МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

Вебплатформа організації перевезень вантажів

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Максим ВАЙСОВ**

«__» _____ 20__ р.

Керівник роботи
ст. викладачка

_____ **Світлана БОРОВЛЬОВА**

«__» _____ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Вайсов Максим

1. Тема кваліфікаційної роботи Вебплатформа організації перевезень вантажів затверджена наказом ректора ЧНУ ім. Петра Могили № 349 від «26» грудня 2026 р.

2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.

3. Очікуваний результат роботи: розроблена вебплатформа відповідно до теми роботи.

4. Перелік питань, що підлягають розробці: предметна область, існуючі рішення, проєктування архітектури та бази даних вебплатформи, реалізація модулів бізнес-логіки та інтерфейсу сутностей, розробка адаптації застосунку для його використання водіями під час виконання перевезень.

5. Перелік графічних матеріалів: презентація.

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «_24_» __жовтня_____ 2025 р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: **Вебплатформа організації перевезень вантажів**.

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	20.10.2025	24.10.2025	Виконано
2.	Огляд літератури за темою роботи	27.10.2025	31.10.2025	Виконано
3.	Складання календарного плану КБР	03.11.2025	06.11.2025	Виконано
4.	Аналіз предметної області	10.11.2025	21.11.2025	Виконано
5.	Розробка проєктних рішень	24.11.2025	12.12.2025	Виконано
6.	Моделювання та конструювання ПЗ	15.12.2025	02.01.2026	Виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	05.01.2026	24.04.2026	Виконано
8.	Оформлення КБР та презентації	27.04.2026	15.05.2026	Виконано
9.	Попередній захист	27.05.2026	27.05.2026	Виконано
10.	Відгук керівника КБР	01.06.2026	02.06.2026	Виконано
11.	Рецензування	03.06.2026	04.06.2026	Виконано
12.	Завершення оформлення КБР та презентації	05.06.2026	12.06.2026	
13.	Захист кваліфікаційної роботи			

Здобувач _____ **Максим ВАЙСОВ**

«6» __листопада__ 2025 р.

Керівник роботи _____ **Світлана БОРОВЛЬОВА**

ст. викладачка «6» __листопада__ 2025 р.

АНОТАЦІЯ

До кваліфікаційної бакалаврської роботи

Вебплатформа організації перевезень вантажів

Здобувач 409 гр.: Вайсов Максим

Керівник: ст. викладачка Боровльова Світлана

Актуальність. Організація вантажоперевезень є важливою частиною сфер людської діяльності, що пов’язані з роботою з фізичними товарами. Даний процес є багатостороннім, оскільки в ньому беруть участь як власники товару, так і перевізники із власним автопарком та персоналом. Існуючі застосунки не мають змоги надати обом сторонам можливість організувати власні ресурси для реалізації перевезень, а також їх подальшої структуризації та створення звітності.

Метою роботи є розробка вебзастосунку для автоматизації процесів керування перевезеннями вантажів.

Об’єктом роботи є процес автоматизації керування перевезеннями вантажів.

Предметом роботи є технології та архітектурні підходи розробки застосунку для автоматизації керування перевезеннями вантажів.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання. У вступі наведено актуальність, визначено мету, завдання, об’єкт, предмет теми роботи. У першому розділі проаналізовано предметну область та досліджено існуючі рішення. У другому розділі розглянуто використані технології, визначено функціональні та нефункціональні вимоги до програмного забезпечення. У третьому розділі змодельовано сценарії використання та архітектуру системи. У четвертому розділі описано результати кодування та тестування модулів. У висновках проаналізовано результати роботи у попередніх розділах.

КБР викладена на 76 сторінках, вона містить 4 розділи, 31 ілюстрацію, 4 таблиці, 27 джерел у переліку посилань.

Ключові слова: організація вантажоперевезень, React, Next.js, Nest.js, WebSockets.

ABSTRACT

to the qualifying bachelor's thesis

Freight transportation management web platform

Student of 409 group: Vaisov Maksym

Supervisor: senior lecturer Borovlova Svitlana

Relevance. The organization of freight transportation is a vital component of human activities involving the handling of physical goods. This process is multifaceted, as it involves both goods owners and carriers with their own fleets and staff. Existing applications are unable to provide both parties with the ability to organize their own resources for carrying out shipments, as well as for their subsequent structuring and reporting.

The goal of this work is to develop a web application for automating freight transportation management processes.

The object of this work is the process of automating freight transportation management.

The subject of this work is the technologies and architectural approaches to developing an application for automating freight transportation management.

This thesis consists of an introduction, four chapters, conclusions, and a list of references. The introduction outlines the relevance of the topic and defines the goal, objectives, subject, and scope of the research. The first chapter analyzes the subject area and examines existing solutions. The second chapter discusses the technologies used and defines the functional and non-functional requirements for the software. The third chapter models usage scenarios and the system architecture. The fourth chapter describes the results of coding and testing the modules. The conclusions analyze the results of the work in the previous chapters.

The thesis is 76 pages long and contains 4 chapters, 31 illustrations, 4 tables, and 27 sources in the reference list.

Keywords: freight transportation management, React, Next.js, Nest.js, WebSockets.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

СКБД	–	система керування базами даних
ШІ	–	штучний інтелект
ADR	–	Accord relatif au transport international des marchandises Dangereuses par Route
API	–	application programming interface
DOM	–	document object model
GDPR	–	general data protection regulation
HTTP	–	hypertext transfer protocol
HTTPS	–	hypertext transfer protocol secure
IP	–	internet protocol
LLM	–	large language model
OAuth	–	open authorization
RBAC	–	role-based access control
SMTP	–	simple mail transfer protocol
SQL	–	structured query language

ВСТУП

Актуальність. Організація вантажоперевезень є важливою частиною сфер людської діяльності, що пов’язані з роботою з фізичними товарами. Даний процес є багатостороннім, оскільки в ньому беруть участь як власники товару, так і перевізники, які можуть бути як окремими водіями, так і логістичними фірмами із власним автопарком та персоналом. Існуючі застосунки та інструменти не мають змоги надати обом сторонам можливість організувати власні ресурси для реалізації перевезень, а також їх подальшої структуризації та створення звітності. Через це виникає потреба в створенні інформативних засобів, що дозволять зробити процес вигідним та ефективним в роботі обох сторін.

Метою роботи є розробка вебзастосунку для автоматизації процесів керування перевезеннями вантажів.

Відповідно до мети необхідно виконати такі завдання:

– проаналізувати предметну область, зокрема існуючі рішення;
– спроєктувати архітектуру застосунку;
– розробити структуру бази даних;
– реалізувати модулі бізнес-логіки та інтерфейсу для сутностей;
– розробити адаптацію застосунку для його використання водіями під час виконання перевезень;
– провести тестування реалізованих модулів клієнтської та серверної частин системи.

Об’єктом роботи є процес автоматизації керування перевезеннями вантажів.

Предметом роботи є технології та архітектурні підходи розробки застосунку для автоматизації керування перевезеннями вантажів.

Обґрунтування необхідності нової розробки. Війна в Україні створила багато факторів, що впливають на логістичні процеси. Через пошкодження інфраструктури та блокування транспортних шляхів внаслідок бойових дій, необхідно мати змогу перепланувати маршрути, строки виконання, а також

повідомляти про надзвичайні ситуації в контексті замовлення в режимі реального часу. Також війна вплинула на наявність кадрів у компаніях, зокрема актуалізовано питання пошуку водіїв на замовлення, а також пошуку замовлень водіями в різних частинах країни. Враховуючи дані проблеми, постає необхідність створення застосунку, який зможе надавати актуальну інформацію всім залученим до перевезення сторонам та можливість швидкого пошуку як водіїв, так і вантажів. Для реалізації поставлених цілей можливо використати сучасні технології, такі як WebSocket для обміну повідомленнями між сторонами до та під час процесу перевезення в реальному часі, а також React (Next.js) для оптимізованого завантаження вебсторінок та Nest.js для модульної побудови серверної частини застосунку.

Основними напрямками роботи є:

– дослідження предметної області організації перевезень вантажів,

– пошук існуючих рішень та виявлення проблем, що потребують усунення в них;

– дослідження, порівняння технологій, які використовуються в системах даної предметної області, обґрунтування вибору технологій для майбутньої системи та основі цих даних;

– моделювання функціональних та нефункціональних вимог, сценаріїв використання, архітектури та інтерфейсу системи;

– кодування та тестування системи на основі специфікацій.

Апробація результатів. Використання фреймворків LangChain та LangGraph для створення багатоагентного робочого процесу обробки файлів представлено на XXVIII Всеукраїнській науково-практичній конференції «МОГИЛЯНСЬКІ ЧИТАННЯ – 2025: досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти» (10–14 листопада 2025 року).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Перед проєктуванням майбутньої вебплатформи необхідно дослідити предметну область, її особливості, наявні рішення, а також їх актуальні проблеми. Після цього буде можливо описати базовий функціонал системи, що усуватиме ці недоліки та сприятиме вирішенню завдань існуючих сторін даної предметної області.

1.1 Опис предметної області

Вантажоперевезення є основою економічної діяльності країни. Вони дозволяють поєднати сировину із устаткуванням для її переробки, а дистриб’ютора із кінцевим споживачем.

За даними Державної служби статистики, частка перевезень автомобільним транспортом після початку війни збільшилася до 77% і склала 677 млн. т., що є більшим за показники до 2022 р [1]. Також варто відмітити, що частка перевезень залізничним транспортом зменшилася до 17%, що є меншим за показники до 2022 р. Найбільшу частку вантажів, що перевозяться автотранспортними підприємствами, складає продукція добувної промисловості та розробки кар’єрів (22.2%), продукція сільського господарства (19.8%), зокрема зернові культури (14.8%), а також харчові продукти (15.6%) [1].

Згідно з індексом ефективної логістики Світового банку, Україна має значення 2.7, у той час як найближча країна ЄС Румунія має індекс 3.2 [2]. Найбільша різниця, а отже кількість проблем, спостерігається в оцінці роботи митних процесів незважаючи на електронне декларування в експортних процесах, рівня інфраструктури, логістичної компетентності та якості роботи логістичних компаній, а також ступеня надійності інформаційних систем для відстеження вантажів. Щодо середньої кількості днів доставки поштових замовлень, Україна має значення 3.9, у той час як більшість країн ЄС мають значення приблизно в два рази менше [2].

Щодо процесу організації вантажоперевезень, наразі більше 80% замовлень на перевезення оформлюються по телефону і логістичні оператори

витрачають більшість свого робочого часу на дзвінки для узгодження деталей відправлення, доставки, тарифів та обробку непередбачуваних подій [3]. Також, серед операторів поширене використання email пошти, цінна інформація в якій знаходиться на багатьох листах, документах та Excel таблицях для організації процесу перевезень, що призводить до погіршення ефективності, збільшенню часових та грошових витрат на організацію перевезень.

У країнах ЄС, незважаючи на більш розвинену інфраструктуру та більшу ступінь діджиталізації процесу організації перевезень, приблизно 20% вантажівок переміщується без вантажу щодня. Причиною цього є обмежений доступ до наявних вантажів для перевізників, а власники вантажів мають обмежений доступ до наявних перевізників поблизу, що зменшує прибуток перевізників та створює необґрунтоване забруднення навколишнього середовища.

Існуючі програмні рішення в ЄС, як-от «TIMOCOM», «Trans.eu», «Teleroute» та «Wtransnet» хоча й накопичили велику базу компаній, що представляють різні сторони процесу перевезень, але потребують плату виключно за доступ до цих даних без гарантії покращення бізнес-процесів. Також вони не надають готові рішення для інтеграції процесів, що пов’язані з перевезеннями, зокрема створення замовлень, документообіг, комунікація між сторонами, та виставлення платежів.

Через цей стан ринку програмних рішень, великі компанії змушені створювати власні системи, що надають необхідний функціонал, проте в такому випадку процес обміну даними стає асинхронним та дублюється між внутрішніми системами різних компаній. Таким чином, виробники товару обмежуються вже існуючими перевізниками, не маючи можливостей на пошук та верифікацію інших, потенційно більш вигідних кандидатів, окрім пошуку в Google та холодних дзвінків. З іншого боку, перевізники мають обмежені можливості пошуку замовлень через велику вартість підписок на існуючі сервіси, повторні надсилання тих же документів новим власникам товару та невизначеність щодо оплати від нових клієнтів.

1.2 Аналіз системи що розробляється

Вебплатформа для організації перевезень вантажів «FreightConnect» дозволить збільшити степінь автоматизації та прозорості організації перевезень, надавши усім сторонам можливість ефективно реалізовувати власні бізнес-процеси, пов’язані з логістикою. Застосунок має забезпечувати такі основні можливості для організації перевезень: облік власних ресурсів підприємства, пошук та виконання замовлень, комунікація між сторонами та інтеграція зі сторонніми системами.

Для реалізації цих можливостей буде реалізовано наступні функції:

- реєстрація компанії із вказанням публічно-доступних даних для її видимості для інших учасників платформи;
- запрошення користувачів до компанії за електронною адресою;
- керування ролями користувачів та призначення ролей користувачам;
- облік і керування вантажами та складами;
- створення та пошук замовлень на перевезення;
- створення пропозицій на виконання замовлення та можливість надати відповідь на них;
- облік та керування автопарком із можливістю призначення користувачів в якості водіїв для виконання замовлень;
- відслідковування виконання замовлення в режимі реального часу всіма сторонами завдяки використанню технології WebSockets;
- автоматичне планування маршруту з урахуванням погодних умов, трафіку, вимог до транспорту та вантажу із можливістю планувати вручну;
- комунікація між користувачами всередині компанії та між компаніями завдяки Talk.Js;
- надсилання повідомлень про події в системі та їх групування на окремій сторінці для зручності відслідковування процесів при інтенсивному використанні;

- можливість шукати компанії як власникам вантажів, так і перевізникам, залишати відгуки по виконанні замовлень та пропонувати замовлення перевізникам напряму;
- керування API ключами та їх рівнями доступу для інтеграції зі сторонніми системами;
- можливість експорту даних, що зберігаються на платформі;
- можливість імпорту даних різних форматів із використанням LLM для забезпечення відсутності прив’язанності до певної структури даних;
- можливість отримати аналітичні дані за використання платформи у різних форматах інфографіки;
- керування платежами за фактичне використання платформи завдяки фінансовому сервісу Paddle.

Реалізація RBAC системи надаватиме гнучку можливість створення ролей із довільним набором дозволів, що можна призначити довільній кількості користувачів. Також для кожної компанії створюватимуться стандартні ролі для зручного початку налаштувань та освоєння платформи:

- роль «адміністратор» має доступ на виконання операцій читання та запису для всіх сутностей;
- роль «оператор» може переглядати склади, переглядати та оновлювати вантажі, переглядати транспорт, керувати призначеннями та його доступністю, переглядати та оновлювати замовлення та пропозиції, може переглядати та створювати чати;
- роль «глядач» має право на перегляд складів, вантажів, автопарку, його доступності та призначень, замовлень та пропозицій;
- роль «водій» має право на перегляд призначеного йому транспорту, перегляд та оновлення статусу призначеного йому замовлення.

Також важливо забезпечити відповідність таким нормативно-правовим документам, вміст яких використовується при створенні замовлень та наданні пропозицій:

– Європейська Угода «Про міжнародне дорожнє перевезення небезпечних вантажів» [4];

– Закон України «Про єдині вимоги до конструкції та технічного стану колісних транспортних засобів, що експлуатуються» [5];

– Загальний регламент про захист даних [6].

1.3 Огляд існуючих аналогів

Проведення огляду існуючих аналогів направлене на отримання інформації щодо існуючого функціоналу для кожної сторони, що бере участь у перевезенні, даних щодо недоліків існуючих рішень на ринку для їх виправлення у власній платформі, а також перенесення вже перевірених архітектурних підходів та рішень, що позитивно сприймаються цільовою аудиторією даних застосунків. На основі отриманих даних можливо створити специфікацію щодо функціональних та нефункціональних вимог майбутньої системи, що дозволить зробити її конкурентоспроможною на ринку. Для огляду обрано такі існуючі платформи для організації вантажоперевзень: Della, Ant Logistics, Roolz.

Della

Della – це вебзастосунок, розроблений однойменною компанією і який надає послуги пошуку та публікації вантажів та транспорту [7]. Він побудований за трьохланковою архітектурою (база даних, сервер, браузерна частина). Серверна частина реалізована на мові програмування PHP, клієнтська частина використовує технології JavaScript, зокрема Ajax.

Серед основного функціоналу застосунку є можливість роботи з вантажами, зокрема пошук існуючих запитів на перевезення та створення власних вантажів, які можуть бути доступні для перегляду іншим користувачам. При створенні вантажу можливо вказати: дати та місця завантаження та розвантаження, тип вантажу та машин, які необхідні для його перевезення, вартість та деталі оплати, способи зв’язку та інші подробиці, зокрема документи, бажані способи завантаження та умови перевезення. При пошуку вантажу можливо вказати межі дат, маси, об’єму, локації завантаження та вигруження.

Також можлива фільтрація за вартістю, типом оплати, типом завантаження, ADR та гуманітарні вантажі.

Для кожного замовлення можливо відкрити приблизний розрахунок маршруту із вказанням проміжних населених пунктів. Результуючий розрахунок представлений у вигляді посилання на Google Maps.

При створенні оголошення про вільний транспорт вказуються такі дані: бажане місце завантаження та розвантаження, тип транспорту, вантажопідйомність та вільний об’єм, наявність GPS, а також подробиці, що дублюються з вантажами. Завдяки веденню історії записів про перевезення, застосунок надає можливість переглянути зведення за популярними маршрути перевезень по Україні, для яких вказується відстань, остання вага, ставка та вартість за кілометр перевезеного вантажу за цим маршрутом. Вигляд інтерфейсу пошуку застосунку зображено на рисунку 1.1.

Рисунок 1.1 – Інтерфейс пошуку транспорту Della

Della має також недоліки у реалізацію свого функціоналу, зокрема:

1) відсутність API інтеграцій, експорту чи імпорту даних для завантаження даних про транспорт та вантажі, або для використання платформи в інших застосунках;

2) відсутність інструкцій для початку використання, оскільки для отримання розуміння, як користуватися застосунком в якості

вантажовідправника чи вантажоперевізника, необхідно вводити дані, які одразу будуть доступні іншим користувачам;

3) відсутність детального опису можливостей, які надає преміум-пакет, окрім доступу до контактних даних оголошень;

4) розрахунок довжини маршруту здійснюється через сервіс Google Maps у режимі легкового автомобілю, що часто не відповідає дійсності у випадку з вантажним транспортом.

Ant Logistics

Ant Logistics – рішення для середнього та великого бізнесу, що надає можливості планування керування транспортом [8]. Застосунок побудовано за мікросервісною архітектурою, запущеною в хмарі Google Cloud Platform, де кожна підсистема має власний сервер. Мова реалізації серверної частини невідома, мова реалізації клієнтської частини – JavaScript. Серед основних підсистем платформи – планування маршрутів, контроль за дотримання маршрутів, товарообіг, аналітика щодо результатів роботи компанії, інтегрована з логістичними даними.

Підсистему планування маршрутів реалізовано з урахуванням дорожньої обстановки залежно від дня тижня та доби за поточними та історичними даними. Також надається можливість ручного налаштування ділянок маршруту, зокрема вказання типу дороги, середньої швидкості, коефіцієнту швидкості. Існує підтримка геозон для одночасного налаштування всіх ділянок доріг у певній географічній області. Підтримуються такі види маршрутизації:

1) гео-зональна – створення області на карті, за якою можна закріплювати водіїв чи транспорт для планування більш ефективних маршрутів в цій зоні;

2) групи адрес – кожна група може характеризуватися певними вимогами до габаритно-вагових характеристик транспорту, наприклад вантажівки для супермаркетів та легкові фургони для невеликих магазинів;

3) багаторейсова – організація перевезень за кілька завантажень та розвантажень у випадках, коли місткість транспорту не дозволяє зробити перевезти необхідний вантаж за одну ітерацію;

4) крос-докінг – процес планування міжміських чи регіональних перевезень з урахуванням перевантаження в проміжних населених пунктах;

5) за групами товарів, які є різними типами і можуть не бути сумісними між собою для довантаження або мають певні спільні вимоги щодо транспортних засобів, зокрема продукти харчування, продукти переробки нафти.

Також існують глобальні налаштування маршрутизації, зокрема години роботи, розрахунок з пріоритетом на кілометри чи гроші, ступінь важливості часових меж, дотримання норм навантаження, за коефіцієнтом часу виконання рейсу. Існує мобільний застосунок для водіїв, в якому вони можуть редагувати маршрути та повідомляти про оновлення статусу виконання перевезення. Загальний інтерфейс застосунку зображено на рисунку 1.2.

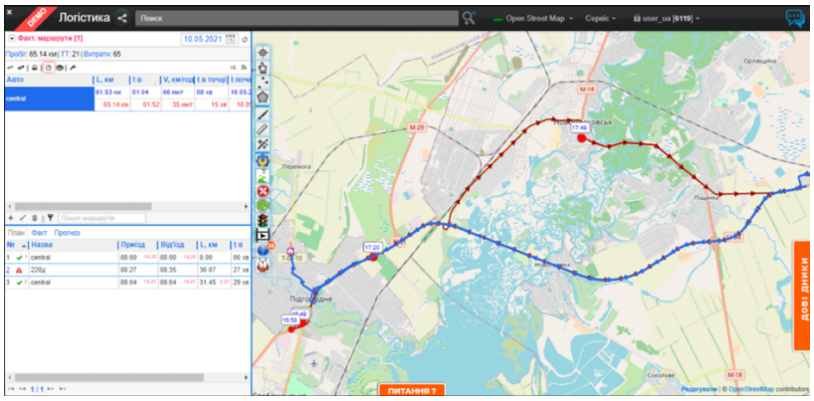

Рисунок 1.2 – Інтерфейс відстеження запланованого та фактичного маршруту на платформі Ant Logistics

Ant Logistics підтримує інтеграції з різними ERP та CRM системами, загальний інтерфейс імпорту даних, SMS-повідомлення клієнтів про статус виконання замовлень, та інтеграція GPS-трекерів. Підсистема аналітики надає можливості зображення дотримання маршрутів, зупинок та відхилень, показ маршрутних листів із загальними відомостями щодо продукції, пробігу та часових проміжків виконання кожного етапу виконання перевезення. Також

надається вибір із видів графіків для унаочнення зведення по маршрутах, грошових і часових витратах, пробігу транспорту. Недоліками Ant Logistics є:

1) обов’язкова презентація плану використання платформи та введення початкових даних від інженера, що надається компанією Ant Logistics;

2) формалізація процесу інтеграції у вигляді необхідності надання звіту про впровадження системи;

3) відсутня реалізація частини системи, яка б дозволяла кінцевим отримувачам вантажу переглядати поточний стан перевезення та брати участь у плануванні перевезень, через що виникає необхідність використання інших каналів зв’язку та інструментів для сумісного планування логістики.

Roolz

Roolz – вебплатформа пошуку вантажів для перевезень та доступного транспорту, що реалізована за принципом біржі [9]. Застосунок побудовано за трьохланковою архітектурою, де клієнтська та серверна частини реалізовані мовою програмування JavaScript, зокрема сервер використовує середовище виконання Node.js та вебфреймворк Express.js. Основним функціоналом платформи є створення та пошук замовлень на перевезення вантажів, створення та пошук оголошень про наявний транспорт для виконання перевезень, система ставок, пошук та комунікація між компаніями. При створенні замовлення на перевезення вантажу можливо вказати такі відомості: назва, тип, вага, об’єм, габарити, ADR, вартість, термоконтроль, вимоги до транспортування, прикріплені файли, умови оплати та доступність оголошення до перегляду. Система пропонує такі види торгів:

1) торги – перевізники розміщують пропозиції, а власник товару обирає за ціною та умовами транспортування;

2) аукціон – автоматичне обрання пропозиції з найнижчою ціною;

3) торги з фіксованою ціною – пропозиції з можливістю конкурувати тільки умовами;

4) оголошення – пропонування, обговорення ціни та умов відбувається шляхом прямого спілкування електронною поштою та дзвінками.

Під час створення оголошення доступна функція планування маршруту із вказанням проміжних точок, які можуть бути, крім пунктами завантаження та розвантаження, також пункти перетину кордону та довантаження. Кожне замовлення можливо створити як чернетку з метою подальшого редагування, а також зберегти як шаблон для часто створюваних замовлень.

Вказуються такі дані про наявний транспортний засіб для використання в замовленнях: категорія, тип кузова, доступний тоннаж та об’єм, розміри, особливості перевезення, вартість за кілометр. Замість точного маршруту для транспорту вказується бажана дата і локація завантаження та доставки. Під час пошуку вантажів чи транспорту можливо вказати фільтрацію за місцем і дати початку та закінчення перевезення, доступний чи необхідний тоннаж чи об’єм.

Переглядаючи кожне замовлення, можливо побачити деталі без необхідності попередньої оплати на платформі включно з маршрутом, а також можливо почати комунікацію з клієнтом в межах платформи. Для отримання контактних даних із метою комунікації поза межами системи необхідний базовий рівень підписки.

Також система надає відкритий реєстр зареєстрованих компаній із фільтрацією за видами перевезень (авіаперевезення, судноплавство, залізничні перевезення) та за послугами (перевізники, вантажовласники, експедитори, зберігання, оренда складських приміщень, крос-докінг). Для кожної компанії можна переглянути такі дані: назва, категорія, послуг та видів перевезень, розмір автопарку, кількість працівників, поточні активні оголошення про перевезення та доступні вантажі. Основний інтерфейс застосунку наведено на рисунку 1.4.

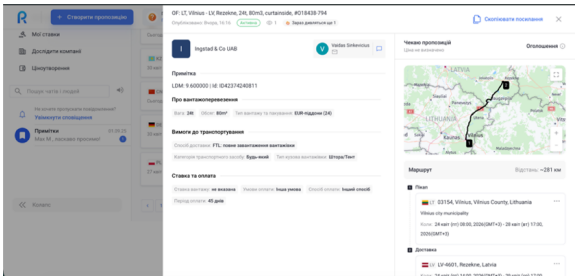

Рисунок 1.4 – Інтерфейс оголошення про вантаж на платформі Roolz

Недоліками Roolz є:

1) відсутність аналітики та глобальних показників, таких як середня вартість та історія цін перевезень за маршрутами, популярні маршрути;

2) неповний список фільтрів при пошуку вантажних перевезень (зокрема тип вантажу, спосіб доставки, умови оплати);

3) недостатня деталізація опублікованого транспорту та кількості умов перевезення;

4) під час торгів видно усіх учасників та їх пропозиції, що може порушувати комерційну таємницю.

Після проведеного аналізу існуючих рішень у області організації вантажоперевезень, можливо підбити підсумки у вигляді таблиці переваг та недоліків кожного розглянутого застосунку.

Таблиця 1.1 – Підсумки щодо функціоналу існуючих рішень

Назва	Розробник	Архітектура	Недоліки
Della	Della	Трьохланкова (PHP, JavaScript)	– відсутність API інтеграцій, експорту чи імпорту даних; – відсутність інструкцій для початку використання; – відсутність детального опису можливостей, які надає преміум-пакет; – розрахунок довжини маршруту здійснюється через сервіс Google Maps у режимі легкового автомобілю.
Ant Logistics	ANT-Logistics	Мікросервісна	– обов’язкова презентація плану використання та введення початкових даних від інженера Ant Logistics; – необхідність надання звіту про впровадження системи; – відсутня реалізація комунікації та сумісного планування з кінцевим одержувачем вантажу.
Roolz	Roolz	Трьохланкова (Node.js, JavaScript)	– відсутність аналітики та глобальних показників; – неповний список фільтрів при пошуку вантажних перевезень; – недостатня деталізація опублікованого транспорту; – під час торгів видно усіх учасників та їх пропозиції, що може порушувати комерційну таємницю.

Проаналізовані існуючі рішення, окрім специфічних для кожної платформи недоліків, мають спільну особливість недостатнього інформаційного забезпечення користувачів, або для роботи із системою, або для інтеграції з

іншими рішеннями, що вже інтегровані в бізнес-процеси. Це ускладнює як процес інтеграції, так і впливає на ефективність їх подальшого використання.

Висновки до розділу 1

У даному розділі проаналізовано предметну область організації перевезень вантажів. Визначено автотранспорт як основний вид перевезень вантажів в Україні, встановлено проблеми організації перевезень, що існують як в межах України, так і країнах ЄС, зокрема порожні перевезення, значні витрати часу на організацію даних серед різних джерел даних, узгодження та пошук перевізників серед дорогих рішень, які не надають гарантій знаходження вантажу чи транспорту, і обмежені чи ресурсозатратні можливості пошуку поза ними.

Розглянуто існуючі рішення Della, Ant Logistics та Roolz, їх можливості та недоліки, що призводять до погіршення ефективності організації вантажоперевезень. На основні зібраних даних визначено основні можливості та їх функціонал майбутньої вебплатформи, обґрунтовано використання гнучкої системи ролей RBAC замість ригідного набору наперед визначеної іменованої сукупності прав доступу.

Кожна платформа надає або рішення в межах окремої компанії, ізолюючи її від комунікації з іншими сторонами, або обмежений доступ до них, або такий публічний доступ, що може зашкодити її комерційній таємниці. Також поширені недоліки у сфері надання даних, зокрема для аналітики, з можливістю їх імпорту або експорту.

Не менш важливою є зручність використання платформи, зокрема наявність докладних фільтрів за різними деталями бажаного вантажу чи транспорту, а також точність розрахунку маршруту, що буде використаний для оцінки вартості перевезення та безпосередньо водіями. Наявні платформи, що спеціалізуються на наданні бази оголошень про доступний транспорт чи вантажі, мають обмежуючі умови оплати у вигляді лише доступу до даних, не спонукаючи користувачів до безпосередньої інтеграції та оптимізації цілих бізнес-процесів за рахунок всебічного використання платформи замість кількох розрізнених рішень.

2 ОГЛЯД ТЕХНОЛОГІЙ ТА СПЕЦИФІКАЦІЯ ВИМОГ

Після аналізу предметної області можливо перейти до визначення стеку технологій, функціональних та нефункціональних вимог. Це уможливить більш детальне проєктування функціоналу та реалізацію системи згідно даних вимог.

2.1 Огляд стеку технологій

Для початку проєктування застосунку необхідно визначити набір технологій, які дозволять реалізувати ту чи іншу архітектуру. Дані технології мають надавати універсальність, не накладаючи обмежень щодо способу побудови коду або роблячи це згідно принципів, що роблять код масштабованим, легким у розробці, підтримці та подальшим оновленням. Не менш важливою складовою вибору є ступінь активності спільноти розробників у підтримці технології, частота її оновлення, оскільки це впливає як на безпеку застосунку й даних, так і на зручність при реалізації та наявність помилок, які не були вирішені та унеможливлюють її подальше використання [10].

Подальший вибір технологій залежить від вибору мов програмування, на яких будуть побудовані складові вебплатформи. На даний час універсальність, покращену якість коду та меншу кількість багів під час розробки надає мова програмування TypeScript [11].

TypeScript – це сильно типізована, об'єктно-орієнтована, компільована мова програмування, яка розроблена Microsoft та базується на JavaScript. Таким чином, код JavaScript можна одразу перенести у формат TypeScript, який автоматично побудує систему типів без додаткових анотацій. Головним принципом даної мови програмування є «качина типізація», тобто значення порівнюються на відповідність типів за їх формою.

Для підтримки випадків, коли автоматичної типізації недостатньо, TypeScript надає можливості для створення та позначення типів даних. Зокрема підтримуються оголошення «interface», що вказують, які поля та методи має містити об’єкт [12].

Для поєднання типів існує окремий вид позначень «type», що дозволяє як створювати вбудовані інтерфейси, так і поєднувати різні інтерфейси та літеральні типи («number», «string», і т.д.) між собою для позначення, що значення може бути різних типів, що особливо корисно під час типізації функцій чи їх параметрів. TypeScript також надає команду «tsc» для компіляції .ts файлів у .js із перевіркою типів.

В якості бібліотеки для розробки клієнтської частини застосунку обрано React – це JavaScript бібліотека для побудови адаптивних вебінтерфейсів, що дозволяє розділити складові сторінки на компоненти, які можна налаштовувати та повторно використовувати у різних частинах застосунку. Також існує можливість перенесення застосунку на мобільні пристрої завдяки загортанню нативного коду (Swift / Kotlin) в компоненти React Native [13].

Компонент у React – це JavaScript функція, що повертає розмітку, яка є розширенням синтаксису JavaScript та має назву JSX [14]. Вона є обгорткою для створення JavaScript об’єктів, що представляють віртуальне DOM дерево, завдяки якому React може визначати відмінності між реальним DOM деревом, яке оновлюється браузером, після оновлення стану, та оновлювати лише ті частини дерева, що потребують зміни згідно нового стану. Алгоритм порівняння справжнього та віртуального дерев називається реконсиліацією та заснований на двох евристичних принципах, що дозволяють оптимізувати алгоритмічну проблему оновлення дерева за найменшу кількість операцій:

- два елементи різних типів означають різні піддерева
- розробник може «підказати», як дочірні елементи вершини однакових типів мають різний вміст та як саме вони співвідносяться між рендерами завдяки аргументу «key» [15]. Даний аргумент найбільш часто використовується при рендеру списків, особливо тих, що мають підтримку перестановки та оновлення вмісту.

Принцип дії алгоритму можливо побачити на рисунку 2.1. На ньому зображено віртуальне та фактичне дерева компонентів, виявлення компонентів, яких фактично змінилися, та найбільш оптимальне оновлення фізичного дерева.

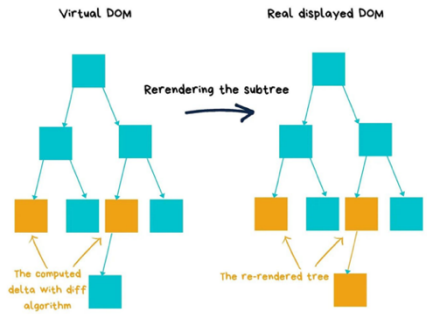

Рисунок 2.1 – Візуалізація реконсиліації

Окрім порівняно ефективного принципу роботи [16], іншими перевагами React поміж таких бібліотек, як Angular, Vue та Svelte є активне використання у великій кількості проєктів серед розробників [17], а також наявність метафреймворку Next.js. Next.js – це метафреймворк, заснований на бібліотеці React, що дозволяє покращити оптимізацію React застосунків, а також надає багато важливих функцій для спрощення побудови зокрема серверної частини цих застосунків.

Першим механізмом, з використання якого починається розробка Next.js застосунку – механізм маршрутизації. У версії Next.js 13 опубліковано новий алгоритм «App Router», що має такі властивості [18]:

1) шляхи маршрутизації визначаються за файловою структурою (починаючи з «/src/app»);

2) шлях для відкриття браузером визначається файлом із назвою «page», шлях для API ендпойнтів визначається файлом із назвою «route»;

3) шлях, який позначає обгортку інтерфейсу для піддерева шляху визначається файлом «layout» (зберігає стан між рендерами) чи «template» (не зберігає стан між рендерами).

Рендер сторінок в Next.js може відбуватися як на стороні клієнта, так і сервера. Самі сторінки можуть складатися з двох видів компонентів.

Серверні компоненти – вид компонентів за замовчуванням, який передбачає виконання всього JavaScript коду, пов’язаного з компонентом, включно із запитами до сторонніх сервісів, на стороні серверу, а браузер отримує готову HTML-розмітку. Це дозволяє розвантажити клієнта від зайвих запитів до

сервісів та здійснює рендер на більш потужному апаратному забезпеченні серверів. Проте дані компоненти позбавлені інтерактивності та не можуть мати стану. Увімкнути даний режим можливо директивою «use server» на початку файлу оголошення компоненту.

Клієнтські компоненти – компоненти, рендер яких відбувається повністю на стороні клієнта зі збереженням стану, адже сервер надсилає React для виконання. Увімкнути даний режим можливо директивою «use client».

Завдяки механізму кешування Next.js дозволяє зберігати результати запитів до сторонніх сервісів та баз даних під час рендеру, а також кешувати результати рендерів окремих компонентів та сторінок із можливістю інвалідації кешу за потреби. Це покращує продуктивність роботи клієнтської частини, а також зменшує навантаження на сторонні джерела. Окрім цього, Next.js надає механізми оптимізації зображень, надаючи версії, що необхідні для конкретного розміру екрану завдяки компоненту «Image», а також оптимізації клієнтської навігації завдяки фоновому завантаженню сторінок із автоматичним скасуванням завдяки компоненту посилання «Link».

Реалізація запитів на клієнтській частині використовуватиме бібліотеку React Query, що дозволяє задіяти механізм кешування відповідей, а також та зменшити кількість коду завдяки використанню React хуків [19]. Дана технологія має підтримку із технологією серверних та клієнтських компонентів Next.js, інтегруючись у механізм гідрації на клієнті для завантаження кешу запитів, попередньо зроблених із серверних компонентів.

Дизайн інтерфейсу буде реалізовано завдяки UI-бібліотеці Shadcn, що заснована на Radix та має готові компоненти, адаптовані під людей з обмеженими можливостями та має базову дизайн-систему із гнучким налаштуванням [20]. Безпосереднє налаштування даних компонентів здійснюватиметься CSS фреймворком Tailwind, що надає готові класи та систему глобальної конфігурації кольорів, розмірів, шрифтів, CSS змінних, що спрощує розробку інтерфейсу із єдиним джерелом налаштувань.

Технологією серверної розробки, що використовує можливості мови програмування TypeScript, а також середовища виконання Node.js, обрано фреймворк NestJS. NestJS вирішує такі проблеми, що існують у інших фреймворків або реалізації серверу виключно вбудованими засобами Node.js [21].

Відсутність чіткого архітектурного підходу: NestJS структурує частини серверу у модулі, які мають сервіси та контролери та які підтримують контрольований експорт своєї логіки. Приклад модульної структури серверу NestJS зображено на рисунку 2.2.

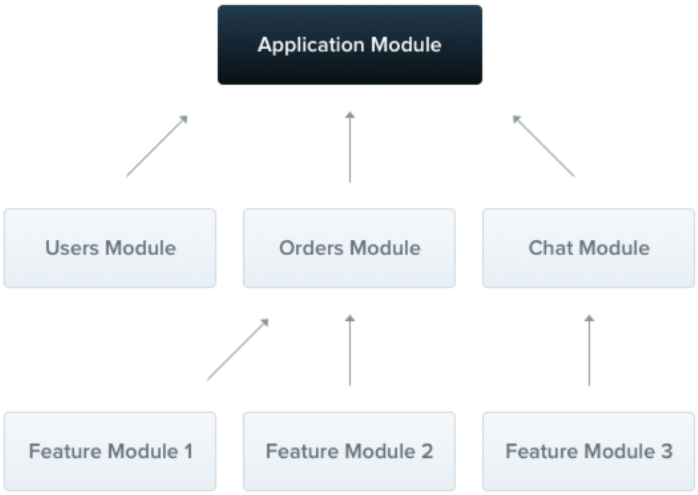

Рисунок 2.2 – Модульна структура NestJS серверу

Відсутність механізму ін’єкції залежностей: завдяки модульній архітектурі та можливості контрольованого експортування сервісів, Nest.js реалізує паттерн «декоратор» для реєстрації сервісів у DI-контейнері та для їх ін’єкції у різних частинах застосунку. Зокрема підтримується ін’єкція через параметри конструктору, а також через параметри методу. Самі сервіси можуть мати різний час життя: створення один раз під час запуску серверу, створення один раз на вхідний запит, створення в кожному місці ін’єкції окремо для кожного класу, що споживає сервіс.

Слабка підтримка TypeScript: поки Node.js має лише експериментальну підтримку TypeScript [22], а вбудовані модулі не містять оголошень типів та потребують окремого налаштування. NestJS має не тільки вбудовану підтримку

типів, але й використовує TypeScript декоратори для анотування класів, їх полів та методів [23]. Також NestJS надає вбудовані можливості для написання тестів до модулів, а CLI-інструменти дозволяють генерувати частини серверу автоматично.

У якості бази даних буде використано реляційну СКБД PostgreSQL. На даний вибір вплинуло кілька факторів: наявність чіткої структури даних та багатьох зв’язків між різними сутностями, використанню транзакцій для оновлення різних сутностей, необхідність у підтримці різних видів обмежень для атрибутів сутностей [24].

Для реалізації функціоналу, що потребує зв’язку в режимі реального часу, між частинами буде використано протокол WebSockets та бібліотеку Socket.io, що забезпечує гнучкий перехід комунікації до WebSockets та відкат до HTTPS «Long Polling» механізму за відсутності підтримки WebSockets з боку клієнта [25].

2.2 Специфікація вимог до програмного забезпечення

Після аналізу основних можливостей, що має забезпечувати вебплатформа для організації перевезень вантажів, а також надання початкового переліку пов’язаних функцій та системи ролей майбутніх користувачів, необхідно визначити специфікацію вимог до даного програмного забезпечення. Вона має описувати його призначення, межі, функціональні та нефункціональні вимоги.

1. Призначення та межі проєкту

Вебплатформа організації перевезень вантажів «FreightConnect» призначена для зменшення ресурсів, які необхідно витратити на пошук контрагентів, отримання деталей щодо існуючих замовлень і подання пропозицій. Також система має полегшувати комунікацію між сторонами до та під час здійснення перевезення, їх координацію для відслідковування та перепланування перевезень у режимі реального часу.

1.2 Погодження, ухвалені в програмній документації

– можливості та функціональні особливості визначено шляхом аналізу існуючих рішень, що наразі користуються попитом серед перевізників, логістів та власників вантажу, а також шляхом безпосереднього неформального опитування;

– визначено набір основних технологій, на основі яких буде спроєктовано та побудовано архітектуру вебплатформи: TypeScript, Next.js, NestJS;

– для забезпечення доступу до застосунку було обрано формат вебзастосунку із початковою адаптацією під мобільні пристрої, оскільки застосунок планується використовувати водіями для відстеження та оновлення статусу перевезення.

1.3 Межі проєкту

– вебплатформа буде реалізована за трьохланковою архітектурою із забезпечення можливості переходу на мікросервіси завдяки модульній архітектурі серверної частини та хмарних технологій масштабування, що надаються під час розміщення клієнтської частині на серверах Vercel;

– застосунок реалізовуватиме функціонал обліку користувачів, вантажів, складів, керування та супровід замовлень на перевезення вантажів, комунікацію між сторонами;

– система забезпечуватиме оновлення даних щодо доступних замовлень та статусу активних замовлень у режимі реального часу завдяки технології WebSockets (Socket.io);

– система матиме інтеграцію із сервісами збереження зображень, документів та чату;

– система підтримуватиме експортування та імпортування даних у загальних форматах, зокрема з підтримкою файлів довільних форматів через їх обробку штучним інтелектом;

– система не ставить за мету замінити існуючі CRM, ERP та TMS системи, що використовуються в бізнес-процесах.

– система має надати можливості інтеграції в існуючі рішення та процеси із метою забезпечення більш ефективного використання наявних ресурсів бізнесу під час реалізації логістичних потреб.

2. Загальний опис

2.1 Сфера застосування

Дана вебплатформа може використовуватися компаніями, що займаються організацією логістичних процесів, експедиторами, продуктовими компаніями, що мають організувати перевезення власного вантажу, а також транспортними компаніями, що шукають вантаж для перевезення.

2.2 Характеристики користувачів

Хоча система підтримує гнучку систему ролей, що дозволяє створювати власні ролі із різними рівнями доступу до кожної сутності, можливо виокремити такі категорії користувачів:

– власник компанії або її представник реєструє компанію, вказує її деталі, реєструє співробітників компанії на платформі, налаштовує ролі доступу, API ключі;

– менеджер (також може бути власником) виконує імпорт або ручне введення даних про вантажі, склади, транспорт, призначає водіїв, створює замовлення на перевезення або шукає існуючі та подає пропозиції, проводить комунікацію з контрагентами, відслідковує процес виконання замовлення;

– водій оновлює статус замовлення на перевезення згідно транспорту, який вказано в пропозиції.

2.3 Загальна структура і склад системи

– клієнтська частина – Next.js (React), Shadcn (Radix), Tailwind CSS;

– серверна частина – сервер API NestJS, що використовує базу даних PostgreSQL;

– технологія комунікації між частинами в режимі реального часу – WebSockets.

2.4 Загальні обмеження

– WebSockets протокол потребує сучасного браузеру;

– реалізація мобільного додатку в якості клієнтської частини займає більше часу для адаптації під кожну мобільну операційну систему, тому замість цього буде реалізовано веб-версію із адаптацією інтерфейсу до мобільних пристроїв;

– імпортування файлів буде обмежено документами до 10 МБ та лише найбільш поширеними форматами: CSV, XLSX, DOCX, PDF, зображення;

– розрахунок маршрутів, а також розділ чату з підтримкою файлів, груп та системних сповіщень буде делеговано зовнішньому сервісу через складність фактичної реалізації.

3. Функції системи

3.1 Керування ідентифікацією та доступом (модуль «IAM»)

3.1.1 Опис функції

Даний модуль відповідає за реєстрацію компаній, створення та керування користувачами, ролями, генерування API ключів та надання їм дозволів. Також надаватиметься доступ до аудиту подій, пов’язаних із модулем.

3.1.2 Вхідна і вихідна інформація

– вхідна – інформація про компанію (назва, пошта, телефон, ЄДРПОУ, юридична адреса), інформація про користувача (ПІБ, телефон), інформація про ролі (назва, опис, перелік дозволів), інформація про API ключі (назва, обмеження частоти запитів, час експірації, дозволи);

– вихідна – створені сутності компанії, користувача, ролі, API ключа, кукі сесії з токеном для автентифікації.

3.1.3 Функціональні вимоги

– створення компанії за заповненими даними (назва, пошта, телефон, ЄДРПОУ, юридична адреса);

– автоматичне створення шаблонів ролей «Адміністратор», «Менеджер», «Оператор», «Водій» для компанії;

– створення облікового запису першого користувача за заповненими даними (ПІБ, телефон), призначення йому ролі «Адміністратор»;

– аутентифікація через механізм «magic-link» та інтеграцію з Google («Google Auth»);

– запрошення користувачів за електронною адресою та керування ними: зміна даних, ролей, деактивація та видалення;

– створення ролей із довільним доступом;

– створення API ключів із довільним доступом, обмеженням по частоті використання, із датою експірації та можливістю регенерувати чи деактивувати ключі;

– логування подій: вхід, оновлення ролей, використання API ключів (час, категорія, ID цільової сутності, IP-адреса;

– експортування даних про активність власного облікового запису.

3.2 Керування складами та вантажами (модуль «Warehouse»)

3.2.1 Опис функції

Даний модуль відповідає за створення складів та вантажів, керування ними, прикріплення складів до вантажів для зручності заповнення замовлення на перевезення, а також пошук по ним. Надається можливість аудиту подій, пов’язаних із керуванням складами та вантажами.

3.2.2 Вхідна і вихідна інформація

– вхідна – інформація про склад (назва, адреса, локація, телефон), інформація про вантаж (тип, кількість, опис, склад, вага, об’єм, ADR код, габарити, температурний режим, чи швидкопсувний);

– вихідна – сутності складу та вантажу.

3.2.3 Функціональні вимоги

– створення та керування складами за заповненими даними;

– створення та керування вантажами;

– пошук вантажів із використанням фільтрації за полями через рядок, а також за приналежністю до складу;

– можливість завантаження та керування документами, пов’язаних із вантажем;

– видалення вантажів відбуватиметься через встановлення мітки часу замість повного видалення для збереження цілісності історії перевезень.

3.3 Керування автопарком (модуль «Transport»)

3.3.1 Опис функції

Даний модуль відповідає за створення транспорту, оновлення його локації та графіку доступності для взяття участі в замовленнях. Також модуль надаватиме можливість призначення водіїв на транспорт для виконання замовлень.

3.3.2 Вхідна і вихідна інформація

– вхідна – інформація про транспорт (номерний знак, тип транспорту, тип кузову, поточна локація, можливості щодо перевезення вантажів: габарити, вага, об’єм, ADR, тип завантаження, підтримка температурного режиму та швидкопсувних товарів), інформація про призначення водія (сутність водія, опціональна кінцева дата призначення), інформація про доступність транспорту (початкова та кінцева дати, доступні локації, доступна вага та об’єм, тариф за кілометр і за тонну, можливість приймати часткові завантаження;

– вихідна – сутності транспорту, доступності, призначення водія, аудит подій транспорту.

3.3.3 Функціональні вимоги

– створення транспорту на основі введених даних;

– можливість перегляду деталей про кожен введений транспорт;

– можливість перегляду зведення по статусу задіяного транспорту, по можливостях автопарку (загальна вантажопідйомність, об’єм), по наявних проміжках конфігурації автопарку для його використання у замовленнях

– можливість перегляду статистики використання транспорту в замовленнях;

– оновлення відомостей щодо транспорту;

– можливість пошуку транспорту за полями, фільтрації за статусом, типом, наявністю водія;

– створення графіку доступності транспорту;

– створення призначення водія на транспорт;

– логування подій, пов’язаних із налаштуваннями сутностей в даному модулі;

– можливість прикріплення документів щодо транспорту;

– можливість оновлення поточної локації транспорту (поза межами замовлення).

3.4 Керування замовленнями (модуль «Deliveries»)

3.4.1 Опис функції

Даний модуль відповідає за створення та керування замовленнями на перевезення вантажів, надсилання пропозицій на виконання замовлень, пошук транспорту та пропонування замовлення перевізнику. Також цей модуль надає можливості відстеження процесу виконання замовлення та створення запитів на скасування активного замовлення.

3.4.2 Вхідна і вихідна інформація

– вхідна – інформація про замовлення: назва, опис, пов’язаний вантаж, локація і дата завантаження, локація і дата розвантаження, бюджет, вимоги до транспортування (тип пакування, кількість, оголошена вартість), ставка та оплата (спосіб та умови оплати, період оплати, ціна з ПДВ чи ні). Інформація про пропозицію від перевізника (ціна, обраний транспорт, приблизна дата завантаження та розвантаження, примітки), місцезнаходження водія;

– вихідна – сутності замовлень, пропозицій, розрахований маршрут для відстеження водієм із використанням зовнішнього сервісу, статус виконання активного замовлення.

3.4.3 Функціональні вимоги

– створення чернеток замовлень, їх публікація та повернення до статусу чернетки;

– перегляд власних замовлень та їх зведення по статусу, окремий перегляд активних замовлень;

– розрахунок маршруту замовлення за вимогою;

- автоматичний розрахунок маршруту замовлення після початку виконання замовлення;
- пошук замовлень за системою фільтрації: текстовий пошук, за місцем завантаження і розвантаження, дати, вага та об’єм, бюджет, вимоги до вантажу;
- пошук наявного транспорту за системою фільтрації: локація, тип транспорту та причепу, доступні дати, місткість, тариф, підтримка вимог до вантажу;
- перегляд деталей про доступний транспорт та надання прямої пропозиції про перевезення вантажу;
- перегляд зведення про активні вхідні та вихідні пропозиції;
- можливість створити запит на відміну виконання замовлення;
- перегляд активних та проведених запитів на відміну замовлення;
- перегляд деталей про активне замовлення: статус, історія змін, мапа з маршрутом та переміщенням водія;
- можливість перегляду замовлення з точки зору водія з можливістю оновити статус, перепланувати маршрут, отримати наступні вказівки для слідкування за маршрутом.

4. Вимоги до інформаційного забезпечення

4.1 Джерела і зміст вхідної інформації

- дані про компанію вносяться її представником під час реєстрації;
- початкові дані користувача вносяться адміністратором під час запрошення користувача;
- детальні дані користувача вносяться самим користувачем під час реєстрації;
- дані про склади та вантажі, замовлення на перевезення вносяться представником компанії, що володіє вантажем, із відповідною роллю;
- дані про пропозиції щодо замовлень на перевезення вносяться представником компанії-перевізника;

- оновлення статусу активного замовлення на перевезення та поточної локації вантажу робляться водієм або уповноваженою особою від компанії-перевізника;
- файли для імпорту даних у вебплатформу надходять від представників компанії, що мають право на операції імпортування;
- ціни щодо спожитих сервісів встановлюються адміністратором платформи та отримуються через платіжний сервіс Paddle;
- згода на використання кукі та AI для імпортування даних надається користувачем, що безпосередньо користується вебплатформою та функціями імпорту даних.

4.2 Вимоги до способів організації, збереження та ведення інформації

- інформація вводиться засобами Prisma ORM та зберігається в структурованому вигляді в реляційній базі даних PostgreSQL, що розміщена на серверах компанії Supabase;
- файлові додатки щодо вантажів, транспорту та аватарів користувачів зберігаються на серверах компанії Supabase (сервіс «Supabase Storage»);
- відбувається хешування паролів делегуванням операцій хешування та звірення хешів бібліотеці Better Auth, що використовує алгоритм scrypt;
- відбувається хешування API ключів за алгоритмом SHA-256;
- відбувається версіонування маршрутів доставки з метою підтримки перебудови маршруту та логування історії змін;
- відбувається видалення вантажів з метою збереження цілісності даних із пов’язаними замовленнями;
- відбувається анонімізація персональних даних та даних компанії при їх видаленні з подальшим повним видаленням через 30 днів від моменту отримання запиту на видалення;
- відбувається логування подій по кожному сервісу з метою надання можливості аудиту, логи зберігаються 7 років.

5. Вимоги до технічного забезпечення

– використання хмарної інфраструктури Vercel для розміщення як клієнтської, так і серверної частини застосунку. Серверна частина застосунку буде доступна у форматі одної хмарної функції, яка надаватиме доступ до всіх ендпойнтів, у той час як клієнтська частина використовуватиме хмарні функції для кожного ендпойнту та SSR, а також CDN для шрифтів та інших статичних файлів, що надається інфраструктурою Vercel. Хмарні функції можуть масштабуватися згідно поточного навантаження;

– використання БД PostgreSQL та сховища файлів на серверах Supabase;

– використання браузерів із підтримкою WebSockets.

6. Вимоги до програмного забезпечення

6.1 Архітектура програмної системи

Вебплатформа матиме трьохланкову архітектуру із клієнтською частиною у вигляді хмарних функцій, що проводять SSR, серверною частиною у вигляді хмарної функції, що надає доступ до моноліту, а також реляційної бази даних.

6.2 Системне програмне забезпечення

– середовище виконання клієнтської та серверної частини: Node.js версії пізніше 20;

– сервера клієнтської та серверної частин – Vercel (ОС Amazon Linux 2023);

– автоматичне розгортання оновлень через систему CI/CD GitHub (ОС Linux).

6.3 Мережне програмне забезпечення

– підтримка захищеного підключення HTTPS (протоколи TLS/SSL);

– підтримка протоколу WebSockets є опціональною з можливістю переходу на механізм HTTPS «Long Polling».

6.4 Програмне забезпечення ведення інформаційної бази

– СКБД PostgreSQL;

– засіб об’єктно-реляційного відображення даних, міграції структури та початкового наповнення БД – Prisma ORM;

– засіб моніторингу використання, налаштування доступу, резервного копіювання та перегляду вмісту БД – Supabase Console.

6.5 Мова і технологія розробки

– в якості мови програмування обрано TypeScript;

– клієнтська частина – React, Next.js, Tailwind CSS, «shadcn/ui», React Hook Form, TanStack Query, Recharts, Leaflet, next-intl;

– серверна частина – NestJS, Prisma ORM, Socket.IO, TomTom;

– інтеграція AI для імпортування даних – LangChain, LangGraph, Anthropic API;

– в якості бази даних обрано PostgreSQL;

– платіжна система – Paddle.

7. Вимоги до зовнішніх інтерфейсів

7.1 Інтерфейс користувача

– підтримка масштабів екрану від 375x667 до 2560x2488 (підтримувані масштаби екрану в Google Chrome DevTools);

– дизайн із початковою підтримкою мобільних пристроїв, зокрема сторінка відстеження замовлення для водія;

– наявна бічна панель для навігації в межах сервісу;

– підтримка кількох мов локалізації інтерфейсу, починаючи з англійської та української з можливістю перемикання між ними;

– підтримка світлої та темної тем із визначенням значення за замовчуванням від системи та можливістю ручного перемикання.

7.2 Апаратний інтерфейс

Немає особливостей щодо вимог апаратного інтерфейсу. Підтримка GPS не є обов’язковою, оскільки надаватиметься можливість ручного введення локації за картою чи текстовою адресою.

7.3 Програмний інтерфейс

– буде використано архітектурний підхід REST API для запитів між клієнтською та серверною частинами;

- автентифікація використовуватиме cookies для встановлення та зберігання інформації про сесію користувача;
- API інтеграція із SMTP-сервером для надсилання посилань для входу в систему;
- API інтеграція із сервісом розрахунку маршруту;
- API інтеграція для прямого та зворотного геокодингу;
- API інтеграція із сервісом чатів;
- API інтеграція із провайдером моделі LLM;
- API інтеграція із сховищем файлів;
- API інтеграція з поширеним OAuth провайдером для спрощеного отримання поштової адреси для входу.

7.4 Комунікаційний протокол

- використання HTTPS для стандартних запитів між клієнтом та сервером;
- використання WebSockets (HTTPS у випадку відсутності) для комунікації про події в режимі реального часу;
- використання SMTP для надсилання листів на електронну пошту.

8. Властивості програмного забезпечення

8.1 Доступність

Vercel забезпечує доступність 99.99% часу на рік.

8.2 Супроводжуваність

- розбиття серверної частини на бізнес-модулі для кращого розуміння коду та розмежування обов’язків;
- створення README файлів та реалізація опису API серверної частини згідно специфікації OpenAPI;
- організація файлової структури клієнтської частини за функціональним принципом;
- збір аналітики Vercel для перегляду результатів запитів, помилок, часу завантажень.

8.3 Переносимість

Next.js та NestJS можливо хостити за межами Vercel як звичайні Node.js сервери. При цьому втрачається гнучкість хмарних функцій, які дозволяють адаптуватися до зміни кількості запитів в одиницю часу. Також буде відсутня змога використати CDN сервери, що покращують швидкість завантаження сторінок незалежно від місцезнаходження клієнта.

8.4 Продуктивність

Час здійснення запитів на читання чи запис не має перебільшувати 1 секунди, що реалізується засобами збільшення кількості запущених хмарних функцій із максимальною кількістю 100,000 функцій працюючих одночасно.

Час здійснення імпорту даних засобами ШІ не має перевищувати 30 секунд для файлів розміром менше 10 МБ.

8.5 Надійність

- видалення сутностей вантажів та замовлень відбувається шляхом встановлення часової мітки для збереження цілісності даних та аудиту;
- завантаження файлів відбувається в два етапи (генерація посилання від Supabase, створюється запис про файл у БД із статусом «pending», і успішне завантаження файлу за посиланням, після чого запис в БД стає «confirmed»). Записи про файли, що залишилися «pending», видаляються через періодичну перевірку записів.

8.6 Безпека

- автентифікація клієнтської частини за допомогою session token cookie;
- автентифікація за допомогою API ключів;
- наявність RBAC-системи для детального опису прав доступу як користувачів, так і API ключів;
- використання Vercel аналітики лише за згодою користувача;
- використання імпорту даних за допомогою ШІ лише за згодою користувача;
- обмеження на кількість виконання запитів до серверної частини в одиницю часу в межах API ключа;

– при запиті на видалення користувача чи компанії відбувається анонімізація даних та видалення згідно GDPR.

9. Інші вимоги

Можливість оновлення лімітів безкоштовного плану вебплатформи без необхідності повторного розміщення клієнтської та серверної частин завдяки використанню метаданих платіжного сервісу, де вказано ціни на кожен вид послуги.

Висновки до розділу 2

У даному розділі проведено аналіз існуючих технологій реалізації клієнтської та серверної частин, обґрунтовано вибір мови програмування та стеку, що буде використано для проєктування та реалізації архітектури вебплатформи. Зокрема обрано екосистему Node.js із мовою програмування TypeScript. Клієнтська частина матиме оптимізований процес багатьох компонентів на сервері для зменшення навантаження на браузер кінцевого користувача, надаючи вже готову HTML-розмітку.

Серверна частина буде побудована за модульним принципом, що полегшує процес розробки та супроводу, ізоляцію бізнес-логіки та виставлення окремих інтерфейсів лише за вимогою, що структурує та зменшує взаємозалежність коду. Обрано тип та вид СКБД, що буде використано під час проєктування моделі даних та реалізації алгоритмів збереження та отримання даних.

Сформовано специфікацію вимог до майбутньої платформи організації перевезень вантажів на основі функціоналу існуючих рішень та їх недоліків. Описано призначення та межі проєкту, функціональні вимоги кожного модулю, стек технологій, а також нефункціональні вимоги, зокрема доступність, супроводжуваність, надійність, переносимість, продуктивність та безпека.

Також зазначено вимоги щодо інтерфейсу користувача, апаратного та програмного інтерфейсу.

3 АРХІТЕКТУРА, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ЗАСТОСУНКУ

Після аналізу предметної області, визначення стеку технологій, функціональних та нефункціональних вимог, можливо перейти до проєктування вебзастосунку. Буде визначено сценарії використання, змодельовано загальну архітектуру, структуру бази даних та класів, стани та інтерфейс системи.

3.1 Аналіз сценаріїв використання

На основі функціональних вимог можливо змоделювати більше детальний опис щодо способів використання системи кінцевими користувачами. Візуалізацію сценаріїв використання системи наведено в додатку А. Для текстового представлення сценаріїв використано техніку структурованого опису Use Cases.

Назва Use Case: Реєстрація компанії на платформі.

Опис: Власник або представник компанії вказує електронну адресу, на яку приходить магічне посилання, або використовує свій обліковий запис Google. Після переходу за посиланням чи повернення з вікна авторизації Google створюється його обліковий запис та здійснюється перенаправлення на заповнення даних про компанію у випадку їх відсутності. Користувач вводить назву, ЄДРПОУ, телефон, адресу компанії, а також власні дані: ім’я, прізвище і телефон. Система створює компанію із наданими даними, створює стандартний набір ролей, зокрема роль «Адміністратор» із повним доступом, яку автоматично задано для першого користувача.

Назва Use Case: Додавання нового користувача до платформи від компанії.

Опис: Користувач, що має роль із можливістю додавання користувачів, відкриває модуль «IAM», підрозділ керування користувачами. Користувач вводить електронну адресу, ім’я та прізвище, телефон, а також призначає ролі для майбутнього користувача, після чого система створює обліковий запис та

пов’язує його з обраною роллю. Дія створення користувача записується в журналі аудиту.

Альтернативні сценарії:

1) введена електронна адреса вже існує;

2) обрана роль була видалена іншим адміністратором на момент підтвердження створення користувача;

3) у випадку недостатніх прав створення користувача необхідний інтерфейс для початку сценарію відсутній.

Назва Use Case: Створення ролі з довільним набором дозволів.

Опис: Користувач, що має роль із можливістю створення ролей, відкриває модуль «IAM», підрозділ керування ролями. У формі створення ролі вказується назва, опис, обирається кожен дозвіл щодо сутності окремо, або використовуються допоміжні кнопки «Обрати всі» та «Тільки читання». Після відправки форми система повідомляє про успішне створення та відображає нову роль серед списку інших ролей. Створення ролі записується в журналі аудиту.

Альтернативні сценарії:

1) назва ролі вже існує;

2) при видаленні ролі система видаляє її зв’язки з користувачами, після чого видаляє сам запис про роль. Користувачі залишаються без ролі для використання;

3) оновлення ролі автоматично оновлює фактичний доступ користувачів, що асоціюються з нею.

Назва Use Case: Додавання відомостей про транспортний засіб.

Опис: Користувач, що має роль із доступом до створення транспортного засобу, відкриває модуль «Transport Fleet». У формі вводиться номерний знак, тип транспорту (фургон, тент, цистерна та ін.), тип кузову (вантажівка, причіп, фургон та ін.), опціональна поточна локація, опціональні відомості про підтримуваний вантаж: вага, тоннаж, класи ADR, габарити, підтримка швидкопсувних продуктів, температурний контроль, тип завантаження (збоку, зверху та ін.), умови транспорту (жорсткий борт, м’який тент, підйомник та ін.).

Операції логуються в журналі аудиту. Після створення запису про транспорт система відображає його в списку та надає можливість завантажити документи таких форматів: «jpeg», «png», «webp», «pdf», «docx», «xlsx» розміром до 10 МБ кожен.

Альтернативні сценарії:

1) номерний знак вже зареєстрований на платформі;

2) документ не відповідає зазначеним вимогам та не може бути завантажений до системи;

3) під час завантаження документу виникла помилка. Документ залишається обраним та може бути повторно надісланий.

Назва Use Case: Створення замовлення на перевезення.

Scope: Модуль керування замовленнями.

Level: Мета користувача (user goal).

Primary Actor: користувач компанії власника вантажу (менеджер чи адміністратор).

Stakeholders and Interests:

- компанія, що володіє вантажем: зацікавлена у можливості отримання релевантних та економічно вигідних пропозицій на виконання перевезення;
- компанії, що надає послуги з перевезення: зацікавлені в знаходженні вантажу з доступними подробицями щодо маршруту та оплати;
- платформа: зацікавлена в отриманні повного обсягу даних для публікації вантажу та коректного розрахунку маршруту.

Preconditions:

- менеджер має роль із правом створення замовлень;
- компанія додала відомості про вантаж, який необхідно перевезти.

Success Guarantee:

- замовлення збережене зі статусом «чернетка»;
- після публікації чернетки відбувається розрахунок маршруту згідно вказаних точок завантаження та розвантаження з урахуванням вказаних обмежень на транспортування вантажу;

– після публікації чернетки замовлення доступне для пошуку на платформі та подання пропозицій на перевезення.

Main Success Scenario:

– менеджер відкриває форму створення замовлення;

– вказуються назва замовлення, опис, пов’язаний запис про вантаж;

– вказуються місця завантаження та розвантаження зі списку складів або через сервіс прямого та зворотного геокодування, дату та бюджет;

– вказується існуючий вантаж та деталі оплати: спосіб (банківський переказ, переказ на картку, готівка), умови (передплата, після завантаження, після доставки, відкладений платіж), період здійснення платежу, чи включає ціна ПДВ;

– вказуються вимоги щодо транспорту: спосіб доставки (FTL/LTL), тип транспорту та кузову, кількість транспорту;

– вказується сценарій розпорядження вантажем після виконання замовлення: видалити вантаж, передати у володіння перевізнику, іншій компанії чи залишити без змін;

– система створює замовлення зі статусом «чернетка» з можливістю зміни даних, публікації та запиту на розрахунок маршруту із збереженням версій кожного розрахованого маршруту;

– менеджер публікує замовлення, воно стає доступним для пошуку перевізниками.

Extensions:

1) сервіс маршрутизації недоступний – система повертає помилку та надає менеджеру можливість повторити запит до сервісу;

2) досягнуто ліміту безкоштовних замовлень – можливо створити, проте не можна опублікувати замовлення, доки не оформлено підписку;

3) обраний тип транспорту не сумісний згідно вимог до класу ADR – система повідомляє про помилку та не дозволяє створити замовлення;

4) дані про замовлення змінюються до його публікації – система одразу зберігає зміни;

5) дані про замовлення змінюються після його публікації – система зберігає зміни та анулює пропозиції, а перевізники отримують повідомлення про анулювання їх пропозицій через зміну замовлення;

6) дані про замовлення необхідно змінити після прийняття пропозиції – можливо змінити лише дати, місця завантаження і розвантаження, дані про пакування вантажу. Створюється запит на здійснення змін, який необхідно прийняти перевізнику для їх застосування.

Special Requirements:

Сервіс для розрахунку маршруту має враховувати такі дані: вага та габарити транспорту, клас ADR.

Technology and Data Variations List:

- адреси завантаження та розвантаження можливо вказати через адресу існуючого складу або довільну адресу, вказану в текстовому форматі чи обрану на карті;
- дані про вантаж можливо обрати як вказавши існуючий вантаж, так і ввівши дані про нього безпосередньо під час створення замовлення.

Frequency of Occurrence:

Замовлення можуть створюватися як раз на кілька днів, так і кілька десятків разів на день залежно від масштабу компанії, що має вантажі для перевезення.

Miscellaneous:

Можливість автоматичної пропозиції створених замовлень перевізникам, з якими була попередня співпраця та які мають доступний транспорт, що задовольняє вимогам щодо перевезення.

Назва Use Case: надання пропозицій перевізниками на виконання замовлення.

Scope: Модуль керування замовленнями.

Level: Мета користувача (user goal).

Primary actor: користувач компанії-перевізника (менеджер чи адміністратор).

Stakeholders and Interests:

– компанії-перевізники: зацікавлені знайти економічно-вигідне замовлення згідно можливостей транспорту, а також із відомим маршрутом і деталями оплати;

– компанія-вантажовласник: зацікавлена в знаходженні економічно-вигідних пропозицій від перевірених перевізників із підходящими умовами транспортування вантажу;

– платформа: зацікавлена в заключенні згоди на перевезення між перевізником та вантажовласником для отримання прибутку.

Preconditions:

– замовлення опубліковано та доступно для пошуку (статус «опубліковано» чи «прийом пропозицій»;

– перевізник має транспорт, сумісний із класом безпеки вантажу;

– перевізник не вичерпав ліміт на виконання замовлень у межах свого плану.

Success Guarantee:

– пропозиція створена та відображається для перевізника та власника вантажу на сторінці замовлення;

– перша пропозиція переводить замовлення в статус «прийом пропозицій»;

– власник вантажу отримує сповіщення щодо пропозицій в режимі реального часу;

– для компаній з’являється можливість почати чат в межах замовлення.

Main Success Scenario:

– менеджер компанії-перевізника знаходить та відкриває сторінку замовлення, де вказано назву, опис, маршрут із картою, відомості про пов’язаний товар, дати початку та завершення виконання замовлення, бюджет, вимоги до транспорту, що відбувається із записом про вантаж після виконання замовлення, а також форму створення пропозиції;

– менеджер вводить суму, за яку компанія може виконати замовлення, пропонований транспорт, дату початку та завершення, опис пропозиції, і надсилає пропозицію;

– система надає повідомлення про успішно створену пропозицію та замінює форму створення на перегляд із можливістю зміни та відкликання пропозиції;

– система надсилає повідомлення власнику вантажу в реальному часі та оновлює таблицю пропозицій на сторінці замовлення.

Extensions:

1) компанія-перевізник видаляє транспорт, доступність чи призначення водія – система автоматично відкликає пропозиції, що пов’язані з транспортом пропозиції;

2) компанія-перевізник вичерпала кількість прийнятих замовлень згідно плану – система забороняє створення нових пропозицій;

3) компанія-перевізник відкликала замовлення – компанія-власник вантажу отримує сповіщення, пропозиція зберігається зі статусом «відкликана» для аудиту;

4) компанія-перевізник оновила замовлення – компанія-власник вантажу отримує сповіщення.

Special Requirements:

– сповіщення мають надходити в режимі реального часу та зберігатися для перегляду користувачами в майбутньому;

– компанії повинні мати змогу переглянути сторінки відомостей потенційного контрагента;

– компанія-власник вантажу має змогу переглянути відомості про транспорт, водія, а також ціну, дату, та опис пропозиції.

Technology and Data Variations List:

– можливість надати пропозицію як через вебінтерфейс, так і через прямий запит до серверу;

– наявність класу ADR вантажу автоматично фільтрує транспорт, який можливо додати до пропозиції;

– орієнтовні дати початку та завершення замовлення є опціональними для пропозиції.

Frequency of Occurrence:

Можливі десятки пропозицій на день на одне замовлення.

Назва Use Case: Прийняття пропозиції та призначення замовника.

Scope: Модуль керування замовленнями.

Level: Мета користувача (user goal).

Primary Actor: користувач компанії власника вантажу (менеджер чи адміністратор).

Stakeholders and Interests:

– компанія-власник вантажу: зацікавлена у виконанні замовлення із найкращим співвідношенням ціни, якості та швидкості перевезення;

– компанія-перевізник: зацікавлена у виконанні замовлення з можливістю комунікації з контрагентом та маршрутизації в режимі реального часу;

– платформа: зацікавлена в прийнятті пропозиції та початку виконання замовлення для застосування тарифного плану для обох сторін.

Preconditions:

Компанія-власник вантажу замовлення не досягла ліміту на замовлення, опублікувала замовлення та має активну пропозицію для нього зі сторони компанії-перевізника.

Success Guarantee:

– замовлення має статус «призначено», та знімається з загального доступу;

– компанія-перевізник має доступ до замовлення та можливість оновлювати його статус виконання;

– існуючі пропозиції інших перевізників скасовано;

– система здійснює запис про факт оновлення замовлення та прийняття пропозиції.

Main Success Scenario:

– менеджер компанії-власника вантажу відкриває сторінку опублікованого замовлення і переглядає таблицю пропозицій: назва компанії, ціна, транспорт, орієнтовні дати початку та завершення, статус, дата подачі, швидкі дії (прийняти чи відхилити). При натисканні на рядок відкривається модальне вікно із подробицями пропозиції;

– менеджер компанії-власника обирає прийняти пропозицію, система запитує підтвердження, менеджер підтверджує операцію;

– система скасовує інші пропозиції, задає перевізника, транспорт і водія в якості виконавців замовлення, сповіщає обидві сторони та робить системний запис у чаті замовлення;

– відбувається розрахунок маршруту у випадку його відсутності на етапі публікації замовлення.

Extensions:

– обрана пропозиція на момент прийняття була скасована – система помічає ставку як скасована для цілей аудиту та не дозволяє її прийняти;

– прийнято пропозицію щодо іншого замовлення – поточне замовлення знімається з публікації, пропозиції скасовуються системою;

– помилка при розрахунку маршруту – контрагенти можуть перепланувати та запросити повторний розрахунок після прийняття пропозиції.

Special Requirements:

– задання перевізника як поточного та скасування інших пропозицій мають відбуватися в одній транзакції;

– повідомлення мають зберігатися в базі даних для можливості їх прочитання незалежно від перебування на платформі в момент прийняття пропозиції.

Technology and Data Variations List:

– сервіс розрахунку маршруту має підтримувати габарити, вагу транспорту та клас небезпеки вантажу;

– оновлення подробиць замовлення має бути можливе через окремий механізм узгодження змін в двосторонньому порядку для гарантії інформування контрагентів;

– скасування виконання активного замовлення також відбувається шляхом узгодження обох контрагентів.

Frequency of Occurrence: один раз на замовлення.

3.2 Архітектура застосунку

Архітектура застосунку визначає складові майбутньої системи та взаємозв’язки між ними як на фізичному рівні розміщення на серверах, так і на рівні логіки виконання функціональних частин. Під час проєктування даних компонентів необхідно визначити їх зміст, зв’язки із зовнішніми елементами, а також надати можливості виконання нефункціональних вимог, зокрема швидкості виконання операцій та доступності складових під час великих навантажень, їх здатність адаптуватися до динамічних змін, що особливо важливо із наявністю сезонних коливань попиту на транспортні перевезення сільськогосподарської продукції. Із метою унаочнення отриманої архітектури буде використано діаграми розгортання, компонентів, класів та станів для її розгляду на логічному та фізичному рівні.

Згідно зі сформованою специфікацією вимог, вебплатформа організації перевезень вантажів буде побудована згідно трьохланкової архітектури із клієнтською і серверною частинами, а також базою даних. Її реалізація передбачає використання кілька основних та допоміжних серверів. Вони відповідатимуть як за реалізацію головного функціоналу, так і за надання окремих можливостей, що покращують досвід майбутніх користувачів.

Клієнтська частина буде використовувати хмарний сервіс компанії Vercel, що складається з таких частин:

– безсерверні функції, на яких відбуватиметься рендер сторінок, обробка API запитів для фреймворку авторизації, та проксі-функція, що перевіряє сесію користувача та робить перенаправлення у випадку її відсутності при спробі відкрити захищені сторінки;

– мережа доставки контенту, що надає статичні сторінки та ресурси вебплатформи.

Серверна частина складатиметься з серверу NestJS, який отримує всі запити від клієнтської частини, реалізує бізнес-логіку, а також підтримує фреймворк автентифікації, який використовує клієнтська частина. Також тут здійснюється комунікація із сервером бази даних та допоміжними сервісами.

База даних використовуватиме сервер PostgreSQL, що надається компанією Supabase. Він має вебінтерфейс для адміністрування, налаштування безпеки, перегляду даних та логування.

Допоміжні сервери включають в себе:

– Supabase сервери зберігання файлів, що використовуватиметься для підтримки завантаження документів стосовно транспорту чи замовлень на платформі;

– TalkJS сервери для зберігання чатів;

– Paddle сервери для обробки та виставлення рахунків за використання вебплатформи користувачами, Anthropic сервери для використання можливостей LLM;

– сервери GeoApify для прямого та зворотного геокодування;

– сервери TomTom для цілей маршрутизації із врахуванням погодних умов, габаритів транспорту та класів ADR.

Подальший процес проєктування полягає у створенні структури NestJS застосунку. Даний фреймворк заснований на парадигмі модулів, що мають власні сервіси та можуть поширювати їх через механізм ін’єкції залежностей (імпорт модулів).

Головним модулем NestJS застосунку є AppModule, який імпортує як модулі бізнес-логіки, так і глобальні системні модулі, а також налаштовує і

здійснює запуск застосунку. Наступні за важливістю модулі підключають базу даних, провайдера автентифікації та авторизації, конфігурації, після яких йдуть модулі окремих сервісів, що являють собою згруповану бізнес-логіку вебплатформи, зокрема:

– модуль керування доступом, що надаватиме як сервіси керування користувачами, ролями, API ключами, реєстрації, аудиту та згоди, але й Guard-класи для захисту кінцевих точок самого серверу та накладання обмежень на кількість запитів за одиницю часу до кожної кінцевої точки;

– модуль керування вантажами, що також відповідатиме за керування складами та відповідну аналітику;

– модуль керування транспортом, що також відповідатиме за керування призначеннями, їх пошуком, доступністю та аналітикою;

– модуль керування замовленнями, що також відповідає за керування пропозиціями, обрання пропозиції, виконання замовлень, запити на скасування та оновлення активних замовлень;

– модуль імпортування та експортування даних з та до файлів;

– модуль керування підписками та платежами (інтеграція з Paddle);

– модуль сповіщень, що дозволяє їх створення з інших сервісів, зберігання та читання;

– глобальний системний модуль, що реєструє cron-функції.

Для інтеграції повідомлень із WebSocket шлюзом варто створити уніфікований інтерфейс на рівні всього застосунку, який можливо буде поширити та використати серед інших сервісів. Також варто уніфіковати в окремий модуль допоміжні сервіси, що використовуються в різних частинах застосунку, як-от GeoApify та сервіс контролю за частотою звернень до кінцевих точок.

Компонентну діаграму серверної частини вебплатформи наведено на рис. 3.1. На ній наведено модулі, що будуть створені для кожного сервісу бізнес-логіки, із групуванням за пакетами, що являтимуть потенційні сервери мікросервісів. Пакет логістики відповідатиме за логіку керування сутностями

складів, вантажів, транспорту, а також пропозицій, замовлень та їх виконання. Пакети авторизації та підписки скаладаються з одного модулю, проте є важливими складовими платформи, що використовуються в усіх її частинах, а тому потребують можливості більш гнучкого масштабування їх серверів.

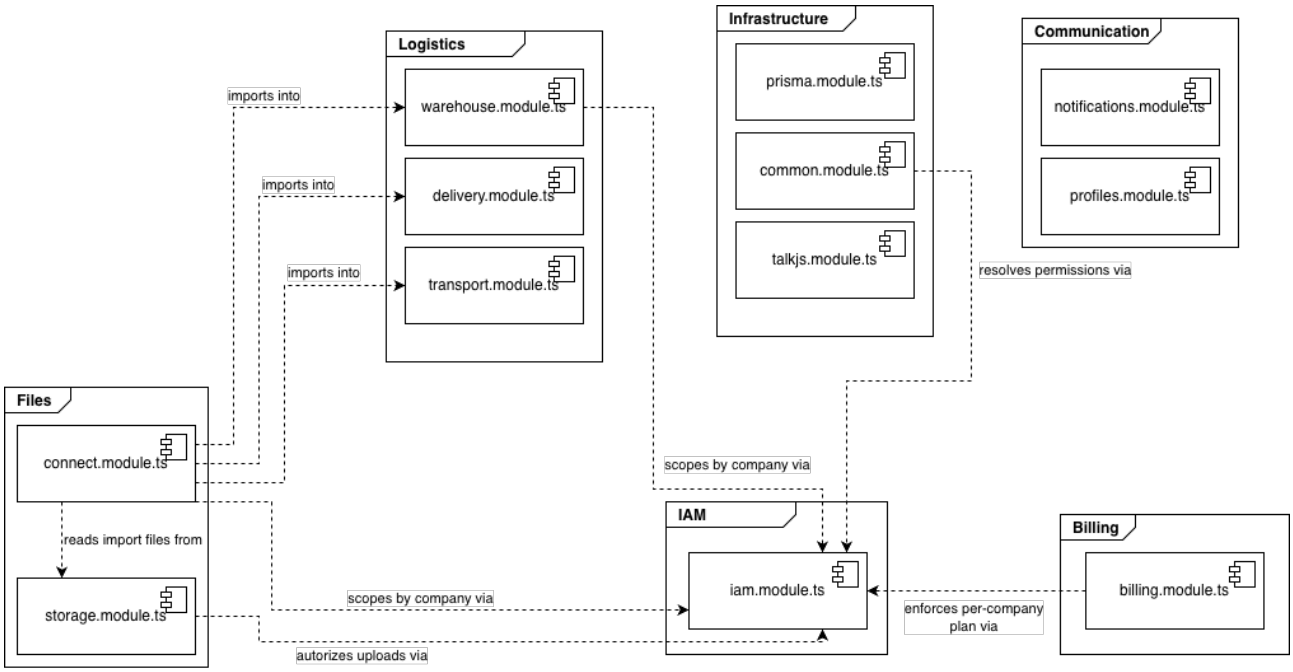

Рисунок 3.1 – Діаграма компонентів вебплатформи

Дана структура та взаємозв’язки компонентів дозволяють полегшити розробку завдяки розділенню обов’язків відповідно до бізнес-логіки, а також потенційний перехід на мікросервісну архітектуру, інкапсулювавши кожен пакет компонентів у окремий сервіс.

3.3 Структура бази даних та діаграма класів

Подальше проєктування архітектури полягає в створення структури та зав’язків на рівні даних, з якими працюватиме платформа. Для унаочнення побудови фізичної моделі даних буде створено діаграму бази даних, що містить таблиці сутностей з їх атрибутами, типами даних та зв’язками.

Це дозволить ефективно перейти до реалізації схеми бази даних з урахуванням особливостей обраної бази даних PostgreSQL, а також спростить подальше потенційне розширення вебплатформи із додаванням нових та зміною існуючих сутностей згідно функціональних та нефункціональних вимог.

Діаграму бази даних представлено на рисунку 3.2. Із метою унаочнення умовного відношення сутностей до певних модулів бізнес-логіки використано кольори.

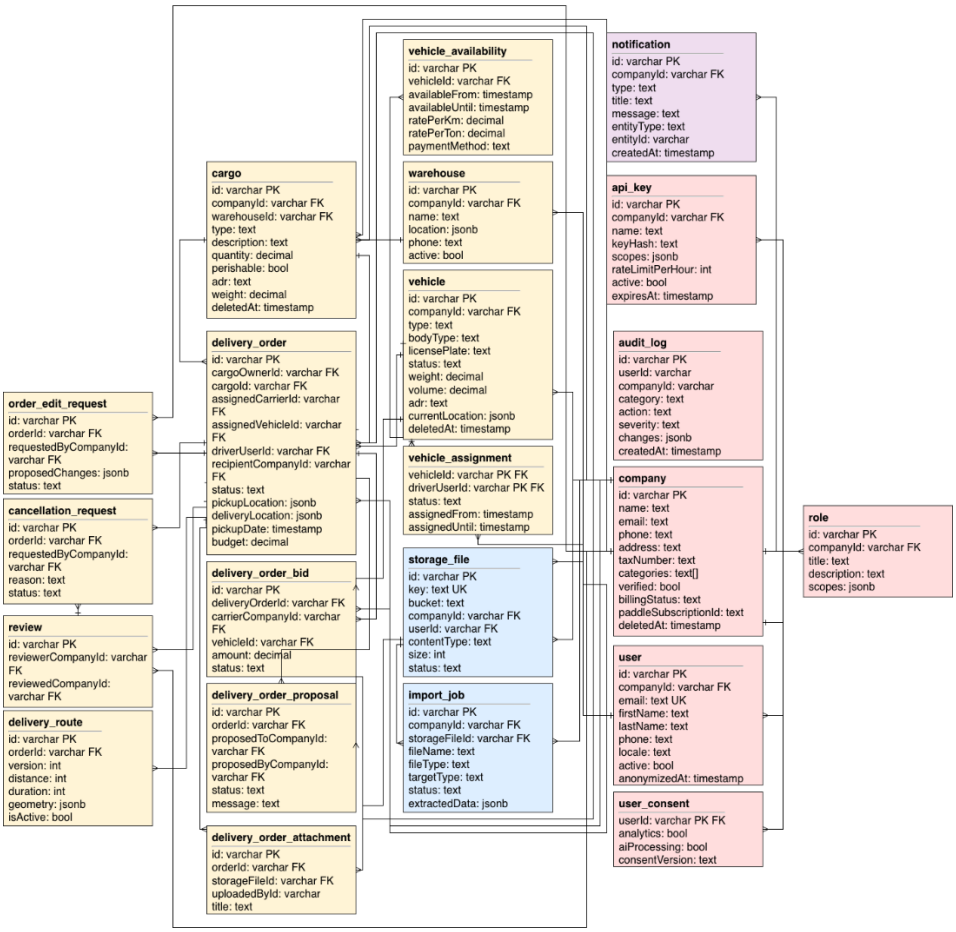

Рисунок 3.2 – Схема бази даних вебплатформи

На схемі позначено різними кольорами семантичне поєднання сутностей. Так, головними сутностями, до яких прив’язані інші дані та за якими визначається доступ до них, є сутності: компанії, користувача, ролі та API ключі, а також адміністративні сутності, зокрема аудит та набір згод від користувача.

Компанія має зв’язки з усіма іншими сутностями для ізоляції даних поміж різними компаніями та наданні доступу користувачам та API ключам до всіх ресурсів в межах одної компанії в залежності від ролей, що надають доступ до операцій над сутностями.

Останній є важливим для використання аналітичних сервісів та надання згоди на доступ до файлів під час автоматизованого імпорту з файлів довільних форматів. Для можливості перегляду сповіщень у випадку відсутності

користувача на платформі в момент їх виникнення, а також з метою їх логування використовується окрема сутність, що містить тип сповіщення, пов’язану сутність, в контексті якої відбулася зміна, заголовок та опис.

Для імпортування файлів використано сутність окремого файлу, яка зберігає відомості про місцезнаходження файлу на сервері Supabase File Storage, а також його приналежність до компанії, користувача, та інших сутностей, які підтримують файли, як-от вантаж, транспорт і замовлення. Для зручності організації асинхронного процесу обробки вмісту файлів використано окрему сутність, що містить дані про окреме завдання обробки файлу, поточний етап обробки та попередньо отримані дані.

Наступними ключовими сутностями є: транспорт, вантаж, замовлення та пропозиції. Пов’язані сутності з транспортом є призначення водія та публічна доступність транспорту. Вантаж може бути пов’язаний із сутністю складу, на якому він зберігається, а також він є обов’язково пов’язаний із замовленням на його перевезення. Після виконання замовлення вантаж може бути переведений до власності іншої компанії чи видалений зі збереженням можливості аудиту в межах історії замовлень.

Для замовлення також дотичними сутностями є запити на зміну та скасування замовлення, пропозиції як від перевізників, так і потенційним перевізникам, що подаються їм напряму від власників вантажу. Окремо зберігаються сутності розрахованих маршрутів для замовлень, що підтримують версіонування, а також сутність відгуку щодо здійсненого перевезення, що може бути переглянута іншими компаніями в межах перевірки рейтингу як компаній, що надають замовлення, так і компаній-перевізників.

Дану схему бази даних можливо описати з метою подальшої генерації запитів та версіонування засобами Prisma ORM, що надає Prisma Schema Language [26]. На її основі Prisma генерує допоміжні інтерфейси сутностей та клієнт, що дозволяє здійснювати типізовані запити до кожної сутності. Схема вказує атрибути і зв’язки між сутностями та дозволяє зробити точну копію схеми

бази даних у вигляді діаграми класів. Концептуальну діаграму класів для даної схеми бази даних наведено на рисунку 3.3.

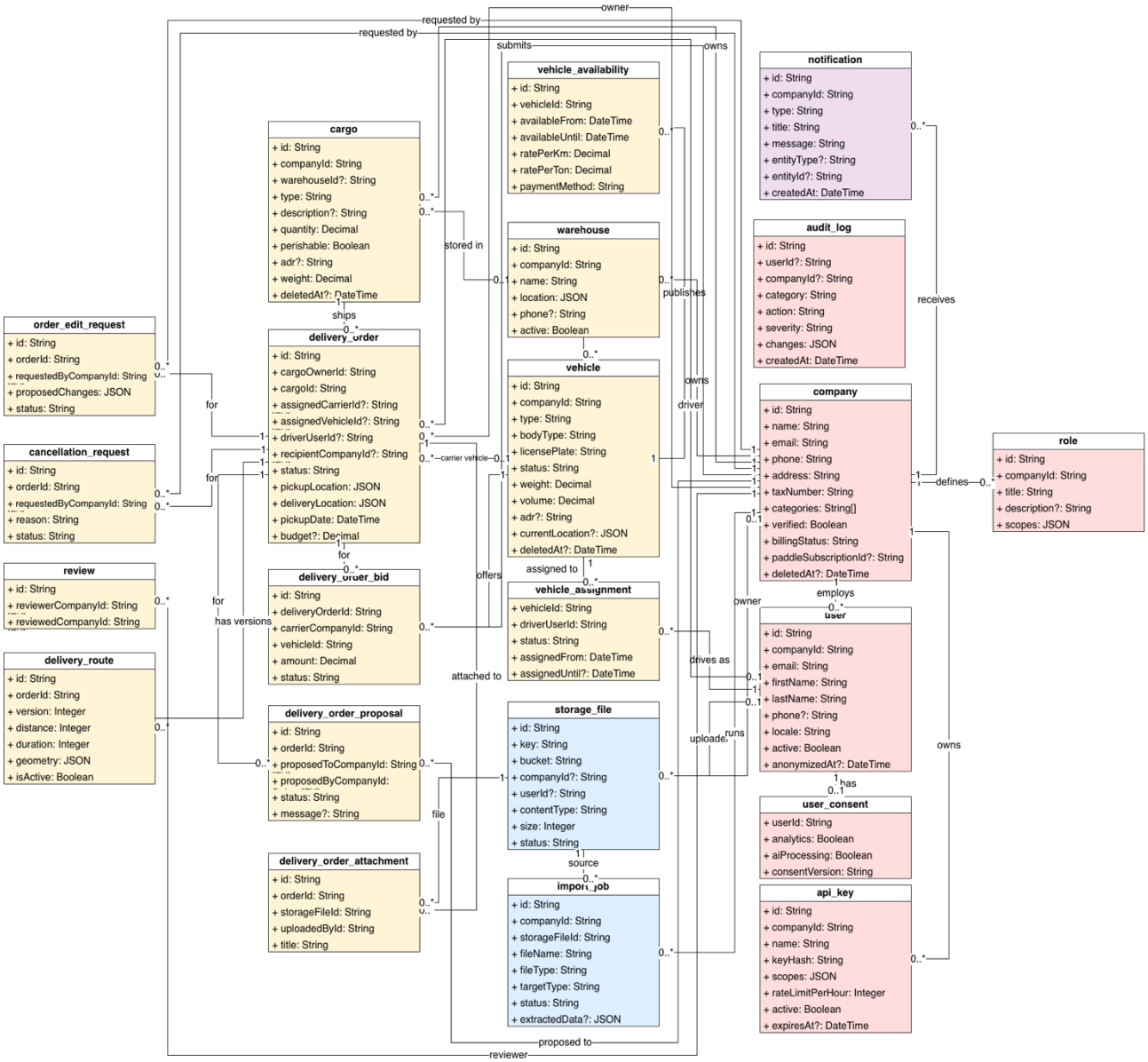

Рисунок 3.3 – Концептуальна діаграма класів вебплатформи

Дана архітектура схеми даних та їх класів в об’єктно-реляційному представленні дозволяє здійснювати гнучкі запити для отримання пов’язаних сутностей один-до-одного та один-до-багатьох із дотриманням нормалізації та розподілом відповідальності за збереження та операції над певним видом даних.

3.4 Діаграми станів та послідовностей

Діаграма станів дозволяє моделювати стани, в яких може перебувати система, і переходи між ними в залежності від подій, які можуть виникати

всередині системи або надходити за її межами. Даний етап моделювання архітектури є важливим для розуміння та реалізації системи, уникаючи конфліктів станів різних частин системи. Почати моделювання станів варто з ключового процесу – життєвий цикл замовлення. Його діаграму станів зображено на рисунку 3.4.

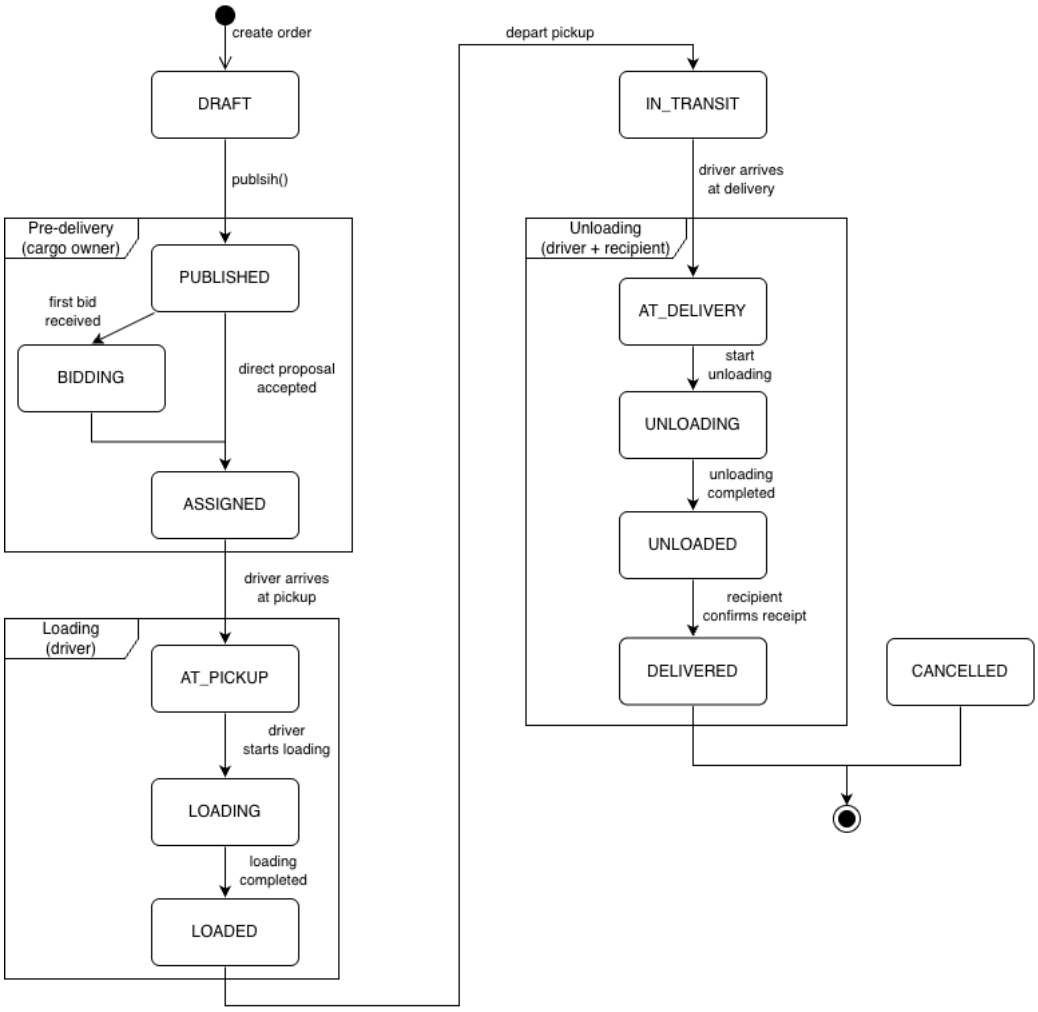

Рисунок 3.4 – Діаграма станів замовлення

Спочатку замовлення створюється в режимі чернетки та потім публікується. Замовлення може бути прийняте або через процес відбору пропозицій від перевізників, або через прийняття перевізником прямої пропозиції на перевезення від власника вантажу. Після цього розпочинається процес перевезення, де водій оновлює статус замовлення в залежності від етапу перевезення. Також активне замовлення може бути скасовано окремим запитом, що має бути узгоджений обома сторонами. Після успішного виконання

замовлення відбувається процес переміщення запису про вантаж залежно від обраної опції власником під час створення замовлення.

Для візуалізації порядку передачі даних та виконання операцій між компонентами системи варто розглянути створення діаграми послідовностей. Найбільш важливим процесом, який необхідно розглянути та змоделювати з точки зору послідовності виконання дій, є процес прийняття пропозиції на виконання замовлення. Його діаграму послідовностей наведено на рис. 3.5.

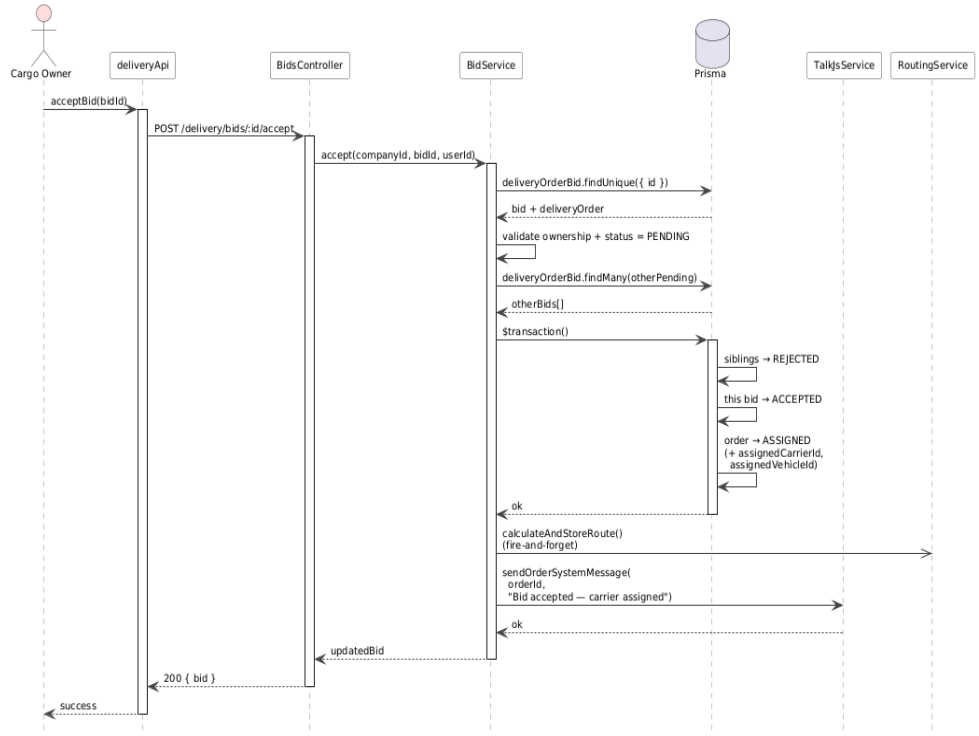

Рисунок 3.5 – Діаграма послідовності прийняття пропозиції

Дана діаграма зображує взаємодію клієнтської частини із серверною через HTTPS запит, що спочатку обробляється контролером відповідного модулю, потім передається на сервіс, де здійснюються запити до бази даних та оновлення пов’язаних сутностей у режимі транзакції. Після успішного оновлення даних відбувається розрахунок маршруту замовлення та створення системного повідомлення в сервісі TalkJS.

3.5 Діаграма розгортання застосунку

Для унаочнення фізичної архітектури платформи побудовано діаграму розгортання, наведену на рис. 3.6. На ній наведено основні сервери

інфраструктури Vercel, на яких виконується код клієнтської частини, інфраструктури Railway для виконання коду серверної частини, а також допоміжні сервери та зв’язки між ними.

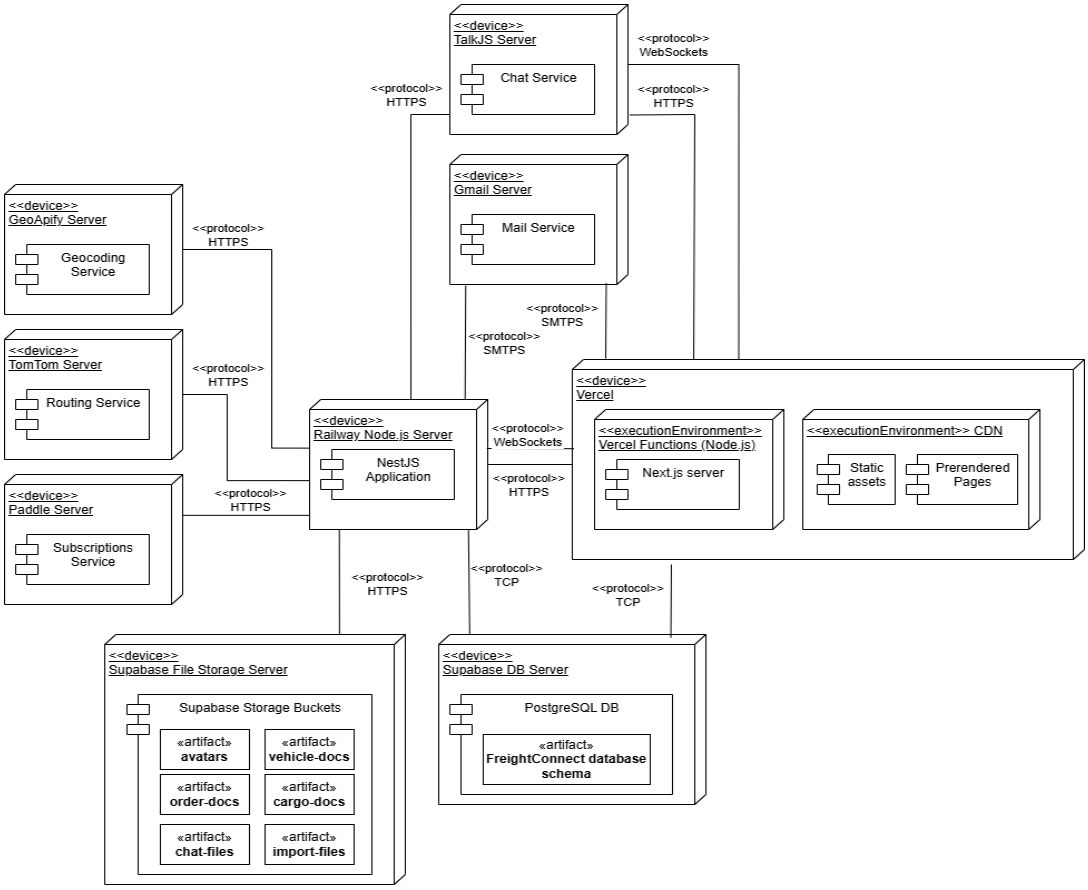

Рисунок 3.6 – Діаграма розгортання вебплатформи

Дана архітектура дозволяє реалізувати розподілену систему, яка використовує різні сервери для окремого набору завдань, що підвищує загальну відмовостійкість, а також дозволяє автоматично пристосовуватися до зміни навантаження шляхом горизонтального та вертикального масштабування. Спрощується потенційна можливість переходу на мікросервісну архітектуру для серверної частини.

3.6 Інтерфейс застосунку

Для побудови інтерфейсу повторно використовується принцип модульності, що й для серверної частини. Це спростить процес адаптації

користувача до вмісту системи завдяки організації логіки без значної кількості переходів між підсистемами.

В якості системи дизайну та принципів побудови інтерфейсу використовується бібліотека «shadcn/ui». Вона надає попередньо підготовлені компоненти в мінімалістичному дизайні із повною можливістю кастомізації завдяки фактичному копіюванню вихідного коду компонентів до репозиторію замість їх зберігання в окремому зовнішньому пакеті, як це відбувається з іншими бібліотеками компонентів. Вони дозволяють швидко створювати прототипи завдяки своїй готовності та використанню стандартних принципів дизайну разом із підтримкою клавіатури та налаштувань для людей із обмеженими можливостями, оскільки базуються на так само відкритих компонентах бібліотеки Radix.

Моделювання інтерфейсу застосунку починається із головної сторінки, яка надаватиме представлення щодо сервісу для користувачів, що починають знайомство із системою, а також має зацікавити їх до початку використання за принципом лендінгу. Попередню структуру даної сторінки наведено на рисунку 3.7.

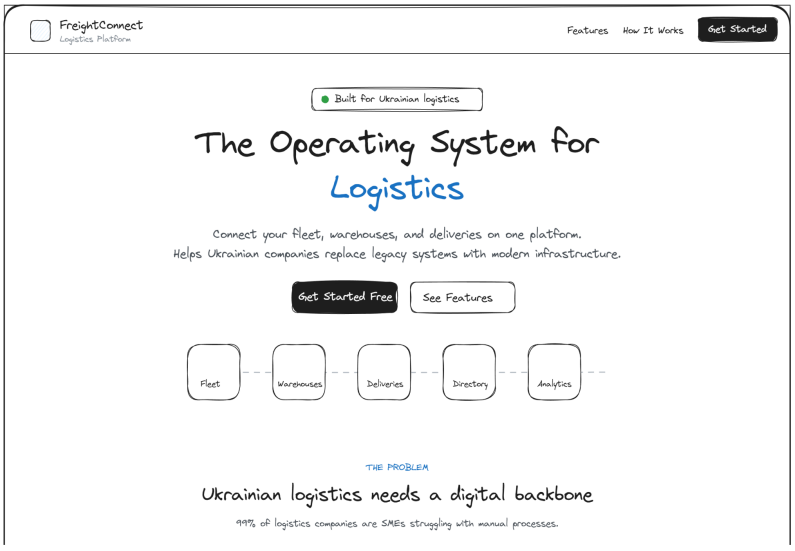

Рисунок 3.7 – Структура домашньої сторінки

Після авторизації та реєстрації важливою є сторінка навігації по всіх сервісах. Вона має бути інтуїтивно зрозумілою та не перевантажувати користувача інформацією. Її структуру наведено на рисунку 3.8.

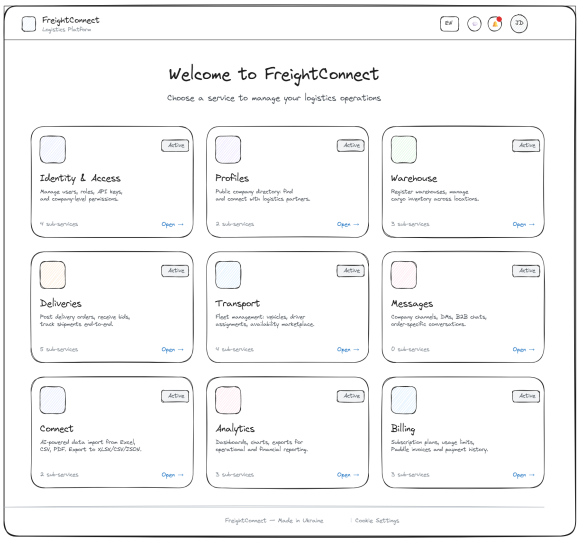

Рисунок 3.8 – Структура сторінки навігації по сервісах системи

Головна сторінка кожного сервісу має бути уніфікованою за структурою. Для цього вони матимуть загальну бічну панель навігації за підсервісами та головну панель, де відображатиметься вміст підсервісу разом із Breadcrumbs компонентом для навігації по вкладених відображеннях кожного підсервісу.

Висновки до розділу 3

У даному розділі проведено детальний аналіз та описано головні сценарії використання вебплатформи, зокрема використано коротку, поверхневу та повну форми опису сценаріїв. Змодельовано архітектуру, зокрема визначено трьохланкову архітектуру, основні та допоміжні сервери, створено діаграму їх розгортання. Наведено детальний опис складових серверної частини застосунку.

Визначено структуру бази даних та описано основні сутності, що будуть задіяні при розробці системи, і наведено їх діаграму. Також визначено концептуальну діаграму класів згідно схеми, що використовується для моделювання бази даних засобами Prisma ORM.

Створено діаграми станів та послідовностей для головних процесів у системі, що дозволить краще реалізувати фактичний алгоритм їх виконання. Також визначено базову розмітку інтерфейсу основних сторінок застосунку.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ

Після визначення сценаріїв використання та моделювання архітектури системи, структури її бази даних, класів та інтерфейсу основних сторінок, можливо перейти до програмної реалізації та тестування модулів. Реалізація полягає в кодуванні інтерфейсу клієнтської частини, бізнес-логіки на стороні серверу, накладанні обмежень для покриття таких окремих випадків, як: неавторизовані користувачі, доступ до логіки напряму без клієнтської частини через API ключі.

Також будуть створені набори тестів для перевірки роботи основного функціоналу та обробки окремих випадків. Для цього необхідно визначити тестові дані, шляхи обробки запитів та різні умови, а також створити моки для функцій, перевірка яких не є частиною тест-кейсів для даного модулю та які тестуються окремо.

4.1 Реалізація та тестування модулю аутентифікації та авторизації

Модуль аутентифікації та авторизації містить в собі логіку реєстрації компанії, керування її налаштуваннями та пов’язаними організаційними сутностями. Для реєстрації компанії спочатку необхідно створити користувача, що являтиме її власника чи головного адміністратора. Для цього етапу варто передбачити як створення облікового запису із вказанням електронної адреси вручну, так із її вказанням через використання облікового запису Google. Аби забезпечити більший ступінь надійності запису користувача разом із полегшенням реєстрації на платформі та подальшого входу, використано плагін, що реалізує магічне посилання в межах фреймворку Better Auth. Із використанням цього механізму, користувач має лише вказати електронну адресу, на яку буде надіслано посилання для входу. Після переходу за ним автоматично генерується пароль, який зберігається в базі даних у вигляді хешу. Панель реєстрації та входу наведено на рисунку 4.1. Її реалізовано у вигляді контейнеру з полем введення електронної адреси та кнопкою «Sign in with Google», що починає сценарій Google Oauth.

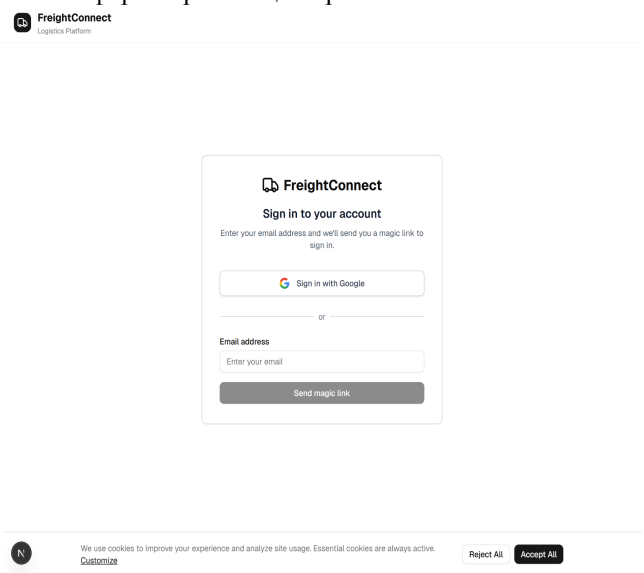

Рисунок 4.1 – Панель реєстрації

Після створення облікового запису система переадресовує користувача на сторінку заповнення деталей щодо компанії та безпосередньо його облікового запису. Сторінка являє собою двоетапну форму, що використовує можливості бібліотеки React Hook Form для динамічної реєстрації полів у спільному стані в компоненті-контейнері. Якщо набір полів для поточного кроку проходить валідацію правил, що вказані для кожного поля, активується можливість переходу на інший крок. Форму кроку для даних компанії наведено на рисунку 4.2.

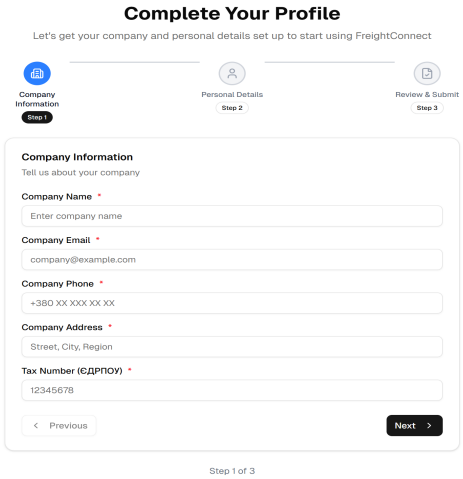

Рисунок 4.2 – Форма заповнення даних про компанію

Для компанії необхідно заповнити такі поля: назва, електронна адреса, телефон, юридична адреса, ЄДРПОУ. Для користувача необхідні такі поля: ПІБ,

робочий номер телефону. Перед відправкою форми можливо переглянути дані, заповнені в обидва кроки. Інтерфейс попереднього перегляду наведено на рисунку 4.3.

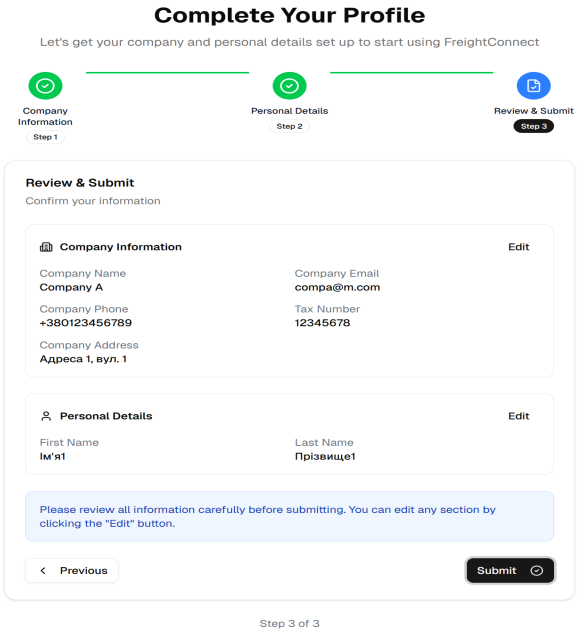

Рисунок 4.3 – Попередній перегляд багатокрокової форми

Створення запису про компанію, деталі користувача, стандартні ролі та призначення ролі «Адміністратор» першому користувачу здійснюється у транзакції Prisma, аби запобігти частковим змінам у випадку помилки на одному з кроків. Після проходження даної форми користувач отримує повний доступ до можливостей платформи. У модулі «IAM» можливо змінювати відомості про компанію, керувати ролями, генерувати API ключі та створювати нових користувачів, які отримуватимуть запрошення на електронну адресу і вже матимуть зв’язок із компанією в момент входу на сайт за магічним посиланням. Форму запрошення користувача на сторінці керування користувачами наведено на рисунку 4.4. Для її відправки необхідно ввести електронну адресу, ПІБ, а також робочий номер телефону. Після цього створюється запис про користувача та його деталей в межах одної транзакції Prisma.

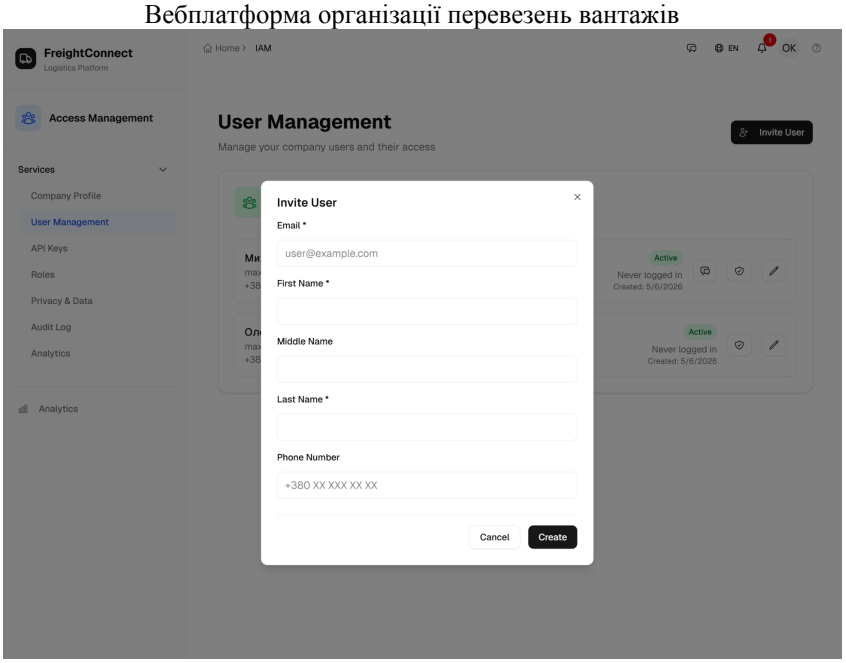

Рисунок 4.4 – Форма запрошення нового користувача

Налаштування ролі передбачає можливість контролю доступу за кожною сутністю із вказанням права на перегляд, оновлення, видалення, а також можливість безпосередньо змінювати ролі іншим користувачам. Форму налаштування ролі для користувачів наведено на рисунку 4.5

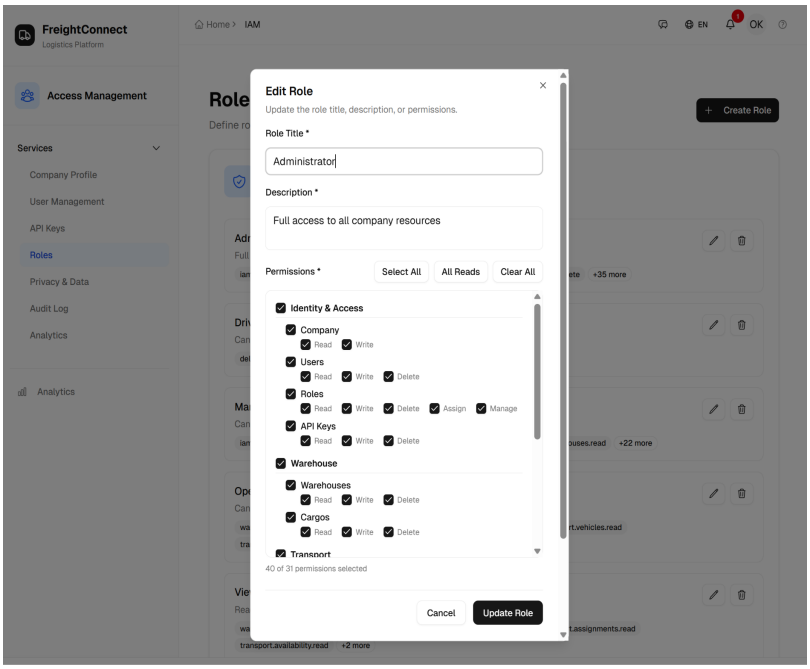

Рисунок 4.6 – Форма налаштування ролі

Для перевірки роботи функціоналу модулю на стороні серверу реалізовано юніт-тести, які перевіряють роботу кожного сервісу, з якого складається модуль.

Для зручності створення даних тестів використано фреймворк Vitest. Він надає необхідне середовище виконання, функції для порівняння значень та створення мок-функцій [27]. Тест-кейси для даного модулю наведено в таблиці 4.1.

Таблиця 4.1 – Основні тест-кейси сервісів модулю «IAM»

Назва кейсу	Вхідні дані	Очікуваний результат

Має відхиляти спробу повторного онбордингу при наявності компанії в користувача	{
	userId: “user-1”,
	dto: {
	companyName: 'Acme Logistics',
	companyEmail: 'ops@acme.test',
	companyPhone: '+380501110001',
	companyAddress: 'Kyiv, Khreshchatyk 1',
	taxNumber: '12345678',
	firstName: 'Олександр',
	lastName: 'Коваленко',
	}
	}
		throw ConflictException

Має бути ідемпотентним та не створювати дублікат призначення ролі	{
	companyId: 'c1',
	userId: 'u1',
	roleId: 'r1'
	};
		expect(
		service.assignToUser('company-1', 'user-1', 'role-1'),
		).resolves.toBeUndefined()

Має зберігати SHA-256 хеш API ключу, повертаючи оригінальне значення лише при створенні	{
	companyId: 'company-1',
	dto: {
	name: 'Integration key',
	scopes: ['delivery.orders.read']
	}
	}
		expect(response.key).toMatch(/^fc_/);
		expect(response.key.length)
		.toBeGreaterThan(10);
		
		expect(createCall.data.keyHash)
		.toBe(sha256Hex(response.key));
		expect(createCall.data).not
		.toHaveProperty('key');
		expect(createCall.data.keyHash)
		.not.toBe(response.key);

Результат виконання всіх тест-кейсів для модулю наведено на рисунку 4.7. Тести охоплюють класи «RoleService», «OnboardingService» та «ApiKeyService».

```
PS D:\Diploma\freight-web-services\packages\server>  npx vitest --reporter=verbose run iam

 RUN  v3.2.4 D:/Diploma/freight-web-services/packages/server

 ✓ src/iam/services/role.service.spec.ts (3 tests) 11ms
   ✓ RoleService (3)
     ✓ create() (1)
       ✓ throws ConflictException when a role with the same title already exists for that company 7ms
     ✓ assignToUser() (1)
       ✓ is idempotent — returns silently and does not create a duplicate UserRole when the assignment already exists 1ms
     ✓ findOne() (1)
       ✓ throws NotFoundException for a role that belongs to a different company (tenant isolation) 2ms
 ✓ src/iam/services/onboarding.service.spec.ts (3 tests) 12ms
   ✓ OnboardingService (3)
     ✓ rejects a second onboarding when the user already has a company 7ms
     ✓ composes user.name as "First Middle Last" when middleName is provided 3ms
     ✓ falls back to "First Last" with a single space when middleName is absent 1ms
 ✓ src/iam/services/api-key.service.spec.ts (6 tests) 14ms
   ✓ ApiKeyService (6)
     ✓ validateApiKey() (5)
       ✓ returns null when the key prefix is wrong (no DB lookup performed) 6ms
       ✓ returns null when the key hash is unknown 2ms
       ✓ returns null when the key is deactivated 1ms
       ✓ returns null when the key is expired 1ms
       ✓ returns the key payload for an active, non-expired key 1ms
     ✓ create() (1)
       ✓ stores only the SHA-256 hash (never the raw key) and returns the raw key once for one-time reveal 2ms

 Test Files  3 passed (3)
      Tests  12 passed (12)
   Start at  22:58:54
   Duration  664ms (transform 93ms, setup 0ms, collect 1.08s, tests 37ms, environment 0ms, prepare 370ms)
```

Рисунок 4.7 – Результат виконання тест-кейсів для модулю

Протестовано окремі випадки під час створення ролей, запрошення користувачів, а також під час операцій із API ключами. Тестування відбувається у режимі показу подробиць із залученням файлів тест-кейсів всього модулю, що спрощує створення звітів після кожної зміни пов’язаної логіки.

4.2 Реалізація та тестування модулю складів та вантажів

Модуль складів та вантажів дозволяє керувати записами щодо складів, вантажів, та зв’язками між ними. Його можливості є початковим етапом використання вебплатформи, оскільки кожне замовлення на перевезення потребує вантажу, який необхідно перевезти, а також визначити дії, які потрібно зробити з вантажем в межах платформи після його виконання.

Головним функціоналом є керування вантажами. Для їх створення необхідно вказати: тип, опис, опціональний зв’язок із складом, вага, об’єм, код та клас ADR, габарити, чи швидкопсувний та потребує температурного контролю, тип пакування, кількість одиниць, заявлена вартість та валюта. Форму створення замовлення наведено на рисунку 4.8.

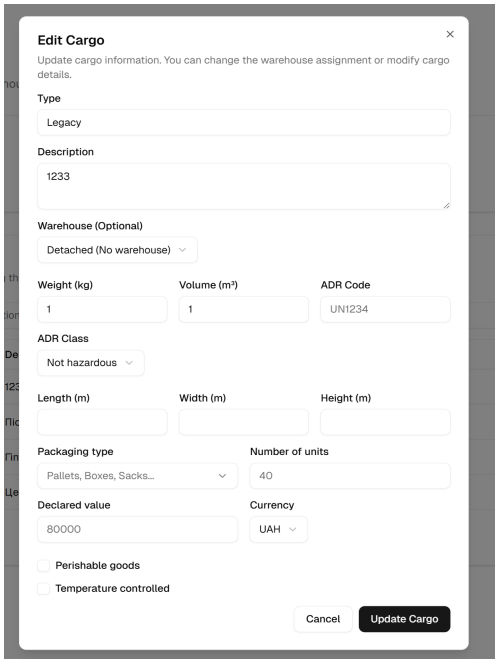

Рисунок 4.8 – Форма створення вантажу

Система має надавати можливість навігації по створених вантажах із пошуком за назвою, фільтрацією за складами, наявністю температурного

контролю чи класу ADR. Також варто додати базову аналітику щодо наявних вантажів: загальна кількість, скільки вантажів знаходяться на складах, скільки доставлено, загальний тоннаж, що важливо для частих перевезень таких сипучих вантажів, як зерно. Інтерфейс сторінки вантажів наведено на рисунку 4.9.

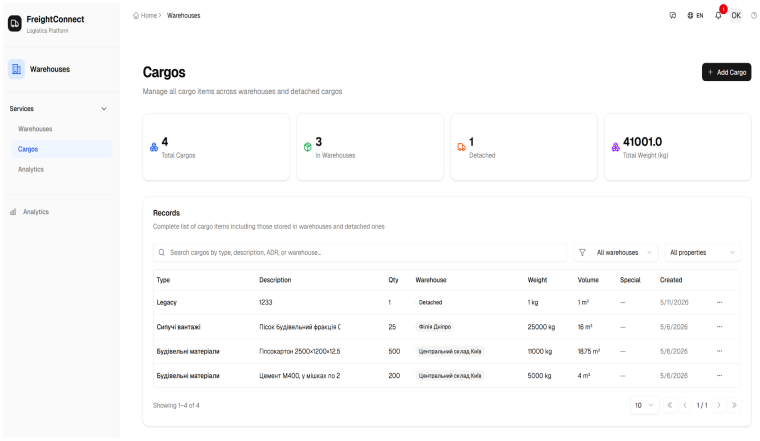

Рисунок 4.9 – Навігація по вантажах

Також вантажі мають можливість додавання пов’язаних документів завдяки окремій формі, що зберігає документи із префіксом компанії та ID замовлення у сховищі Supabase File Storage. Форму завантаження файлів наведено на рисунку 4.10. Надалі вони доступні для перегляду зі сторінки пов’язаного замовлення.

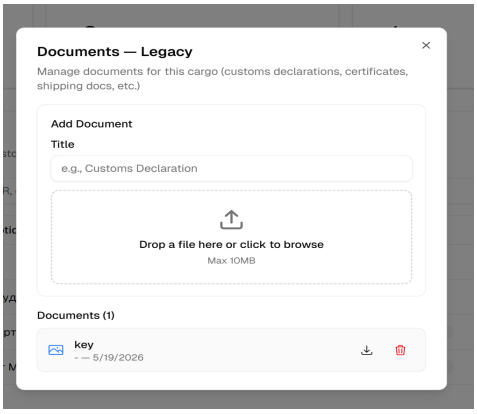

Рисунок 4.10 – Форма перегляду та завантаження файлів вантажу

Інтерфейс перегляду та керування складами є аналогічним до вантажів. Для створення складу необхідно вказати його назву, адресу, та опціональний робочий номер телефону. Інтерфейс сервісу складів разом із формою зміни існуючого запису наведено на рисунку 4.11.

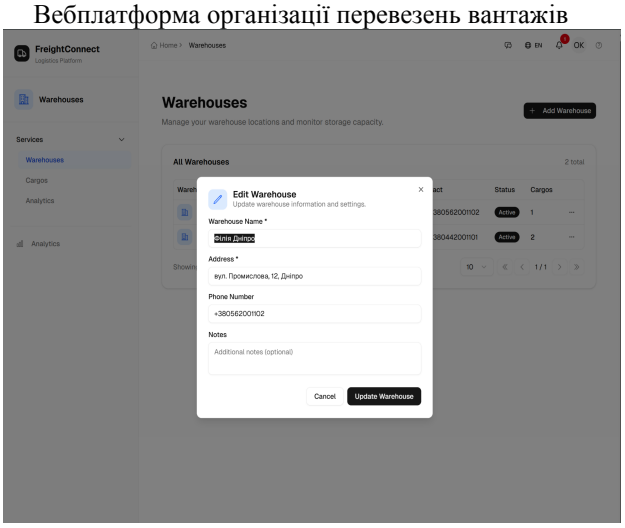

Рисунок 4.11 – Форма зміни відомостей про склад

Тестування для даного модулю охоплюватиме операції створення, оновлення, читання та видалення із складами та вантажами в сервісах «CargoService» та «WarehouseService». Основні тест-кейси модулю наведено в таблиці 4.2.

Таблиця 4.2 – Основні тест-кейси сервісів модулю «Warehouse»

Назва кейсу	Вхідні дані	Очікуваний результат

Має забороняти доступ до складу іншої компанії	{
	companyId = 'company-1'
	warehouseId = 'w-belongs-to-c2'
	}
		throw NotFoundException

Має забороняти видалення складу за наявності прив'язаних вантажів	{
	companyId = 'company-1'
	warehouseId = 'w-1'
	};
	prisma.cargo.count має значення 3.
		throw BadRequestException;
		відсутність виклику prisma.warehouse.delete.

Має створювати сутності «Cargo» та «CargoDetail» у межах одної транзакції	dto={
	type: 'pallet',
	description: 'Test pallet',
	perishable: false,
	tempControl: false,
	weight: 100,
	volume: 1.5,
	};
		prisma.$transaction викликано тільки 1 раз.

Має не повертати вантажі з міткою «deletedAt» в результатах запиту на читання	{
	companyId = 'company-1'
	}
		prisma.cargo.findMany викликано з:
		{
		where: {
		companyId: 'company-1',
		deletedAt: null
		}
		}

Результати виконання тест-кейсів наведено на рисунку 4.12. Протестовано операції таких класів модулю: «WarehouseService», «CargoService».

```
✓ src/warehouse/services/warehouse.service.spec.ts (3 tests) 11ms
  ✓ WarehouseService (3)
    ✓ findOne() (1)
      ✓ throws NotFoundException when warehouse belongs to another company (tenant isolation) 8ms
    ✓ create() (1)
      ✓ defaults active=true and serializes address into location.address JSON 2ms
    ✓ remove() (1)
      ✓ throws BadRequestException when warehouse still has cargos linked 1ms
✓ src/warehouse/services/cargo.service.spec.ts (7 tests) 16ms
  ✓ CargoService (7)
    ✓ create() (4)
      ✓ persists weight and volume as Prisma Decimal (not plain numbers) 7ms
      ✓ includes tempRange when tempControl=true with both min and max 2ms
      ✓ excludes tempRange when tempControl=false even if tempMin/tempMax are provided 1ms
      ✓ creates CargoDetail before Cargo inside a single $transaction call 1ms
    ✓ findAllByCompany() (1)
      ✓ filters out soft-deleted cargos via deletedAt: null 2ms
    ✓ remove() (1)
      ✓ performs soft-delete by setting deletedAt and does NOT call cargo.delete 1ms
    ✓ update() (1)
      ✓ throws NotFoundException for a soft-deleted cargo (deletedAt is not null) 2ms

Test Files  2 passed (2)
     Tests  10 passed (10)
  Start at  23:24:50
  Duration  800ms (transform 76ms, setup 0ms, collect 857ms, tests 27ms, environment 0ms, prepare 257ms)
```

Рисунок 4.12 – Результати тестування сервісів модулю «Warehouse»

У класі«WarehouseService» протестовано окремі випадки під час пошуку, створення та видалення складів. Клас «CargoService» протестовано на коректність створення вантажів за різними вхідними даними, а також протестовано методи фільтрації, видалення та оновлення відповідних записів.

4.3 Реалізація та тестування модулю керування транспортом

Модуль керування транспортом надає можливості створювати, змінювати, видаляти транспорт, записи про його доступність, що буде використовуватися для виставлення публічної доступності транспорту на платформі, а також створювати записи про призначення водіїв на певний транспорт і додавання пов’язаних із транспортом файлів для подальшого перегляду. Головна сторінка модулю має надавати можливість перегляду зведення щодо автопарку компанії: загальна кількість транспорту, скільки одиниць мають призначеного водія, скільки одиниць мають вказану доступність, який транспорт потребує налаштувань водія та доступності, а також загальна місткість автопарку в тоннах, об’ємі, підтримці температурного контролю та класів ADR. Інтерфейс головної сторінки модулю наведено на рисунку 4.13.

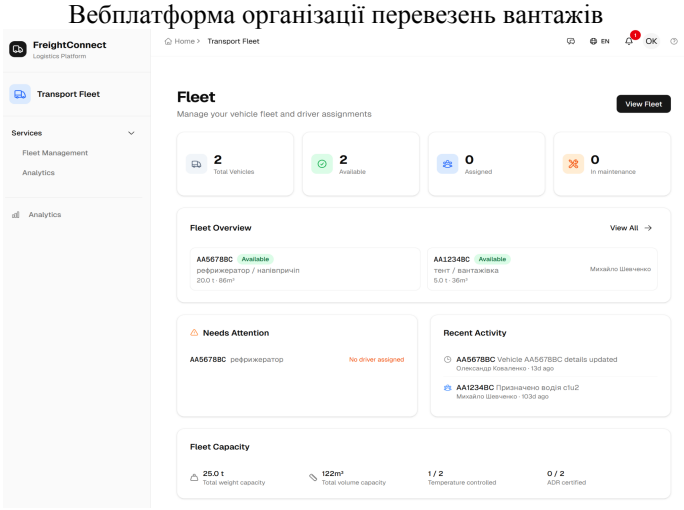

Рисунок 4.13 – Головна сторінка модулю керування транспортом

Для створення запису про транспортний засіб необхідно вказати: номерний знак, тип транспорту, тип причепу, поточна локація. Також вказуються відомості про підтримуваний вантаж: габарити, вага, клас ADR, підтримка швидкопсувного вантажу та температурного контролю, можливості завантаження та розвантаження. Інтерфейс форми наведено на рисунку 4.14.

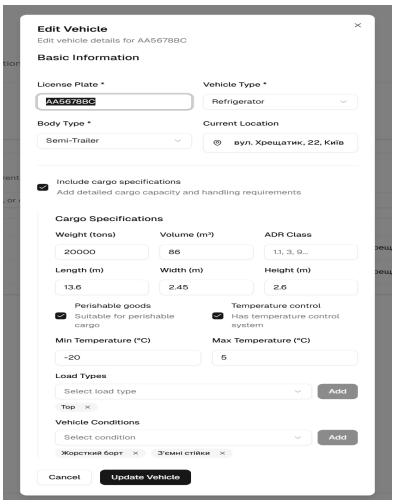

Рисунок 4.14 – Форма створення та редагування відомостей про транспорт

Для вказання локації транспорту та початкових і кінцевих локацій при створенні та редагуванні замовлення створено загальний компонент, що дозволятиме вказувати локацію як із вказанням адреси у вигляді тексту із можливістю автозаповнення існуючими адресами (пряме геокодування), за якими треба отримати координати, так і за обранням точки на карті, з якої буде отримано текстову адресу (зворотне геокодування). Можливості геокодування

забезпечуватиме сервіс API Geoapify, а для рендерингу інтерактивного інтерфейсу карти використано бібліотеку Leaflet. Інтерфейс вказання локації для транспорту наведено на рисунку 4.15.

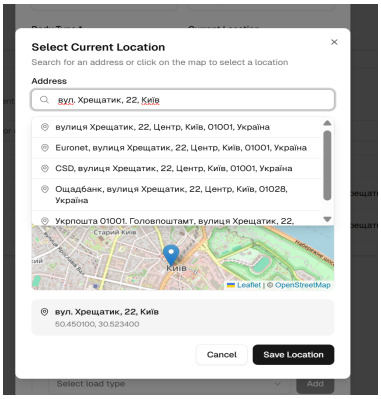

Рисунок 4.15 – Інтерфейс обрання адреси для транспорту

Для створення запису про доступність транспорту використовуються такі дані: дати початку та завершення, локації відповідальності даного транспорту, доступний тоннаж і габарити, тариф за тонну та за кілометр, можливість часткового завантаження. Для призначення водія до транспорту використовується окрема форма, де обирається користувач компанії, а також опціональна дата завершення зміни.

Тестування даного модулю полягає в перевірці логіки створення доступності транспорту, а також взаємодії пов’язаних сутностей з іншими модулями. Перелік тест-кейсів наведено в таблиці 4.3.

Таблиця 4.3 – Основні тест-кейси сервісів модулю «Warehouse»

Назва кейсу	Вхідні дані	Очікуваний результат

Має забороняти видалення транспорту з активним замовленням	{
	companyId: 'company-1', vehicleId: 'vehicle-1'
	}
		throw BadRequestException

Має помічати транспортний засіб як видалений із скасуванням пропозицій, доступності та призначення водія	{
	companyId: 'company-1', vehicleId: 'vehicle-1'
	}
		Виклик prisma.$transaction, bidService.invalidatePendingBidsForVehicle, vehicleAssignment.updateMany(status=inactive), vehicleActivity.create.

Має забороняти створювати доступність без призначеного водія	prisma.vehicle.findFirst повертає: { id: 'vehicle-1', type: 'truck', assignments: [], deletedAt: null }	throw BadRequestException
Має забороняти повторне призначення водія	prisma.vehicleAssignment.findUnique повертає: { vehicleId, driverUserId, status: ‘active’ }	throw ConflictException

4.4 Реалізація та тестування модулю замовлень на перевезення

Модуль замовлень є головним для реалізації взаємодії між власниками вантажу та перевізниками. Він дозволяє використовувати раніше створені вантажі, а також транспорт з метою створення замовлень на перевезення вантажів, створення пропозицій по замовленню як зі сторони перевізників, так і для власників вантажів завдяки механізму прямої пропозиції та пошуку транспорту.

Головна сторінка модулю надає зведення по замовленням, кількості активних та виконаних замовлень, нещодавніх замовлень та пропозицій. Сторінка «Мої замовлення» показує всі замовлення, створені компанією, з їх статусами, можливістю фільтрації за текстовими полями і статусом, а також переходу на сторінку окремого замовлення. Інтерфейс сторінки замовлень наведено на рисунку 4.16.

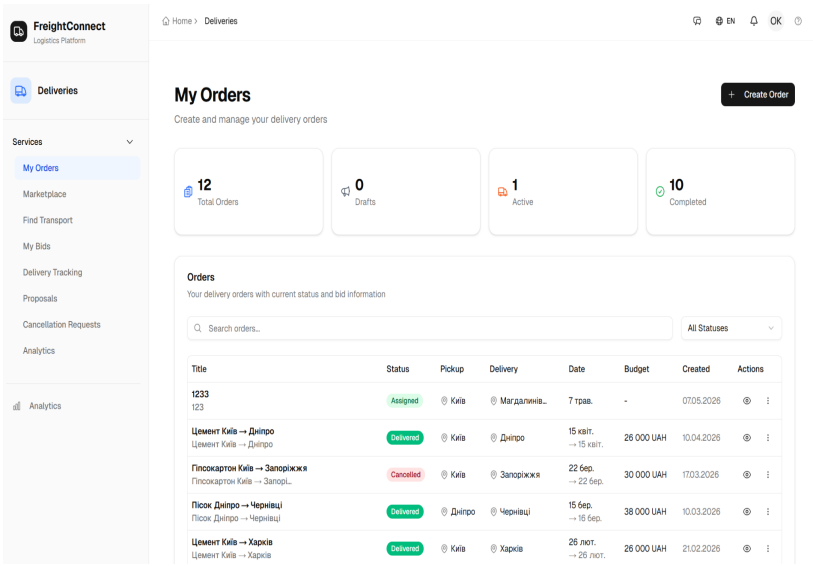

Рисунок 4.16 – Інтерфейс сторінки «Мої замовлення»

Для створення замовлення необхідно ввести такі дані: назва, опис, пов’язаний вантаж, локація початку та завершення транспортування, опціональний бюджет, деталі оплати, а також що робити із записом про вантаж після виконання замовлення. Після створення, замовлення має статус «чернетка» і не є доступним для інших компаній. Інтерфейс сторінки подробиць замовлення наведено на рисунку 4.17.

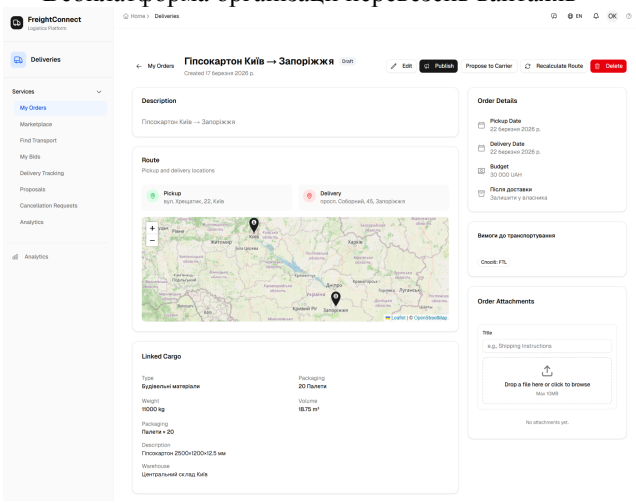

Рисунок 4.17 – Інтерфейс сторінки замовлення

Замовлення можливо оновити, опублікувати, видалити, а також зробити повторний розрахунок маршруту. Після публікації воно стає доступним для пошуку на сторінці «Маркетплейс», де можливо фільтрувати замовлення за текстовими полями, локацією, бюджетом, тоннажем та габаритами. Фільтр локацій працює за методом області:

– користувач вводить адресу локації та обирає варіант із результатів пошуку (вулиця, місто, область, регіон, країна);

– з клієнтської частини відбувається виклик сервісу Geoapify для отримання координат меж обраної зони (мінімальна та максимальна широта та довгота) і надсилає координати до серверу;

– сервер створює запит до БД, вказуючи дані межі координат локацій замовлень.

Після того, як перевізник знайшов замовлення та переглянув його деталі, він може надіслати свою пропозицію на виконання. Далі власник вантажу отримує сповіщення в режимі реального часу та може переглянути пропозиції на сторінці відповідного замовлення. Інтерфейс перегляду пропозиції містить дані про транспорт, водія, умови та суму оплати з можливістю прийняти чи відхилити пропозицію. Його зображено на рисунку 4.18.

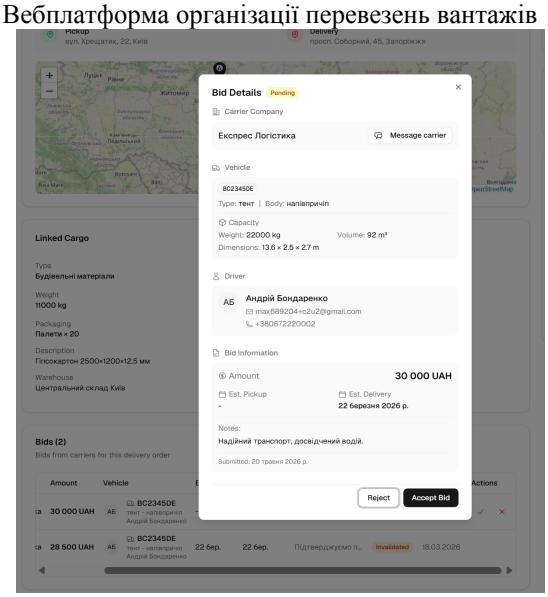

Рисунок 4.18 – Інтерфейс перегляду пропозиції на виконання перевезення

Після його прийняття для обох сторін стає доступним інтерфейс відстеження процесу виконання замовлення. Він складається із наочного представлення всіх статусів, через які має пройти замовлення, запланований маршрут, фактичний маршрут, який занесено до платформи водієм,. Сторінку зображено на рисунку 4.19.

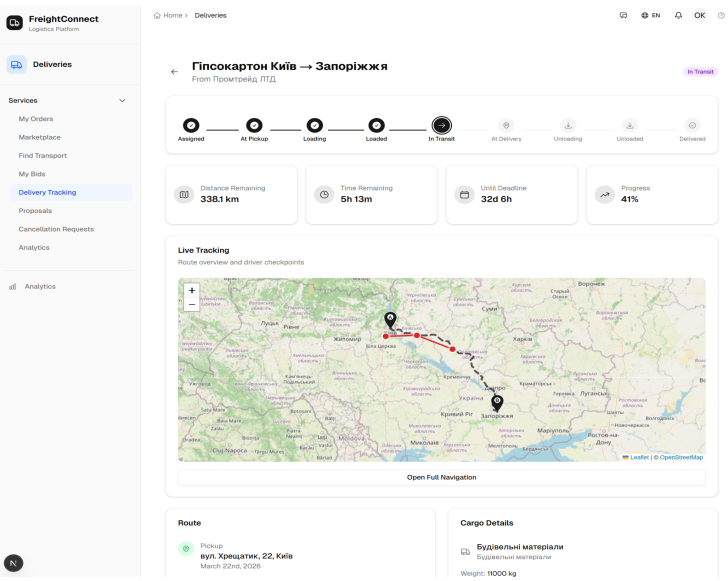

Рисунок 4.19 – Сторінка відстеження виконання замовлення

Також надається окрема сторінка, призначена для водія. Вона має бути в першу чергу зручною для використання з мобільних пристроїв і містить карту з маршрутом, приблизною дистанцією та тривалістю залишку маршруту, а також кнопки оновлення: статусу, поточної локації водія, запланованого маршруту.

Інші користувачі обох компаній мають доступ до нього в адміністративних цілях. Її інтерфейс наведено на рисунку 4.20.

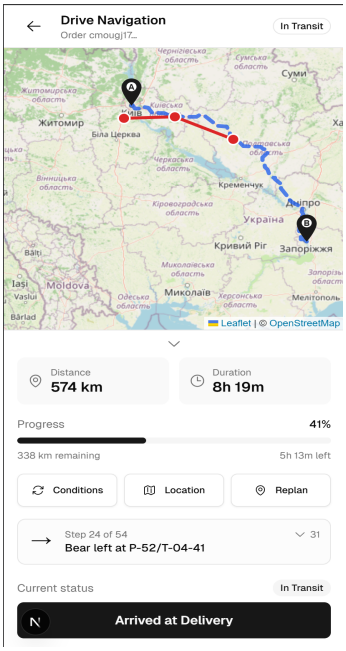

Рисунок 4.20 – Сторінка виконання замовлення в мобільному представленні

У випадку зміни обставин щодо виконання активного замовлення, власник вантажу може запропонувати зміну умов замовлення, на яке має погодитися інша сторона. Інтерфейс запиту на зміну наведено на рисунку 4.21.

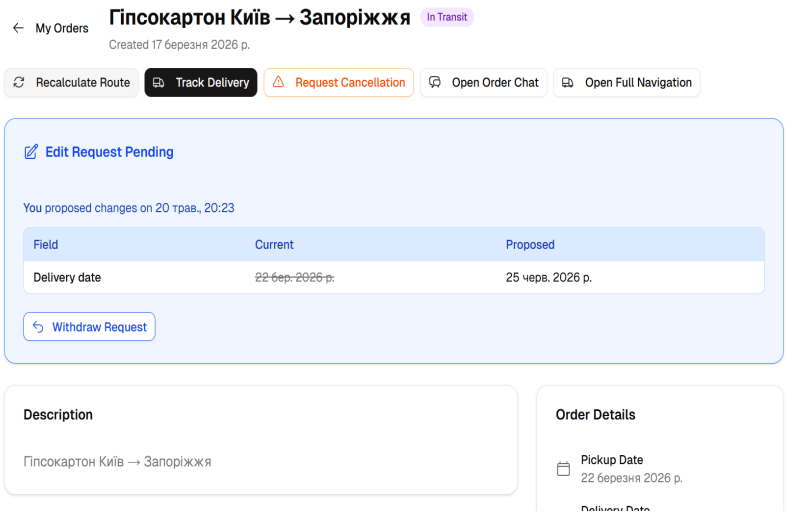

Рисунок 4.21 – Надісланий запит на зміну замовлення (очікувана дата доставки)

Також обидві сторони можуть створити запит на скасування замовлення із вказанням причини. Такий запит може бути відхилено чи прийнято, і його інтерфейс зображено на рисунку 4.22.

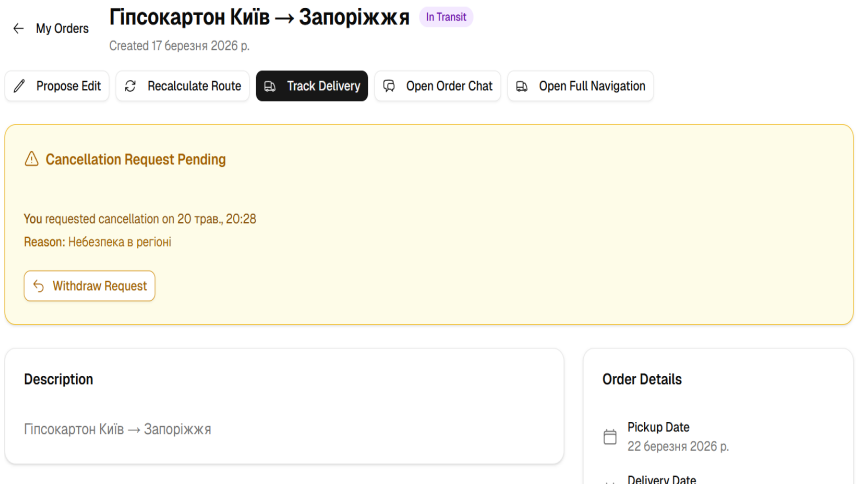

Рисунок 4.22 – Надісланий запит на скасування замовлення

Окрім цього, можливо надсилати шукати перевізників, переглядати їх доступний транспорт, та надсилати прямі пропозиції при наявності опублікованого замовлення, яке ще не перейшло в статус виконання. Інтерфейс перегляду транспорту та надання прямої пропозиції наведено на рисунку 4.23.

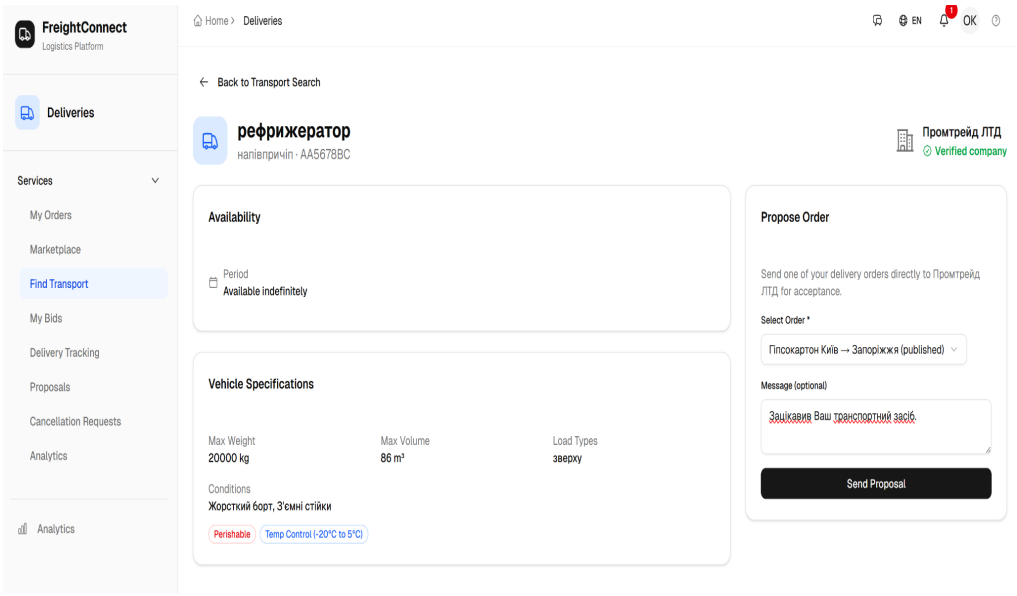

Рисунок 4.23 – Інтерфейс перегляду доступного транспорту для пропозицій

Тестування даного модулю є обширним та включає в себе перевірку таких сервісів: «DeliveryOrderService», «BidService», «DriverDeliveryService», «CancellationRequestService», «OrderEditRequestService», «ProposalService». Результат проходження 31 тесту в 7 тест-файлах наведено на рисунку 4.24.

Рисунок 4.24 – Результати тестування модулю «Delivery»

Протестовано операції керування сутностями замовлення, пропозицій та функціонал виконання замовлення зі сторони водія. Також протестовано окремі випадки при створенні запитів на зміну та скасування активного замовлення разом із сервісом аналітики для отримання зведених даних по перевезенням.

Висновки до розділу 4

У даному розділі описано результати кодування головних модулів вебплатформи, зокрема модулі аутентифікації та авторизації, керування вантажами та складами, транспортом і призначеннями водіїв, замовленнями та пропозиціями на перевезення. Описано функціональну частину кожного модулю, інтерфейс та технологічні рішення, що були використані під час їх реалізації. Наведено таблиці із основними сценаріями тестування, вказавши їх назви, вхідні дані та очікувані результати, а також наведено результати проведених тестувань для кожного модулю.

ВИСНОВКИ

У ході виконання кваліфікаційної бакалаврської роботи покращено степінь автоматизації в процесах організації вантажоперевезень завдяки розробці вебплатформи для організації перевезень вантажів. Для реалізації заданої мети виконано такі завдання:

- проаналізовано предметну область, зокрема існуючі рішення;
- спроєктовано архітектуру застосунку;
- розроблено структуру бази даних;
- реалізовано модулі бізнес-логіки та інтерфейсу для сутностей;
- розроблено адаптацію застосунку для його використання водіями під час виконання перевезень;
- проведено тестування реалізованих модулів системи.

Описано предметну область, визначено основні проблеми галузі перевезень вантажів в Україні та ЄС, зокрема значні витрати часу на пошук та перевірку контрагентів. Проаналізовано існуючі рішення, що не гарантують знаходження необхідних контрагентів, із відсутністю аналітичної складової, чи можливості інтеграції з існуючими системами, або з обмеженою видимістю процесу для всіх сторін. На основі отриманих даних визначено основний функціонал майбутньої платформи. Проведено огляд існуючих технологічних рішень, визначено стек клієнтської та серверної частин вебплатформи. Сформульовано специфікацію вимог, що включає як опис призначення і меж, сфери застосування проєкту, так і загальну структуру і обмеження системи. Описано сценарії використання, структуру бази даних, стани та послідовності виконання основних процесів у системі, змодельовано інтерфейс застосунку.

Результатом програмної реалізації є вебплатформа організації перевезень вантажів, яка дозволяє ефективно знаходити вантажі та транспорт із можливістю безперебійної комунікації сторін на кожному етапі пошуку та виконання замовлень, дозволяючи адаптуватися до вимог логістики у військовий час із дотриманням стандартів щодо безпеки зберігання та передачі комерційних даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Державна служба статистики України. Транспорт України: статистичний збірник. За ред. І. Петренко. Київ, 2024. 94 с. URL: https://www.ukrstat.gov.ua/druk/publicat/kat_u/2024/zb/10/zb_Trans_23.pdf (дата звернення: 28.04.2026).

2. World Bank. Logistics performance index (LPI) 2023: International LPI. 73 p. URL: https://lpi.worldbank.org/sites/default/files/2023-04/LPI_2023_report_with_layout.pdf (accessed: 28.04.2026).

3. United Nations Economic Comission for Europe (UNECE). ADR 2025: Agreement concerning the International Carriage of Dangerous Goods by Road : files. 2025. URL: https://unece.org/adr-2025-files (accessed: 28.04.2026).

4. Про єдині вимоги до конструкції та технічного стану колісних транспортних засобів, що експлуатуються : Постанова Каб. Міністрів України від 22.12.2010 № 1166 : станом на 14 січ. 2021 р. URL: https://zakon.rada.gov.ua/laws/show/1166-2010-п#Text (дата звернення: 28.04.2026).

5. GDPR українською. Загальний регламент про захист даних (GDPR) українською. URL: https://www.gdpr.org.ua/ (дата звернення: 28.04.2026).

6. Truckscanner. The digital revolution of freight transport. *Truckscanner Blog*. 2026. URL: https://truckscanner.it/en/blog/digital-freight-marketplace-revolution (accessed: 28.04.2026).

7. DELLA Вантажні перевезення. DELLA Вантажні перевезення. URL: https://della.ua/ (дата звернення: 27.05.2026).

8. Автоматизація транспортної логістики. Ant Logistics. URL: https://ant-logistics.com/ (дата звернення: 27.05.2026).

9. Roolz. Roolz. URL: https://app.roolz.net/ (accessed: 27.05.2026).

10. What Rationales Drive Architectural Decisions? An Empirical Inquiry / K. Borowa et al. *Software Architecture*. Cham, 2023. P. 303–318. URL: https://doi.org/10.1007/978-3-031-42592-9_21 (accessed: 28.04.2026).

11. Bogner J., Merkel M. To type or not to type?. *MSR '22: 19th international conference on mining software repositories*, Pittsburgh Pennsylvania. New York, NY, USA, 2022. PP. 658-669. URL: https://doi.org/10.1145/3524842.3528454 (accessed: 28.04.2026).

12. Documentation – Everyday Types. *TypeScript: JavaScript With Syntax For Types.* URL: https://www.typescriptlang.org/docs/handbook/2/everyday-types.html#interfaces (accessed: 28.04.2026).

13. Core Components and Native Components · React Native. *React Native · Learn once, write anywhere*. URL: https://reactnative.dev/docs/intro-react-native-components (accessed: 29.04.2026).

14. JSX In Depth – React. *React – A JavaScript library for building user interfaces*. URL: https://legacy.reactjs.org/docs/jsx-in-depth.html (accessed: 28.04.2026).

15. Reconciliation – React. *React – A JavaScript library for building user interfaces*. URL: https://legacy.reactjs.org/docs/reconciliation.html (accessed: 28.04.2026).

16. Ollila R., Mäkitalo N., Mikkonen T. Modern Web Frameworks: A Comparison of Rendering Performance. *Journal of Web Engineering*. 2022. Vol. 21, no. 03, pp. 789–814. URL: https://doi.org/10.13052/jwe1540-9589.21311 (accessed: 28.04.2026).

17. Swacha J., Kulpa A. Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks. *Electronics*. 2023. Vol. 12, no. 17. P. 3563. URL: https://doi.org/10.3390/electronics12173563 (accessed: 28.04.2026).

18. Vercel. API Reference: File-system conventions | Next.js. *Next.js by Vercel – The React Framework*. URL: https://nextjs.org/docs/app/api-reference/file-conventions (accessed: 28.04.2026).

19. Caching Examples | TanStack Query React Docs. *TanStack | The open-source application stack for the*

web. URL: https://tanstack.com/query/v4/docs/framework/react/guides/caching (acce ssed: 28.04.2026).

20. Introduction. *The Foundation for your Design System – shadcn/ui*. URL: https://ui.shadcn.com/docs (accessed: 28.04.2026).

21. Prasad M. P., Padma U. Performance Analysis of ExpressJS and Fastify in NestJS. *Intelligent Control, Robotics, and Industrial Automation*. Singapore, 2023. P. 1037–1049. URL: https://doi.org/10.1007/978-981-99-4634-1_82 (accessed: 28.04.2026).

22. Running TypeScript Natively | Node.js Learn. *Node.js – Run JavaScript Everywhere*. URL: https://nodejs.org/learn/typescript/run-natively (accessed: 28.04.2026).

23. Documentation | NestJS – A progressive Node.js framework. *Documentation | NestJS – A progressive Node.js framework*. URL: https://docs.nestjs.com/custom-decorators#param-decorators (accessed: 28.04.2026).

24. SQL and NoSQL Database Software Architecture Performance Analysis and Assessments–A Systematic Literature Review / W. Khan et al. *Big Data and Cognitive Computing*. 2023. Vol. 7, no. 2. P. 97. URL: https://doi.org/10.3390/bdcc7020097 (accessed: 28.04.2026).

25. How it works | Socket.IO. *Socket.IO*. URL: https://socket.io/docs/v3/how-it-works/ (accessed: 28.04.2026).

26. Prisma schema. *Prisma | Serverless Postgres, Type-Safe ORM, and Database Tools*. URL: https://www.prisma.io/docs/orm/prisma-schema/overview (accessed: 17.05.2026).

27. Vitest. *Vitest | Next Generation testing framework*. URL: https://vitest.dev/guide/learn/writing-tests.html (accessed: 19.05.2026).

ДОДАТОК А

Діаграма сценаріїв використання вебплатформи