

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Вебплатформа автоматизації логістичних процесів
вантажних перевезень»**

Виконав студент групи ІПЗ-22-3
КРАСНОБОКИЙ Михайло Вадимович

Керівник роботи:
ЛЕВКІВСЬКИЙ Віталій Леонідович

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КРАСНОБОКИЙ Михайло Вадимович**

Керівник роботи: **ЛЕВКІВСЬКИЙ Віталій Леонідович**

Тема роботи: **«Вебплатформа автоматизації логістичних процесів вантажних перевезень»**,

затверджена наказом закладу вищої освіти №92/с від «27» лютого 2026р.

Вихідні дані для роботи: **об'єктом дослідження** є процеси організації, планування та автоматизації вантажних перевезень у логістичній галузі. **Предметом дослідження** є програмна архітектура, серверна та клієнтська логіка вебплатформи для управління транспортною логістикою, методи автоматизації документообігу та моніторингу вантажів, а також механізми взаємодії користувачів (адміністратор, диспетчер, водій).
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Левківський В. Л.	27.02.2026	27.02.2026
2	Левківський В. Л.	27.03.2026	27.03.2026
3	Левківський В. Л.	27.04.2026	27.04.2026

РЕФЕРАТ

Кваліфікаційна робота бакалавра складається з програмного комплексу та пояснювальної записки. Пояснювальна записка до роботи містить 72 сторінки, включає 31 ілюстрацію, 20 таблиць, 1 додаток та 35 використаних літературних джерел.

Метою роботи є розробка вебплатформи автоматизації логістичних процесів вантажних перевезень, що дозволяє оптимізувати облік замовлень, керування автопарком, маршрутизацію та відстеження водіїв, а також автоматизувати створення супровідної документації.

У роботі проведено аналіз предметної області, сформульовано вимоги до функціоналу системи, розроблено архітектуру з урахуванням клієнт-серверної моделі. Для реалізації серверної частини використано технології Node.js, ORM Sequelize та систему управління базами даних MySQL. Клієнтська частина реалізована за допомогою бібліотеки React.js із використанням Bootstrap 5 для стилізації та інтеграцією відкритих картографічних сервісів для моніторингу геолокації в реальному часі.

Передбачено підтримку мультикомпанійної архітектури та кількох ролей користувачів. Платформа забезпечує управління транспортними засобами, генерацію товарно-транспортних накладних у форматі PDF, надсилення сповіщень електронною поштою та візуалізацію статистичних даних на аналітичні панелі.

Система має гнучку модульну структуру: управління замовленнями, моніторинг автопарку, формування документації та аналітики продуктивності, що дозволяє легко масштабувати продукт та швидко інтегрувати нові функції в майбутньому.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ, ВЕБПЛАТФОРМА, ЛОГІСТИКА, СИСТЕМА МОНІТОРИНГУ, УПРАВЛІННЯ ВАНТАЖОПЕРЕВЕЗЕННЯМИ.

					ІПЗ.КР.Б – 121 – 26 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа автоматизації логістичних процесів вантажних перевезень Пояснювальна записка	Лім.	Арк.	Аркушів	
Розроб.		Краснобокий М.В							
Керівник		Левківський В.Л.					3	138	
Рецензент		Чижмоторя О.Г.				Житомирська політехніка, група ІПЗ-22-3			
Зав. каф.		Вакалюк Т. А.							

ABSTRACT

The Bachelor's qualification project consists of a software suite and an explanatory note. The explanatory note to the qualification project contains 72 pages, includes 31 illustrations, 20 tables, 1 appendix, and 35 references.

The aim of the work is to develop a web platform for automating the logistics processes of freight transportation, which allows optimizing order registration, fleet management, routing, and driver tracking, as well as automating the creation of shipping documentation.

The project conducts an analysis of the subject area, formulates requirements for the system's functionality, and develops an architecture based on a client-server model. The server side is implemented using Node.js, Sequelize ORM, and the MySQL database management system. The client side is implemented using the React.js library with Bootstrap 5 for styling and the integration of open mapping services for real-time geolocation monitoring.

It provides support for a multi-company architecture and several user roles. The platform ensures vehicle management, generation of consignment notes in PDF format, sending email notifications, and statistical data visualization on an analytical dashboard.

The system has a flexible modular structure: order management, fleet monitoring, document generation, and performance analytics, which allows for easy product scaling and quick integration of new features in the future.

KEYWORDS: AUTOMATION, FREIGHT TRANSPORTATION MANAGEMENT, LOGISTICS, TRACKING SYSTEM, WEB PLATFORM.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ТА ПОСТАНОВКА ЗАВДАННЯ.....	10
1.1. Постановка завдання у предметній області та основні завдання автоматизації.....	10
1.2. Аналіз існуючих аналогів програмного продукту	12
1.3. Вибір архітектури програмного забезпечення.....	17
1.4. Обґрунтування вибору інструментальних засобів.	21
Висновки до першого розділу	25
РОЗДІЛ 2. ПРОЄКТУВАННЯ ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ВАНТАЖОПЕРЕВЕЗЕННЯМИ	27
2.1. Визначення варіантів використання та структури системи	27
2.2. Проєктування реляційної бази даних	32
2.3. Проєктування та розробка алгоритмів роботи вебплатформи.....	39
2.4. Реалізація функціоналу вебплатформи.	43
Висновки до другого розділу.....	48
РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБПЛАТФОРМИ АВТОМАТИЗАЦІЇ ЛОГІСТИЧНИХ ПРОЦЕСІВ	49
3.1. Порядок налаштування параметрів та розгортання системи	49
3.2. Алгоритм використання інтерфейсу користувача.....	52
3.3. Тестування роботи платформи автоматизації логістичних процесів.....	63
Висновки до третього розділу	66
ВИСНОВКИ	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	70
ДОДАТКИ.....	73

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface / Програмний інтерфейс прикладного програмування

CI/CD - Continuous Integration / Continuous Delivery / Безперервна інтеграція та доставка

CRUD - Create, Read, Update, Delete / Основні операції з даними

ERP - Enterprise Resource Planning / Система планування ресурсів підприємства

JWT - JSON Web Token / Веб-токен у форматі JSON

ORM - Object-Relational Mapping / Об'єктно-реляційне відображення

RBAC - Role-Based Access Control / Контроль доступу на основі ролей

REST - Representational State Transfer / Технологія архітектури обміну даними

SaaS - Software as a Service / Програмне забезпечення як послуга

SPA - Single Page Application / Односторінковий вебдодаток

SQL - Structured Query Language / Мова структурованих запитів

TMS - Transport Management System / Система управління транспортними перевезеннями

UX - User Experience / Користувацький досвід

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Сучасний ринок вантажних перевезень в Україні та світі переживає етап активної цифрової трансформації. Ефективність функціонування транспортних і логістичних компаній сьогодні безпосередньо залежить від швидкості обробки великих масивів інформації, точності координації між усіма учасниками ланцюга постачання та можливості оперативного реагування на позаштатні ситуації [1]. Традиційні підходи до управління автопарком і маршрутами, що спираються на ручний контроль, телефонну комунікацію диспетчерів з водіями та паперовий документообіг, вже не відповідають вимогам сучасного динамічного бізнесу. Вони призводять до значної кількості людських помилок, неефективного використання ресурсів, затримок у рейсах та загального зниження рентабельності підприємств [2].

Особливої актуальності набуває проблема автоматизації для підприємств малого та середнього бізнесу. На відміну від великих корпорацій, що можуть дозволити собі впровадження високовартісних систем класу ERP чи громіздких TMS (Transport Management System), менші компанії потребують гнучких, легких та доступних вебрішень. Впровадження спеціалізованих хмарних вебплатформ дозволяє централізувати управління транспортними засобами, автоматизувати рутинні процеси формування супровідної документації (зокрема товарно-транспортних накладних і рахунків) та забезпечити прозорий моніторинг виконання замовлень у реальному часі. Крім того, перехід до електронних систем обліку сприяє підвищенню рівня контролю за фінансовими витратами (паливо, амортизація) та спрощує взаємодію з клієнтами. З огляду на вищезазначене, розробка сучасної, масштабованої вебплатформи для автоматизації логістичних процесів вантажних перевезень є вкрай актуальним завданням, яке має високу практичну цінність та повністю відповідає сучасним тенденціям цифровізації української економіки [3].

Мета роботи - розробка алгоритмів, проектування та практична реалізація програмного комплексу вебплатформи для автоматизації логістичних процесів вантажних перевезень, яка забезпечить оптимізацію транспортного

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

документообігу, моніторинг статусів рейсів та ефективну рольову взаємодію між персоналом компанії.

Завдання роботи. Встановлена мета обумовлює виконання наступних науково-практичних завдань:

- провести системний аналіз предметної області логістичного управління та здійснити порівняльний огляд існуючих комерційних рішень;
- сформулювати технічні вимоги до функціональності, безпеки та архітектури майбутньої системи;
- обґрунтувати вибір сучасного стеку технологій та інструментальних засобів для реалізації серверної та клієнтської частин;
- розробити інформаційне забезпечення, спроектувати структуру реляційної бази даних з підтримкою мультитенантності;
- розробити алгоритми роботи основних модулів системи (маршрутизація, генерація документів, авторизація);
- спроектувати та реалізувати користувацький інтерфейс для різних ролей (суперадміністратор, адміністратор, диспетчер, водій);
- здійснити практичну програмну реалізацію платформи та провести її комплексне тестування.

Об’єкт дослідження - процеси автоматизації управління, планування та моніторингу вантажних перевезень у транспортних компаніях.

Предмет дослідження - архітектура, алгоритми, методи об’єктно-реляційного відображення та сучасні вебтехнології для розробки інформаційної платформи управління логістичними процесами.

Методи дослідження. Під час виконання кваліфікаційної роботи було використано комплекс методів наукового пізнання: методи системного та порівняльного аналізу - для дослідження предметної області й існуючих аналогів; методи об’єктно-орієнтованого та структурного проектування (зокрема UML-модельювання) - для побудови архітектури програмного продукту; методи реляційної алгебри - для проектування та нормалізації бази даних; методи вебпрограмування та UI/UX дизайну - для реалізації клієнт-серверної взаємодії;

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

методи емпіричного тестування - для перевірки працездатності та надійності створеного програмного забезпечення.

Практичне значення отриманих результатів полягає у створенні повністю функціонального програмного продукту, який може бути безпосередньо впроваджений у господарську діяльність транспортно-експедиційних підприємств. Використання платформи дозволяє скоротити час на обробку замовлень, зменшити ризики людського фактору при генерації документів та підвищити загальну прозорість логістичних операцій завдяки геолокаційному моніторингу рейсів. Архітектура застосунку дозволяє його подальше комерційне масштабування за моделлю SaaS.

Структура та обсяг роботи. Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 35 найменувань (3 стор.), 1 додаток (10 стор.), 20 таблиць та 31 ілюстрація по тексту. Повний обсяг роботи становить 82 сторінки, з них 72 сторінки основного тексту.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Постановка завдання у предметній області та основні завдання автоматизації

Сфера вантажних перевезень є однією з найбільш динамічних галузей сучасної економіки. Оперативність доставки, збереження вантажу та чіткість документообігу безпосередньо впливають на конкурентоспроможність транспортної компанії. Щодня логістичній установі доводиться контролювати велику кількість змінних: від технічного стану вантажівок та розкладу водіїв, до зміни погодних умов, пробок на маршрутах і своєчасної готовності товарів на складах вантажовідправника.

На практиці багато перевізників малого та середнього бізнесу використовують підхід, який можна назвати «фрагментованою автоматизацією». Це означає, що для різних бізнес-процесів застосовуються незв'язані між собою інструменти. Наприклад, для обліку замовлень і розрахунків рентабельності найчастіше використовується Microsoft Excel або Google Sheets, тоді як для оперативного зв'язку та інструктажу водіїв диспетчери застосовують мобільні месенджери, такі як Viber чи Telegram. Водночас для створення первинної бухгалтерської документації, зокрема товарно-транспортних накладних (ТТН), рахунків-фактур та актів виконаних робіт, використовуються заготовлені шаблони в Microsoft Word, куди всі дані копіюються вручну [29].

Така розрізненість створює цілий ряд проблем. По-перше, виникає потреба в постійному багаторазовому ручному введенні одних і тих же даних (наприклад, даних про машину, водія та компанію-замовника). Це не лише займає багато часу у операторів-диспетчерів, а й призводить до неминучих людських помилок (опечаток у номерах документів, реєстраційних знаках або реквізитах). По-друге, повністю відсутня загальна прозорість процесу: керівник або менеджер компанії не може в один клік побачити агреговану статистику доходів, витрат або проаналізувати продуктивність кожного конкретного водія чи транспортного засобу за місяць. По-третє, контроль виконання замовлення покладається

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

виключно на телефонні дзвінки, що робить неможливим отримання даних про об'єктивні координати вантажівки в режимі реального часу без сторонніх GPS-трекерів із їх власним окремим програмним забезпеченням.

Повноцінна автоматизація процесів управління логістикою передбачає концептуальне об'єднання облікових, операційних та документальних функцій у межах єдиної програмної екосистеми. Серед головних завдань такої автоматизації насамперед потрібно виділити створення спільної бази даних. Там мають зберігатися профілі клієнтів, список працівників, а також інформація про всі автомобілі, включаючи їхню вантажопідйомність та технічний стан. Крім того, програма повинна підтримувати повний цикл роботи із замовленнями. Тобто кожна заявка має послідовно проходити всі етапи від її створення до остаточного виконання та оплати. Це допоможе уникнути плутанини і чітко закріпити відповідальних працівників за кожним кроком. Важливо також автоматизувати роботу з документами. Програма повинна самостійно брати інформацію із замовлення і формувати готові до друку товарно-транспортні накладні або рахунки у форматі PDF [9].

Оскільки з платформою працюватимуть різні співробітники, обов'язково потрібно налаштувати розмежування доступу за ролями. Наприклад, керівництву необхідний повний контроль над налаштуваннями компанії, тоді як менеджери будуть займатися клієнтами, а диспетчери керуватимуть маршрутами. Для водіїв достатньо розробити мобільний доступ, щоб вони могли просто переглядати свої рейси та відмічати їхнє виконання. Для додаткової зручності диспетчерів варто додати інтеграцію з картами для відстеження поточного місцезнаходження вантажівок. Враховуючи, що платформою зможуть користуватися кілька різних компаній, її архітектура одразу має бути розрахована на багатьох орендарів. Завдяки цьому кожна компанія працюватиме у власному закритому середовищі і матиме доступ виключно до свого автопарку та своїх замовлень.

Вирішення описаних завдань дозволить логістичній компанії суттєво зменшити час на рутинні операції, знизить вплив людського фактору, оптимізує

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

навантаження на автопарк і дозволить керівництву приймати рішення на основі об'єктивних аналітичних показників у режимі реального часу [10].

1.2. Аналіз існуючих аналогів програмного продукту

Для формування повного, об'єктивного та конкурентоспроможного переліку вимог до власної розроблюваної системи було проведено детальне дослідження та аналіз існуючих на сучасному ринку рішень класу TMS (Transport Management System). Ринок логістичного програмного забезпечення пропонує досить широкий вибір: від гігантських корпоративних ERP-модулів глобальних розробників до локальних українських стартапів, які спеціалізуються на окремих вузьких процесах (наприклад, лише на електронному документообігу). Проведення такого аналізу дозволяє виявити сильні сторони, запозичити кращі архітектурні ідеї, а головне - виділити слабкі місця конкурентів, щоб перетворити їх на переваги власної вебплатформи.

SAP Transportation Management (SAP TM)

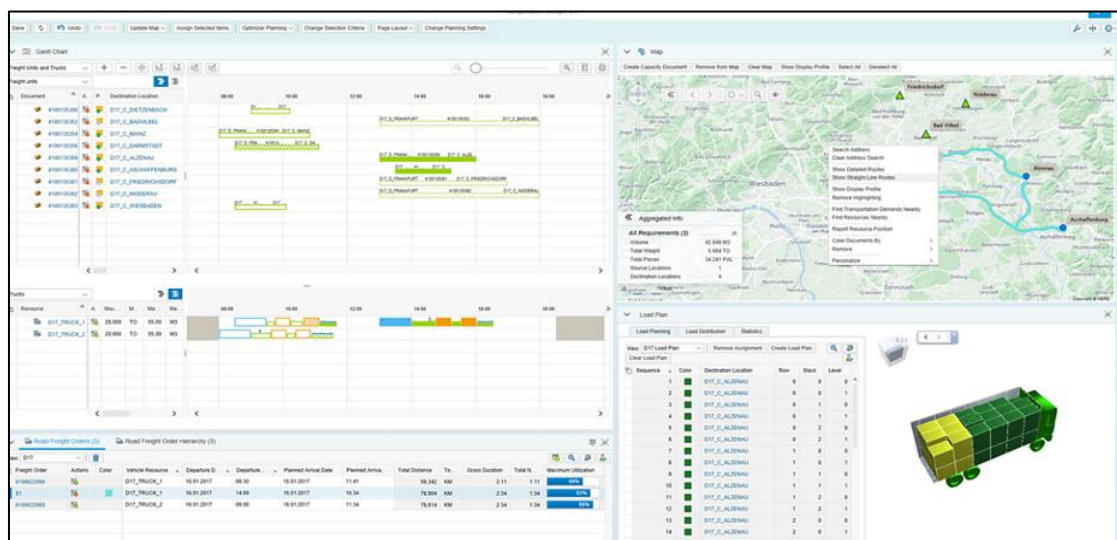


Рис. 1.1. Інтерфейс планування маршрутів у системі SAP TM

SAP TM - це одна з найбільш відомих і потужних підприємств-платформ глобального ринку. Рішення розроблене для великих міжнародних корпорацій із складними ланцюгами постачань. Воно покриває практично будь-який із можливих логістичних сценаріїв: мультимодальні перевезення (де задіяно і авто, і морський, і залізничний транспорт у межах одного замовлення), автоматичне 3D-планування

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

заповнення кузовного простору та прогнозування затримок за допомогою штучного інтелекту [4,11].

Розглядаючи SAP Transportation Management (SAP TM), варто відзначити її як одну з найбільш потужних енттерпрайз-платформ глобального ринку. Її головними перевагами є абсолютна функціональна повнота, найвищий ступінь інтеграції з фінансовими, податковими і складськими модулями материнської екосистеми SAP ERP, а також потужні аналітичні панелі на базі SAP HANA та підтримка всіх світових стандартів безпеки. Проте для підприємств малого та середнього бізнесу ця система має ряд суттєвих недоліків. Зокрема, екстремально висока вартість ліцензій та послуг інтеграторів робить її фінансово недоступною для невеликих компаній. Крім того, тривалий процес впровадження та історично складний для швидкого навчання інтерфейс значно ускладнюють роботу диспетчерів. Для компаній з парком до 50 автомобілів весь заявлений корпоративний функціонал стає надлишковим та не використовується повною мірою.

Oracle Transportation Management (OTM)

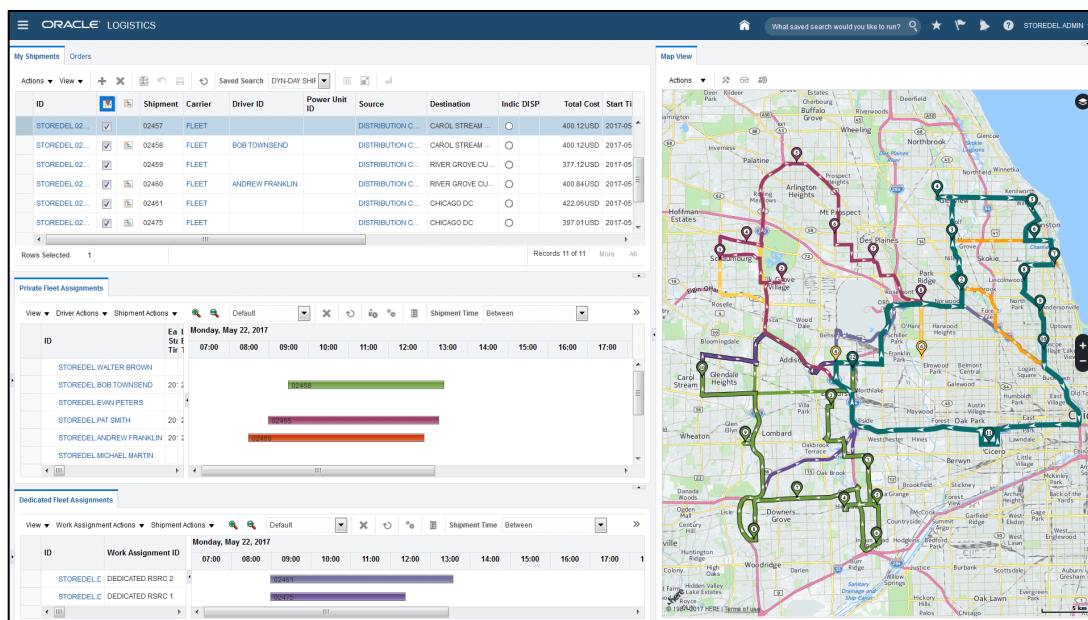


Рис. 1.2. Аналітична панель керівника у Oracle Transportation Management

Глобальна хмарна платформа від відомого розробника баз даних Oracle позиціонується як інструмент для транснаціонального бізнесу. Вона вирізняється сильною системою моделювання різноманітних логістичних сценаріїв,

можливістю гнучкого налаштування бізнес-правил та глибокою підтримкою міжнародного документообігу. Водночас недоліки ОТМ подібні до рішень від SAP: її архітектура є дуже масивною, а система не завжди здатна гнучко адаптуватися під специфічні локальні потреби невеликого перевізника, якому необхідно лише швидко згенерувати базову супровідну документацію. Вартість володіння цією платформою часто виявляється непомірною для регіональних українських перевізників.

Транспортна інформаційна система Sovtes

Sovtes - це популярна транспортна IT-система, розроблена безпосередньо в Україні. Вона добре адаптована під вітчизняний документообіг і часто використовується не лише перевізниками, а й вантажовласниками для організації тендерів на перевезення партій товару. Sovtes по суті є містком між замовником та виконавцем транспортної послуги.

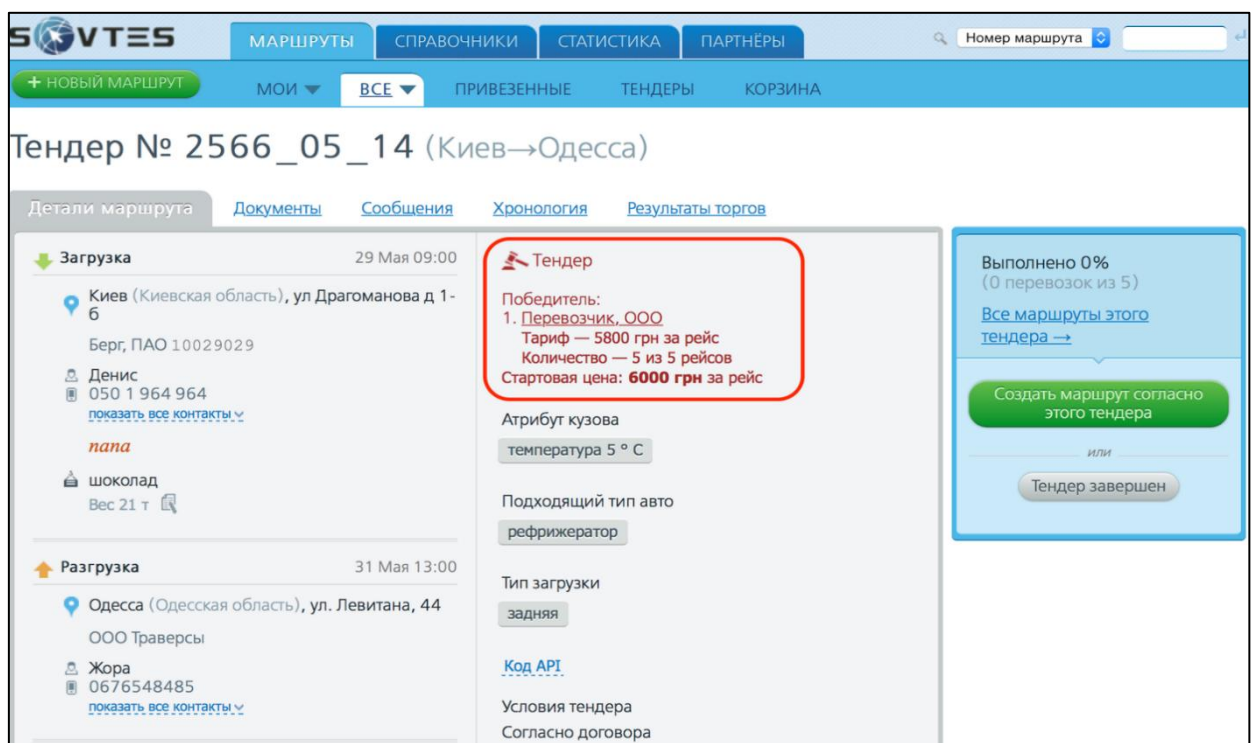


Рис. 1.3. Робочий стіл диспетчера в системі Sovtes

Аналізуючи вітчизняну транспортну інформаційну систему Sovtes, можна виділити її глибоку адаптацію під український ринок та його законодавчі формати, зокрема нативну підтримку електронної товарно-транспортної накладної (е-ТТН). Зручною є наявність власної біржі вантажів для автоматичного розподілу заявок та

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

використання хмарної моделі за підпискою, що робить старт роботи відносно недорогим. Однак ця платформа значною мірою зосереджена саме на процесах зовнішньої взаємодії та проведенні тендерів між вантажовласником та перевізником, тоді як внутрішні аспекти управління автопарком і його технічним обслуговуванням відходять на задній план. Крім того, система має досить жорстку філософію бізнес-процесу, що ускладнює її використання для підприємств, які не звикли працювати за тендерною моделлю.

Платформа Linqo

Ще одна SaaS-платформа, популярна серед експедиторів. Вона фокусується на швидкому обліку замовлень і фінансів для транспортних посередників і середніх транспортних компаній. Інтерфейс програми організований навколо реєстру заявок з можливістю бачити дохід і витрати по кожній із них.

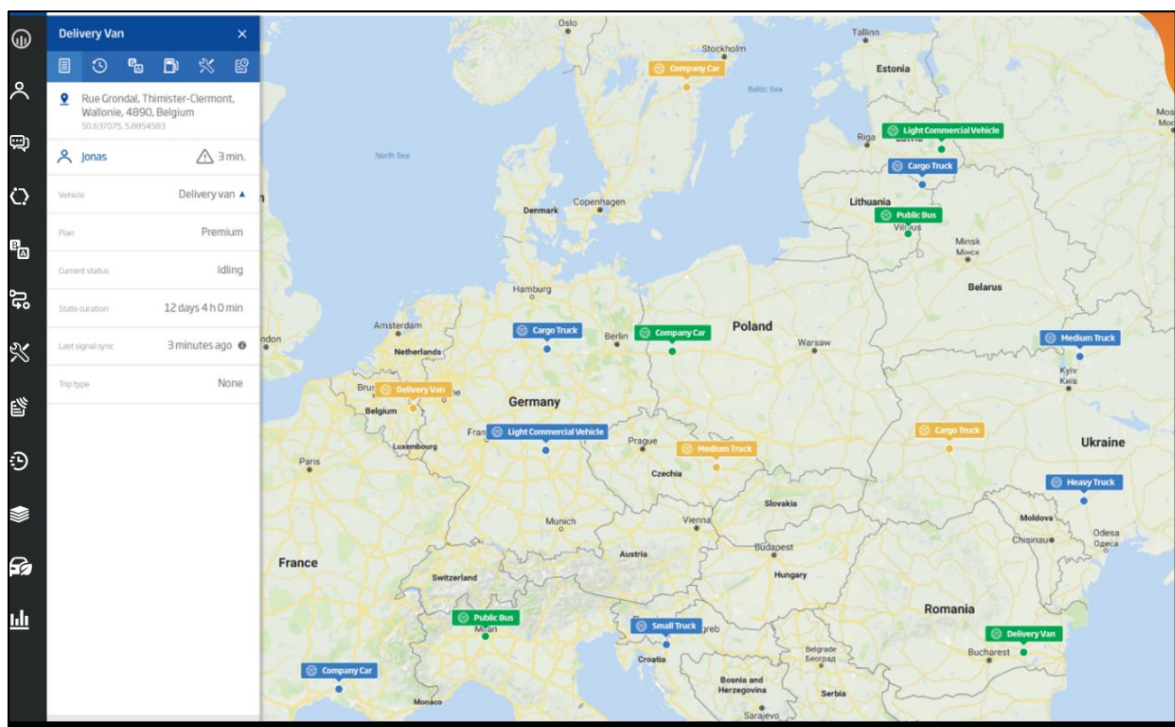


Рис. 1.4. Реєстр заявок в платформі Linqo

Платформа Linqo, яка є популярною серед транспортних експедиторів, пропонує зрозумілий інструментарій для обліку маржинальності та зручні засоби для генерації рахунків-фактур і договорів за готовими шаблонами. Незважаючи на ці переваги, її інтерфейс, побудований у вигляді масивних таблиць, часто виявляється неадаптивним та візуально перевантаженим під час роботи з екранів

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

мобільних пристроїв чи планшетів. Можливості відображення геоданих або відстеження водіїв у базовій комплектації є досить обмеженими і часто вимагають підключення платних зовнішніх рішень через API. До того ж, комерційна модель платформи передбачає оплату за кожне робоче місце, що призводить до не виправданих витрат при розширенні штату співробітників компанії.

Усі розглянуті системи мають як суттєві переваги, так і певні недоліки для обраної цільової аудиторії - транспортних компаній малого і середнього масштабу. Великі рішення (SAP, Oracle) недосяжні фінансово і заскладні функціонально, тоді як українські рішення (Sovtes, Linco) пропонують специфічні інструменти, але мають свої обмеження в гнучкості та інтерактивному гео-трекінгу безпосередньо в базі. Створювану в межах даної кваліфікаційної роботи вебплатформу вирішено було назвати LogisticSoft. В таблиці 1.1 відобразимо функціонал існуючих аналогів, а також позначимо плановані функції для нашої вебплатформи.

Таблиця 1.1.

Порівняльна характеристика існуючих програмних продуктів

Параметр порівняння	SAP TM	Oracle OTM	Sovtes	Linco	LogisticSoft
Масштаб платформи	Ентерпрайз (Великий бізнес)	Ентерпрайз (Великий бізнес)	МСБ та корпорації	МСБ та експедитори	Малий та середній бізнес
Архітектура/ Розгортання	On-premise / Хмарна	Хмарна	SaaS	SaaS	Хмарна стійка до швидкого розгортання
UX/Зручність інтерфейсу	Дуже складний	Складний	Середній рівень	Перевантажений таблицями	Інтуїтивно зрозумілий дизайн
Гео-відстеження / Карти	Нативна система + інтеграції	Нативна система	Обмежено/Інтегровано	Базова або зовнішні інтеграції	Інтеграція Leaflet для безкоштовного моніторингу

Генерація PDF (ТТН, рахунки)	Високоточна, але складна в налаштуванні	Є повна генерація документів	Адаптовано під український ринок	Автоматичні шаблони	Вбудована динамічна генерація PDF-файлів
Мультикомпа нійна архітектура	Так	Так	Частково	Так	Так (чітка ізоляція за tenant_id)
Відкрите API	Потужне, але платні конектори	Потужне	Є API	Комерційне	RESTful API за замовчуванням
Вартість експлуатації	Екстремально висока (до \$100 000+)	Дуже висока	Середня (плата за компанію або транзакції)	Оплата за кожне робоче місце	Низькі витрати за рахунок відкритого ПЗ

Враховуючи виявлені переваги та недоліки аналогів, вебплатформа "LogisticSoft" має бути спроектована таким чином, щоб заповнити нішу "ідеального балансу". Вона повинна надати виключно той функціонал, який справді необхідний: сучасний швидкий інтерфейс керування замовленнями, управління складом власного автопарку, можливість миттєвої генерації ТТН у форматі PDF, а також відображення об'єктів на безкоштовній відкритій карті. Наявність жорстко розмежованої рольової моделі зможе замінити хаотичне спілкування в месенджерах. Такий збалансований підхід дозволить зробити рішення потужним, але незатребуваним до величезних ресурсів при його розгортанні у клієнта.

1.3. Вибір архітектури програмного забезпечення

Проектування архітектури є визначальним етапом створення програмного забезпечення, оскільки від нього залежить надійність, продуктивність і здатність системи до подальшого розвитку. У сфері розробки інформаційних вебсистем

виділяють кілька класичних підходів, кожен з яких має свої переваги та обмеження в контексті автоматизації логістичних процесів [10].

Розглядаючи можливі архітектурні рішення, насамперед варто проаналізувати монолітну архітектуру. У такому підході модулі інтерфейсу користувача, бізнес-логіки та доступу до бази даних об'єднуються у єдиний кодовий базі та виконуються як один спільний процес на стороні сервера [14].

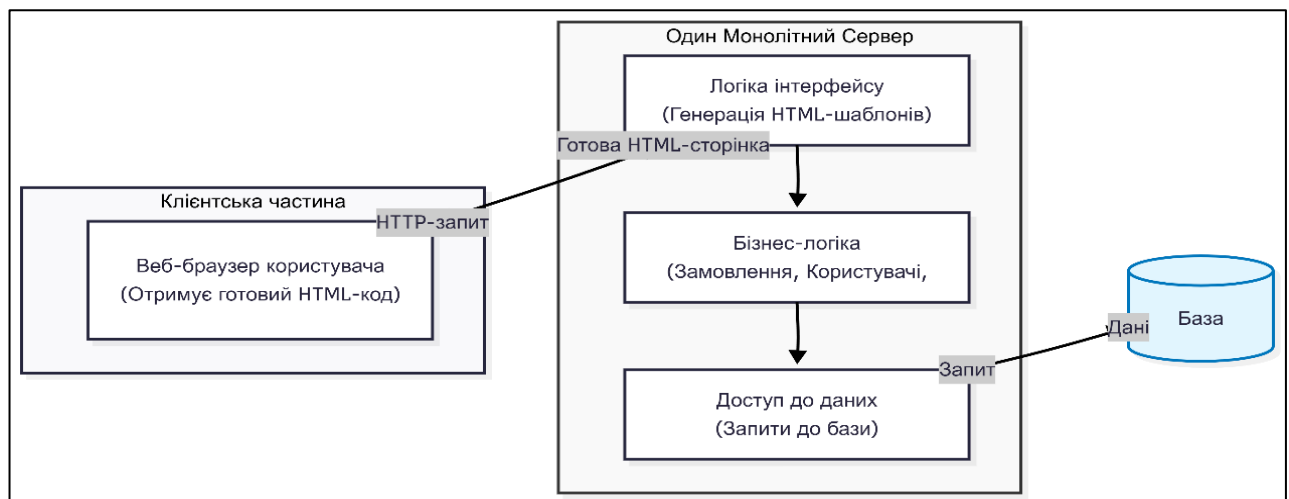


Рис. 1.5. Схема монолітної архітектури

Незважаючи на простоту розгортання на початкових етапах, для сучасних логістичних систем монолітний підхід є вкрай неефективним. Основною проблемою є тісна зв'язність компонентів: будь-яка зміна у логіці відображення вимагає перекомпіляції та перезапуску всього сервера. Крім того, класичний моноліт часто передбачає постійну генерацію статичних HTML-сторінок на стороні сервера, що суттєво збільшує навантаження на обчислювальні ресурси та вимагає стабільного широкосмугового з'єднання, яке не завжди доступне водіям під час виконання рейсів.

Альтернативним сучасним підходом є мікросервісна архітектура. Вона передбачає поділ єдиної інформаційної системи на множину повністю незалежних компонентів (сервісів), кожен з яких виконує вузькоспеціалізовану задачу та працює з власною базою даних [15]. Хоча цей підхід забезпечує максимальну гнучкість при масштабуванні глобальних проєктів, для розробки продукту орієнтованого на малий та середній бізнес він є надлишковим. Мікросервіси вимагають складного налаштування інфраструктури, управління контейнерами, а

також вирішення проблем із мережевими затримками при спілкуванні сервісів між собою. З огляду на необхідність створення надійної та простої в підтримці системи, цей архітектурний стиль було відхилено [26].

Враховуючи недоліки попередніх підходів, найбільш раціональним і збалансованим рішенням стало використання клієнт-серверної моделі із застосуванням архітектурного стилю REST API. Цей варіант базується на суворому фізичному та логічному розмежуванні системи на дві самостійні частини [13].

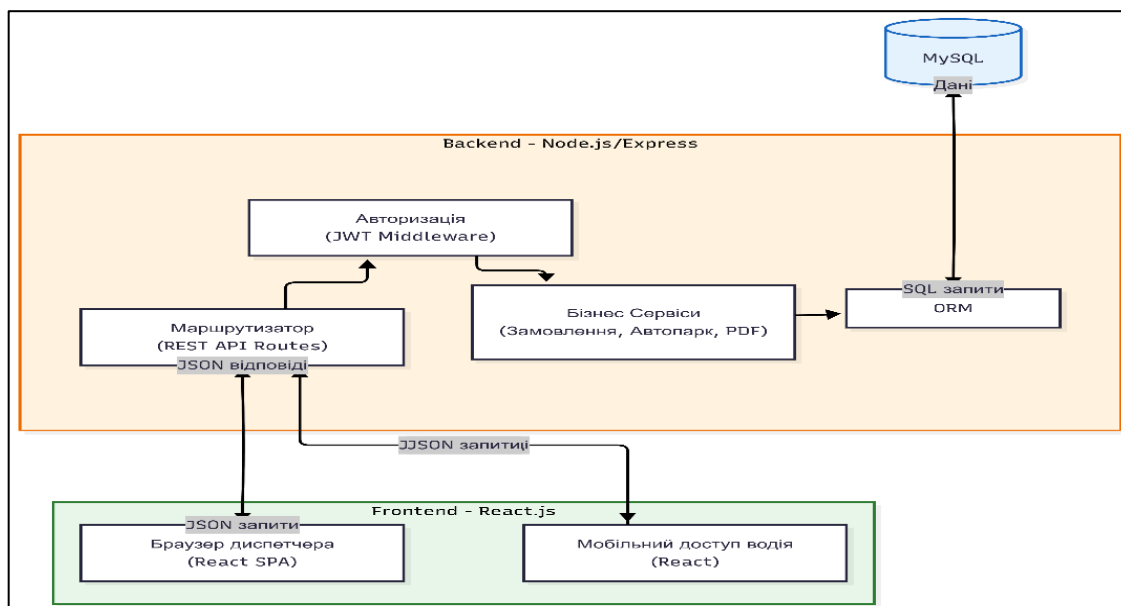


Рис. 1.6. Схема клієнт-серверного REST API

Серверна частина у цій моделі відповідає виключно за збереження і управління цілісністю інформації, обробку алгоритмів бізнес-логіки та перевірку прав доступу. Вона не бере участі у формуванні візуального інтерфейсу, а лише надає потрібні дані за запитом у стандартизованому текстовому форматі JSON [16]. Таке рішення дозволяє серверу функціонувати в ролі універсального та незалежного центру управління. Завдяки цій незалежності, до одного і того ж сервера можна одночасно підключати веббраузери диспетчерів, мобільні пристрої водіїв та зовнішні інтеграційні модулі, не змінюючи при цьому серверну логіку.

Клієнтська частина у такій архітектурі бере на себе всю відповідальність за взаємодію з користувачем. Застосування концепції односторінкового застосунку кардинально змінює підхід до відображення інформації [17]. Замість того, щоб щоразу завантажувати нові сторінки з сервера, браузер користувача завантажує

візуальний каркас програми лише один раз на початку роботи. Уся подальша комунікація із сервером здійснюється у фоновому режимі за допомогою асинхронних запитів. Коли диспетчер відкриває нове замовлення або водій оновлює статус рейсу, оновлюється лише певний інформаційний блок на екрані. Це мінімізує обсяг переданих через мережу даних і забезпечує високий візуальний комфорт роботи, уникаючи дратівливих перезавантажень сторінки.

Додатковою перевагою обраної REST API архітектури є відсутність стану на сервері. Кожен запит від клієнта містить усю необхідну інформацію для його обробки. Це знімає з сервера навантаження по зберігання сесій користувачів у оперативній пам'яті, що робить розроблену платформу значно стійкішою до збоїв та готовою до розширення у разі збільшення кількості компаній-перевізників.

Під час архітектурного проектування також було глибоко досліджено можливість застосування концепції безсерверних обчислень, яка набуває значної популярності у сучасній інженерії. Цей підхід передбачає розгортання окремих функцій системи у хмарних середовищах (наприклад, AWS Lambda або Google Cloud Functions), де оплата здійснюється виключно за час фактичного виконання коду [13]. Теоретично, це могло б суттєво знизити витрати на утримання серверної інфраструктури для невеликих логістичних компаній, оскільки у нічний час, коли активність платформи мінімальна, ресурси б не споживалися. Однак, детальний аналіз виявив критичний недолік цього підходу для логістичної сфери - так званий «холодний старт». Під час ініціалізації середовища для виконання функції виникають мікрозатримки, які є неприпустимими для системи моніторингу транспорту, що вимагає безперервного WebSocket-з'єднання для трансляції геолокації у режимі реального часу. Саме тому концепцію Serverless було відхилено на користь класичного постійно працюючого сервера.

Важливим аспектом обраної архітектури є стратегія її майбутнього масштабування. Монолітні бази даних часто стають «вузьким місцем» при експоненційному зростанні кількості користувачів. Тому архітектура «LogisticSoft» відпочатково проектується з урахуванням можливості горизонтального масштабування. Логіка Node.js-сервера побудована таким чином,

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

що вона не зберігає жодного локального стану. Це означає, що при зростанні навантаження адміністратори зможуть розгорнути два, три або десять ідентичних екземплярів сервера за балансувальником навантаження (наприклад, Nginx) [15]. Запити від мобільних додатків водіїв та браузерів диспетчерів рівномірно розподілятимуться між цими серверами. Оскільки сесії користувачів не прив'язані до конкретного вузла, а передаються у вигляді зашифрованих JWT-токенів безпосередньо у заголовках запитів, відмова одного із серверів залишиться абсолютно непомітною для кінцевих користувачів платформи.

На рівні архітектури зберігання інформації закладено можливість впровадження моделі реплікації. У майбутньому, коли обсяг даних (архівів рейсів, фінансових рахунків та аудит-логів) досягне терабайтних значень, архітектура дозволить розділити потоки даних. Основна база даних буде обробляти виключно транзакції запису (створення нових замовлень, оновлення статусів водіями), тоді як кілька підпорядкованих баз даних візьмуть на себе навантаження з читання [22]. Така архітектурна гнучкість гарантує, що побудована платформа не втратить своєї актуальності та продуктивності навіть за умови підключення десятків великих корпоративних автопарків.

Комплексний підхід до проектування структури, який включає розділення на frontend та backend, відмову від збереження станів на сервері та готовність до кластеризації, формує надійний, безпечний і стійкий до навантажень фундамент. Це дозволяє зосередити подальшу увагу безпосередньо на виборі конкретних інструментів та фреймворків для фізичного написання програмного коду, що є наступним логічним кроком у життєвому циклі розробки програмного забезпечення.

1.4. Обґрунтування вибору інструментальних засобів.

Стійкість та продуктивність будь-якого програмного комплексу безпосередньо залежить від обґрунтованості вибору технологічного стека та сумісності використаних інструментів. При проектуванні платформи «LogisticSoft» було прийнято стратегічне рішення щодо використання єдиної екосистеми на базі мови

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

програмування JavaScript для обох архітектурних ланок системи. Такий підхід дозволяє суттєво зменшити когнітивне навантаження під час розробки, оскільки відпадає необхідність постійного перемикавання між різними синтаксичними парадигмами [21]. Це також спрощує подальшу підтримку коду та обмін моделями даних між клієнтом і сервером.

Для реалізації серверної частини розглядалися кілька популярних технологій, зокрема мови Python та Java. Однак оптимальним вибором для логістичної платформи стало середовище виконання Node.js [18, 19]. Його головна перевага полягає у неблокувальній, асинхронно-подієвій моделі управління вводом-виводом. У контексті транспортної логістики, де система повинна постійно обробляти паралельні запити, транслювати координати та оновлювати статуси рейсів, традиційні синхронні моделі вимагали б значно більших апаратних ресурсів. Node.js дозволяє опрацьовувати тисячі подібних операцій у межах єдиного потоку без надмірного споживання оперативної пам'яті. Маршрутизація цих мережевих запитів забезпечується за допомогою мінімалістичного фреймворку Express, який залишає архітектору повну свободу у проектуванні логіки без нав'язування жорстких структур [20].

Особливої уваги вимагав вибір механізмів зберігання інформації. Незважаючи на популярність нереляційних баз даних, сфера транспортної логістики вимагає дуже жорсткої дисципліни. Інформаційні сутності, такі як перевізник, водій, транспортний засіб та замовлення, перебувають у строгих взаємозалежностях. Саме тому основою сховища було обрано класичну реляційну систему управління базами даних MySQL [21]. Вона автоматично контролює обмеження посиловної цілісності та повністю відповідає вимогам транзакційності ACID, що є критично важливим фактором при збереженні фінансових рахунків [22]. Для програмної взаємодії з базою даних застосовується інструмент об'єктно-реляційного відображення Sequelize [27]. Використання ORM дозволяє абстрагуватися від написання сирих SQL-запитів, автоматично захищає систему від спроб несанкціонованих ін'єкцій коду та забезпечує зручний механізм управління версіями структури бази даних.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Клієнтський інтерфейс розробляється за допомогою фронтенд-бібліотеки React.js. Під час аналізу альтернатив розглядалися інші популярні фреймворки, проте React.js запропонував найкращий баланс між гнучкістю та продуктивністю для односторінкового застосунку [23]. Завдяки концепції віртуальної об'єктної моделі документа (Virtual DOM), ця технологія обчислює різницю між поточним та новим станом екрана, виконуючи точкові маніпуляції лише з тими блоками, де змінилася інформація. Це забезпечує плавне оновлення інтерфейсу без перезавантаження сторінки браузера, що створює комфортні умови для багатогодинної роботи диспетчера. Візуальне оформлення, побудова модульної сітки та адаптивність дизайну під екрани мобільних пристроїв реалізовані за допомогою стилістичного фреймворку Bootstrap 5 [29].

Додатковий технологічний шар формують спеціалізовані програмні модулі. Для вирішення ключового завдання - відображення геоданих та маршрутів перевезення - було вирішено відмовитися від платних комерційних сервісів. Натомість у систему інтегровано відкриту картографічну бібліотеку Leaflet у поєднанні з безкоштовними картами OpenStreetMap [24]. Генерація юридично значущих документів у форматі PDF, зокрема товарно-транспортних накладних, виконується безпосередньо на стороні сервера за допомогою бібліотеки PDFKit [30]. Інформаційна безпека, ідентифікація користувачів та перевірка прав доступу реалізовані через індустріальний стандарт JSON Web Token (JWT) [32]. Такий підхід усуває необхідність збереження сесій у пам'яті сервера, що суттєво підвищує відмовостійкість архітектури.

Для якісної реалізації такого комплексу використовувалися професійні інструменти життєвого циклу розробки. Для контролю версій програмного коду та уникнення втрати даних застосовувалася система Git. Тестування працездатності серверної логіки та симуляція клієнтських запитів здійснювалися за допомогою програмного середовища Postman [35]. Застосування цього збалансованого комплексу інструментальних засобів гарантує створення надійної та готової до майбутнього масштабування логістичної платформи.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Для забезпечення повної впевненості в надійності обраного стеку, варто більш детально розглянути внутрішні механізми середовища Node.js. В його основі лежить рушій V8, розроблений компанією Google, який виконує компіляцію JavaScript-коду безпосередньо в машинний код. Але головною рушійною силою платформи є цикл подій та бібліотека libuv. У багатопотокових мовах, таких як Java або PHP, кожен новий запит від клієнта створює окремий потік операційної системи, що резервує значний обсяг оперативної пам'яті (від 2 до 8 МБ на потік). У пікові моменти, коли тисячі водіїв одночасно надсилають свої GPS-координати, багатопотоковий сервер міг би вичерпати доступну пам'ять [19]. Натомість Node.js працює в одному основному потоці. Коли надходить запит на читання з бази даних або генерацію PDF, система не блокує потік очікуванням, а відправляє завдання в пул потоків на рівні операційної системи, продовжуючи приймати нові запити. Саме ця неблокувальна природа I/O операцій робить Node.js безальтернативним лідером для логістичних систем реального часу.

З боку клієнтської частини вибір React.js також підкріплюється його екосистемою управління станами. У процесі роботи диспетчера сторінка моніторингу постійно отримує нові масиви даних про замовлення. У класичних додатках це вимагало б складного маніпулювання об'єктною моделлю документа, що неминуче призводить до уповільнення відмальовування інтерфейсу браузером. React.js вирішує цю проблему завдяки алгоритму узгодження. Бібліотека тримає в оперативній пам'яті легку копію структури сторінки (Virtual DOM). Отримавши нові координати вантажівки, React.js швидко обчислює різницю між поточним та новим станом, і застосовує зміни виключно до того HTML-елемента, який відповідає за маркер на карті [23]. Інша частина сторінки (таблиці, меню, графіки) залишається недоторканою, що економить ресурси процесора на робочому комп'ютері диспетчера.

Окрему увагу в процесі розробки приділено інструментам криптографічного захисту. Оскільки система зберігає конфіденційні дані контрагентів та комерційні рахунки, застосування стандарту JSON Web Token було підсилено механізмами хешування паролів за допомогою алгоритму bcrypt. На відміну від застарілих

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

алгоритмів на кшталт MD5 чи SHA-256, bcrypt є адаптивним алгоритмом формування ключа. Він автоматично додає до кожного пароля випадкову послідовність символів (так звану "сіль", salt), що робить атаки з використанням веселкових таблиць математично неможливими [34]. Крім того, алгоритм навмисно сповільнює процес обчислення хешу (фактор вартості), що ефективно протидіє спробам злому шляхом прямого перебору паролів на потужних відеокартах. Комплексне використання Node.js для асинхронної обробки, React.js для високопродуктивного інтерфейсу та сучасних стандартів шифрування гарантує створення безпечного програмного продукту промислового рівня.

Висновки до першого розділу

У першому розділі було здійснено комплексний аналіз предметної області управління вантажними перевезеннями. Встановлено, що використання неінтегрованих засобів управління та дублювання інформації в електронних таблицях значно знижують швидкість роботи диспетчерів та створюють постійні ризики організаційних збоїв.

Дослідження існуючих аналогів довело наявність ринкової потреби в збалансованій, незалежній та доступній SaaS-платформі для підприємств малого та середнього бізнесу. Виявлено, що користувачам потрібен лише критично важливий функціонал: облік замовлень, оперативне управління автопарком, моніторинг перевезень і автоматична генерація документації (зокрема ТТН) без надмірних ускладнень.

Відповіддю на ці вимоги стало проектування вебплатформи «LogisticSoft».

Ухвалено рішення про перехід до класичної клієнт-серверної системи з архітектурою односторінкового застосунку (SPA) через прикладний інтерфейс REST API. Затверджений технологічний стек на базі платформи JavaScript (Node.js/Express) та реляційної системи баз даних MySQL обґрунтовано відповідає інженерним вимогам швидкості обробки транзакцій та гарантії цілісності даних. Водночас використання бібліотеки React.js у зв'язці з відкритою картографією Leaflet забезпечує високу візуальну та операційну гнучкість фронтенду.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Сформований технологічний фундамент та професійна організація ітеративної розробки під управлінням системи контролю версій Git повністю підготували підґрунтя до наступного етапу системи. Цей перехід до безпосереднього проєктування концептуальної моделі реляційної бази даних, аналізу варіантів використання та розробки загальних алгоритмів взаємодії буде детально розкрито у другому розділі кваліфікаційної роботи.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ ВАНТАЖОПЕРЕВЕЗЕННЯМИ

2.1. Визначення варіантів використання та структури системи

Проектування інформаційної системи базується на формалізації вимог, що визначають архітектурні, функціональні та експлуатаційні характеристики майбутнього програмного продукту. Для забезпечення системного підходу до розробки всі вимоги було класифіковано за їхнім цільовим призначенням та зведено у відповідні специфікації.

Повний перелік можливостей системи, які визначають поведінку застосунку та взаємодію користувача з ним, наведено у вигляді функціональних вимог (табл. 2.1). Якісні характеристики платформи, такі як надійність, продуктивність, масштабованість та безпека, описано у структурі нефункціональних вимог (табл. 2.2). Технічний базис, необхідний для успішного розгортання та стабільної роботи інформаційної системи, сформовано у вигляді вимог до апаратного та програмного забезпечення (табл. 2.3).

Таблиця 2.1.

Функціональні вимоги до вебплатформи

Категорія вимог	Опис вимоги
Ізоляція просторів (Multitenancy)	Забезпечення роботи єдиної кодової бази для багатьох транспортних компаній з жорсткою ізоляцією даних на рівні ідентифікатора орендаря (Company ID) у кожній таблиці БД.
Керування компаніями	Можливість подачі зовнішніх заявок на створення профілю компанії з подальшим їх розглядом та переведенням в активний статус суперадміністратором.
Керування доступом (RBAC)	Підтримка багаторівневої системи доступу на основі ролей. Заборона горизонтального доступу до даних між користувачами різних компаній.

продовж. табл. 2.1

Адміністрування маршрутів	Формування замовлень із фіксацією ваго-габаритних параметрів, географічних координат пунктів завантаження/вивантаження та розрахунком вартості перевезення.
Робота зі специфікаціями	Передача та збереження специфікацій вантажу (номенклатури товарів) за допомогою серіалізації масивів у формат JSON.
Керування користувачами	Забезпечення процедури реєстрації працівників, зміни паролів, скидання активних сеансів безпеки та ведення обліку останньої активності.
Управління автопарком	Виконання базових операцій (збереження, читання, оновлення, видалення) з даними про транспортні засоби підприємства та їх жорстка прив'язка до відповідальних водіїв. Перегляд додаткової інформації такої як номерний знак.

Таблиця 2.2.

Нефункціональні вимоги до вебплатформи

Категорія вимог	Опис вимоги
Безпека даних та авторизація	Використання моделі авторизації JWT. Обов'язкове хешування паролів за алгоритмом bcrypt. Захист від поширених вразливостей.
Продуктивність та швидкодія	Обробка запитів до API у межах 500 мс. Використання архітектури односторінкового застосунку для мінімізації перезавантажень сторінки.
Масштабованість	Здатність реляційної моделі даних та сервера до збільшення кількості компаній-орендарів без втрати продуктивності.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

Сумісність та кросбраузерність	Коректне відображення клієнтського інтерфейсу у сучасних версіях популярних веббраузерів (Google Chrome, Safari, Mozilla Firefox).
Адаптивність інтерфейсу	Використання принципів Responsive Web Design (зокрема сітки Bootstrap) для гарантування зручної роботи мобільних користувачів (водіїв) з екранів смартфонів.
Надійність роботи	Збереження цілісності даних завдяки механізмам каскадних оновлень (CASCADE) у базі даних та налаштуванню системи автоматичного перезапуску сервера (PM2) при критичних збоях.

Таблиця 2.3.

Вимоги до апаратного та програмного забезпечення

Компонент системи	Мінімальні системні вимоги
Серверна частина (Production)	ОС: Linux (Ubuntu 22.04+ або Debian 11+). Процесор: 2 ядра (2.0 GHz та вище). ОЗП: 2 ГБ. Диск: 40 ГБ SSD (для ОС, СУБД MySQL та логів). Мережа: від 100 Мбіт/с.
Робоче місце диспетчера/адміністратора (ПК)	ОС: Windows 10/11, macOS 11+, або сучасні дистрибутиви Linux. Процесор: Intel Core i3 / AMD Ryzen 3 або еквівалент. ОЗП: 4 ГБ. Дисплей: роздільна здатність від 1366x768. Веббраузер: Google Chrome 90+, Firefox 90+, Safari 14+.
Мобільний клієнт водія (Смартфон)	ОС: Android 8.0+ або iOS 17.0+. ОЗП: від 2 ГБ. Апаратне забезпечення: наявність вбудованого модуля GPS для геолокації. Мережа: підключення 3G/4G/LTE. Веббраузер: мобільний Chrome або Safari.

Для структурування бізнес-вимог реалізовано моделювання варіантів використання системи. Враховуючи обрану модель контролю доступу на основі ролей (RBAC), в інфраструктурі системи виділено чотири основні типи користувачів [10]. Найвищий рівень повноважень має суперадміністратор, який керує системою глобально і приймає рішення щодо схвалення заявок від нових логістичних компаній. На рівні окремого підприємства-орендаря головним є адміністратор компанії. Він відповідає за початкове налаштування системи, реєстрацію працівників у базі та внесення даних про автопарк. Основною операційною ланкою виступає диспетчер, у чії обов'язки входить ініціювання нових замовлень, призначення водіїв на рейси та формування фінансових рахунків. Останньою ланкою є водій, який через мобільний клієнт отримує доступ виключно до читання призначених йому маршрутів та можливість змінювати їхні поточні статуси, без прав на редагування реквізитів.

Візуалізація загальної діаграми прецедентів системи стосовно модуля транспортування наведена на рис. 2.1.

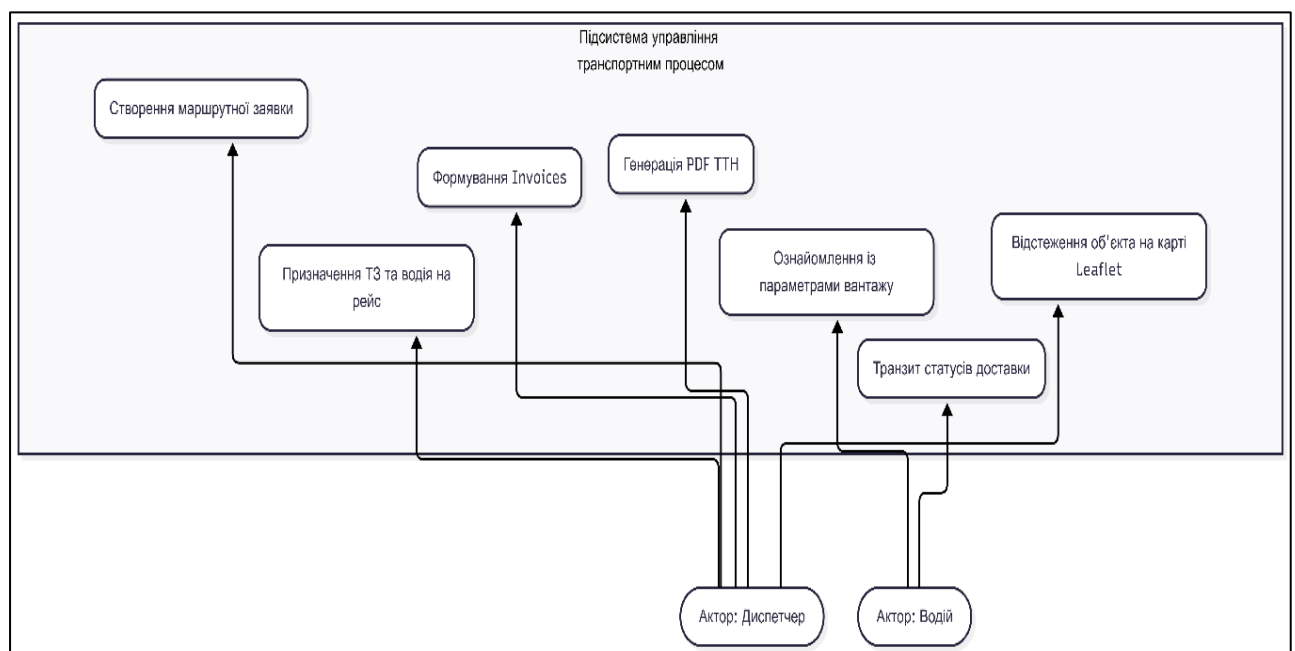


Рис. 2.1. Розподіл варіантів використання в підсистемі управління рейсом

На основі UML-моделювання розроблено архітектурні специфікації до кожного процесу, які виступають технічним завданням для написання коду застосунку.

Таблиця 2.4.

Специфікація прецеденту UC-1 «Формування заявки»

Параметр специфікації	Опис та значення параметра
Назва прецеденту	UC-1 «Формування заявки»
Основний актор (Ініціатор)	Диспетчер
Умова старту (Pre-condition)	Наявна активна сесія авторизації користувача (валідний JWT-токен)
Основний сценарій (Алгоритм дій)	1. Актор заповнює вхідну вебформу параметрами замовлення. 2. Клієнтська частина надсилає асинхронний POST-запит із JSON-масивом даних. 3. Обробник маршруту на сервері виконує валідацію типів вхідного тіла запиту. 4. Після успішної перевірки викликається ORM-метод для створення сутності Order. 5. У поле companyId автоматично додається ідентифікатор із розшифрованого JWT-токена диспетчера для ізоляції даних.
Кінцевий стан (Post-condition)	Нову сутність успішно записано в реляційну БД. Системний журнал (Audit Logs) зафіксував подію створення об'єкта.

Таблиця 2.5.

Специфікація прецеденту UC-2 «Зміна статусу доставки»

Параметр специфікації	Опис та значення параметра
Назва прецеденту	UC-2 «Зміна статусу доставки»
Основний актор (Ініціатор)	Водій
Умова старту (Pre-condition)	Система містить замовлення у фазі ASSIGNED, до якого безпосередньо прив'язано унікальний ID даного водія.
Основний сценарій (Алгоритм дій)	1. Водій активує інтерфейсний тригер зміни етапу рейсу (наприклад, «Розвантажено») у мобільному клієнті. 2. Застосунок ініціює асинхронний PUT-запит на серверний ендпоінт /orders/{id}/status.
Основний сценарій (Алгоритм дій)	3. Серверний маршрутизатор перевіряє відповідність цільового поля order.driverId до поточного token.userId. 4. Виконується транзакційний SQL-запит оновлення стану: UPDATE table SET status = 'UNLOADED'.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

Кінцевий стан (Post-condition)	Статус рейсу синхронізовано в базі даних. Подія оновлення через WebSocket миттєво візуалізується в диспетчерському інтерфейсі моніторингу автопарку.
--------------------------------	--

2.2. Проектування реляційної бази даних

Для забезпечення гарантії цілісності інформаційного масиву обрано класичну реляційну систему управління базами даних MySQL. Інтеграція між застосунком Node.js і MySQL реалізована за допомогою інструменту Sequelize. Об'єктно-реляційне відображення (ORM) дозволило абстрагуватися від "базового" SQL і описати таблиці через визначення моделей JavaScript, що підвищує кросплатформеність коду.

Схема нормалізована до 3НФ. Усунуті транзитивні залежності. Впроваджено 11 табличних сутностей, пов'язаних строгими обмеженнями зовнішніх ключів.

Опис структури об'єктів бази даних. Нижче наведена структура основних таблиць із вказівкою типів даних бази, модифікаторів обмеження та логічного обґрунтування атрибутів.

Таблиця 2.6.

Структура сутності «COMPANIES

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Ідентифікатор компанії	PRIMARY KEY, AUTO_INCREMENT
name	VARCHAR(255)	Назва транспортної компанії	NOT NULL
address	VARCHAR(255)	Юридична адреса компанії	NULL
edrpou	VARCHAR(50)	Код ЄДРПОУ компанії	NULL
phone	VARCHAR(50)	Контактний номер телефону	NULL

продовж. табл. 2.6

email	VARCHAR(255)	Контактна електронна пошта	NULL
isActive	BOOLEAN	Статус активності (схвалено)	DEFAULT true
isSystem	BOOLEAN	Маркер системної компанії	DEFAULT false

Таблиця 2.7.

Структура сутності «USERS»

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Системний номер користувача	PRIMARY KEY, AUTO_INCREMENT
email	VARCHAR(150)	Логін (електронна пошта)	UNIQUE, NOT NULL
password	VARCHAR(255)	Хешоване значення доступу	NOT NULL
fullName	VARCHAR(255)	Фізичне ПІБ співробітника	NOT NULL
role	ENUM	Набір жорстких ролей	DEFAULT 'DRIVER'
companyId	INT	Належність працівника до підприємства	FK REFERENCES Companies(id)
twoFactorEnabled	BOOLEAN	Статус двофакторної авторизації	DEFAULT FALSE

Таблиця 2.8.

Структура сутності «VEHICLES»

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Системний номер автотранспорту	PRIMARY KEY, AUTO_INCREMENT
make	VARCHAR(100)	Марка транспортного засобу	NULL
licensePlate	VARCHAR(20)	Державні знаки реєстрації автомобіля	UNIQUE, NOT NULL

продовж. табл. 2.8

capacity	FLOAT	Параметри вантажопідйомності (т)	NULL
driverId	INT	Ідентифікатор закріпленого водія	UNIQUE FK REFERENCES Users(id)
companyId	INT	Ізоляційний ключ власника	FK REFERENCES Companies(id)

Таблиця 2.9.

Структура сутності «ORDERS»

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Реєстраційний індекс заявки	PRIMARY KEY, AUTO_INCREMENT
description	VARCHAR(255)	Найменування товарної групи	NOT NULL
weight	FLOAT	Контрольна вага (кг / т)	NOT NULL
pickupAddress	VARCHAR(500)	Географічна точка завантаження товару	NOT NULL
deliveryAddress	VARCHAR(500)	Географічна точка призначення	NOT NULL
status	ENUM ('NEW', 'LOADED'...)	Маркер стану проходження маршруту	DEFAULT 'NEW'
cargoItems	JSON	Серіалізована специфікація номенклатури вантажу	JSON TYPE
creatorId	INT	Відсилка на обліковий запис диспетчера	FK REFERENCES Users(id)
driverId	INT	Відсилка на виконавця рейсу	FK REFERENCES Users(id)
companyId	INT	Базовий ключ багатоорендарності БД	FK REFERENCES Companies(id)
customerId	INT	Індекс контрагента вантажовідправника	FK REFERENCES Customers(id)

Окрім наведених основних сутностей, у фізичній моделі бази даних також реалізовано ряд допоміжних таблиць для забезпечення внутрішньої логіки

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

та безпеки платформи: CUSTOMERS, INVOICES, NOTIFICATIONS, COMPANY_REQUESTS, AUDIT_LOGS, ORDER_EXPENSES та ORDER_COMMENTS. Детальна структура цих допоміжних сутностей наведена у таблицях 2.10 - 2.16. Їхні зв'язки з основними таблицями відображено на загальній ER-діаграмі реляційної моделі даних.

Таблиця 2.10.

Структура сутності «CUSTOMERS»

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Ідентифікатор запису клієнта	PRIMARY KEY, AUTO_INCREMENT
name	VARCHAR(255)	Назва компанії-замовника	NOT NULL
edrpou	VARCHAR(100)	Код ЄДРПОУ / ПІН	NULL
address	VARCHAR(255)	Юридична адреса	NULL
phone	VARCHAR(50)	Контактний телефон	NULL
email	VARCHAR(150)	Електронна пошта	NULL
companyId	INT	Належність клієнта до компанії-орендаря	FK REFERENCES Companies(id)

Таблиця 2.11.

Структура сутності «INVOICES»

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Системний номер рахунку	PRIMARY KEY, AUTO_INCREMENT
orderId	INT	Відсилка на замовлення	FK REFERENCES Orders(id), NOT NULL
customerId	INT	Відсилка на клієнта	FK REFERENCES Customers(id), NOT NULL
tempid	INT	Тимчасовий ідентифікатор	FK REFERENCES Customers(id), NOT NULL
companyId	INT	Ізоляційний ключ компанії	FK REFERENCES Companies(id), NOT NULL

продовж. табл. 2.11

amount	FLOAT	Сума рахунку до сплати	NOT NULL
status	ENUM	Статус оплати ('UNPAID', 'PAID', 'CANCELLED')	DEFAULT 'UNPAID'
fileUrl	VARCHAR(255)	Посилання на згенерований PDF-документ	NULL

Таблиця 2.12.

Структура сутності «NOTIFICATIONS»

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Системний номер сповіщення	PRIMARY KEY, AUTO_INCREMENT
message	VARCHAR(255)	Текст системного сповіщення	NOT NULL
isRead	BOOLEAN	Статус прочитання	DEFAULT FALSE
type	VARCHAR(50)	Категорія/тип сповіщення	NULL
userId	INT	Отримувач сповіщення	FK REFERENCES Users(id)

Таблиця 2.13.

Структура сутності «COMPANY_REQUESTS»

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Системний номер заявки	PRIMARY KEY, AUTO_INCREMENT
companyName	VARCHAR(255)	Назва юридичної особи, що реєструється	NOT NULL
uniqueId	VARCHAR(100)	Бажаний унікальний код (аліас)	UNIQUE, NOT NULL
adminEmail	VARCHAR(150)	Пошта головного адміністратора	UNIQUE, NOT NULL
adminPassword	VARCHAR(255)	Хешований пароль майбутнього адміна	NOT NULL
phone	VARCHAR(50)	Контактний телефон заявника	NOT NULL

продовж. табл. 2.13

status	ENUM	Статус розгляду заявки	DEFAULT 'PENDING'
processedAt	DATETIME	Час обробки заявки	NULL
processedBy	INT	Суперадміністратор, який обробив запит	NULL

Таблиця 2.14.

Структура сутності «ORDER_EXPENSES»

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Ідентифікатор витрати	PRIMARY KEY, AUTO_INCREMENT
orderId	INT	Замовлення, з яким пов'язана витрата	FK REFERENCES Orders(id), NOT NULL
category	ENUM	Категорія витрати (FUEL, TOLL_ROADS тощо)	NOT NULL
amount	FLOAT	Грошова сума витрати	NOT NULL
description	VARCHAR(255)	Додатковий опис або примітка	NULL
companyId	INT	Належність витрати до компанії	FK REFERENCES Companies(id), NOT NULL
createdBy	INT	Працівник, який вніс витрату	FK REFERENCES Users(id), NOT NULL

Таблиця 2.15.

Структура сутності «AUDIT_LOGS»

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Ідентифікатор запису журналу	PRIMARY KEY, AUTO_INCREMENT
ipAddress	VARCHAR(50)	IP-адреса, з якої надійшов запит	NULL
userId	INT	Користувач, який ініціював дію	NULL

продовж. табл. 2.15

action	ENUM	Тип дії ('CREATE', 'UPDATE', 'DELETE'...)	NOT NULL
entity	VARCHAR(100)	Назва таблиці/сутності, що була змінена	NOT NULL
entityId	INT	Ідентифікатор зміненого запису	NULL
details	JSON	Серіалізовані деталі змін або помилок	NULL

Таблиця 2.16.

Структура сутності «ORDER_COMMENTS»

Назва поля	Тип даних SQL	Призначення	Обмеження / Ключі
id	INT	Ідентифікатор коментаря	PRIMARY KEY, AUTO_INCREMENT
orderId	INT	Відсилка на замовлення (рейс)	FK REFERENCES Orders(id), NOT NULL
userId	INT	Автор коментаря	FK REFERENCES Users(id), NOT NULL
message	TEXT	Текстовий вміст коментаря	NOT NULL

Для проєктування архітектурно-цілісних залежностей в ORM-моделях Sequelize було визначено чіткі типи реляційних зв'язків. Загальна ER-діаграма логічного взаємозв'язку між таблицями (рис. 2.2) наочно ілюструє забезпечення нормалізації інформаційної інфраструктури застосунку. Важливим аспектом архітектури є впровадження зовнішніх ключів, що гарантує стабільність бази даних під час масових операцій видалення. Зокрема, правила каскадного оновлення та видалення (ON DELETE SET NULL або CASCADE) налаштовані таким чином, щоб забезпечити абсолютну недоторканість архівних даних у випадку видалення облікового запису працівника, який раніше створював чи виконував замовлення.

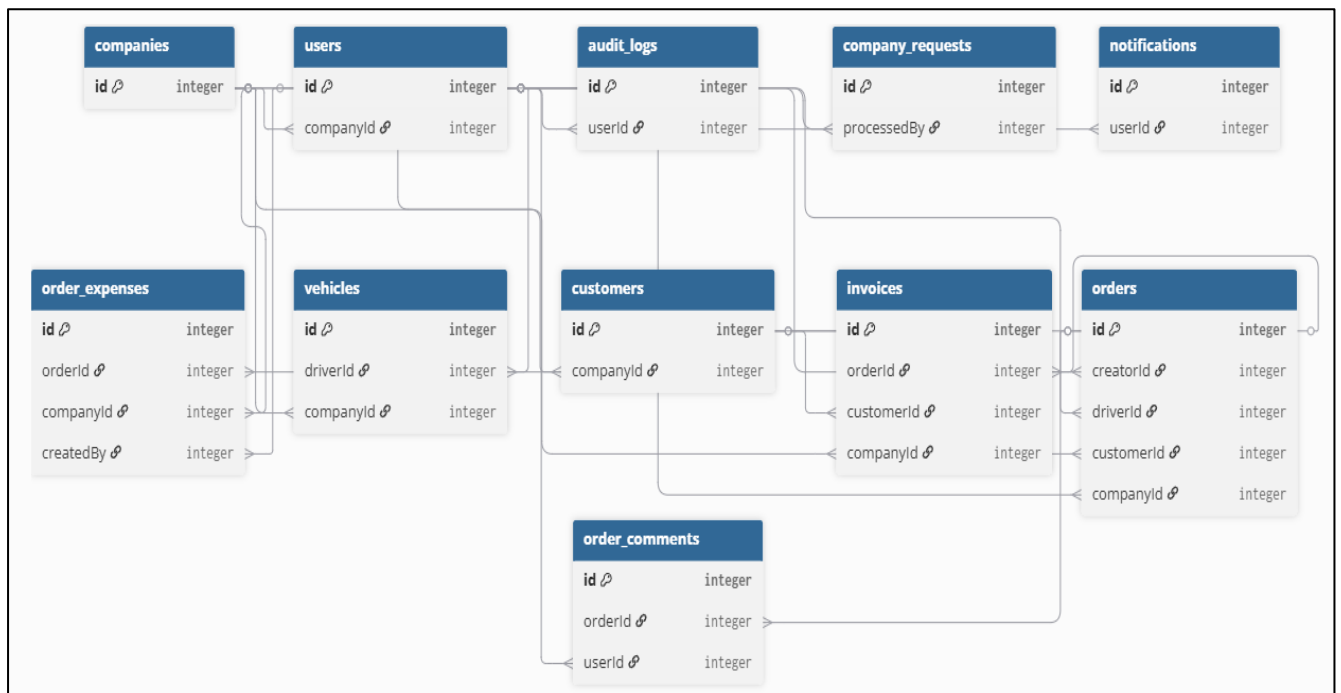


Рис. 2.2. ER-Діаграма реляційної моделі даних

2.3. Проєктування та розробка алгоритмів роботи вебплатформи.

На прикладний інтерфейс програмування (API) скомпоновано за моделлю архітектури REST. Обмін даними здійснюється по протоколу HTTP(S) у форматі JSON. Для управління всіма 10 сутностями бази даних спроектовано близько 40 маршрутних вузлів. Кожен вузол має чітку спеціалізацію щодо методів *GET*, *POST*, *PUT*, *DELETE*.

Для ідентифікації клієнта та безпечного обміну даними (без необхідності зберігати інформаційні сесії на сервері) впроваджено алгоритм Stateless JWT Auth.

Логіка транзакції авторизації за допомогою JSON Web Token (JWT) реалізована у вигляді послідовного серверного алгоритму. Процес розпочинається з того, що клієнтський застосунок надсилає POST-запит на мережевий маршрут `/api/auth/login`, передаючи у тілі запиту електронну пошту та пароль. Після отримання запиту контролер ініціює асинхронний метод `User.findOne()` для пошуку відповідного запису. За умови присутності користувача в базі даних, алгоритм задіює криптографічну бібліотеку `bcrypt`. Її вбудований метод `compareSync` здійснює перевірку, звіряючи хешований у базі рядок пароля з тим, що був надісланий клієнтом, використовуючи механізм криптографічної "солі" та

багатоетапного перетворення. У разі успішної перевірки автентичності контролер формує об'єкт Payload, до якого обов'язково включаються ідентифікатор користувача (userId), його системна роль та унікальний ідентифікатор компанії. На фінальному етапі глобальний закритий ключ сервера шифрує сформований Payload алгоритмом цифрового підпису, після чого згенерований рядок токена повертається до браузера клієнта.

Цей алгоритм перевірки та валідації виконується при здійсненні кожного запиту до закритої частини API. Витягування атрибуту ізоляції companyId безпосередньо під час розшифрування JWT усуває потенційну можливість ін'єкції або несанкціонованого доступу за прямим ресурсовим ID до бази даних чужої фірми (рис. 2.3).

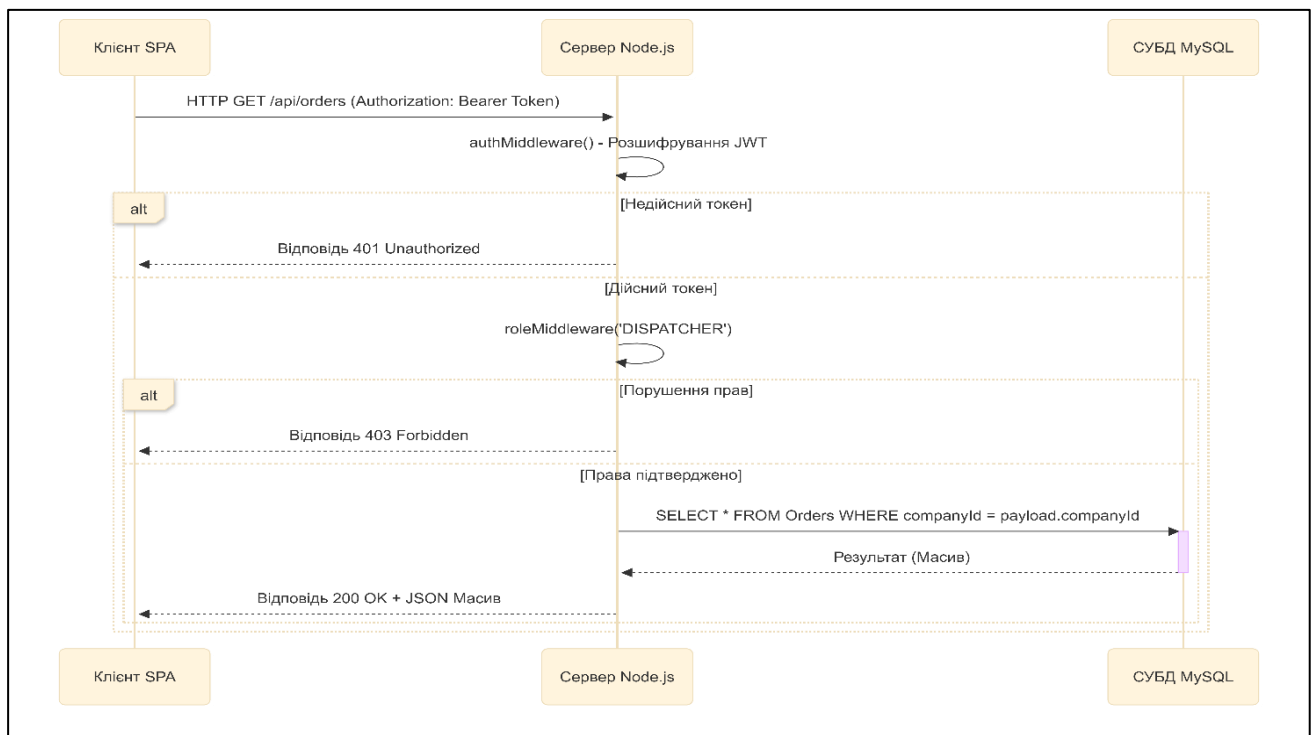


Рис. 2.3. Алгоритм виконання фільтрації доступу за допомогою проміжних обробників

Щодо алгоритмічної моделі генерації документів, серверну архітектуру було розроблено таким чином, щоб максимально мінімізувати навантаження на систему збереження файлів. Завдання автоматичного формування товарно-транспортних накладних (ТТН) успішно вирішено через застосування бібліотеки координатного рендерингу PDFKit та механізмів буферизації.

Процес генерації документа розпочинається при виклику відповідного мережевого маршруту, після чого контролер ініціює попереднє завантаження зв'язаних даних за допомогою багатотабличного запиту (JOIN). Це дозволяє агрегувати детальну інформацію про замовника, підприємство, водія та вантаж у єдиному масиві. Далі в оперативній пам'яті сервера створюється віртуальний об'єкт документа у форматі A4, для якого оголошуються координатні сітки. Сервер по чергово форматує юридичні реквізити сторін, використовуючи лінійне зміщення шрифтів, а специфікація вантажу обробляється ітераційним циклом із постійною перевіркою висоти сторінки для автоматичного додавання нових аркушів. Важливою архітектурною особливістю є те, що після завершення рендерингу згенерований бінарний код не зберігається на жорсткий диск сервера як фізичний файл. Натомість середовище Node.js перехоплює цей об'єкт через інтерфейс потоків і безпосередньо передає байт-код у HTTP-відповідь клієнтському додатку, де він миттєво конвертується та відкривається для друку.

Реалізовані логічні механізми виключають накопичення тимчасових системних файлів і скорочують час на обробку рутинних транспортних задач обривком до 2-4 секунд.

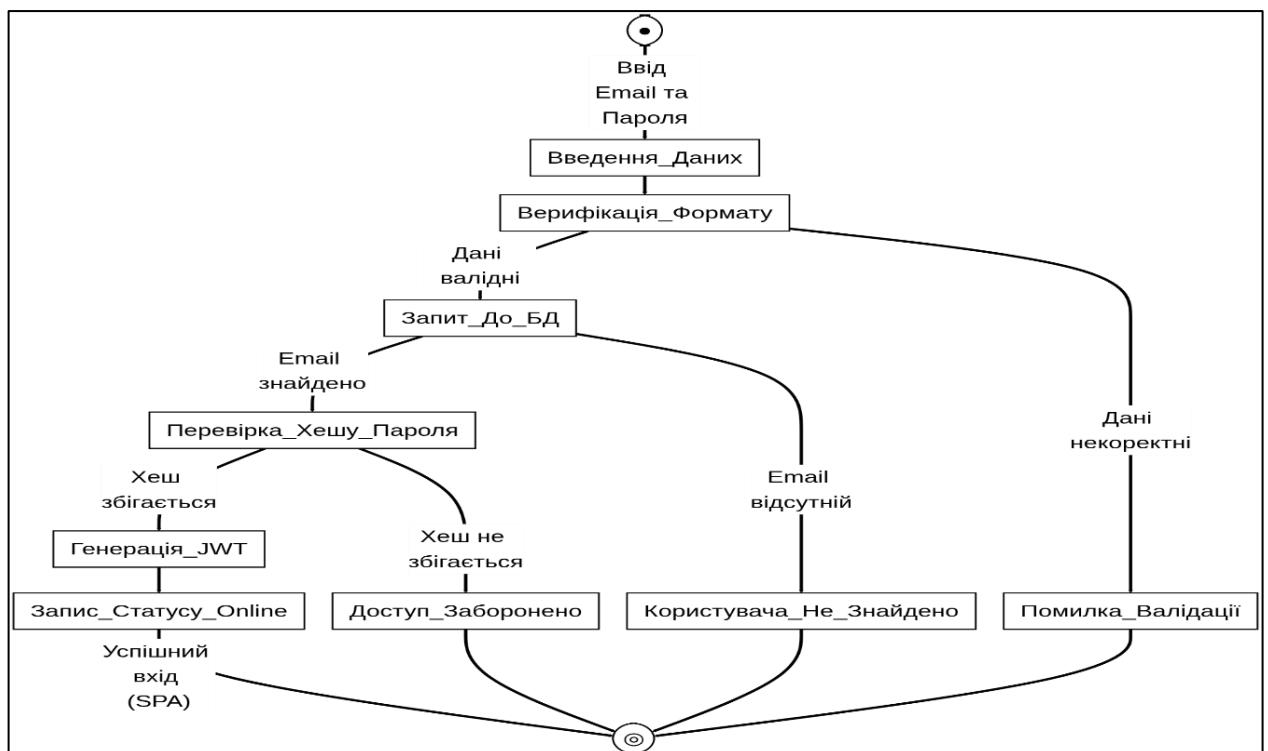


Рис. 2.4. Діаграма процесу входу в систему

Для візуалізації динамічних моделей процесів у системі було розроблено відповідні Activity та Sequence діаграми. Зокрема, діаграма станів (діяльності) процесу автентифікації користувача (рис. 2.4) детально описує логічний шлях від моменту вводу облікових даних до генерації ключа сесії. Своєю чергою, діаграма послідовності процедури реєстрації працівника у системі (рис. 2.5) відображає транзакційну взаємодію між клієнтським додатком, серверним маршрутизатором та базою даних. Процес ініціюється виключно користувачем із роллю COMPANY_ADMIN. Адміністратор надсилає запит із даними нового співробітника, після чого сервер перевіряє унікальність електронної пошти в базі даних. Якщо адреса вільна, контролер запускає алгоритм хешування пароля за допомогою бібліотеки bcrypt, зберігає нового користувача в реляційну таблицю з відповідним ідентифікатором компанії та повертає на клієнт повідомлення про успішне створення профілю.

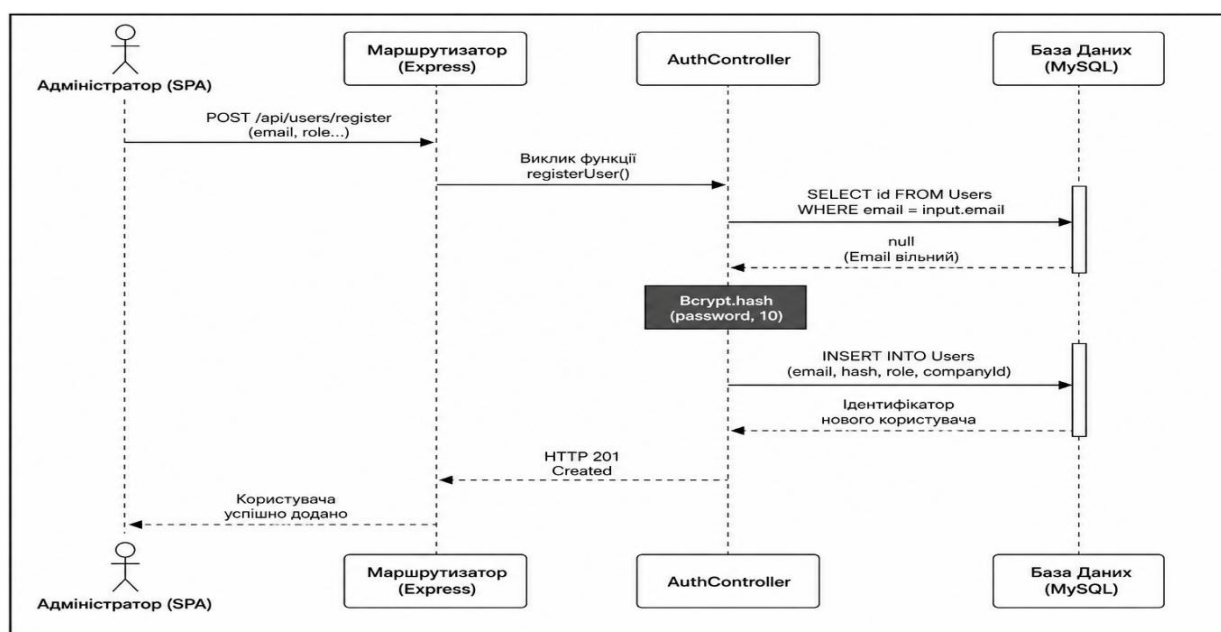


Рис. 2.5. Діаграма послідовності алгоритму реєстрації працівника

Діаграма діяльності створення замовлення диспетчером (рис. 2.6) ілюструє покроковий алгоритм обробки логістичної заявки. Процес починається із заповнення диспетчером екранної форми, після чого система виконує первинну валідацію: перевіряє наявність усіх обов'язкових параметрів (адреси, вага, ціна). Блок-схема чітко відображає вузли прийняття рішень - якщо дані невалідні,

користувач отримує помилку і повертається до редагування. У разі успіху формується масив даних, який серіалізується у формат JSON та передається на сервер для запису. Завершальним етапом алгоритму є збереження даних, та автоматичне створення запису про цю подію в системному журналі аудиту.

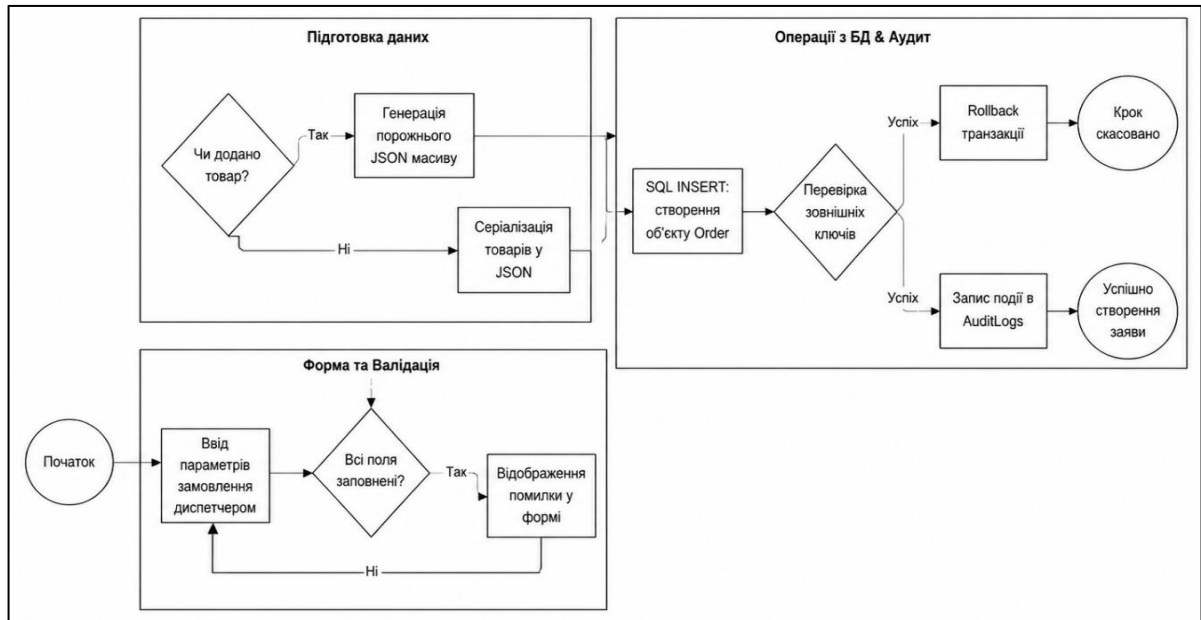


Рис. 2.6. UML Діаграма діяльності процесу створення замовлення

Блок-схема відображає точки прийняття рішень щодо валідності полів (вузли розгалуження).

Комплексне застосування UML-моделювання сформувало інженерну основу для якісного написання програмного коду серверної та клієнтської частин платформи.

2.4. Реалізація функціоналу вебплатформи.

Важливим етапом створення вебплатформи стала організація процесу розробки, яка здійснювалася під управлінням системи контролю версій Git з використанням платформ GitHub та GitLab. Для ефективної роботи було впроваджено стратегію розгалуження (Branching Strategy), де гілка "develop" слугувала для активної розробки та інтеграції нових функцій, а гілка "main" - для стабільних релізів. Використання Git забезпечило контрольоване збереження логічних частин коду у вигляді комітів, можливість повернення до стабільного стану та розмежування паралельних завдань без ризику пошкодження основної

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

структури системи [27]. Крім того, для проєкту було налаштовано CI/CD pipeline, що забезпечив автоматичну перевірку коду та його безперервне розгортання.

Безпосереднє написання програмного коду здійснювалося у спеціалізованому текстовому редакторі Visual Studio Code. Для прототипування алгоритмів прикладного інтерфейсу та попередньої перевірки форматів обміну даними активно використовувалася програма Postman. Вона дала змогу тестувати мережеві запити та відповіді сервера абсолютно незалежно від поточного стану готовності клієнтського інтерфейсу [28].

Водночас для глибшого розуміння внутрішньої архітектури програмного забезпечення було застосовано інструментарій візуального моделювання UML. Моделювання виконано на рівні класів, станів та послідовностей, що дозволило детально відобразити як статичні, так і динамічні аспекти застосунку. Фундаментальною основою об'єктно-орієнтованого підходу в цьому проєкті стала діаграма класів, яка комплексно описує структуру системи, зокрема самі класи, їхні атрибути, методи та взаємозв'язки. Загальна структура класів розробленої системи (рис. 2.7) наочно демонструє моделі даних, що обробляються на сервері через механізми об'єктно-реляційного відображення ORM Sequelize.

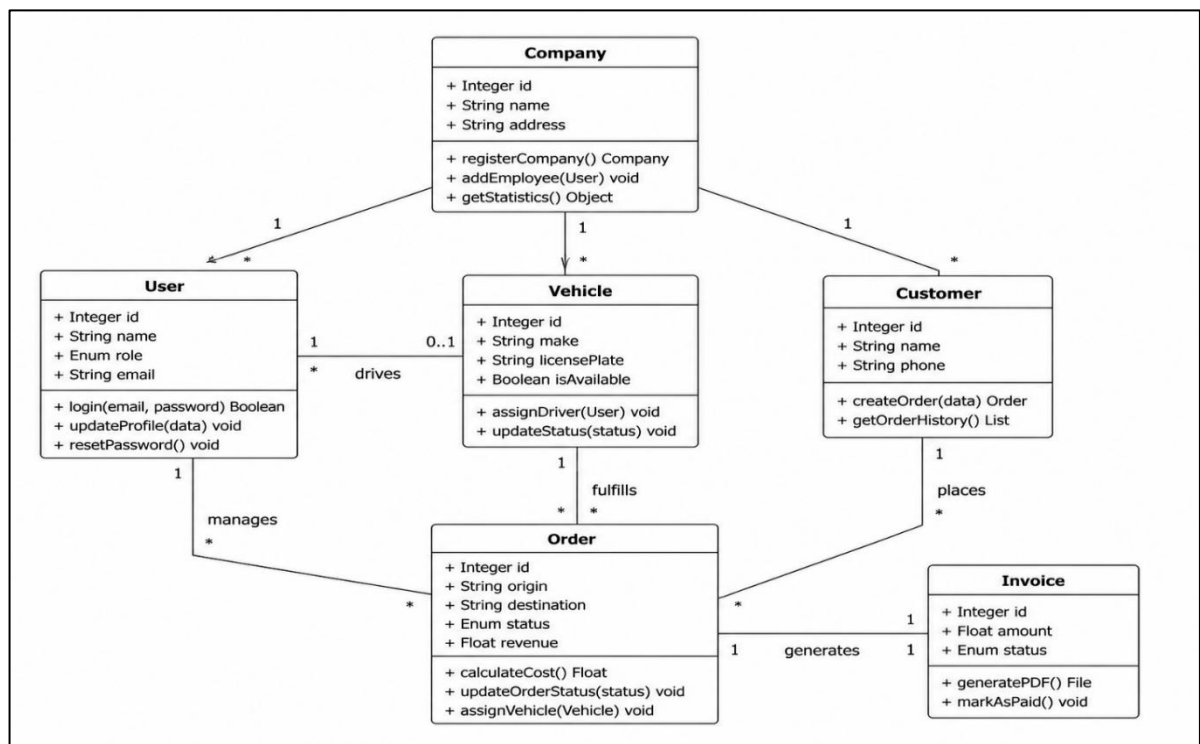


Рис. 2.7. Діаграма класів інформаційної системи

Для детального опису архітектури даних інформаційної системи нижче наведено опис шести основних класів, представлених на діаграмі (див. рис. 2.7), їх атрибути та зв'язки з іншими компонентами системи, які зведено у табл. 2.17.

Таблиця 2.17.

Опис основних класів моделі даних інформаційної системи

Назва класу	Призначення	Основні атрибути (поля)	Зв'язки з іншими класами
User	Облікові записи працівників (суперадміністратора, диспетчер, водій тощо)	id (PK), email, password, fullName, role, status, companyId (FK), isOnline, last Seen	1:1 з Vehicle (водій) 1:Багато х з Order (менеджер/водій) Багато:1 з Company
Company	Моделювання логістичної компанії (орендаря платформи)	id (PK), name, uniqueId, address, phone, email, isActive, isSystem	1:Багато х з User, Vehicle, Order, Customer, Invoice
Customer	Контрагенти (замовники вантажоперевезень)	id (PK), name, edrpou, address, phone, email, companyId (FK)	1:Багато х з Order, Invoice Багато:1 з Company
Vehicle	Автомобілі автопарку компанії	id (PK), brand, model, plateNumber, capacity, isAvailable, driverId (FK), companyId (FK)	Багато:1 з User (водій), Company 1:Багато х з Order
Order	Центральний клас, що описує перевезення вантажу (рейс)	id (PK), description, weight, pickupAddress, deliveryAddress, status, price, distance, creatorId (FK), vehicleId (FK)	Багато:1 з Customer, User, Vehicle 1:1 з Invoice
Invoice	Фінансові рахунки, виставлені клієнтам за виконані рейси	id (PK), orderId (FK), creatorId (FK), amount, status	1:1 з Order Багато:1 з Customer, Company

Варто зазначити, що окрім цих базових класів, на рівні бази даних та ORM реалізовано також ряд допоміжних сутностей (таких як AuditLog для ведення історії операцій, Notification для сповіщень та OrderExpense для фіксації витрат).

Для ілюстрації практичної реалізації розроблених класів даних та зв'язків між ними за допомогою ORM Sequelize нижче наведено фрагменти коду визначення ключових моделей бази даних. Щоб забезпечити чистоту архітектури, модель користувача розбито на базові атрибути та параметри безпеки.

Лістинг 2.1. Визначення базових атрибутів моделі User

```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/db');

const User = sequelize.define('User', {
  id: { type: DataTypes.INTEGER, autoIncrement: true, primaryKey: true },
  email: { type: DataTypes.STRING, allowNull: false, unique: true },
  password: { type: DataTypes.STRING, allowNull: false },
  fullName: { type: DataTypes.STRING, allowNull: false },
  role: {
    type: DataTypes.ENUM('SUPER_ADMIN', 'COMPANY_ADMIN', 'DISPATCHER',
'DRIVER'),
    defaultValue: 'DRIVER'
  },
  companyId: {
    type: DataTypes.INTEGER, allowNull: true,
    references: { model: 'Companies', key: 'id' }
  }
});
```

Окрім базових даних для ідентифікації, сутність користувача містить розширений набір атрибутів, що відповідають за моніторинг активності та параметри двофакторної автентифікації. Ці поля ініціалізуються значеннями за замовчуванням при створенні профілю.

Лістинг 2.2. Атрибути безпеки та активності моделі User

```
// Продовження конфігурації моделі User
status: { type: DataTypes.ENUM('PENDING', 'APPROVED'), defaultValue: 'PENDING' },
isOnline: { type: DataTypes.BOOLEAN, defaultValue: false },
lastSeen: { type: DataTypes.DATE, defaultValue: DataTypes.NOW },
twoFactorEnabled: { type: DataTypes.BOOLEAN, defaultValue: false },
twoFactorCode: { type: DataTypes.STRING, allowNull: true },
twoFactorCodeExpires: { type: DataTypes.BIGINT, allowNull: true }
});
module.exports = User;
```

Визначальним етапом роботи з ORM є налаштування реляційних зв'язків між моделями. Конфігураційний файл `models/index.js` збирає всі сутності та встановлює обмеження зовнішніх ключів. Розглянемо частину коду, яка відповідає за об'єднання логіки компаній, автопарку та користувачів.

Лістинг 2.3. Ініціалізація зв'язків для компаній, автопарку та штату

```
const Vehicle = require('./Vehicle');
const Company = require('./Company');
const User = require('./User');
```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

```
// Зв'язки автопарку та водіїв
Vehicle.belongsTo(User, { as: 'driver', foreignKey: 'driverId' });
User.hasOne(Vehicle, { foreignKey: 'driverId' });

// Зв'язки корпоративної ієрархії
Company.hasMany(User, { foreignKey: 'companyId', as: 'users' });
User.belongsTo(Company, { as: 'company', foreignKey: 'companyId' });

Company.hasMany(Vehicle, { foreignKey: 'companyId' });
Vehicle.belongsTo(Company, { as: 'company', foreignKey: 'companyId' });
```

Для правильного функціонування логістичного процесу критично важливо пов'язати сутності замовлень (Orders) із клієнтами, фінансовими рахунками та виконавцями. Завдяки цим асоціаціям сервер може виконувати складні запити з агрегацією даних. Це мінімізує кількість окремих звернень до бази даних, суттєво знижує навантаження на підсистему вводу-виводу сервера та забезпечує миттєве оновлення оперативної інформації в інтерфейсі користувача.

Лістинг 2.4. Встановлення реляційних зв'язків логістичних замовлень

```
const Order = require('./Order');
const Invoice = require('./Invoice');
const Customer = require('./Customer');

// Зв'язки замовлень із клієнтами та персоналом
Customer.hasMany(Order, { foreignKey: 'customerId' });
Order.belongsTo(Customer, { as: 'customer', foreignKey: 'customerId' });

User.hasMany(Order, { foreignKey: 'driverId', as: 'drivenOrders' });
Order.belongsTo(User, { as: 'driver', foreignKey: 'driverId' });

// Фінансова аналітика: зв'язок замовлень із рахунками
Invoice.belongsTo(Order, { as: 'order', foreignKey: 'orderId' });
Order.hasOne(Invoice, { as: 'invoice', foreignKey: 'orderId' });
```

Наведена структура реляційних зв'язків, реалізована засобами ORM Sequelize, повністю відображає концептуальну модель бази даних. Такий підхід дозволяє автоматизувати складні багатотабличні запити (JOIN) під час агрегації даних для генерації товарно-транспортних накладних та формування фінансової звітності. Крім того, налаштовані зв'язки гарантують каскадне збереження цілісності даних при будь-яких змінах в організаційній структурі компаній-перевізників.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до другого розділу

У рамках створення другого розділу розроблено вичерпний інженерний підхід до проєктування вебсистеми автоматизації логістики. Розподіл параметрів на структуровані функціональні та нефункціональні вимоги дозволив формалізувати межі відповідальності багатокомпонентної архітектури застосунку і спростити вибір технологічного стеку.

Моделювання варіантів використання підтвердило обґрунтованість впровадження рольового поділу (RBAC), розмежувавши функціональні допуски диспетчера та мобільного водія у рамках закритого віртуального простору обраної компанії. Виконано імплементацію реляційної системи баз даних. Опис сутностей підтвердив дотримання нормалізації моделі (Третя нормальна форма) і забезпечив технічне уникнення надлишковості транзакційних даних за допомогою жорстких зв'язків.

Розроблено та оптимізовано головні сервісні алгоритми роботи API. Авторизація за моделлю Stateless JWT мінімізувала час реакції сервера та зменшила навантаження на таблиці маршрутизації бази даних. Побудована алгоритмічна послідовність формування документів (Invoices, PDF) повністю звільняє файлову систему від зберігання залишкових тимчасових даних, забезпечуючи передачу динамічно згенерованого бінарного коду в режимі реального часу. На підставі сформованих архітектурних вимог, діаграм і структур імплементується реалізація користувацького інтерфейсу і клієнтської частини застосунку, що складає предмет дослідження третього розділу.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБПЛАТФОРМИ АВТОМАТИЗАЦІЇ ЛОГІСТИЧНИХ ПРОЦЕСІВ

3.1. Порядок налаштування параметрів та розгортання системи

Розроблена платформа автоматизації логістичних процесів вантажних перевезень «LogisticSoft» спроектована за клієнт-серверною архітектурою з використанням сучасного стеку технологій на базі JavaScript. Для успішного запуску, функціонування та інтеграції системи в IT-інфраструктуру підприємства необхідно виконати послідовність дій з конфігурування серверного та клієнтського середовищ розгортання [31]. Процес встановлення орієнтований на стандартні операційні системи (Windows, Linux, macOS) та потребує мінімального набору системних інструментів розробника. Для розгортання серверної та клієнтської частин на локальній робочій станції або виділеному сервері в першу чергу має бути встановлене середовище виконання Node.js версії v22.x або вище, яке містить вбудований менеджер пакетів npm для керування зовнішніми залежностями. В якості сховища даних необхідно інстальювати реляційну систему управління базами даних MySQL версії 8.x. Крім того, для зручного отримання копії вихідного коду з віддаленого репозиторію система вимагає наявності встановленого клієнта контролю версій Git.

Процес розгортання розпочинається з отримання копії вихідного коду. Для цього виконується завантаження проекту з репозиторію за допомогою командного рядка:

```
git clone https://github.com/LevrikM/logisticsoft.git  
cd logisticsoft
```

Проект структуровано на дві ізольовані директорії: server (серверна частина на базі Node.js та Express) та client (клієнтська частина на базі React.js). Налаштування параметрів системи починається з конфігурування сервера. Для цього у директорії server необхідно створити файл налаштувань середовища .env на основі наявного шаблону .env.example. У цьому файлі вказуються параметри доступу до бази даних, секретний ключ для підпису сесійних токенів авторизації

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

JWT та реквізити SMTP-сервера електронної пошти для автоматичного розсилання повідомлень користувачам.

Для кращої систематизації параметрів налаштування середовища розроблено структуру конфігураційного файлу, яку зведено у табл. 3.1.

Таблиця 3.1.

Специфікація параметрів конфігураційного файлу сервера (.env)

Назва змінної	Тип даних	Опис та призначення параметра
DB_HOST	Рядок	Адреса хоста сервера бази даних MySQL
DB_USER	Рядок	Ім'я користувача для підключення до СКБД
DB_PASSWORD	Рядок	Пароль доступу користувача до СКБД
DB_NAME	Рядок	Назва цільової схеми бази даних (logisticsoft)
PORT	Число	Порт, на якому запускається серверний процес
JWT_SECRET	Рядок	Секретний ключ шифрування підпису JWT токенів
SMTP_HOST	Рядок	Адреса поштового сервера SMTP
SMTP_PORT	Число	Мережевий порт поштового сервера SMTP
SMTP_USER	Рядок	Електронна адреса відправника системних листів
SMTP_PASS	Рядок	Пароль додатку для авторизації на SMTP-сервері

Після налаштування конфігураційного файлу необхідно завантажити та встановити всі сторонні пакети та бібліотеки, зазначені у файлі маніфесту package.json. Встановлення залежностей виконується за допомогою вбудованої у

Node.js утиліти npm:

```
cd server
```

```
npm install
```

Наступним кроком є підготовка бази даних MySQL. Користувачеві необхідно створити порожню базу даних з назвою, що відповідає значенню змінної *DB_NAME* у файлі конфігурації. Для ініціалізації структури бази даних, створення всіх необхідних одинадцяти таблиць, зв'язків між ними, а також наповнення довідників початковими даними в проєкті передбачено скрипт автоматичного заповнення *seed.js*. Цей скрипт запускає вбудовану синхронізацію Sequelize, очищає попередні таблиці (якщо вони існували), створює системну компанію та головного адміністратора системи (*SUPER_ADMIN*) з обліковими даними за замовчуванням (email: *logistic0soft@gmail.com*, пароль: *LogSoftP@2026*). Запуск ініціалізації виконується наступним чином:

```
node seed.js
```

Після успішного завершення очищення та первинного заповнення бази даних можна запускати безпосередній процес виконання вебсервера:

```
node app.js
```

Сервер ініціалізує WebSocket-з'єднання через бібліотеку Socket.io, перевіряє аутентифікацію підключення до MySQL та починає прослуховування вказаного порту (<http://localhost:5000>).

Налаштування клієнтської частини виконується аналогічно. Необхідно перейти у каталог *client*, встановити залежності фронтенд-додатку React.js та запустити сервер розробки, який автоматично відкриє вікно браузера з адресою клієнта (<http://localhost:3000>):

```
cd ../client
```

```
npm install
```

```
npm start
```

Під час розгортання системи в реальному виробничому середовищі (Production) з метою забезпечення стійкості, високої швидкості доступу та безпеки передачі даних за протоколом HTTPS вебплатформу було успішно розгорнуто на хмарному

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

сервері з використанням доменного імені <https://logisticsoft.pp.ua/>. Для цього було налаштовано проксі-сервер Nginx, виконано збірку клієнтського бандлу через команду `npm run build` та налаштовано менеджер процесів PM2 для автоматичного перезапуску сервера Node.js у разі виникнення помилок чи перезавантаження операційної системи (рис. 3.1).

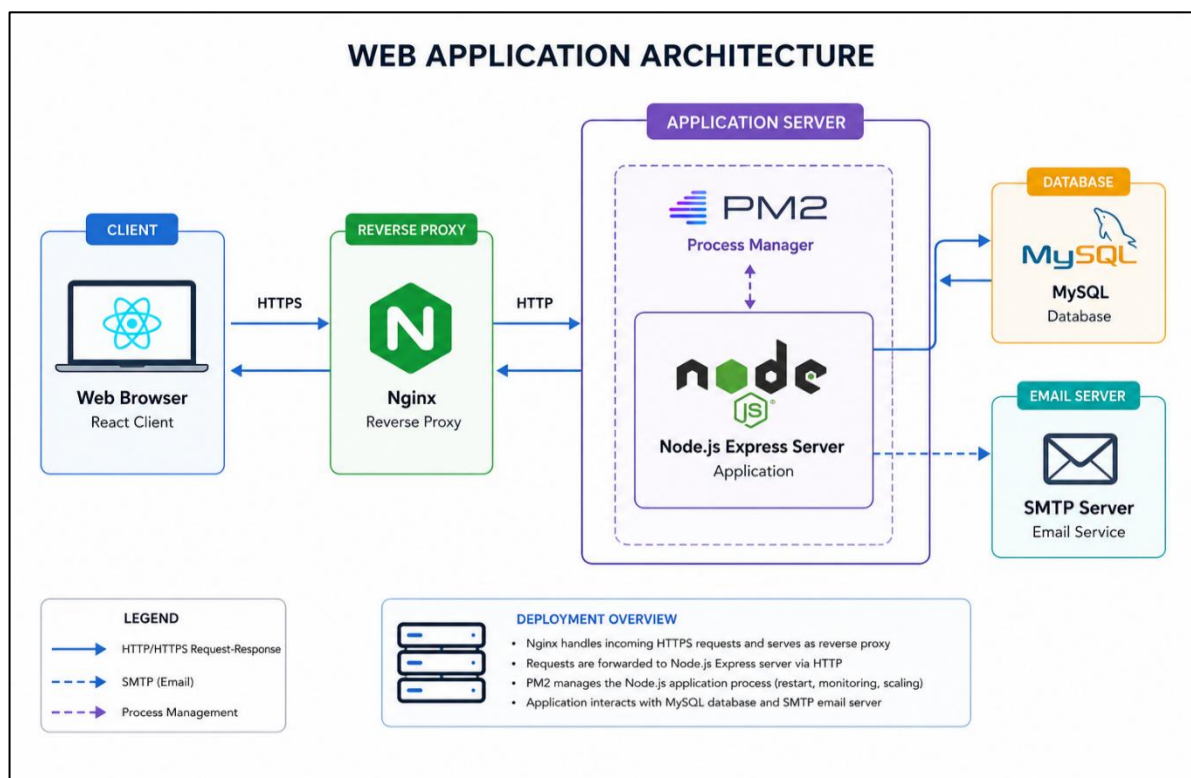


Рис. 3.1. Схема архітектури розгортання та взаємодії компонентів

3.2. Алгоритм використання інтерфейсу користувача

Взаємодія користувачів із логістичною платформою здійснюється через сучасний адаптивний вебінтерфейс.

Початковою точкою взаємодії для нових користувачів є публічна головна сторінка системи. Ця сторінка презентує основні можливості платформи (моніторинг перевезень, фінансову аналітику, керування автопарком і водіями, внутрішній чат для зв'язку) за допомогою динамічного інтерактивного слайдера. З головної сторінки здійснюється перехід до авторизації, реєстрації або до форми надсилання заявки на реєстрацію нової логістичної компанії [8]. Для зручності користувачів на сторінці інтегровано функціонал швидкої зміни колірної теми (світлої та темної) (рис. 3.2).

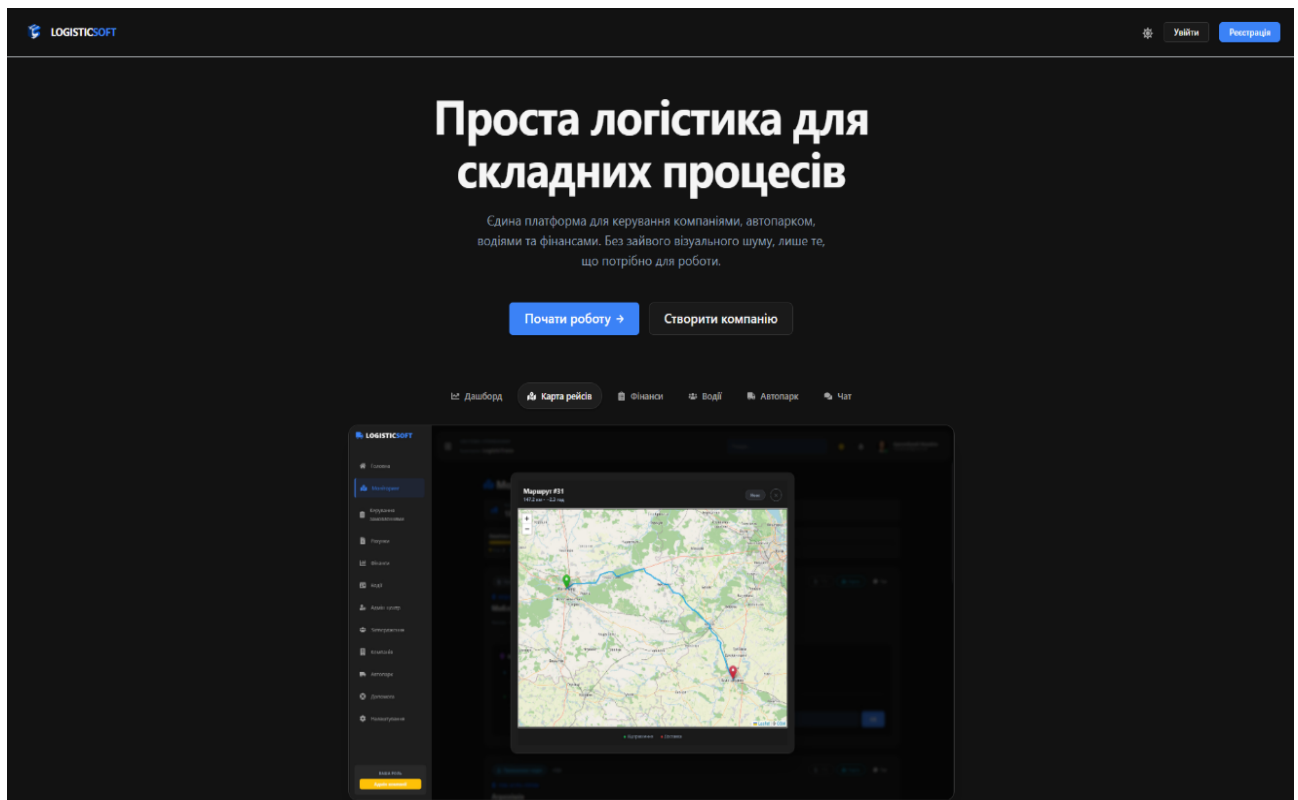


Рис. 3.2. Головна публічна сторінка системи

Авторизовані користувачі проходять єдину точку входу. Вся логіка побудована на рольовій моделі доступу (RBAC), яка ізолює права перегляду та редагування даних залежно від посади співробітника та приналежності до конкретної компанії. Кожен користувач вводить пару логін/пароль (рис. 3.3), після чого система генерує JWT-токен, що зберігається в локальному сховищі браузера та супроводжує всі наступні HTTP-запити до REST API [32].

Для забезпечення підвищеного рівня інформаційної безпеки та захисту облікових записів користувачів у системі передбачено можливість підключення двофакторної автентифікації (2FA). Якщо цю функцію активовано користувачем у профілі, то під час авторизації сервер автоматично генерує тимчасовий шестизначний цифровий код доступу та надсилає його на підтверджену електронну пошту користувача через SMTP-сервер. Цей код є дійсним протягом 5 хвилин. При кожній наступній спробі входу, окрім стандартних облікових даних, користувач обов'язково має ввести надісланий на пошту перевірочний код для підтвердження сесії (рис. 3.4).

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

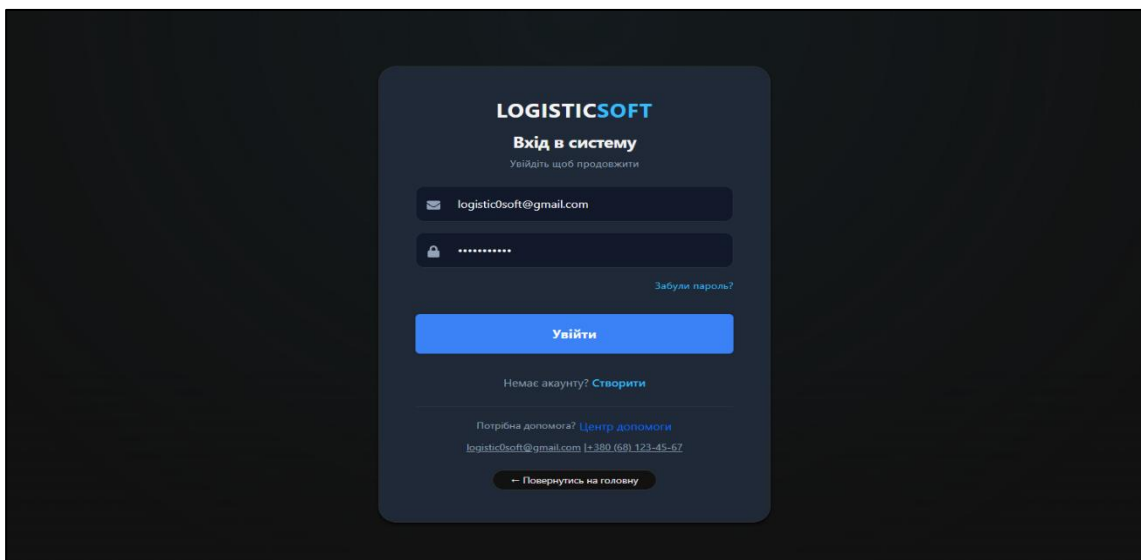


Рис. 3.3. Сторінка входу та авторизації користувача

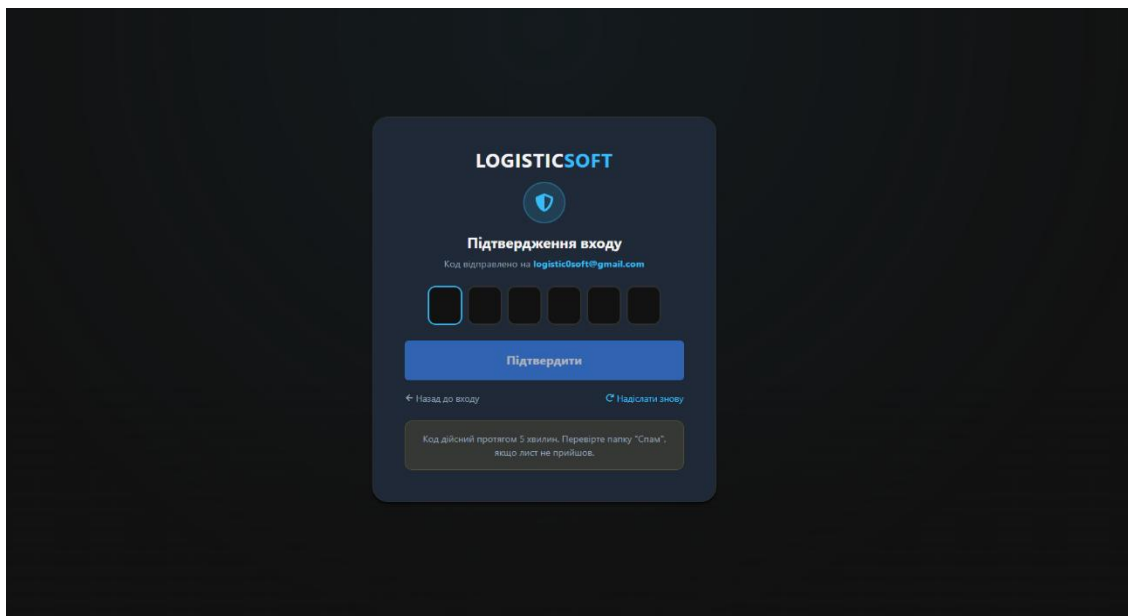


Рис. 3.4. Вікно введення коду двофакторної автентифікації, отриманого на пошту

Після успішного входу користувач автоматично перенаправляється на головну сторінку особистого кабінету, яка виконує роль роль-орієнтованого навігаційного хабу. Замість статичного меню, цей інтерфейс динамічно адаптується під повноваження поточного користувача. На екрані відображається персоналізоване вітання із зазначенням імені користувача та назви логістичної компанії. Основну частину екрана займає інтерактивна сітка карток «Швидкі дії». Для адміністратора компанії тут відображаються переходи до моніторингу рейсів, керування замовленнями, автопарком, адмінцентру, затвердження нових

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

користувачів, виставлення рахунків та налаштувань. Кожна картка реалізована у вигляді інтерактивного елемента зі звуковим відгуком при взаємодії.

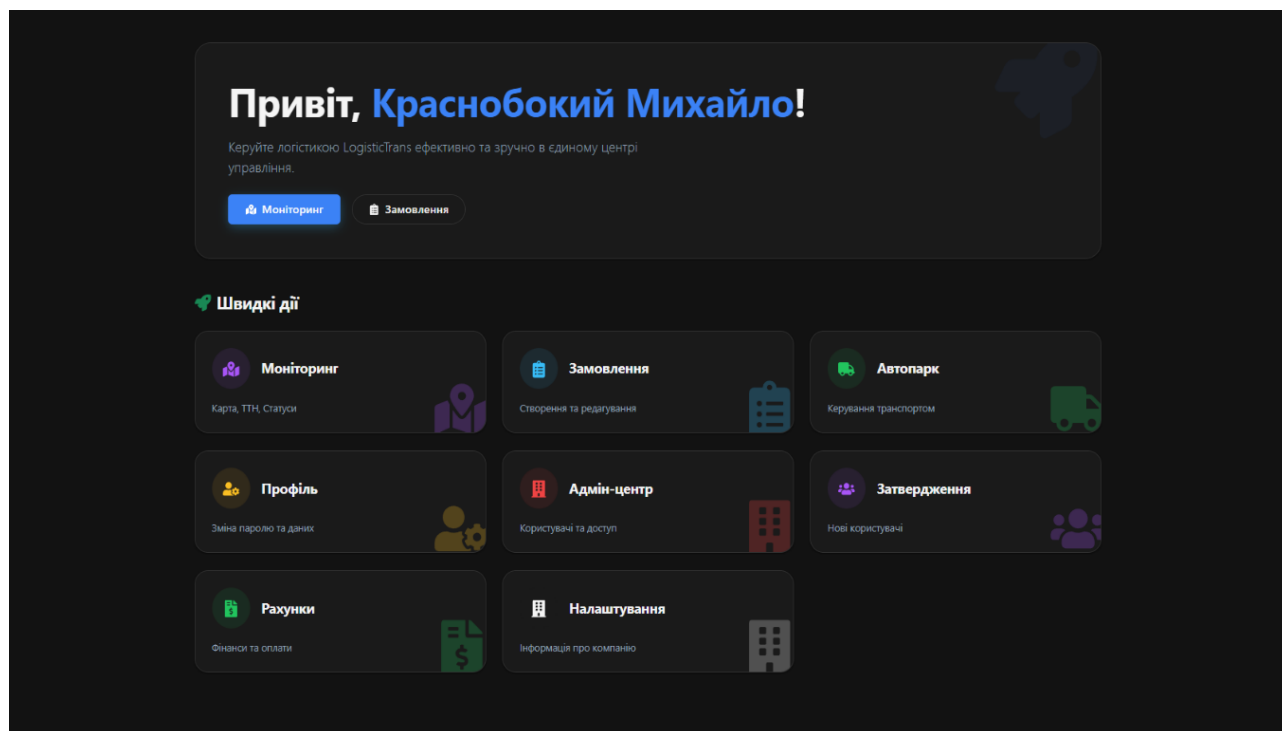


Рис. 3.5. Роль-орієнтований головний екран після входу в систему

Алгоритм використання вебінтерфейсу системи відрізняється для кожної з чотирьох ролей користувачів, що забезпечує чітке розділення обов'язків у процесі транспортування вантажів.

Перша роль - Суперадміністратор (SUPER_ADMIN). Його робоче місце представлене глобальним центром системи. На відміну від інших ролей, які обмежені даними лише своєї компанії, суперадміністратор має повний доступ до перегляду та керування всіма користувачами, транспортними засобами, клієнтами та замовленнями в межах усієї платформи. Окрім глобального моніторингу замовлень і користувачів компаній, головна панель суперадміністратора містить модуль обробки заявок на реєстрацію нових логістичних компаній (рис. 3.6).

Коли новий перевізник заповнює публічну форму реєстрації, суперадміністратор перевіряє правильність даних, код ЄДРПОУ та підтверджує або відхиляє запит. При підтвердженні автоматично створюється кабінет компанії, а також обліковий запис її адміністратора зі статусом APPROVED, що дає йому

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

можливість одразу авторизуватися в системі з вказаними під час подачі заявки обліковими даними.

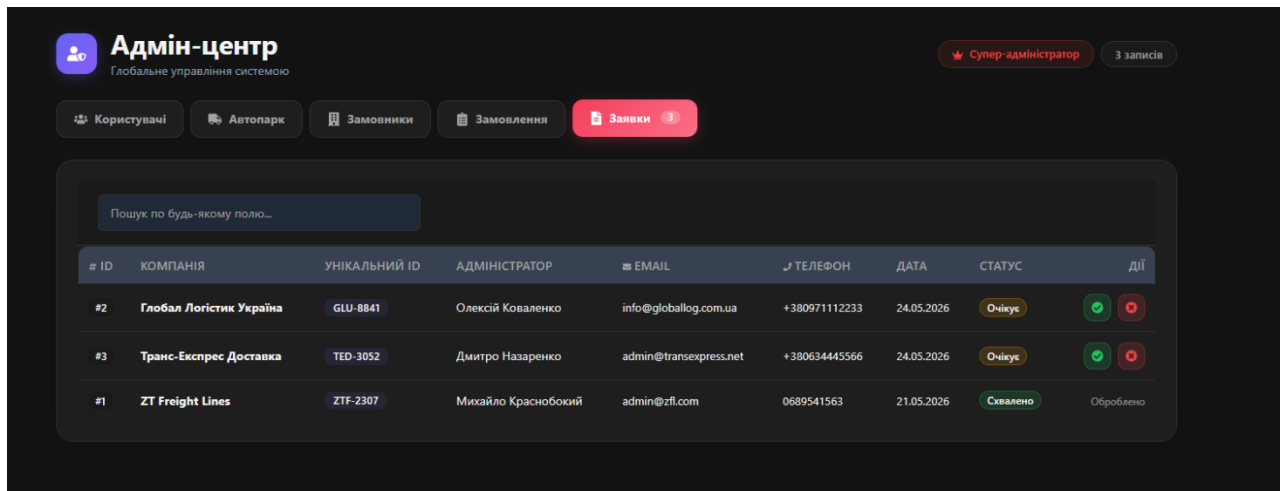


Рис. 3.6. Інтерфейс суперадміністратора системи із журналом заявок компаній

Друга роль - Адміністратор компанії (COMPANY_ADMIN). Після схвалення заявки адміністратор здійснює вхід до системи та виконує початкове наповнення кабінету своєї фірми. Основним інструментом адміністрування штату компанії є вкладка «Користувачі» в Адмінцентрі. Вона відображає інтерактивну таблицю всіх співробітників компанії з відображенням їх аватарів, ПІБ, контактних email, призначених ролей (водій, диспетчер, адмін) та поточного статусу в мережі (індикатор Online/Offline з виведенням часу останнього візиту). Адміністратор компанії має повні права на редагування профілів користувачів, зміну їх ролей, видалення або тимчасове зняття затвердження профілю для обмеження доступу.

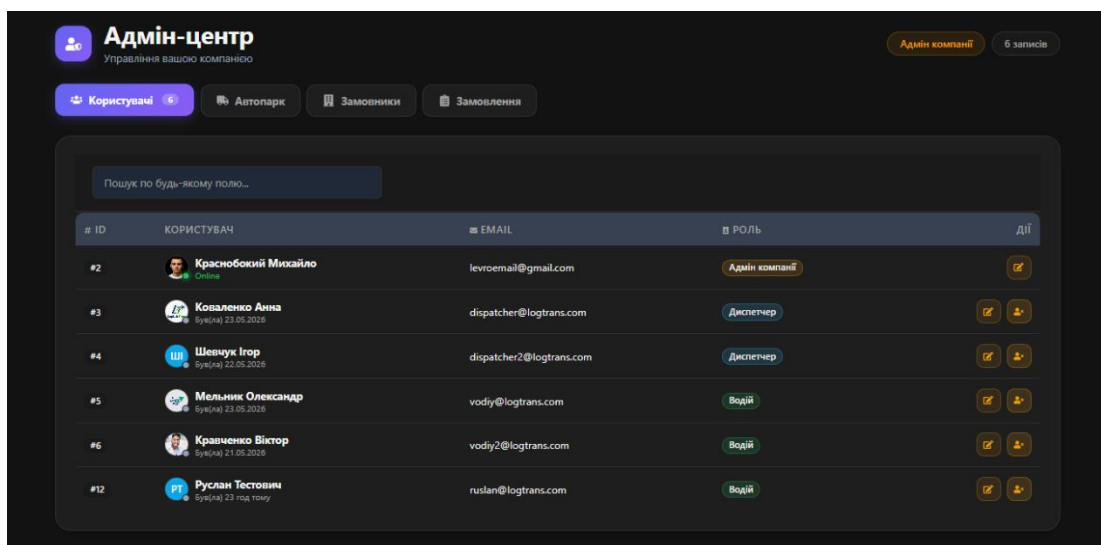


Рис. 3.7. Вкладка «Користувачі» в Адмінцентрі для керування штатом компанії

Окрім адміністрування загального штату, в системі для адміністратора компанії та диспетчера реалізовано спільну сторінку «Водії». Ця сторінка виступає важливим інструментом оперативного моніторингу водійського складу. Кожна картка водія відображає його аватар, ПІБ, email, статус у мережі (Online/Offline) та прикріплений автомобіль із його державним номером. Важливою особливістю є візуалізація поточного статусу зайнятості водія: якщо за водієм немає активного рейсу, відображається статус «Вільний» зеленим кольором, а якщо рейс активний - відображається плашка поточного статусу вантажоперевезення із зазначенням номера замовлення, опису вантажу та точного маршруту доставки (адреса завантаження та вивантаження). Це дозволяє адміністратору та диспетчеру миттєво оцінити зайнятість штату без необхідності переходу в інші розділи (рис. 3.8).

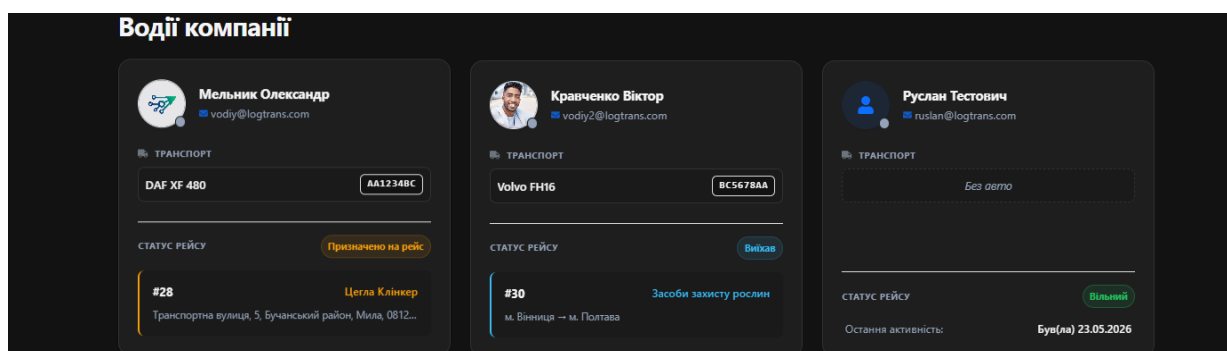


Рис. 3.8. Сторінка «Водії» для моніторингу статусів водійського складу

У формі додавання автомобіля вказуються марка, реєстраційний державний номер, вантажопідйомність у тоннах та виконується прив'язка до конкретного водія зі штату. Усі створені об'єкти автоматично маркуються унікальним ідентифікатором компанії companyId, що унеможлиблює доступ до них користувачів інших організацій (рис. 3.9)

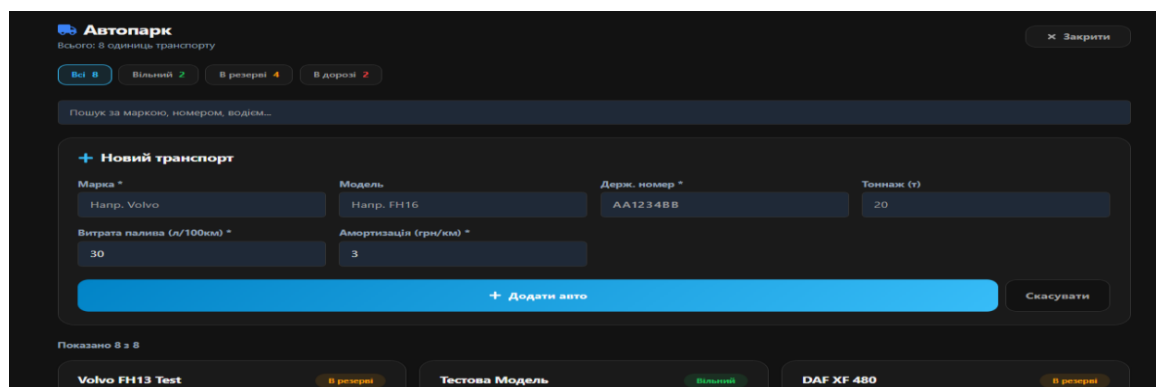


Рис. 3.9. Форма реєстрації нового транспортного засобу адміністратором компанії

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

Окрім адміністрування штату та автопарку, у кабінеті адміністратора компанії реалізовано повноцінний модуль фінансового аналізу та обліку - сторінка «Фінанси». Цей модуль інтегрує бібліотеку інтерактивних графіків Recharts для наочної візуалізації фінансових потоків. Адміністратор має можливість переглядати аналітику за обраний період (місяць, квартал, рік або за весь час). На сторінці відображаються чотири інформаційні блоки з ключовими показниками: загальний дохід компанії, загальна сума витрат, чистий прибуток (розраховується автоматично) та кількість завершених рейсів.

Для детального аналізу в модуль інтегровано два типи діаграм: стовпчикову діаграму для порівняння доходу, витрат і прибутку за місяцями та кругову діаграму для аналізу структури витрат компанії, розділену за категоріями: витрати на паливо, виплата заробітної плати водіям, амортизація транспортних засобів, оплата платних доріг, технічне обслуговування, страхування та інші витрати. Нижня частина сторінки містить інтерактивну таблицю прибутковості по водіях, де деталізується персональна ефективність кожного працівника: кількість рейсів, сума доходу, обсяг витрат, фінальний прибуток та загальний пробіг у кілометрах (рис.3.10).

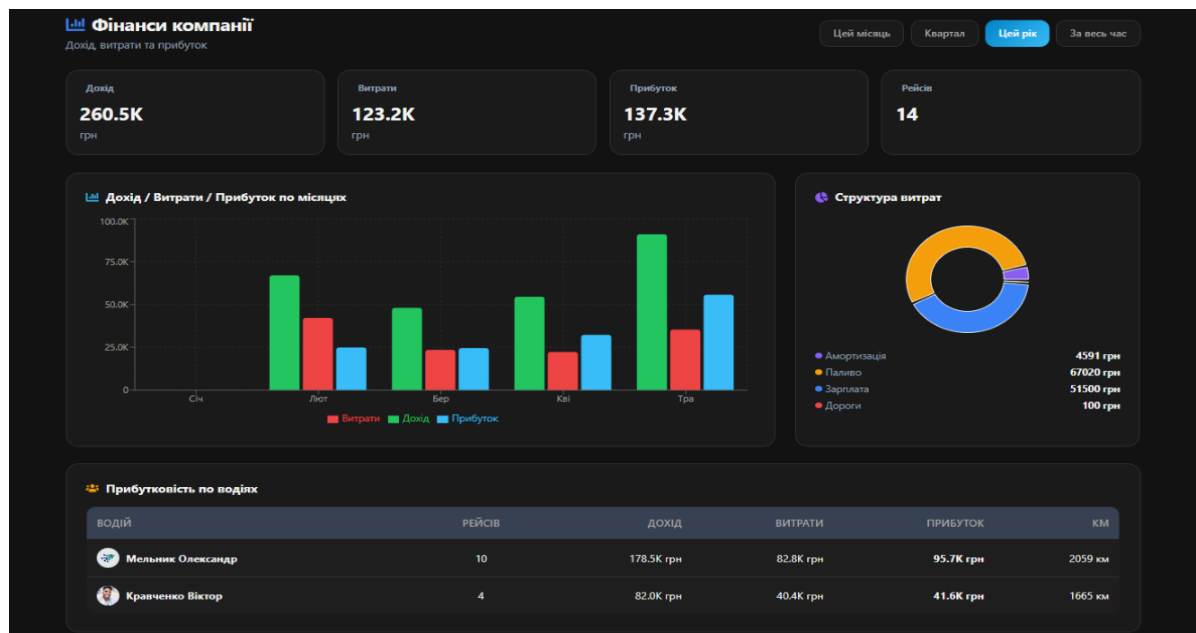


Рис. 3.10. Інтерфейс фінансового аналізу та статистики компанії

Третя роль - Диспетчер (DISPATCHER). Диспетчер виконує основну щоденну роботу з логістичного планування, моніторингу рейсів та обробки

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

замовлень компанії. Робоче місце диспетчера складається з двох основних розділів: сторінки моніторингу активних рейсів та сторінки керування всіма замовленнями. На сторінці «Моніторинг» диспетчер може контролювати виконання поточних рейсів у реальному часі. Диспетчер може переглянути інформацію про призначений рейс, статус водія та зв'язатися з ним через чат (рис. 3.11).

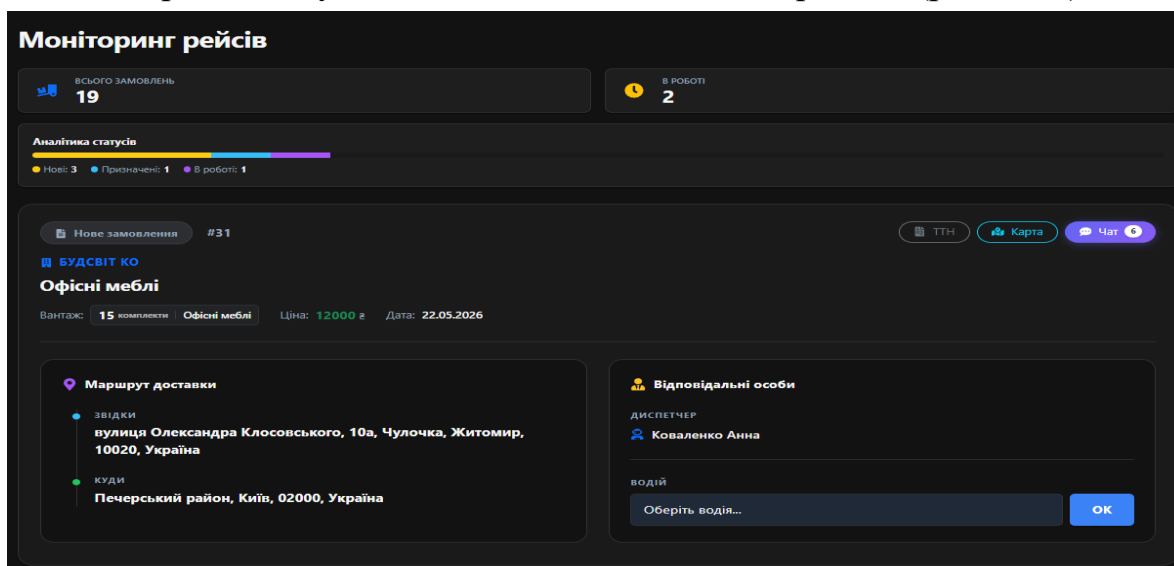


Рис. 3.11. Сторінка моніторингу активних замовлень диспетчера

Другою важливою складовою інтерфейсу є сторінка «Керування замовленнями». Тут відображається загальний список усіх замовлень компанії за весь період (включаючи завершені, активні, скасовані та нові) з детальною таблицею, пошуком і можливістю фільтрації. З цієї сторінки диспетчер ініціює створення нових замовлень, редагування витрат тощо (рис. 3.12).

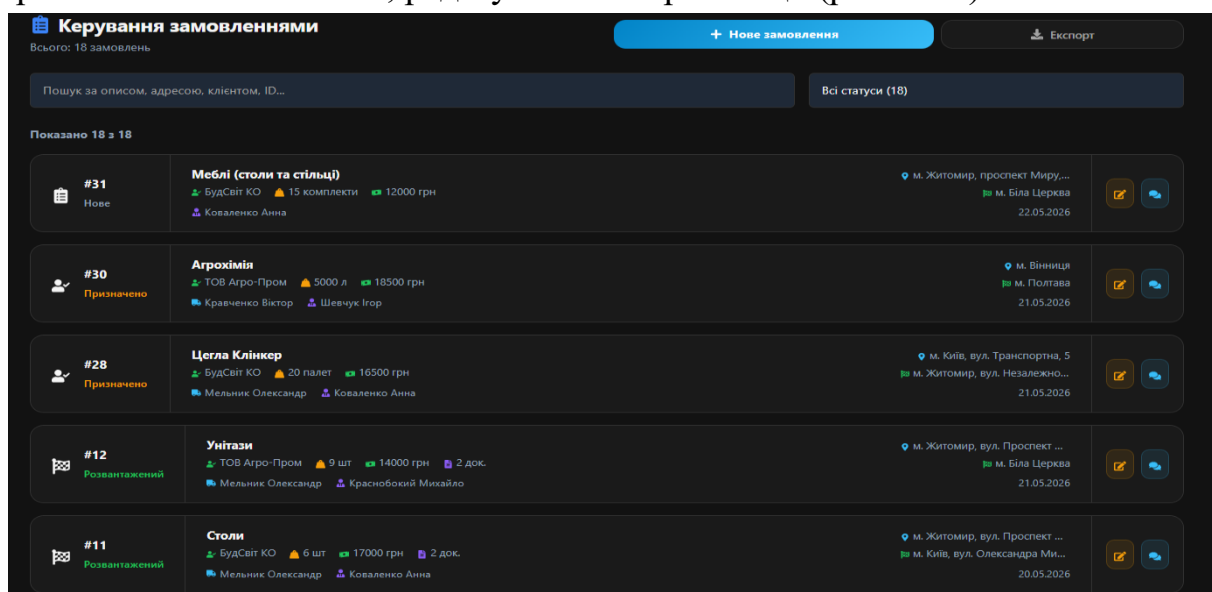


Рис. 3.12. Сторінка керування всіма замовленнями в кабінеті диспетчера

Процес створення замовлення диспетчером виглядає наступним чином. Диспетчер відкриває екран створення нового рейсу, вибирає клієнта-замовника, вказує текстовий опис вантажу, його загальну вагу, точні адреси завантаження та вивантаження. У спеціальній динамічній формі можна деталізувати специфікацію вантажу, яка автоматично серіалізується в JSON-масив перед відправкою на сервер. Далі диспетчер вибирає, на сторінці "Моніторинг", зі списку вільний автомобіль з водієм та зберігає замовлення (рис. 3.13).

Після завершення рейсу диспетчер має можливість автоматично згенерувати фінансові та товарні документи. Одним кліком миші система формує рахунок-фактуру для клієнта та створює офіційний бланк товарно-транспортної накладної (ТТН) у форматі PDF за допомогою серверної бібліотеки PDFKit(рис. 3.14).

Рис. 3.13. Вікно створення нового замовлення на перевезення

Рис. 3.14. Візуалізація згенерованого бланка товарно-транспортної накладної в інтерфейсі

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

Четверта роль - Водій (DRIVER). Інтерфейс кабінету водія адаптований для використання на мобільних пристроях і планшетах. Водій бачить виключно список тих рейсів, які призначені безпосередньо йому. Кожне замовлення проходить фіксований життєвий цикл статусів. Водій керує цим процесом за допомогою простих інтерактивних кнопок: прибувши на завантаження, він натискає кнопку «Завантажено» (статус змінюється на LOADED), вирушаючи в рейс - «У дорозі» (IN_TRANSIT), а після успішної доставки - «Доставлено» (DELIVERED)(рис.3.15).

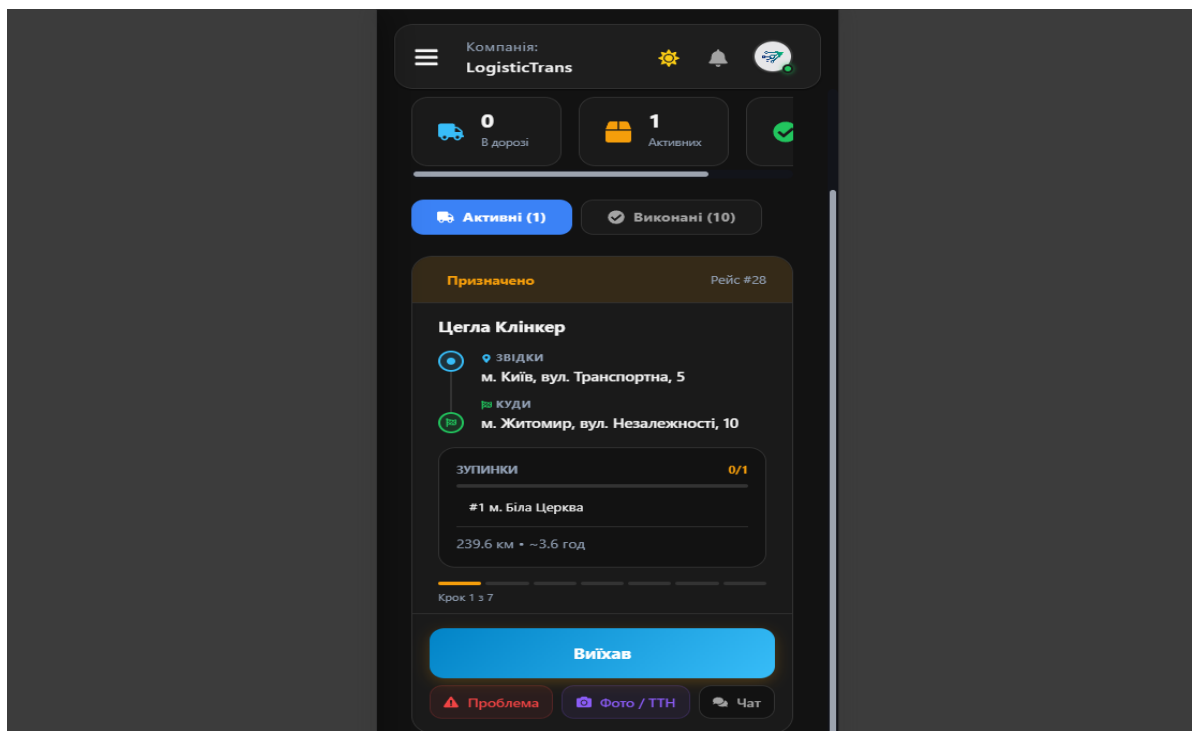


Рис. 3.15. Мобільний інтерфейс водія зі списком рейсів

Журнал аудиту 29						Пошук...
ЧАС	КОРИСТУВАЧ	ДІЯ	СУТНІСТЬ	ID	ДЕТАЛІ	
24.05.2026 18:29:58	Головний Адміністратор logisticsoft@gmail.com LogisticSoft Administration	LOGIN	User	#1	Успішний вхід в систему	
24.05.2026 15:45:41	Головний Адміністратор logisticsoft@gmail.com LogisticSoft Administration	LOGIN	User	#1	Успішний вхід в систему	
24.05.2026 15:44:24	Краснобокий Михайло kivromail@gmail.com LogisticTrans	LOGIN	User	#2	Успішний вхід в систему	
24.05.2026 15:43:26	Краснобокий Михайло kivromail@gmail.com LogisticTrans	LOGIN	User	#2	Успішний вхід в систему	
24.05.2026 15:38:06	Головний Адміністратор logisticsoft@gmail.com LogisticSoft Administration	LOGIN	User	#1	Успішний вхід в систему	
24.05.2026 15:36:28	Краснобокий Михайло kivromail@gmail.com LogisticTrans	UPDATE	Order	#31	Дані замовлення оновлено	
24.05.2026 15:36:09	Краснобокий Михайло kivromail@gmail.com LogisticTrans	UPDATE	Order	#31	Водія від'єзжено від замовлення	
24.05.2026 15:36:02	Краснобокий Михайло kivromail@gmail.com LogisticTrans	LOGIN	User	#2	Успішний вхід в систему	

Рис. 3.16 - Інтерфейс перегляду журналу логів аудиту дій користувачів

Для контролю безпеки та запобігання втрати інформації в системі реалізовано модуль логування дій користувачів, який доступний супер-адміністраторам. Будь-яке створення, модифікація чи видалення сутностей фіксується в журналі аудиту з вказівкою автора дії, компанії, точного часу та детальної специфікації змін (рис. 3.16).

З метою налаштування та персоналізації роботи користувачів і компаній у системі передбачено окремі інтерфейси налаштувань. Сторінка «Налаштування» акаунту дозволяє будь-якому користувачеві редагувати свої ПІБ та опис «Про себе», оновлювати фото (аватар), змінювати діючий пароль доступу та контактний email (через обов'язкове підтвердження паролем). Також на цій сторінці інтегровано функціональні перемикачі для активації звукових ефектів інтерфейсу, увімкнення розсилки системних email-сповіщень та підключення двофакторної автентифікації (2FA) (рис.3.17).

Рис. 3.17. Сторінка налаштувань профілю та двофакторної автентифікації користувача

Сторінка «Інформація про компанію» доступна адміністратору компанії та відображає основні реквізити фірми: її офіційну назву, юридичну адресу, телефон,

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

email та унікальний ідентифікатор компанії (unique ID), який використовується іншими користувачами при реєстрації для приєднання до її штату. Зміна цих реквізитів з метою безпеки заблокована для самостійного редагування та виконується лише через запит до технічної підтримки системи (рис.3.18).

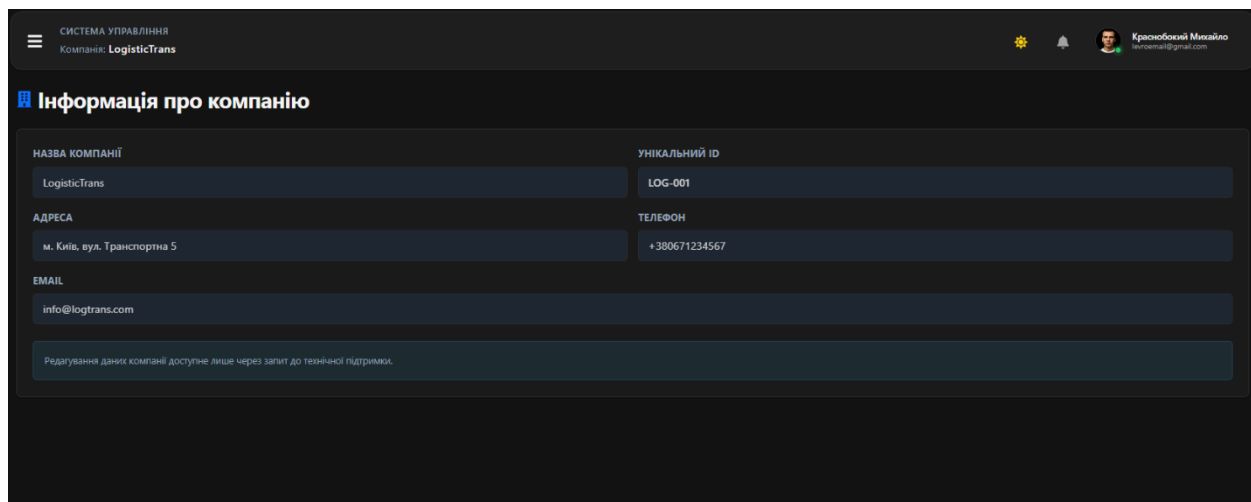


Рис. 3.18. Інтерфейс перегляду інформації про компанію в кабінеті адміністратора

3.3. Тестування роботи платформи автоматизації логістичних процесів

Важливим етапом життєвого циклу розробки є проведення всебічної перевірки працездатності створеного програмного забезпечення. Метою тестування вебплатформи «LogisticSoft» є підтвердження відповідності її функціоналу технічним вимогам, перевірка коректності обміну даними між клієнтською та серверною частинами додатку, а також верифікація механізмів безпеки та розмежування прав доступу [33]. Процес тестування було розділено на три ключові етапи.

Під час перевірки клієнтського інтерфейсу особлива увага приділялася валідації полів введення на формах реєстрації та авторизації, щоб унеможливити відправку некоректних даних. Також тестувалася реакція системи на дії користувача, зокрема швидкість відображення сповіщень та зміна стану кнопок. Окремо перевірялася адаптивність дизайну при симуляції екранів різних мобільних пристроїв.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

Наступним етапом стало тестування серверної логіки та API за допомогою програми Postman [35]. Для цього були створені колекції запитів, які імітували повний цикл дій користувача, починаючи від реєстрації і закінчуючи генерацією рахунків. Це дозволило переконатися у правильній обробці всіх HTTP-методів та коректності кодів відповідей сервера.

Заключним етапом була перевірка механізмів інформаційної безпеки та ізоляції. Зокрема, тестувалася робота проміжних обробників авторизації шляхом надсилання запитів без токенів доступу, які система успішно блокувала [34]. Перевірка мультикомпанійної архітектури підтвердила, що користувачі не можуть отримати доступ до даних інших організацій, оскільки сервер автоматично фільтрує запити за ідентифікатором компанії. Крім того, було перевірено захист системи від підбору паролів та DoS-атак за допомогою механізмів обмеження частоти запитів (Rate Limiting). Під час симуляції брутфорс-атаки, після перевищення ліміту в 5 запитів з однієї IP-адреси, система автоматично блокувала подальші спроби входу, повертаючи відповідну помилку.

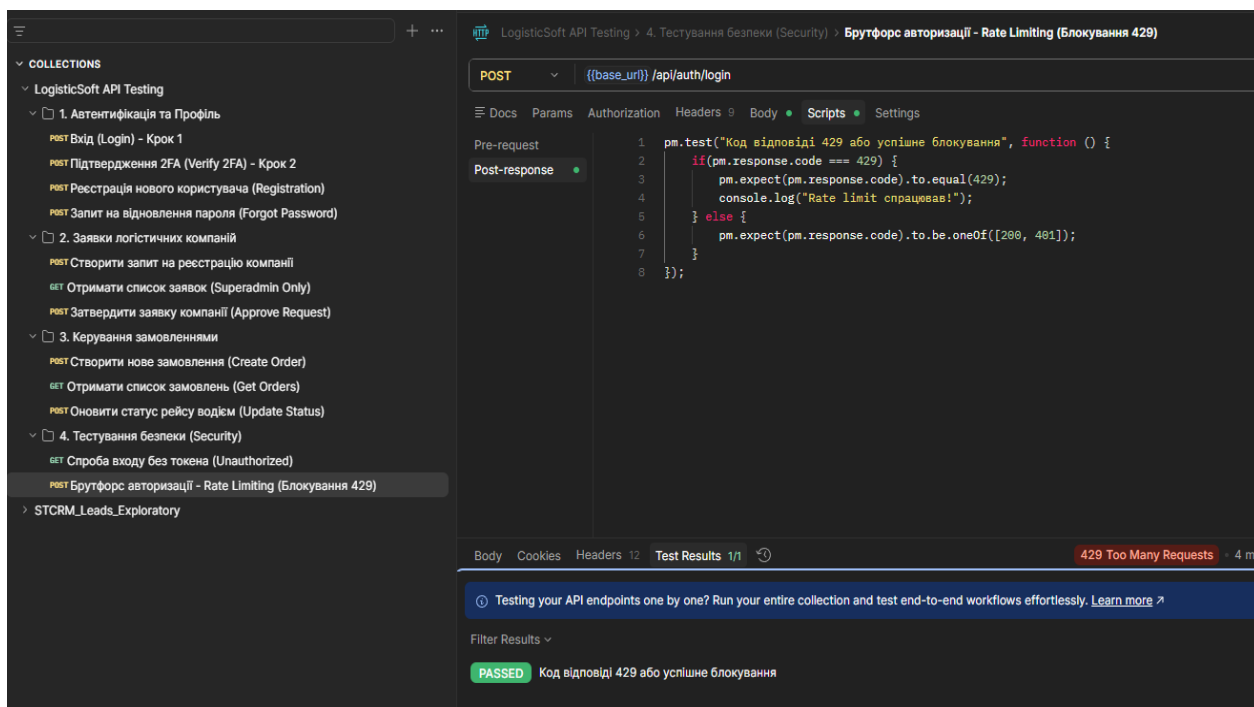


Рис. 3.19. Результат перевірки симуляції брутфорсу

Для систематизації проведених випробувань та фіксації їх результатів було розроблено тестові сценарії, результати виконання яких наведено в таблиці 3.2.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 3.2.

Специфікація та результати виконання тестових сценаріїв

№	Назва сценарію	Опис дій та вхідні дані	Очікуваний результат	Отриманий результат	Статус
1	Авторизація без токена	Запит GET до /api/orders без JWT-токена в заголовках	Помилка авторизації, код 401	Блокування запиту, код 401, статус "Не авторизовано"	Пройдено
2	Спроба входу з некоректними даними	Форма входу: неіснуючий email або неправильний пароль	Відображення помилки "Невірний email або пароль", код 401	Повідомлення про помилку на екрані, код 401	Пройдено
3	Реєстрація дублікату компанії	Повторний запит на реєстрацію з унікальним ID компанії, яка є	Відхилення запиту сервером, код 400	Код помилки 400, повідомлення про дублювання унікального ID	Пройдено
4	Створення замовлення диспетчером	Заповнення форми замовлення валідними даними та вибір водія	Створення запису в БД, зміна статусу водія на "ASSIGNED"	Запис створено, код 201, оновлення інтерфейсу диспетчера	Пройдено
5	Зміна статусу водієм	Водій натискає кнопку "В дорозі" у мобільному інтерфейсі	Оновлення статусу замовлення на "IN_TRANSIT" в БД, код 200	Статус оновлено, код 200, відправка події через WebSocket	Пройдено
6	Перегляд даних іншої компанії	Запит GET до /api/customers від імені водія Компанії А з параметром companyId Компанії Б	Повне приховування записів Компанії Б або код помилки 403	Запити іншої компанії не відображаються, ізоляція успішна	Пройдено

7	Динамічна генерація PDF бланка ТТН	Запит GET до /api/export/pdf/:id для завершеного замовлення	Рендеринг бланка ТТН та повернення файлу у вигляді BufferStream	Завантаження бланка PDF з оригінальними даними замовлення	Пройдено
8	Захист від брутфорсу	Понад 5 послідовних запитів на хвилину до /api/auth/login з однієї IP	Тимчасове блокування IP-адреси, код помилки 429	Повне блокування запитів, код 429, статус "Too Many Requests"	Пройдено

Проведене ручне функціональне тестування підтвердило стабільність роботи клієнтської та серверної частин вебплатформи «LogisticSoft», відсутність критичних помилок під час виконання базових операцій, надійність рольового розмежування доступу та повну готовність системи до використання в реальних бізнес-процесах логістичних підприємств малого та середнього бізнесу.

Висновки до третього розділу

У третьому розділі було детально розглянуто практичні аспекти розгортання, налаштування та тестування розробленого програмного забезпечення вебплатформи «LogisticSoft». Наведено чітку послідовність дій з конфігурування локального та виробничого серверних середовищ розгортання, описано структуру змінних конфігураційного файлу .env та алгоритм початкового очищення й заповнення бази даних тестовими даними.

Описано структуру та детальні алгоритми використання користувацького інтерфейсу платформи для чотирьох основних ролей: суперадміністратора, адміністратора компанії, диспетчера та водія. Продемонстровано роботу ключових модулів системи: інтерактивної карти Leaflet для моніторингу транспорту в реальному часі, модуля двофакторної автентифікації через електронну пошту, сторінок моніторингу та керування замовленнями в кабінеті диспетчера, модуля фінансового аналізу й статистики з інтерактивними діаграмами Recharts,

інтерфейсів налаштування облікового запису та перегляду даних компанії, вкладки «Користувачі» в Адмінцентрі та спільної сторінки «Водії компанії» для моніторингу зайнятості персоналу, роль-орієнтованого головного меню та публічної головної сторінки (Landing Page) системи, а також механізму динамічної генерації товарно-транспортних накладних у форматі PDF без засмічення файлової системи сервера.

Наведено методологію та результати комплексного ручного тестування системи. Функціональна перевірка основних API-ендпоінтів за допомогою Postman та тестування інтерфейсу в браузері підтвердили стабільність взаємодії між клієнтською та серверною частинами додатку. Тестування безпеки підтвердило надійну ізоляцію даних користувачів у межах архітектури Multitenancy та коректність роботи фільтрації запитів за допомогою authMiddleware.js. Результати випробувань продемонстрували відповідність системи встановленим вимогам швидкодії та безпеки, що підтверджує високу надійність та готовність вебплатформи «LogisticSoft» до впровадження у реальний бізнес-процес логістичних підприємств малого та середнього бізнесу.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було здійснено повний цикл проєктування, практичної реалізації та тестування вебплатформи автоматизації логістичних процесів вантажних перевезень – LogisticSoft.

На першому етапі було проведено детальний аналіз предметної області управління транспортною логістикою та досліджено чинне законодавство України у сфері регулювання вантажних перевезень. Проведений порівняльний аналіз існуючих комерційних рішень класу TMS (SAP TM, Oracle OTM, Sovtes, Linco) виявив їхні ключові переваги та функціональні обмеження для малого бізнесу, що дало змогу сформулювати вимоги до розроблюваної системи. Обґрунтовано вибір сучасного стеку технологій на базі концепції односторінкового застосунку та клієнт-серверної архітектури з використанням RESTful API. В якості інструментів розробки обрано середовище виконання Node.js з фреймворком Express для серверної частини, бібліотеку React.js для побудови вебінтерфейсу та реляційну СУБД MySQL з ORM Sequelize для надійного збереження даних.

В роботі розроблено архітектурні та інженерні рішення для створення вебплатформи. Побудовано UML-діаграми варіантів використання, послідовності та класів для моделювання логіки роботи системи. Спроектовано структуру реляційної бази даних, яка складається з одинадцяти пов'язаних таблиць та враховує вимоги мультикомпанійності (Multitenancy) шляхом ізоляції даних кожної логістичної фірми. Описано ключові алгоритми системи, зокрема алгоритм авторизації користувачів із генерацією сесійних JWT-токенів та алгоритм відстеження геолокації водіїв.

Також виконано практичну реалізацію та розгортання розробленої системи. Описано покроковий процес налаштування параметрів середовища, конфігурування бази даних за допомогою скрипту seed.js та запуск серверної й клієнтської частин. Для розгортання у виробничому середовищі (Production) було використано хмарний сервер на базі вебсервера Nginx як зворотного проксі та

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

менеджера процесів PM2, що забезпечило доступність системи за адресою <https://logisticsoft.pp.ua/>.

Детально описано користувацький інтерфейс для чотирьох основних ролей системи. Зокрема, суперадміністратор управляє запитами на реєстрацію нових компаній в глобальному Адмінцентрі. На рівні підприємства адміністратор компанії здійснює керування штатом, додавання транспортних засобів та моніторинг водійського складу, а також проводить фінансовий аналіз витрат і доходів за допомогою інтерактивних графіків Recharts. Основну операційну роботу виконує диспетчер, який створює замовлення, призначає водіїв, відстежує їхнє переміщення в реальному часі на інтерактивній карті та генерує бланки ТТН і рахунків у форматі PDF. Своєю чергою, водій використовує адаптивний мобільний інтерфейс виключно для перегляду призначених рейсів та зміни поточних статусів доставки.

З метою гарантування безпеки інтегровано механізми двофакторної автентифікації (2FA) через Email та обмеження частоти запитів за допомогою бібліотеки `express-rate-limit` для запобігання брутфорс-атакам на критичні ендпоінти. Проведене ручне функціональне тестування інтерфейсу та REST API за допомогою Postman підтвердило стабільність роботи вебплатформи, надійність рольового розмежування доступу та повну готовність продукту до впровадження у реальну діяльність логістичних підприємств.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Біловодська О. А. Транспортна логістика : навч. посіб. Київ : Центр учбової літератури, 2021. 224 с.
2. Логістика : теорія та практика : підручник / В. М. Майорова та ін. Львів : Магнолія-2006, 2020. 368 с.
3. Про розвиток цифрової логістики в Україні. Логістика : проблеми та рішення. 2023. № 2. С. 14-19.
4. SAP Transportation Management (SAP TM) : official website. URL: <https://www.sap.com/products/technology-platform.html> (дата звернення: 10.02.2026).
5. Oracle Transportation Management (OTM) : official website. URL: <https://www.oracle.com/scm/logistics/transportation-management/> (дата звернення: 17.02.2026).
6. Транспортна інформаційна система Sovtes : офіційний сайт. URL: <https://sovt.es.ua> (дата звернення: 21.02.2026).
7. SaaS-платформа Linco для експедиторів та перевізників : офіційний сайт. URL: <https://linco.com.ua> (дата звернення: 28.02.2026).
8. Закон України «Про транспортно-експедиторську діяльність» від 01.07.2004 р. № 1955-IV. Відомості Верховної Ради України. 2004. № 52. Ст. 562.
9. Впровадження системи електронного документообігу е-ТТН в Україні : Міністерство розвитку громад, територій та інфраструктури України. URL: <https://mindev.gov.ua> (дата звернення: 09.03.2026).
10. Гура О. М., Гура Д. О. Автоматизовані системи управління на транспорті. Харків : ХНАДУ, 2022. 180 с.
11. Bowersox D. J., Closs D. J., Cooper M. B. Supply Chain Logistics Management. 5th ed. New York : McGraw-Hill Education, 2019. 512 p.
12. Christopher M. Logistics & Supply Chain Management. 5th ed. London : Pearson Education, 2016. 320 p.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Diss. ... doctor of philosophy. Irvine : University of California, 2000. 180 p.
14. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley Professional, 2002. 576 p.
15. Coulouris G., Dollimore J., Kindberg T., Blair G. Distributed Systems : Concepts and Design. 5th ed. Boston : Addison-Wesley, 2011. 1064 p.
16. Richardson L., Ruby S. RESTful Web Services. Sebastopol : O'Reilly Media, 2007. 448 p.
17. Mozilla Developer Network. Single Page Applications (SPA) : glossary. URL: <https://developer.mozilla.org/en-US/docs/Glossary/SPA> (дата звернення: 11.03.2026).
18. Official Node.js Documentation. URL: <https://nodejs.org/docs/> (дата звернення: 15.03.2026).
19. Cantelon M., Harter M., Holowaychuk T., Rajlich N. Node.js in Action. Shelter Island : Manning Publications, 2014. 416 p.
20. Official Express.js Documentation. URL: <https://expressjs.com> (дата звернення: 21.03.2026).
21. Official MySQL Documentation. URL: <https://dev.mysql.com/doc/> (дата звернення: 28.03.2026).
22. Elmasri R., Navathe S. B. Fundamentals of Database Systems. 7th ed. Boston : Pearson, 2015. 1280 p.
23. Official React.js Documentation. URL: <https://react.dev> (дата звернення: 04.04.2026).
24. Official Leaflet Documentation. URL: <https://leafletjs.com> (дата звернення: 07.04.2026).
25. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns : Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.
26. Martin R. C. Clean Architecture : A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

27. Official Sequelize ORM Documentation. URL: <https://sequelize.org> (дата звернення: 09.04.2026).
28. Socket.io Official Documentation. URL: <https://socket.io/docs/> (дата звернення: 12.04.2026).
29. Bootstrap 5.3 Official Documentation. URL: <https://getbootstrap.com/docs/5.3/> (дата звернення: 13.04.2026).
30. PDFKit Library Documentation. URL: <https://pdfkit.org> (дата звернення: 13.04.2026).
31. Official Nginx Documentation. URL: <https://nginx.org/en/docs/> (дата звернення: 18.04.2026).
32. JWT.IO. Introduction to JSON Web Tokens. URL: <https://jwt.io/introduction/> (дата звернення: 21.04.2026).
33. express-rate-limit library documentation. URL: <https://www.npmjs.com/package/express-rate-limit> (дата звернення: 28.04.2026).
34. OWASP Cheat Sheet Series. Password Storage Cheat Sheet. URL: <https://cheatsheetseries.owasp.org/> (дата звернення: 02.05.2026).
35. Postman Learning Center. API Testing Guide. URL: <https://learning.postman.com/docs/writing-scripts/intro-to-scripts/> (дата звернення: 03.05.2026).

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг програмного коду

Модель замовлення/рейсу (Order.js)

```

const Order = sequelize.define('Order', {
  id: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true
  },
  description: {
    type: DataTypes.STRING,
    allowNull: false
  },
  weight: {
    type: DataTypes.FLOAT,
    allowNull: false
  },
  pickupAddress: {
    type: DataTypes.STRING,
    allowNull: false
  },
  deliveryAddress: {
    type: DataTypes.STRING,
    allowNull: false
  },
  status: {
    type: DataTypes.ENUM('NEW', 'ASSIGNED', 'DEPARTED', 'AT_LOADING',
      'LOADED', 'IN_TRANSIT', 'AT_UNLOADING', 'UNLOADED', 'CANCELLED'),
    defaultValue: 'NEW'
  },
  price: {
    type: DataTypes.FLOAT,
    allowNull: true
  },
  cargoItems: {
    type: DataTypes.JSON,

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        allowNull: true,
        defaultValue: null
    },
    deliveryStops: {
        type: DataTypes.JSON,
        allowNull: true,
        defaultValue: null
    },
    distance: {
        type: DataTypes.FLOAT,
        allowNull: true,
        defaultValue: null
    },
    estimatedDuration: {
        type: DataTypes.FLOAT,
        allowNull: true,
        defaultValue: null
    },
    documents: {
        type: DataTypes.JSON,
        allowNull: true,
        defaultValue: []
    },
    creatorId: {
        type: DataTypes.INTEGER,
        allowNull: true
    },
    vehicleId: {
        type: DataTypes.INTEGER,
        allowNull: true
    },
    companyId: {
        type: DataTypes.INTEGER,
        allowNull: true,
        references: {
            model: 'Companies',

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        key: 'id'
    }
}
});

module.exports = Order;

```

Модель користувача (User.js)

```

const { DataTypes } = require('sequelize');
const sequelize = require('../config/db');

// Модель користувачів системи (адміністратор, диспетчер, водій)
const User = sequelize.define('User', {
  id: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true
  },
  email: {
    type: DataTypes.STRING,
    allowNull: false,
    unique: true
  },
  password: {
    type: DataTypes.STRING,
    allowNull: false
  },
  fullName: {
    type: DataTypes.STRING,
    allowNull: false
  },
  role: {
    type: DataTypes.ENUM('SUPER_ADMIN', 'COMPANY_ADMIN', 'DISPATCHER',
'DRIVER'),
    defaultValue: 'DRIVER'
  },

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

companyId: {
  type: DataTypes.INTEGER,
  allowNull: true,
  references: {
    model: 'Companies',
    key: 'id'
  }
},
status: {
  type: DataTypes.ENUM('PENDING', 'APPROVED'),
  defaultValue: 'PENDING'
},
avatar: {
  type: DataTypes.STRING,
  allowNull: true
},
aboutMe: {
  type: DataTypes.TEXT,
  allowNull: true
},
emailNotifications: { type: DataTypes.BOOLEAN, defaultValue: false },
resetCode: { type: DataTypes.STRING },
resetCodeExpires: { type: DataTypes.BIGINT },
isOnline: { type: DataTypes.BOOLEAN, defaultValue: false },
lastSeen: { type: DataTypes.DATE, defaultValue: DataTypes.NOW },
twoFactorEnabled: { type: DataTypes.BOOLEAN, defaultValue: false },
twoFactorCode: { type: DataTypes.STRING, allowNull: true },
twoFactorCodeExpires: { type: DataTypes.BIGINT, allowNull: true }
});
module.exports = User;

```

Модель компанії-орендаря (Company.js)

```

const { DataTypes } = require('sequelize');
const sequelize = require('../config/db');

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const Company = sequelize.define('Company', {
  id: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true
  },
  name: {
    type: DataTypes.STRING,
    allowNull: false
  },
  uniqueId: {
    type: DataTypes.STRING,
    allowNull: false,
    unique: true
  },
  address: {
    type: DataTypes.STRING,
    allowNull: true
  },
  phone: {
    type: DataTypes.STRING,
    allowNull: true
  },
  email: {
    type: DataTypes.STRING,
    allowNull: true
  },
  isActive: {
    type: DataTypes.BOOLEAN,
    defaultValue: true },
  isSystem: {
    type: DataTypes.BOOLEAN,
    defaultValue: false
  }});
module.exports = Company;

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

Модель транспортного засобу (Vehicle.js)

```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/db');

const Vehicle = sequelize.define('Vehicle', {
  id: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true
  },
  brand: {
    type: DataTypes.STRING,
    allowNull: false
  },
  model: {
    type: DataTypes.STRING,
    allowNull: false
  },
  plateNumber: {
    type: DataTypes.STRING,
    allowNull: false,
    unique: true
  },
  capacity: {
    type: DataTypes.FLOAT,
    allowNull: false
  },
  isAvailable: {
    type: DataTypes.BOOLEAN,
    defaultValue: true
  },
  fuelConsumption: {
    type: DataTypes.FLOAT,
    allowNull: true,
    defaultValue: null
  },
},
```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    depreciationPerKm: {
      type: DataTypes.FLOAT,
      allowNull: true,
      defaultValue: null
    },
    driverId: {
      type: DataTypes.INTEGER,
      allowNull: true
    },
    companyId: {
      type: DataTypes.INTEGER,
      allowNull: true,
      references: {
        model: 'Companies',
        key: 'id'
      }
    }
  }
});
module.exports = Vehicle;

```

Модель контрагента (Customer.js)

```

const { DataTypes } = require('sequelize');
const sequelize = require('../config/db');

const Customer = sequelize.define('Customer', {
  id: { type: DataTypes.INTEGER, autoIncrement: true, primaryKey: true },
  name: { type: DataTypes.STRING, allowNull: false },
  edrpou: { type: DataTypes.STRING, allowNull: true },
  address: { type: DataTypes.STRING, allowNull: true },
  phone: { type: DataTypes.STRING, allowNull: true },
  email: { type: DataTypes.STRING, allowNull: true },
  companyId: {
    type: DataTypes.INTEGER,
    allowNull: true,
    references: {
      model: 'Companies',

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        key: 'id'
      }
    }
  });
module.exports = Customer;

```

Модель фінансового рахунку (Invoice.js)

```

const { DataTypes } = require('sequelize');
const sequelize = require('../config/db');

const Invoice = sequelize.define('Invoice', {
  id: {
    type: DataTypes.INTEGER,
    autoIncrement: true,
    primaryKey: true
  },
  orderId: {
    type: DataTypes.INTEGER,
    allowNull: false
  },
  customerId: {
    type: DataTypes.INTEGER,
    allowNull: false
  },
  companyId: {
    type: DataTypes.INTEGER,
    allowNull: false
  },
  amount: {
    type: DataTypes.FLOAT,
    allowNull: false
  },
  status: {
    type: DataTypes.ENUM('UNPAID', 'PAID', 'CANCELLED'),
    defaultValue: 'UNPAID'
  },
},

```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

```
    fileUrl: {  
        type: DataTypes.STRING,  
        allowNull: true  
    }  
});
```

```
module.exports = Invoice;
```

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		