УДК 004.75

Н. Є. Русакова, канд. техн. наук, доц.; Д. О. Горкун; Є. Е. Чубаров; Д. А. Рубель; Т. С. Налєскіна

МІКРОСЕРВІСНА АРХІТЕКТУРА ДЛЯ КЕРУВАННЯ ЗАРЯДНИМИ СТАНЦІЯМИ ЕЛЕКТРОМОБІЛІВ: ВПРОВАДЖЕННЯ ONION АРХІТЕКТУРИ ТА ОСРР ДЛЯ МАСШТАБОВАНОСТІ Й ВІДМОВОСТІЙКОСТІ

В статті розглянуто розробку програмної системи для централізованого керування та моніторингу зарядних станцій електромобілів. Метою дослідження є розробка системи оптимізації витрат електроенергії, використання стандартної для серверної частини Onion архітектури у клієнтській частині застосунку, що дасть можливість краще структурувати код і забезпечити гнучкість та масштабованість, реалізацію гнучких алгоритмів для роботи з часовими інтервалами споживання енергії, створення інтеграції з сучасними технологіями для обробки даних у реальному часі.

В роботі здійснено реалізацію серверної та клієнтської частин системи управління зарядними станціями електромобілів. Система успішно розгорнута і протестована з використанням Test-Driven Development підходів, демонструючи високу ефективність та відповідність поставленим вимогам. 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 проблеми, що з'являються, 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. Реалізація основних функцій охоплює управління зарядними станціями, резервацію, моніторинг та аналітику, обмеження споживання енергії, роботу з таблицями та локалізацію, зокрема, створено зручний інтерфейс для налаштування depo, який включає можливості як введення загальних, так і точкових обмежень.

Ключові слова: *електромобілі, зарядні станції, мікросервісна архітектура.*

Вступ

Глобальний ринок електромобілів продовжує стрімко зростати. Згідно з даними Міжнародного енергетичного агентства (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 автомобілів. 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. Також 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.

На ринку вже представлено низку програмних рішень для моніторингу та управління зарядними станціями електричних транспортних засобів (ЕТЗ). Серед них GreenFlux [2], Ampeco [3], Driivz [4] тощо. 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 [5], які передають дані до централізованої серверної інфраструктури. Платформа, у свою чергу, надає доступ до даних через веб-інтерфейс або мобільні додатки. Порівняльний аналіз наявних рішень на ринку наведено у таблиці 1.

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 актуальною стає задача розробки більш гнучких, функціонально розширених систем управління зарядною інфраструктурою ЕТЗ.

Таблиця 1

Порівняння наявниих рішень управління зарядними станціями ЕТЗ

Назва	Опис	Ключові можливості	Цільова аудиторія	Додатково
GreenFlux	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
Ampeco	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	Ізраїльська компанія, що надає хмарну платформу для управління зарядною інфраструктурою електромобілів	Моніторинг, централізоване управління станціями, тарифікація	Оператори мереж зарядних станцій, електромережеві компанії, прокат електроавто	OCPP 1.6 і 2.0.1, звітування та аналітика
ChargeLab	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

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 розроблено комплексне програмне рішення, яке включає такі

ключові компоненти: емулятор зарядних станцій, серверна частина додатку, веб-застосунок для користування системою, мобільний додаток для взаємодії з зарядними станціями. Реалізація цих компонентів дозволяє задовольнити потреби всіх ключових учасників – власників ЕТЗ, операторів станцій та державних органів – та вирішити наявні проблеми у сфері моніторингу та управління зарядною інфраструктурою.

Проєктування архітектури ПЗ

Враховуючи висунуті вимоги до програмної системи, а також дані, отримані в ході аналізу предметної області, було обрано застосувати мікросервісний [6] підхід.

Розробку мікросервісів наведено на мові C# з використанням платформи .NET 8 та фреймворку ASP.NET Core. Вибір саме цих технологій обумовлений підтримкою широкого спектру бібліотек, включно з бібліотеками для роботи з хмарними технологіями, gRPC та RabbitMQ. Зарядна станція взаємодіє через протокол OCPP та веб-сокети з відповідним сервісом обробки підключень (chargepoint websocket service), що відповідає за обмін повідомленнями між станціями та іншими сервісами. Взаємодія між сервісом обробки підключень станцій та іншою частиною системи відбувається асинхронно за допомогою брокера повідомлень RabbitMQ [8].

Протокол OCPP [9] є стандартним протоколом для обміну повідомленнями з зарядними станціями, що дозволяє системі уникати залежності від коректних виробників станцій. Цей протокол налічує перелік повідомлень для взаємодії зі станціями. Таким чином, в залежності від типу повідомлення, що було відправлено зарядною станцією, воно буде скеровано до одного з сервісів:

– charge points service – сервіс для роботи з зарядними станціями. Обробляє повідомлення про підключення зарядних станцій до системи (BootNotificationRequest);

– OCPP tags service – сервіс для роботи з авторизаційними тегами, який також обробляє запити на авторизацію (AuthorizeRequest);

– connectors service – сервіс для роботи роз'ємами (конекторами) зарядних станцій. В контексті обробки повідомлень від зарядних станцій цей сервіс відповідальний за обробку зміни статусів роз'ємів (StatusNotificationRequest);

– transactions service – сервіс обробки транзакцій. Сюди надсилаються запити на початок (StartTransactionRequest) та завершення (StopTransactionRequest) транзакції, а також показники спожитої енергії в ході транзакції (MeterValuesRequest);

– reservations service – сервіс обробки резервацій, що відповідає за бронювання та скасування бронювання конекторів (ReserveNowRequest та CancelReservationRequest);

– charging profiles service – сервіс налаштувань профілів зарядки. В контексті обміну повідомленнями цей сервіс відповідає за встановлення або зміну профілю налаштування процесу заряджання. Також цей сервіс відповідальний за налаштування процесів заряджання в рамках депо;

– heartbeats service – сервіс обробки «серцебиття» станції. Кожна станція під час своєї роботи посилає «серцебиття», сигналізуючи про те, що вона онлайн.

Після обробки повідомлень згідно з бізнес-логікою сервіси надсилають повідомлення-відповіді на сервіс обробки підключень зарядних станцій, який передає повідомлення самій станції. Також окрім сервісів, що безпосередньо взаємодіють з зарядними станціями, є ще наступні:

– depots service – сервіс для роботи з депо зарядних станцій;

– user service – сервіс керування користувачами;

– aggregator – імплементація патерну Aggregator. Цей мікросервіс потрібен для того, щоб збирати комплексні дані з різних мікросервісів і далі відправляти ці дані до зовнішніх споживачів. Наприклад, з різних сервісів можна зібрати дані про депо, зарядні станції цього депо та налаштування заряджання. Таким чином, між фронтендом та бекендом знижується кількість запитів;

– gateway – цей мікросервіс реалізує єдину точку входу в API, що побудоване з використанням мікросервісної архітектури. Як можна побачити з діаграми, він буде побудований на основі бібліотеки Ocelot, яка дозволяє швидко створювати API Gateway та зручно його конфігурувати;

– signalR Hub – сервіс для роботи з SignalR для забезпечення доставки повідомлень в реальному часі за допомогою веб-сокетів, так як під час роботи системи користувача необхідно нотифікувати про такі події, як підключення нової станції, перевищення лімітів споживання енергії та ін.

На рис. 1 можна побачити схему архітектури системи. Взаємодія між клієнтською та серверною частинами відбувається за допомогою контролерів RESTful, а взаємодія між сервісами – за допомогою gRPC (окрім випадків, де комунікація відбувається асинхронно за допомогою RabbitMQ). В якості сховища даних було обрано СКБД Microsoft SQL Server.

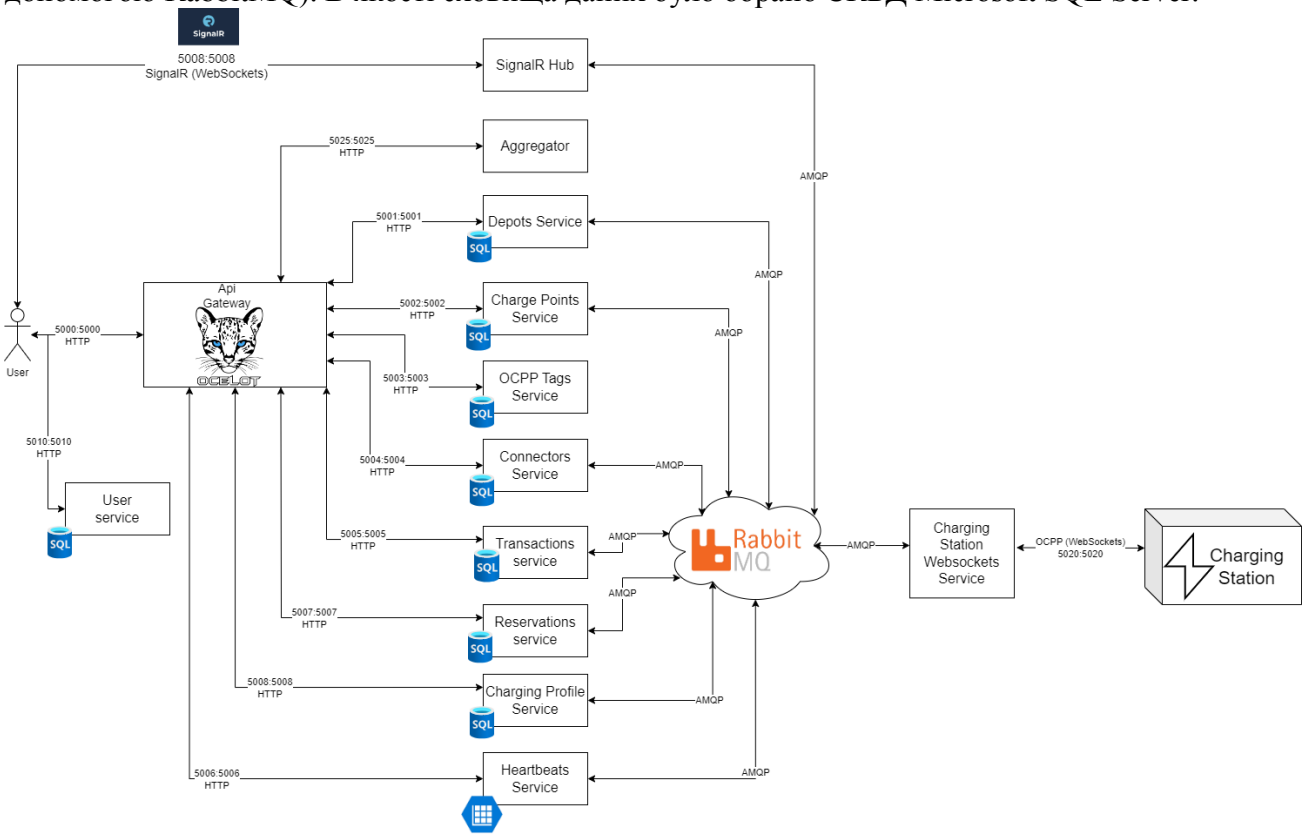

Рис. 1. Архітектура серверної частини програмної системи

Слід зазначити, що всі сервіси мають одне спільне джерело даних, але кожен сервіс взаємодіє лише з тією частиною БД, яка відноситься до зони відповідальності сервісу, тобто таким чином буде реалізовано схему «Shared Instance»[7]. Єдине джерело даних забезпечує можливість використання транзакцій ACID для забезпечення цілісності даних, більш легку розробку та менші фінансові витрати. Головним мінусом такого підходу є те, що з'являється слабке місце у роботі системи – якщо база даних виходить з ладу, то виходить з ладу вся система. Для того, щоб нівелювати цей недолік, було прийнято рішення розгорнути базу даних в хмарі Microsoft Azure. Azure забезпечує чудову підтримку Microsoft SQL Server, надаючи механізми автоматичного масштабування, реплікації даних, створення автоматичних бекапів.

Також окремо хотілося б звернути увагу на сервіс heartbeats. Так як за замовчуванням «серцебиття» відправляється кожні декілька хвилин і враховуючи те, що система потенційно працюватиме з великою кількістю станцій, то для зменшення навантаження на основну БД та економії коштів, було прийнято рішення зберігати дані про «серцебиття» в Azure Table Storage в рамках Azure Storage Account.

UML проєктування програмного забезпечення та бази даних

Функціонал 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 керуванням, користувачами та депо; переглядати статистику, яка включає в себе генерацію звітів Також може моніторити стан зарядки депо, що передбачає і моніторинг конкретного депо. Керувати зарядними станціями, що включають в собі зміну доступності та налаштування профілів OCPP. Керувати графіками зарядки, що передбачає налаштування інтервалів споживання та налаштування енергетичних лімітів. Має доступ до менеджменту OCPP тегів, що включає в собі перегляд OCPP тегів, та до резервації та авторизації.

Адміністратор (Administrator) може займатись керуванням користувачами та депо; переглядати статистику, що включає генерацію звітів. Також може моніторити стан зарядки депо, що передбачає і моніторинг конкретного депо. Керувати зарядними станціями, що включають в собі зміну доступності та налаштування профілів OCPP. Керувати графіками зарядки, що передбачає налаштування інтервалів споживання та налаштування енергетичних лімітів. Має доступ до менеджменту OCPP тегів, що включає в собі перегляд OCPP тегів, та до резервації та авторизації.

Моніторингова система (Monitoring system) відправляє події в реальному часі.

Водій (Driver) може моніторити конкретний роз'єм, переглядати OCPP теги. Має доступ до резервації та авторизації. Працівник (Employee) може моніторити стан зарядки депо, займатись налаштуванням профілів OCPP та енергетичних лімітів. Має доступ до авторизації. Зобразімо необхідний, відповідно до бізнес потреб, функціонал продукту використовуючи Use Case діаграму (рис. 2).

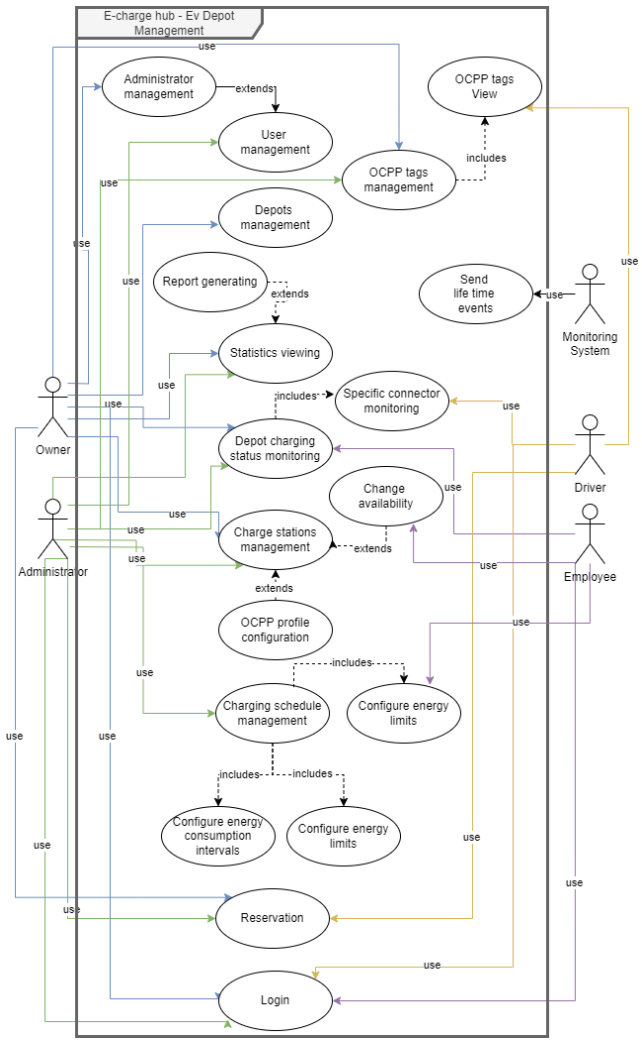

Рис. 2. Діаграма прецедентів

На основі даних, отриманих в ході аналізу предметної області, висунутих вимог, а також особливостей протоколу OCPP, було побудовано інфологічну модель даних.

Розглянемо кожну сутність окремо. Виділимо атрибути сутності «Часова зона»:

– #Id – ідентифікатор зони, первинний ключ;

– назва – назва часової зони;

– відступ відносно UTC – зсув у годинах відносно UTC – Coordinated Universal Time;

– ідентифікатор IANA – ідентифікатор часової зони відповідно до Internet Assigned Numbers Authority;

– ідентифікатор Windows – ідентифікатор часової зони в системі Windows;

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

Сутність «Депо» включає інформацію про депо, що знаходиться в одній з часових зон:

– #Id – ідентифікатор депо, первинний ключ;

– назва – назва депо;

– країна, місто, вулиця, номер будинку, широта та довгота – географічні дані про депо;

– опис – додатковий опис депо за необхідності;

– email та телефон – контактні дані депо;

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

До сутності «Зарядна станція» належать такі атрибути:

– #Id – ідентифікатор автомобіля, первинний ключ;

– назва – назва станції;

– опис – додатковий опис за необхідності;

– в системі – станція може мати різні статуси під час роботи з системою, наприклад, до фізичного підключення станції, запис в БД про цю станцію міститиме статус «Створено»;

– виробник – виробник станції;

– модель – модель станції;

– серійний номер станції;

– версія прошивки;

– дата оновлення прошивки;

– ICCID (Integrated Circuit Card Identifier) – унікальний ідентифікатор, що присвоюється кожній сім-карті. У контексті зарядної станції для електромобілів ICCID може використовуватися, якщо зарядна станція підключається до центральної системи управління через мобільну мережу;

– IMSI (International Mobile Subscriber Identity) – це унікальний номер, який використовується для ідентифікації абонентів у мобільних телефонних мережах. У контексті зарядних станцій для електромобілів IMSI використовується в тому випадку, якщо станція оснащена SIM-картою для забезпечення зв'язку з центральною системою керування через мобільний інтернет;

– тип лічильника електроенергії;

– серійний номер лічильника;

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

До сутності «Роз'єм» належать такі атрибути:

– #Id – ідентифікатор роз'єму, первинний ключ;

– номер роз'єму – числовий номер роз'єму в рамках зарядної станції;

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

Сутності «Статус роз'єму» належать такі атрибути:

– #Id – ідентифікатор запису про статус;

– поточний статус – поточний статус роз'єму відповідно до протоколу OCPP;

– код помилки – в разі помилки станція може надіслати її код разом зі статусом;

– додаткова інформація – за необхідністю станція також може надіслати додаткову інформацію;

– дата оновлення статусу;

– код помилки виробника та ідентифікатор виробника – в залежності від виробника станції, в разі помилки, станція може надсилати додаткову інформацію про помилку, що є специфічною саме для виробника;

Сутність «Показник» містить інформацію про показники, що отримуються під час заряджання:

– #Id – ідентифікатор запису про показники, первинний ключ;

– значення – значення показника;

– формат – в якому форматі дані передано;

– фаза – деякі станції під час передачі деяких показників передають також інформацію про фазу вказуючи, як слід інтерпретувати виміряне значення. (Наприклад, між L1 і нейтраллю (L1-N));

– вимірювана величина – потужність, сила току, напруга і т. ін.;

– одиниця виміру – вольти для напруги, ампери для сили току, вати або кіловати для потужності і т. ін.;

– дата зняття показника.

Сутності «Транзакція» належать такі атрибути:

– #Id – ідентифікатор транзакції, первинний ключ;

– номер транзакції – в контексті роботи протоколу OCPP, кожна транзакція мусить мати

унікальний числовий номер;

– дата початку транзакції;

– дата закінчення транзакції;

– причина закінчення – додаткова інформація про причину закінчення, наприклад, успішне закінчення чи закінчення через технічний збій.

Сутність «Бронювання» включає інформацію про бронювання роз'ємів:

–#Id – ідентифікатор бронювання;

– назва та опис – додаткові необов'язкові параметри;

– дата початку;

– дата закінчення;

– статус – статус в системі. Коли запис про бронь додається, то бронь має статус «створена», після відправки запиту на станцію – «запит відправлено», після підтвердження станцією – «підтверджена»;

– скасована – прапорець для індикації того, чи було бронь скасовано;

– використана – прапорець для індикації того, чи було бронь використано;

– номер бронювання – як і з транзакціями, в контексті роботи протоколу OCPP, кожне бронювання мусить мати унікальний числовий номер;

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

До сутності «Серцебиття», що показує, чи є станція онлайн, належать такі атрибути:

– #Id – ідентифікатор запису, первинний ключ;

Дата «серцебиття».

Сутності «OCPP тег», що містить в собі дані про теги для ідентифікації користувача станцією, належать такі атрибути:

– #Id – ідентифікатор тега, первинний ключ;

– назва тегу;

– заблокований – індикатор того, чи було заблоковано тег системою;

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

Сутність «Профіль заряджання» містить в собі налаштування процесу заряджання для роз'єму або всієї зарядної станції або так званий «профіль». До цієї сутності належать такі атрибути:

– #Id – ідентифікатор профілю, первинний ключ;

– номер профілю – протокол OCPP вимагає, щоб кожний профіль мав унікальний числовий номер;

– дійсний з – дата, з якої налаштування є дійсними;

– дійсний до – дата, до якої налаштування є дійсними;

– тривалість – в який проміжок часу буде працювати цей профіль, наприклад, профіль може бути дійсний місяць, але використовуватись кожен день по годині;

– рівень в стеку – значення, що визначає рівень в ієрархічному стеку профілів. Вищі значення мають перевагу над нижчими. Найнижчий рівень 0;

– мета – визначає призначення розкладу, переданого цим повідомленням. Може мати наступні значення: ChargePointMaxProfile – конфігурація для максимальної потужності або струму, доступного для всієї зарядної станції; TxDefaultProfile – профіль за замовчуванням. Коли розпочинається нова транзакція, цей профіль мусить використовуватися крім тих випадків, коли транзакція була розпочата віддалено запитом RemoteStartTransaction.req, в якому було передано інший профіль; TxProfile – профіль для поточної транзакції. Є валідним до закінчення поточної транзакції;

– тип – тип профілю. Може бути абсолютним (періоди розкладу відносяться до фіксованого моменту часу, визначеного в розкладі), відносним (періоди розкладу відносяться до конкретної ситуації початкової точки (наприклад, початку транзакції), яку визначає зарядна станція) та

повторюваним (розклад періодично перезапускається в перший період розкладу).

– дата початку застосування – початкова точка абсолютного розкладу. Якщо його немає, графік буде відносним до початку заряджання.

– тип повторюваності – вказує на початкову точку повторення. Може бути щоденним і щотижневим. Щоденний вказує на те, що розклад перезапускається кожні 24 години в той самий час, що вказаний в полі «дата початку застосування». Щотижневий вказує на те, що розклад перезапускається кожні 7 днів у той самий час і день тижня, що вказані в полі «дата початку застосування»;

– одиниця вимірювання – визначає в яких одиницях вимірювання визначається ліміт споживання енергії. Це можуть бути вати або ампери;

– мінімальна швидкість заряджання – опціональний параметр. Мінімальна швидкість зарядки, яку підтримує електромобіль. Одиниця вимірювання визначається значенням з поля «одиниця вимірювання». Цей параметр призначений для використання алгоритмом заряджання зарядної станції для оптимізації розподілу потужності у випадку, якщо процес заряджання є неефективним при нижчих рівнях заряджання;

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

Сутність «Період графіка профілю заряджання» включає інформацію про періоди, які входять до профілю заряджання, і містить такі атрибути:

– #Id – ідентифікатор періоду, первинний ключ;

– початок періоду – кількість секунд, що визначають початок періоду відносно початку розкладу;

– ліміт – максимальна потужність або сила струму, що може бути використана;

– кількість фаз – додатковий параметр – кількість фаз, які можна використовувати для зарядки (три за замовчуванням);

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

Використання профілів зарядки дозволяє гнучко налаштовувати параметри заряджання для кожного роз'єму зарядної станції, а також задавати ліміти сили току та потужності.

Сутність «Налаштування споживання енергії депо» призначена для того, щоб налаштувати кількість Ват-годин, яка може бути спожита депо, і містить такі поля:

– #Id – ідентифікатор налаштувань, первинний ключ;

– «дійсні з» та «дійсні до» – з якої дати та по яку налаштування є дійсними;

– ліміт споживання енергії – скільки енергії може спожити депо;

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

Якщо за допомогою профілів зарядки можна обмежувати потужність та силу току, то сутність «Налаштування споживання енергії станцією» дозволяє налаштовувати кількість спожитих Ват-годин, тобто енергії, в рамках лімітів споживання депо. Ця сутність містить такі поля:

– #Id – ідентифікатор налаштувань, первинний ключ;

– «дійсні з» та «дійсні до» – з якої дати та по яку налаштування є дійсними;

– ліміт споживання енергії – скільки енергії може спожити станція;

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

Сутність «Інтервал споживання енергії» схожа на «Період графіка профілю заряджання», але використовується для налаштування інтервалів споживання енергії (Ват-годин) в рамках ліміту споживання для всього депо. Ця сутність містить такі поля:

– #Id – ідентифікатор інтервалу, первинний ключ;

– «початок інтервалу» та «кінець інтервалу» – початок та кінець інтервалу відповідно;

– ліміт споживання енергії – скільки енергії може бути спожито протягом цього інтервалу;

– дата створення та дата оновлення – технічні дані про те, коли було додано або оновлено запис.

Сутність «Користувач» містить інформацію про користувачів системи:

– #Id – ідентифікатор користувача, первинний ключ;

– пароль – пароль користувача;

– прізвище – прізвище користувача;

– ім'я – ім'я користувача;

– телефон – телефон користувача;

– email – електронна пошта користувача.

До сутності «Роль» належать такі атрибути:

– #Id – ідентифікатор ролі, первинний ключ;

– назва – назва ролі, до якої належить користувач.

Розглянувши детально сутності, можна переходити до опису зв'язків між ними. Між сутностями «Часова зона» і «Депо» зв'язок типу «один-до-багатьох», тому що в одній часовій зоні може знаходитись декілька депо.

Аналогічно моделюємо зв'язок «один-до-багатьох» між сутностями «Депо» і «Зарядна станція», тому що до в одному депо може бути багато зарядних станцій.

Кожна станція може мати декілька роз'ємів, тож між сутностями «Зарядна станція» та «Роз'єм» зв'язок типу «один-до-багатьох».

Протягом часу статус роз'ємів може змінюватись, тобто один роз'єм в ході використання може мати багато статусів, тож зв'язок між сутностями «Роз'єм» та «Статус роз'єму» «один-до-багатьох».

З використанням одного роз'єма може бути виконано багато транзакцій, тож між сутностями «Роз'єм» та «Транзакція» «один-до-багатьох». Також в ході однієї транзакції передається багато показників, тому між сутностями «Транзакція» та «Показник» також зв'язок «один-до-багатьох».

На один роз'єм може бути багато бронювань на різний час, таким чином маємо зв'язок «один-до-багатьох» між сутностями «Роз'єм» та «Бронювання». Одне бронювання відповідає лише одній транзакції, тому сутностями «Транзакція» та «Бронювання» є зв'язок «один-до-одного».

Впродовж своєї роботи станція відправляє багато сигналів про «серцебиття», тобто між сутностями «Зарядна станція» та «Серцебиття» зв'язок «один-до-багатьох».

Стосовно профілів заряджання слід зазначити, що один і той самий профіль заряджання може бути застосовано на різних роз'ємах, а один роз'єм може комбінувати декілька профілів заряджання, тож між сутностями «Профіль заряджання» та «Роз'єм» є зв'язок «багато-до-багатьох». Також в рамках одного профілю заряджання є декілька періодів, що означає наявність зв'язку «один-до-багатьох» між сутностями «Профіль заряджання» та «Період графіка профілю заряджання».

Фактично депо в конкретний момент часу може мати лише одні налаштування споживання енергії, але з метою збереження історичних даних для перегляду того, які налаштування були застосовані до депо, зв'язок між сутностями «Депо» та «Налаштування споживання енергії депо» зв'язок «один-до-багатьох». Так само з налаштуваннями споживання енергії для станції, тобто сутності «Зарядна станція» та «Налаштування споживання енергії станцією» мають зв'язок «один-до-багатьох». Налаштування споживання енергії депо можуть містити декілька інтервалів, що означає наявність зв'язку «один-до-багатьох» у сутностей «Налаштування споживання енергії депо» та «Інтервал споживання енергії».

Говорячи про OCPP-теги, слід зазначити, що за допомогою одного тега можна зробити багато транзакцій та багато бронювань, тобто між сутностями «OCPP тег» та «Транзакція», а також «OCPP тег» та «Бронювання» буде зв'язок «один-до-багатьох».

В свою чергу, користувачі можуть мати лише один тег, тож зв'язок між сутністю «Користувач» та сутністю «OCPP тег» є «один-до-одного». Також користувач одночасно може

мати багато ролей, а одна й та сама роль може бути застосована до багатьох користувачів, тобто між користувачами та ролями є зв'язок «багато-до-багатьох». Так само зв'язок «багато-до-багатьох» є між депо та користувачем, бо користувач одночасно може мати доступ до декількох депо, а до одного депо можуть мати доступ декілька користувачів. Створено логічну схему бази даних (рис. 3), в якій сутності та атрибути перекладено на англійську мову, а також додано проміжні таблиці для зв'язків «багато-до-багатьох».

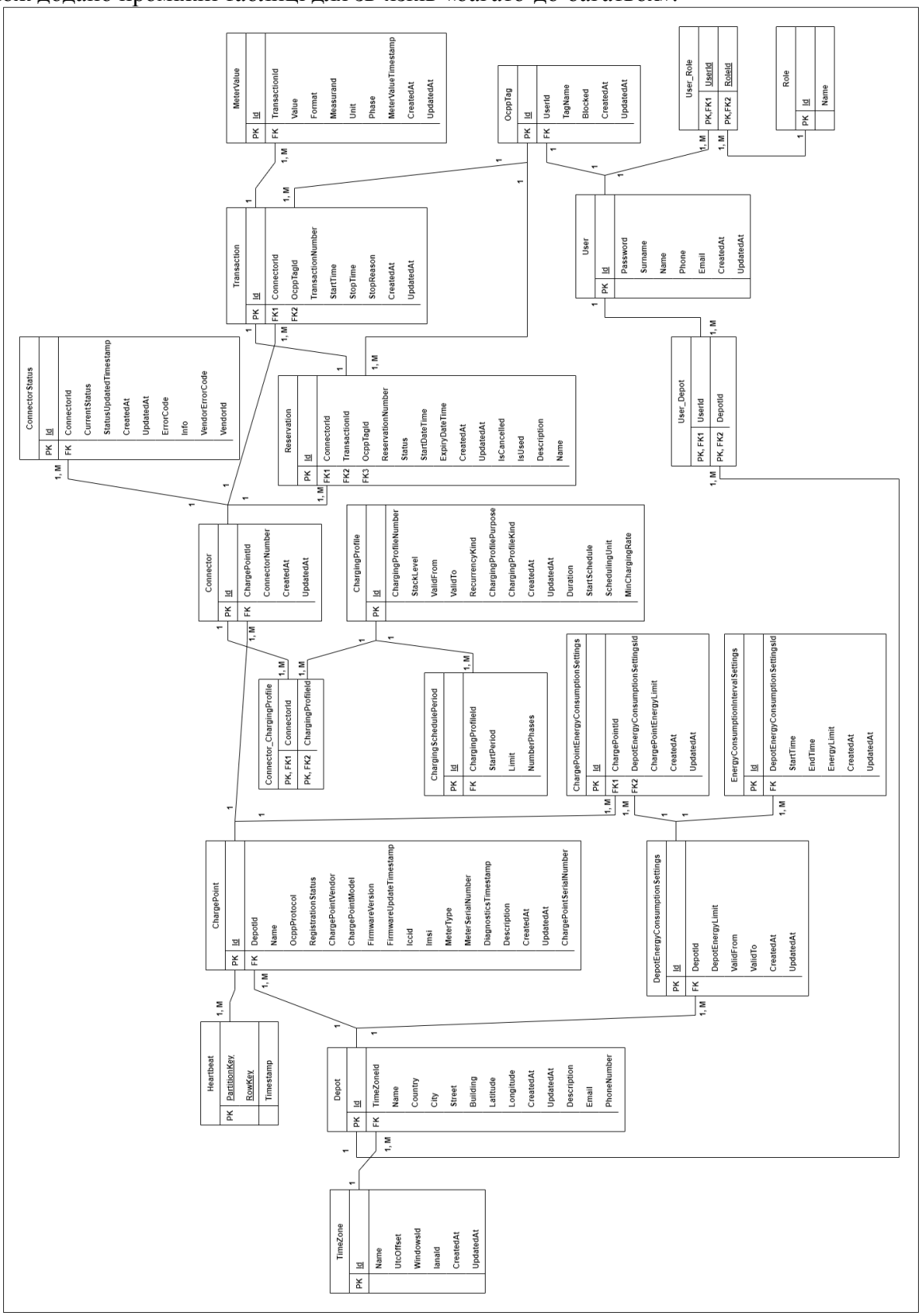

Рис. 3. Логічна модель бази даних

Після того, як систему побудовано, її треба розгорнути в хмарі. Враховуючи, що для імплементації системи використовується мікросервісний підхід, то кращим вибором для розгортання, є використання інструментів контейнеризації, а саме – Docker. Для оркестрації та масштабування використано Kubernetes. Таким чином, Docker забезпечуватиме консистентність незалежно від середовища, в якому працює система (наприклад, dev-середовище або production), а Kubernetes автоматизує розподіл та масштабування контейнерів, забезпечуючи високу доступність та відмовостійкість. Також слід зазначити, що хмара Azure має чудову підтримку обох цих сервісів.

Розробка клієнтської частини

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 зобразимо цей процес (рис. 4).

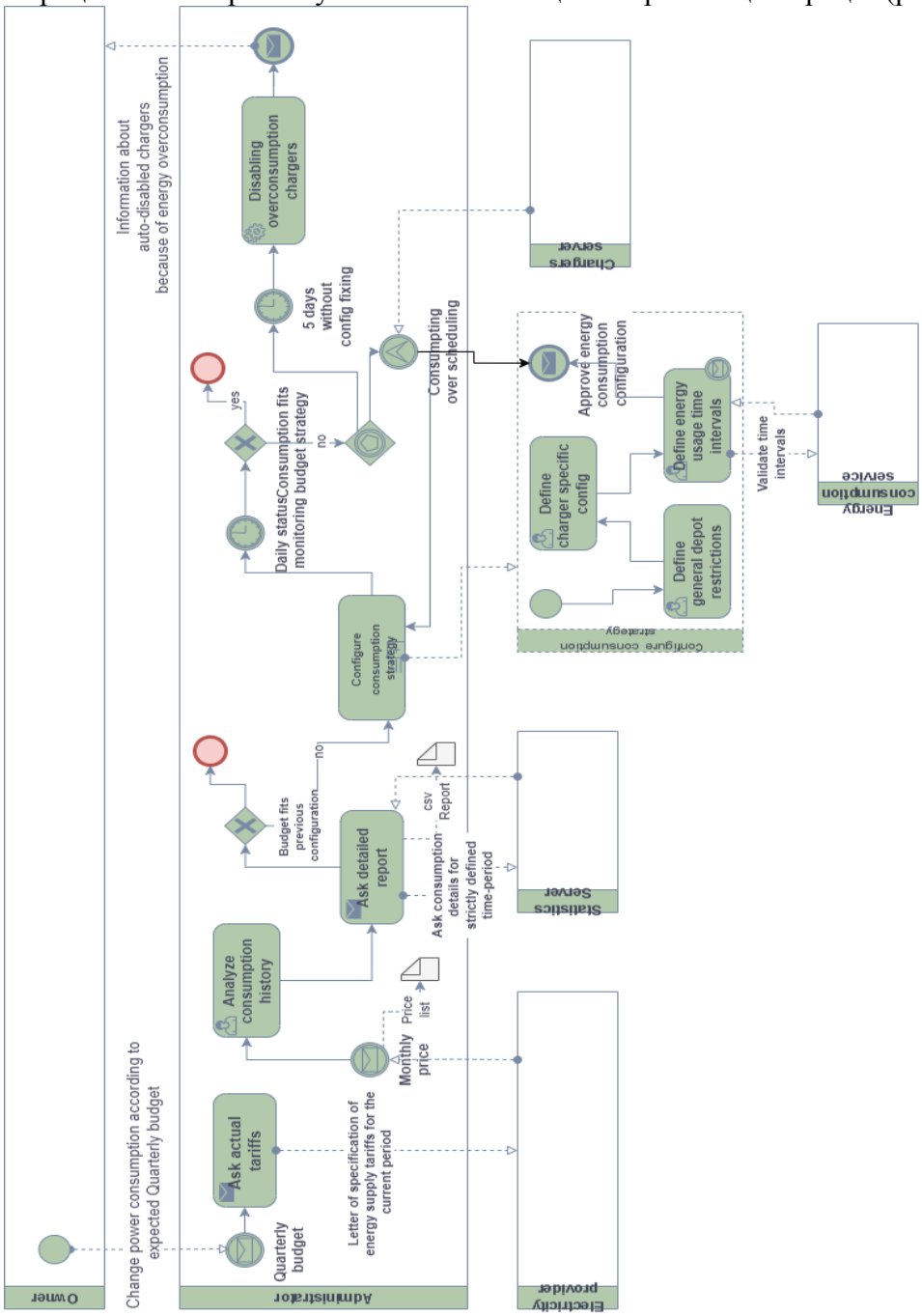

Рис. 4. Процес оновлення конфігурації споживання енергії описана нотацією BPMN

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.

Для досягнення цих цілей було обрано архітектуру Onion [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 (рис. 5.).

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.

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.

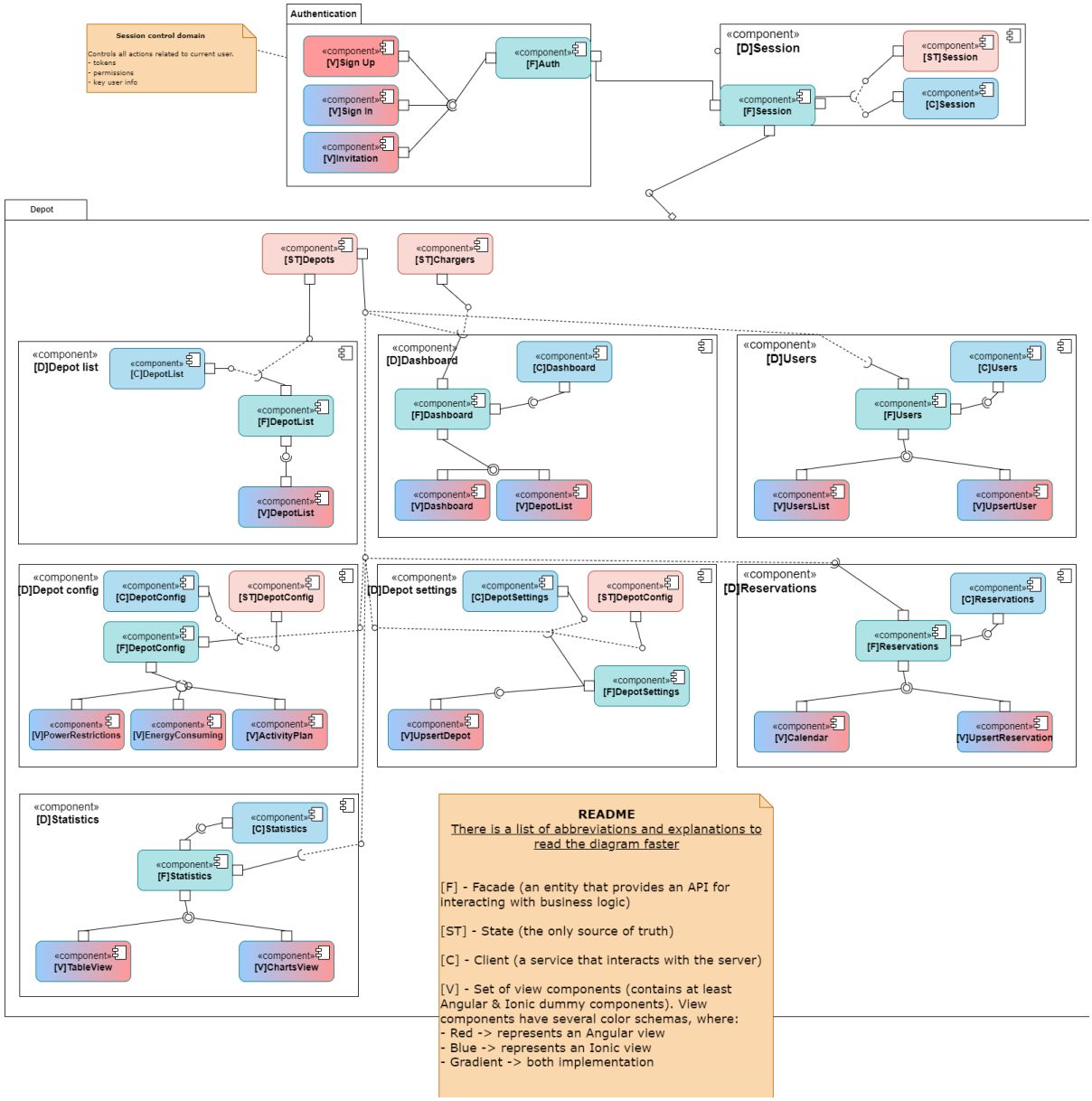

Рис. 5. Діаграма компонентів системи моніторингу зарядних станцій використовуючи Onion архітектуру

Згідно основної мети, яка і є головною цінністю важливим кроком є створення зрозумілого процесу взаємодії користувача з інтерфейсом для налаштування плану споживання електроенергії (рис. 6). З метою успішного виконання цієї задачі спроєктовано алгоритм побудови часових інтервалів споживання електроенергії.

Рис. 6. Процес налаштування плану споживання енергії

Перш за все, виділено функціональні вимоги для алгоритму роботи з часовими інтервалами (табл. 2). 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.

Таблиця 2

Функціональні вимоги для алгоритму управлінням часовими інтервалами споживання електроенергії

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 максимальним и 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 кількість доступної до використання енергії в цьому часовому проміжку	Новостворений інтервал має найвищий пріоритет, що призводить до необхідності змінити наявні часові інтервали. За потреби інтервали будуть: видалені, розбиті на частини, доповнені	Оновлений список з інтервалами, де новостворений інтервал знаходиться у повному обсязі	Якщо початок або кінець плану споживання електроенергії виходить за рамки вказані для депо – план оновлюється згідно налаштувань депо
Редагування наявного інтервалу	У списку наявних інтервалів використовуючи відповідні поля користувач змінює налаштування наявного 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 наявного інтервалу	Користувач має змогу натиснути на відповідну кнопку видалення, якщо плані містить більше 1-го 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. -	Оновлений список інтервалів заповнений з максимальним и доступними депо значеннями використання енергії.	Аналогічно обробці помилок для «Створення нового інтервалу»

Продовження
Таблиця 2

Об'єднання інтервалів	За умови, 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»

Базуючись на визначеній поведінці алгоритму, яка була сформована на основі функціональних вимогах був підготований алгоритм роботи з часовими інтервалами (рис. 7).

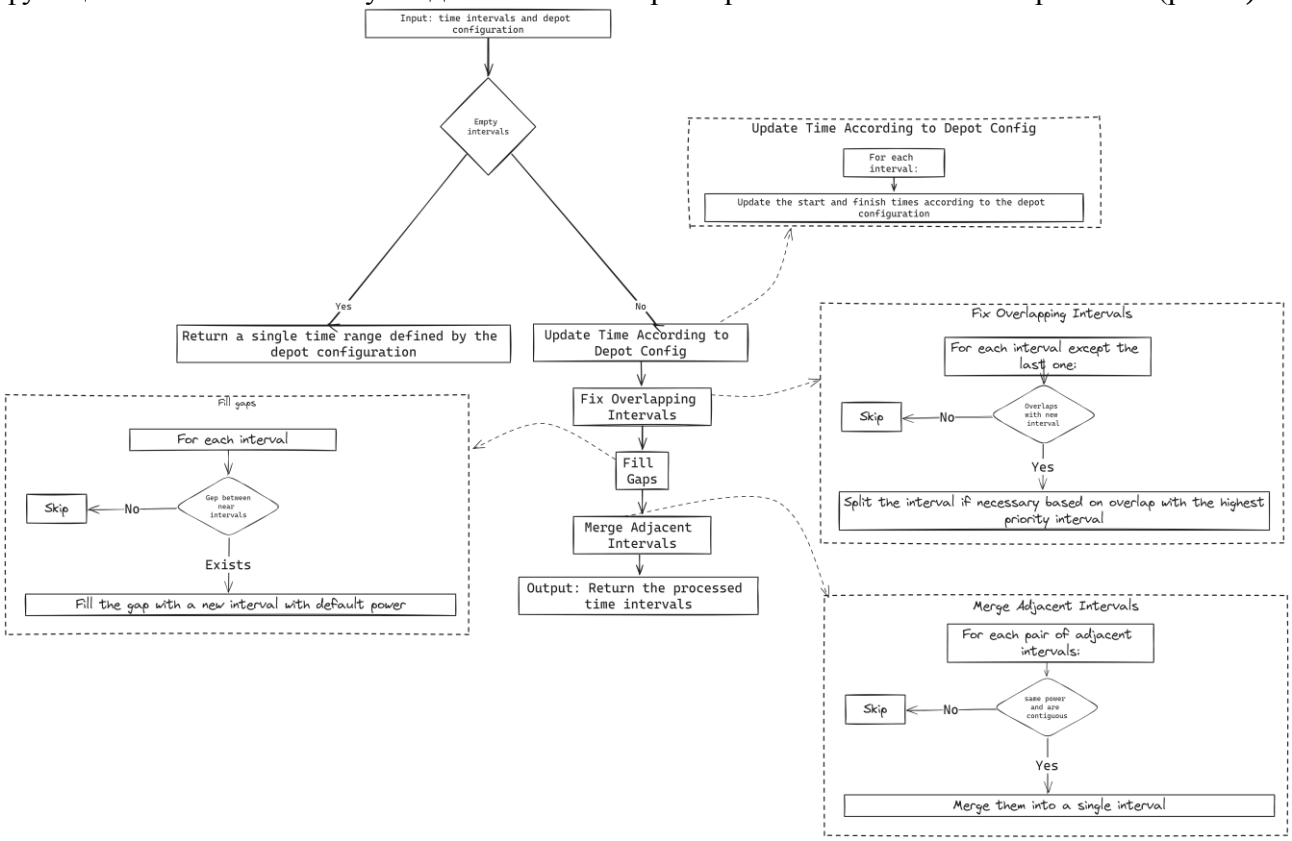

Рис. 7. Алгоритм формування часових інтервалів для процесу налаштування споживання електроенергії

Опис прийнятих програмних рішень

Реєстрація в системі доступна для всіх ролей, окрім супер адміністратора, адже цей користувач, вноситься в систему при купівлі продукту. Адміністратори мають можливість керувати користувачами системи використовуючи інструменти редагування, видалення, фільтрації. Реалізовано можливість запрошувати нових співробітників, інших адміністраторів та водіїв приєднатися до депо. Після запрошення користувач отримує електронний лист із посиланням для завершення реєстрації та зміни пароля.

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, з токена шляхом декодування отримується поточна роль користувача. Після декодування токена дані користувача зберігаються та використовуються додатком для контролю доступу до компонентів системи використовуючи механізм RBAC. Цей механізм використовує композицію з вже наявною директивою та взаємодії з фасадом аутентифікації, що забезпечує миттєву реакцію системи на будь-які зміни доступів.

Валідація полів. Слід зазначити, що перевірка введених значень у системі здійснюється у міру того, як їх вводять. 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.

– 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».

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, що дозволяє ефективно поширювати події та оновлювати стани компонентів.

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.

Для забезпечення зручного користування мобільною версією системи, особливо для водіїв, використовується Ionic 8. Це дозволяє створити адаптивний і інтуїтивний інтерфейс для управління резерваціями на мобільних пристроях.

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.

Для аналізу даних роботи депо було реалізовано три типи графіків, що дозволяють користувачам отримати візуальне уявлення про різні аспекти споживання енергії та роботи системи (рис.8).

Графік спожитої 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. Такий підхід дозволяє швидко оцінити, які станції споживають більше енергії і як розподіляється навантаження між різними станціями.

Другий тип графіка, відображає залежність між запланованим споживанням енергії в заданий період і фактичним споживанням. 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.

Третій тип графіка, представлений у вигляді лінійного графіку і демонструє загальну тенденцію потужності депо протягом певного періоду.

Рис. 8. Графічний аналіз даних роботи депо

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

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.

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

У процесі розробки програмного забезпечення було активно застосоване мануальне тестування та написані юніт-тести з використанням бібліотеки Jest для критичних частин системи.

Висновки

Згідно до мети дослідження розроблено веб-додаток із зручним інтерфейсом для ефективного управління зарядною інфраструктурою, реалізовано систему оптимізації витрат електроенергії, використано у клієнтській частині веб-додатку стандартну для серверної частини Onion архітектуру, що дає можливість краще структурувати код і забезпечити гнучкість та масштабованість, впроваджено гнучкі алгоритми для роботи з часовими інтервалами споживання енергії, створено інтеграцію з сучасними технологіями для обробки даних у реальному часі.

В основі серверної частини проекту знаходиться мікросервісна архітектура, де всі сервісі звертаються до однієї бази даних. Сервіси реалізовані на платформі ASP.NET, міжсервісна комунікація реалізована за допомогою брокера RabbitMQ та протоколу gRPC. Для комунікації зі станціями застосовані стандартні протоколи OCPP та WebSocket. Щодо CI/CD процесів, то вони налагоджені за допомогою Docker, Azure Kubernetes, Azure Container Registry та Azure DevOps Pipelines. Розроблено архітектуру, імплементовано бізнес-логіку та налаштовано CI/CD процеси відповідно до підходів, що було зазначено на етапі проєктування.

Здійснено комплексну реалізацію клієнтської частини системи управління зарядними станціями електромобілів. Після ретельного проєктування 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. В процесі реалізації були застосовані кращі практики оптимізації, проєктування архітектури системи та контролю якості коду. 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.

СПИСОК ЛІТЕРАТУРИ

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

6. What are microservices? *Microservice Architecture*. URL: https://microservices.io/ (дата звернення: 10.05.2024).

7. Motisi F. Database per service, shared instance or shared database? *Medium.com*. URL: https://mts88.medium.com/database-per-service-or-shared-database-e73cfb756aa1 (дата звернення: 10.05.2024).

8. RabbitMQ Documentation. RabbitMQ. URL: https://www.rabbitmq.com/docs (дата звернення: 10.05.2024).

9. Open charge point protocol – Open Charge Alliance. *Open Charge Alliance*. URL: https://openchargealliance.org/protocols/open-charge-pointprotocol/ (дата звернення: 10.05.2024).

Стаття надійшла до редакції 18.09.2024.
Стаття пройшла рецензування 28.09.2024.

Русакова Наталія Євгенівна – канд. техн. наук, доцент кафедри програмної інженерії, e-mail: nataliia.rusakova@nure.ua.

Горкун Дмитро Олександрович – студент.

Чубаров Євген Едуардович – студент.

Рубель Денис Андрійович – студент.

Налєскіна Тетяна Сергіївна – студентка.
Харківський національний університет радіоелектроніки.