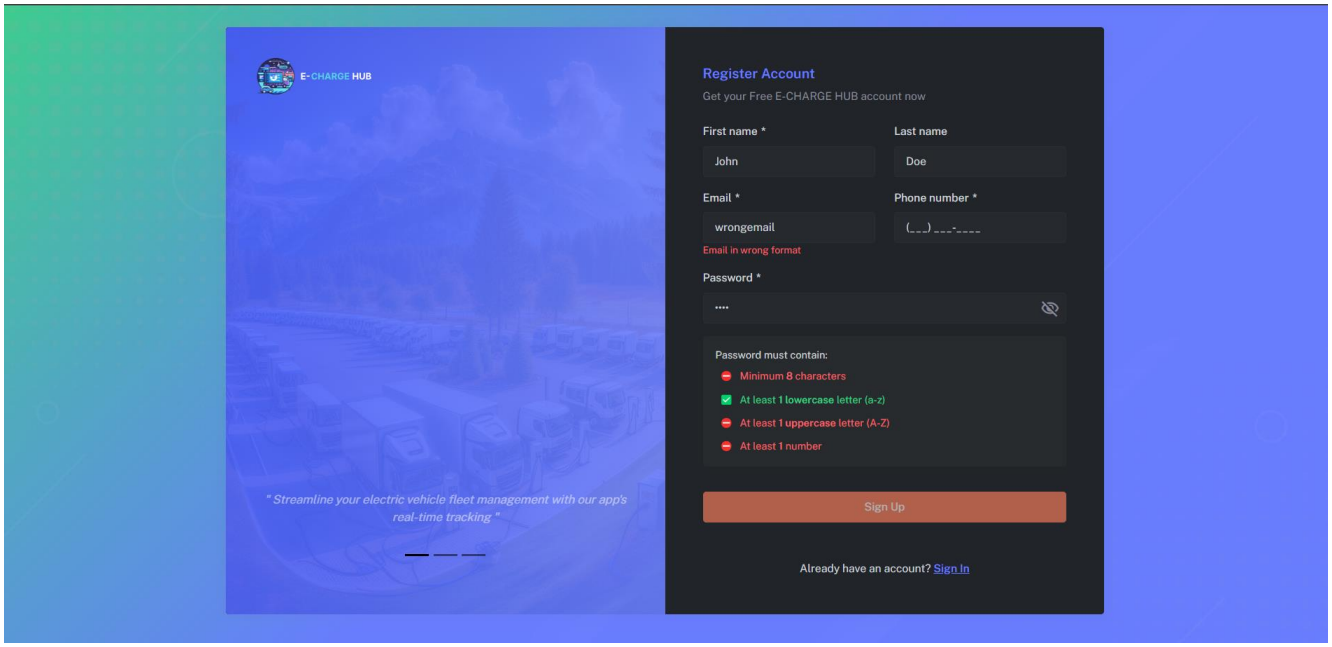


Рисунок 4.1 – Сторінка реєстрації в систему моніторингу зарядних станцій депо електротранспорту (рисунок виконаний самостійно)

Адміністратори мають можливість керувати користувачами системи використовуючи інструменти редагування, видалення, фільтрації (див. рис. 4.2).

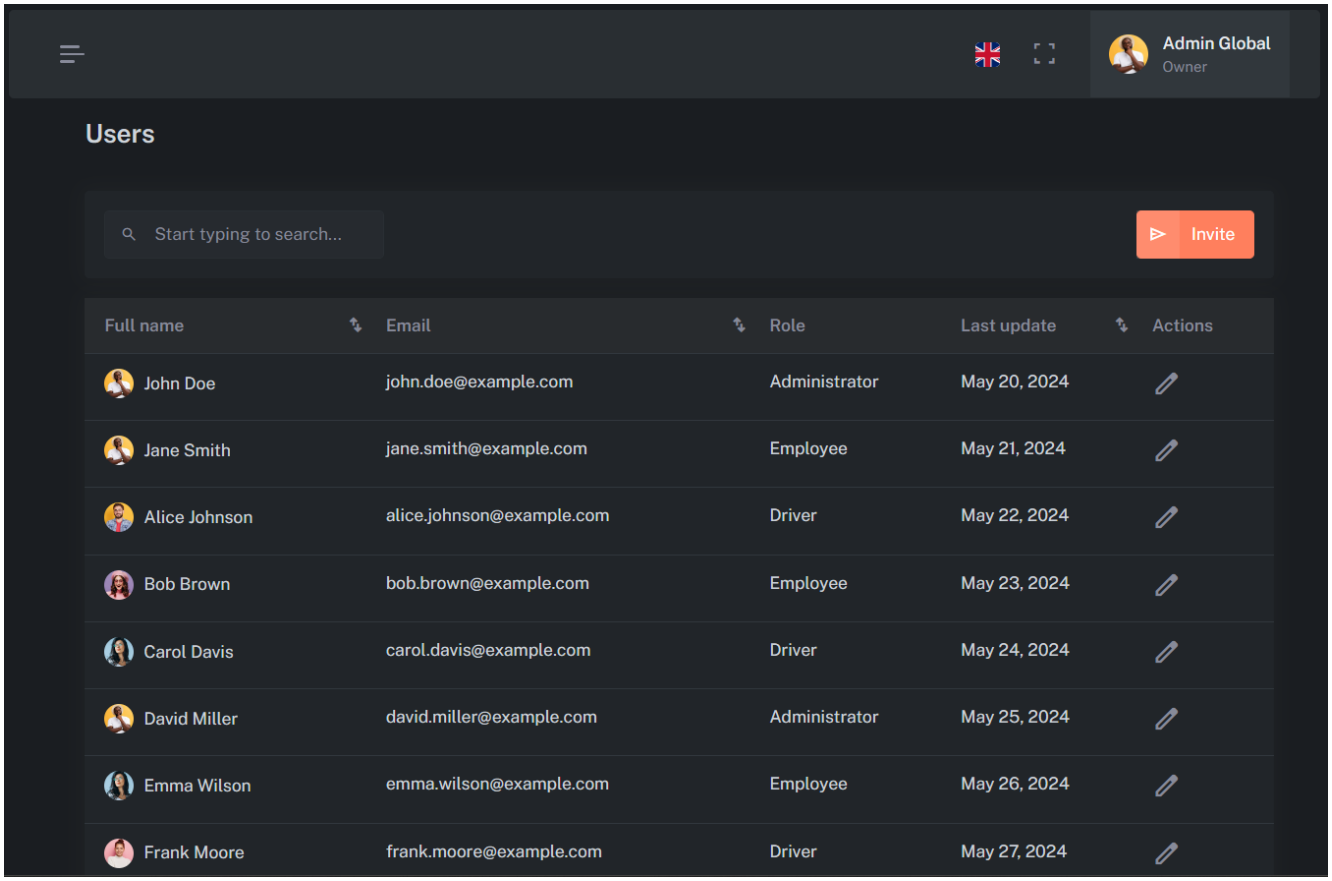


Рисунок 4.2 – Список користувачів системи моніторингу зарядних станцій депо електротранспорту (рисунок виконаний самостійно)

Реалізовано можливість запрошувати нових співробітників, інших адміністраторів та водіїв приєднатися до депо (див. рис. 4.3).

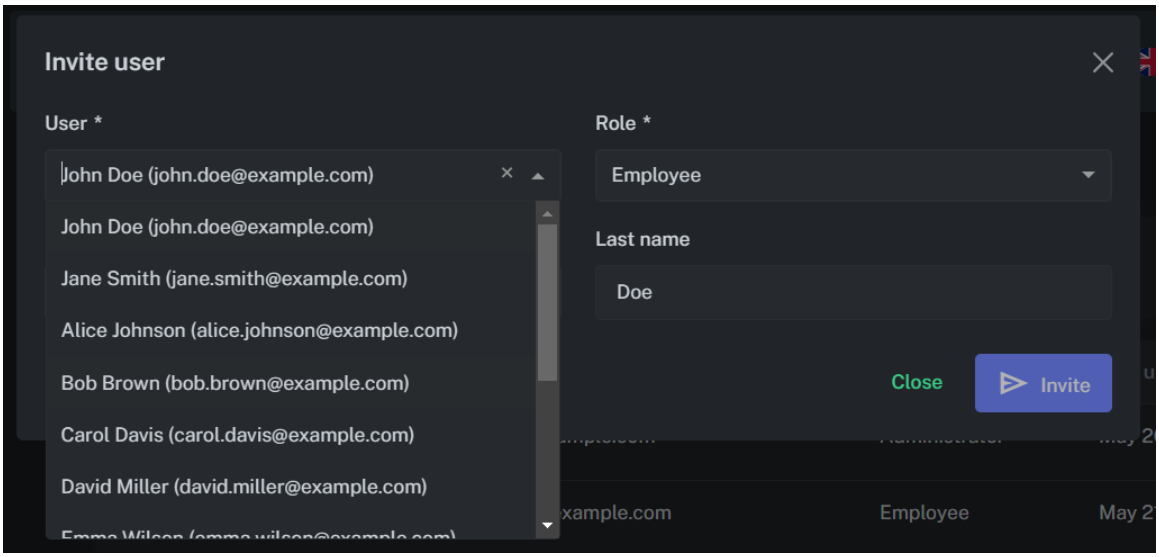


Рисунок 4.3 – Модальне вікно запрошення користувача в депо (рисунок виконаний самостійно)

Після запрошення користувач отримує електронний лист із посиланням для завершення реєстрації та зміни пароля.

4.1.2 Аутентифікація

Аутентифікація в системі виконується за допомогою JWT-токенів. Цей метод забезпечує захищений та ефективний спосіб аутентифікації користувачів та управління їхніми сесіями. Токени використовуються для верифікації кожного запиту до сервера (див. рис. 4.4), з токена шляхом декодування отримується поточна роль користувача (див. рис. 4.5).

```
export const tokenInterceptor: HttpInterceptorFn = (req : HttpRequest<?> , next : HttpHandlerFn ) : Observable<HttpEvent<...>> => {  Show usages  devDmytroHorkun
  const $token : Signal<string>  = inject(SessionStore).token;

  return $token() ? next(req.clone( update: { setHeaders: { Authorization: `Bearer ${$token()}` } })) : next(req);
};
```

Рисунок 4.4 – Функція додавання токена авторизації в кожен запит до серверу (рисунок виконаний самостійно)

```
function base64UrlDecode(str: string) : string  {  Show usages  devDmytroHorkun
  // Replace URL-safe characters and add padding
  str = str.replace( searchValue: /-/g,  replaceValue: '+').replace( searchValue: /_/g,  replaceValue: '/');
  switch (str.length % 4) {
    case 0:
      break;
    case 2:
      str += '==';
      break;
    case 3:
      str += '=';
      break;
    default:
      throw new Error( message: 'Invalid base64 string');
  }

  // Decode Base64
  const decodedString : string  = atob(str);
  // Convert string to Uint8Array
  const bytes : Uint8Array  = new Uint8Array(decodedString.length);
  for (let i : number  = 0; i < decodedString.length; ++i) {
    bytes[i] = decodedString.charCodeAt(i);
  }
  // Decode UTF-8
  const decoder : TextDecoder  = new TextDecoder();
  return decoder.decode(bytes);
}

export const decodeJWT = (token: string): TDecodedToken => {  Show usages  devDmytroHorkun *
  const [headerEncoded : string , payloadEncoded : string , signatureEncoded : string ] = token.split( separator: '.');
  const payloadJson : string  = base64UrlDecode(payloadEncoded);

  return JSON.parse(payloadJson);
};
```

Рисунок 4.5 – Реалізація декодування токена авторизації для отримання даних користувача (рисунок виконаний самостійно)

Після декодування токена дані користувача зберігаються та використовуються додатком для контролю доступу до компонентів системи використовуючи механізм RBAC (див. рис. 4.6).

Рисунок 4.6 – Angular директива для контролю доступу до компонентів системи (рисунок виконаний самостійно)

Даний механізм використовує композицію з вже існуючою директивою та взаємодії з фасадом аутентифікації, що забезпечує миттєву реакцію системи на будь-які зміни доступів.

4.1.3 Валідація полів

Слід зазначити, що перевірка введених значень у системі здійснюється по мірі введення значень. Серед найрозповсюдженіших можна виділити:

– перевірка електронної пошти: згідно визначенням RFC 5322 та RFC 5321;
– перевірка паролів: здійснюється перевірка складності, що включає довжину символів, наявність літер в різних регістрах та чисел. Більш наочно перевірка була проілюстрована на рисунку 4.1;
– номери телефону мають чітко визначений формат та «маску», що забезпечує необхідність вводити виключно числа, а спеціальні символи проставляються автоматично;

– одиниці виміру енергії та потужності також використані для відповідних полів в поєднанні з механізмом «маска».

4.2 Обробка змін в реальному часі

Для реалізації взаємодії між клієнтом і сервером в режимі реального часу використовується SignalR. В Angular для обробки цих даних було прийнято рішення використати патерн Event Bus (див. рис. 4.7), що дозволяє ефективно поширювати події та оновлювати стани компонентів.

Даний механізм використовується для:

– транзакцій;
– отримання змін для підключених розеток (див. рис. 4.8);
– отримання сповіщень про перевищення лімітів депо;
– конфігурації профілів зарядки (див. рис. 4.9);
– скасування резервацій;
– змін доступності станцій.

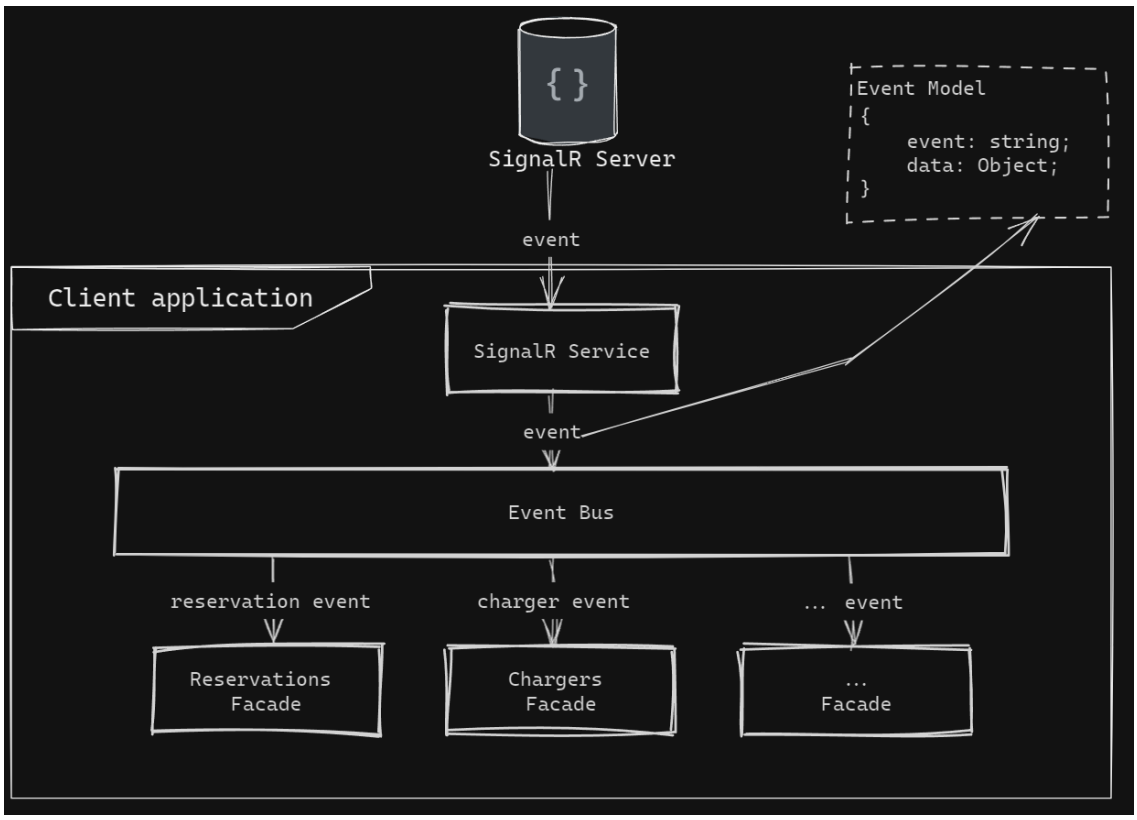


Рисунок 4.7 – Схематичне відображення потоку подій з використанням патерну Event Bus(рисунок виконаний самостійно)

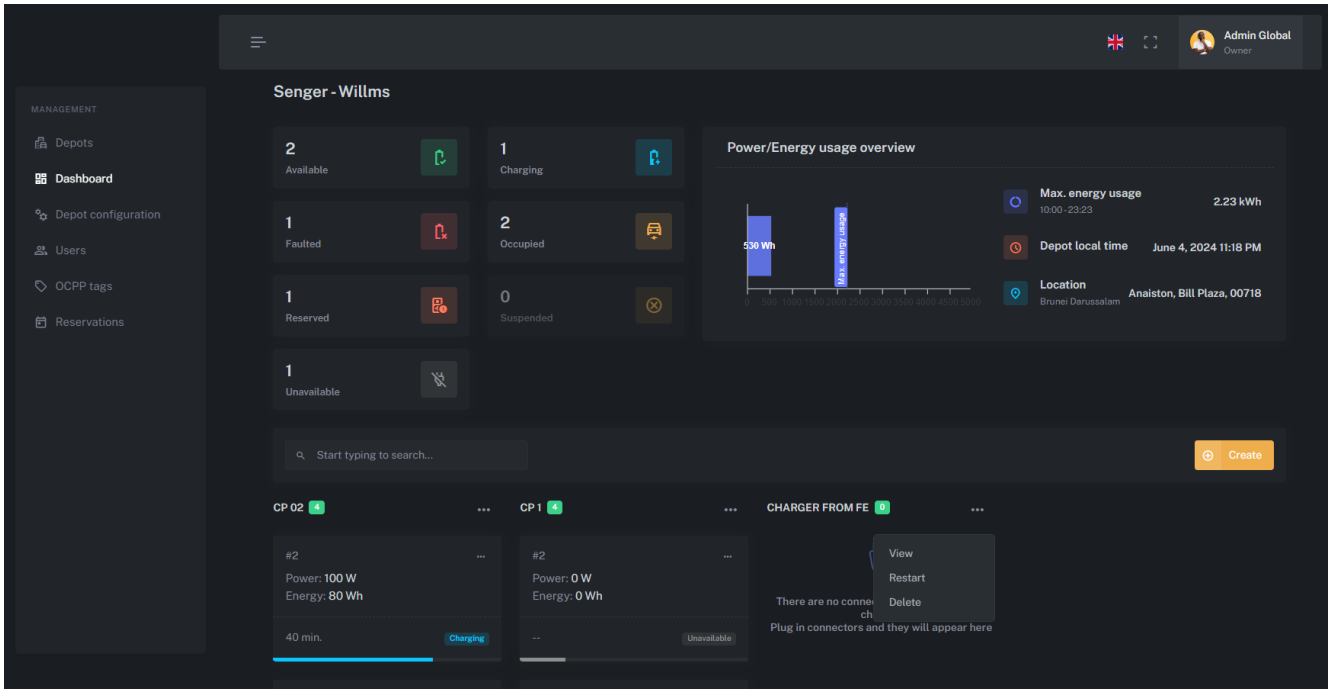


Рисунок 4.8 – Головна сторінка з відображенням процесу роботи зарядних станцій (рисунок виконаний самостійно)

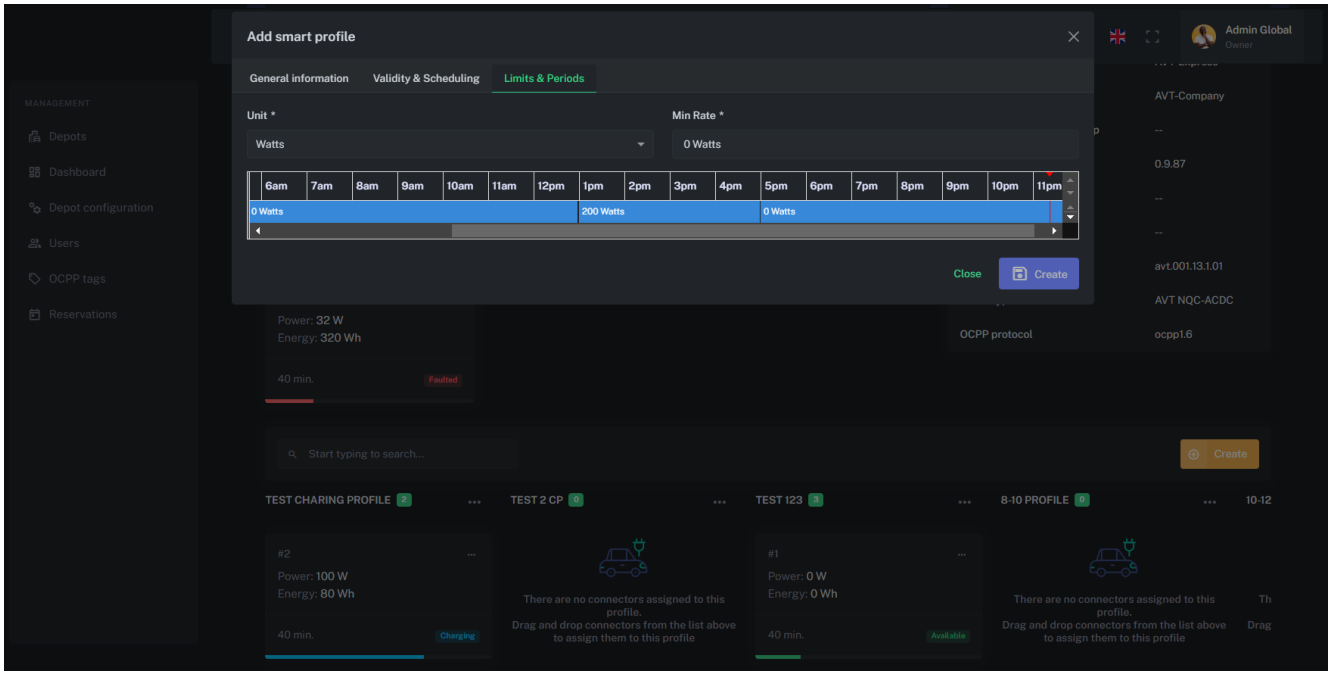


Рисунок 4.9 – Налаштування профіля для обмеження потужності зарядних станцій (рисунок виконаний самостійно)

Event Bus сервіс, реалізація якого представлена на рисунку 4.10 , використовується для поширення подій, отриманих від SignalR сервісу, до інших

компонентів Angular. Це дозволяє різним частинам додатку реагувати на події незалежно одна від одної.

Рисунок 4.10 – Сервіс Event Bus який виконує оркестрацію подій отриманих в реальному часі (рисунок виконаний самостійно)

В цей час фасадний сервіс відповідає за прослуховування повідомлень від Event Bus сервісу та оновлення відповідних станів у рамках необхідного домену. Приклад функції отримання подій представлена на рисунку 4.11;

Рисунок 4.11 – Приклад функція прослуховування подій (рисунок виконаний самостійно)

Даний підхід забезпечив гнучкість і масштабованість рішення, зменшивши зв'язаність між доменами системи.

4.3 Резервації

Резервації зарядних станцій є важливою функцією для управління доступом до ресурсів системи. Вони доступні для Супер адміністратора, Адміністратора та Водія. Для покращення користувацького досвіду резервації були реалізовані у вигляді інтерактивного календаря з використанням FullCalendar. Мобільна версія, яка є особливо важливою для водіїв, була створена з використанням Ionic 8.

4.3.1 Реалізація календаря

FullCalendar[10] є потужним інструментом для відображення та управління подіями у вигляді календаря. Він забезпечує зручний і інтерактивний інтерфейс для роботи з резервaціями (див. рис. 4.12). Окремо варто наголосити на стабільній підтримці даної білбіотеки, що, беззаперечно робить її фаворитом серед аналогічних рішень.

Серед переваг використання варто виділити:

- інтерактивність: Календар підтримує функції drag & drop і resize events, що робить процес управління резервaціями більш інтуїтивним і зручним для користувачів;
- гнучкість: FullCalendar дозволяє легко налаштовувати і адаптувати відображення подій під потреби користувачів;
- масштабованість: Інструмент добре підходить як для десктопної, так і для мобільної версії додатку, забезпечуючи послідовний користувацький досвід.

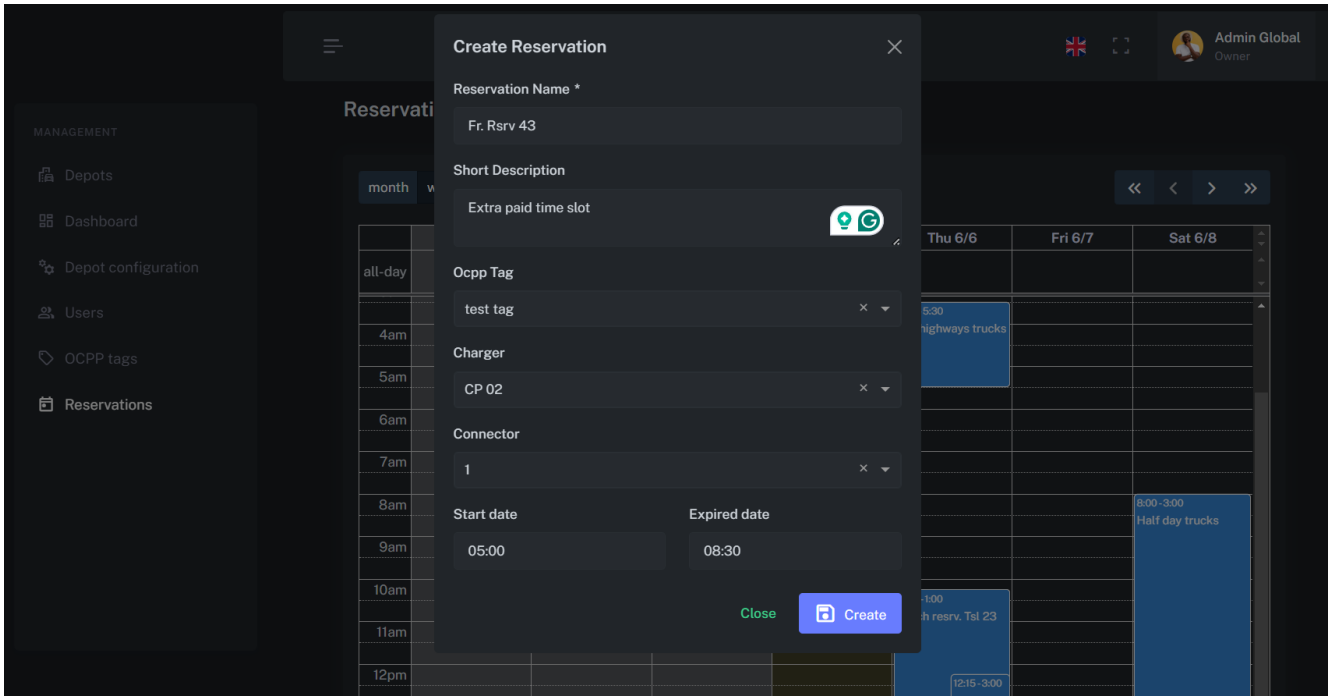


Рисунок 4.12 – Сторінка створення резервації зарядної станції (рисунок виконаний самостійно)

4.3.2 Мобільна версія

Для забезпечення зручного користування мобільною версією системи, особливо для водіїв, використовується Ionic 8[11]. Це дозволяє створити адаптивний і інтуїтивний інтерфейс для управління резерваціями на мобільних пристроях (див. рис. 4.13). Серед переваг особливо важливими є:

– крос-платформеність: Ionic 8 дозволяє створювати додатки, які однаково добре працюють як на IOS, так і на Android, що забезпечує широку доступність для всіх користувачів;
– ефективна розробка: Завдяки використанню єдиного коду для різних платформ, розробка мобільних додатків стає ефективнішою.

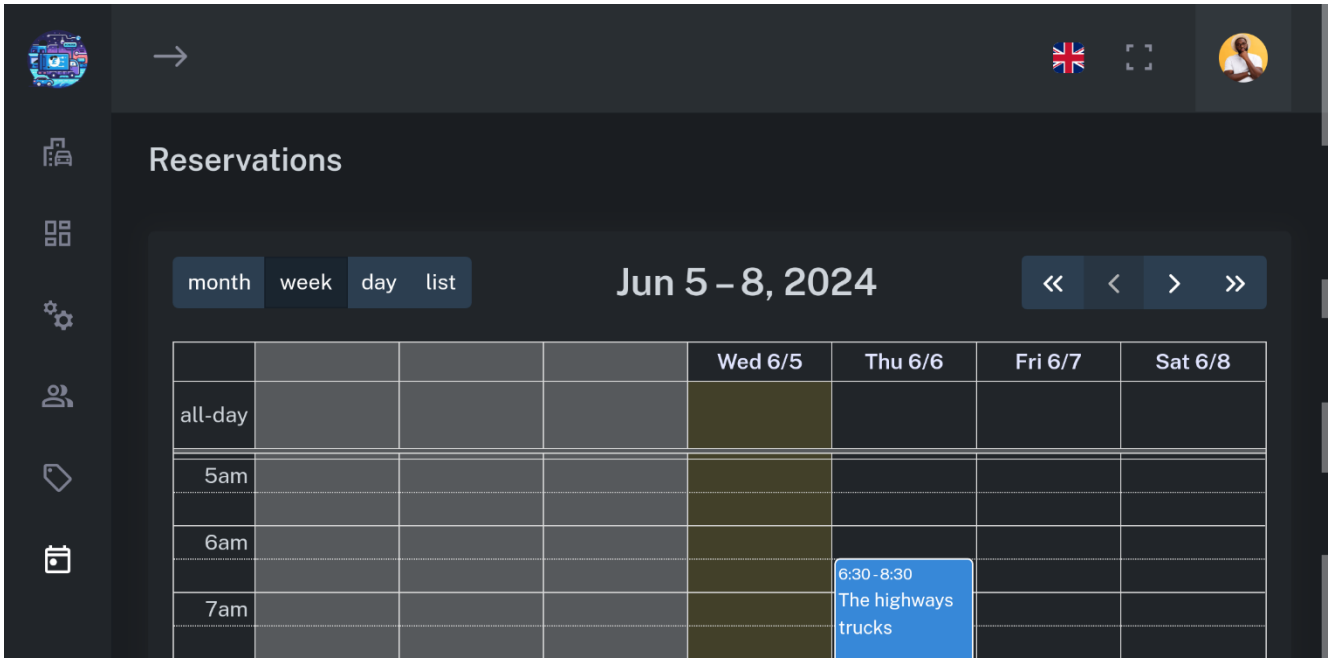


Рисунок 4.13 – Календар резервацій мобільного додатку для Android реалізований з використанням Ionic 8 і Capacitor (рисунок виконаний самостійно)

Реалізація резервацій у вигляді інтерактивного календаря FullCalendar, а також мобільної версії на базі Ionic 8, забезпечує зручний та інтуїтивний інтерфейс для управління зарядними станціями. Користувачі можуть легко створювати, редагувати та відміняти резервації, що робить систему гнучкою та зручною у використанні.

4.4 Обмеження споживання енергії в рамках депо

Обмеження споживання енергії депо є важливою функцією для оптимізації використання ресурсів та запобігання перевантаженню електричної мережі. Даний процес дозволяє накладати обмеження на споживання енергії в певних часових рамках (див. рис. 4.14), забезпечуючи ефективне управління енергетичними ресурсами (див. рис. 4.15).

Деталі реалізації алгоритму були описані у розділі 3.4.

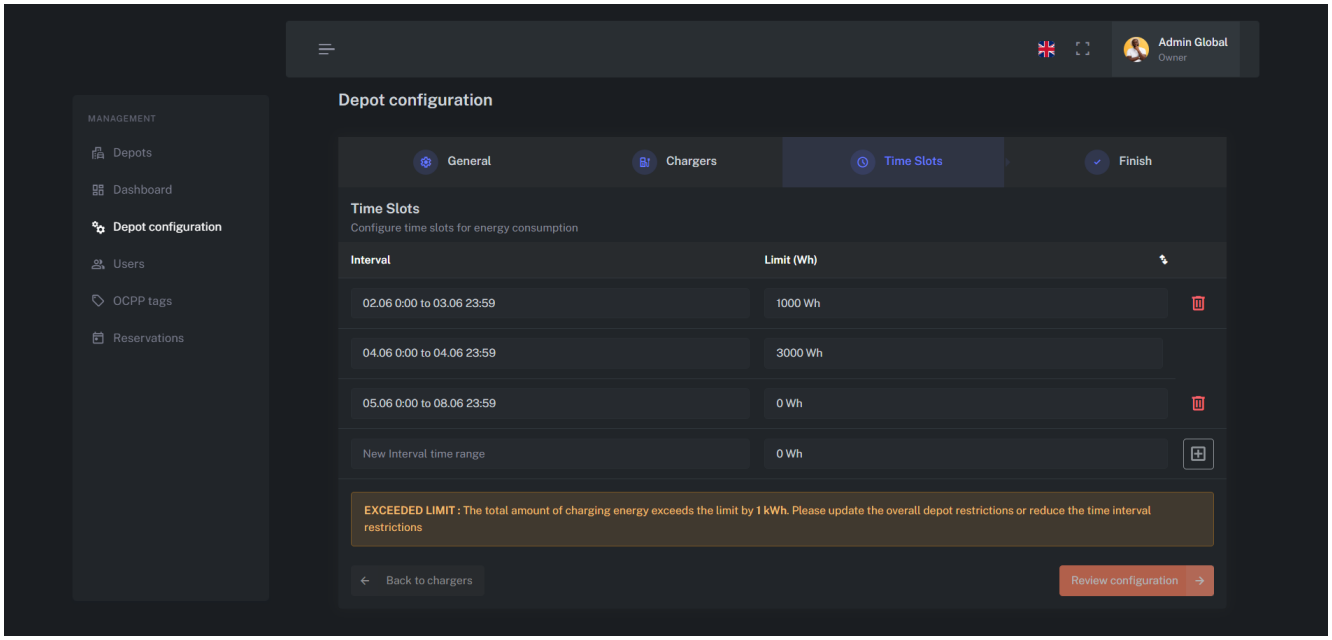


Рисунок 4.14 – Інтерфейс створення часових інтервалів (рисунок виконаний самостійно)

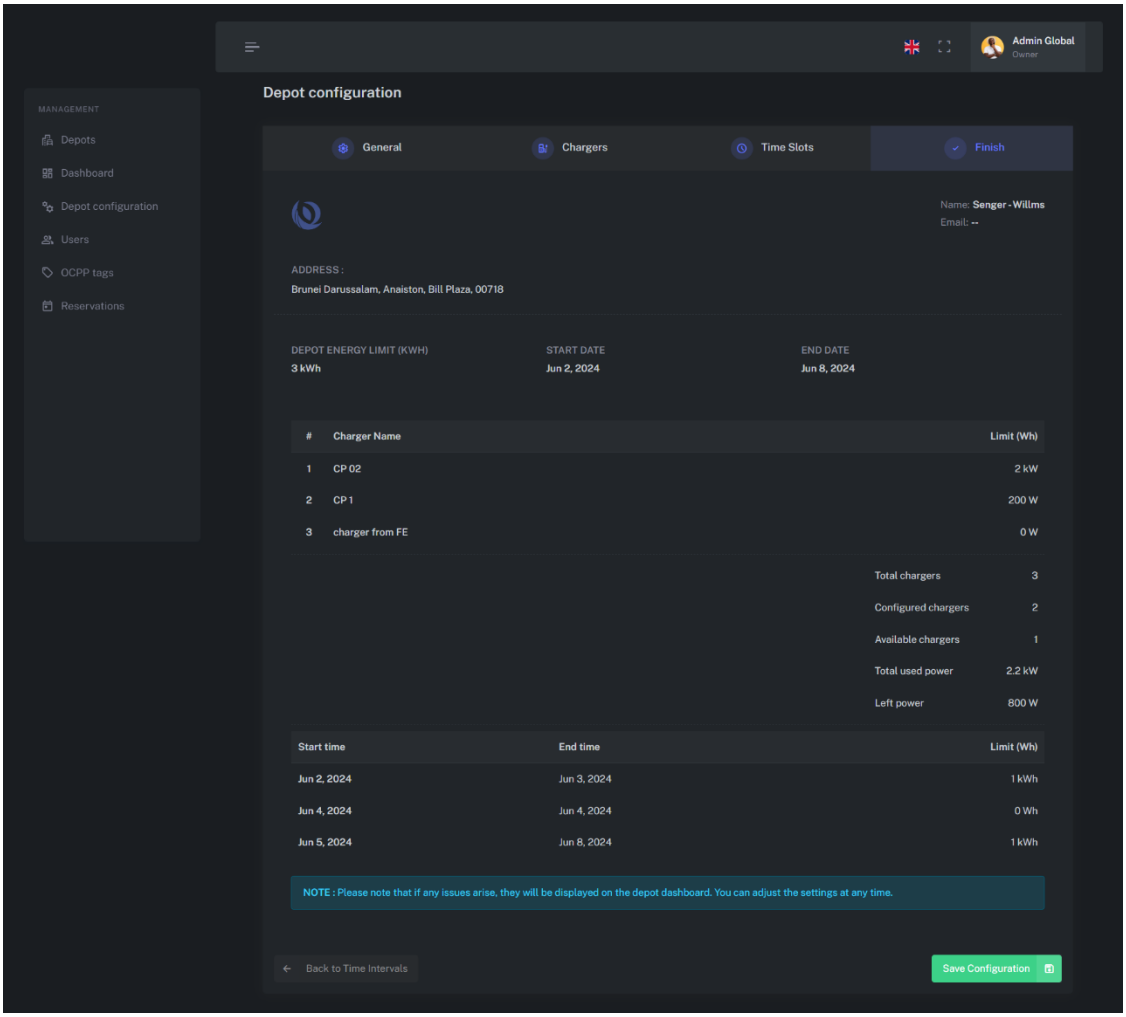


Рисунок 4.15 – Розподіл ресурсів енергії депо (рисунок виконаний самостійно)

Впровадження даної системи обмежень споживання енергії дозволяє оптимізувати використання енергетичних ресурсів депо, зменшити витрати на електроенергію та покращити загальну ефективність роботи системи. В комбінації з динамічними змінами SignalR користувач отримує легкий до взаємодії та зрозумілий інтерфейс.

4.5 Обробка табличних представлень

Табличні представлення у системі є важливим компонентом для відображення та управління даними.

Серед особливостей та застосованих технік оптимізації виділяються: Особливості табличних представлень

- використання Angular Pipes (див. рис. 4.16)дозволяє оптимізувати процес відображення даних шляхом мемоізації, що зменшує навантаження на систему і покращує продуктивність;
- механізм відслідковування змін trackBy ефективніше оновлювати DOM-елементи при змінах у масивах даних, мінімізуючи кількість циклів рендеру;
- пошук даних: всі таблиці мають функціональність пошуку (див. рис. 4.17), який здійснюється сервером, забезпечуючи швидке і ефективне знаходження потрібної інформації;
- фасади: Зв'язаність між таблицями і бізнес логікою реалізована за допомогою фасадів, що додає додатковий шар абстракції і полегшує підтримку коду;
- індикатор завантаження: використовуються для відображення поточного стану системи, покращуючи користувацький досвід при завантаженні великих об’ємів даних (див. рис. 4.18).

Рисунок 4.16 – Використання Angular Pipes для отримання переваг мемоізації даних (рисунок виконаний самостійно)

Рисунок 4.17 – Реалізація базових функцій для пошуку даних в табличних представленнях (рисунок виконаний самостійно)

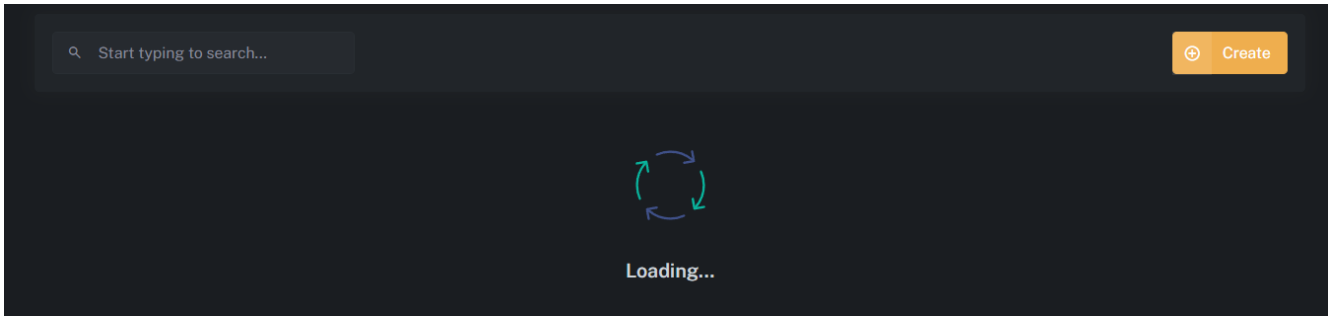


Рисунок 4.18 – Відображення анімованого індикатора завантаження даних (рисунок виконаний самостійно)

Реалізація табличних представлень з використанням вищеописаних технік забезпечує високу продуктивність та швидкий відгук системи на дії користувача, а мінімізація зв’язаності між бізенс логікою і представленнями за допомогою фасадів спрощує процес підтримки продукту.

4.6 Статистична інформація

Для аналізу даних роботи депо було реалізовано три типи графіків, що дозволяють користувачам отримати візуальне уявлення про різні аспекти споживання енергії та роботи системи.

Графік спожитоє енергії зарядними станціями представлений у вигляді «treemap». У цьому графіку кожен окремий блок відповідає зарядній станції, а розмір блоку відображає кількість спожитої енергії за вказаний проміжок часу (див рис. 4.19). Такий підхід дозволяє швидко оцінити, які станції споживають більше енергії і як розподіляється навантаження між різними станціями.

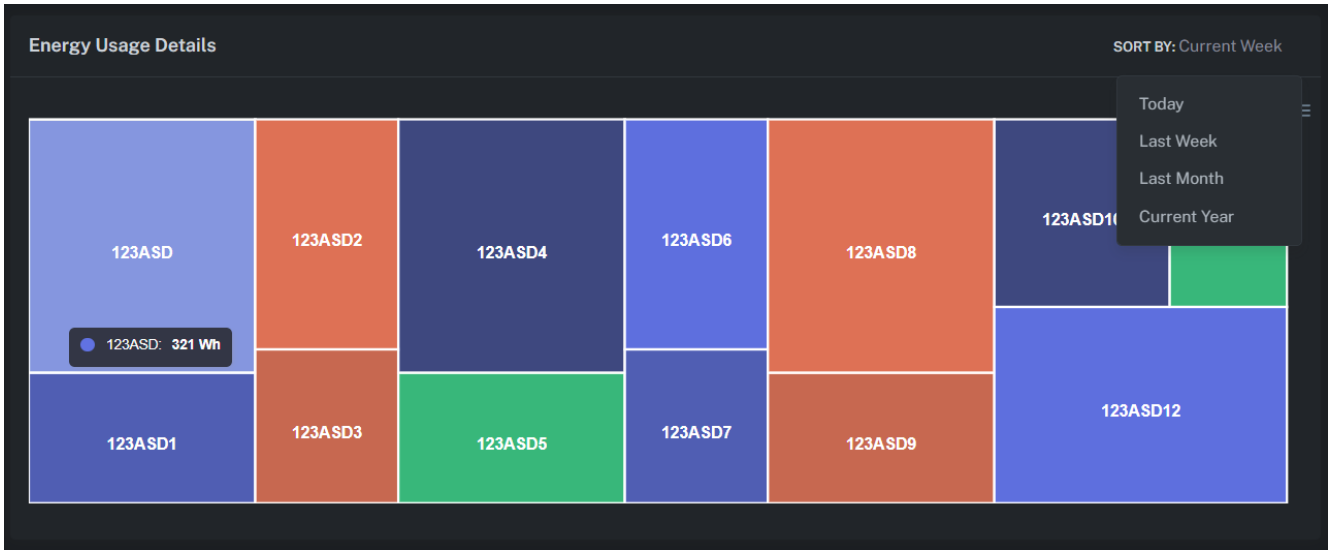


Рисунок 4.19 – Графічне відображення розподілу енергії між зарядними станціями депо (рисунок виконаний самостійно)

Другий тип графіка, представлений у вигляді лінійного графіку і демонструє загальну тенденцію потужності депо протягом певного періоду (див. рис. 4.20). Це дозволяє аналізувати зміни потужності у часі та виявляти пікові навантаження, що є важливим для ефективного управління енергоспоживанням.

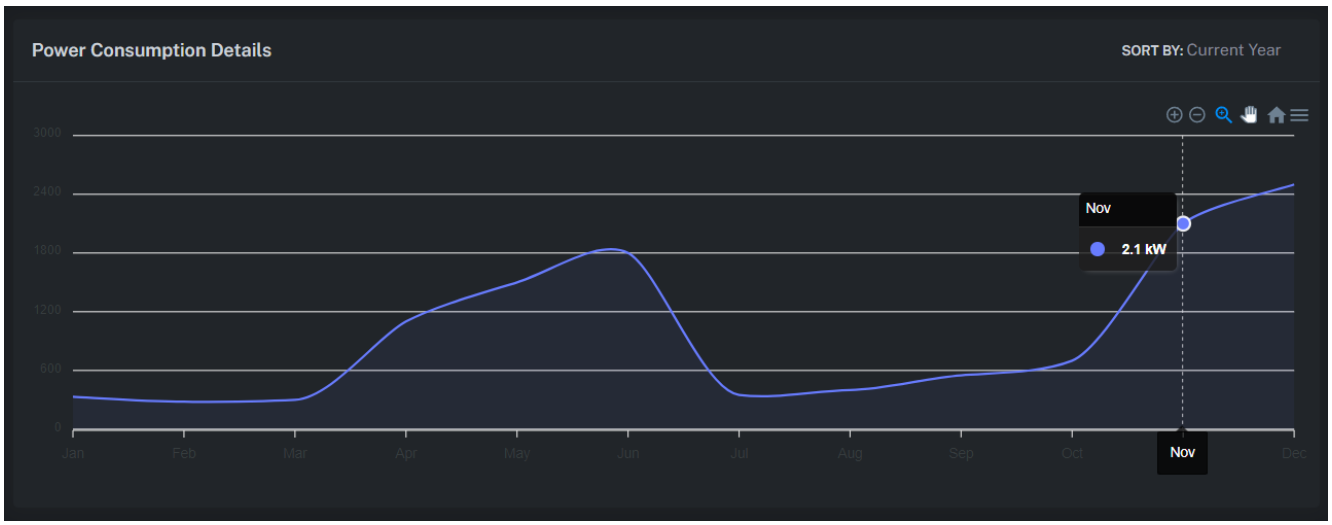


Рисунок 4.20 – Графічне відображення змін в потужності депо протягом року (рисунок виконаний самостійно)

Третій тип графіка, відображає залежність між запланованим споживанням енергії в заданий період і фактичним споживанням (див. рис. 4.21). Алгоритм налаштування запланованого споживання був детально описаний в підрозділі

"Обмеження споживання енергії депо". Цей графік дозволяє порівнювати заплановані показники з реальними, що допомагає виявляти відхилення і оптимізувати планування відповідно до потреб бізнесу.

Рисунок 4.21 – Графічне відображення різниці між запланованим споживанням енергії і фактичними значеннями (рисунок виконаний самостійно)

Усі графіки мають можливість експорту даних у формат CSV, SVG, що дозволяє користувачам легко зберігати і аналізувати дані в зовнішніх програмах.

Сформований CSV наведений на рисунку 4.22, а функцію генерації на рисунку 4.23.

	A	B
1	Charger Name	Energy Usage (Wh)
2	123ASD	321
3	123ASD1	165
4	123ASD2	184
5	123ASD3	123
6	123ASD4	321
7	123ASD5	165
8	123ASD6	184
9	123ASD7	123
10	123ASD8	321
11	123ASD9	165
12	123ASD10	184
13	123ASD11	123
14	123ASD12	321
15		

Рисунок 4.22 – CSV звіт розподілу енергії між зарядними станціями (рисунок виконаний самостійно)

```
private exportEnergyUsageCsv() : void  {  Show usages  new *
  this.getLocalizedHeaders( headerKeys: ['depot.config.chargers.charger-name', 'stats.energy-usage.title'])
      .pipe(
        take( count: 1),
        map( project: (headers : any[] ) : [A, any[][]]  => {
          const data : any[][]  = this.getChartAxiesData(this.EnergyUsageChart.series);

          return [headers, data];
        }),
        map( project: ([headers : any[] , data : any[] ]) => [headers, data] as [string[], string[][]])
      ) Observable<[string[], string[][]>
      .subscribe( observerOrNext: {
        next: (csvData : [string[], string[]] ) => this.downloadCsvContent(csvData)
      });
};

private getLocalizedHeaders(headerKeys: Array<string>) : Observable<any[]>  {  Show usages  new *
  return forkJoin(headerKeys.map(key : string  => this.translateService.get(key)));
}

private getChartAxiesData(series: ApexAxisChartSeries) : any[][]  {  Show usages  new *
  return series[0].data.map((row: any) => [row.x, row.y]);
}

private downloadCsvContent(content: [string[], string[][]]) : void  {  Show usages  new *
  const csvContent : string  = content.map(e : string[]  => e.join(',')).join('\n');
  const blob : Blob  = new Blob( blobParts: [csvContent],  options: { type: 'text/csv;charset=utf-8;' });
  const link : HTMLAnchorElement  = document.createElement( tagName: 'a');

  link.href = URL.createObjectURL(blob);
  link.setAttribute( qualifiedName: 'download',  value: `${this.getFileNameTimeRange()} .csv`);
  document.body.appendChild(link);
  link.click();
  document.body.removeChild(link);
};

private getFileNameTimeRange() : string  {  Show usages  new *
  const start : dayjs.Dayjs  = dayjs().startOf( unit: 'week');
  const end : dayjs.Dayjs  = dayjs().endOf( unit: 'week');

  return [start, end].map((date : dayjs.Dayjs ) => date.format( template: 'LLL')).join(' - ');
}
```

Рисунок 4.23 – Функція формування CSV звіту (рисунок виконаний самостійно)

Використання бібліотеки ApexCharts[12] для реалізації графіків в системі забезпечує високу якість візуалізації даних, зручність у використанні та широкі можливості для аналізу. Завдяки підтримці різних типів графіків, інтерактивності та налаштовуваності, користувачі отримують потужний інструмент для моніторингу і аналізу споживання енергії та інших показників роботи депо.

4.7 Локалізація програмного продукту

Для забезпечення багатомовної підтримки та динамічної зміни мови у додатку була використана бібліотека ngx-translate. Вона дозволяє зручно додавати нові мови, керувати перекладами та динамічно змінювати мову інтерфейсу без перезавантаження сторінки (див. рис. 4.24).

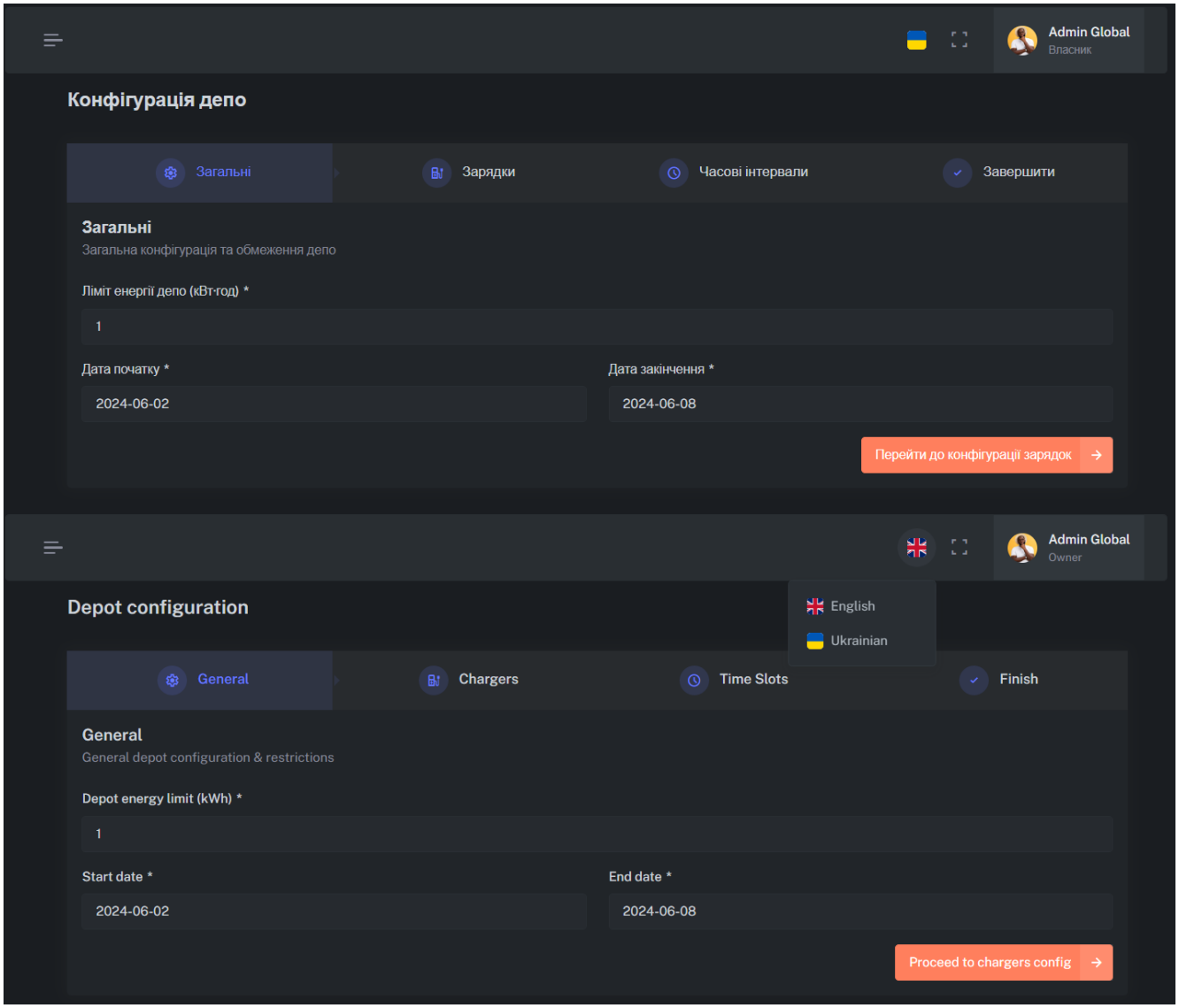


Рисунок 4.24 – Зміна мови відображення інтерфейсу додатку (рисунок виконаний самостійно)

Для роботи з датами використовується бібліотека dayjs, яка забезпечує легку і швидку обробку дат та часу. Для надання візуального представлення елементів вибору дати та часу була обрана бібліотека Flatpickr.

Використання даного набору інструментів забезпечує широку функціональність продукту, що забезпечує адаптивність системи під користувачів з різних регіонів.

4.8 Розгортання програмного продукту

У процесі впровадження програмного забезпечення було використано сучасні підходи, що забезпечують надійність, масштабованість та безперебійність роботи системи. Основні етапи впровадження включають розгортання на платформі Render[13] та використання інструменту DeepSource для контролю якості коду.

4.8.1 Платформа Render

Для розгортання програмного забезпечення було обрано платформу Render, яка забезпечує простий та зручний процес розгортання, а також можливості автоматичного масштабування та моніторингу (див. рис. 4.25). Render дозволяє легко керувати середовищем розгортання та забезпечує високу доступність сервісу. Серед основних переваг варто виділити:

- автоматичне розгортання: Render автоматично виконує розгортання нових версій додатка, зменшуючи час і зусилля, необхідні для ручного розгортання;
- HTTPS хостинг: Платформа забезпечує автоматичне отримання та оновлення SSL-сертифікатів, що гарантує безпечний HTTPS-зв'язок;
- CDN (Content Delivery Network): Використання CDN дозволяє прискорити доставку контенту користувачам, зменшуючи затримки та підвищуючи швидкість завантаження сторінок;
- моніторинг та логування: Render надає вбудовані інструменти для моніторингу стану додатка та збирання логів, що допомагає виявляти та вирішувати проблеми в реальному часі.

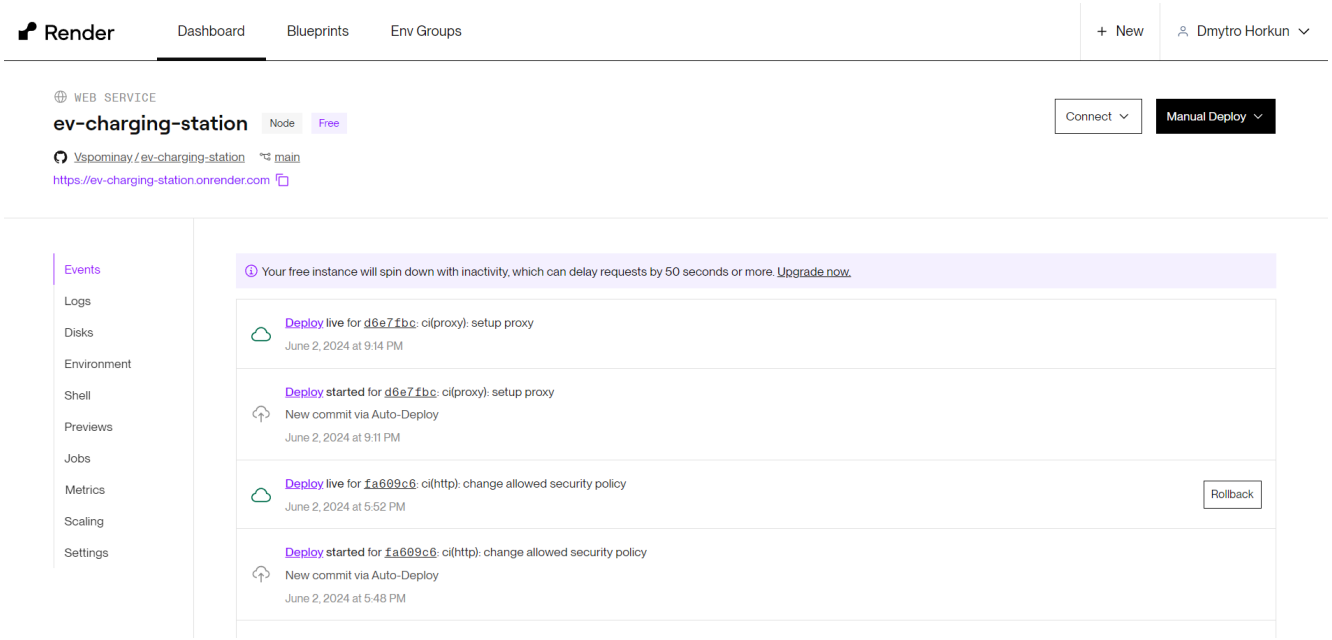


Рисунок 4.25 – Сторінка налаштувань розгорнутого додатку (рисунок виконаний самостійно)

Слід зазначити, що Render надає можливість безкоштовного розміщення сервісів, що, беззаперечно є перевагою для невеликого продукту.

4.8.2 Контроль якості коду

Для забезпечення високої якості коду в процесі розробки було використано інструмент DeepSource: JavaScript. DeepSource дозволяє автоматично виконувати статичний аналіз коду та виявляти потенційні проблеми ще на етапі розробки (див. рис. 4.26).

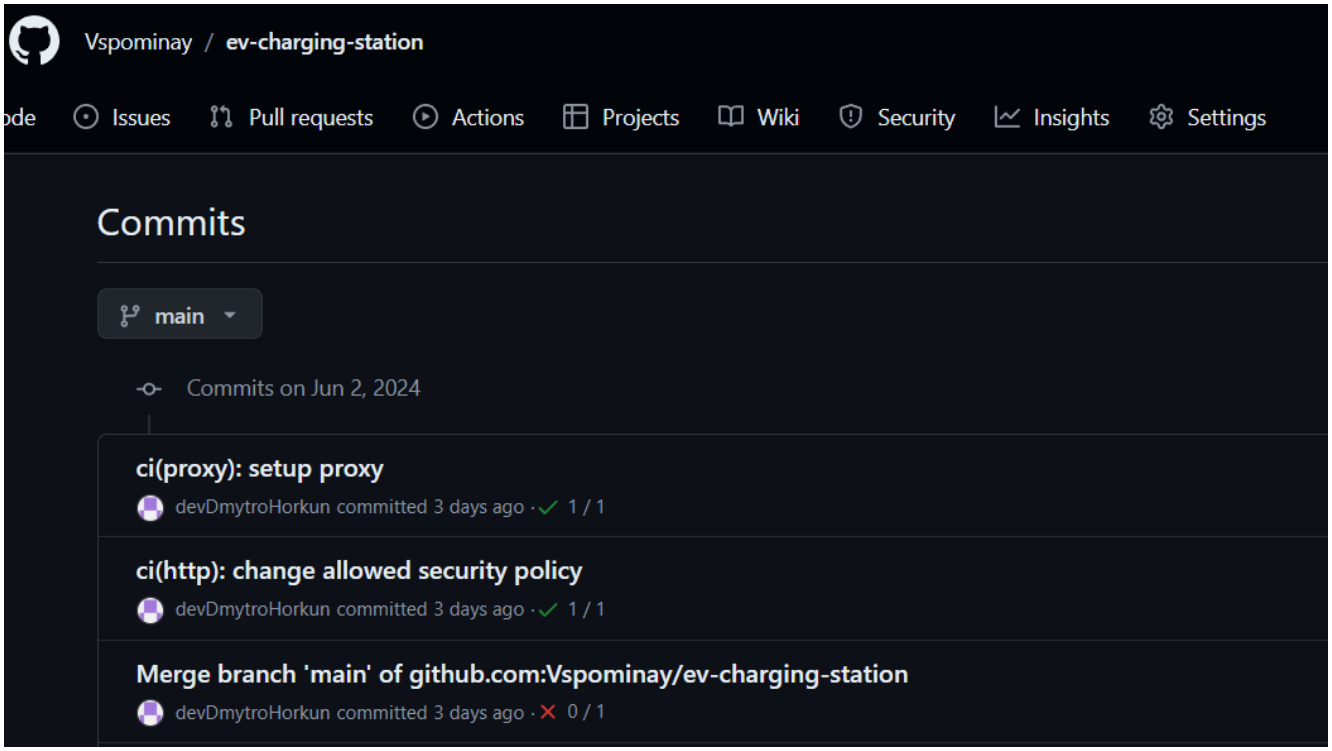


Рисунок 4.26 – Результати виконання перевірок якості коду (рисунок виконаний самостійно)

Отже, активна інтеграція з вищенаведеними інструментами для впровадження програмного забезпечення дозволила забезпечити високу якість та надійність системи, що відповідає вимогам сучасного бізнесу та користувачів.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У процесі розробки програмного забезпечення було активно застосоване мануальне тестування та написані юніт тести з використанням бібліотеки Jest для критичних частин системи. Розроблені юніт тести були описані в розділі з демонстрацією найцікавішого алгоритму.

У цьому розділі зосередимося на описі підходів до тестування та важливих тест-кейсах.

5.1 Формування вимог до якості продукту

На основі функціональних вимог до системи була підготована таблиця вимог до якості системи (див. табл. 5.1).

Таблиця 5.1 – Вимоги до якості системи моніторингу зарядних станцій (таблиця виконана самостійно)

Ідентифікатор	Вимога	Критерій прийняття	Реалізовано
QF-01	Система повинна підтримувати реєстрацію користувачів з ролями Супер Адміністратор, Адміністратор, Співробітник, Водій.	Реєстрація користувачів з різними ролями виконується коректно, кожен користувач отримує відповідні права доступу.	Так
QF-02	Система повинна підтримувати створення, редагування та видалення зарядних станцій.	Операції створення, видалення виконуються фізично, і система отримує відповідні повідомлення, в результаті яких оновлюється інтерфейс	Так
QF-03	Система повинна підтримувати бронювання зарядних станцій водіями.	Водії можуть бронювати зарядні станції через інтерфейс, всі бронювання відображаються коректно.	Так

Кінець таблиці 5.1

QF-04	Система повинна відправляти повідомлення в реальному часі про статус зарядки	Інтерфейс додатку і процес заряду станцій оновлюється в реальному часі по мірі	Так
QF-05	Система повинна відображати статистичну інформацію у вигляді графіків.	Статистична інформація коректно відображається у вигляді графіків, можливий експорт в CSV.	Так
QN-01	Система повинна підтримувати зміну мови інтерфейсу.	Користувачі можуть змінювати мову інтерфейсу без перезавантаження сторінки. Обов'язковими є українська та англійська мови.	Так
QN-02	Система повинна працювати з високою продуктивністю при великій кількості користувачів.	Система працює без значних затримок при одночасному використанні великою кількістю користувачів.	Не перевірено
QN-03	Система повинна бути безпечною і захищеною від несанкціонованого доступу.	Всі дані передаються через HTTPS, використовується JWT для аутентифікації.	Так
QN-04	Система повинна бути доступною з мобільних пристроїв.	Інтерфейс адаптивний і коректно відображається на мобільних пристроях. Функціонал необхідний водіям реалізовано в мобільному додатку.	Так

Дана таблиця допомагає зосередитися на ключових аспектах проекту та гарантує, що всі критичні функції будуть ретельно перевірені та запроваджені.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Одним із важливих аспектів розробки програмного забезпечення є публікація результатів роботи в наукових виданнях та конференціях. У рамках реалізації даної системи управління зарядними станціями електромобілів було здійснено кілька наукових публікацій, які висвітлюють різні аспекти проектування та впровадження системи.

Тези на тему проектування Angular додатку для програмної системи моніторингу зарядних станцій електромобілів з використанням Onion архітектури були представлені на V Міжнародній студентській науковій конференції «Розвиток суспільства та науки в умовах цифрової трансформації». У цих тезах розглядаються ключові аспекти проектування клієнтської частини системи, зокрема використання Onion архітектури для забезпечення модульності та зручності у підтримці коду. Було детально описано структуру додатку, підходи до реалізації основних функцій та забезпечення продуктивності системи.

Також, члени нашої команди, без моєї безпосередньої участі, публікували тези про проєктування серверної частини системи керування та моніторингу зарядними станціями на конференції «Інформаційні інтелектуальні системи» XXVIII Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті». Ці тези висвітлюють аспекти проектування серверної частини системи, зокрема використання сучасних технологій та підходів для забезпечення надійності, масштабованості та безпеки системи.

Публікації, здійснені в рамках даної роботи, не лише підтверджують важливість та актуальність розробленої системи, але й сприяють поширенню знань і досвіду серед наукової спільноти. Вони демонструють успішність застосованих підходів та технологій, а також надають можливість іншим дослідникам використовувати отримані результати у своїх проектах.

ВИСНОВКИ

У процесі роботи було здійснено комплексну реалізацію клієнтську частину системи управління зарядними станціями електромобілів, що включає функціональні та нефункціональні складові. Вивчення ринку, статистичних даних та потреб користувачів дозволило визначити основні проблеми та сформувати чіткі вимоги до розроблюваної системи. Після ретельного проектування та реалізації програмного забезпечення, система була успішно розгорнута і протестована використовуючи TDD підходи, демонструючи високу ефективність та відповідність поставленим вимогам.

Реалізація основних функцій охоплює управління зарядними станціями, резервацію, моніторинг та аналітику, обмеження споживання енергії, роботу з таблицями та локалізацію. Зокрема, був створений зручний інтерфейс для налаштування депо, який включає можливості як введення загальних так і точкових обмежень, а також систему моніторингу для відстеження стану зарядних станцій у реальному часі. Для покращення користувацького досвіду впроваджено інтерактивний календар для бронювання зарядних станцій та використання Ionic 8 для мобільної версії додатку, що робить його доступним та зручним для водіїв.

В аналітичній частині реалізовано різні графічні представлення даних для аналізу роботи депо, що дозволяє здійснювати ефективний моніторинг та приймати обґрунтовані рішення. Окрім того, була впроваджена можливість експорту даних у форматі CSV. Важливим механізмом є можливість обмеження споживання енергії депо, що дозволяє налаштовувати споживання у певних часових рамках, оптимізуючи роботу системи.

В процесі реалізації були застосовані кращі практики оптимізації, проектування архітектури системи та контролю якості коду.

Локалізація була реалізована з динамічною зміною мови за допомогою бібліотеки ngx-translate, а також використанням бібліотеки dayjs для роботи з датами та Flatpicker для візуального представлення компонентів вибору дати.

Розроблене програмне забезпечення було розгорнуто на платформі Render, що забезпечило автоматичне розгортання, HTTPS хостинг, CDN та логування, спрощуючи управління та підтримку системи. Для контролю якості коду використовувався інструмент DeepSource, що виконує перевірки для оновлень в репозиторії, підтримуючи високу якість коду.

Поставлені завдання виконано в повному обсязі, а отримані теоретичні та практичні результати дозволили сформувати підґрунтя для подальших досліджень в доменній області, вже з використанням інтелектуальних алгоритмів керування та прогнозування навантаження на електромережі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Global EV outlook 2023 – analysis - IEA. IEA. URL: https://www.iea.org/reports/global-ev-outlook-2023 (дата звернення: 05.05.2024).

2. GreenFlux | fortum charge & drive. Fortum Charge & Drive. URL: https://chargedrive.com/greenflux (дата звернення: 05.05.2024).

3. All-in-one EV charging software - AMPECO. AMPECO. URL: https://www.ampeco.com/ (дата звернення: 05.05.2024).

4. Electric vehicle charging management software solution | driivz. Driivz. URL: https://driivz.com/ (дата звернення: 05.05.2024).

5. ChargeLab: Electric vehicle charging software. ChargeLab: Electric vehicle charging software. URL: https://chargelab.co/ (дата звернення: 05.05.2024).

6. News - open charge alliance. Open Charge Alliance. URL: https://openchargealliance.org/my-oca/ocpp/ (дата звернення: 05.05.2024).

7. Dubetcky O. ISO/IEC 27001:2022. Система управління інформаційною безпекою. Medium. URL: https://oleg-dubetcky.medium.com/iso-iec-27001-2022-система-управління-інформаційною-безпекою-db24615a276d (дата звернення: 05.05.2024).

8. Горкун Д., Налєскіна Т. Проектування angular додатку для програмної системи моніторингу зарядних станцій електромобілей використовуючи onion архітектур. Розвиток суспільства та науки в умовах цифрової трансформації : Міжнар. студент. наук. конф., м. Умань, 2 трав. 2024 р. Умань, 2024. С. 60–62. URL: https://archive.liga.science/plugins/generic/pdfJsViewer/pdf.js/web/viewer.html?file=https://archive.liga.science/index.php/conference-proceedings/issue/download/inter-02.02.2024/62#[%7B"num":137,"gen":0%7D,%7B"name":"FitR"%7D,-193,362,789,842] (дата звернення: 05.05.2024).

9. Contributors to Wikimedia projects. Test-driven development - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Test-driven_development (дата звернення: 05.05.2024).

10. Nirmal V. Implementing FullCalendar in Angular: A Step-by-Step Guide. Medium. URL: https://vaibhavnirmal2001.medium.com/implementing-fullcalendar-in-angular-a-step-by-step-guide-e5caf3e89644 (дата звернення: 01.06.2024).

11. DeBeasi L. Ionic 8 is here! - Ionic Blog. Ionic. URL: https://ionic.io/blog/ionic-8-is-here (дата звернення: 05.06.2024).

12. Darshan theerth. Building charts with ApexCharts and Angular. Medium. URL: https://medium.com/@darshantheerth/building-charts-with-apexcharts-and-angular-db58475fae2b (дата звернення: 05.06.2024).

13. Alkhateeb M. How To Deploy Angular Server-Side Rendering App To Cpanel (NameCheap). Medium. URL: https://medium.com/@malkhateeb2022/how-to-deploy-angular-server-side-rendering-app-to-cpanel-namecheap-49c62cd0e89e (дата звернення: 02.06.2024

ДОДАТОК А

Canva бізнес-модель

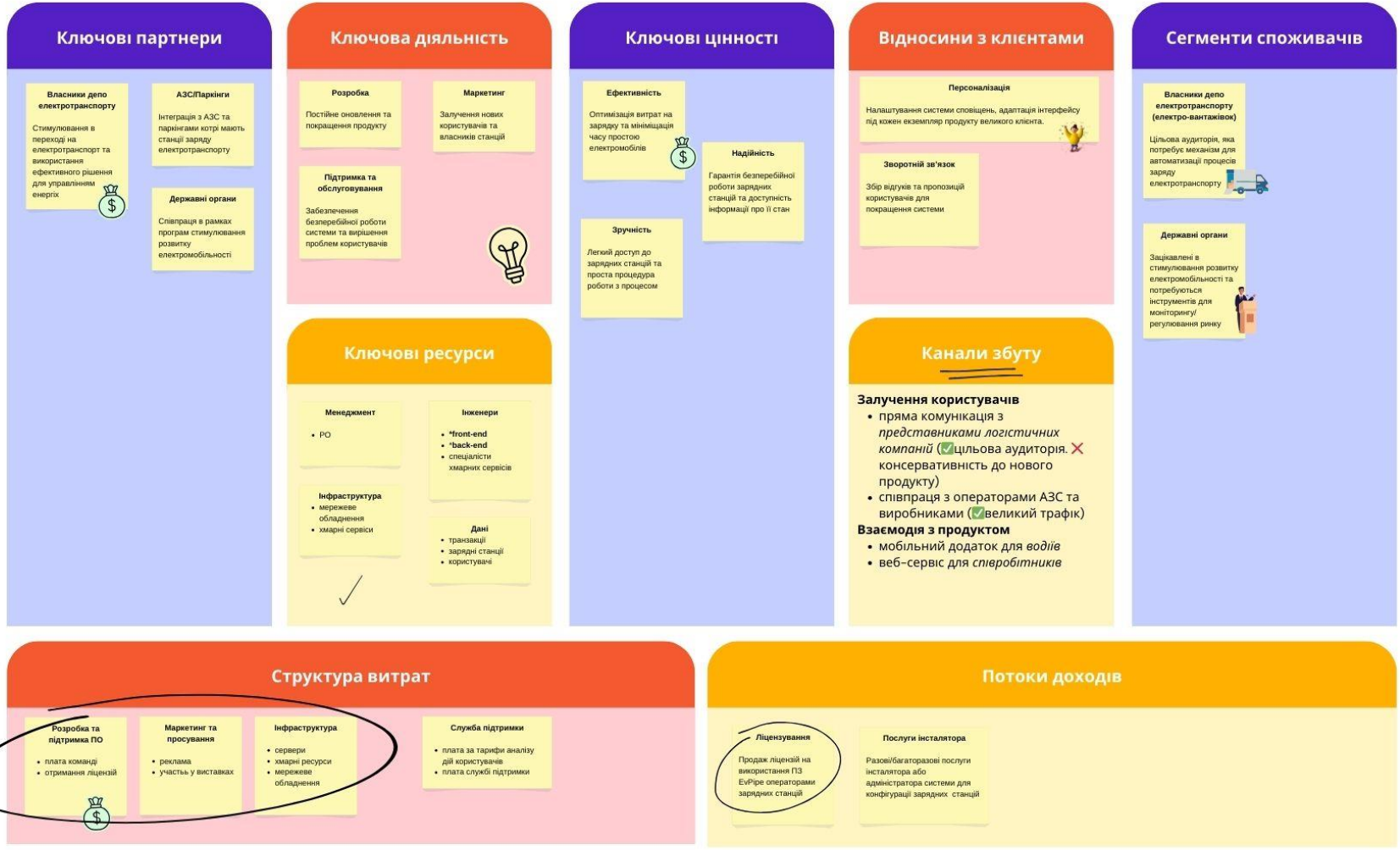


Рисунок А.1 – Бізнес-модель Canva сервісу по моніторингу зарядних станцій (рисунок виконано самостійно)

ДОДАТОК Б

Тести алгоритму генерації часових інтервалів

```
describe('TimeRangesBuilderService', () => {
  let service: TimeRangesBuilderService;
  const readableFormat = 'YYYY-MM-DD HH:mm:ss';

  const buildTimeRange = (startTime: string, finishTime:
string, power: number) => ({
    startTime: dayjs(startTime, readableFormat),
    finishTime: dayjs(finishTime, readableFormat),
    power
  });

  const expectTimeRange = (actual: TTimeRange, expected:
TTimeRange) => {

expect(actual.startTime.format()).toBe(expected.startTime.forma
t());

expect(actual.finishTime.format()).toBe(expected.finishTime.for
mat());
    expect(actual.power).toBe(expected.power);
  };

  const expectIntervals = (actual: Array<TTimeRange>, expected:
Array<TTimeRange>) => {
    expect(actual.length).toBe(expected.length);

    expected.forEach((expected, index) => {
      expectTimeRange(actual[index], expected);
    });
  };

  const defaultPower = 100;
  const depotRestrictions = buildTimeRange('2024-01-01
00:00:00', '2024-01-10 23:59:59', defaultPower);

  beforeEach(() => {
    service = new TimeRangesBuilderService();
  });

  it('creates successfully', () => {
    expect(service).toBeTruthy();
  });
```

```
  it('returns depot restrictions when no time ranges', () => {
    const result = service.processTimeRanges([],
depotRestrictions);

    expectIntervals(result, [depotRestrictions]);
  });

  it('process inserting time range at the middle', () => {
    const existingRange = { ...depotRestrictions };
    const newRange = buildTimeRange('2024-01-02 00:00:00',
'2024-01-03 00:00:00', 50);

    const result = service.processTimeRanges([existingRange,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-01
23:59:59', defaultPower),
      buildTimeRange('2024-01-02 00:00:00', '2024-01-03
00:00:00', 50),
      buildTimeRange('2024-01-03 00:00:01', '2024-01-10
23:59:59', defaultPower),
    ];

    expectIntervals(result, expectedRanges);
  });

  it('process inserting time range at the end', () => {
    const existingRange = { ...depotRestrictions };
    const newRange = buildTimeRange('2024-01-10 00:00:00',
'2024-01-11 00:00:00', 50);

    const result = service.processTimeRanges([existingRange,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-09
23:59:59', defaultPower),
      buildTimeRange('2024-01-10 00:00:00', '2024-01-10
23:59:59', 50),
    ];

    expectIntervals(result, expectedRanges);
  });

  it('process inserting time range at the start', () => {
    const existingRange = { ...depotRestrictions };
```

```
    const newRange = buildTimeRange('2023-12-31 00:00:00',
'2024-01-02 00:00:00', 50);

    const result = service.processTimeRanges([existingRange,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
00:00:00', 50),
      buildTimeRange('2024-01-02 00:00:01', '2024-01-10
23:59:59', defaultPower),
    ];

    expectIntervals(result, expectedRanges);
  });

  it('process inserting time range at the middle with full 1
interval overlapping', () => {
    const timeRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
00:00:00', 50),
      buildTimeRange('2024-01-02 00:00:01', '2024-01-03
00:00:00', 75),
      buildTimeRange('2024-01-03 00:00:01', '2024-01-10
23:59:59', 0),
    ];
    const newRange = buildTimeRange('2024-01-01 14:00:00',
'2024-01-04 14:00:00', 66);

    const result = service.processTimeRanges([...timeRanges,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-01
13:59:59', 50),
      buildTimeRange('2024-01-01 14:00:00', '2024-01-04
14:00:00', 66),
      buildTimeRange('2024-01-04 14:00:01', '2024-01-10
23:59:59', 0),
    ];

    expectIntervals(result, expectedRanges);
  });

  it('process inserting time range at the middle with full
partial overlapping', () => {
    const timeRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
```

```
00:00:00', 50),
      buildTimeRange('2024-01-02 00:00:01', '2024-01-03
00:00:00', 75),
      buildTimeRange('2024-01-03 00:00:01', '2024-01-10
23:59:59', 0),
    ];
    const newRange = buildTimeRange('2024-01-01 14:00:00',
'2024-01-02 14:00:00', 66);

    const result = service.processTimeRanges([...timeRanges,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-01
13:59:59', 50),
      buildTimeRange('2024-01-01 14:00:00', '2024-01-02
14:00:00', 66),
      buildTimeRange('2024-01-02 14:00:01', '2024-01-03
00:00:00', 75),
      buildTimeRange('2024-01-03 00:00:01', '2024-01-10
23:59:59', 0),
    ];

    expectIntervals(result, expectedRanges);
  });

  it('process inserting time range at the middle fully
overlapping 2 intervals', () => {
    const timeRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
00:00:00', 50), // partial overlap
      buildTimeRange('2024-01-02 00:00:01', '2024-01-03
00:00:00', 75), // full overlap
      buildTimeRange('2024-01-03 00:00:01', '2024-01-03
15:00:00', 0), // full overlap
      buildTimeRange('2024-01-03 15:00:01', '2024-01-04
23:59:59', 10), // partial overlap
      buildTimeRange('2024-01-05 00:00:00', '2024-01-10
23:59:59', 0),
    ];
    const newRange = buildTimeRange('2024-01-01 14:00:00',
'2024-01-04 14:00:00', 66);

    const result = service.processTimeRanges([...timeRanges,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-01
```

```
13:59:59', 50),
      buildTimeRange('2024-01-01 14:00:00', '2024-01-04
14:00:00', 66),
      buildTimeRange('2024-01-04 14:00:01', '2024-01-04
23:59:59', 10),
      buildTimeRange('2024-01-05 00:00:00', '2024-01-10
23:59:59', 0),
    ];

    expectIntervals(result, expectedRanges);
  });

  it('process inserting splitting the 1st time range', () => {
    const timeRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
00:00:00', 50), // partial
      buildTimeRange('2024-01-02 00:00:01', '2024-01-03
00:00:00', 75),
      buildTimeRange('2024-01-03 00:00:01', '2024-01-10
23:59:59', 0),
    ];
    const newRange = buildTimeRange('2024-01-01 14:00:00',
'2024-01-01 20:00:00', 66);

    const result = service.processTimeRanges([...timeRanges,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-01
13:59:59', 50), // 1st part
      buildTimeRange('2024-01-01 14:00:00', '2024-01-01
20:00:00', 66), // new interval
      buildTimeRange('2024-01-01 20:00:01', '2024-01-02
00:00:00', 50), // 2nd part
      buildTimeRange('2024-01-02 00:00:01', '2024-01-03
00:00:00', 75),
      buildTimeRange('2024-01-03 00:00:01', '2024-01-10
23:59:59', 0),
    ];

    expectIntervals(result, expectedRanges);
  });

  it('process inserting splitting the last time range', () => {
    const timeRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
00:00:00', 50),
```

```
      buildTimeRange('2024-01-02 00:00:01', '2024-01-03
00:00:00', 75),
      buildTimeRange('2024-01-03 00:00:01', '2024-01-10
23:59:59', 0), // partial
    ];
    const newRange = buildTimeRange('2024-01-03 14:00:00',
'2024-01-04 20:00:00', 66);

    const result = service.processTimeRanges([...timeRanges,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
00:00:00', 50),
      buildTimeRange('2024-01-02 00:00:01', '2024-01-03
00:00:00', 75),
      buildTimeRange('2024-01-03 00:00:01', '2024-01-03
13:59:59', 0), // 1st part
      buildTimeRange('2024-01-03 14:00:00', '2024-01-04
20:00:00', 66), // new interval
      buildTimeRange('2024-01-04 20:00:01', '2024-01-10
23:59:59', 0), // 2nd part
    ];

    expectIntervals(result, expectedRanges);
  });

  it('process inserting at the middle with the start value as
the previous interval finish value', () => {
    const timeRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
00:00:00', 50), // overlap finish
      buildTimeRange('2024-01-02 00:00:01', '2024-01-03
00:00:00', 75), // overlap start
      buildTimeRange('2024-01-03 00:00:01', '2024-01-10
23:59:59', 0),
      buildTimeRange('2024-01-03 00:00:00', '2024-01-04
00:00:00', 100),
    ];
    const newRange = buildTimeRange('2024-01-02 00:00:00',
'2024-01-02 14:00:00', 66);

    const result = service.processTimeRanges([...timeRanges,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-01
23:59:59', 50), // cut 1s finish
```

```
      buildTimeRange('2024-01-02 00:00:00', '2024-01-02
14:00:00', 66), // new interval
      buildTimeRange('2024-01-02 14:00:01', '2024-01-03
00:00:00', 75), // cut 2s start
      buildTimeRange('2024-01-03 00:00:01', '2024-01-10
23:59:59', 0),
    ];

    expectIntervals(result, expectedRanges);
  });

  it('process inserting at the middle with the finish value as
the next interval start value', () => {
    const timeRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
00:00:00', 50),
      buildTimeRange('2024-01-02 00:00:01', '2024-01-03
00:00:00', 75), // partial overlap
      buildTimeRange('2024-01-03 00:00:01', '2024-01-05
23:59:59', 10), // partial overlap
      buildTimeRange('2024-01-06 00:00:00', '2024-01-10
23:59:59', 0),
    ];
    const newRange = buildTimeRange('2024-01-02 14:00:00',
'2024-01-03 00:00:01', 66);

    const result = service.processTimeRanges([...timeRanges,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
00:00:00', 50),
      buildTimeRange('2024-01-02 00:00:01', '2024-01-02
13:59:59', 75), // 1st part
      buildTimeRange('2024-01-02 14:00:00', '2024-01-03
00:00:01', 66),
      buildTimeRange('2024-01-03 00:00:02', '2024-01-05
23:59:59', 10), // 2nd part
      buildTimeRange('2024-01-06 00:00:00', '2024-01-10
23:59:59', 0),
    ];

    expectIntervals(result, expectedRanges);
  });

  it('process same power merging', () => {
    const timeRanges = [
```

```
      buildTimeRange('2024-01-01 00:00:00', '2024-01-02
00:00:00', 50),
      buildTimeRange('2024-01-02 00:00:01', '2024-01-03
00:00:00', 10),
      buildTimeRange('2024-01-03 00:00:01', '2024-01-10
23:59:59', 50),
    ];
    const newRange = buildTimeRange('2024-01-00 14:00:00',
'2024-01-05 00:00:01', 50);

    const result = service.processTimeRanges([...timeRanges,
newRange], depotRestrictions);
    const expectedRanges = [
      buildTimeRange('2024-01-01 00:00:00', '2024-01-10
23:59:59', 50),
    ];

    expectIntervals(result, expectedRanges);
  });
});
```

ДОДАТОК В

Сервіс для управління часовими інтервалами

```
@Injectable({
  providedIn: 'root'
})
export class TimeRangesBuilderService {
  processTimeRanges(
    intervals: TTimeRange[],
    depotConfig: TTimeRange
  ): TTimeRange[] {
    const isEmpty = intervals.length === 0;
    if (isEmpty) {
      return [depotConfig];
    }

    const ranges = intervals.map((range) => {
      return {
        startTime:
this.updateTimeAccordingDepotConfig(range.startTime,
depotConfig),
        finishTime:
this.updateTimeAccordingDepotConfig(range.finishTime,
depotConfig),
        power: numberAttribute(range.power, 0),
      };
    });

    logReadableIntervals(ranges);

    const fixedOverlaps = this.fixOverlappingIntervals(ranges,
depotConfig);
    const filledTimeRanges = this.fillGaps(fixedOverlaps,
depotConfig);
    const mergedIntervals =
this.mergeAdjacentIntervals(filledTimeRanges, depotConfig);

    return mergedIntervals;
  }

  private updateTimeAccordingDepotConfig(
    time: dayjs.Dayjs,
    depotConfig: TTimeRange
  ): dayjs.Dayjs {
    if (time.isBefore(depotConfig.startTime)) {
```

```
      return depotConfig.startTime.clone();
    } else if (time.isAfter(depotConfig.finishTime)) {
      return depotConfig.finishTime.clone();
    } else {
      return time.clone();
    }
  }

  private fixOverlappingIntervals(
    intervals: TTimeRange[],
    depotConfig: TTimeRange
  ): TTimeRange[] {
    const highestPriorityInterval = intervals[intervals.length
- 1];
    const otherIntervals = intervals.slice(0, intervals.length
- 1);

    const fixedIntervals: TTimeRange[] = [];

    for (const range of otherIntervals) {
      // Check if current range overlaps with the
highestPriorityInterval and split if necessary
      if (

range.startTime.isBefore(highestPriorityInterval.finishTime) &&
        isSameOrAfter(range.finishTime,
highestPriorityInterval.startTime)
      ) {
        if
(range.startTime.isBefore(highestPriorityInterval.startTime)) {
          fixedIntervals.push({
            startTime: range.startTime,
            finishTime:
this.updateTimeAccordingDepotConfig(highestPriorityInterval.sta
rtTime.clone().subtract(1, 'seconds'), depotConfig),
            power: Number(range.power)
          });
        }

        if
(range.finishTime.isAfter(highestPriorityInterval.finishTime))
{
          fixedIntervals.push({
            startTime:
this.updateTimeAccordingDepotConfig(highestPriorityInterval.fin
ishTime.clone().add(1, 'seconds'), depotConfig),
```

```
            finishTime: range.finishTime.clone(),
            power: Number(range.power)
          });
        }
      } else {
        fixedIntervals.push(range);
      }
    }

    // Add the highestPriorityInterval
    fixedIntervals.push(highestPriorityInterval);

    // Sort the intervals by startTime
    fixedIntervals.sort((a, b) => {
      if (a.startTime.isBefore(b.startTime)) {
        return -1;
      } else if (a.startTime.isAfter(b.startTime)) {
        return 1;
      } else {
        return 0;
      }
    });

    // Fix adjacent intervals where end of first interval is
the start of the next interval
    for (let i = 0; i < fixedIntervals.length - 1; i++) {
      if (
        fixedIntervals[i].finishTime.isSame(
          fixedIntervals[i + 1].startTime,
          'seconds'
        )
      ) {
        fixedIntervals[i + 1].startTime =
this.updateTimeAccordingDepotConfig(fixedIntervals[i +
1].startTime

.clone()

.add(1, 'seconds'), depotConfig);
      }
    }

    return fixedIntervals;
  }

  private fillGaps(
```

```
    intervals: TTimeRange[],
    depotConfig: TTimeRange
  ): TTimeRange[] {
    const filledTimeRanges: TTimeRange[] = [];

    const startTime = depotConfig.startTime.clone();
    const endTime = depotConfig.finishTime.clone();

    const defaultPower = depotConfig.power;
    let previousFinishTime = depotConfig.startTime.clone();

    for (const range of intervals) {
      if (range.startTime.isAfter(previousFinishTime)) {
        filledTimeRanges.push({
          startTime: previousFinishTime.isSame(startTime,
'seconds')
            ? previousFinishTime
            :
this.updateTimeAccordingDepotConfig(previousFinishTime.add(1,
'seconds'), depotConfig),
          finishTime:
this.updateTimeAccordingDepotConfig(range.startTime.clone().sub
tract(1, 'seconds'), depotConfig),
          power: defaultPower
        });
      }
      filledTimeRanges.push(range);
      previousFinishTime = range.finishTime.clone();
    }

    if (previousFinishTime.isBefore(endTime)) {
      filledTimeRanges.push({
        startTime: previousFinishTime.isSame(startTime,
'seconds')
          ? previousFinishTime
          :
this.updateTimeAccordingDepotConfig(previousFinishTime.add(1,
'seconds'), depotConfig),
        finishTime: endTime,
        power: defaultPower
      });
    }

    return filledTimeRanges.filter(({ startTime, finishTime })
=>
      finishTime.isAfter(startTime)
```

```
    );
  }

  private mergeAdjacentIntervals(
    intervals: TTimeRange[],
    depotConfig: TTimeRange
  ): TTimeRange[] {
    const mergedTimeRanges: TTimeRange[] = [];
    let currentRange: TTimeRange | null = null;

    for (const range of intervals) {
      if (!currentRange) {
        currentRange = range;
        continue;
      }

      const isAdjacentAndSamePower =
        isSameOrAfter(currentRange.finishTime
                                  .clone()
                                  .add(1, 'seconds'),
range.startTime, 'seconds') &&
        Number(currentRange.power) === Number(range.power);

      if (isAdjacentAndSamePower) {
        currentRange.finishTime =
this.updateTimeAccordingDepotConfig(dayjs.max(
          currentRange.finishTime,
          range.finishTime
        )!, depotConfig);
      } else {
        if (
          isSameOrAfter(currentRange.finishTime,
range.startTime, 'seconds') &&
          currentRange.power !== range.power
        ) {
          currentRange.finishTime =
this.updateTimeAccordingDepotConfig(range.startTime.clone().sub
tract(1, 'seconds'), depotConfig);
        } else {
          mergedTimeRanges.push(currentRange);
        }
        currentRange = range;
      }
    }

    if (currentRange) {
```

```
      mergedTimeRanges.push(currentRange);
    }

    return mergedTimeRanges;
  }
}

export const isSameOrAfter = (a: dayjs.Dayjs, b: dayjs.Dayjs,
unit?: dayjs.OpUnitType): boolean => {
  return a.isSame(b, unit) || a.isAfter(b, unit);
};
```

ДОДАТОК Г

Тези доповіді

V Міжнародна студентська наукова конференція «РОЗВИТОК СУСПІЛЬСТВА ТА НАУКИ В УМОВАХ ЦИФРОВОЇ ТРАНСФОРМАЦІЇ»

Горкун Дмитро Олександрович, здобувач вищої освіти факультету комп'ютерних наук
Харківський національний університет радіоелектроніки, Україна

Налєскіна Тетяна Сергіївна, здобувач вищої освіти факультету комп'ютерних наук
Харківський національний університет радіоелектроніки, Україна

Науковий керівник: Русакова Наталія Євгенівна, доцент кафедри програмної інженерії
Харківський національний університет радіоелектроніки, Україна

ПРОЕКТУВАННЯ ANGULAR ДОДАТКУ ДЛЯ ПРОГРАМНОЇ СИСТЕМИ МОНІТОРИНГУ ЗАРЯДНИХ СТАНЦІЙ ЕЛЕКТРОМОБІЛЕЙ ВИКОРИСТОВУЮЧИ ONION АРХІТЕКТУРУ

Розробка сучасних веб-додатків вимагає використання гнучких та масштабованих підходів до архітектури програмного забезпечення. Це особливо важливо для комплексних бізнес-систем, таких як управління автопарками електромобілів. Ефективне споживання електроенергії для українських бізнесів пов'язаних з електротранспортом є важливим, тож зручний додаток, основним завданням якого є автоматизація процесу управління споживання електроенергії, повинен надавати максимально швидку, стійку та інтуїтивно зрозумілу клієнтську частину. Подібні проєкти містять багато бізнес-логіки, інтеграцій з зовнішніми сервісами, користувацьких інтерфейсів та іншої інфраструктури. Щоб ефективно керувати такою складністю, потрібні структуровані архітектурні рішення.

Onion архітектура[1] далеко не інноваційний підхід, котрий використовується на серверній стороні додатків, що не можна сказати про клієнтську частину. Вона поділяє додаток на послідовні рівні відповідальності. Головною перевагою такої організації є слабке зв'язування між компонентами системи та їх висока змінюваність.

Слід відмітити, що Angular та інші сучасні фреймворки підтримують асинхронний код. Що робить цю архітектуру ще цікавішою для реалізації!

Метою даної архітектури є забезпечення розподілу обов'язків і збереження логіки домену ізольованою.

Найбільш критичними частинами Angular-проєкту є:

– наявність легких та чистих компонентів;
– управління станом системи;
– уникнення розповсюдження бізнес-логіки додатку між компонентами та сервісами.

Тепер, визначімо, що є нашим доменом і що потрібно від нього відокремити. У Frontend-проєктах можна виділити чотири великі частини:

– представлення, що відображаються користувачеві, тобто компоненти;
– клієнти, які використовуються для спілкування з API;
– стан, наше єдине джерело істини;
– домен, який містить бізнес-логіку.

Уявімо, що користувачеві доступний список депо. Коли користувач видаляє

Рисунок Г.1 – Перша сторінка доповіді зі збірника (рисунок виконано самостійно)

елемент, виконується оркестрація видалення через клієнтський шлюз, а потім видаляє елемент зі стану через шлюз зберігання. Ця оркестрація прихована за методами фасаду [2].

У домені ці шлюзи є інтерфейсами, котрі використовують мезанізм Dependency Injection Angular фреймворку. Використовуючи даний підхід реалізації інтерфейсів можуть бути динамічно замінені в процесі розробки, до прикладу на моковий сервіс в процесі тестування. Бізнес-логіка – це будь-яка логіка, не пов'язана з модулями представлення, клієнта і стану. Якщо додатку потрібно виконувати бізнес-логіку на клієнтській стороні, ця логіка виконується доменом. Відповідно до описаної архітектури була побудована діаграма компонентів (рис. 1).

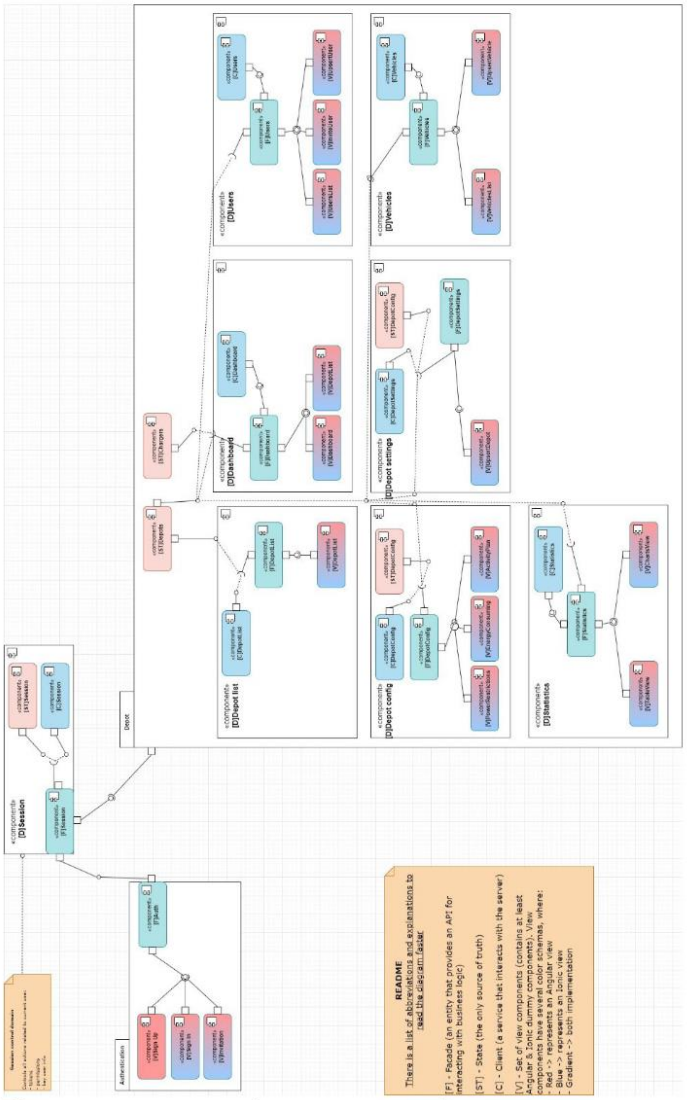


Рис. 1. **Діаграма компонентів системи моніторингу зарядних станцій використовуючи Onion архітектуру**

Рисунок Г.2 – Друга сторінка доповіді зі збірника (рисунок виконано самостійно)

Діаграма компонентів дозволить більш наочно ознайомитися з підходом використання Onion архітектуру в контексті розробки веб-додатку моніторингу зарядних станцій електромобілем.

Під час розробки додатку важливо приділити час визначенню чіткої архітектури. Впровадження Onion архітектури в проєкті Angular – це вагомий крок на шляху створення добре структурованого програмного забезпечення.

Інтеграція її архітектурних принципів не лише підвищує якість коду, але й полегшує співпрацю та масштабування.

Отже, розпочинаючи впровадження Onion архітектури у Angular проєкт, треба пам'ятати, що додаткові зусилля, спрямовані в організацію коду та розподіл обов'язків, окупляться в довгостроковій перспективі.

Список використаних джерел:

1. NG Poland Conf // YouTube: [Веб-сайт]. 2023. URL: https://youtu.be/nNIUTBHaiG8?si=-6N7aI5YVEevc2e_ (дата звернення: 27.11.2023).
2. Jérôme Navez // Medium: [Веб-сайт]. 2023. URL: https://medium.com/@navez.jerome/applying-the-onion-architecture-to-angular-projects-b37736d2c996 (дата звернення: 01.12.2023).

Рисунок Г.3 – Третя сторінка доповіді зі збірника (рисунок виконано самостійно)

ДОДАТОК Д

Use Case Діаграма

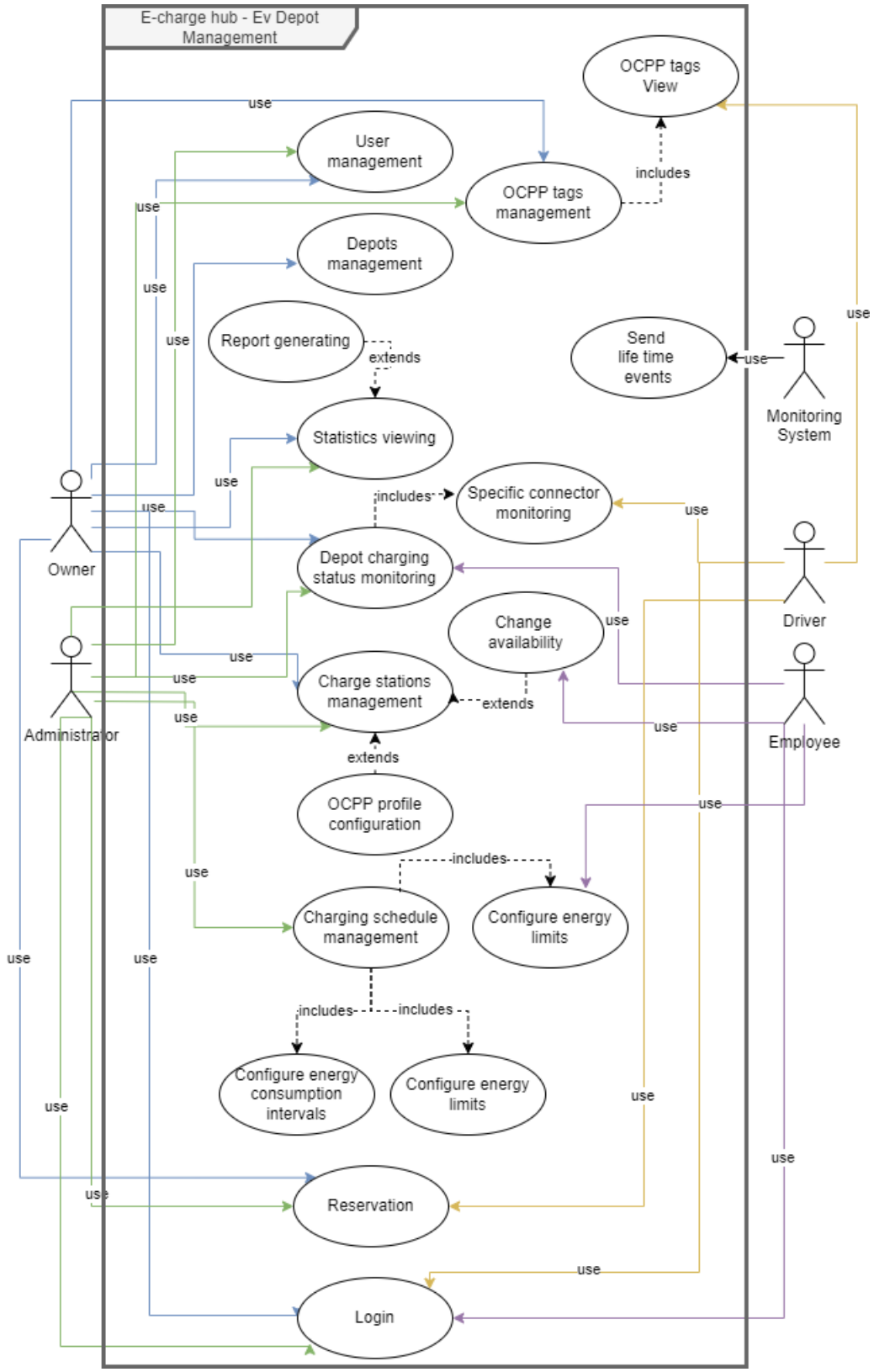


Рисунок Д.1 – Use Case діаграма додатку для управління депо електротранспорту (рисунок виконано самостійно)

ДОДАТОК Е

Діаграма компонентів

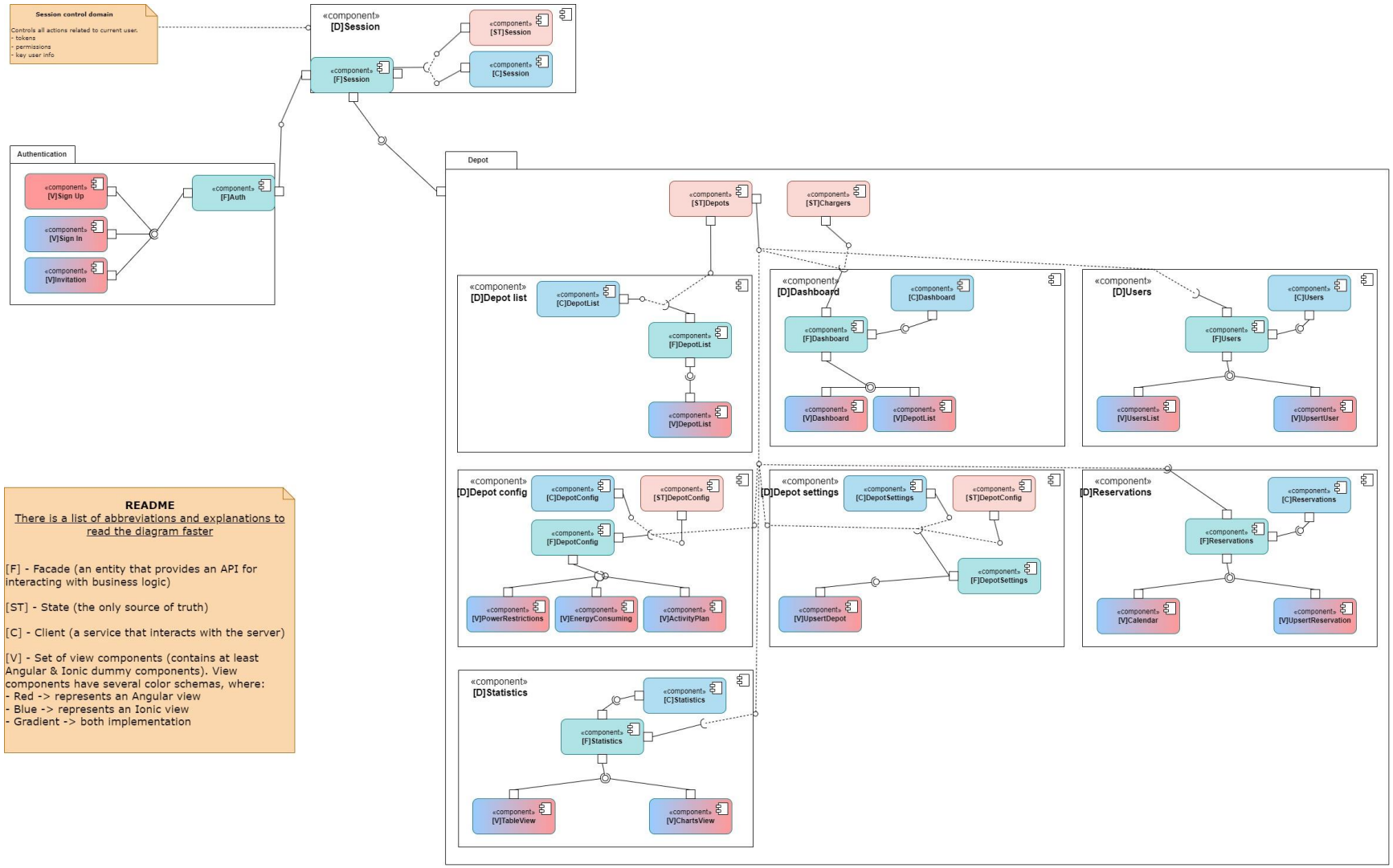


Рисунок Е.1 – Діаграма компонентів згідно Onion архітектури клієнтської частини (рисунок виконано самостійно

ДОДАТОК Ж

Діаграма діяльності

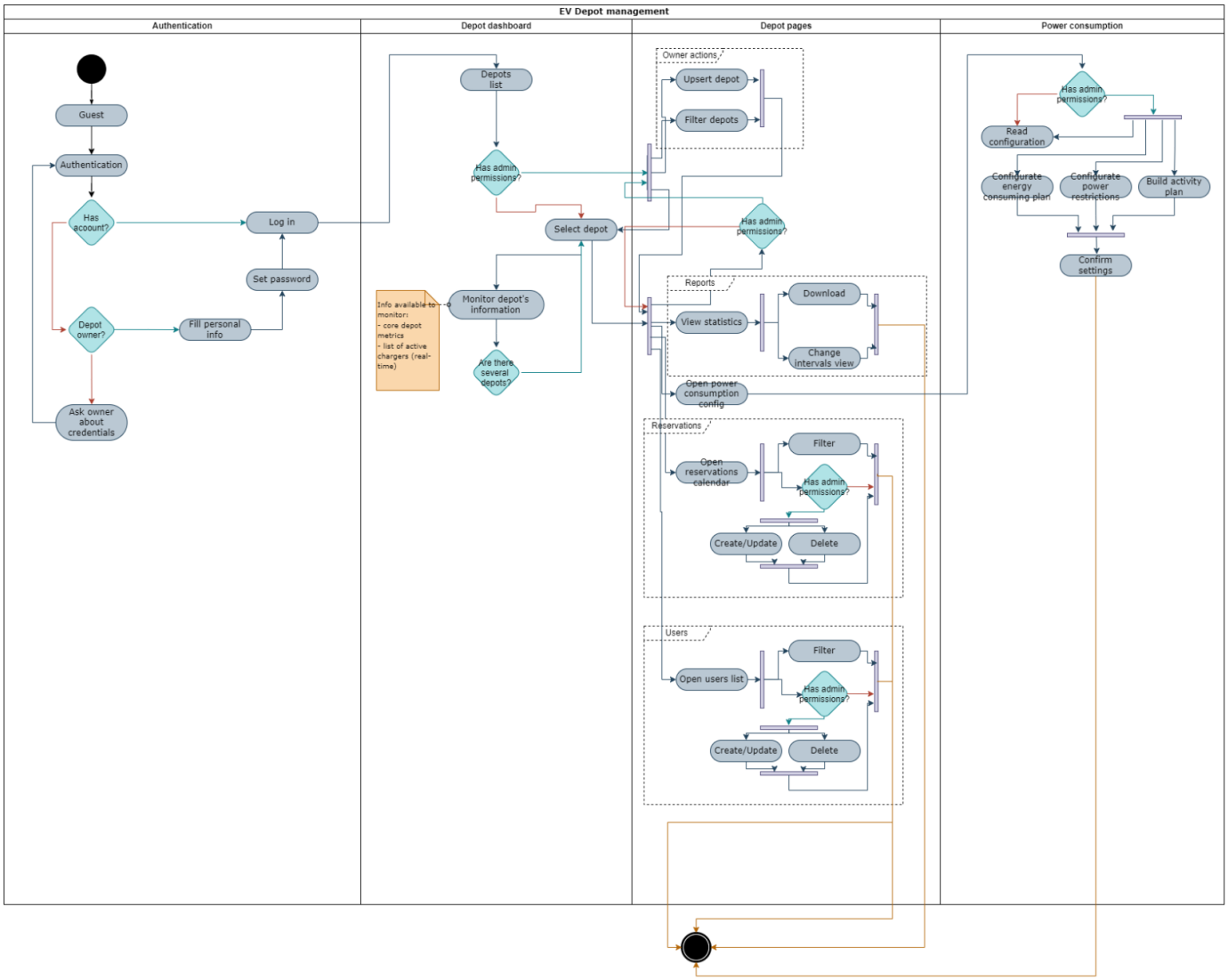


Рисунок Ж.1 – Діаграма діяльності системи для управлінням споживання електроенергії депо електротранспорту адміністротором системи (рисунок виконано самостійно)

ДОДАТОК И

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

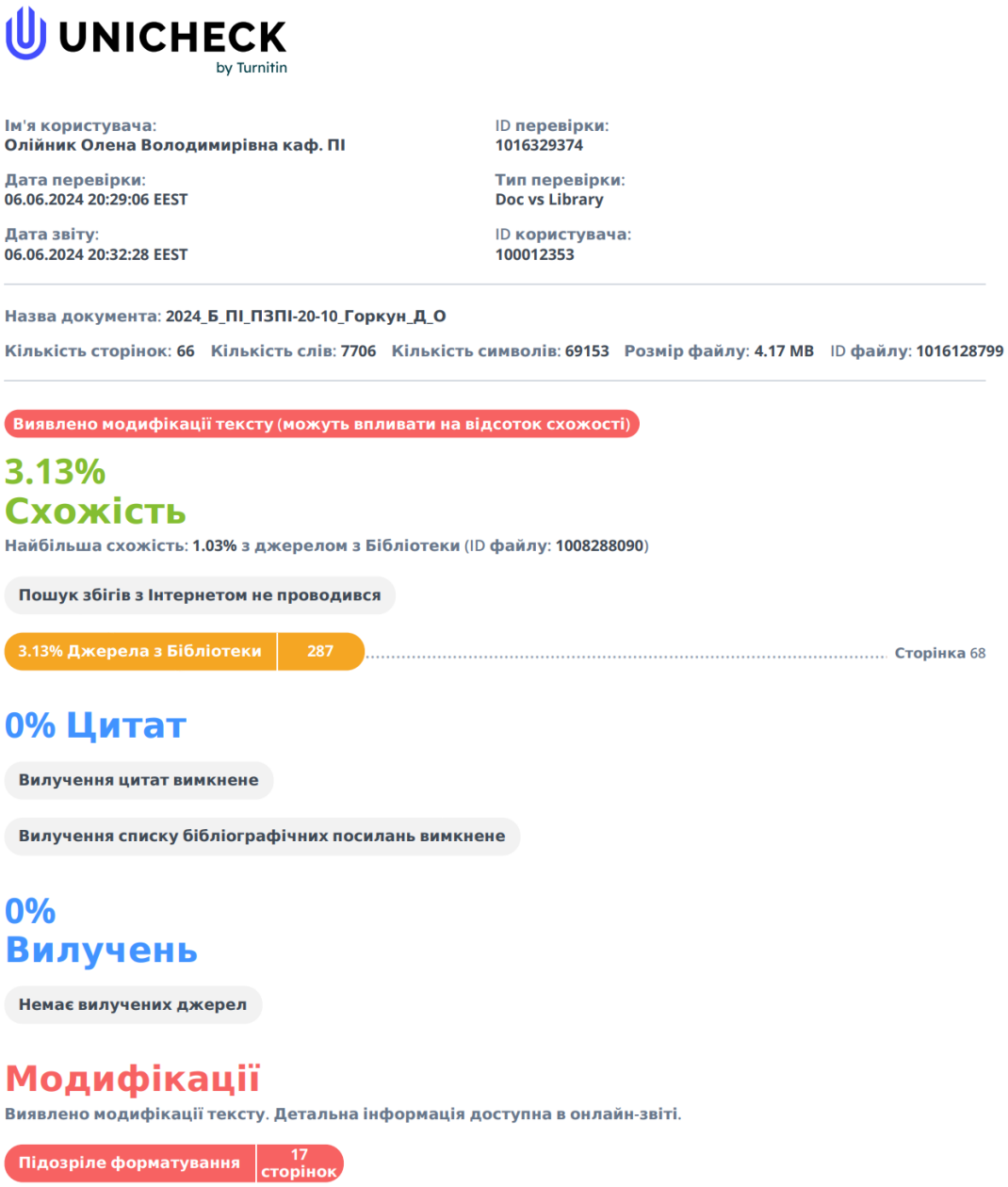


Рисунок И.1 – Результат перевірки записки на антиплагіат

ДОДАТОК К

Слайди презентації

E-charge Hub

Харківський національний університет радіоелектроніки
Кафедра ПІ
Кваліфікаційна робота бакалавра

Програмна система моніторингу зарядних станцій електромобілів. Розробник клієнтської частини

Виконав: ст. гр. ПЗПІ-20-10 Горкун Д.О.
Керівник: доц. каф. ПІ Русакова Н.Є.

Зручність & Ефективність

E-charge Hub

Мета дослідження

Створення сучасної та ефективної системи управління зарядними станціями для електротранспорту

Ми прагнемо надати інструмент для контролю, моніторингу та оптимізації процесу зарядки, що підвищить ефективність використання ресурсів і знизить експлуатаційні витрати

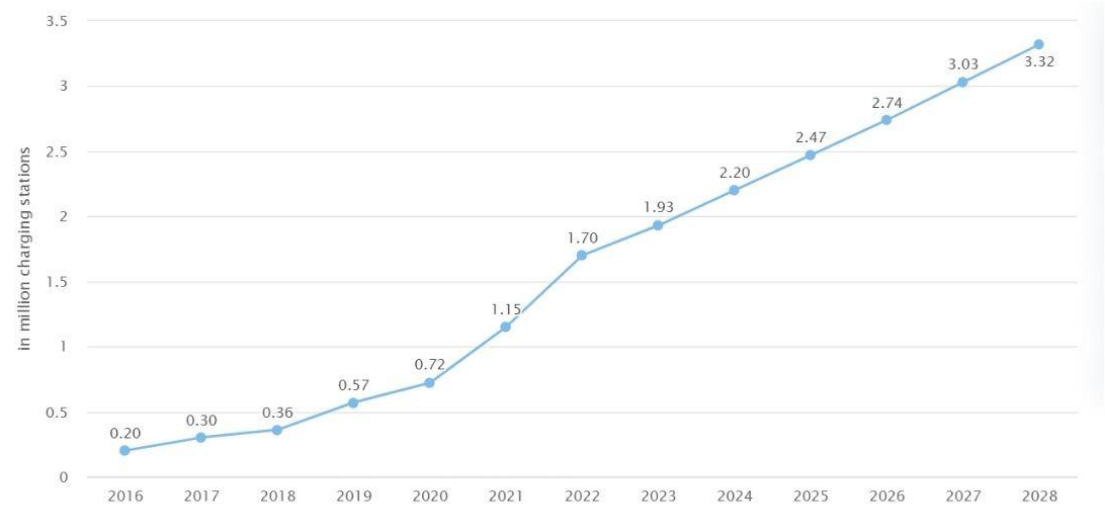
Підготував: Горкун Д.О.

02

E-charge Hub

Аналіз ринку

К-ть зарядних станцій



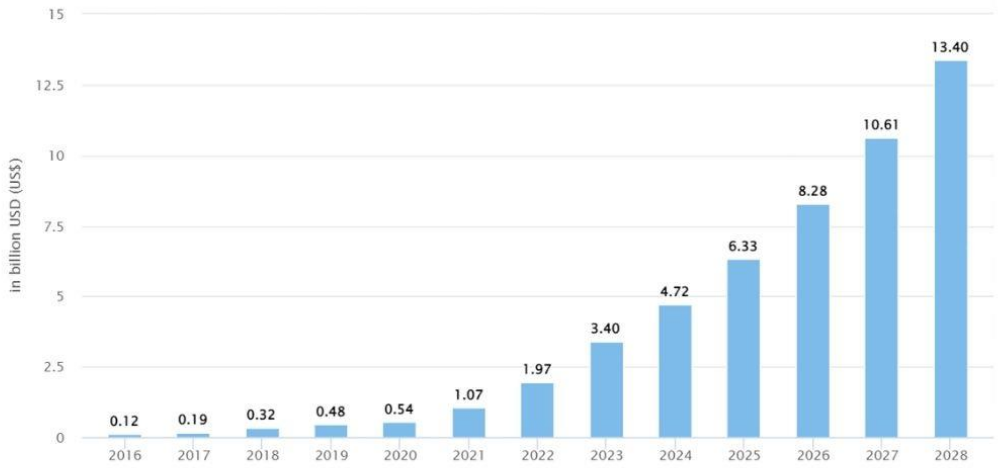
Most recent update: Sep 2023

Source: Statista Market Insights

E-charge Hub

Аналіз ринку

Дохід від зарядних станцій



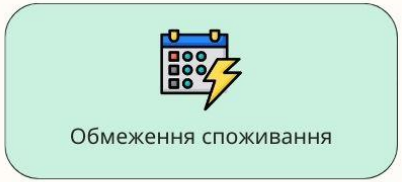
Notes: Data shown is using current exchange rates. Data shown does reflect market impacts of Russia-Ukraine war.

Most recent update: Sep 2023

Source: Statista Market Insights

E-charge Hub

Задачі продукту



Масштабованість

Підтримка великої кількості зарядних станцій та користувачів без зниження продуктивності системи.

Продуктивність

Швидка обробка запитів та мінімізація затримок в роботі інтерфейсу

Надійність

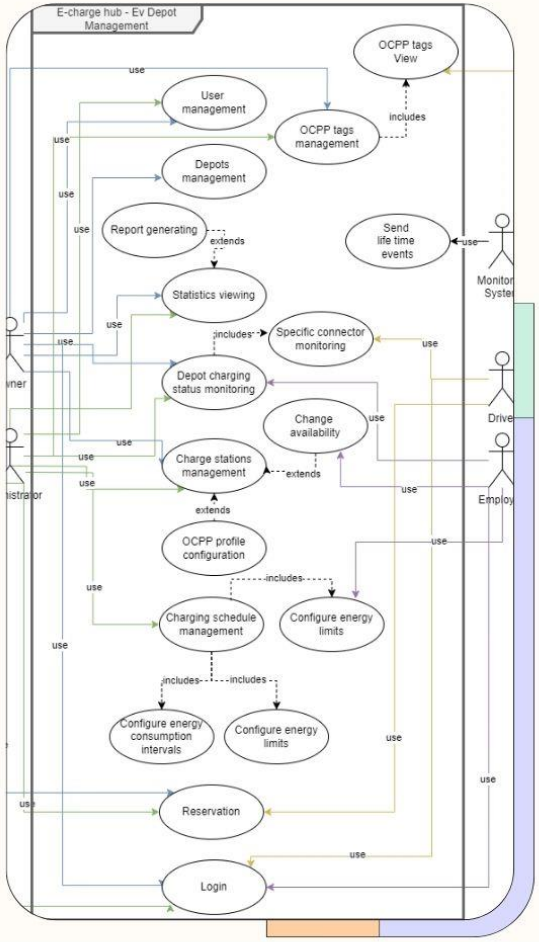
Забезпечення високої доступності системи та мінімізація часу простою

E-charge Hub

Постановка задачі

Розробка інтуїтивного, швидкодійного та надійного інтерфейсу для E-Charge Hub. Основні завдання:

- Розробити інтерфейс для управління мережею депо та моніторингу зарядних станцій
- Розробити функціоналу резервацій з інтерактивним календарем
- Оптимізувати інтерфейсу використовуючи мемоізацію, CDN, gzip, віртуальний DOM, мініфікацію, лініве завантаження
- Реалізувати систему обмеження споживання енергії з візуальними індикаторами
- Інтегрувати бібліотеки ApexCharts для візуалізації даних та підтримка експорту в CSV
- Забезпечити багатомовності з використанням ngx-translate та бібліотеки dayjs для роботи з датами



Користувачі системи

Власники

Керують мережею депо, налаштовують параметри системи. Делегують контроль адміністраторам

Адміністратори

Відповідають за моніторинг і оперативне управління зарядними станціями

Співробітники

Забезпечують підтримку та обслуговування зарядних станцій

Водії

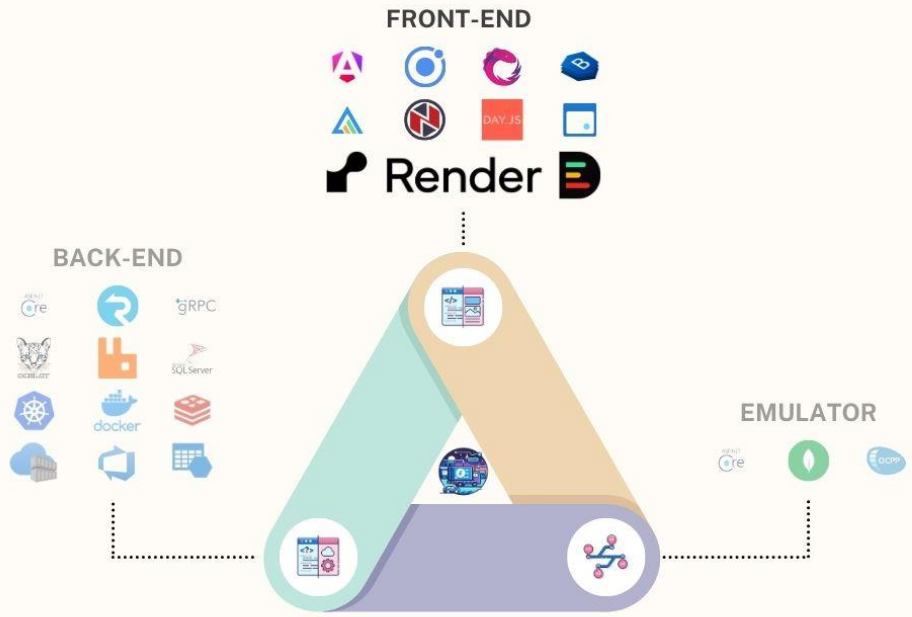
Використовують систему для бронювання та контролю процесу зарядки електромобіля

Програмно використовується

RBAC **Для керування дозволами**

E-charge Hub

Технології



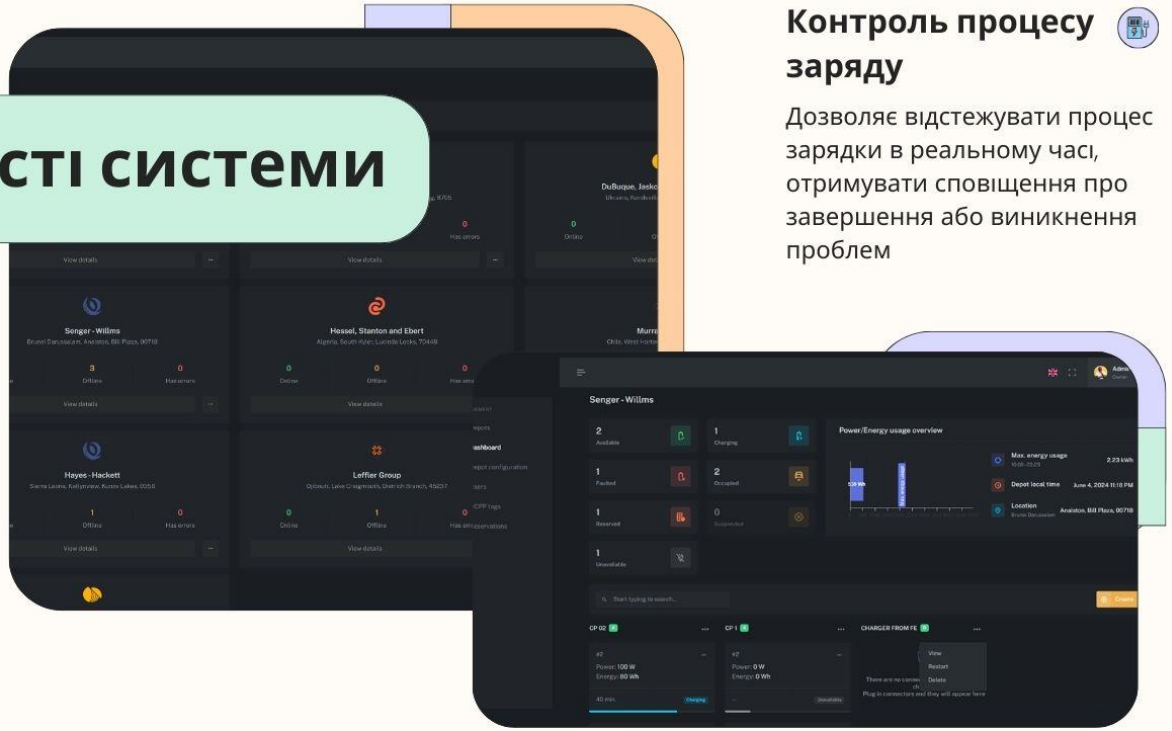
- **Angular**: Створення динамічного і інтерактивного фронтенду
- **SignalR**: Реалізація обміну повідомленнями в реальному часі
- **Ionic**: Забезпечення мобільної версії додатку
- **FullCalendar**: Інтерактивний календар резервацій
- **ApexCharts**: Візуалізація статистичних даних
- **ngx-translate**: Динамічна зміна мови інтерфейсу
- **Render**: Хостинг та автоматичне розгортання додатку
- **DeepSource**: Автоматичний аналіз якості коду

E-charge Hub

Можливості системи

Управління мережею депо

Забезпечує централізоване керування всіма зарядними станціями, моніторинг їх стану та контроль параметрів

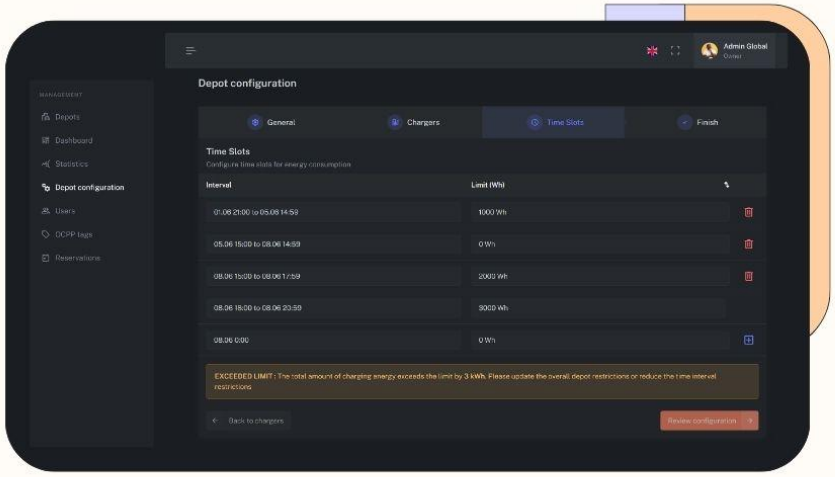


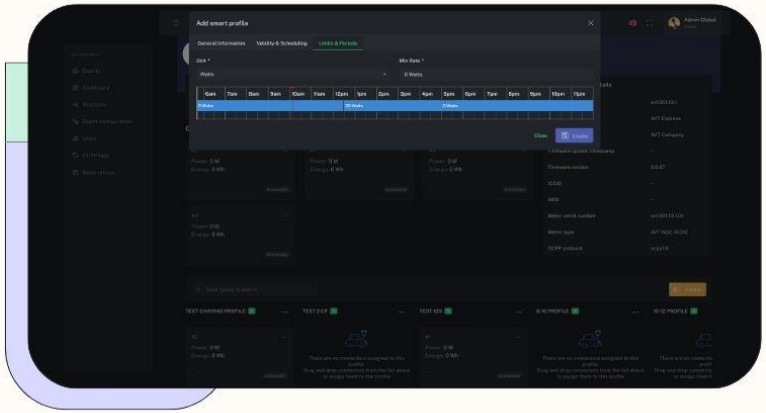
Контроль процесу заряду

Дозволяє відстежувати процес зарядки в реальному часі, отримувати сповіщення про завершення або виникнення проблем

Обмеження споживання енергії

Впроваджує механізми для контролю максимального споживання енергії депо в певні часові рамки





Обмеження потужності

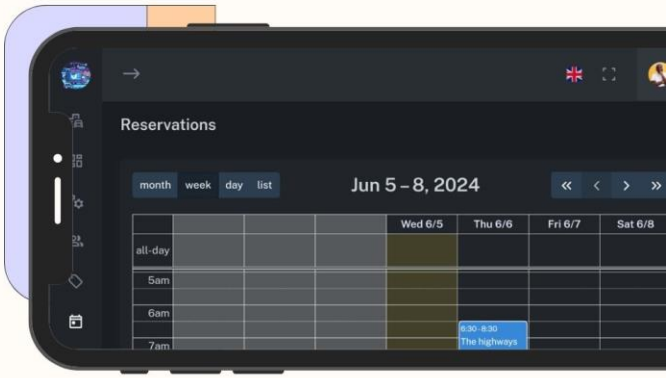
Дозволяє оптимізувати використання потужності зарядних станцій, забезпечуючи рівномірний розподіл навантаження використовуючи протокол OCPP

E-charge Hub

Можливості системи

Резервації

Реалізовані через інтерактивний календар, що дозволяє водіям бронювати зарядні станції заздалегідь.

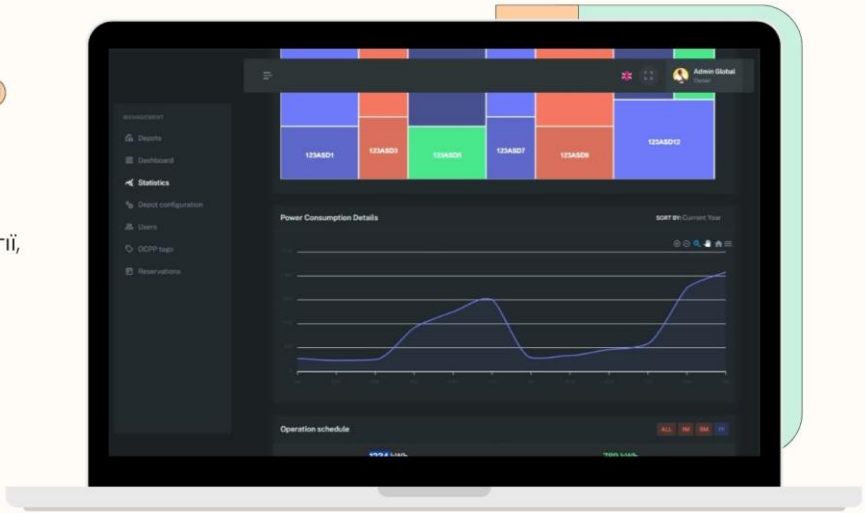


E-charge Hub

Можливості системи

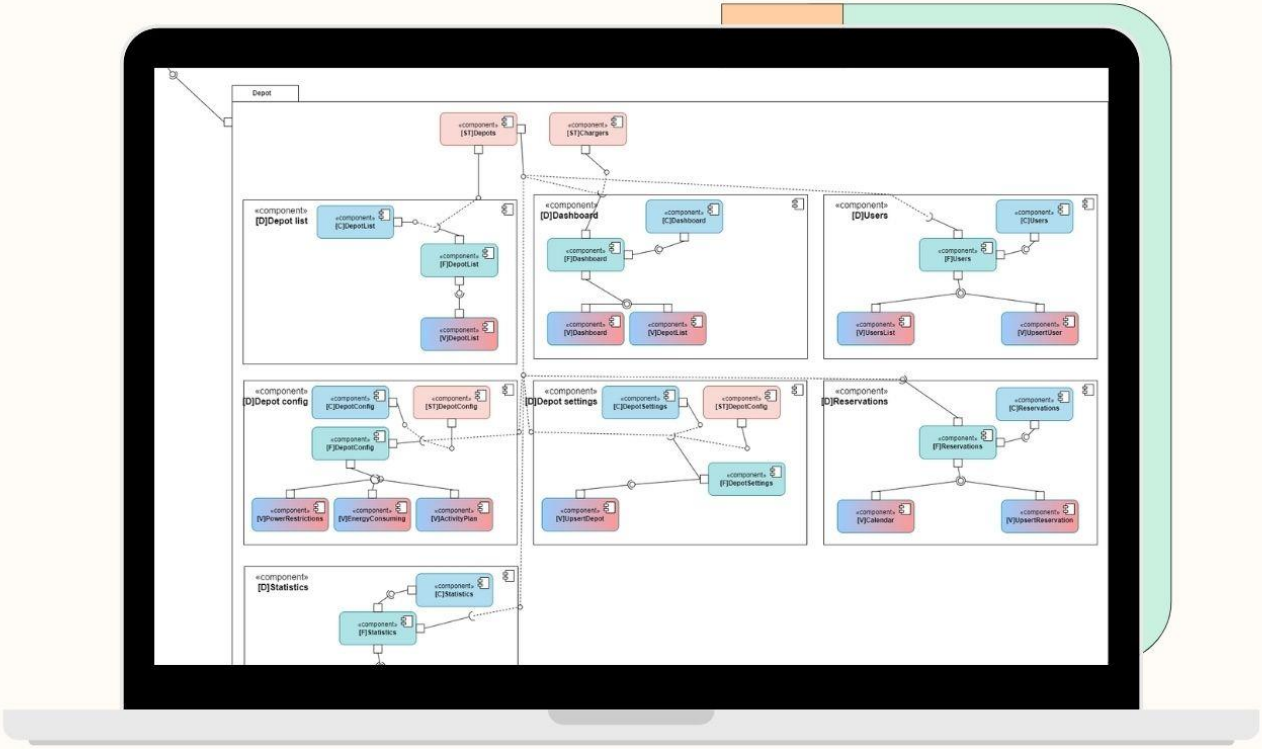
Гнучка статистика

Надає інструменти для аналізу даних щодо використання енергії, потужності та роботи зарядних станцій.



Onion Архітектура

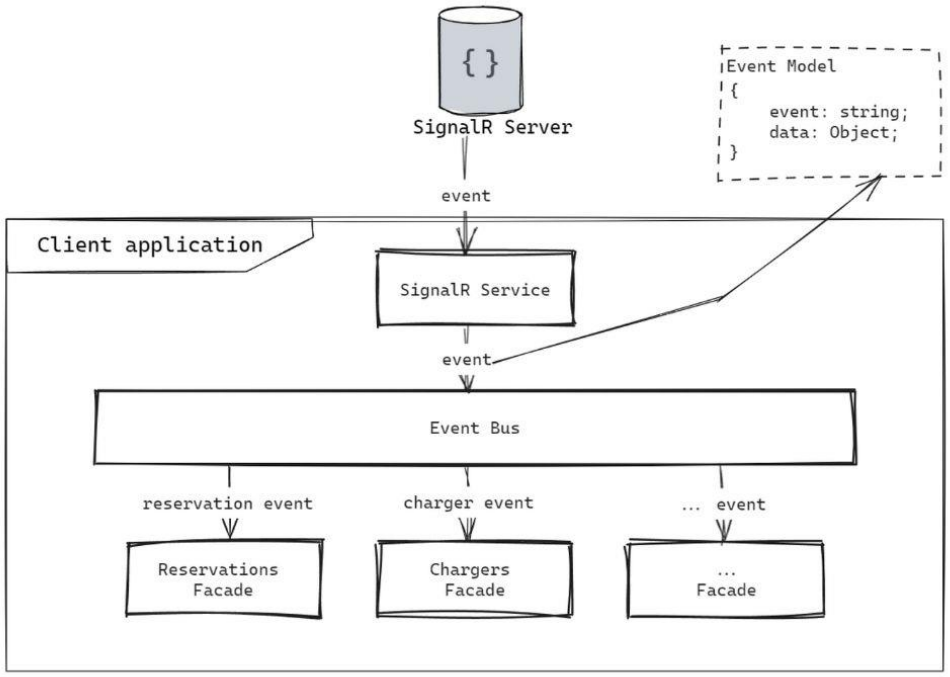
Система побудована за принципами Onion архітектури, що забезпечує високий рівень абстракції та гнучкості.
Це дозволяє легко масштабувати та розширювати систему, підтримувати високий рівень модульності та зменшувати залежності між компонентами.



Live-time зміни

Система E-Charge Hub підтримує обробку подій в реальному часі за допомогою SignalR та шаблону проектування Event Bus.

Це забезпечує оперативну передачу даних про стан зарядних станцій, сповіщення про завершення зарядки, помилки тощо.



E-charge Hub

Контроль якості коду

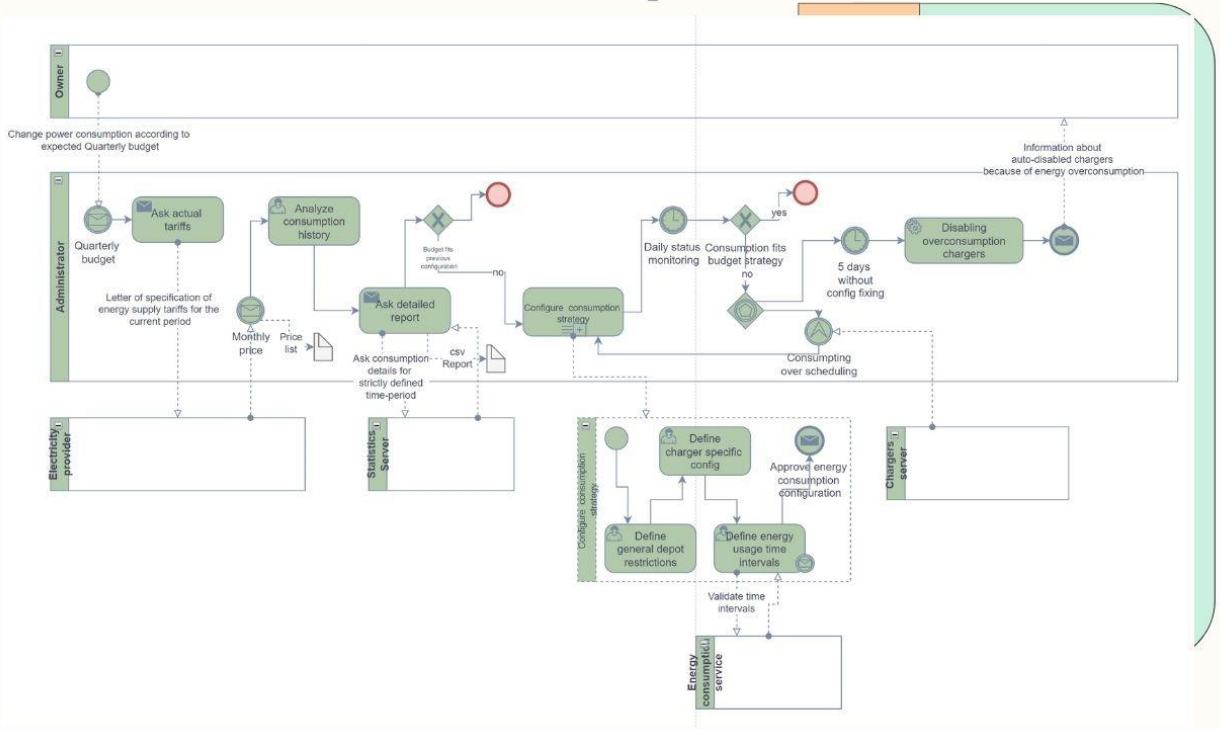
DeepSource

Для забезпечення високої якості коду в процесі розробки було використано інструмен DeepSource.

Переваги:

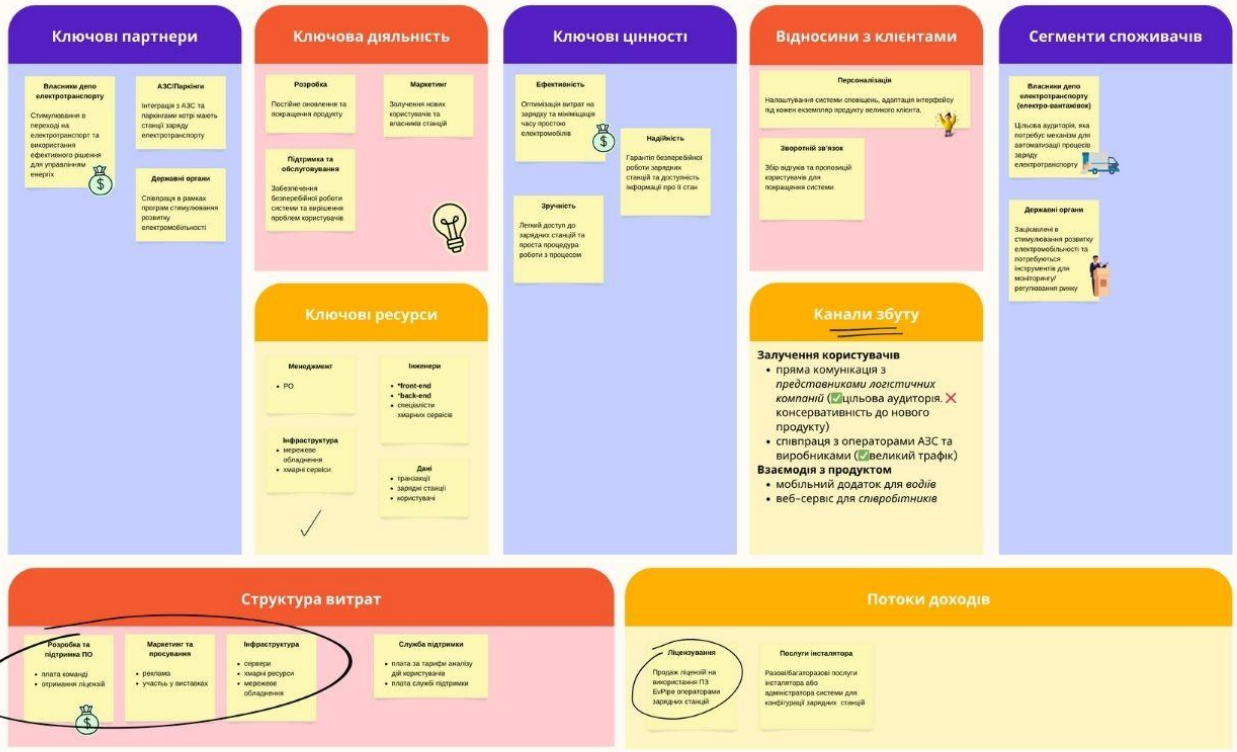
- Виявлення потенційних багів
- Надання рекомендацій
- Інтеграція з CI/CD

Обмеженння енергоспоживання



Діаграма BPMB (Business Process Model and Notation)

Canva бізнес модель

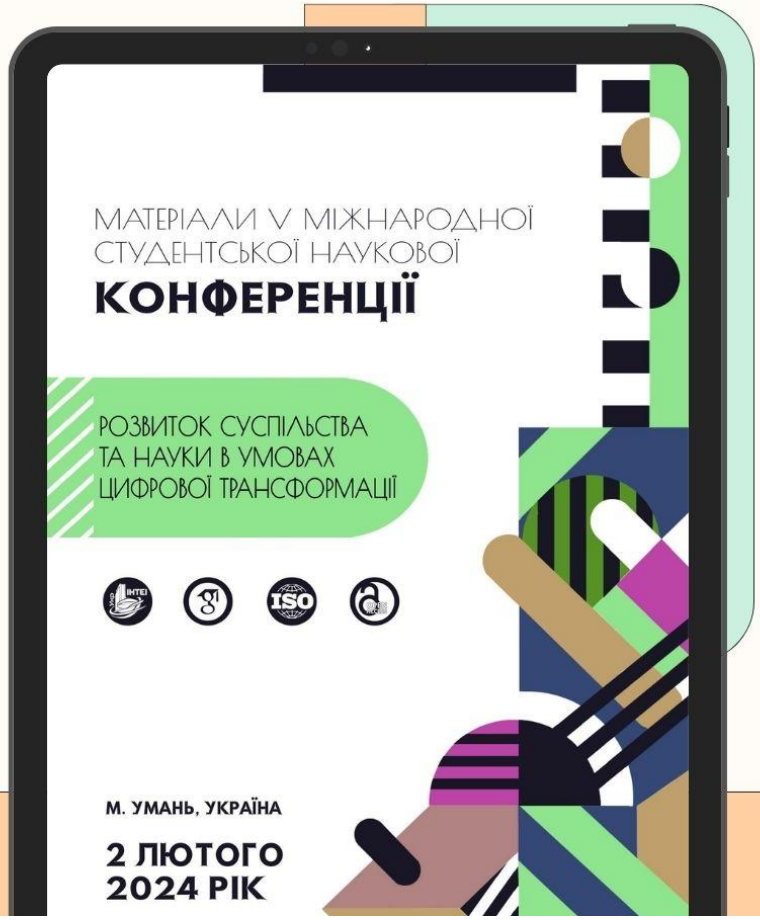


E-charge Hub

Апробація

Зе результатами аналізу та реалізації закладеної архітектури архітектури були опубліковані тези:

ПРОЕКТУВАННЯ ANGULAR ДОДАТКУ ДЛЯ ПРОГРАМНОЇ СИСТЕМИ МОНІТОРИНГУ ЗАРЯДНИХ СТАНЦІЙ ЕЛЕКТРОМОБІЛЕЙ ВИКОРИСТОВУЮЧИ ONION АРХІТЕКТУРУ



E-charge Hub

Комплексне рішення

Стійка архітектура

Крос-платформний продукт

Контроль якості коду

Висновки

Система E-Charge Hub - сучасне рішення для управління зарядними станціями електротранспорту. Вона забезпечує ефективне керування ресурсами, контроль процесу зарядки, оптимізацію споживання енергії та надання статистичної інформації.

ДОДАТОК Л

Специфікація ПЗ

E-Charge Hub

Специфікація вимог до програмного забезпечення

3.0

01.06.2024

Налєскіна Тетяна

Горкун Дмитро

Чубаров Євген

Рубель Денис

Історія версій

Дата	Опис	Автор	Коментар
01.05.24	Версія 1.0	Налєскіна Тетяна	Створення документу
20.05.24	Версія 2.0	Горкун Дмитро	Оновлення використаних технологій
01.06.24	Версія 3.0	Чубаров Євген	Оновлення використаних технологій з боку бекенду

Зміст

1. Вступ

1.1 Огляд продукту

E-Charge Hub – програмна система моніторингу зарядних станцій електромобілів забезпечує контроль, управління та оптимізацію процесу зарядки електромобілів. Основні функції включають збір даних, аналіз ефективності станцій, управління процесом зарядки, відстеження споживання енергії та забезпечення безпеки даних. Система спрямована на ефективне управління зарядними станціями з урахуванням зростання попиту на електромобілі та необхідності розвитку відповідної інфраструктури. Монетизація проекту досягається шляхом придбання програмного забезпечення нашого продукту.

1.2 Мета

Метою програмної системи моніторингу зарядних станцій електромобілів є забезпечення зручності та ефективності процесу зарядки для користувачів електромобілів. Наша програмна система спрямована на те, щоб забезпечити адміністраторам зарядних станцій інноваційний та інтегрований підхід до управління, моніторингу та оптимізації їх інфраструктури. Ми розглянемо, як система надає централізоване управління, реальний час, аналітику, а також забезпечує високий рівень безпеки та енергоефективності. Розкриючи ці аспекти, ми створимо повністю осяжний образ та зрозуміння проекту, який сприятиме не тільки подальшому розвитку інфраструктури зарядних станцій, але й покращенню користувацького досвіду та сприянню сталому розвитку.

1.3 Межі

Програмна система E-Charge Hub має надавати можливості що дозволить зручно керувати депо та зарядними станціями в них, завчасно бронювати зарядні станції для подальшого використання та моніторингу енергоспоживання. Продукт має складатися з веб-додатку та мобільної версії, яка буде повністю відтворювати функціонал продукту. Система повинна мати можливість реєстрації та аутентифікації користувачів з різними ролями та надавати доступ до функцій згідно з ролями користувачів системи.

Серверна частина має бути реалізована з використанням мікросервісної архітектури, клієнтська з використанням Onion архітектури. Здійснення контролю над процесом розробки має проводитися за допомогою програмного забезпечення Jira.

1.4 Посилання

Текст документу містить наступні посилання:

1. Горкун Д.О., Налєскіна Т.С. Проектування Angular додатку для програмної системи моніторингу зарядних станцій електромобілів з використанням Onion архітектури. Тези V Міжнародна студентська наукова конференція «РОЗВИТОК СУСПІЛЬСТВА ТА НАУКИ В УМОВАХ ЦИФРОВОЇ ТРАНСФОРМАЦІЇ», м. Умань, 2024. С. 60-62. URL: https://doi.org/10.36074/liga-inter-02.02.2024
2. Чубаров Є.Е., Рубель Д.А. Проєктування серверної частини системи керування та моніторингу зарядними станціями. Тези XXVIII Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» конференції «Інформаційні інтелектуальні системи», м. Харків, 2024. С. 306-308. URL: https://doi.org/10.30837/IYF.IIS.2024.306
3. Посилання на репозиторій: https://github.com/Nexus20/ChargingStation.Backend , https://github.com/Vspominay/ev-charging-station

1.5 Означення та абревіатури

В документі використовуються наступні означення та абревіатури:

OCPP – Open Charge Point Protocol

HTTP – HyperText Transfer Protocol

gRPC – Google Remote Procedure Calls

REST – REpresentational State Transfer

SQL – Structured Query Language

K8S – Kubernetes

ACR – Azure Container Registry

AKS – Azure Kubernetes Service

2. Загальний опис

2.1 Перспективи продукту

У зв'язку зі стрімким розвитком та впровадженням електромобільної технології та зростанням інтересу до сталих енергетичних рішень, створення програмної системи моніторингу зарядних станцій електромобілів стає необхідністю. За даними Міжнародного агентства з енергетики (IEA), кількість електромобілів на світі стрімко зростає, досягнувши позначки у 10 мільйонів одиниць у 2022 році, що свідчить про поступову трансформацію транспортної системи. Цей інноваційний підхід вимагає ретельного врахування численних передумов для успішного розгортання та оптимального функціонування системи.

Програмна система орієнтована на такі сегменти ринку:

1. Ринок послуг з моніторингу та аналізу: розробка системи, яка забезпечує детальний моніторинг ефективності зарядних станцій, створює можливість для підприємців надавати послуги з аналізу та оптимізації зарядних станцій для власників та операторів електротранспорту.

2. Системи управління та сервісні контракти: реалізація систем управління дозволяє підприємцям пропонувати сервісні контракти для підтримки та оптимізації роботи зарядних станцій, що включають в себе віддалене моніторинг, технічну підтримку та регулярне обслуговування.

3. Розвиток інфраструктури зарядних станцій: сприяння розвитку інфраструктури зарядних станцій для електромобілів за допомогою вдосконаленої системи моніторингу може стати ключовим фактором для бізнесу в області будівництва та управління інфраструктурою.

4. Платформи для користувачів електромобілів: розробка мобільних додатків та платформ для користувачів електромобілів, які надають інформацію про доступність та стан зарядних станцій, може створити нові можливості для реклами, партнерських програм та користувачських планів.

5. Аналітика для електромобільних виробників: надання аналітичних звітів та статистики виробникам електромобілів щодо використання їх продукції

може стати основою для покращення технічних характеристик, а також розробки нових моделей, що задовольнять потреби ринку.

6. Енергетичні консалтингові послуги: врахування споживання електроенергії та оптимізація часу зарядки може створити можливості для компаній, які спеціалізуються на консалтингових послугах у галузі енергетики.

7. Екологічно орієнтовані ініціативи:розвиток системи моніторингу може бути використаний для підтримки екологічних ініціатив та отримання фінансової підтримки від урядових та неприбуткових організацій.

2.2 Функції продукту

Програмна система для моніторингу зарядних станцій електромобілів є комплексним інформаційним рішенням, яке об'єднує в собі різноманітні компоненти і функції для ефективного керування та контролю зарядними станціями. Основні складові програмної системи включають:

Моніторинг і збір даних. Система здатна отримувати дані з різних зарядних станцій, включаючи інформацію про стан заряду, використання енергії, доступність станцій та інші параметри.

Аналітика та звітність. Можливість аналізувати накопичені дані для отримання важливої інформації щодо ефективності зарядних станцій, споживання енергії, роботи системи тощо. Генерація звітів та статистики для подальшого аналізу та прийняття управлінських рішень.

14 Управління зарядками. Функціонал для керування процесом зарядки,

включаючи планування заряду, встановлення пріоритетів, управління через віддалений доступ тощо.

Система оповіщень та аварійного управління. Можливість автоматичного виявлення проблем та аварій, надсилання сповіщень та управління ситуаціями навіть у відсутності користувача.

Інтерфейс для користувачів. Зручний та інтуїтивно зрозумілий інтерфейс для користувачів, що дозволяє легко взаємодіяти з програмною системою, переглядати дані, встановлювати параметри та отримувати звіти.

Захист та безпека. Забезпечення захисту даних, використання шифрування, аутентифікації користувачів та інших заходів для забезпечення безпеки і конфіденційності інформації.

2.3 Характеристики користувачів

Функції, які планується реалізувати у першому релізі, було розділено на групи користувачів для полегшення їх представлення. Відомо, що система буде взаємодіяти з п'ятьма типами користувачів: власники, адміністратори від компаній, моніторингова система, водії та працівники.

Власник (Owner) може займатись менеджментом адміністраторів та депо, моніторингом статусу зарядних станцій депо, що включає в собі моніторинг конкретного роз'єму, авторизуватись у системі.

Адміністратор (Administrator) може займатись менеджментом користувачів, дивитись статистику, що отримується зі звітів, зробити резервацію, авторизуватись. Займатись менеджментом графіку зарядки, що включає налаштування інтервалів споживання електроенергії та налаштування обмежень потужності.

Моніторингова система (Monitoring system) відправляє пуш-повідомлення.

Водій (Driver) може моніторити конкретний роз’єм.

Працівник (Employee) може моніторити статус зарядних станцій депо, займатись менеджментом графіку зарядки, що включає налаштування інтервалів споживання електроенергії та налаштування обмежень потужності, робити резервації та авторизуватись у системі.

2.4 Загальні обмеження

Програмна система для моніторингу зарядних станцій електромобілів склається з серверної та клієнтської частини. Серверна частина буде реалізована за допомогою мікросервісної архітектури. Концепція мікросервісної архітектури полягає у дизайні архітектури програмної системи таким чином, що функціональність розподіляється між сервісами. Розробка мікросервісів для системи буде проводитися на мові програмування C# з використанням платформи .NET 8 та фреймворку ASP.NET Core. Обрання цих конкретних

технологій пояснюється їхньою здатністю підтримувати широкий спектр бібліотек, зокрема ті, які використовуються для роботи з хмарними технологіями, gRPC і RabbitMQ. Технологічний стек для розгортання програмної системи моніторингу зарядних станцій електромобілів використовує сучасні інструменти та підходи для ефективного управління та функціонування системи. У цьому контексті, використання Docker є важливим елементом для забезпечення ізольованості сервісів, а K8S є ключовим елементом оркестрації ресурсів. Для якнайшвидшої доставки оновлень клієнту, налаштовано CI/CD процеси за допомогою Azure DevOps Pipelines, що дозволяє автоматизувати розгортання рішення в хмару з використанням ACR та AKS відповідно.

Клієнтська частина розробляється за допомогою декількох технологій: Angular, Ionic, RxJs, Ng-bootstrap, FullCalendar, Ngx-translate, dayjs, Jest. Для управління базами даних у проекті обрано систему Microsoft SQL Server (MS SQL Server).

2.5 Припущення та залежності

Припущення 1. Продукт буде затребуваний на ринку;

Припущення 2. Вихід обладнання з експлуатації може призвести до відмови сервісу;

Припущення 3. Зворотна сумісність;

Залежність 1. Використання протоколу OCPP для стандартизації підходів комунікації між станціями та центральною системою;

Залежність 2. Серверна частина буде реалізована на платформі ASP.NET;

Залежність 3. Використання СУБД Azure SQL Server та Azure Table Storage зберігання даних;

Залежність 4. Використання Docker для контейнеризації та K8S для оркестрації ресурсів;

Залежність 5. Використання брокеру повідомлень RabbitMQ для асинхронної комунікації;

Залежність 6. Використання Redis Cache для кешування;

Залежність 7. Застосування Azure DevOps Pipelines для налаштування процесів CI/CD;

Залежність 8. Використання ACR та AKS для розгортання серверної частини в хмарі Azure;

Залежність 9. Для реалізації клієнтської частини використовується Angular, Ionic, RxJs, Ng-bootstrap, FullCalendar, Ngx-translate, dayjs, Jest.

Залежність 10. Застосування Render для розгортання клієнтської частини.

Залежність 11. Швидкість відгуку та оновлення залежать від потужності та стійкості сервера, який приймає та обробляє дані, а також від швидкості інтернет-з'єднання;

Залежність 12. Залежність часу розробки цього проекту від часу, який розробники витрачають вивчення та розробку паралельних завдань.

3. Конкретні вимоги

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Неавторизований користувач має потрапляти на сторінку входу в систему (рисунок 1).

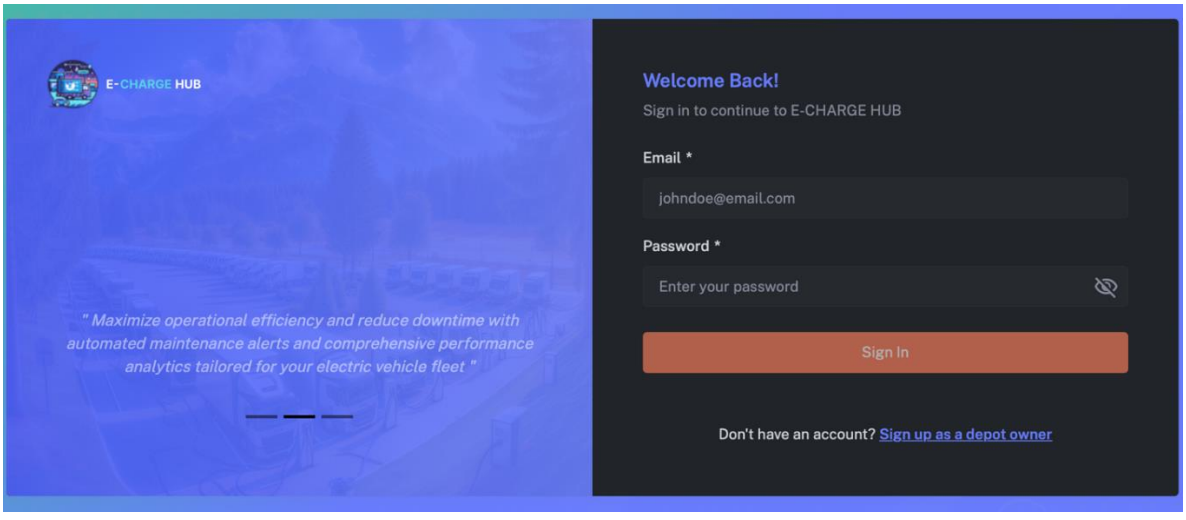


Рисунок 1 – Лендінг сторінка

На сторінці входу мусить бути логотип та слайдер з описом переваг системи. Також має бути форма входу в систему за поштою та паролем. Тут же мусить бути посилання на сторінку реєстрації.

На формі реєстрації (рисунок 2) повинні бути наступні елементи. Поле для введення електронної пошти та номеру телефону, де користувач вводить свою адресу електронної пошти для реєстрації в системі. Поле для введення пароля, яке дозволяє користувачеві створити пароль для захисту свого облікового запису. Також необхідно мати поле для введення імені, де користувач вводить своє ім'я, і поле для введення прізвища, де вводиться прізвище. Після заповнення всіх полів користувач натискає кнопку "Реєстрація", щоб завершити процес реєстрації. Для тих, хто вже зареєстрований, повинна бути кнопка "Увійти" для переходу на сторінку авторизації. Крім того, на формі повинно бути посилання на політику конфіденційності, яку користувач повинен прийняти під час реєстрації.

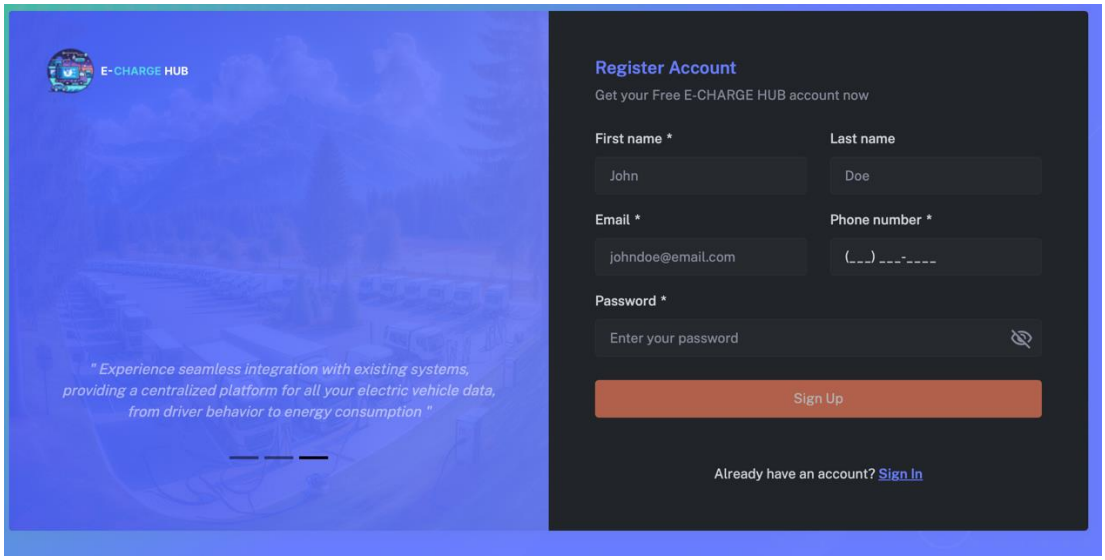


Рисунок 2 – Форма для залишення заявки

На сторінці списку користувачів системи моніторингу зарядних станцій депо (рисунок 3) необхідно відобразити заголовок сторінки, який ясно вказує, що це список користувачів системи. Далі має бути пошуковий рядок, що дозволяє швидко знайти конкретного користувача за ім'ям або іншими критеріями. Сам список користувачів повинен містити таблицю з такими даними: ім'я користувача, прізвище, електронна пошта, роль у системі (наприклад, адміністратор) та дату останнього входу в систему. Кожний рядок таблиці повинен мати кнопки для редагування або видалення користувача, а також можливість перегляду детальної

інформації про користувача. Для зручності можна додати можливість сортування даних у таблиці за різними критеріями, такими як ім'я або дата останнього входу. Також бажано мати кнопку для додавання нового користувача, яка перенаправляє на форму реєстрації нового користувача.

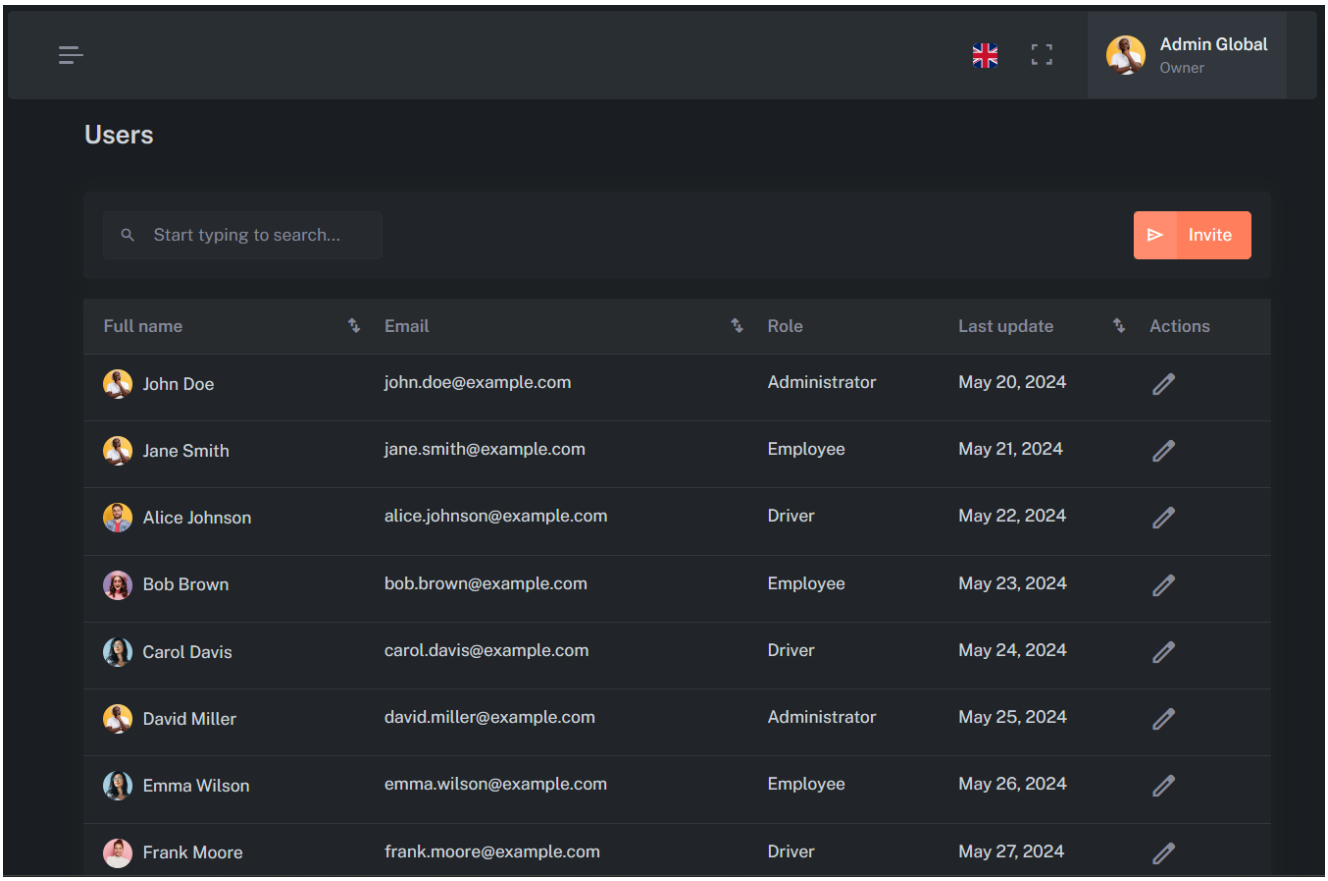


Рисунок 3 - Сторінка списку користувачів системи моніторингу зарядних станцій депо

На сторінці створення резервації зарядної станції (рисунок 4) повинні бути представлені наступні елементи. Спочатку має бути заголовок сторінки, який чітко вказує, що це форма для створення резервації зарядної станції. Повинно бути поле для вибору зарядної станції зі списку доступних станцій, де користувач може вибрати потрібну станцію для резервації та Ocpp тег. Додати поле для коментарів, де користувач може залишити додаткові примітки щодо резервації, початок зарядки та кінець. Кнопка підтвердження резервації повинна бути добре помітною, щоб користувач міг легко завершити процес. Після натискання на кнопку

підтвердження повинно з'явитися повідомлення про успішне створення резервації або повідомлення про помилку у разі некоректного введення даних. На сторінці також має бути кнопка для скасування, яка дозволяє користувачеві повернутися до попередньої сторінки без збереження змін.

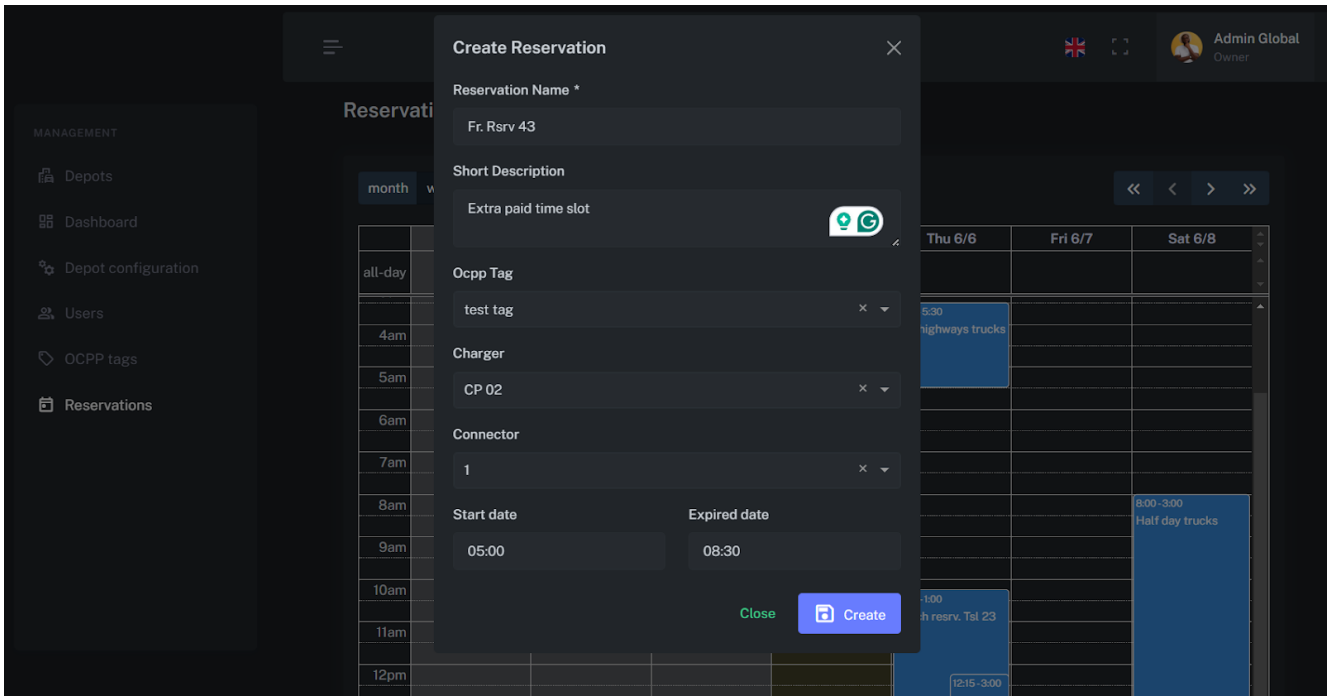


Рисунок 4 - Сторінка створення резервації зарядної станції

На сторінці календаря (рисунок 5) повинні бути представлені наступні елементи. Заголовок сторінки, який чітко вказує, що це календар, де відображаються заплановані резервації зарядних станцій. Сам календар має бути інтерактивним, дозволяючи користувачам переглядати дні, тижні та місяці. Користувачі повинні мати можливість легко переходити між різними періодами, використовуючи кнопки для переміщення вперед і назад. У кожній клітинці календаря має бути відображена інформація про наявні резервації, такі як час і станція, яка зарезервована. Для зручності можна додати можливість перегляду деталей резервації при наведенні курсору або натисканні на конкретну резервацію. Повідомлення про помилки або відсутність резервацій також мають бути чітко відображені.

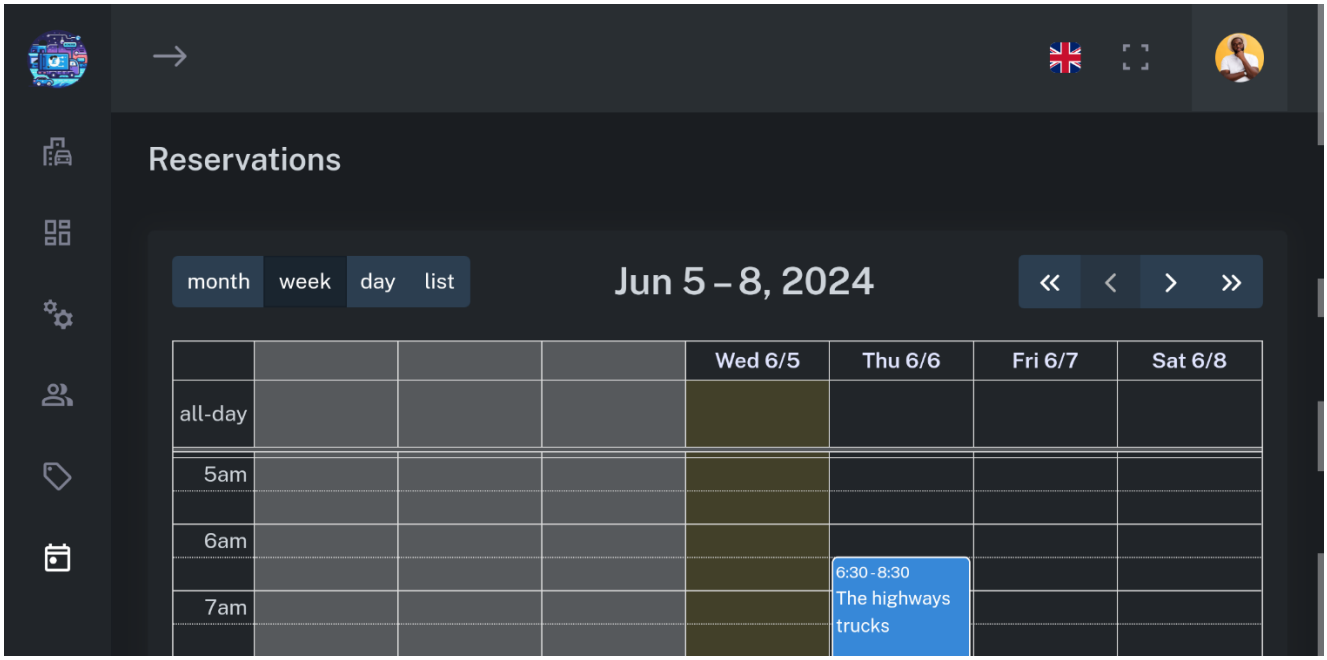


Рисунок 5 - Календар резервацій

.

3.1.2 Апаратний інтерфейс

Оскільки веб-сайт не має призначеного обладнання, прямих апаратних інтерфейсів немає.

3.1.3 Програмний інтерфейс

Веб-додаток повинен бути розроблений з урахуванням сумісності з усіма сучасними браузерами, такими як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge тощо. Використання технологій HTML5, CSS3 та JavaScript ES6 допоможе забезпечити сумісність та підтримку різних браузерів. Для роботи з мобільною версією потрібно буде її завантажити через Play Market та App Store.

3.1.4 Комунікаційний протокол

Зарядна станція взаємодіє через протокол OCPP та веб-сокети з відповідним сервісом обробки підключень. RabbitMQ підтримує протокол AMQP. Протокол OCPP є стандартним протоколом для обміну повідомленнями з зарядними станціями. Взаємодія між сервісами – за допомогою gRPC (окрім випадків, де комунікація відбувається асинхронно за допомогою RabbitMQ). В

3.1.5 Обмеження пам’яті

Веб-сайт не може безпосередньо контролювати кількість доступної пам'яті на комп'ютері користувача. Прямого обмеження пам'яті комп'ютера для користування веб-сайтом немає.

3.1.6 Операції

Для розробки веб-додатку використовується архітектурний стиль REST, що базується на принципах HTTP-протоколу. Основні операції, які можна виконувати з використанням REST, включають наступні:

- GET - для отримання ресурсу або його представлення;
- POST - для створення нового ресурсу;
- PUT - для оновлення існуючого ресурсу або створення нового, якщо він не існує;
- DELETE - для видалення ресурсу.

REST також використовує URL-шаблони для ідентифікації ресурсів та параметризації запитів.

3.1.7 Функції продукту

ГФ-1: Управління інформацією; створення депо. Надання власникам мереж депо можливості керувати інформацією; створювати свої депо.

ГФ-2: Створення співробітника. Надання адміністрації депо можливості створювати користувачів (співробітників/адміністраторів) для управління мережами зарядних станцій.

ГФ-3: Управління стану заряду станцій. Забезпечення інструментів для управління роботи станцій.

ГФ-4: Моніторинг стану депо в реальному часі. Надати можливість слідкувати за станом депо/зарядними станціями в реальному часі.

ГФ-5: Налаштування споживання електроенергії в рамках депо. Можливість глобального налаштування обмежень споживання енергії в заданих часових інтервалах депо.

ГФ-6: Статистика та звітність. Збір та надання статистичних даних для аналізу ефективності роботи зарядних станцій.

ГФ-7: Створення профілів зарядки. Конфігурація індивідуальних та групових профілів для контролю потужності зарядних станцій.

ГФ-8: Управління резервацією зарядних станцій. Організація системи резервації для ефективного використання станцій.

ГФ-9: Зміна мови додатку. Надані інструменту для зміни мови додатку

ГФ-10: Запрошення існуючого співробітника в депо. Реалізувати можливість надсилати запрошення наявним користувачам в системі приєднатися до депо.

3.2 Властивості програмного продукту

Користувачі розроблюваної програмної системи будуть широко розпорошені. Для доступу до системи необхідне стабільне інтернет-з’єднання. Оскільки дані будуть зберігатися віддалено на сервері, максимальний час відгуку встановлено до хвилини. Доступ до різноманітних функцій буде залежати від типу користувача. Цілісність та безпека зберігання даних буде забезпечуватися базою даних.

Програмна система має мобільну версію.

3.3 Атрибути програмного продукту

3.5.1 Надійність

3.5.1.1 Система повинна мати високий рівень надійності, користувач повинен отримувати тільки перевірені дані.

3.5.1.2 В разі виникнення помилки, система не повинна припиняти свою роботу, користувач повинен побачити зрозуміле йому повідомлення з описом проблеми, що виникла (без коду помилки і інформації, яку зрозуміти зможе тільки розробник).

3.5.2 Доступність

3.5.2.1 Користуватися веб-додатком зможуть всі люди, що володіють українською або англійською мовою.

3.5.2.2 Для отримання доступу до мобільної версії потрібно її завантажити.

3.5.2.3 Тільки авторизовані користувачі можуть отримати доступ до функцій продукту.

3.5.2.4 Тільки адміністратору депо доступні функції видалення / додавання / зміни інформації про працівників.

3.5.3 Безпека

3.5.3.1 Кожен обліковий запис в системі буде захищено паролем, який встановлює користувач. Надійний пароль має складатися з більш ніж з 6 символів і обов'язково містити в собі принаймні одну цифру.

3.5.3.2 Увійти в обліковий запис можна тільки після введення правильних пароля та логіна.

3.5.3.3 Усі дані компанії мають зберігатися в окремій захищеній базі.

3.5.3.4 Вкладки мають бути доступні в залежності від дозволів ролі співробітника.

3.5.4 Супровід

3.5.4.1. Система повинна бути розроблена таким чином, щоб була можливість додавання нових функцій в майбутньому.

3.5.4.2. Система повинна бути розділена на частини (серверна, клієнтська, мобільно версії), щоб виправлення помилок займало якомога менше часу.

3.5.5 Переносимість

3.5.5.1 Веб-додаток повинен запускатися в усіх сучасних браузерах незалежно від пристрою, який використовує користувач.

3.4.5.2 Мобільна версія повинен працювати на Android та iOS пристроях.

3.5.6 Продуктивність

3.5.6.1 Час відгуку на перехід між вкладками веб-додатку повинен бути менше 1 секунди.

3.5.6.2 Після натискання кнопки "Додати співробітника" співробітник має бути одразу доданий у список працівників, йому має прийти лист для встановлення паролю на пошту протягом 5 хвилин.

3.4 Вимоги бази даних

В якості сховища даних було обрано СКБД Microsoft SQL Server, яка має наступні переваги:

масштабованість та продуктивність: MS SQL Server демонструє високу продуктивність та масштабованість. Він відмінно працює на великих корпоративних серверах, так і на десктопах або дрібних серверах, дозволяючи ефективно керувати великими обсягами даних;

інтеграція з іншими продуктами Microsoft: Оскільки MS SQL Server — продукт Microsoft, він бездоганно інтегрується з іншими продуктами Microsoft, такими як Windows Server, .NET, SharePoint, а також з Microsoft Office;

безпека даних: MS SQL Server має потужні механізми безпеки. Він пропонує різноманітні функції для контролю доступу, шифрування даних та аудиту, що дозволяє компаніям відповідати вимогам регуляторів.

3.5 Інші вимоги

Система не повинна дозволяти незареєстрованим користувачам отримувати доступ до функцій сервісу. Лендінг сторінка має бути окремою частиною, яка буде загальнодоступною, а веб-сайт з функціональністю доступний лише при переході на авторизацію. Використання зображень, фото та інших матеріалів на сайті не повинно порушувати авторських прав. Інтерфейс веб додатку повинен змінюватися (підганяти розміри) під пристрої з різними розмірами екрану. Інтерфейс повинен бути інтуїтивно зрозумілий і приємний у використанні для людей будь-яких вікових категорій.

4. Додаткові матеріали

4.1 Діаграми баз даних

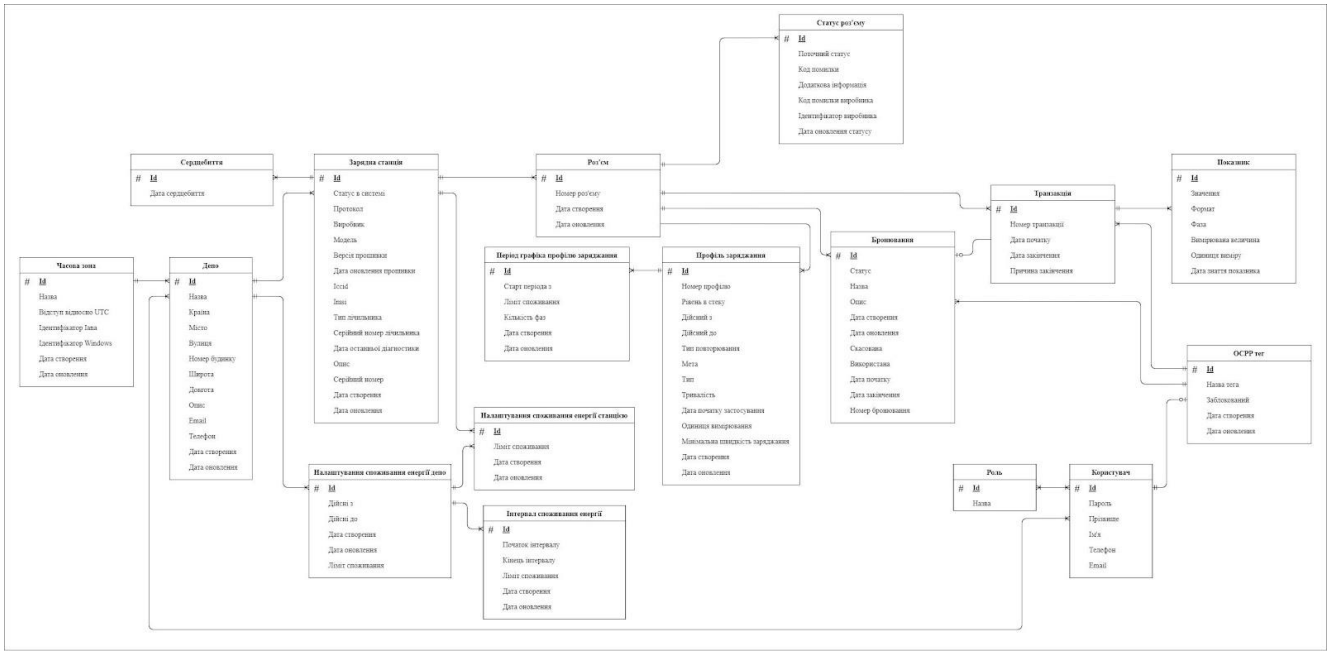


Рисунок 4 – ER-діаграма бази даних

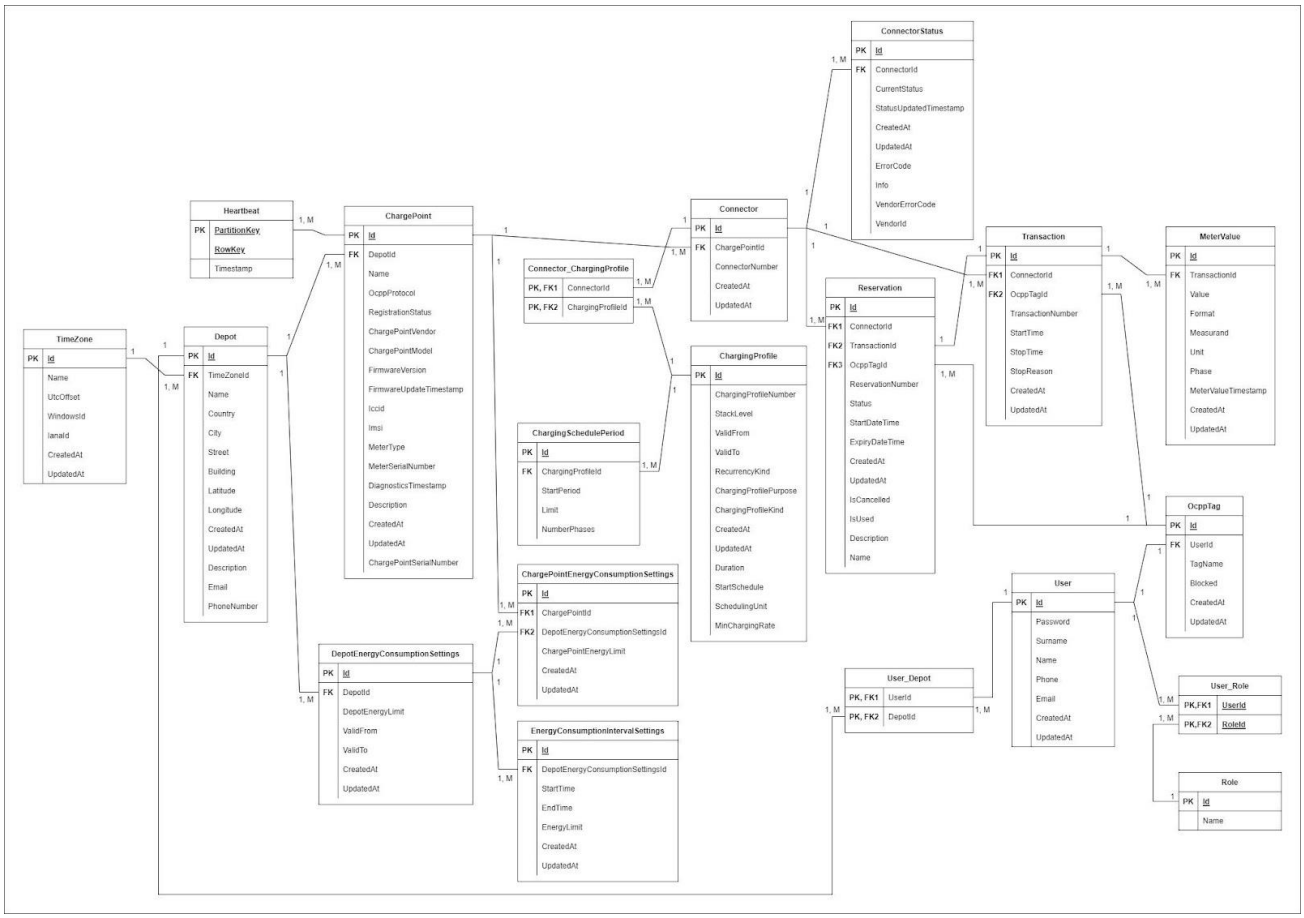


Рисунок 5 – Логічна модель бази даних

4.2 Діаграма прецедентів

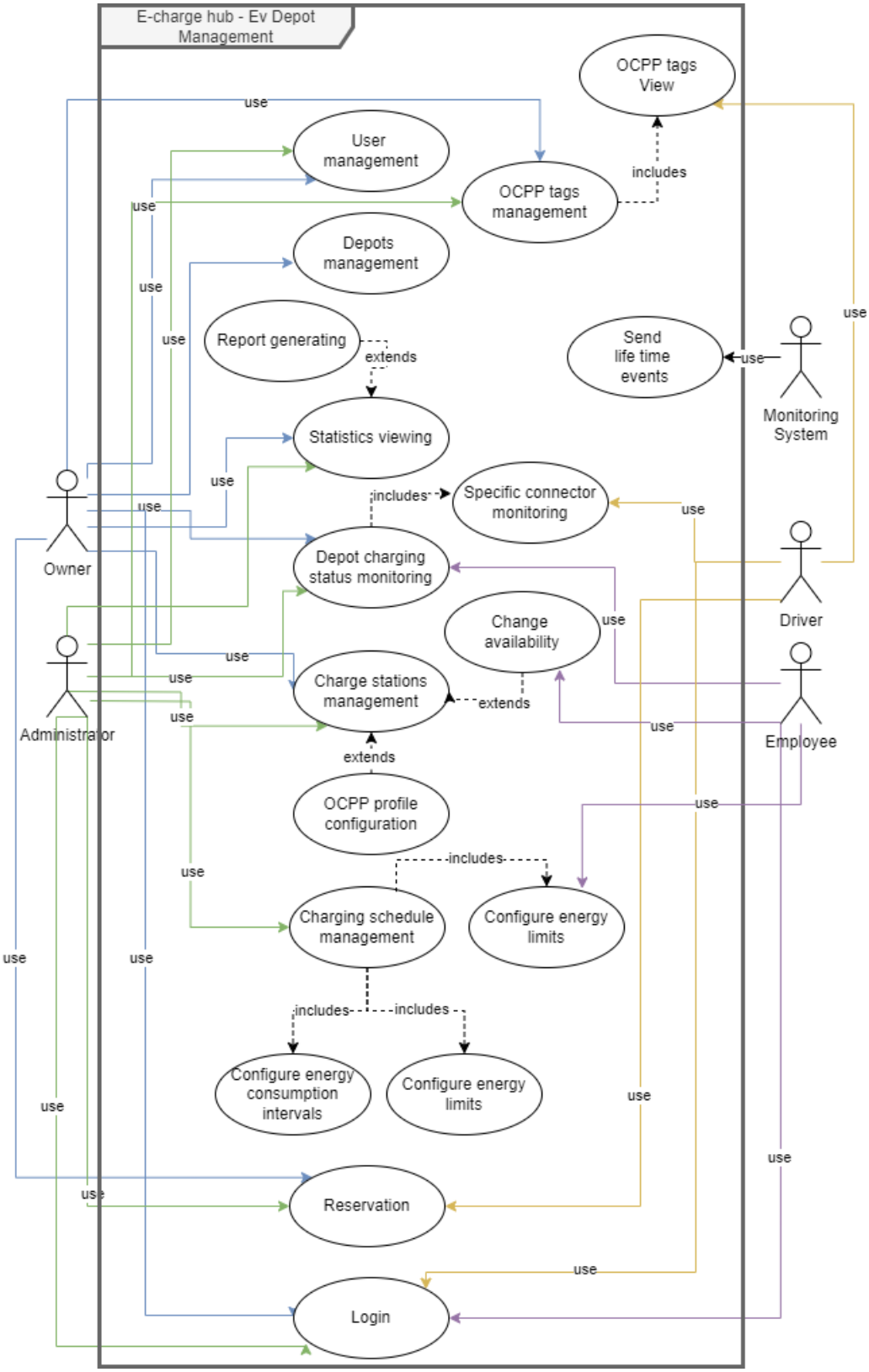


Рисунок 6 – Діаграма прецедентів

4.3 Діаграма діяльності

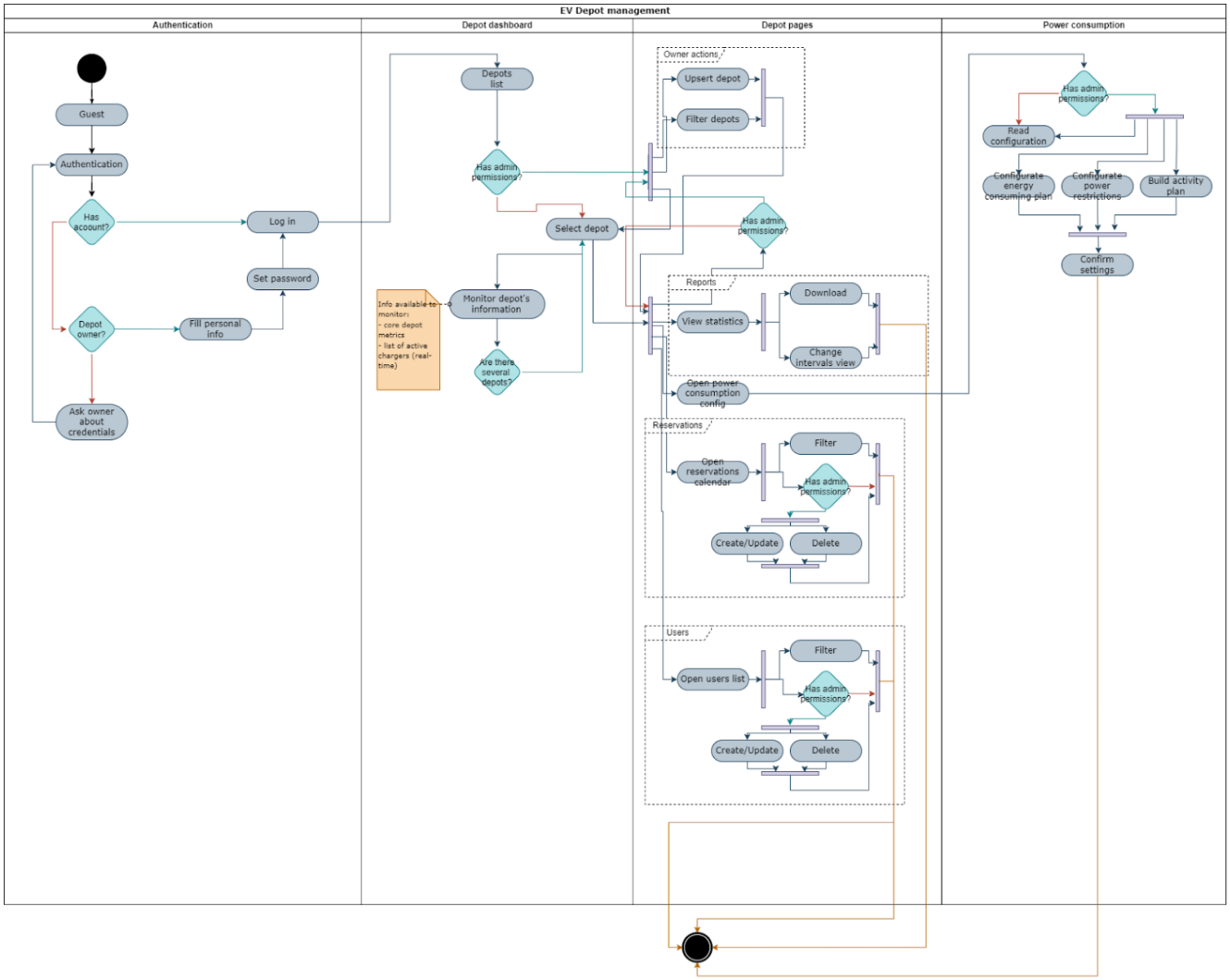


Рисунок 7 – Діаграма діяльності системи для управлінням споживання електроенергії депо електротранспорту