ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка системи моніторингу логістики для оптимізації доставки»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Максим МАРТИНОВЧЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Максим МАРТИНОВЧЕНКО

Керівник: Вікторія КОРЕЦЬКА
канд.пед.наук, доц., проф..

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Мартиновченко Максиму Віталійовичу

1. Тема кваліфікаційної роботи: «Розробка системи моніторингу логістики для оптимізації доставки».
керівник кваліфікаційної роботи канд.пед.наук, доц., проф. Вікторія КОРЕЦЬКА, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про організацію логістичних процесів, методи моніторингу доставки, підходи до контролю маршрутів, транспортних засобів, складів і статусів перевезень, принципи оцінювання операційних ризиків у логістиці, опис клієнт-серверної архітектури web-застосунків та технічна документація.

4. Зміст розрахунково-пояснювальної записки:

1. Огляд та аналіз предметної галузі моніторингу логістики та оптимізації доставки.
2. Аналіз існуючих програмних засобів для управління доставками, маршрутами, транспортом і складськими даними.
3. Огляд засобів реалізації web-платформи logitrack.
4. Проєктування структури web-платформи, бази даних, ролей користувачів та основних модулів системи.
5. Проєктування і реалізація модуля route risk radar для оцінювання ризику затримки доставки.
6. Програмна реалізація серверної та клієнтської частин web-платформи.
7. Тестування web-платформи logitrack.
8. Апробація результатів дослідження.

5. Перелік графічного матеріалу: *презентація*

1. Логічна структура web-платформи logitrack.
2. ERD-діаграма бази даних платформи logitrack.
3. UML-діаграма варіантів використання.
4. UML-діаграма класів web-платформи.
5. UML-діаграма послідовності створення доставки.
6. Структура основних модулів платформи logitrack.
7. Структура REST API платформи logitrack.
8. Алгоритм розрахунку riskscore у модулі Route Risk Radar.
9. Взаємодія клієнтської частини, серверної частини та бази даних.
10. Екранні форми авторизації, dashboard, доставок, Route Risk Radar, довідників, профілю, користувачів та аналітики.
11. Апробація результатів дослідження.

6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури з теми логістики, моніторингу доставки та web-застосунків	23.02–01.03.2026	
2	Аналіз існуючих програмних засобів для управління доставками, маршрутами та логістичними процесами	02.03–08.03.2026	
3	Аналіз предметної галузі моніторингу логістики та визначення вимог до системи	09.03–16.03.2026	
4	Огляд засобів реалізації web-платформи LogiTrack	17.03–23.03.2026	
5	Проєктування web-платформи LogiTrack, бази даних, ролей користувачів та UML-діаграм	24.03–06.04.2026	
6	Програмна реалізація серверної та клієнтської частин системи	07.04–24.04.2026	
7	Реалізація модуля Route Risk Radar та сторінок профілю, довідників, користувачів і аналітики	25.04–01.05.2026	
8	Тестування функціональності, ролей доступу та модуля оцінювання ризиків	02.05–08.05.2026	
9	Оформлення роботи: вступ, висновки, реферат, список джерел та додатки	09.05–14.05.2026	
10	Розробка демонстраційних матеріалів і підготовка до попереднього захисту	15.05–22.05.2026	

Здобувач(ка) вищої освіти _____ Максим МАРТИНОВЧЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Вікторія КОРЕЦЬКА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 65 стор., 5 табл., 26 рис., 20 джерела.

Мета роботи – спрощення контролю логістичних процесів та своєчасного виявлення ризикових доставок за рахунок розробки web-застосунку LogiTrack для моніторингу доставок, маршрутів, складів, транспорту та операційних ризиків.

Об'єкт дослідження – процеси моніторингу логістики та контролю виконання доставок на підприємстві.

Предмет дослідження – засоби, методи та технології реалізації web-платформи для моніторингу логістичних процесів і оцінювання ризиків доставки.

Короткий зміст роботи: У роботі проаналізовано предметну галузь логістики та визначено основні проблеми, пов'язані з розрізненим зберіганням інформації про доставки, маршрути, транспортні засоби, склади та статуси перевезень. Встановлено, що відсутність централізованої системи ускладнює контроль доставки, збільшує час обробки інформації, знижує прозорість логістичних процесів і не дозволяє своєчасно виявляти рейси з підвищеним ризиком затримки. Проаналізовано існуючі програмні засоби для управління логістикою та доставкою: Excel / Google Sheets, SAP Transportation Management, Oracle Transportation Management, Routific, OptimoRoute та Zoho Creator Logistics.

Розроблено структуру web-платформи LogiTrack та визначено її ключові функціональні можливості, зокрема: реєстрацію та авторизацію користувачів, розмежування ролей ADMIN, MANAGER та VIEWER, керування доставками, маршрутами, складами, транспортними засобами, перегляд аналітики, адміністрування користувачів, ведення профілю користувача та роботу з довідниками. Особливістю системи є модуль Route Risk Radar, який автоматично оцінює ризик затримки доставки на основі статусу, поточної затримки, завантаження маршруту, пріоритету доставки, стану транспорту та завантаженості складу. Модуль формує riskScore, рівень ризику, перелік причин і рекомендовану дію для логіста.

У роботі використано Java та Spring Boot для реалізації серверної частини, PostgreSQL для зберігання даних, React і TypeScript для створення клієнтського інтерфейсу, а також REST API для організації взаємодії між клієнтською та серверною частинами системи.

Сферою використання web-платформи є логістичні відділи підприємств, служби доставки, торговельні компанії, складські комплекси та організації, які потребують централізованого інструменту для контролю доставок, маршрутів, транспорту, складів, статусів перевезень, аналітики та операційних ризиків.

КЛЮЧОВІ СЛОВА: ЛОГІСТИКА, МОНІТОРИНГ ДОСТАВКИ, LOGITRACK, WEB-ПЛАТФОРМА, МАРШРУТ, СКЛАД, ТРАНСПОРТ, ROUTE RISK RADAR, РИЗИК ЗАТРИМКИ, JAVA, SPRING BOOT, POSTGRESQL, REACT, TYPESCRIPT.

ЗМІСТ

РЕФЕРАТ 7
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 10
ВСТУП 12
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ 16
1.1 Загальна характеристика логістичних процесів 16
1.2 Актуальність створення платформи моніторингу логістики 18
1.3 Цільова аудиторія та основні користувачі системи 19
1.4 Основні функції та бізнес-логіка системи 20
1.5 Способи вирішення задачі та роль інфокомунікаційних засобів 21
1.6 Операційні ризики в логістичних процесах 22
1.7 Аналіз проблем і недоліків існуючих підходів 24
2 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ЛОГІСТИКИ ТА ОПТИМІЗАЦІЇ ДОСТАВКИ 25
2.1 Excel / Google Sheets 25
2.2 SAP Transportation Management 26
2.3 Oracle Transportation Management 27
2.4 Routific 29
2.5 OptimoRoute 30
2.6 Zoho Creator Logistics 31
2.7 Порівняльний аналіз програмних засобів 33
2.8 Функціональні та нефункціональні вимоги до платформи LogiTrack 36
3 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ WEB-ПЛАТФОРМИ LOGITRACK 38
3.1 Логічна архітектура web-платформи 38
3.2 Мова програмування Java 39
3.3 Spring Boot та Spring Framework 40
3.4 PostgreSQL 41
3.5 React та TypeScript 42
3.6 REST API та взаємодія клієнтської і серверної частин 44

3.7 Засоби безпеки та рольового доступу 45
4 ПРОЄКТУВАННЯ WEB-ПЛАТФОРМИ LOGITRACK 46
4.1 Логічна структура системи 46
4.2 Загальна архітектура платформи LogiTrack 48
4.3 Проєктування бази даних 49
4.4 UML-діаграма варіантів використання 52
4.5 UML-діаграма класів web-платформи 52
4.6 UML-діаграма послідовності створення доставки 53
4.7 Activity-діаграма роботи модуля Route Risk Radar 55
4.8 Структура основних модулів платформи LogiTrack 57
4.9 Проєктування матриці прав доступу користувачів 58
5 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ WEB-ПЛАТФОРМИ LOGITRACK 60
5.1 Загальна характеристика реалізованої web-платформи 60
5.2 Реалізація серверної частини 61
5.3 Реалізація клієнтської частини 62
5.4 Реалізація авторизації та реєстрації користувачів 63
5.5 Реалізація сторінки доставок 64
5.6 Реалізація сторінки деталей доставки 66
5.7 Реалізація сторінки аналітики 67
5.8 Реалізація модуля Route Risk Radar 68
5.9 Реалізація сторінки довідників 70
5.10 Реалізація сторінки профілю 71
5.11 Реалізація сторінки користувачів 72
5.12 Тестування програмного забезпечення 73
5.14 Результати тестування 75
ВИСНОВКИ 76
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79
ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ 81
ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ 92

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс прикладного програмування, що забезпечує взаємодію між різними програмними компонентами або сервісами.
CRUD – базові операції роботи з даними: створення, читання, оновлення та видалення записів.
CSS – мова стилів, що використовується для оформлення зовнішнього вигляду web-сторінок.
DB – база даних, призначена для збереження структурованої інформації системи.
DTO – об’єкт передавання даних між клієнтською та серверною частинами застосунку.
ERD – діаграма «сутність-зв’язок», що використовується для проєктування структури бази даних.
ETA – очікуваний час прибуття доставки або транспортного засобу до визначеної точки маршруту.
Git – система контролю версій, що дозволяє відстежувати зміни у програмному коді.
GitHub – web-сервіс для зберігання репозиторіїв, спільної роботи з кодом та керування версіями проєкту.
HTML – мова розмітки гіпертексту, що використовується для створення структури web-сторінок.
HTTP – протокол передавання даних у мережі Інтернет.
HTTPS – захищена версія протоколу HTTP, що використовується для безпечного обміну даними.
IDE – інтегроване середовище розробки програмного забезпечення.
JPA – специфікація Java для роботи з реляційними базами даних через об’єктну модель.
JSON – текстовий формат обміну даними між клієнтською та серверною частинами системи.
JWT – формат токена, що може використовуватися для авторизації та перевірки

прав доступу користувача.
KPI – ключовий показник ефективності, який використовується для оцінювання результативності логістичних процесів.
LogiTrack – web-платформа для моніторингу логістики, доставок, маршрутів, складів, транспорту та операційних ризиків.
ORM – технологія об’єктно-реляційного відображення, яка забезпечує зв’язок між об’єктами програми та таблицями бази даних.
PostgreSQL – об’єктно-реляційна система керування базами даних.
React – бібліотека JavaScript для створення компонентного користувацького інтерфейсу.
REST – архітектурний стиль побудови web-сервісів для обміну даними між клієнтом і сервером.
Route Risk Radar – модуль web-платформи LogiTrack, призначений для автоматичного оцінювання ризику затримки доставки.
SPA – односторінковий web-застосунок, у якому основна взаємодія з користувачем відбувається без повного перезавантаження сторінки.
SQL – мова структурованих запитів для роботи з реляційними базами даних.
Spring Boot – фреймворк для швидкої розробки серверних web-застосунків мовою Java.
TMS – система управління транспортом, яка використовується для планування, контролю та аналізу перевезень.
TypeScript – мова програмування на основі JavaScript із підтримкою статичної типізації.
UML – уніфікована мова моделювання, що використовується для побудови діаграм програмної системи.

ВСТУП

Актуальність теми**.** Логістика є важливою складовою діяльності підприємств, які займаються доставкою товарів, перевезенням вантажів, обслуговуванням замовлень та роботою зі складськими підрозділами. Від якості організації логістичних процесів залежить швидкість виконання доставок, рівень задоволеності клієнтів, ефективність використання транспорту, завантаженість складів і здатність підприємства своєчасно реагувати на проблемні ситуації.

У сучасних умовах логістичний відділ повинен працювати не лише з переліком замовлень, а й з великою кількістю взаємопов'язаних даних: маршрутами, транспортними засобами, складами, водіями, статусами доставок, пріоритетами замовлень, очікуваним часом прибуття та можливими затримками. Якщо така інформація зберігається в різних таблицях, документах або зовнішніх сервісах, працівникам складно швидко отримати актуальну картину роботи. Це призводить до втрати часу, дублювання операцій, помилок у плануванні маршрутів і несвоєчасного виявлення ризикових доставок.

Однією з важливих проблем логістичного управління є відсутність простого інструменту для пріоритезації проблемних рейсів. Логіст може бачити список доставок, але не завжди одразу зрозуміло, які з них потребують першочергової уваги. На ризик затримки можуть впливати різні фактори: поточна затримка, завантаження маршруту, пріоритет доставки, стан транспортного засобу, завантаженість складу та поточний статус перевезення. Тому доцільним є створення системи, яка не лише відображає логістичні дані, а й допомагає визначати доставки з підвищеним операційним ризиком.

Web-платформа LogiTrack спрямована на централізований моніторинг логістичних процесів, контроль доставок, маршрутів, складів, транспорту та операційних ризиків. Особливістю системи є модуль Route Risk Radar, який автоматично оцінює ризик затримки доставки на основі простих показників і формує для користувача зрозумілий рівень ризику, перелік причин і рекомендовану

дію. Це дозволяє логісту не просто переглядати таблицю доставок, а швидше визначати проблемні рейси та приймати рішення до того, як затримка стане критичною.

Мета роботи – спрощення контролю логістичних процесів та своєчасного виявлення ризикових доставок за рахунок розробки web-платформи LogiTrack для моніторингу доставок, маршрутів, складів, транспорту та операційних ризиків.

Об’єкт дослідження – процеси моніторингу логістики та контролю виконання доставок на підприємстві.

Предмет дослідження – засоби, методи та технології реалізації web-платформи для моніторингу логістичних процесів і оцінювання ризиків доставки.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Проаналізувати предметну галузь моніторингу логістики та оптимізації доставки.
2. Визначити основні проблеми контролю доставок, маршрутів, складів і транспортних засобів.
3. Провести аналіз існуючих програмних засобів для управління логістичними процесами.
4. Визначити функціональні та нефункціональні вимоги до web-платформи LogiTrack.
5. Обґрунтувати вибір технологій для реалізації клієнтської, серверної частини та бази даних.
6. Спроєктувати логічну структуру системи, базу даних, ролі користувачів та основні UML-діаграми.
7. Розробити модуль Route Risk Radar для оцінювання ризику затримки доставки.
8. Реалізувати основні модулі web-платформи LogiTrack.
9. Провести тестування програмного забезпечення та оцінити результати перевірки.

Методи дослідження. У роботі використовуються методи аналізу предметної галузі, порівняльного аналізу програмних аналогів, об’єктно-орієнтованого

проєктування, UML-моделювання, проєктування бази даних, розробки клієнт-серверного web-застосунку, реалізації рольового доступу та тестування програмного забезпечення.

Наукова новизна одержаних результатів полягає у формуванні структури web-платформи LogiTrack, яка поєднує централізований моніторинг доставок, маршрутів, складів, транспорту, аналітику логістичних процесів і модуль автоматичного оцінювання ризику затримки доставки Route Risk Radar.

Практична значущість одержаних результатів полягає в тому, що розроблена web-платформа може бути використана в логістичних відділах підприємств для контролю доставок, відстеження статусів перевезень, перегляду завантаженості маршрутів і складів, керування довідниками транспорту та своєчасного виявлення ризикових рейсів. У подальшому систему можна розширити за рахунок додавання GPS-моніторингу, ETA Simulator, What-if сценаріїв переназначення доставки та інтеграції з зовнішніми TMS або ERP-системами.

Апробація результатів та публікації. Основні наукові рішення та результати бакалаврської роботи були доповідані на науково-практичних конференціях, що відбулися на базі Державного університету інформаційно-комунікаційних технологій:

– Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Розробка платформи моніторингу логістики для оптимізації доставки.

– Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація», 1–3 травня 2026 року, Київ, ДУІКТ. Перспективи реалізації web-платформи LogiTrack для моніторингу логістики та оцінювання ризиків доставки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальна характеристика логістичних процесів

Логістична діяльність є важливою складовою роботи підприємств, які займаються доставкою товарів, перевезенням вантажів, обслуговуванням замовлень, складськими операціями та взаємодією з клієнтами. Від того, наскільки якісно організовано логістичні процеси, залежить швидкість виконання доставок, рівень задоволеності клієнтів, завантаженість транспорту, ефективність використання складів і загальні витрати підприємства.

У сучасних умовах логістика не обмежується лише переміщенням товару з однієї точки в іншу. Вона включає планування маршрутів, контроль транспорту, розподіл замовлень між рейсами, відстеження статусів доставки, роботу зі складами, оцінювання ризиків затримки та аналіз результатів виконання перевезень. Тобто логістичний процес є комплексною системою, у якій кожен елемент впливає на загальний результат.

Для підприємства важливо мати актуальну інформацію про те, які доставки вже виконані, які перебувають у процесі, які мають затримки, які маршрути перевантажені, який транспорт доступний, а які склади мають високе навантаження. Якщо така інформація зберігається в різних таблицях, документах або окремих сервісах, логістичному відділу складно швидко оцінити реальний стан доставки та прийняти правильне рішення.

У межах даної кваліфікаційної роботи розглядається задача створення web-платформи LogiTrack, яка призначена для централізованого моніторингу доставок, маршрутів, складів, транспорту та операційних ризиків. Основна ідея системи полягає в тому, щоб надати логісту не просто таблицю з доставками, а інструмент для аналізу поточної ситуації та швидкого виявлення проблемних рейсів.

Рис. 1.1 Загальна схема логістичного процесу підприємства

Загальний логістичний процес можна подати як послідовність етапів: створення або отримання замовлення, перевірка складу, формування маршруту, призначення транспорту, виконання доставки, оновлення статусу, завершення рейсу та формування звітності. На кожному з цих етапів можуть виникати проблеми: недостатня кількість транспорту, перевантаження маршруту, затримка на складі, зміна пріоритету доставки або технічний стан транспортного засобу.

Саме тому для логістичної системи важливо не лише зберігати дані, а й забезпечувати їх зручне відображення, фільтрацію, аналіз і використання для прийняття управлінських рішень.

1.2 Актуальність створення платформи моніторингу логістики

Актуальність створення платформи моніторингу логістики пов’язана з тим, що підприємства часто працюють з великою кількістю доставок, маршрутів, транспортних засобів і складських операцій. За відсутності єдиної інформаційної системи дані можуть дублюватися, втрачатися або оновлюватися із запізненням. Це ускладнює контроль логістичних процесів і може призводити до затримок доставки.

Однією з типових проблем є використання електронних таблиць для ведення доставок. Такий підхід може бути зручним на початковому етапі, коли кількість замовлень невелика. Проте зі збільшенням обсягу роботи таблиці стають незручними: складно контролювати права доступу, відстежувати історію змін, швидко знаходити проблемні доставки та автоматично формувати аналітичні показники.

Іншою проблемою є відсутність єдиного механізму оцінювання ризиків. Наприклад, доставка може мати високий пріоритет, маршрут може бути перевантажений, склад може працювати з підвищеним навантаженням, а транспортний засіб може мати обмежену доступність. Окремо ці фактори можуть здаватися не критичними, але разом вони формують високий ризик затримки. Якщо система не виділяє такі доставки окремо, логіст може помітити проблему занадто пізно.

Платформа LogiTrack вирішує цю задачу за рахунок централізації логістичних даних і впровадження модуля Route Risk Radar. Цей модуль дозволяє автоматично оцінювати ризик затримки доставки на основі кількох показників: статусу доставки, поточної затримки, завантаження маршруту, пріоритету, стану транспорту та завантаженості складу. Завдяки цьому логіст отримує не просто список доставок, а пріоритезований перелік рейсів, які потребують уваги.

Таким чином, створення web-платформи LogiTrack є актуальним, оскільки вона дозволяє підвищити прозорість логістичних процесів, зменшити кількість

ручних операцій, прискорити обробку інформації та забезпечити своєчасне виявлення проблемних доставок.

1.3 Цільова аудиторія та основні користувачі системи

Цільовою аудиторією web-платформи LogiTrack є підприємства, які регулярно виконують доставку товарів, працюють з маршрутами, складами, транспортом і потребують контролю логістичних процесів. До таких організацій можна віднести логістичні компанії, служби доставки, інтернет-магазини, торговельні підприємства, складські комплекси та виробничі компанії з власною системою перевезень.

Основними користувачами системи є адміністратор, логіст або диспетчер, а також керівник або користувач з правами перегляду. Для спрощення реалізації в системі передбачено три ролі: ADMIN**,** MANAGER та VIEWER.

Роль ADMIN призначена для адміністратора системи. Такий користувач має повний доступ до функціональності платформи. Він може керувати користувачами, змінювати ролі, активувати або вимикати облікові записи, створювати, редагувати та видаляти довідники складів, транспорту, маршрутів і доставок. Також адміністратор має доступ до аналітики та модуля Route Risk Radar.

Роль MANAGER відповідає логісту або диспетчеру. Це основний робочий користувач системи, який працює з доставками, змінює статуси, оновлює ETA, додає нотатки, призначає транспорт і контролює ризикові рейси. MANAGER може переглядати довідники складів, транспорту та маршрутів, але не має повного адміністративного доступу до системних даних і не може змінювати ролі користувачів.

Роль VIEWER призначена для керівника або спостерігача. Такий користувач має доступ лише до перегляду інформації. Він може бачити dashboard, аналітику, деталі доставок, профіль користувача та результати оцінювання ризиків, але не може створювати, редагувати або видаляти записи.

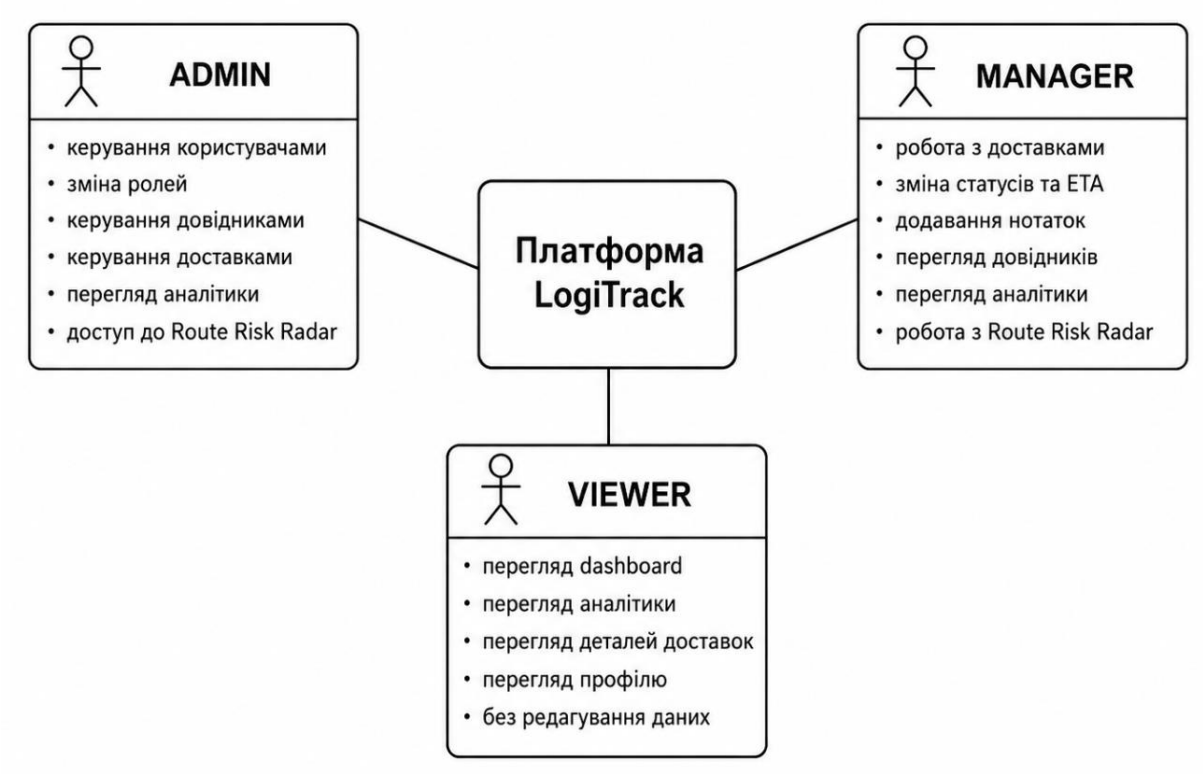


Рис. 1.2 Основні користувачі платформи LogiTrack та їх функції

Такий поділ ролей дозволяє зробити систему зрозумілою та безпечною. Кожен користувач отримує лише ті можливості, які відповідають його функціональним обов'язкам. Це зменшує ризик випадкового видалення або зміни важливих даних і робить роботу з платформою більш контрольованою.

1.4 Основні функції та бізнес-логіка системи

Web-платформа LogiTrack повинна забезпечувати централізовану роботу з логістичними даними. Її основна бізнес-логіка побудована навколо доставок, маршрутів, складів, транспортних засобів, користувачів і ризиків затримки.

Однією з ключових функцій є робота з доставками. Користувач з відповідними правами може створювати нові доставки, переглядати їх список, редагувати дані, змінювати статус, вказувати ETA, додавати нотатки та прив'язувати доставку до маршруту, транспорту і складу. Для зручності система повинна підтримувати фільтрацію доставок за статусом, пріоритетом, маршрутом, датою або рівнем ризику.

Другою важливою функцією є робота з довідниками. У системі передбачено довідники складів, транспортних засобів і маршрутів. Вони потрібні для того, щоб користувачі не вводили однакові дані вручну, а могли обирати їх зі структурованих списків. Це зменшує кількість помилок і робить дані більш узгодженими.

Третім функціональним блоком є аналітика. Платформа повинна відображати короткий summary щодо кількості доставок, частки затримок, найбільш завантажених маршрутів, проблемних зон і завантаженості складів. Така аналітика допомагає керівнику або логісту швидше оцінити загальний стан логістичних процесів.

Окреме місце займає модуль Route Risk Radar. Його задача полягає в автоматичному визначенні доставок з підвищеним ризиком затримки. Система аналізує кілька факторів і формує riskScore від 0 до 100. Залежно від отриманого значення доставка отримує рівень ризику: LOW, MEDIUM, HIGH або CRITICAL. Крім цього, система повинна показувати причини ризику та рекомендовану дію, наприклад: зв’язатися з клієнтом, перенести рейс на доступний транспорт або поставити доставку під ручний контроль.

Бізнес-логіка LogiTrack побудована таким чином, щоб користувач не витрачав час на ручне порівняння великої кількості показників. Система сама виділяє найбільш проблемні доставки, а логіст уже приймає рішення на основі підготовленої інформації.

1.5 Способи вирішення задачі та роль інфокомунікаційних засобів

Задачу моніторингу логістики можна вирішувати різними способами. Найпростішим варіантом є ручне ведення обліку в паперових документах або електронних таблицях. Такий підхід не потребує складного впровадження, але має суттєві обмеження. Він не забезпечує автоматичного оновлення даних, контролю доступу, централізованого аналізу та швидкого виявлення проблемних доставок.

Іншим варіантом є використання готових комерційних TMS-систем. Такі системи можуть мати широкий функціонал, підтримувати планування маршрутів,

управління транспортом, інтеграцію з іншими сервісами та аналітику. Проте для малого або середнього підприємства вони можуть бути надто складними, дорогими або перевантаженими функціями, які не використовуються в повсякденній роботі.

Доцільним рішенням у межах даної роботи є створення власної web-платформи, адаптованої під конкретну задачу: моніторинг доставок, контроль маршрутів, роботу зі складами, транспортом і ризиками затримки. Web-формат дозволяє користувачам працювати з системою через браузер без встановлення окремого програмного забезпечення. Це спрощує доступ до платформи та робить її зручною для різних типів користувачів.

Інфокомунікаційні засоби відіграють важливу роль у вирішенні цієї задачі, оскільки вони забезпечують швидкий обмін даними між клієнтською і серверною частинами системи, централізоване збереження інформації в базі даних, відображення актуального стану доставок і можливість подальшого розширення функціональності. Наприклад, у майбутньому до LogiTrack можна додати GPS-моніторинг, інтеграцію з зовнішніми сервісами доставки, автоматичне надсилання повідомлень клієнтам або модуль моделювання альтернативних логістичних рішень.

Таким чином, використання web-технологій є доцільним способом реалізації системи, оскільки воно забезпечує доступність, масштабованість, централізоване зберігання даних і можливість подальшого розвитку платформи.

1.6 Операційні ризики в логістичних процесах

Операційні ризики в логістиці — це фактори, які можуть негативно вплинути на своєчасність, якість або стабільність виконання доставки. Вони можуть виникати на різних етапах логістичного процесу: під час підготовки товару на складі, формування маршруту, призначення транспорту, виконання рейсу або оновлення статусу доставки.

Одним із найпоширеніших ризиків є затримка доставки. Вона може бути пов’язана з перевантаженням маршруту, технічним станом транспорту,

складськими затримками або високим навантаженням на водія. Якщо затримка не виявлена вчасно, вона може вплинути на клієнтський сервіс, графік наступних доставок і загальну ефективність роботи логістичного відділу.

Іншим важливим фактором є завантаження маршруту. Якщо маршрут має занадто велику кількість доставок або перевищує допустиме навантаження, зростає ймовірність того, що частина замовлень буде виконана із запізненням. Також важливо враховувати пріоритет доставки. Наприклад, критична або високопріоритетна доставка потребує більшої уваги, ніж стандартне замовлення.

На ризик також впливає стан транспортного засобу. Якщо транспорт перебуває в ремонті, має статус недоступного або вже використовується в іншому рейсі, це може створити додаткові проблеми для виконання доставки. Окремо слід враховувати завантаженість складу. Якщо склад має високий відсоток завантаження або не встигає підготувати товар до відправлення, це також може збільшити ризик затримки.

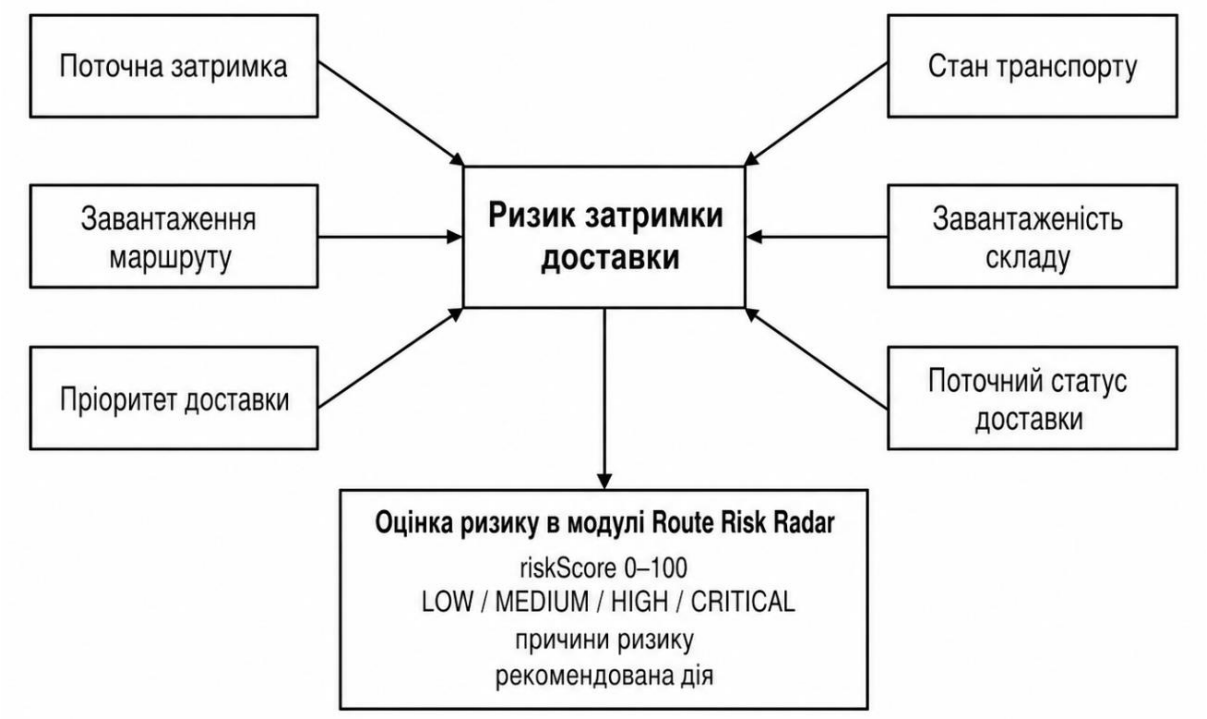


Рис. 1.3 Схема операційних ризиків доставки

У платформі LogiTrack ці фактори враховуються в модулі Route Risk Radar. Система аналізує поточний стан доставки та формує оцінку ризику. Такий підхід не замінює рішення логіста, але допомагає швидше побачити проблемні рейси та сконцентрувати увагу на тих доставках, які можуть створити найбільші операційні наслідки.

1.7 Аналіз проблем і недоліків існуючих підходів

Аналіз предметної галузі показує, що традиційні підходи до контролю логістичних процесів мають низку недоліків. Найбільш поширеною проблемою є розрізнене зберігання інформації. Дані про доставки можуть вестися в одній таблиці, інформація про транспорт — в іншій, маршрути — у внутрішніх документах, а комунікація між працівниками — у месенджерах. У результаті складно швидко отримати повну картину поточного стану логістики.

Ще одним недоліком є велика кількість ручних операцій. Якщо логіст самостійно оновлює статуси, перевіряє завантаження маршрутів, аналізує затримки та формує звіти, це потребує багато часу і збільшує ризик помилок. При великій кількості доставок ручний контроль стає менш ефективним, оскільки працівник може не помітити критичну ситуацію вчасно.

Також проблемою є недостатня прозорість статусів доставки. Якщо статуси не оновлюються регулярно або зберігаються в різних джерелах, керівник не може швидко оцінити, скільки доставок виконано, скільки перебуває в процесі, які рейси затримуються і які маршрути перевантажені. Це ускладнює управлінські рішення.

Окремо слід виділити обмежені можливості прогнозування або раннього виявлення ризиків. Багато простих систем дозволяють бачити факт затримки, але не допомагають визначити доставку, яка ще не стала критичною, але вже має ознаки підвищеного ризику. Саме ця проблема є однією з причин розробки модуля Route Risk Radar у платформі LogiTrack.

2 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ЛОГІСТИКИ ТА ОПТИМІЗАЦІЇ ДОСТАВКИ

2.1 Excel / Google Sheets

Одним із найпростіших засобів, які можуть використовуватися для контролю логістичних процесів, є електронні таблиці Excel або Google Sheets. На практиці такий підхід часто застосовується невеликими підприємствами, які ще не мають окремої інформаційної системи для управління доставками. У таблицях можна вести перелік замовлень, зазначати адресу доставки, дату, відповідального водія, маршрут, статус виконання, примітки щодо затримки та іншу інформацію.

Перевагою електронних таблиць є простота використання. Працівникам не потрібно проходити складне навчання, оскільки більшість користувачів уже мають базові навички роботи з таблицями. Також таблиці дозволяють швидко створювати власні шаблони, застосовувати фільтри, сортування, прості формули, умовне форматування та зведені таблиці. Google Sheets додатково підтримує спільну роботу кількох користувачів у режимі онлайн.

Однак для повноцінного моніторингу логістики електронних таблиць недостатньо. Основна проблема полягає в тому, що більшість процесів усе одно виконуються вручну. Логіст повинен самостійно оновлювати статуси, контролювати актуальність даних, перевіряти завантаження маршрутів і формувати звіти. При великій кількості доставок це збільшує ймовірність помилок і ускладнює своєчасне виявлення проблем.

Також електронні таблиці не мають повноцінного рольового доступу. Наприклад, складно чітко розмежувати можливості адміністратора, логіста і користувача, який має лише переглядати дані. Крім того, таблиці не забезпечують автоматичного розрахунку ризику доставки на основі різних факторів, таких як затримка, завантаження маршруту, пріоритет доставки або стан транспорту.

У межах даної роботи Excel / Google Sheets можна розглядати як базовий аналог, який дозволяє вести простий облік, але не вирішує задачу комплексного моніторингу логістичних процесів.

2.2 SAP Transportation Management

SAP Transportation Management є корпоративним програмним рішенням для управління транспортними процесами. Воно орієнтоване переважно на великі підприємства, які мають складні логістичні ланцюги, значну кількість перевезень, декілька складів, різні типи транспорту та потребу в інтеграції з іншими корпоративними системами.

Система дозволяє планувати перевезення, контролювати виконання транспортних операцій, працювати із замовленнями, маршрутами, перевізниками, витратами та аналітичними показниками. Таке рішення може бути корисним для великих компаній, які потребують комплексного управління логістикою на рівні всього підприємства.

До переваг SAP Transportation Management можна віднести масштабованість, широкий функціонал, підтримку складних бізнес-процесів, інтеграцію з іншими продуктами SAP та можливість використання аналітики для прийняття управлінських рішень. Система підходить для підприємств, які працюють з великими обсягами даних і мають потребу в глибокій автоматизації транспортної логістики.

Водночас це рішення має і суттєві недоліки. Насамперед воно є складним для впровадження та потребує значних фінансових витрат. Для налаштування системи часто потрібні спеціалісти, які мають досвід роботи з SAP. Крім того, для малого або середнього підприємства така система може бути надмірною, оскільки містить велику кількість функцій, які не завжди потрібні для базового моніторингу доставок.

У порівнянні з LogiTrack, SAP Transportation Management є значно масштабнішим рішенням, але менш гнучким для швидкої адаптації під конкретну

задачу невеликого логістичного відділу. LogiTrack орієнтований не на повну корпоративну автоматизацію, а на зручний контроль доставок, маршрутів, складів, транспорту та операційних ризиків.

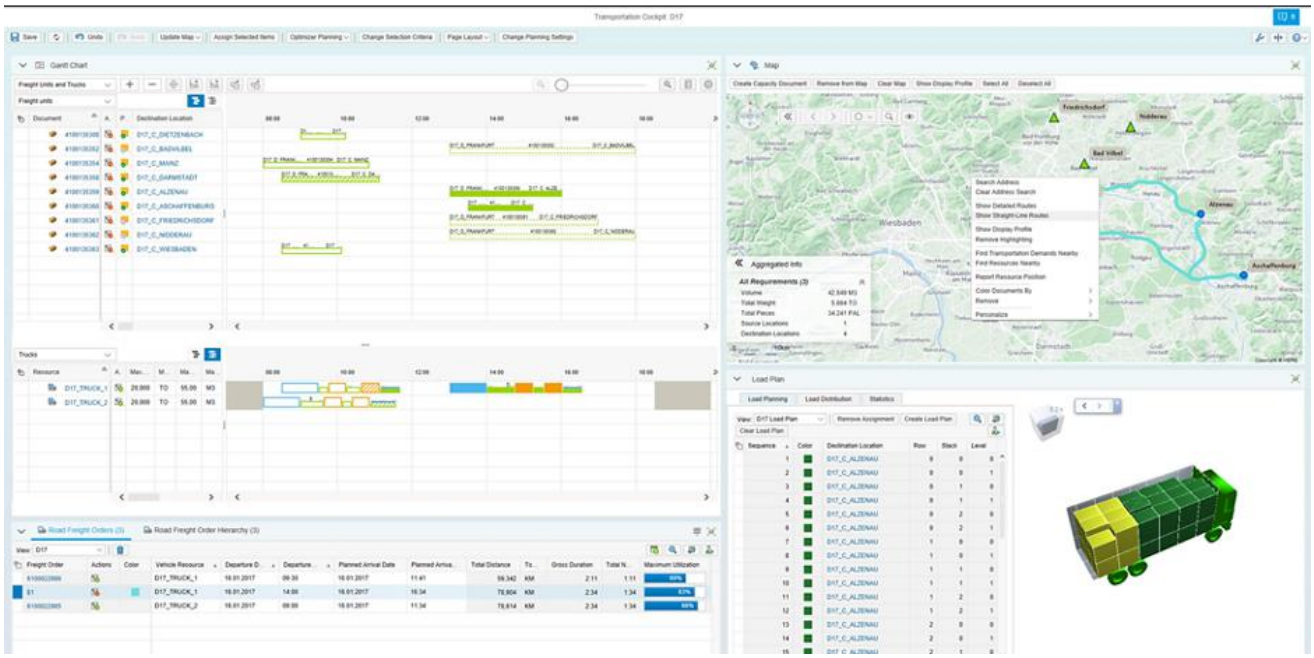


Рис. 2.2 Приклад інтерфейсу SAP Transportation Management

2.3 Oracle Transportation Management

Oracle Transportation Management також належить до корпоративних систем управління транспортом і логістикою. Це програмне забезпечення використовується для планування перевезень, оптимізації логістичних процесів, контролю транспортних операцій, аналізу витрат і управління ланцюгами постачання.

Система може застосовуватися у великих організаціях, де необхідно координувати різні види транспорту, велику кількість замовлень, перевізників і складних маршрутів. Oracle Transportation Management дозволяє централізовано керувати транспортними процесами та отримувати аналітичну інформацію для оптимізації логістики.

Перевагою цього рішення є потужна функціональність. Система підтримує управління перевезеннями, інтеграцію з іншими корпоративними рішеннями,

аналіз логістичних витрат і роботу з великими обсягами інформації. Вона підходить для компаній, які мають складну структуру постачання та потребують комплексного контролю транспортних процесів.

Проте, як і SAP Transportation Management, Oracle Transportation Management має високу складність впровадження. Для повноцінного використання системи потрібно виконати налаштування бізнес-процесів, інтеграцію з іншими сервісами, навчання персоналу та супровід. Це може бути недоцільним для компаній, яким потрібне більш просте рішення для щоденного контролю доставок.

Для задачі, що розглядається в цій роботі, Oracle Transportation Management є важливим аналогом, але він більше підходить для великих підприємств. Розроблювана платформа LogiTrack має бути простішою, зрозумілішою та сфокусованою на практичних функціях: моніторингу доставок, роботі з довідниками, аналітиці та визначенні ризикових рейсів.

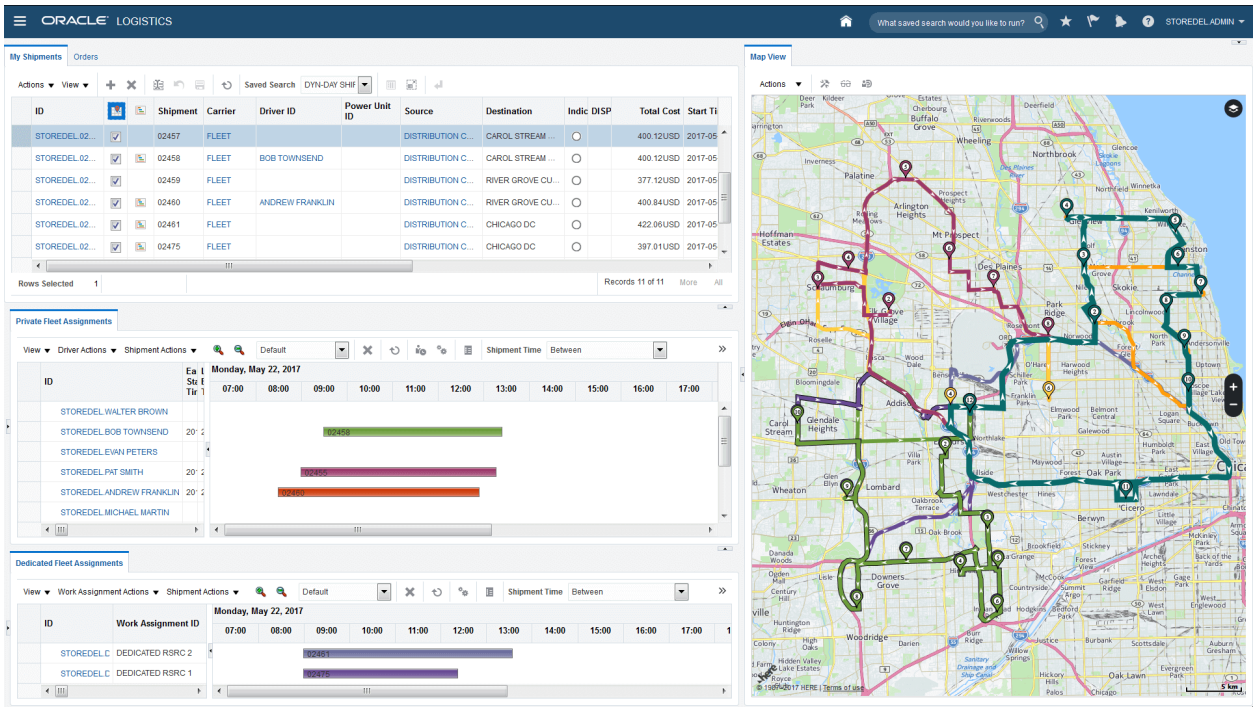


Рис. 2.3 Приклад інтерфейсу Oracle Transportation Management

2.4 Routific

Routific є програмним засобом, який орієнтований на оптимізацію маршрутів доставки. Його основна задача полягає в тому, щоб допомогти компаніям ефективніше планувати маршрути, зменшувати витрати часу на доставку та покращувати роботу водіїв. Такий сервіс особливо корисний для компаній, які мають багато точок доставки протягом дня.

Routific дозволяє створювати маршрути, розподіляти замовлення між водіями, враховувати послідовність точок доставки та аналізувати ефективність поїздок. Це допомагає зменшити дублювання маршрутів, скоротити час у дорозі та зробити процес доставки більш керованим.

Перевагою Routific є зручність використання та орієнтація саме на маршрутизацію. Система не перевантажена зайвими корпоративними функціями, тому може бути зрозумілою для логістів і диспетчерів. Також важливо, що маршрутні сервіси такого типу дозволяють швидко отримати практичний результат у вигляді оптимізованого маршруту.

Водночас Routific має обмеження. Він більше сфокусований саме на плануванні маршрутів, а не на повному обліку логістичних процесів підприємства. Наприклад, для системи LogiTrack важливо працювати не лише з маршрутами, а й з доставками, складами, транспортом, користувачами, ролями доступу, аналітикою і ризиками затримки. У Routific такі можливості можуть бути обмеженими або потребувати додаткової інтеграції.

Таким чином, Routific є корисним аналогом у частині планування маршрутів, але не повністю відповідає задачі розробки централізованої платформи моніторингу логістики.

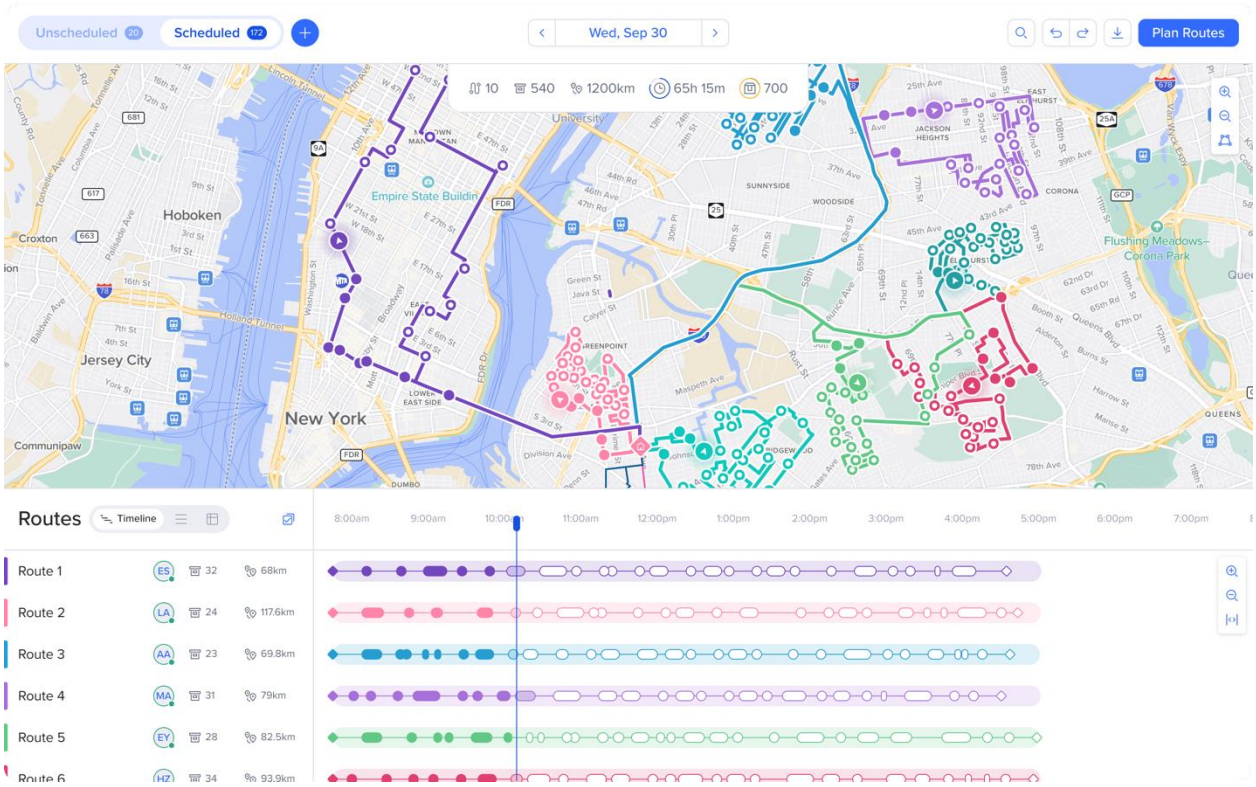


Рис. 2.4 Приклад інтерфейсу Routific для планування маршрутів

2.5 OptimoRoute

OptimoRoute є програмним засобом для планування маршрутів, керування доставкою та контролю роботи водіїв. Система дозволяє диспетчерам формувати маршрути, відстежувати виконання завдань, переглядати очікуваний час прибуття та контролювати статуси доставок.

OptimoRoute може використовуватися службами доставки, кур’єрськими компаніями, виїзними сервісними командами та підприємствами, які виконують багато поїздок протягом дня. Однією з переваг системи є можливість відображення інформації про доставку в зручному інтерфейсі, робота з ETA та підтримка сповіщень для клієнтів.

Перевагою OptimoRoute є те, що він поєднує планування маршрутів і контроль виконання доставки. Для логістичного відділу це важливо, оскільки дозволяє бачити не лише заплановані рейси, а й поточний стан виконання. Також система може бути корисною для оцінки завантаженості водіїв і контролю проблемних ситуацій.

До недоліків можна віднести залежність від зовнішнього сервісу, тарифні обмеження та недостатню гнучкість у випадку, якщо підприємство має власну специфічну структуру логістичних процесів. Крім того, OptimoRoute не є повноцінною внутрішньою платформою для роботи з користувачами, довідниками складів, ролями доступу та власним алгоритмом оцінювання ризиків.

У порівнянні з OptimoRoute, LogiTrack має бути більш адаптованим під навчальний проєкт і конкретну задачу бакалаврської роботи. Основний акцент у LogiTrack робиться не лише на доставках і маршрутах, а й на модулі Route Risk Radar, який допомагає виділяти рейси з підвищеним ризиком затримки.

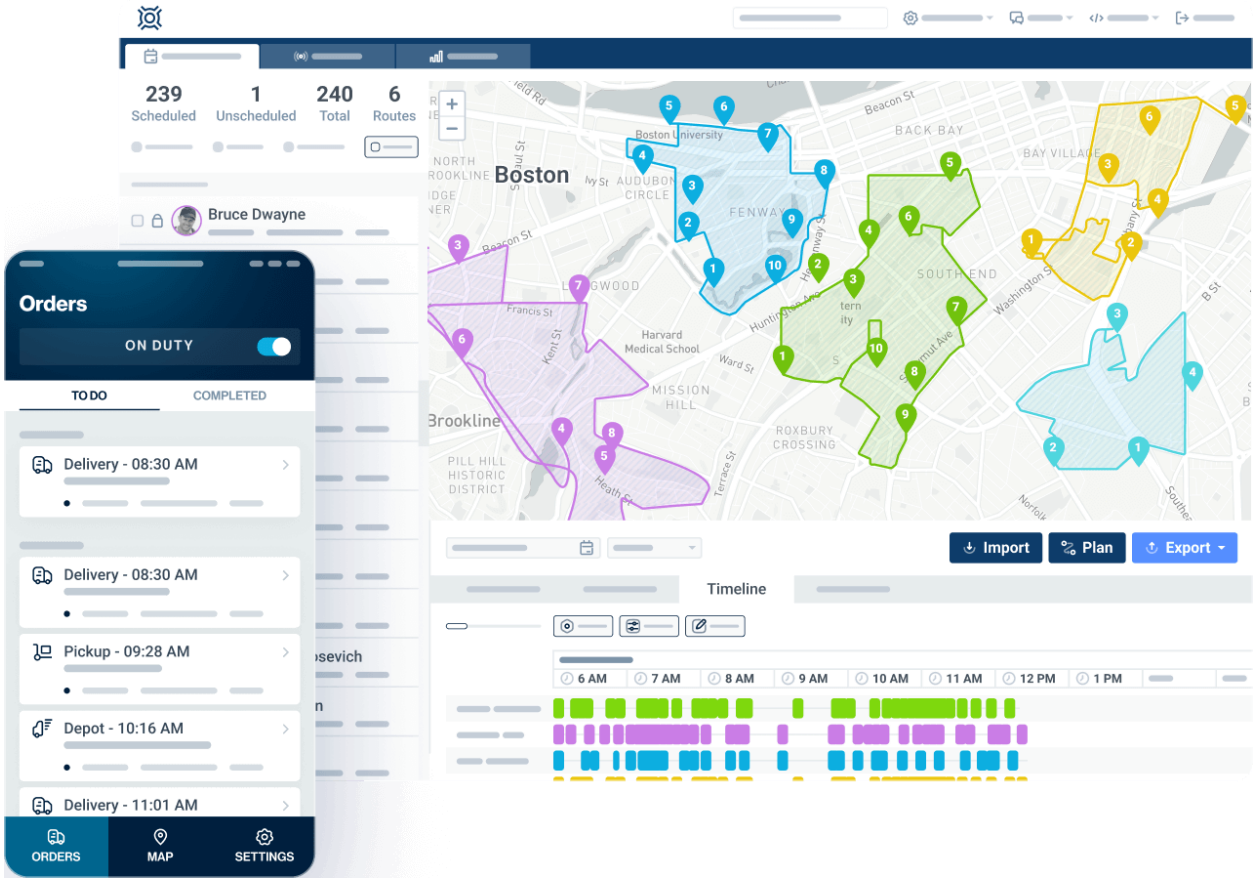


Рис. 2.5 Приклад інтерфейсу OptimoRoute для контролю доставки

2.6 Zoho Creator Logistics

Zoho Creator Logistics можна розглядати як платформу для створення індивідуальних логістичних додатків. На відміну від повністю готових TMS-

систем, Zoho Creator є low-code платформою, яка дозволяє будувати власні форми, таблиці, процеси, звіти та інтеграції. Це робить її гнучким інструментом для компаній, які хочуть створити систему під власні бізнес-процеси.

У сфері логістики Zoho Creator може використовуватися для обліку доставок, контролю складів, керування відправленнями, відстеження статусів і формування звітності. Перевагою такого підходу є можливість адаптувати структуру системи під конкретні потреби підприємства без повної розробки програмного забезпечення з нуля.

До переваг Zoho Creator можна віднести гнучкість, можливість швидкого створення форм, підтримку інтеграцій і наявність інструментів для роботи з даними. Це рішення може бути зручним для малого та середнього бізнесу, який хоче автоматизувати окремі процеси без впровадження важких корпоративних систем.

Однак така платформа також має обмеження. Насамперед користувач залежить від можливостей зовнішнього сервісу, його тарифів і технічних обмежень. Якщо потрібно реалізувати власну логіку оцінювання ризиків, окрему рольову модель або специфічну структуру REST API, low-code платформа може виявитися недостатньо гнучкою. Крім того, повний контроль над архітектурою, базою даних і програмним кодом у такому випадку обмежений.

Саме тому для розробки LogiTrack доцільно використовувати власну клієнт-серверну архітектуру на основі Java, Spring Boot, PostgreSQL, React і TypeScript. Це дозволяє реалізувати необхідну бізнес-логіку, власний модуль Route Risk Radar і гнучке розмежування прав доступу.

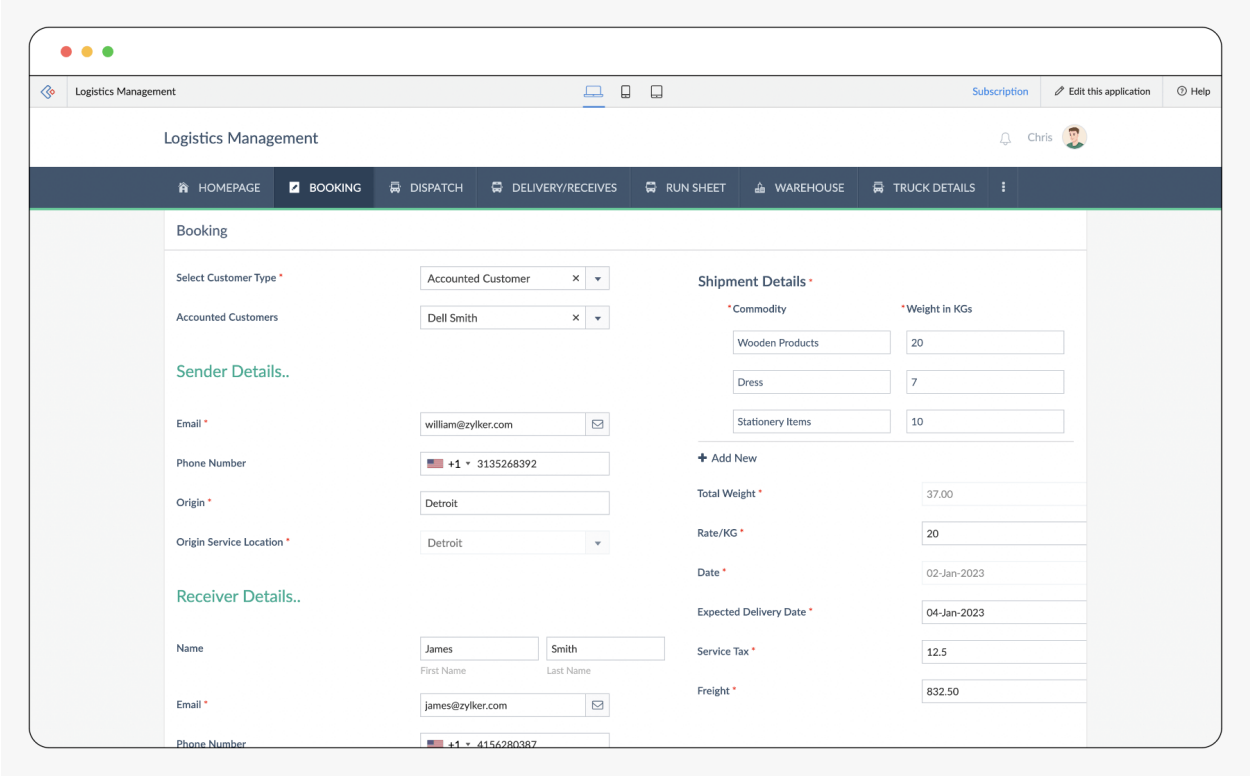


Рис. 2.6 Приклад інтерфейсу Zoho Creator для керування логістичними процесами

2.7 Порівняльний аналіз програмних засобів

Після розгляду програмних аналогів можна зробити висновок, що кожне рішення має власну сферу застосування. Excel і Google Sheets підходять для простого ручного обліку, але не забезпечують повної автоматизації. SAP Transportation Management та Oracle Transportation Management є потужними корпоративними системами, однак вони складні та дорогі для впровадження. Routific і OptimoRoute добре вирішують задачі маршрутизації та контролю доставки, але не охоплюють усю структуру логістичного моніторингу. Zoho Creator є гнучким інструментом для побудови власних додатків, але залежить від зовнішньої платформи.

У межах даної кваліфікаційної роботи доцільно розробити власну платформу LogiTrack, яка буде поєднувати потрібні функції без надлишкової складності. Основною перевагою такого підходу є можливість адаптувати систему під конкретні задачі: контроль доставок, маршрути, склади, транспорт, аналітику, рольовий доступ і автоматичне оцінювання ризику затримки.

Таблиця 2.1

Порівняльна характеристика програмних аналогів

Показник	**SAP Transportation Management**	**Oracle Transportation Management**	**Routific**	**OptimoRoute**	**Zoho Creator Logistics**	**LogiTrack**
Облік доставок	+	+	+	+	+	+
Управління маршрутами	+	+	+	+	+	+
Облік транспорту	+	+	Частково	Частково	+	+
Облік складів	+	+	-	-	+	+
Контроль статусів доставки	+	+	+	+	+	+
Аналітика та звітність	+	+	Частково	Частково	+	+
Рольовий доступ	+	+	Частково	Частково	+	+
Гнучкість адаптації	Висока, але складна	Висока, але складна	Обмежена	Обмежена	Висока	Висока

Продовження таблиці 2.1

Порівняльна характеристика програмних аналогів

Показник	**SAP Transportation Management**	**Oracle Transportation Management**	**Routific**	**OptimoRoute**	**Zoho Creator Logistics**	**LogiTrack**
Простота впровадження	-	-	+	+	Середня	Середня
Оцінювання ризику доставки	Частково	Частково	-	Частково	Частково	+
Власний модуль Route Risk Radar	-	-	-	-	-	+
Основні недоліки	Висока складність і вартість	Складне впровадження	Обмеження поза маршрутизацією	Залежність від зовнішнього сервісу	Залежність від low-code платформи	Потребує власної розробки

Порівняльна таблиця показує, що LogiTrack не є прямою заміною великих корпоративних систем, але має перевагу для конкретної задачі бакалаврської роботи. Платформа орієнтована на централізований моніторинг логістики, зрозумілу рольову модель і окремий модуль оцінювання ризиків доставки.

2.8 Функціональні та нефункціональні вимоги до платформи LogiTrack

На основі аналізу предметної галузі та програмних аналогів можна сформулювати основні вимоги до web-платформи LogiTrack. Вимоги поділяються на функціональні та нефункціональні. Функціональні вимоги визначають, які можливості повинна мати система. Нефункціональні вимоги описують якість роботи системи, її продуктивність, безпеку, зручність і сумісність.

Таблиця 2.2

Функціональні вимоги до платформи LogiTrack

№	Вимога	Опис
1	Авторизація користувачів	Система повинна забезпечувати вхід користувача за email і паролем.
2	Рольовий доступ	У системі повинні бути ролі ADMIN, MANAGER та VIEWER.
3	Керування користувачами	ADMIN повинен мати можливість переглядати користувачів, змінювати ролі та активність облікових записів.
4	Профіль користувача	Користувач повинен мати сторінку профілю з можливістю оновлення телефону, посади, відділу та кольору аватара.
5	Керування доставками	ADMIN і MANAGER повинні мати можливість створювати та редагувати доставки.
6	Видалення доставок	Видалення доставок повинно бути доступне лише ADMIN.
7	Робота зі статусами	Система повинна підтримувати статуси доставки та можливість їх оновлення.
8	Керування довідниками	У системі повинні бути довідники складів, транспорту та маршрутів.
9	Аналітика	Система повинна відображати основні показники логістичних процесів.
10	Route Risk Radar	Система повинна автоматично оцінювати ризик затримки доставки та показувати причини ризику.
11	Рекомендовані дії	Для ризикових доставок система повинна формувати рекомендовану дію для логіста.
12	Пошук і фільтрація	Система повинна підтримувати фільтрацію доставок за статусом, маршрутом, пріоритетом або рівнем ризику.

Таблиця 2.3

Нефункціональні вимоги до платформи LogiTrack

№	Вимога	Кількісне уточнення
1	Швидкодія	Основні сторінки та запити повинні виконуватися за 2–3 секунди.
2	Кількість користувачів	Система повинна підтримувати одночасну роботу не менше 20 активних користувачів.
3	Обсяг даних	База даних повинна підтримувати збереження не менше 10 000 записів доставок.
4	Сумісність	Web-платформа повинна коректно працювати в Chrome, Firefox та Microsoft Edge.
5	Зручність використання	Основні дії користувача повинні виконуватися не більше ніж за 3–5 кроків.
6	Безпека	Паролі користувачів повинні зберігатися у захищеному вигляді.
7	Контроль доступу	Функції створення, редагування, видалення та перегляду повинні залежати від ролі користувача.
8	Масштабованість	Архітектура повинна дозволяти додавання нових модулів, наприклад ETA Simulator або GPS-моніторингу.
9	Надійність даних	Дані повинні зберігатися в PostgreSQL без втрати після оновлення сторінки або повторного входу в систему.
10	Зрозумілість аналітики	Ризик доставки повинен відображатися у вигляді riskScore 0–100 та рівня LOW / MEDIUM / HIGH / CRITICAL.

Сформульовані вимоги є основою для подальшого проєктування web-платформи LogiTrack. Вони визначають, які модулі повинні бути реалізовані, які ролі користувачів передбачені в системі, як має працювати Route Risk Radar і за якими критеріями буде оцінюватися якість реалізованого програмного забезпечення.

3 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ WEB-ПЛАТФОРМИ LOGITRACK

3.1 Логічна архітектура web-платформи

Для реалізації web-платформи LogiTrack доцільно використати клієнт-серверну архітектуру. Такий підхід дозволяє розділити систему на декілька логічних частин: клієнтський інтерфейс, серверну частину, базу даних і додаткові сервіси. Це спрощує розробку, тестування, подальше розширення функціональності та підтримку програмного забезпечення.

Клієнтська частина відповідає за взаємодію користувача із системою. Саме через неї користувач авторизується, переглядає dashboard, працює з доставками, маршрутами, довідниками, аналітикою, профілем і модулем Route Risk Radar. У web-платформі LogiTrack клієнтська частина повинна бути зручною для різних ролей користувачів: ADMIN, MANAGER та VIEWER. Тому інтерфейс має відображати лише ті кнопки, сторінки та дії, які дозволені конкретній ролі.

Серверна частина відповідає за бізнес-логіку системи. Вона обробляє запити від клієнта, перевіряє права доступу, працює з даними користувачів, доставок, складів, маршрутів, транспортних засобів і виконує розрахунок ризику затримки доставки. Саме на серверному рівні доцільно реалізувати основний алгоритм Route Risk Radar, оскільки він повинен працювати з даними з різних сутностей: доставки, маршруту, транспорту та складу.

База даних використовується для централізованого збереження інформації. У LogiTrack вона повинна містити дані про користувачів, ролі, доставки, маршрути, склади, транспорт, статуси, пріоритети та інші пов’язані сутності. Використання реляційної бази даних є доцільним, тому що предметна галузь має чіткі зв’язки між об’єктами: доставка пов’язана з маршрутом, транспортом, складом, відповідальним користувачем і статусом.

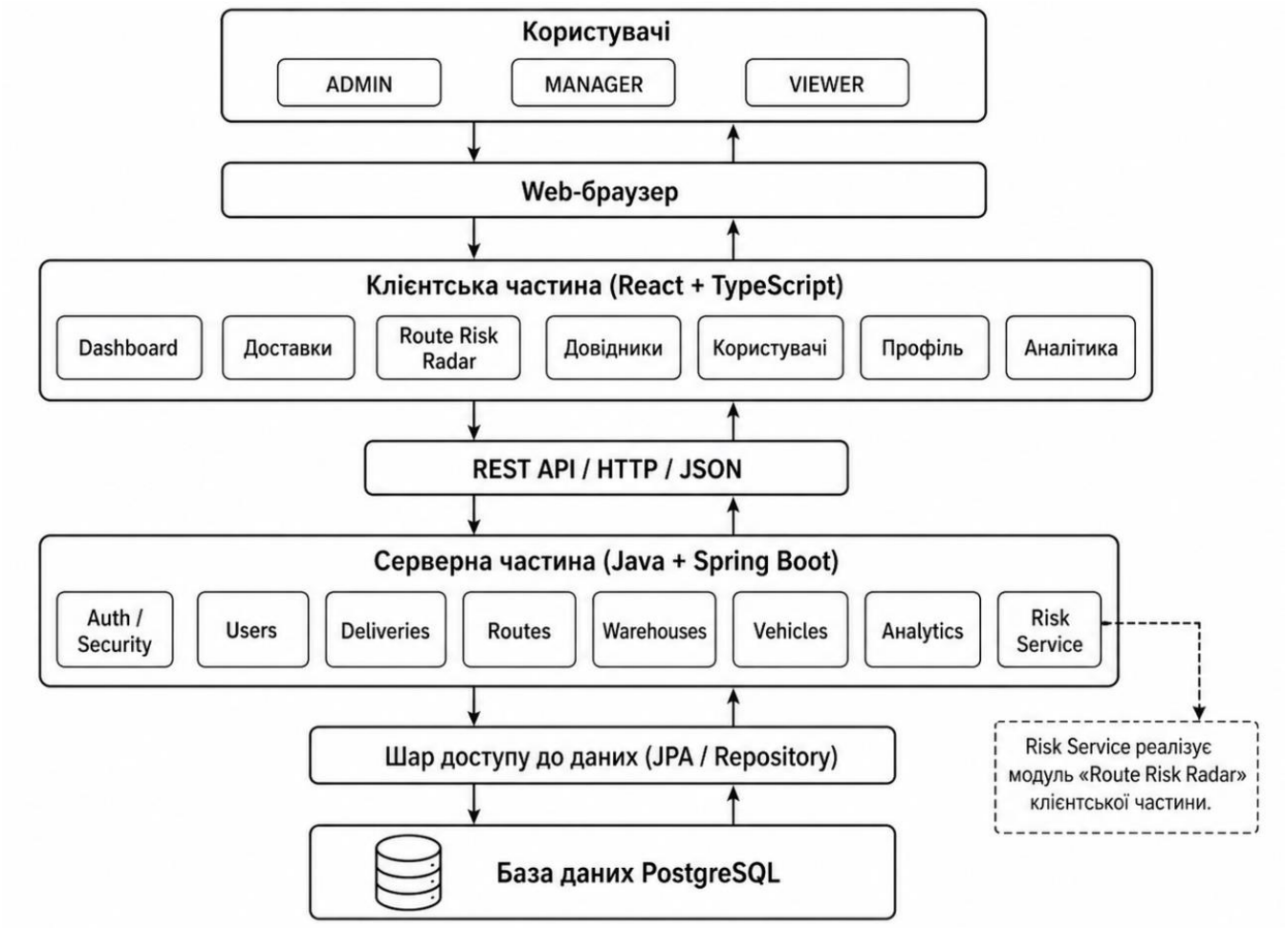


Рис. 3.1 Загальна архітектура платформи LogiTrack

У загальному вигляді архітектура LogiTrack передбачає таку взаємодію: користувач працює з клієнтською частиною через браузер, клієнтська частина надсилає HTTP-запити до серверної частини через REST API, сервер обробляє ці запити, виконує перевірку доступу, працює з базою даних PostgreSQL і повертає відповідь у форматі JSON. Така структура є зрозумілою, поширеною і добре підходить для web-застосунків з кількома ролями користувачів.

3.2 Мова програмування Java

Для реалізації серверної частини web-платформи LogiTrack обрано мову програмування Java. Java є об’єктно-орієнтованою мовою програмування, яка широко використовується для створення серверних, корпоративних і web-

застосунків. Її перевагою є стабільність, велика кількість бібліотек, підтримка багатошарової архітектури та зручність роботи зі складними бізнес-сутностями [4].

У межах LogiTrack Java підходить для опису основних об’єктів предметної галузі: користувачів, доставок, маршрутів, складів, транспортних засобів і результатів оцінювання ризику. Кожна сутність може бути представлена окремим класом, що робить структуру програми більш зрозумілою і зручною для подальшої підтримки.

Також Java дозволяє добре організувати серверну логіку через поділ на контролери, сервіси, репозиторії, DTO-класи та класи сутностей. Для web-платформи LogiTrack це важливо, тому що система має не лише виконувати CRUD-операції, а й реалізовувати додаткову логіку: перевірку ролей користувачів, активність облікових записів, розрахунок riskScore, формування причин ризику та рекомендованих дій.

Ще однією перевагою Java є можливість використання готових інструментів для роботи з базами даних, безпекою, REST API та тестуванням. Це дозволяє не реалізовувати всі механізми вручну, а використовувати перевірені рішення, що зменшує кількість помилок у коді.

У даній роботі Java використовується як основна мова серверної частини, оскільки вона забезпечує надійність, структурованість і можливість подальшого розширення платформи.

3.3 Spring Boot та Spring Framework

Для створення серверної частини web-платформи LogiTrack доцільно використовувати Spring Boot. Spring Boot є частиною екосистеми Spring і спрощує розробку серверних застосунків мовою Java. Він дозволяє швидко створювати REST API, підключати базу даних, налаштовувати залежності, працювати з безпекою та організовувати багаторівневу структуру проєкту [5].

У LogiTrack Spring Boot використовується для побудови серверної частини, яка обробляє запити клієнтського інтерфейсу. Наприклад, клієнтська частина може

надсилати запит на отримання списку доставок, зміну статусу, перегляд профілю користувача, створення маршруту або отримання списку ризикових рейсів. Серверна частина приймає ці запити, виконує необхідну логіку і повертає відповідь клієнту.

Структура серверної частини може бути поділена на декілька рівнів. Контролери відповідають за приймання HTTP-запитів. Сервіси містять основну бізнес-логіку. Репозиторії забезпечують роботу з базою даних. DTO-класи використовуються для передавання даних між клієнтською та серверною частинами. Такий поділ дозволяє зробити код більш чистим і зрозумілим.

У межах LogiTrack доцільно виділити такі основні серверні модулі:

– модуль авторизації та автентифікації;

– модуль користувачів;

– модуль доставок;

– модуль складів;

– модуль транспортних засобів;

– модуль маршрутів;

– модуль аналітики;

– модуль Route Risk Radar.

Spring Boot також зручно використовувати для реалізації REST API. У системі можуть бути передбачені такі endpoint-и: /api/auth, /api/users, /api/users/me, /api/deliveries, /api/warehouses, /api/vehicles, /api/routes, /api/analytics, /api/risks. Така структура робить серверну частину зрозумілою і дозволяє клієнтській частині отримувати необхідні дані через стандартизовані запити.

3.4 PostgreSQL

Для збереження даних web-платформи LogiTrack обрано PostgreSQL. Це об’єктно-реляційна система керування базами даних з відкритим кодом, яка використовується для роботи зі структурованими даними та підтримує мову SQL [6].

PostgreSQL підходить для даної системи, оскільки логістична предметна галузь має чітко визначені сутності та зв’язки між ними. Наприклад, користувач може бути відповідальним за доставку, доставка може бути пов’язана з маршрутом, транспортним засобом і складом, а маршрут може мати відсоток завантаження. Така структура добре описується у вигляді реляційної моделі.

У базі даних LogiTrack доцільно передбачити такі основні таблиці:

– users — дані користувачів системи;

– deliveries — інформація про доставки;

– routes — маршрути доставки;

– vehicles — транспортні засоби;

– warehouses — склади;

– delivery_notes або delivery_history — нотатки та історія змін доставки;

– risk_results — результати оцінювання ризику, якщо вони зберігаються окремо.

Окрему увагу потрібно приділити таблиці користувачів. З урахуванням реалізації LogiTrack вона повинна містити не лише email і пароль, а й додаткові поля: ім’я, телефон, посаду, відділ, колір аватара, роль, активність користувача, дату створення та дату оновлення. Це дозволяє реалізувати повноцінний профіль користувача і панель адміністрування.

PostgreSQL також підходить для подальшого формування аналітики. Наприклад, на основі даних з таблиць доставок, маршрутів і складів можна обчислювати кількість активних доставок, відсоток затримок, завантаженість маршрутів, кількість критичних рейсів і частку доставок з високим ризиком.

3.5 React та TypeScript

Для реалізації клієнтської частини web-платформи LogiTrack обрано React і TypeScript. React є бібліотекою JavaScript для створення користувацьких інтерфейсів на основі компонентного підходу [7]. TypeScript, у свою чергу,

розширює JavaScript за рахунок статичної типізації, що дозволяє зменшити кількість помилок під час розробки [8].

React добре підходить для LogiTrack, тому що інтерфейс системи складається з багатьох повторюваних елементів: таблиць, форм, карток, фільтрів, панелей статистики, кнопок, списків і модальних вікон. Кожен такий елемент можна реалізувати як окремий компонент. Наприклад, окремими компонентами можуть бути таблиця доставок, картка ризику, форма редагування профілю, блок аналітики або вкладки довідників.

TypeScript є корисним для роботи з даними, які надходять із серверної частини. Наприклад, доставка, користувач, маршрут або результат оцінки ризику мають визначену структуру. Якщо описати ці структури у вигляді типів або інтерфейсів, розробнику простіше контролювати, які поля використовуються в клієнтській частині. Це особливо важливо для модуля Route Risk Radar, де потрібно працювати з riskScore, riskLevel, списком причин ризику та рекомендованою дією.

У клієнтській частині LogiTrack можна реалізувати такі сторінки:

– сторінка авторизації;

– dashboard;

– сторінка доставок;

– сторінка Route Risk Radar;

– сторінка довідників;

– сторінка користувачів для адміністратора;

– сторінка профілю;

– сторінка аналітики.

Важливою частиною клієнтської логіки є врахування ролей користувачів. Наприклад, VIEWER не повинен бачити кнопки створення, редагування або видалення. MANAGER може працювати з доставками, але не може змінювати ролі користувачів. ADMIN має доступ до всіх сторінок і функцій. Такий підхід реалізується через перевірку ролі користувача в контексті авторизації.

3.6 REST API та взаємодія клієнтської і серверної частин

Для обміну даними між клієнтською і серверною частинами web-платформи LogiTrack використовується REST API. REST API дозволяє клієнтській частині надсилати HTTP-запити до серверної частини, а серверна частина повертає відповідь у структурованому форматі JSON.

Такий підхід є зручним, оскільки дозволяє чітко розділити інтерфейс користувача і серверну логіку. React відповідає за відображення даних і взаємодію з користувачем, а Spring Boot — за обробку запитів, перевірку прав доступу, роботу з базою даних і виконання бізнес-логіки.

Особливу роль REST API має для модуля Route Risk Radar. Клієнтська частина може звертатися до endpoint-а /api/risks, а серверна частина формує список доставок, розраховує для них riskScore, визначає riskLevel, причини ризику та рекомендовану дію. Після цього клієнтська частина відображає отримані результати у зручному вигляді.

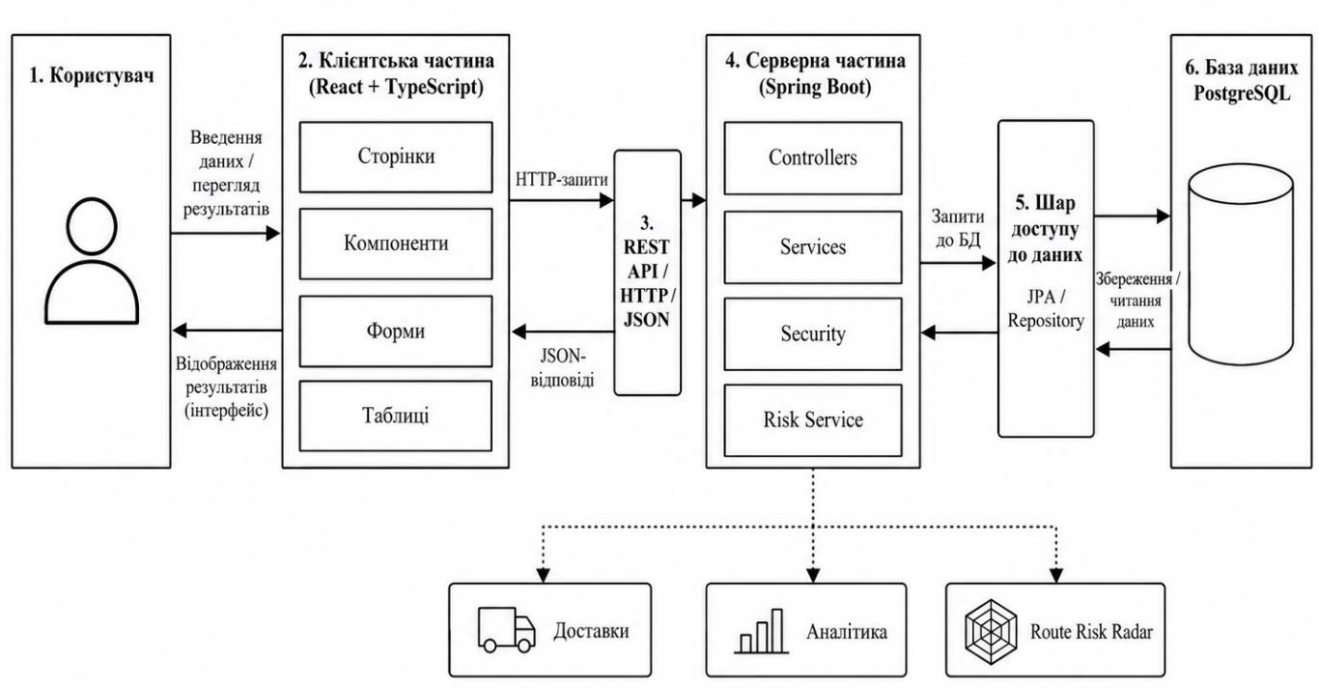


Рис. 3.2 Взаємодія клієнтської частини, серверної частини та бази даних

3.7 Засоби безпеки та рольового доступу

Оскільки web-платформа LogiTrack працює з логістичними даними, користувачами, статусами доставок і внутрішніми довідниками підприємства, важливо забезпечити контроль доступу до функцій системи. Для цього в системі використовується рольова модель доступу.

У LogiTrack передбачено три основні ролі: ADMIN, MANAGER та VIEWER. Кожна роль має власний набір дозволених дій. ADMIN має повний доступ до системи, може керувати користувачами, ролями, довідниками, доставками та аналітикою. MANAGER може працювати з доставками, змінювати статуси, ETA, нотатки, переглядати довідники, аналітику і Route Risk Radar. VIEWER має лише права перегляду і не може змінювати дані.

На серверному рівні контроль доступу доцільно реалізувати через перевірку ролей для окремих endpoint-ів. Наприклад, endpoint-и для керування користувачами повинні бути доступні лише ADMIN. Створення та редагування доставок можуть бути доступні ADMIN і MANAGER. Видалення записів має бути дозволене тільки ADMIN. Перегляд основної інформації може бути доступний усім авторизованим користувачам.

Крім рольового доступу, важливим є захищене зберігання паролів. Паролі користувачів не повинні зберігатися у відкритому вигляді. Також доцільно використовувати токен авторизації, наприклад JWT, який дозволяє перевіряти, чи має користувач право виконувати певні дії.

На клієнтському рівні також потрібно враховувати роль користувача. Якщо користувач має роль VIEWER, інтерфейс не повинен показувати йому кнопки створення, редагування або видалення. Якщо користувач має роль MANAGER, він може бачити робочі дії з доставками, але не повинен мати доступу до панелі керування користувачами. Це робить інтерфейс не лише безпечнішим, а й простішим для користувача.

4 ПРОЄКТУВАННЯ WEB-ПЛАТФОРМИ LOGITRACK

4.1 Логічна структура системи

Проєктування web-платформи LogiTrack передбачає визначення її загальної логіки роботи, основних модулів, ролей користувачів, структури бази даних та взаємодії між клієнтською і серверною частинами. На цьому етапі важливо не лише описати, які функції має виконувати система, а й показати, як ці функції пов’язані між собою.

LogiTrack розробляється як web-платформа для централізованого моніторингу логістичних процесів. Основними об’єктами системи є доставки, маршрути, склади, транспортні засоби, користувачі та результати оцінювання ризиків. Усі ці об’єкти мають бути пов’язані між собою, оскільки доставка не існує окремо від маршруту, транспорту, відповідального користувача або складу.

Логічно систему можна поділити на декілька основних функціональних блоків:

– блок авторизації та профілю користувача;

– блок керування доставками;

– блок довідників складів, транспорту та маршрутів;

– блок аналітики;

– блок Route Risk Radar;

– блок адміністрування користувачів;

– блок рольового доступу.

Перший блок забезпечує вхід користувача до системи, збереження даних профілю, перевірку ролі та відображення доступних функцій. Другий блок відповідає за створення, перегляд, редагування та контроль доставок. Третій блок потрібен для роботи з довідковими даними: складами, транспортними засобами та маршрутами. Четвертий блок формує аналітичні показники. П’ятий блок виконує

оцінювання ризику затримки доставки. Шостий блок призначений для адміністратора, який керує користувачами та їх активністю.

Особливе місце в логічній структурі займає модуль Route Risk Radar. Він не є звичайним довідником або CRUD-модулем, оскільки виконує обчислення на основі даних з різних частин системи. Для оцінки ризику використовуються такі параметри, як поточна затримка, завантаження маршруту, пріоритет доставки, стан транспортного засобу, завантаженість складу та статус доставки. Це дозволяє системі не просто зберігати інформацію, а формувати операційний індикатор для логіста.

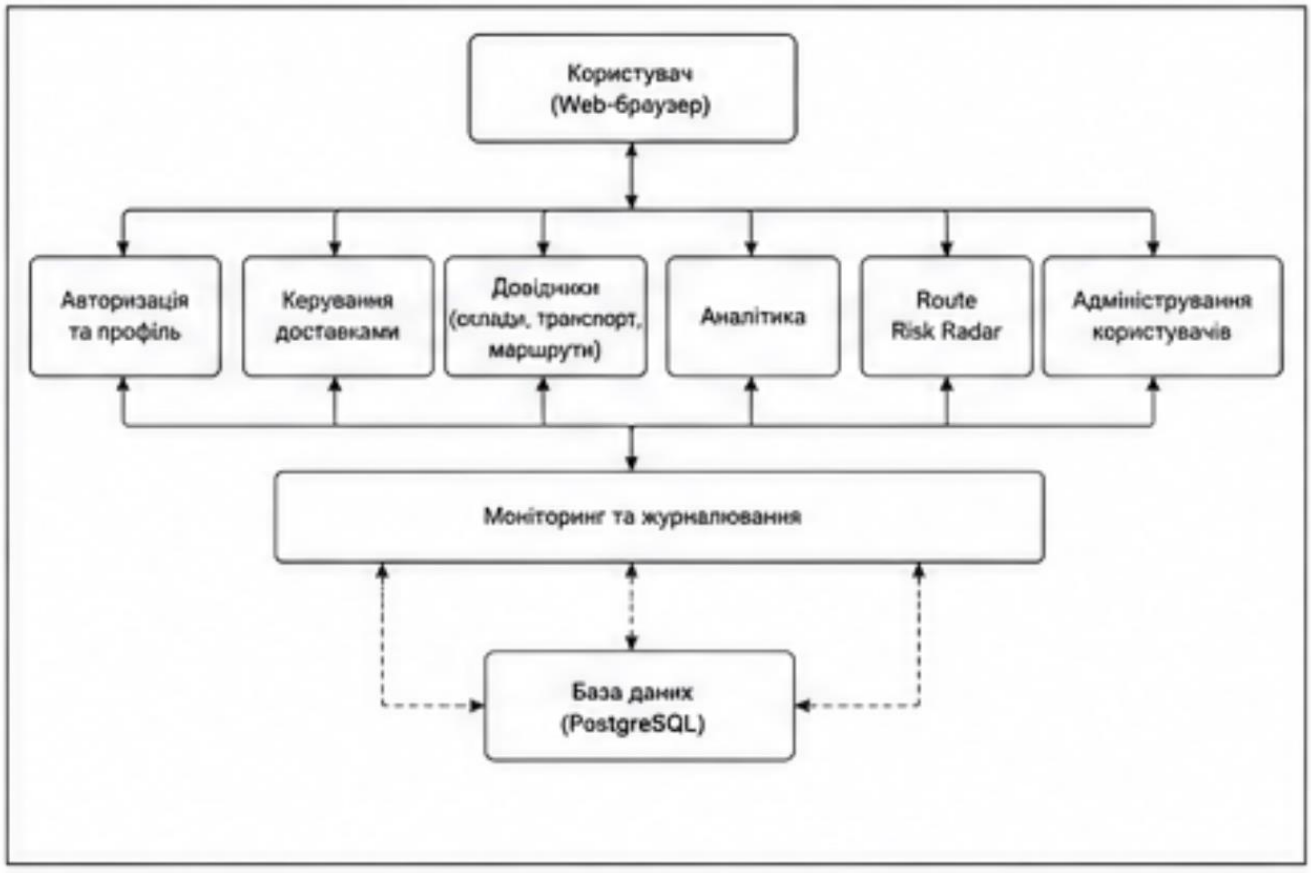


Рис. 4.1 – Логічна структура web-платформи LogiTrack

У межах логічної структури користувач взаємодіє з інтерфейсом системи через web-браузер. Залежно від ролі користувач бачить різний набір сторінок і дій. Наприклад, ADMIN має доступ до всіх модулів, MANAGER працює переважно з

доставками, довідниками, аналітикою та ризиками, а VIEWER може лише переглядати інформацію без зміни даних.

Такий підхід дозволяє зробити систему безпечнішою та зрозумілішою, оскільки кожен користувач працює тільки з тими функціями, які відповідають його обов’язкам.

4.2 Загальна архітектура платформи LogiTrack

Архітектура web-платформи LogiTrack побудована за клієнт-серверним принципом. Це означає, що система складається з клієнтської частини, серверної частини та бази даних. Кожен із цих елементів виконує свою роль і взаємодіє з іншими через визначені механізми.

Клієнтська частина реалізується за допомогою React і TypeScript. Вона відповідає за відображення інтерфейсу, обробку дій користувача, надсилання запитів до сервера та відображення отриманих результатів. Саме на клієнтській частині розміщуються сторінки dashboard, доставок, довідників, аналітики, профілю, користувачів та Route Risk Radar.

Серверна частина реалізується за допомогою Java та Spring Boot. Вона відповідає за бізнес-логіку системи, перевірку прав доступу, обробку запитів, взаємодію з базою даних і формування відповідей для клієнтської частини. На серверному рівні реалізуються контролери, сервіси, репозиторії, сутності, DTO-класи та механізми безпеки.

База даних PostgreSQL використовується для збереження структурованих даних системи. У ній зберігаються користувачі, доставки, маршрути, склади, транспортні засоби, статуси, пріоритети та інші дані, необхідні для роботи платформи. Реляційна структура бази даних дозволяє забезпечити зв’язки між сутностями та підтримувати цілісність інформації.

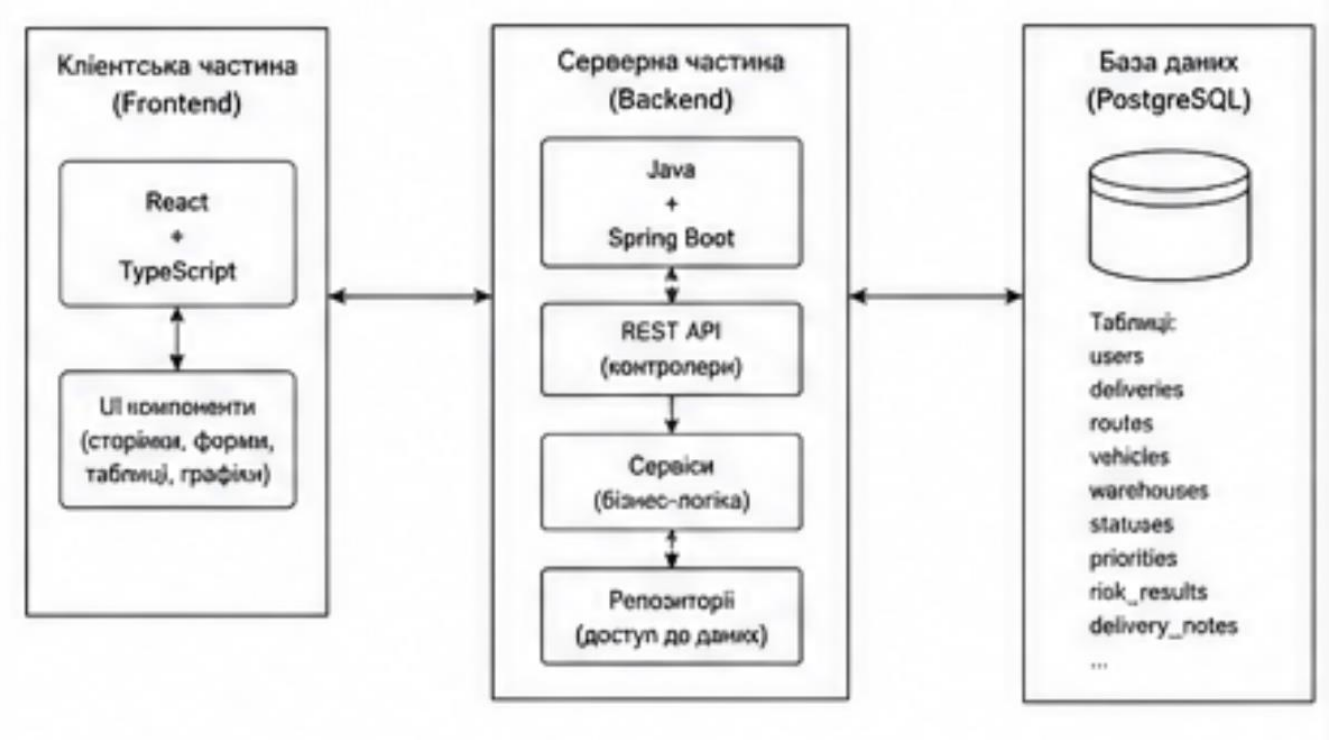


Рис. 4.2 – Загальна архітектура web-платформи LogiTrack

Загальна архітектура системи дозволяє відокремити логіку інтерфейсу від серверної логіки. Це є важливим, оскільки зміни в зовнішньому вигляді сторінок не повинні прямо впливати на структуру бази даних або серверні сервіси. Аналогічно, зміни в алгоритмі розрахунку ризику не повинні вимагати повного переписування клієнтської частини.

Крім того, така архітектура дає можливість у майбутньому розширювати систему. Наприклад, до серверної частини можна підключити мобільний додаток, GPS-моніторинг, зовнішню TMS-систему або модуль прогнозування ETA.

4.3 Проєктування бази даних

База даних є одним із ключових елементів web-платформи LogiTrack, оскільки саме вона забезпечує збереження основної інформації системи. Під час проєктування бази даних необхідно врахувати структуру логістичних процесів, ролі користувачів, зв’язки між доставками, маршрутами, складами і транспортом, а також дані, які використовуються для оцінювання ризику доставки.

Основними сутностями бази даних є:

– користувач;

– доставка;

– маршрут;

– транспортний засіб;

– склад;

– статус доставки;

– пріоритет доставки;

– нотатка або історія змін;

– результат оцінювання ризику.

Сутність користувача зберігає дані про осіб, які працюють із платформою. Для користувача доцільно передбачити такі поля: ідентифікатор, ім’я, email, пароль, телефон, посада, відділ, роль, колір аватара, активність облікового запису, дата створення та дата оновлення. Така структура дозволяє реалізувати не лише авторизацію, а й повноцінний профіль користувача.

Сутність доставки містить основні дані про рейс або замовлення на доставку. Вона може включати номер доставки, клієнта, адресу відправлення, адресу призначення, статус, пріоритет, ETA, фактичну затримку, відповідального користувача, маршрут, склад відправлення, склад призначення і транспортний засіб.

Сутність маршруту зберігає інформацію про напрямок перевезення, наприклад «Київ – Львів» або «Дніпро – Запоріжжя». Для маршруту важливо зберігати не лише назву, а й відсоток завантаження, орієнтовний час виконання, відстань і поточний стан.

Сутність транспортного засобу описує автомобіль або інший транспорт, який використовується для доставки. Вона може включати номер, тип, вантажність, статус, водія або відповідального працівника. Стан транспорту важливий для модуля Route Risk Radar, оскільки транспорт у статусі MAINTENANCE або недоступний транспорт підвищує ризик затримки доставки.

Сутність складу описує логістичні точки, через які проходить доставка. Для складу важливо зберігати назву, адресу, поточне завантаження, максимальну

місткість і статус. Завантаженість складу також впливає на ризик затримки, оскільки перевантажений склад може повільніше обробляти замовлення.

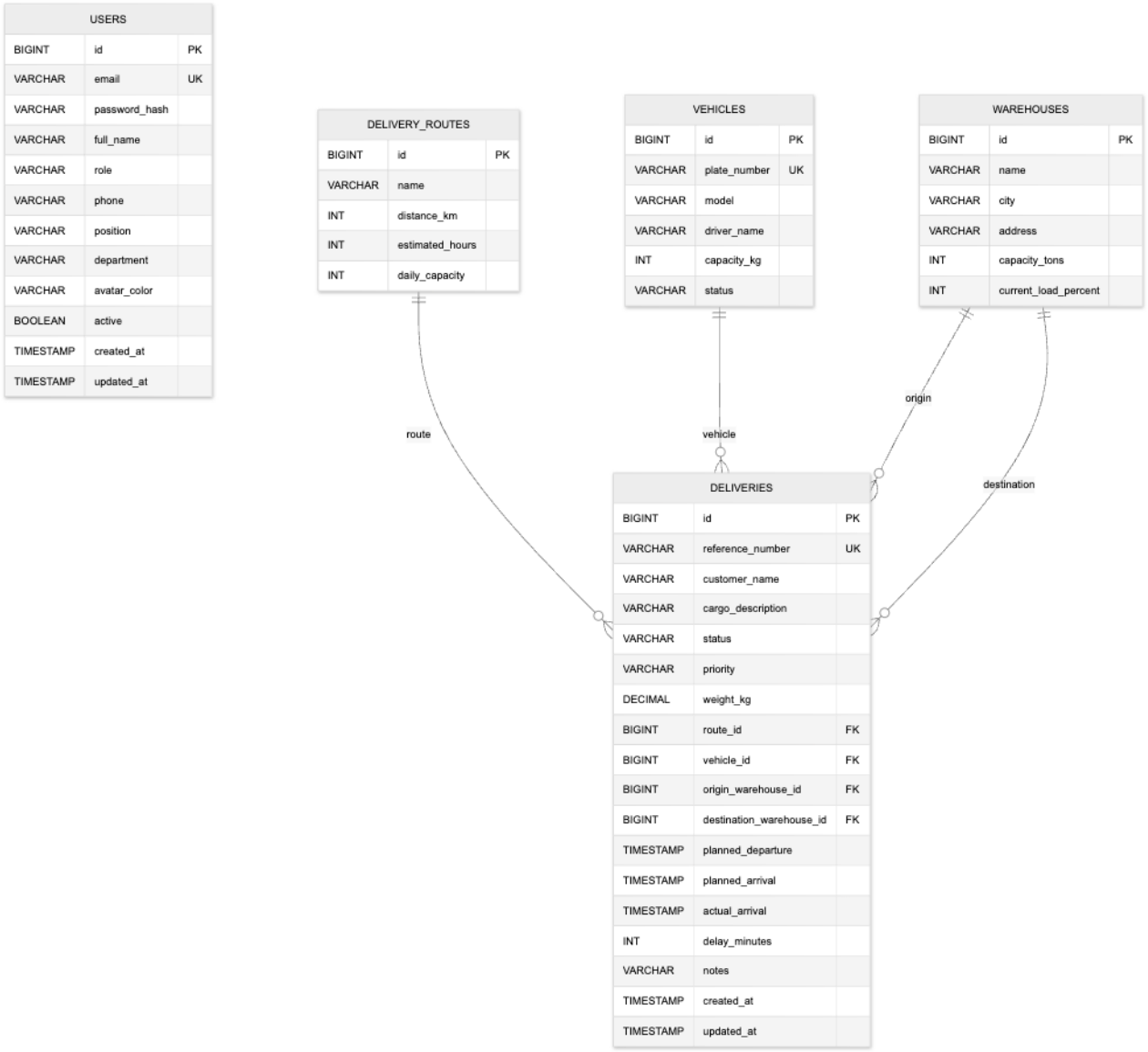


Рис. 4.3 Схема бази даних платформи LogiTrack

База даних повинна забезпечувати не лише збереження даних, а й можливість швидкої вибірки для сторінок системи. Наприклад, сторінка доставок повинна швидко отримувати список активних доставок із маршрутом, транспортом і статусом. Сторінка Route Risk Radar повинна отримувати дані з декількох пов’язаних таблиць, щоб сервер міг розрахувати ризик для кожної доставки.

4.4 UML-діаграма варіантів використання

Діаграма варіантів використання дозволяє показати, які користувачі взаємодіють із системою та які основні функції вони можуть виконувати. Для web-платформи LogiTrack основними акторами є ADMIN, MANAGER та VIEWER.

ADMIN має найширший набір можливостей. Він може керувати користувачами, змінювати ролі, активувати або вимикати облікові записи, керувати довідниками, створювати, редагувати та видаляти доставки, переглядати аналітику і працювати з Route Risk Radar.

MANAGER є основним робочим користувачем системи. Він може створювати та редагувати доставки, змінювати статуси, ETA, додавати нотатки, переглядати довідники, аналітику та сторінку ризиків. При цьому MANAGER не може видаляти системні довідники або змінювати ролі користувачів.

VIEWER має доступ лише до перегляду інформації. Він може бачити dashboard, аналітику, деталі доставок, профіль користувача та результати Route Risk Radar, але не може створювати або редагувати записи.

Діаграма варіантів використання допомагає уточнити функціональні межі системи. Вона показує, які дії є спільними для всіх користувачів, а які доступні лише окремим ролям. Це важливо для подальшої реалізації рольового доступу на серверному та клієнтському рівнях.

4.5 UML-діаграма класів web-платформи

Діаграма класів використовується для опису структури основних сутностей системи та зв’язків між ними. У web-платформі LogiTrack основними класами є User, Delivery, Route, Vehicle, Warehouse, DeliveryNote, RiskResult та допоміжні класи або enum-типи для ролей, статусів і пріоритетів.

Клас User описує користувача системи. Він містить інформацію про ім’я, email, пароль, телефон, посаду, відділ, роль і активність облікового запису. З цим

класом пов’язані дії авторизації, оновлення профілю та адміністрування користувачів.

Клас Delivery є центральним класом предметної галузі. Він описує доставку та містить зв’язки з маршрутом, транспортним засобом, складом і відповідальним користувачем. Саме на основі доставки формується більшість операцій у системі: перегляд, редагування, зміна статусу, додавання нотаток і розрахунок ризику.

Клас Route описує маршрут доставки. Він містить початкову і кінцеву точки, відсоток завантаження, орієнтовний час і додаткові параметри. Клас Vehicle описує транспортний засіб, а клас Warehouse — склад. Ці сутності використовуються як довідники, але також впливають на логіку Route Risk Radar.

Клас RiskResult або DTO DeliveryRiskResponse містить результат оцінювання ризику. Він включає riskScore, riskLevel, список причин ризику та рекомендовану дію. Такий клас може не зберігатися постійно в базі даних, а формуватися динамічно під час запиту до endpoint-а /api/risks.

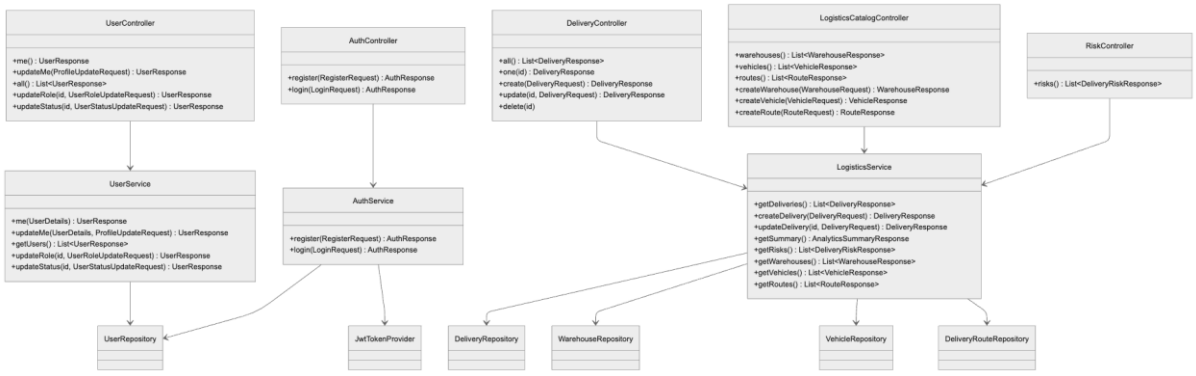


Рис. 4.4 UML-діаграма класів web-платформи LogiTrack

Діаграма класів дозволяє краще зрозуміти внутрішню структуру системи до початку програмної реалізації. Вона показує, які сутності є головними, які поля вони містять і як вони пов’язані між собою.

4.6 UML-діаграма послідовності створення доставки

Діаграма послідовності дозволяє описати порядок взаємодії між користувачем, клієнтською частиною, серверною частиною та базою даних під час виконання конкретного сценарію. Для LogiTrack одним із базових сценаріїв є створення нової доставки.

Процес починається з того, що користувач із роллю ADMIN або MANAGER відкриває сторінку доставок і натискає кнопку створення нової доставки. Клієнтська частина відкриває форму, у якій користувач вводить дані: клієнта, маршрут, склад, транспорт, пріоритет, ETA та інші параметри.

Після заповнення форми клієнтська частина виконує первинну перевірку даних. Наприклад, перевіряється, чи заповнені обов’язкові поля. Після цього формується HTTP-запит до серверної частини. Сервер приймає запит, перевіряє токен авторизації та роль користувача. Якщо користувач має право створювати доставку, сервер переходить до обробки даних.

На серверному рівні дані передаються до сервісу доставок. Сервіс перевіряє коректність маршруту, транспорту і складу, після чого формує нову сутність Delivery і передає її до репозиторію. Репозиторій зберігає запис у базі даних PostgreSQL. Після успішного збереження сервер повертає відповідь клієнтській частині.

Клієнтська частина отримує відповідь, оновлює список доставок і показує користувачу результат виконання операції. У разі помилки система повинна відобразити повідомлення, наприклад про відсутність прав доступу або некоректні дані.

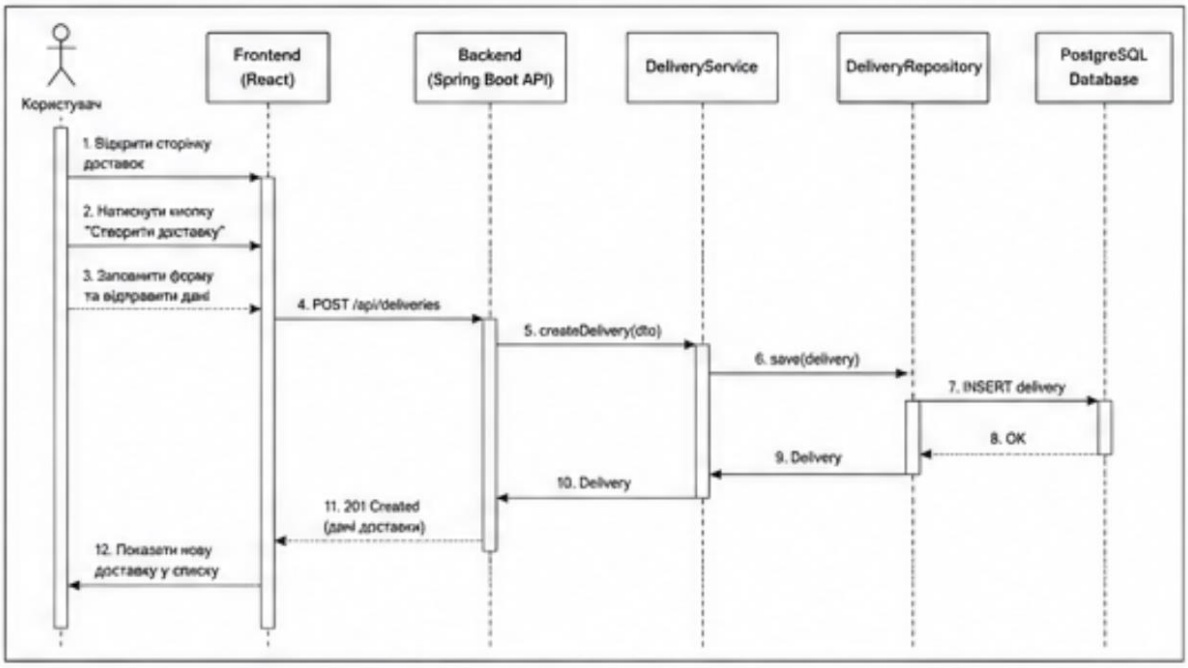


Рис. 4.5 UML-діаграма послідовності створення доставки

Такий сценарій показує, як взаємодіють усі рівні системи: інтерфейс, API, серверна логіка, репозиторій і база даних. Це важливо для розуміння принципу роботи web-платформи та перевірки правильності її архітектури.

4.7 Activity-діаграма роботи модуля Route Risk Radar

Модуль Route Risk Radar є однією з ключових особливостей web-платформи LogiTrack. Його задача полягає в автоматичному оцінюванні ризику затримки доставки на основі кількох факторів. Для опису логіки роботи цього модуля доцільно використати activity-діаграму.

Робота модуля починається з отримання списку доставок. Серверна частина завантажує дані про доставки, маршрути, транспортні засоби, склади, статуси та пріоритети. Далі для кожної доставки виконується перевірка факторів ризику.

Першим фактором є поточна затримка. Якщо delayMinutes більше нуля, система додає певну кількість балів до загального riskScore. Чим більша затримка, тим вищий ризик. Другим фактором є завантаження маршруту. Якщо route.loadPercent перевищує 90%, система також підвищує ризик.

Третім фактором є пріоритет доставки. Якщо доставка має високий або критичний пріоритет, вона повинна отримати додаткову увагу логіста, тому riskScore збільшується. Четвертим фактором є стан транспорту. Якщо транспортний засіб перебуває в ремонті або має недоступний статус, це також підвищує ризик.

П’ятим фактором є завантаженість складу. Якщо склад відправлення або склад призначення має завантаження понад 80%, це може створити додаткову затримку. Окремо враховується статус доставки. Якщо доставка вже має статус DELAYED, ризик повинен значно збільшуватися.

Після перевірки всіх факторів система формує підсумковий riskScore у межах від 0 до 100. На основі цього значення визначається рівень ризику: LOW, MEDIUM, HIGH або CRITICAL. Далі система формує список причин ризику та рекомендовану дію. Наприклад, якщо доставка має високу затримку і критичний пріоритет, рекомендованою дією може бути ручний контроль рейсу або зв’язок із клієнтом.

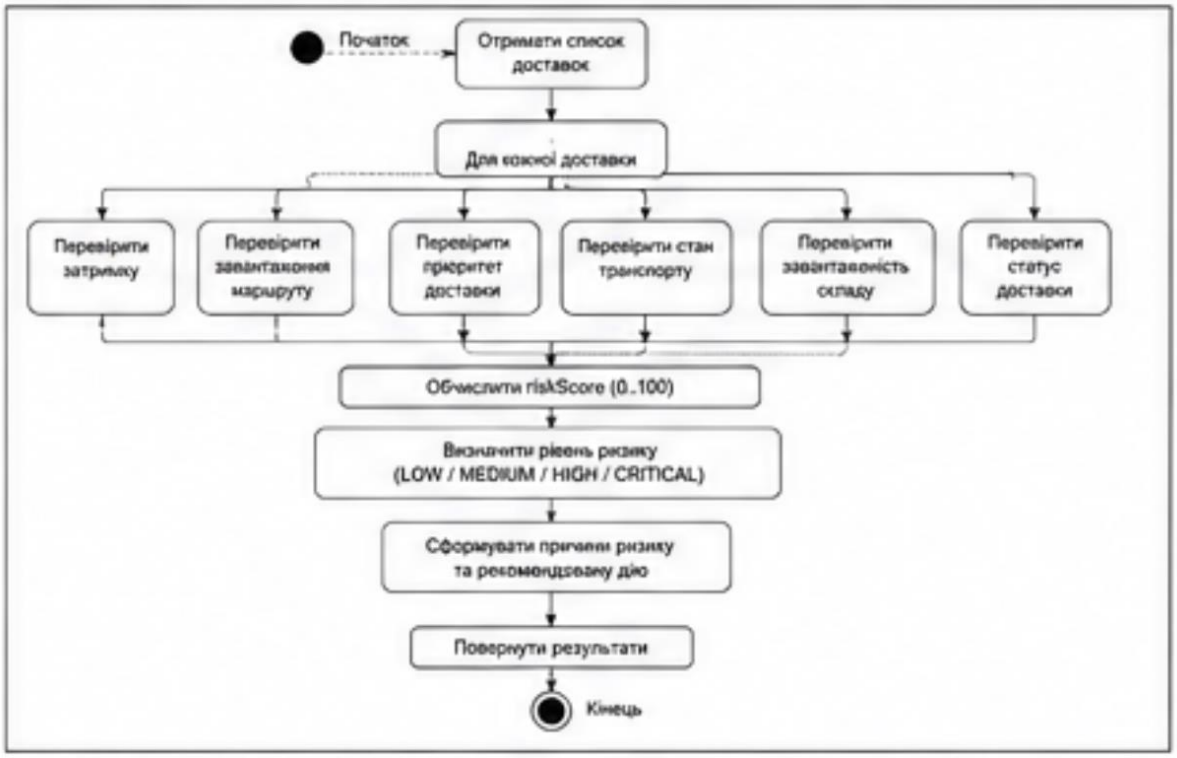


Рис. 4.6 Діаграма діяльності роботи модуля Route Risk Radar

Така логіка дозволяє зробити модуль зрозумілим для користувача. Система не просто показує числовий показник, а пояснює, чому доставка отримала певний рівень ризику. Це важливо для практичного використання, оскільки логісту потрібно розуміти причини проблеми, а не лише бачити абстрактний бал.

4.8 Структура основних модулів платформи LogiTrack

Web-платформа LogiTrack складається з кількох основних модулів, кожен із яких відповідає за окрему частину функціональності. Такий поділ дозволяє краще організувати програмний код і спростити подальше розширення системи.

Модуль авторизації відповідає за вхід користувачів, перевірку токена, визначення ролі та обмеження доступу до функцій. Він є базовим для всієї системи, оскільки без авторизації користувач не повинен отримувати доступ до внутрішніх сторінок платформи.

Модуль користувачів забезпечує роботу з обліковими записами. ADMIN може переглядати список користувачів, змінювати ролі, активувати або вимикати користувачів. Звичайний користувач може переглядати і редагувати власний профіль.

Модуль доставок є основним робочим модулем системи. Через нього MANAGER або ADMIN створює доставки, змінює статуси, призначає маршрут і транспорт, редагує ETA та додає нотатки. Для VIEWER цей модуль доступний лише в режимі перегляду.

Модуль довідників включає склади, транспортні засоби та маршрути. Він дозволяє структурувати дані й уникати хаотичного введення повторюваної інформації. ADMIN може повноцінно керувати довідниками, а MANAGER і VIEWER переважно використовують їх для перегляду або вибору під час роботи з доставками.

Модуль аналітики відображає ключові показники логістичних процесів: кількість доставок, частку затримок, завантаженість маршрутів, завантаженість

складів і проблемні зони. Його основна задача — швидко показати загальний стан логістики.

Модуль Route Risk Radar виконує оцінювання ризикових доставок і сортує їх за рівнем небезпеки. Для логіста це один із найважливіших інструментів, оскільки він дозволяє бачити не лише всі доставки, а саме ті, які потребують першочергової уваги.

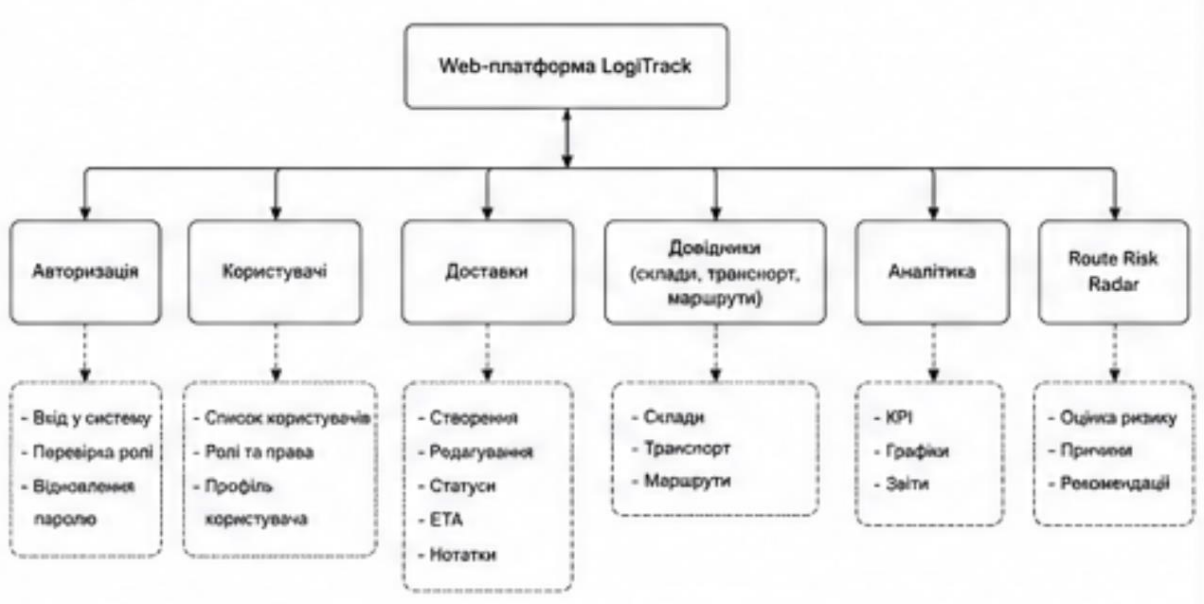


Рис. 4.7 Структура основних модулів платформи LogiTrack

Загалом модульна структура робить систему більш зрозумілою, підтримуваною і готовою до розвитку. У майбутньому до неї можна додати GPS-моніторинг, ETA Simulator, What-if сценарії або інтеграцію із зовнішніми системами.

4.9 Проєктування матриці прав доступу користувачів

Оскільки в системі передбачено декілька ролей користувачів, необхідно чітко визначити, які дії доступні кожній ролі. Це дозволяє уникнути ситуацій, коли користувач випадково або навмисно змінює дані, до яких не повинен мати доступу.

У LogiTrack використовується три основні ролі: ADMIN, MANAGER та VIEWER. ADMIN має повний доступ до системи, MANAGER працює з доставками та операційними модулями, а VIEWER має лише права перегляду.

Таблиця 4.1

Матриця прав доступу користувачів

Функція системи	**ADMIN**	**MANAGER**	**VIEWER**
Перегляд dashboard	+	+	+
Перегляд доставок	+	+	+
Створення доставки	+	+	-
Редагування доставки	+	+	-
Видалення доставки	+	-	-
Зміна статусу доставки	+	+	-
Оновлення ETA	+	+	-
Додавання нотаток	+	+	-
Перегляд Route Risk Radar	+	+	+
Перегляд аналітики	+	+	+
Керування складами	+	Перегляд	Перегляд
Керування транспортом	+	Перегляд	Перегляд
Керування маршрутами	+	Перегляд	Перегляд
Керування користувачами	+	-	-
Зміна ролей користувачів	+	-	-
Активація / вимкнення користувачів	+	-	-
Перегляд профілю	+	+	+
Редагування власного профілю	+	+	+

Матриця прав доступу є основою для реалізації безпеки системи. Вона повинна враховуватися як на серверному рівні, так і на клієнтському. На сервері доступ до endpoint-ів обмежується відповідно до ролі користувача. На клієнті інтерфейс приховує кнопки та сторінки, які не доступні конкретній ролі.

5 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ WEB-ПЛАТФОРМИ LOGITRACK

5.1 Загальна характеристика реалізованої web-платформи

У межах кваліфікаційної роботи було реалізовано web-платформу LogiTrack, призначену для централізованого моніторингу логістичних процесів підприємства. Система об’єднує в одному інтерфейсі роботу з доставками, маршрутами, складами, транспортними засобами, користувачами, аналітичними показниками та операційними ризиками.

Основна ідея реалізованої платформи полягає не лише у збереженні логістичних даних, а й у наданні користувачу інструментів для швидкого виявлення проблемних доставок. Для цього в системі реалізовано модуль Route Risk Radar, який автоматично оцінює ризик кожної доставки та показує рівень небезпеки у вигляді значення riskScore від 0 до 100.

Платформа має клієнт-серверну архітектуру. Клієнтська частина реалізована як web-інтерфейс, через який користувач взаємодіє із системою. Серверна частина відповідає за обробку запитів, перевірку прав доступу, виконання бізнес-логіки та роботу з базою даних. Для збереження інформації використовується PostgreSQL.

Реалізований web-застосунок має такі основні сторінки:

– /login — сторінка авторизації;

– /register — сторінка реєстрації користувача;

– /dashboard — головна робоча сторінка доставок;

– /deliveries/:id — сторінка деталей конкретної доставки;

– /analytics — сторінка аналітики логістичних процесів;

– /risk-radar — сторінка оцінювання ризиків доставки;

– /catalogs — сторінка довідників складів, транспорту та маршрутів;

– /profile — сторінка профілю користувача;

– /users — адміністративна сторінка керування користувачами.

Реалізація сторінок побудована таким чином, щоб користувач бачив тільки ті функції, які відповідають його ролі. Це дозволяє уникнути зайвого перевантаження інтерфейсу та зменшити ризик несанкціонованого редагування даних.

5.2 Реалізація серверної частини

Серверна частина web-платформи LogiTrack реалізована з використанням Java та Spring Boot. Вона відповідає за основну бізнес-логіку системи, обробку HTTP-запитів, роботу з базою даних, перевірку прав доступу та формування відповідей для клієнтської частини.

У серверній частині використано багаторівневу структуру, яка включає:

– контролери;

– сервіси;

– репозиторії;

– сутності бази даних;

– DTO-класи;

– конфігурацію безпеки.

Контролери приймають запити від клієнтської частини. Наприклад, запит на отримання списку доставок, створення нової доставки, оновлення профілю або отримання результатів Route Risk Radar надходить до відповідного REST endpoint-а. Після цього контролер передає дані до сервісного шару.

Сервіси містять основну бізнес-логіку. Наприклад, сервіс доставок відповідає за створення, редагування, пошук і фільтрацію доставок. Сервіс користувачів обробляє профіль, зміну ролей і активність облікових записів. Окремий risk-сервіс відповідає за розрахунок ризику доставки.

Репозиторії використовуються для взаємодії з базою даних PostgreSQL. Вони забезпечують створення, читання, оновлення та видалення записів. Такий підхід дозволяє відокремити роботу з даними від бізнес-логіки системи.

У серверній частині реалізовано такі основні групи endpoint-ів:

– /api/auth — авторизація та реєстрація;

– /api/users — керування користувачами;

– /api/users/me — робота з профілем поточного користувача;

– /api/deliveries — робота з доставками;

– /api/warehouses — робота зі складами;

– /api/vehicles — робота з транспортом;

– /api/routes — робота з маршрутами;

– /api/analytics — отримання аналітичних показників;

– /api/risks — отримання результатів Route Risk Radar.

Spring Security використовується для реалізації автентифікації та авторизації. Офіційна документація визначає Spring Security як фреймворк для автентифікації, авторизації та захисту Spring-застосунків, а method security дозволяє обмежувати доступ до методів за допомогою анотацій.

5.3 Реалізація клієнтської частини

Клієнтська частина web-платформи LogiTrack реалізована з використанням React та TypeScript. React дає можливість будувати інтерфейс із незалежних компонентів, які можна комбінувати в повноцінні сторінки застосунку.

У LogiTrack компонентний підхід використовується для реалізації таблиць, форм, карток статистики, фільтрів, навігаційного меню, блоків ризику, сторінки профілю та адміністративної панелі. Це спрощує підтримку інтерфейсу, оскільки окремі елементи можна змінювати незалежно один від одного.

TypeScript використовується для типізації даних, які надходять із серверної частини. Наприклад, для доставки, користувача, маршруту, складу, транспорту та результату оцінювання ризику можна визначити окремі інтерфейси. Це зменшує кількість помилок під час роботи з полями об’єктів і робить код більш передбачуваним.

Основними сторінками клієнтської частини є:

– сторінка авторизації;

– сторінка реєстрації;

– dashboard;

– сторінка деталей доставки;

– сторінка аналітики;

– сторінка Route Risk Radar;

– сторінка довідників;

– сторінка профілю;

– сторінка користувачів.

Важливою частиною клієнтської реалізації є перевірка ролі користувача. Після входу в систему інформація про користувача та його роль зберігається в контексті авторизації. Далі ця роль використовується для відображення або приховування певних елементів інтерфейсу. Наприклад, користувач із роллю VIEWER не бачить кнопки створення, редагування або видалення доставок.

5.4 Реалізація авторизації та реєстрації користувачів

Сторінка авторизації доступна за маршрутом /login. Вона дозволяє користувачу увійти в систему за допомогою email і пароля. Також на сторінці реалізовано швидкі кнопки входу в демо-акаунти: Admin**,** Manager та Viewer.

Після входу в систему користувач отримує доступ до функціональності відповідно до своєї ролі. Наприклад, ADMIN бачить сторінку користувачів і може керувати системними даними, MANAGER може створювати та редагувати доставки, а VIEWER має лише доступ до перегляду.

Сторінка реєстрації доступна за маршрутом /register. Вона дозволяє створити нового користувача. Новий користувач отримує базову роль логіста або оператора, тобто може працювати з доставками в межах обмежених прав. Це дозволяє додавати нових працівників без ручного втручання адміністратора на початковому етапі.

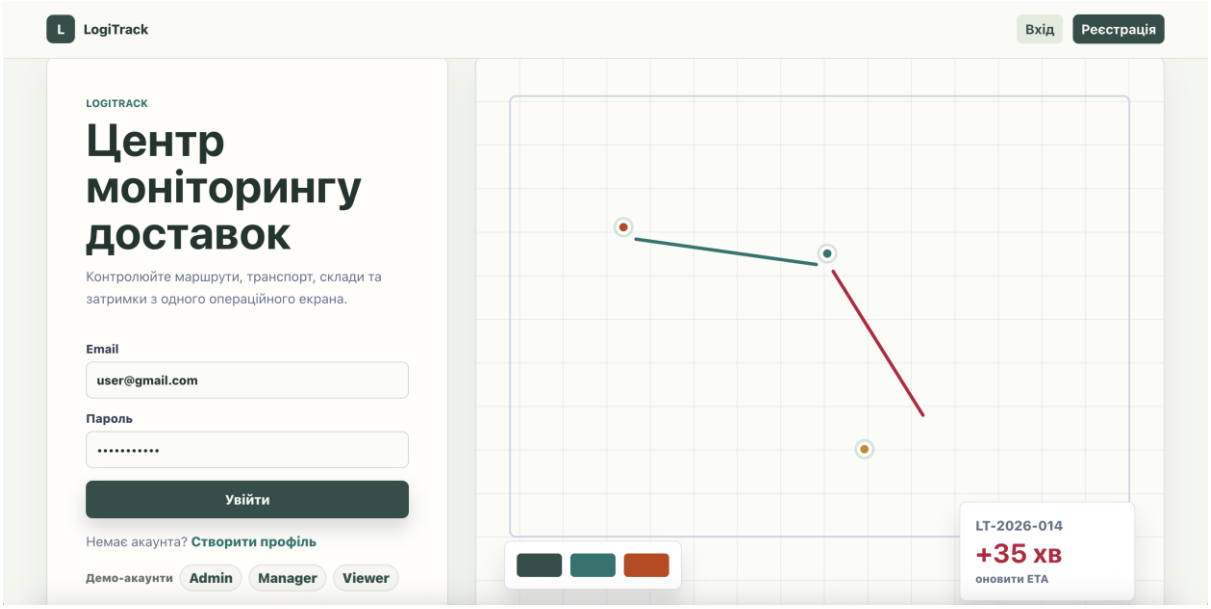


Рис. 5.1 Екранна форма авторизації користувача

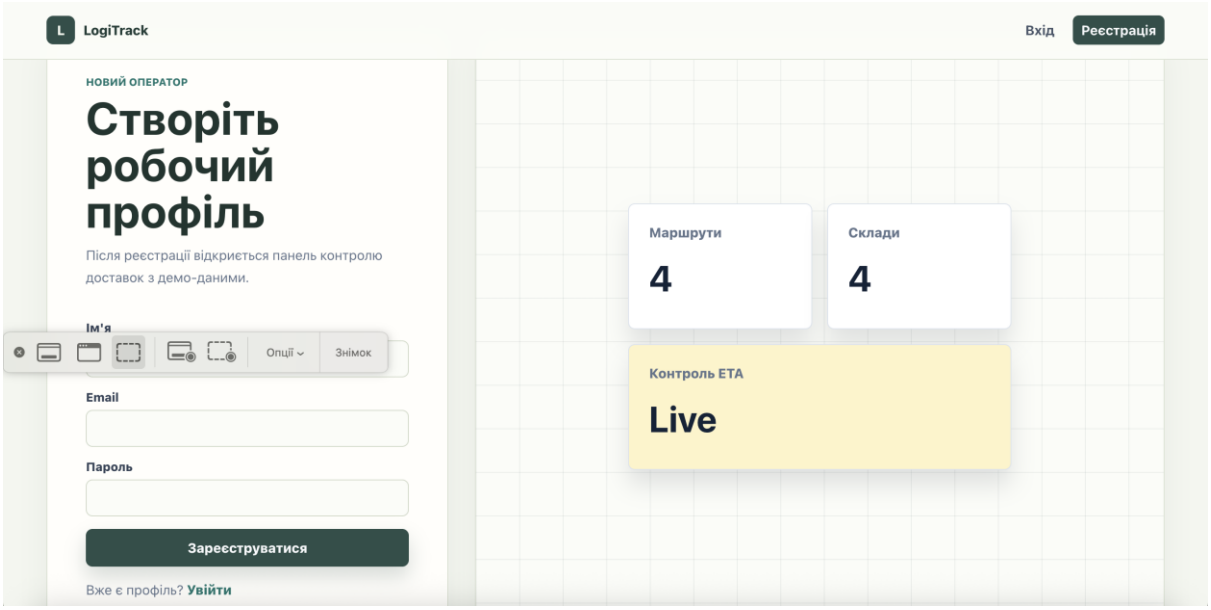


Рис. 5.2 Екранна форма реєстрації користувача

5.5 Реалізація сторінки доставок

Головна робоча сторінка системи доступна за маршрутом /dashboard. Вона призначена для щоденної роботи логіста або диспетчера з доставками. На цій сторінці відображається список усіх доставок, а також додаткова інформація, необхідна для швидкого контролю поточного стану логістики.

На сторінці доставок реалізовано:

– список усіх доставок;

– пошук за номером доставки, клієнтом або маршрутом;

– фільтрацію за статусом;

– відображення найближчого ETA;

– блок маршрутів та їх завантаження;

– кількість активних доставок;

– інформацію про затримки.

Функціональність сторінки залежить від ролі користувача. ADMIN може створювати, редагувати і видаляти доставки. MANAGER може створювати та редагувати доставки, але не має права видаляти записи. VIEWER може лише переглядати список доставок і їх деталі.

Такий підхід відповідає реальній логіці роботи логістичного відділу. Диспетчер або менеджер повинен мати можливість оперативно оновлювати інформацію, але видалення важливих логістичних даних має бути доступне лише адміністратору.

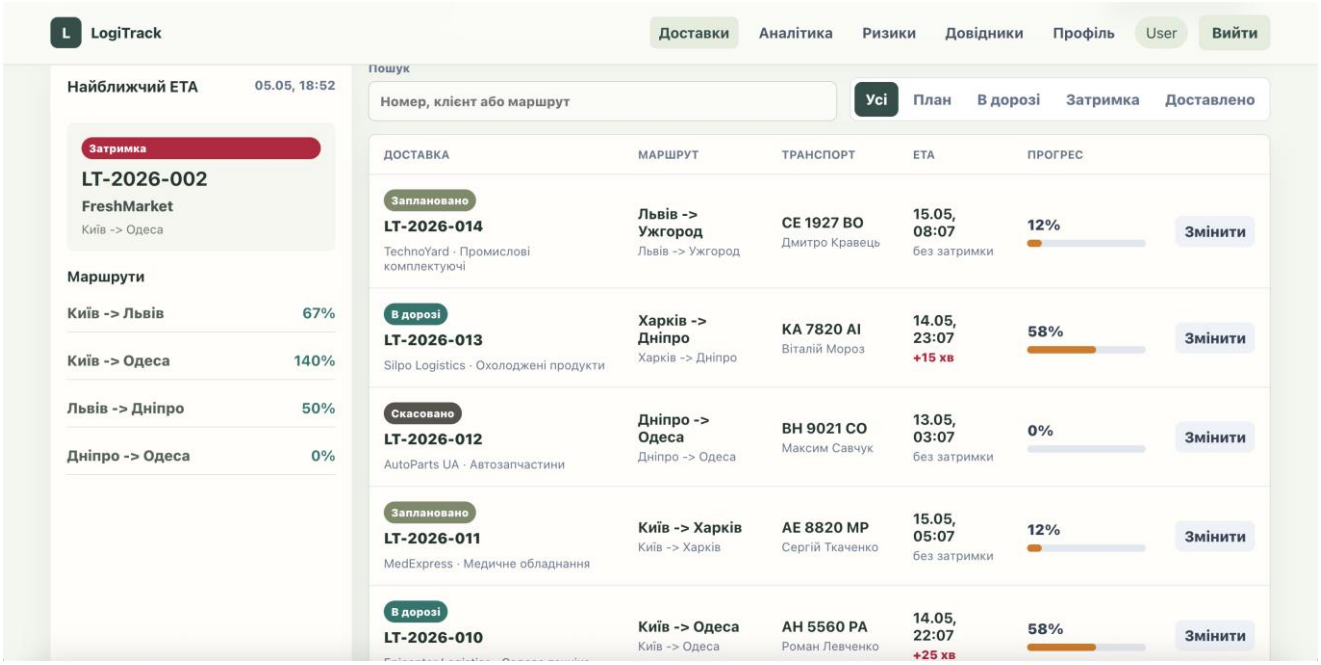


Рис. 5.3 Екранна форма головної сторінки доставок

5.6 Реалізація сторінки деталей доставки

Сторінка деталей доставки доступна за маршрутом /deliveries/:id. Вона призначена для перегляду повної інформації про конкретну доставку. Якщо на dashboard користувач бачить загальний список рейсів, то на сторінці деталей він може розглянути конкретне замовлення більш детально.

На сторінці відображаються:

– номер доставки;

– клієнт;

– вантаж;

– статус;

– маршрут;

– склад відправлення;

– склад призначення;

– транспортний засіб;

– водій;

– планове і фактичне виконання;

– затримка;

– прогрес доставки;

– операційні нотатки.

Ця сторінка важлива для MANAGER, оскільки дозволяє оцінити стан конкретної доставки та за потреби оновити дані. Для VIEWER сторінка також корисна, але тільки в режимі перегляду. Таким чином, система підтримує різні сценарії використання однієї і тієї ж сторінки залежно від ролі користувача.

Операційні нотатки дозволяють фіксувати важливу інформацію щодо доставки. Наприклад, можна вказати причину затримки, уточнення від водія, зміну ETA або коментар щодо клієнта. Це допомагає зберігати історію роботи з доставкою в одному місці.

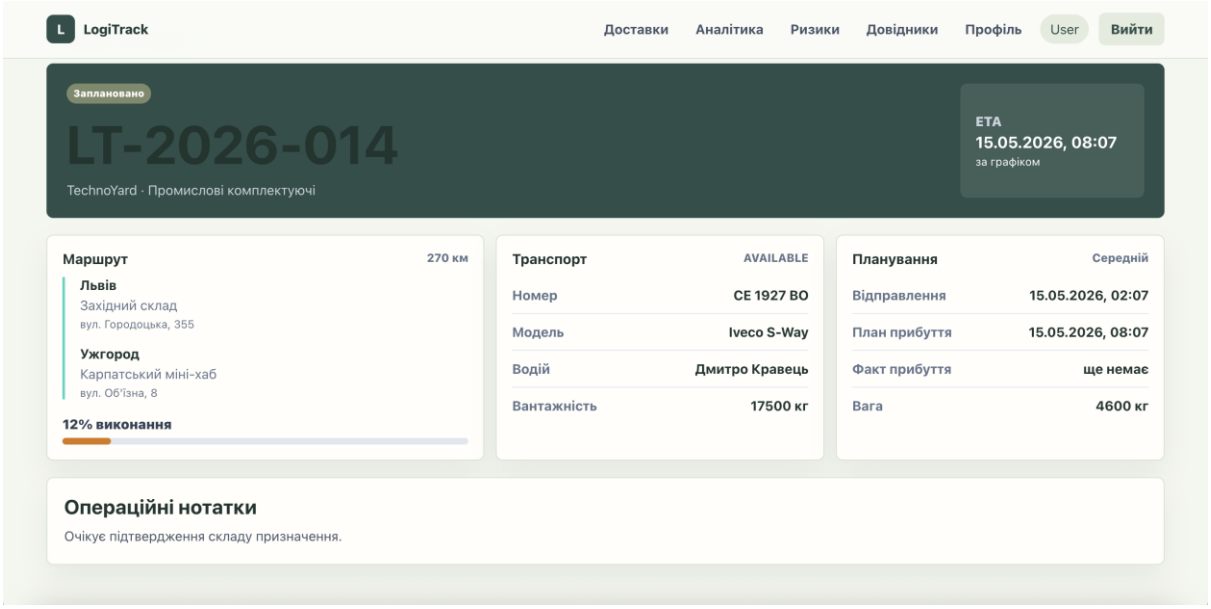


Рис. 5.4 Екранна форма деталей доставки

5.7 Реалізація сторінки аналітики

Сторінка аналітики доступна за маршрутом /analytics. Вона призначена для узагальненого перегляду стану логістичних процесів. Її основними користувачами є керівник, адміністратор або менеджер, який хоче швидко оцінити ефективність доставки.

На сторінці аналітики відображаються такі показники:

– індекс стабільності логістичної мережі;

– загальна кількість доставок;

– кількість активних доставок;

– середня затримка;

– відсоток вчасності;

– завантаження маршрутів;

– розподіл статусів доставки;

– доступний транспорт;

– загальна вага вантажів.

Сторінка аналітики не призначена для редагування даних. Її основна задача — відображення звітної та управлінської інформації. Завдяки цьому керівник може

швидко побачити, наскільки стабільно працює логістична система, чи є проблемні маршрути, чи достатньо доступного транспорту та який рівень затримок спостерігається.

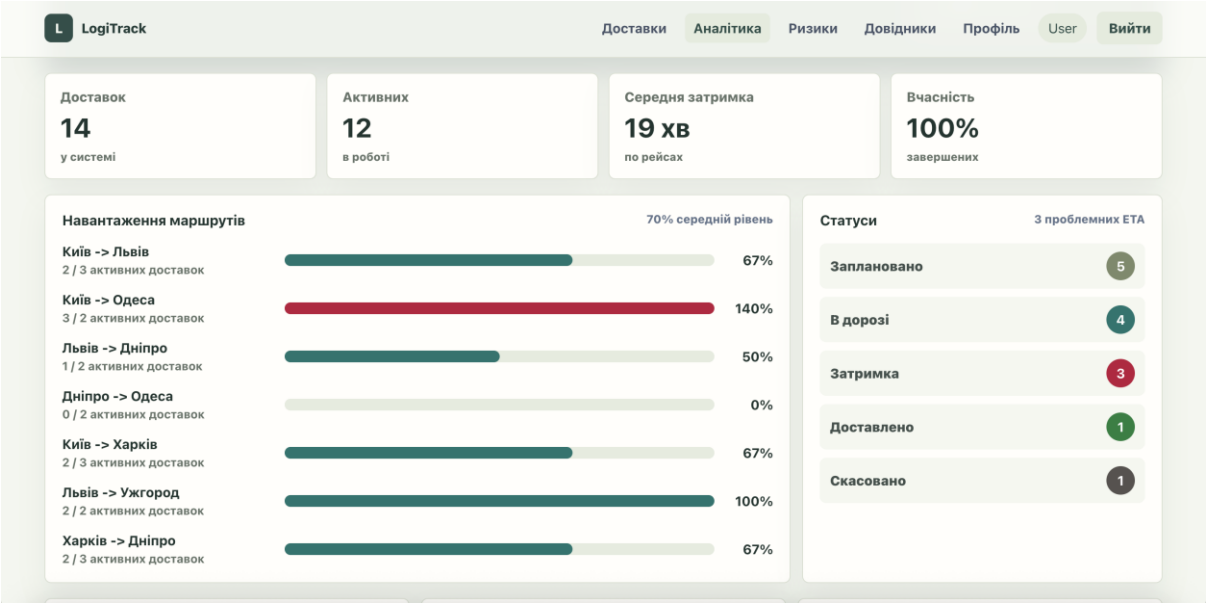


Рис. 5.5 Екранна форма сторінки аналітики

5.8 Реалізація модуля Route Risk Radar

Сторінка Route Risk Radar доступна за маршрутом /risk-radar. Це унікальна функція web-платформи LogiTrack, яка відрізняє її від звичайного CRUD-застосунку для роботи з доставками.

Модуль автоматично оцінює ризик кожної доставки. Для цього враховуються такі фактори:

– поточний статус доставки;

– затримка в хвилинах;

– пріоритет доставки;

– завантаження маршруту;

– стан транспорту;

– завантаженість складів.

Для кожної доставки система формує:

– riskScore від 0 до 100;

– рівень ризику: LOW, MEDIUM, HIGH або CRITICAL;

– список причин ризику;

– рекомендацію системи.

Наприклад, якщо доставка має затримку, високий пріоритет, перевантажений маршрут і склад призначення із завантаженням понад 80%, система підвищує riskScore і може віднести доставку до рівня HIGH або CRITICAL. При цьому користувач бачить не лише рівень ризику, а й пояснення, чому система зробила такий висновок.

Це робить модуль зрозумілим для логіста. Користувач може швидко визначити, які рейси потребують першочергової уваги, і прийняти рішення: зв’язатися з клієнтом, поставити доставку під ручний контроль, перевірити доступність транспорту або змінити маршрут.

На захисті цей модуль можна подати як засіб раннього виявлення проблемних доставок. Його цінність полягає в тому, що система не просто відображає вже наявну затримку, а допомагає логісту бачити потенційно проблемні рейси за сукупністю факторів.

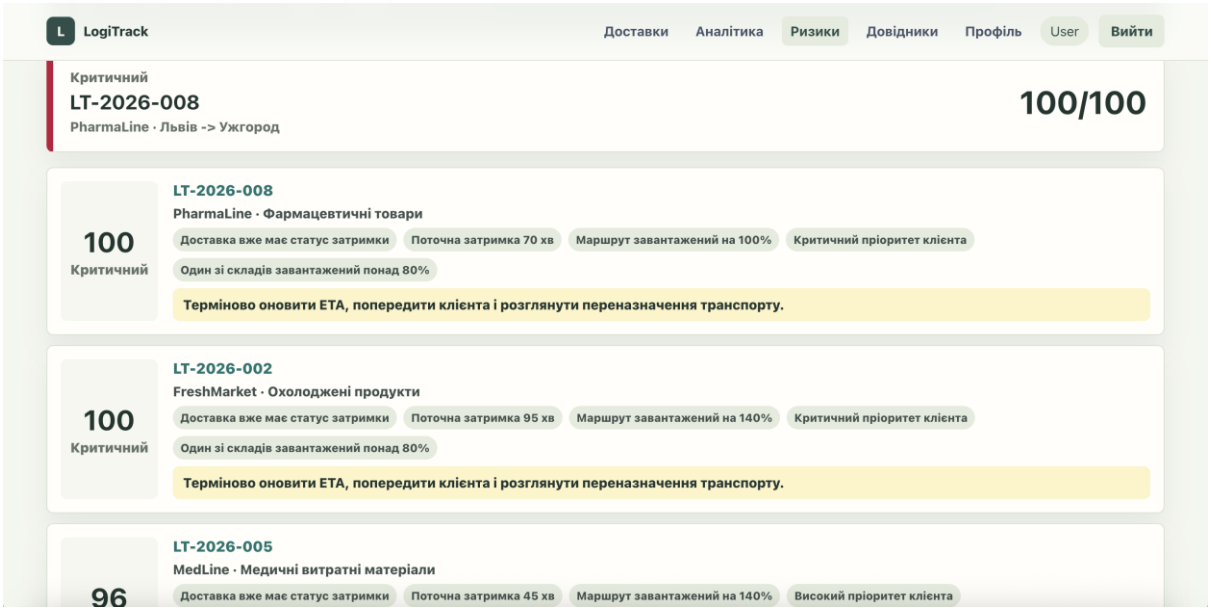


Рис. 5.6 Екранна форма модуля Route Risk Radar

5.9 Реалізація сторінки довідників

Сторінка довідників доступна за маршрутом /catalogs. Вона містить операційні дані, які використовуються при створенні та редагуванні доставок. Усередині сторінки реалізовано вкладки:

– склади;

– транспорт;

– маршрути.

Довідник складів містить інформацію про логістичні точки, їх адресу, поточне завантаження та статус. Довідник транспорту містить дані про автомобілі, їх тип, вантажність, статус і водія. Довідник маршрутів містить напрямки доставки, завантаження маршруту та інші параметри.

Права доступу до довідників залежать від ролі користувача. ADMIN може додавати, редагувати та видаляти записи. MANAGER і VIEWER мають доступ лише до перегляду довідників. Такий підхід дозволяє захистити системні дані від випадкових змін, але водночас залишає їх доступними для роботи з доставками.

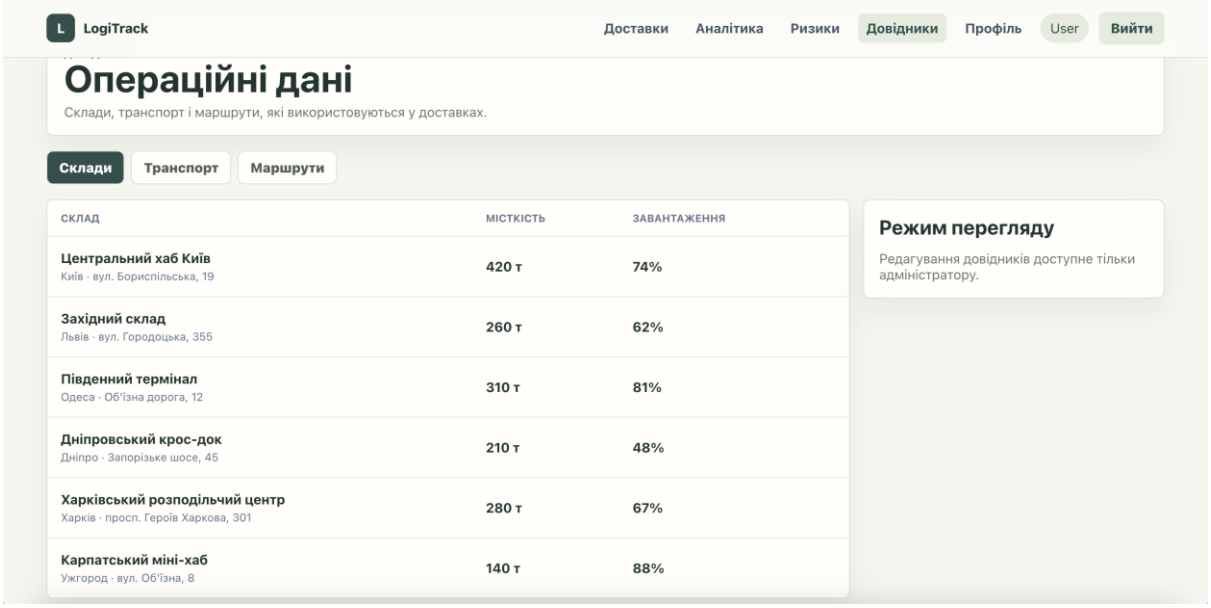


Рис. 5.7 Екранна форма сторінки довідників

5.10 Реалізація сторінки профілю

Сторінка профілю доступна за маршрутом /profile. Вона містить персональні та робочі дані користувача. У поточній реалізації саме на цю сторінку винесено інформацію про користувача, яка раніше надто виділялася на головній сторінці.

На сторінці профілю користувач може редагувати:

– ім’я;

– телефон;

– посаду;

– відділ;

– колір профілю.

Сторінка профілю важлива для персоналізації системи. Вона дозволяє зберігати робочу інформацію про користувача та робить інтерфейс більш завершеним. Крім того, профіль може використовуватися для майбутнього розширення системи, наприклад для налаштувань сповіщень або персональних параметрів dashboard.

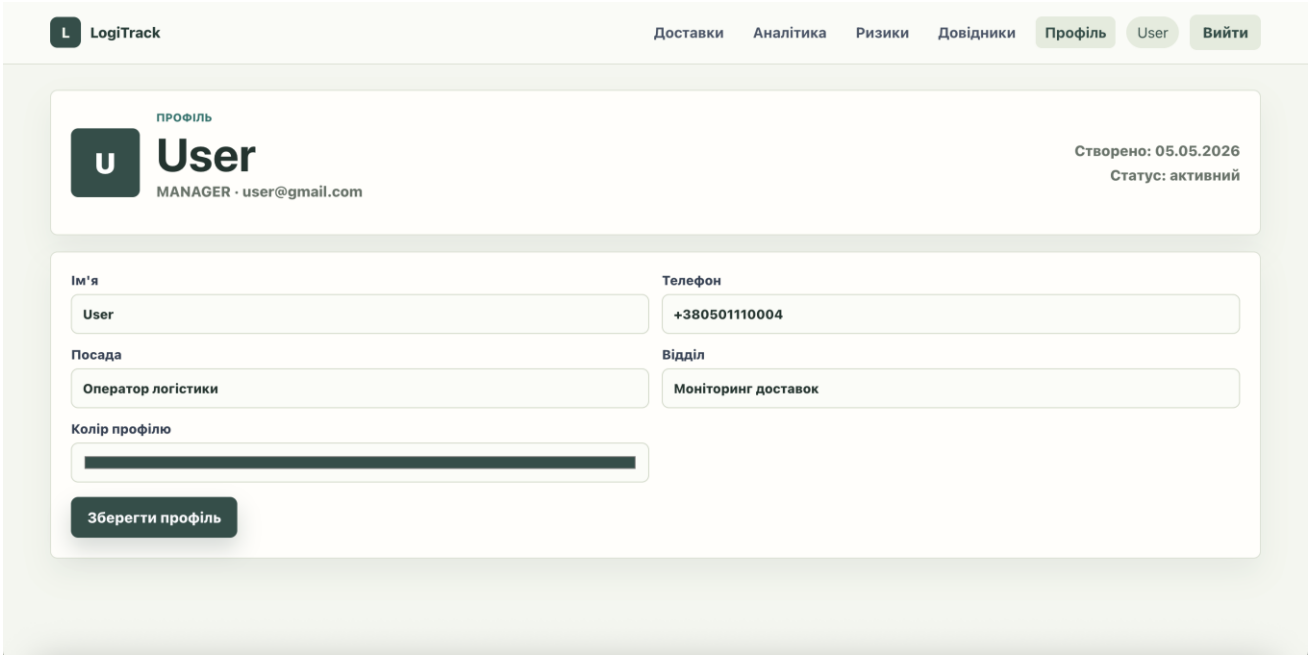


Рис. 5.8 Екранна форма профілю користувача

5.11 Реалізація сторінки користувачів

Сторінка користувачів доступна за маршрутом /users і призначена тільки для ролі ADMIN. Вона виконує функцію адміністративної панелі для керування обліковими записами користувачів.

На цій сторінці адміністратор може:

– переглядати всіх користувачів;

– змінювати ролі;

– активувати або вимикати акаунти;

– переглядати контактні та робочі дані користувачів.

Ця сторінка є важливою для підтримки рольової моделі системи. Наприклад, якщо працівник змінює посаду або переходить на іншу функцію, адміністратор може змінити його роль. Якщо користувач більше не повинен мати доступ до системи, акаунт можна вимкнути без повного видалення.

Доступ до сторінки /users повинен бути обмежений як на серверному, так і на клієнтському рівні. На клієнтському рівні пункт меню не відображається для MANAGER і VIEWER. На серверному рівні запити до endpoint-ів керування користувачами повинні бути дозволені лише ADMIN.

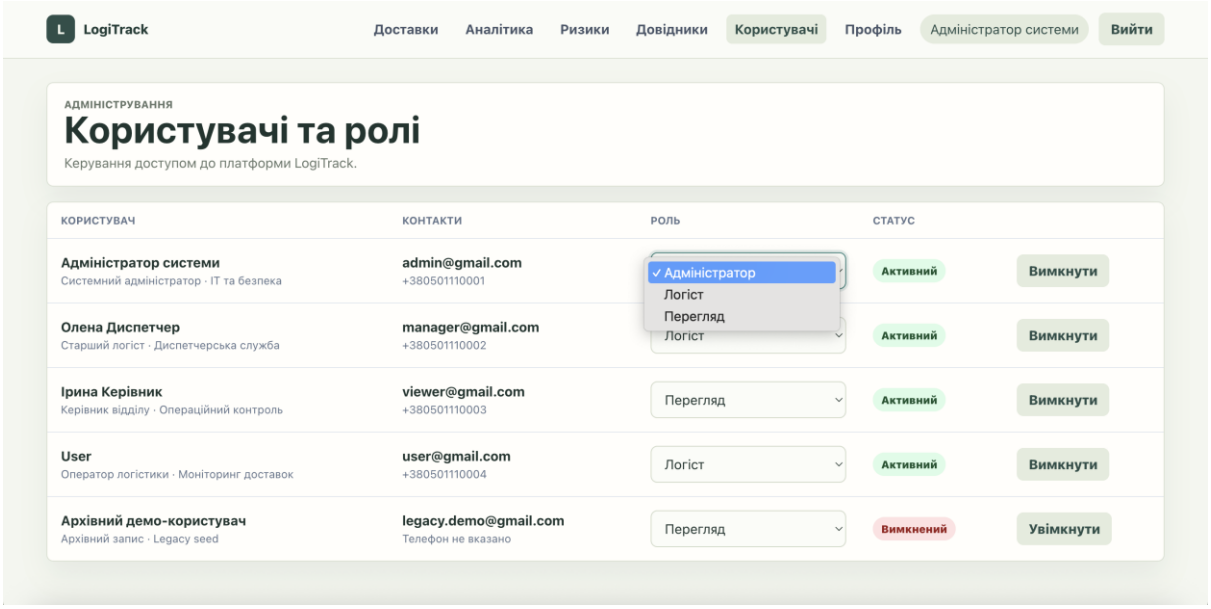


Рис. 5.9 Екранна форма сторінки користувачів

5.12 Тестування програмного забезпечення

Після реалізації основних функцій web-платформи LogiTrack необхідно виконати тестування. Мета тестування полягає в перевірці працездатності основних сторінок, коректності рольового доступу, правильності роботи з доставками, довідниками, профілем, аналітикою та модулем Route Risk Radar.

Тестування проводиться вручну через web-інтерфейс, а також шляхом перевірки збірки клієнтської і серверної частин. Для серверної частини може використовуватися команда mvn test, а для клієнтської — npm run build. Також потрібно перевірити вхід під різними ролями, щоб переконатися, що кожен тип користувача бачить лише дозволені функції.

Таблиця 5.1

Тест-кейси web-платформи LogiTrack

ID	Сценарій	Дії	Очікуваний результат	Статус
TC-01	Авторизація ADMIN	Увійти як admin@logitrack.local	Відкривається система з повним доступом	Пройдено
TC-02	Авторизація MANAGER	Увійти як manager@logitrack.local	Доступні доставки, аналітика, ризики, довідники	Пройдено
TC-03	Авторизація VIEWER	Увійти як viewer@logitrack.local	Доступний лише перегляд даних	Пройдено
TC-04	Реєстрація нового користувача	Заповнити форму /register	Створюється новий користувач із базовою роллю	Пройдено
TC-05	Створення доставки	У ролі ADMIN або MANAGER створити доставку	Доставка з’являється у списку	Пройдено
TC-06	Редагування доставки	Змінити статус або ETA доставки	Дані оновлюються після збереження	Пройдено

Продовження таблиці 5.1

Тест-кейси web-платформи LogiTrack

ID	Сценарій	Дії	Очікуваний результат	Статус
TC-07	Видалення доставки ADMIN	У ролі ADMIN видалити доставку	Запис видаляється із системи	Пройдено
TC-08	Заборона видалення для MANAGER	Увійти як MANAGER	Кнопка видалення недоступна	Пройдено
TC-09	Обмеження VIEWER	Увійти як VIEWER	Кнопки створення і редагування не відображаються	Пройдено
TC-10	Перегляд деталей доставки	Відкрити /deliveries/:id	Відображаються всі деталі доставки	Пройдено
TC-11	Перегляд аналітики	Відкрити /analytics	Відображаються логістичні показники	Пройдено
TC-12	Route Risk Radar	Відкрити /risk-radar	Відображаються riskScore, riskLevel, причини і рекомендації	Пройдено
TC-13	Високий ризик доставки	Перевірити доставку із затримкою та високим пріоритетом	Система визначає HIGH або CRITICAL	Пройдено
TC-14	Перегляд довідників	Відкрити /catalogs	Відображаються склади, транспорт і маршрути	Пройдено
TC-15	Керування довідниками ADMIN	У ролі ADMIN додати або змінити запис	Запис успішно зберігається	Пройдено
TC-16	Обмеження довідників для MANAGER	Увійти як MANAGER	Доступний перегляд без видалення системних даних	Пройдено
TC-17	Редагування профілю	Змінити телефон, посаду або відділ	Дані профілю оновлюються	Пройдено

5.14 Результати тестування

За результатами тестування встановлено, що основні функції web-платформи LogiTrack працюють відповідно до поставлених вимог. Авторизація та реєстрація виконуються коректно, демо-акаунти дозволяють швидко перевірити поведінку системи для різних ролей, а рольова модель правильно обмежує доступ до функцій.

Сторінка доставок дозволяє переглядати, шукати і фільтрувати доставки. Для ADMIN і MANAGER доступні робочі дії з доставками, тоді як VIEWER працює лише в режимі перегляду. Сторінка деталей доставки коректно відображає повну інформацію про рейс, включно зі статусом, маршрутом, складом, транспортом, водієм, ETA, затримкою і нотатками.

Сторінка аналітики відображає основні показники логістичних процесів: кількість доставок, активні доставки, середню затримку, відсоток вчасності, завантаження маршрутів, статуси доставок, доступний транспорт і загальну вагу вантажів.

Модуль Route Risk Radar коректно формує оцінку ризику доставки. Для доставок із затримкою, високим пріоритетом, перевантаженим маршрутом або проблемним транспортом система підвищує riskScore і показує відповідні причини. Це підтверджує, що модуль може використовуватися як інструмент раннього виявлення проблемних рейсів.

Також було перевірено сторінки довідників, профілю та користувачів. Довідники доступні для перегляду всім авторизованим користувачам, але редагування системних даних доступне лише адміністратору. Профіль дозволяє оновлювати персональні дані користувача. Сторінка користувачів доступна лише ADMIN, що відповідає вимогам безпеки.

Отже, реалізована web-платформа LogiTrack відповідає основним функціональним і нефункціональним вимогам. Система забезпечує централізований моніторинг логістики, рольовий доступ, роботу з доставками, довідниками, аналітикою та модулем Route Risk Radar.

ВИСНОВКИ

У даній кваліфікаційній роботі було досліджено процеси моніторингу логістики, контролю виконання доставок, керування маршрутами, транспортними засобами, складами та операційними ризиками. У ході роботи встановлено, що для логістичних відділів важливо мати не лише засіб обліку доставок, а й інструмент швидкого виявлення проблемних рейсів, які можуть призвести до затримок, перевантаження маршрутів або порушення планового графіка доставки.

Було проведено аналіз предметної галузі логістики та визначено основні проблеми, що виникають під час ручного або частково автоматизованого контролю доставок. До таких проблем належать розрізнене зберігання інформації, складність оперативного оновлення статусів, відсутність єдиного бачення завантаження маршрутів і складів, обмежені можливості аналітики та несвоєчасне виявлення ризикових доставок. На основі цього було обґрунтовано доцільність створення web-платформи LogiTrack як централізованого інструменту моніторингу логістичних процесів.

У роботі було розглянуто існуючі програмні засоби для управління доставками та логістичними процесами, зокрема Excel / Google Sheets, SAP Transportation Management, Oracle Transportation Management, Routific, OptimoRoute та Zoho Creator Logistics. У результаті аналізу встановлено, що частина рішень є надто простою і не забезпечує повноцінної автоматизації, а інші системи орієнтовані переважно на великі підприємства, мають високу складність впровадження або потребують значних фінансових витрат. Це підтвердило необхідність розробки власної системи, орієнтованої на конкретні задачі: контроль доставок, маршрутів, складів, транспорту, аналітики та ризиків.

Окрему увагу було приділено вибору засобів реалізації web-платформи. Для серверної частини було обрано Java та Spring Boot, оскільки Spring Boot дозволяє швидко створювати web-сервіси та серверні застосунки на Java. Для клієнтської частини було використано React і TypeScript, оскільки React підтримує побудову

інтерфейсу з окремих компонентів, що зручно для реалізації сторінок доставок, аналітики, довідників, профілю та Route Risk Radar. Для збереження даних було обрано PostgreSQL, який є об’єктно-реляційною системою керування базами даних і підходить для роботи зі структурованими взаємопов’язаними даними. UML-діаграми були використані для моделювання структури та поведінки системи, оскільки UML є стандартом для візуалізації, специфікації та документування програмних систем.

У процесі проєктування було визначено логічну структуру web-платформи LogiTrack, її архітектуру, основні модулі, структуру бази даних, ролі користувачів та права доступу. У системі передбачено три основні ролі: ADMIN, MANAGER та VIEWER. Роль ADMIN має повний доступ до системи, включно з керуванням користувачами, довідниками та доставками. Роль MANAGER орієнтована на щоденну роботу логіста з доставками, статусами, ETA, аналітикою та ризиками. Роль VIEWER призначена для перегляду інформації без можливості внесення змін.

Було спроєктовано та описано основні сторінки системи: сторінку авторизації, реєстрації, dashboard, деталі доставки, аналітику, Route Risk Radar, довідники, профіль користувача та адміністративну сторінку користувачів. Такий набір сторінок дозволяє охопити основні сценарії роботи логістичного відділу: від створення доставки до аналізу її ризику та перегляду загальної ефективності логістичної мережі.

Ключовим результатом роботи стала реалізація модуля Route Risk Radar, який відрізняє LogiTrack від звичайної системи обліку доставок. Модуль виконує автоматичне оцінювання ризику доставки на основі таких факторів, як поточний статус, затримка в хвилинах, пріоритет доставки, завантаження маршруту, стан транспортного засобу та завантаженість складів. На основі цих даних система формує riskScore від 0 до 100, визначає рівень ризику LOW, MEDIUM, HIGH або CRITICAL, показує причини ризику та надає рекомендовану дію для логіста.

Реалізація Route Risk Radar дозволяє перейти від простого перегляду таблиці доставок до більш практичного підходу — раннього виявлення проблемних рейсів. Такий модуль допомагає логісту швидше визначати, які доставки потребують уваги

першими, а також краще розуміти причини потенційної затримки. Це підвищує практичну цінність системи, оскільки вона не тільки зберігає дані, а й підтримує прийняття операційних рішень.

У ході реалізації було створено серверну та клієнтську частини web-платформи, реалізовано REST API, рольовий доступ, сторінки роботи з доставками, довідниками, користувачами, профілем, аналітикою та ризиками. Також було передбачено демо-акаунти для швидкої перевірки поведінки системи під різними ролями: ADMIN, MANAGER та VIEWER.

Результати дослідження були представлені та апробовані на наукових конференціях за тезами:

1. Мартиновченко М. В. Розробка платформи моніторингу логістики для оптимізації доставки. II Всеукраїнська науково-технічна конференція «Виклики та рішення в програмній інженерії», Київ, Державний університет інформаційно-комунікаційних технологій, 26 листопада 2025 р.
2. Мартиновченко М. В. Перспективи реалізації web-застосунку LogiTrack для моніторингу логістики та оцінювання ризиків доставки. Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація», 1–3 травня 2026 року.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ніколаєв І. В. Інформаційні системи в логістичному менеджменті. URL: https://dspace.kntu.kr.ua/bitstreams/924b460e-1dc1-4646-85cd-0d883a8fd8f5/download (дата звернення: 01.03.2026).
2. Михаліцька Н. Я., Верескля М. Р. Логістичний менеджмент : навчальний посібник. Львів : Львівський державний університет внутрішніх справ, 2020. 440 с. ISBN 978-617-511-319-6.
3. Штельмашук М. Цифровізація та автоматизація логістичних процесів: сучасні підходи та перспективи розвитку. Економіка та суспільство. 2024. № 68. URL: https://economyandsociety.in.ua/index.php/journal/article/view/5043 (дата звернення: 16.03.2026).
4. Крикавський Є. В. Логістика. Основи теорії : підручник. Львів : Національний університет «Львівська політехніка», 2004. 416 с.
5. Christopher M. Logistics and Supply Chain Management. 5th ed. Harlow : Pearson Education Limited, 2016. 328 p.
6. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley Professional, 2002. 560 p. ISBN 978-0321127426.
7. SAP Transportation Management. SAP. URL: https://www.sap.com/products/scm/transportation-logistics.html (дата звернення: 08.03.2026).
8. Oracle Transportation Management. Oracle. URL: https://www.oracle.com/scm/logistics/transportation-management/ (дата звернення: 08.03.2026).
9. Routific – Route Optimization Software. URL: https://www.routific.com/ (дата звернення: 08.03.2026).
10. OptimoRoute – Route Planning and Optimization Software. URL: https://optimoroute.com/ (дата звернення: 08.03.2026).

11. Zoho Creator Logistics Management Software. Zoho. URL: https://www.zoho.com/creator/solutions/logistics/logistics-management-lms.html (дата звернення: 08.03.2026).
12. Java Documentation. Oracle. URL: https://docs.oracle.com/en/java/ (дата звернення: 23.03.2026).
13. Spring Boot Documentation. Spring. URL: https://spring.io/projects/spring-boot (дата звернення: 23.03.2026).
14. Spring Security Documentation. Spring. URL: https://docs.spring.io/spring-security/reference/index.html (дата звернення: 23.03.2026).
15. Spring Data JPA Documentation. Spring. URL: https://spring.io/projects/spring-data-jpa (дата звернення: 23.03.2026).
16. PostgreSQL Documentation. PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/ (дата звернення: 23.03.2026).
17. React Documentation. React. URL: https://react.dev/ (дата звернення: 23.03.2026).
18. TypeScript Documentation. Microsoft. URL: https://www.typescriptlang.org/docs/ (дата звернення: 23.03.2026).
19. Docker Documentation. Docker. URL: https://docs.docker.com/ (дата звернення: 23.03.2026).
20. Unified Modeling Language Specification Version 2.5.1. Object Management Group. URL: https://www.omg.org/spec/UML/2.5.1/About-UML (дата звернення: 06.04.2026).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка системи моніторингу логістики для оптимізації доставки

Виконав студент 4 курсу
групи ПД-43
Мартиновченко Максим Віталійович
Керівник роботи
к.п.н., доц., проф. Корецька Вікторія Олександрівна

Київ – 2026

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати предметну галузь моніторингу логістики та оптимізації доставки.
2. Дослідити існуючі програмні засоби для управління доставками, маршрутами та логістичними процесами.
3. Визначити функціональні та нефункціональні вимоги до веб-застосунку LogiTrack.
4. Обґрунтувати вибір технологій реалізації: Java, Spring Boot, PostgreSQL, React та TypeScript.
5. Спроєктувати архітектуру системи, базу даних, ролі користувачів та UML-діаграми.
6. Реалізувати веб-застосунок LogiTrack з модулями доставок, аналітики, довідників, профілю та користувачів.
7. Провести тестування функціональності, рольового доступу та модуля оцінювання ризиків.

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: спрощення контролю логістичних процесів та своєчасного виявлення ризикових доставок за рахунок розробки web-застосунку LogiTrack для моніторингу доставок, маршрутів, складів, транспорту та операційних ризиків.

Об'єкт дослідження: процеси моніторингу логістики та контролю виконання доставок на підприємстві.

Предмет дослідження: засоби, методи та технології реалізації web-платформи для моніторингу логістичних процесів і оцінювання ризиків доставки.

АНАЛІЗ АНАЛОГІВ

Показник	Excel / Google Sheets	SAP Transportation Management	Oracle Transportation Management	Routific	OptimoRoute	Zoho Creator Logistics	LogiTrack
Облік доставок	+	+	+	+	+	+	+
Управління маршрутами	Частково	+	+	+	+	+	+
Облік транспорту	Частково	+	+	Частково	Частково	+	+
Облік складів	Частково	+	+	-	-	+	+
Контроль статусів доставки	Частково	+	+	+	+	+	+
Аналітика та звітність	Частково	+	+	Частково	Частково	+	+
Рольовий доступ	Обмежено	+	+	Частково	Частково	+	+
Гнучкість адаптації	Середня	Висока, але складна	Висока, але складна	Обмежена	Обмежена	Висока	Висока
Оцінювання ризику доставки	-	Частково	Частково	-	Частково	Частково	+
Власний модуль Route Risk Radar	-	-	-	-	-	-	+

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги
– Авторизація та реєстрація користувачів.
– Рольовий доступ.
– Керування доставками: створення, редагування, перегляд, видалення.
– Пошук і фільтрація доставок за статусом, маршрутом, клієнтом і пріоритетом.
– Перегляд деталей доставки: маршрут, склад, транспорт, ETA, затримка, нотатки.
– Робота з довідниками: склади, транспорт, маршрути.
– Аналітика логістичних показників.

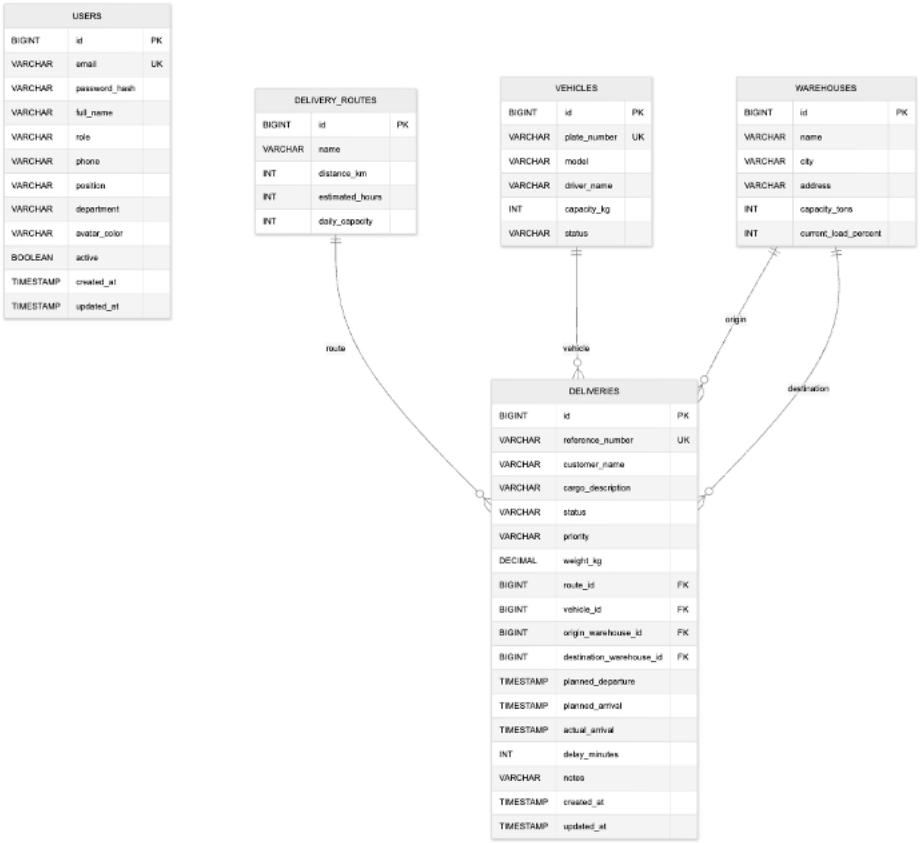
Нефункціональні вимоги
– Час виконання основних запитів: до 2–3 с.
– Підтримка одночасної роботи: не менше 20 користувачів.
– Обсяг БД: не менше 10 000 записів доставок.
– Підтримка браузерів: Chrome, Firefox, Microsoft Edge.
– Основні дії користувача: 3–5 кроків.
– Рівень ризику доставки: riskScore 0–100.
– Категорії ризику: LOW / MEDIUM / HIGH / CRITICAL.
– Захищене зберігання паролів і доступ відповідно до ролі.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

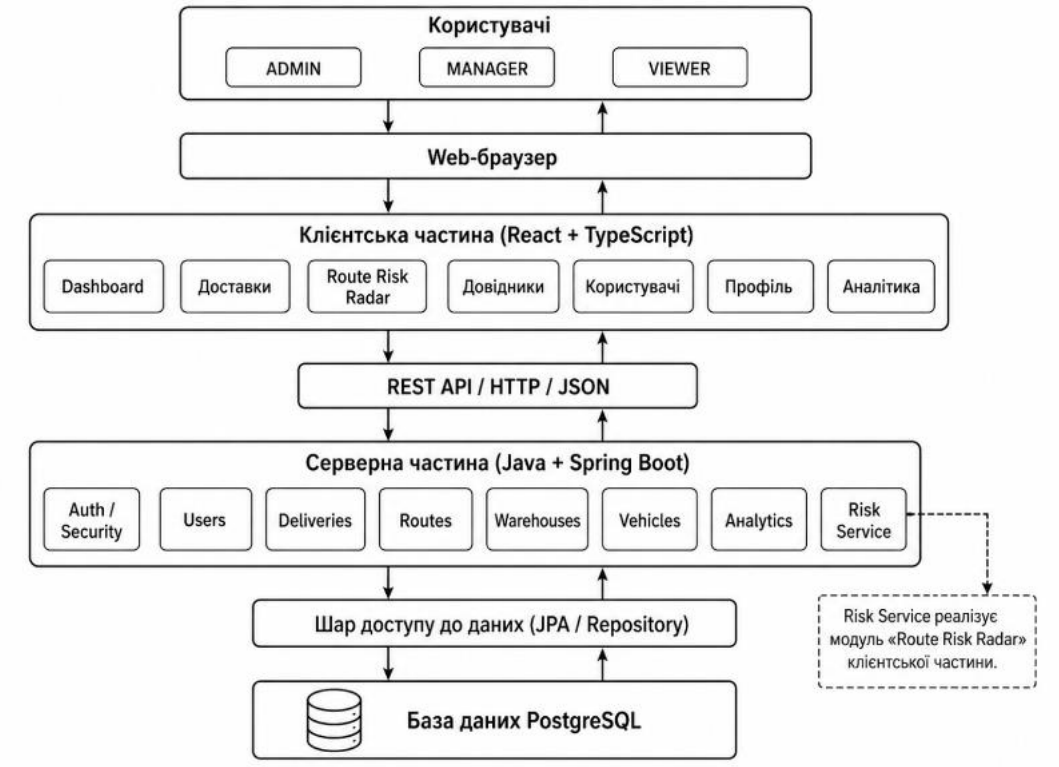
6

СХЕМА БАЗИ ДАНИХ



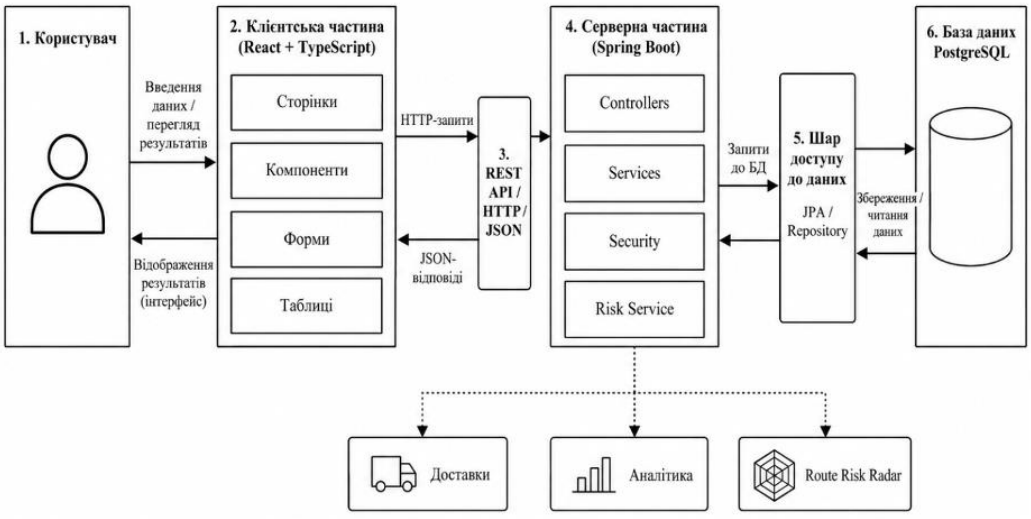
7

ЗАГАЛЬНА АРХІТЕКТУРА ЗАСТОСУНКУ



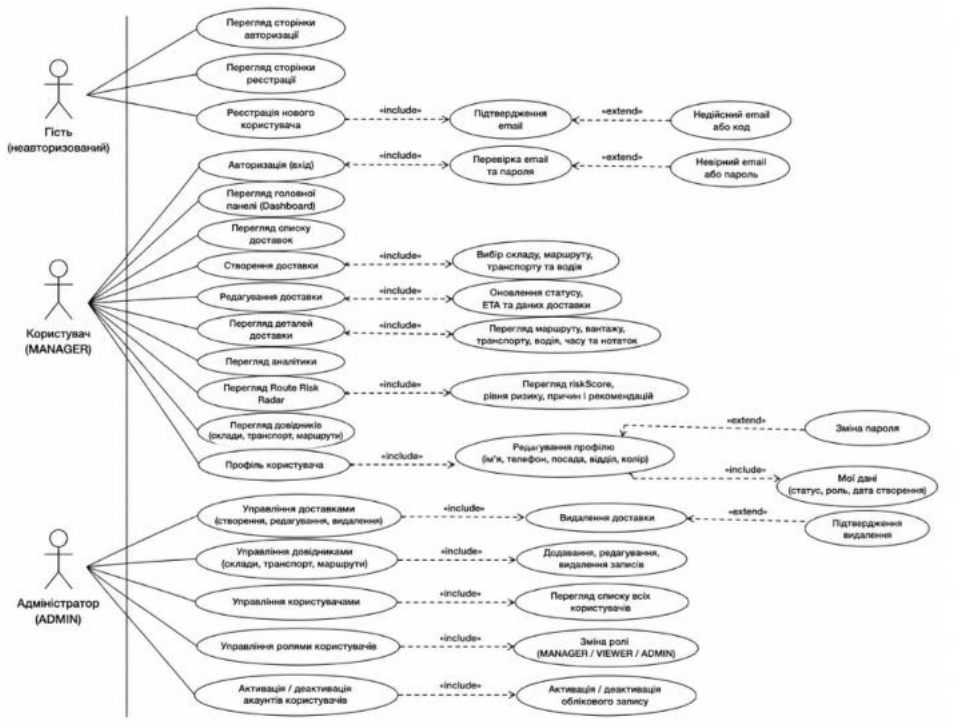
8

ВЗАЄМОДІЯ КЛІЄНТСЬКОЇ ЧАСТИНИ, СЕРВЕРНОЇ ЧАСТИНИ ТА БАЗИ ДАНИХ



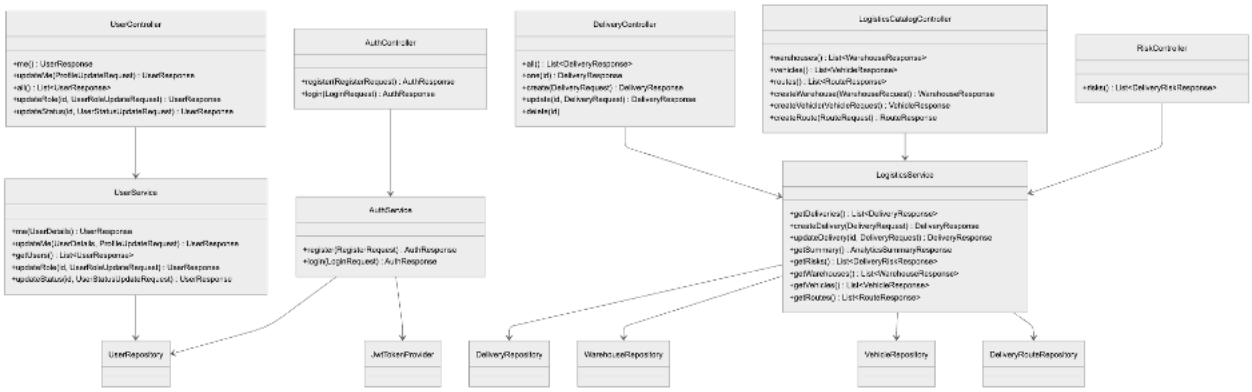
9

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



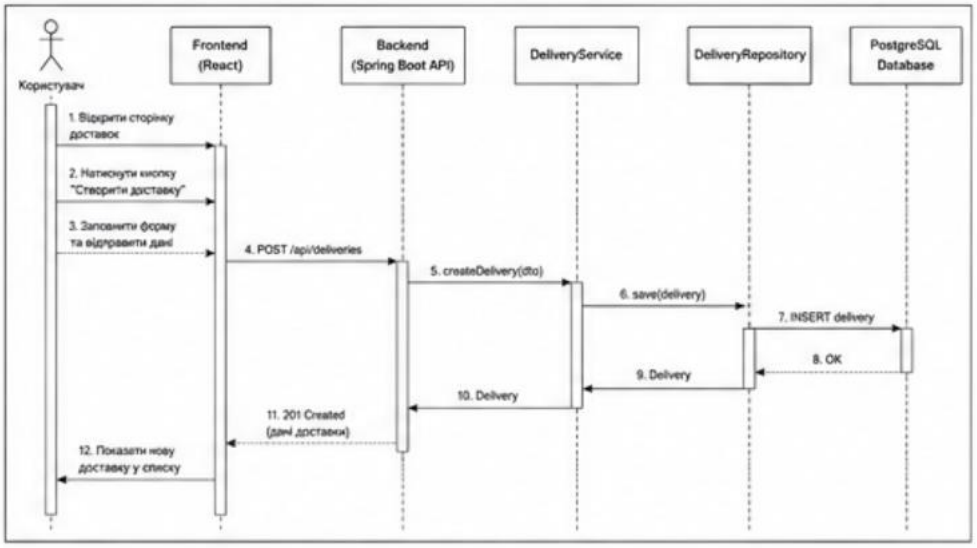
10

ДІАГРАМА КЛАСІВ ОСНОВНОЇ ЛОГІКИ



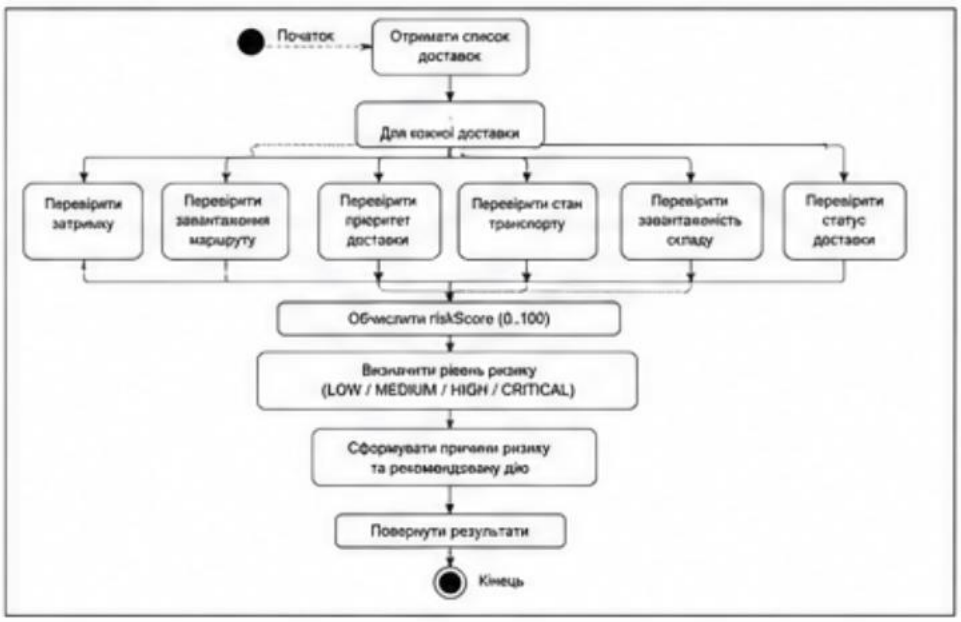
11

ДІАГРАМА ПОСЛІДОВНОСТІ СТВОРЕННЯ ДОСТАВКИ



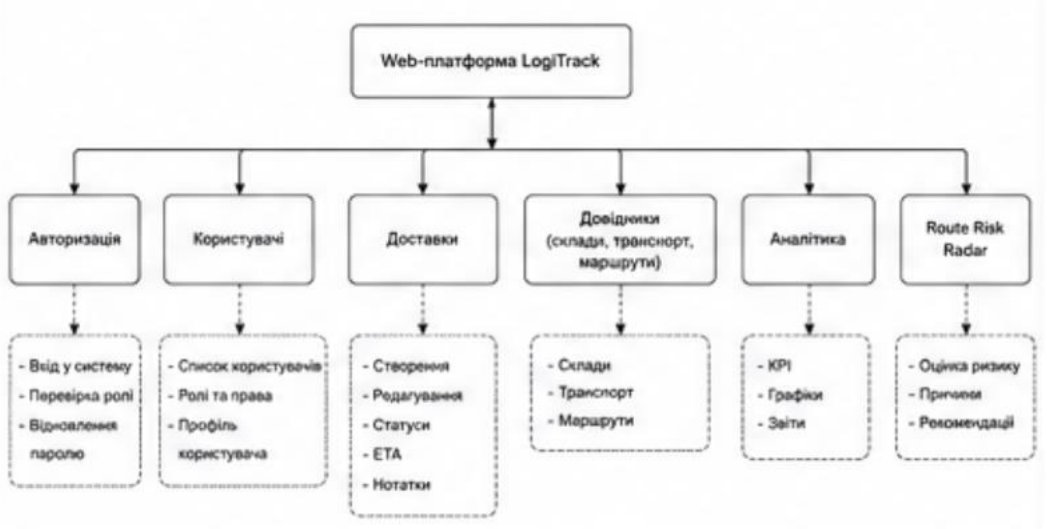
12

ДІАГРАМА ДІЯЛЬНОСТІ РОБОТИ МОДУЛЯ ROUTE RISK RADAR



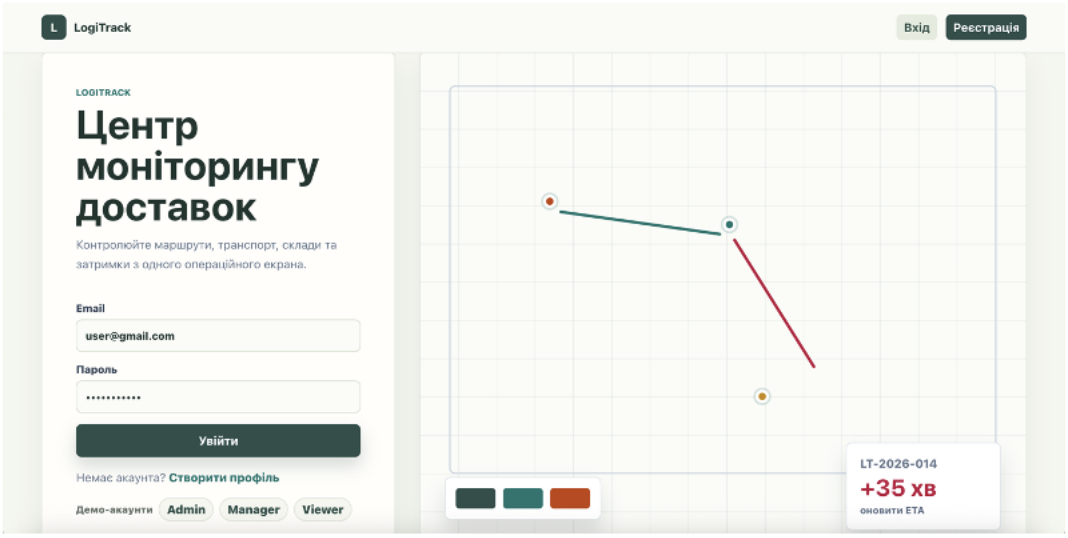
13

СТРУКТУРА ОСНОВНИХ МОДУЛІВ ПЛАТФОРМИ



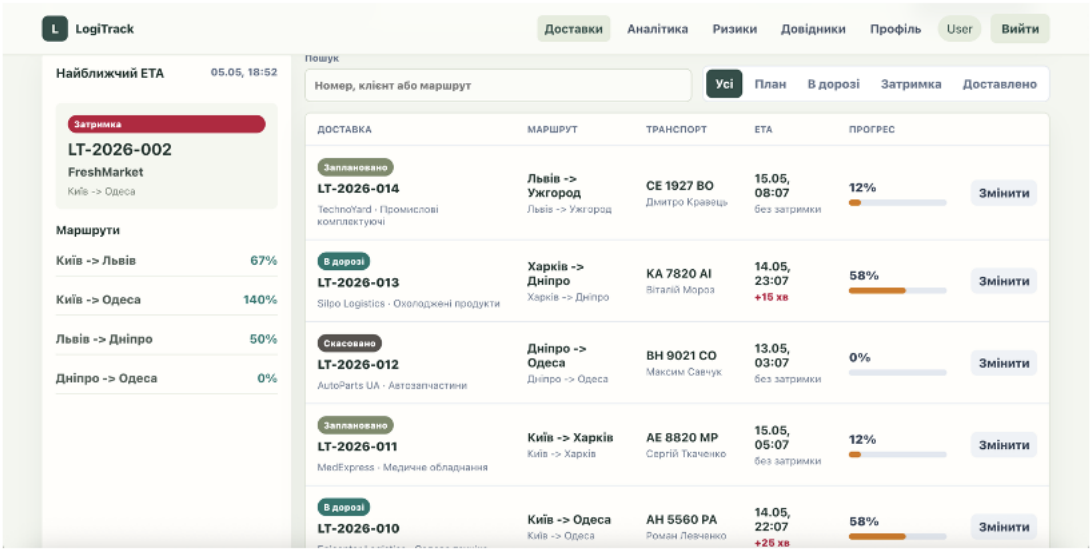
14

ЕКРАННІ ФОРМИ



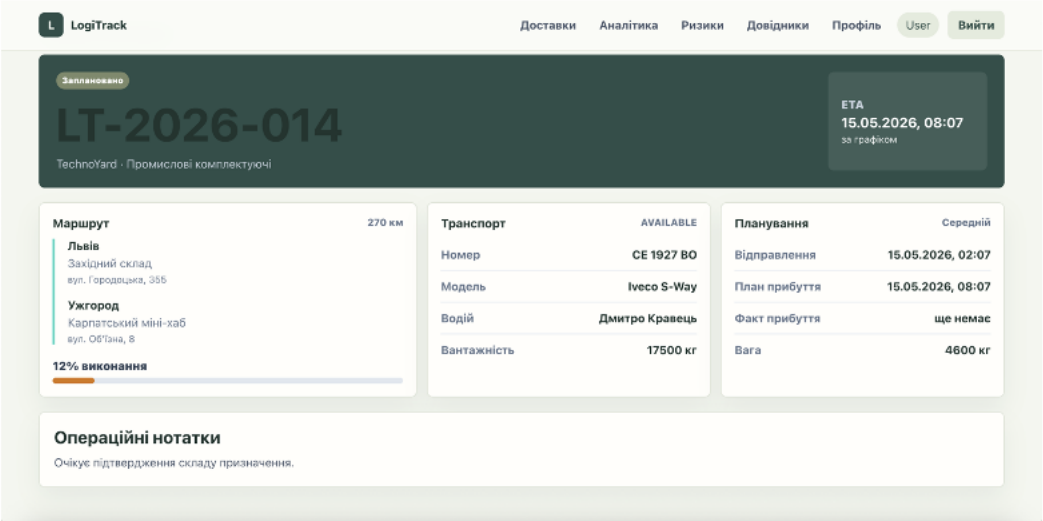
Сторінка входу

ЕКРАННІ ФОРМИ



Головна сторінки доставок

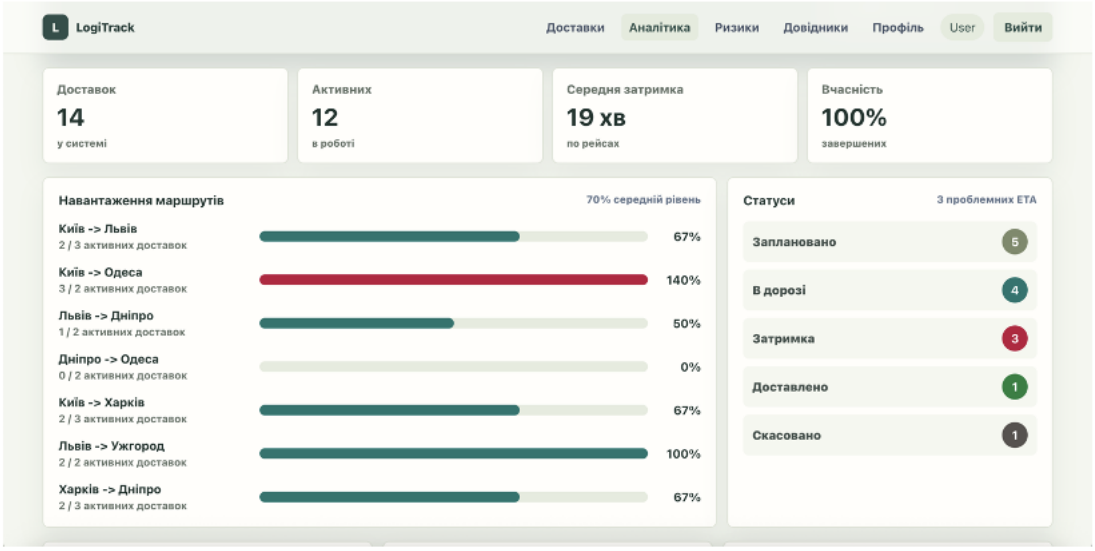
ЕКРАННІ ФОРМИ



Сторінка деталей доставки

17

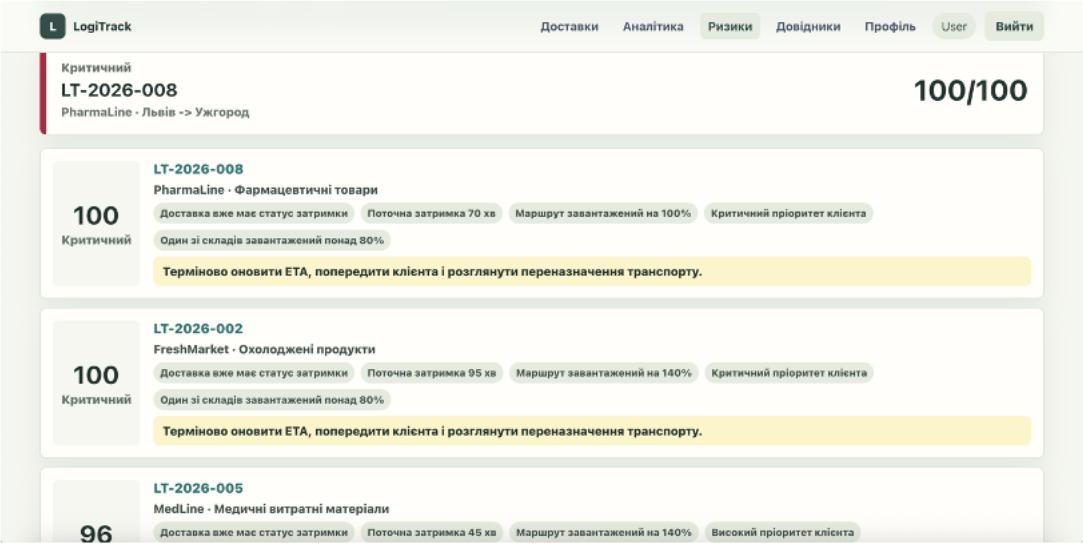
ЕКРАННІ ФОРМИ



Сторінка аналітики

18

ЕКРАННІ ФОРМИ



Сторінка модуля Route Risk Radar

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Мартиновченко М.В., Корецька В.О. Розробка платформи моніторингу логістики для оптимізації доставки. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.488-490.

2. Мартиновченко М.В., Корецька В.О. Перспективи реалізації web-платформи Logitrack для моніторингу логістики та оцінювання ризиків доставки. Матеріали Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація», 1–3 травня 2026 року, Київ, ДУІКТ. Збірник тез. *(подано до друку).*

ВИСНОВКИ

1. Проаналізовано предметну галузь моніторингу логістики та оптимізації доставки.
2. Досліджено існуючі програмні засоби для управління доставками, маршрутами та логістичними процесами.
3. Визначено функціональні та нефункціональні вимоги до веб-застосунку LogiTrack.
4. Обґрунтовано вибір технологій реалізації: Java, Spring Boot, PostgreSQL, React та TypeScript.
5. Спроєктовано архітектуру системи, базу даних, ролі користувачів та UML-діаграми.
6. Реалізовано веб-застосунок LogiTrack з модулями доставок, аналітики, довідників, профілю та користувачів.
7. Проведено тестування функціональності, рольового доступу та модуля оцінювання ризиків.

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Delivery.java

```
package com.example.logitrack.entity;

import
com.example.logitrack.entity.enums.DeliveryStatus;
import jakarta.persistence.*;
import lombok.*;
import org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;

import java.math.BigDecimal;
import java.time.LocalDateTime;

@Entity
@Table(name = "deliveries")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Delivery {

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    @Column(name = "reference_number", nullable =
false, unique = true, length = 40)
    private String referenceNumber;

    @Column(name = "customer_name", nullable = false,
length = 140)
    private String customerName;

    @Column(name = "cargo_description", nullable =
false, length = 180)
    private String cargoDescription;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false, length = 30)
    private DeliveryStatus status;

    @Column(nullable = false, length = 30)
    private String priority;

    @Column(name = "weight_kg", precision = 10, scale
= 2)
    private BigDecimal weightKg;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "route_id", nullable = false)
    private DeliveryRoute route;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "vehicle_id", nullable = false)
    private Vehicle vehicle;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "origin_warehouse_id",
nullable = false)
    private Warehouse originWarehouse;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "destination_warehouse_id",
nullable = false)
    private Warehouse destinationWarehouse;

    @Column(name = "planned_departure", nullable =
false)
    private LocalDateTime plannedDeparture;

    @Column(name = "planned_arrival", nullable = false)
    private LocalDateTime plannedArrival;

    @Column(name = "actual_arrival")
    private LocalDateTime actualArrival;

    @Column(name = "delay_minutes", nullable = false)
    private Integer delayMinutes;

    @Column(length = 900)
    private String notes;

    @CreationTimestamp
    @Column(name = "created_at", updatable = false)
    private LocalDateTime createdAt;

    @UpdateTimestamp
    @Column(name = "updated_at")
    private LocalDateTime updatedAt;
}
```

User.java

```
package com.example.logitrack.entity;

import
com.example.logitrack.entity.enums.UserRole;
import jakarta.persistence.*;
import lombok.*;
import
org.hibernate.annotations.CreationTimestamp;
import org.hibernate.annotations.UpdateTimestamp;

import java.time.LocalDateTime;

@Entity
@Table(name = "users")
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class User {

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;
```

```
    @Column(nullable = false, unique = true, length =
255)
    private String email;

    @Column(name = "password_hash", nullable = false)
    private String passwordHash;

    @Column(name = "full_name", nullable = false,
length = 140)
    private String fullName;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false, length = 30)
    @Builder.Default
    private UserRole role = UserRole.MANAGER;

    @Column(length = 40)
    private String phone;

    @Column(length = 120)
    private String position;

    @Column(length = 120)
    private String department;

    @Column(name = "avatar_color", length = 20)
    @Builder.Default
    private String avatarColor = "#2d4f49";

    @Column(nullable = false)
    @Builder.Default
    private Boolean active = true;

    @CreationTimestamp
    @Column(name = "created_at", updatable = false)
    private LocalDateTime createdAt;

    @UpdateTimestamp
    @Column(name = "updated_at")
    private LocalDateTime updatedAt;
}
```

UserRole.java

```
        package com.example.logitrack.entity.enums;

public enum UserRole {
    ADMIN,
    MANAGER,
    VIEWER,
    USER
}
```

DeliveryController.java

```
        package com.example.logitrack.controller;

        import
com.example.logitrack.dto.request.DeliveryRequest;
        import
com.example.logitrack.dto.response.DeliveryResponse;
        import
com.example.logitrack.service.LogisticsService;
        import jakarta.validation.Valid;
        import lombok.RequiredArgsConstructor;
        import org.springframework.http.HttpStatus;
        import
org.springframework.security.access.prepost.PreAuthoriz
e;
        import org.springframework.web.bind.annotation.*;

        import java.util.List;

@RestController
@RequestMapping("/api/deliveries")
@RequiredArgsConstructor
public class DeliveryController {

    private final LogisticsService logisticsService;

    @GetMapping
    public List<DeliveryResponse> getDeliveries() {
        return logisticsService.getDeliveries();
    }

    @GetMapping("/{id}")
    public DeliveryResponse getDelivery(@PathVariable
Long id) {
        return logisticsService.getDelivery(id);
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)

@PreAuthorize("hasAnyRole('ADMIN','MANAGER')")
    public DeliveryResponse createDelivery(@Valid
@RequestBody DeliveryRequest request) {
        return logisticsService.createDelivery(request);
    }

    @PutMapping("/{id}")

@PreAuthorize("hasAnyRole('ADMIN','MANAGER')")
    public DeliveryResponse updateDelivery(
            @PathVariable Long id,
            @Valid @RequestBody DeliveryRequest request
    ) {
        return logisticsService.updateDelivery(id, request);
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    @PreAuthorize("hasRole('ADMIN')")
    public void deleteDelivery(@PathVariable Long id) {
        logisticsService.deleteDelivery(id);
    }
}
```

RiskController.java

```
        package com.example.logitrack.controller;

        import
com.example.logitrack.dto.response.DeliveryRiskRespon
se;
        import
com.example.logitrack.service.LogisticsService;
        import lombok.RequiredArgsConstructor;
```

```
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/risks")
@RequiredArgsConstructor
public class RiskController {

   private final LogisticsService logisticsService;

   @GetMapping
   public List<DeliveryRiskResponse> getRisks() {
      return logisticsService.getRisks();
   }
}
```

DeliveryRiskResponse.java

```
package com.example.logitrack.dto.response;

import lombok.AllArgsConstructor;
import lombok.Data;

import java.util.List;

@Data
@AllArgsConstructor
public class DeliveryRiskResponse {

   private DeliveryResponse delivery;
   private Integer riskScore;
   private String riskLevel;
   private List<String> reasons;
   private String recommendation;
}
```

UserController.java

```
package com.example.logitrack.controller;

import
com.example.logitrack.dto.request.ProfileUpdateRequest
;
import
com.example.logitrack.dto.request.UserRoleUpdateRequ
est;
import
com.example.logitrack.dto.request.UserStatusUpdateReq
uest;
import
com.example.logitrack.dto.response.UserResponse;
import com.example.logitrack.service.UserService;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import
org.springframework.security.access.prepost.PreAuthoriz
e;
import
org.springframework.security.core.annotation.Authentica
tionPrincipal;
import
org.springframework.security.core.userdetails.UserDetail
s;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/api/users")
@RequiredArgsConstructor
public class UserController {

   private final UserService userService;

   @GetMapping("/me")
   public UserResponse
getCurrentUser(@AuthenticationPrincipal UserDetails
userDetails) {
      return
userService.getByEmail(userDetails.getUsername());
   }

   @PutMapping("/me")
   public UserResponse updateProfile(
         @AuthenticationPrincipal UserDetails
userDetails,
         @Valid @RequestBody ProfileUpdateRequest
request
   ) {
      return
userService.updateProfile(userDetails.getUsername(),
request);
   }

   @GetMapping
   @PreAuthorize("hasRole('ADMIN')")
   public List<UserResponse> getUsers() {
      return userService.getUsers();
   }

   @PutMapping("/{id}/role")
   @PreAuthorize("hasRole('ADMIN')")
   public UserResponse updateRole(
         @PathVariable Long id,
         @Valid @RequestBody UserRoleUpdateRequest
request
   ) {
      return userService.updateRole(id, request);
   }

   @PutMapping("/{id}/active")
   @PreAuthorize("hasRole('ADMIN')")
   public UserResponse updateStatus(
         @PathVariable Long id,
         @Valid @RequestBody
UserStatusUpdateRequest request
   ) {
      return userService.updateStatus(id, request);
   }
}
```

Фрагмент LogisticsService.java с уникальной логикой оценки рисков

```
@Transactional(readOnly = true)
public List<DeliveryRiskResponse> getRisks() {
      return
deliveryRepository.findAllByOrderByUpdatedAtDesc()
```

```
        .stream()
        .map(this::toRiskResponse)

.sorted(Comparator.comparing(DeliveryRiskResponse::getRiskScore).reversed())
        .toList();
        }

private DeliveryRiskResponse toRiskResponse(Delivery delivery) {
        List<String> reasons = new ArrayList<>();
        int score = 10;

        if (delivery.getStatus() == DeliveryStatus.DELAYED) {
        score += 35;
        reasons.add("Доставка вже має статус затримки");
        }

        if (delivery.getDelayMinutes() != null && delivery.getDelayMinutes() > 0) {
        int delayScore = Math.min(25, delivery.getDelayMinutes() / 4);
        score += delayScore;
        reasons.add("Поточна затримка " + delivery.getDelayMinutes() + " хв");
        }

        RouteResponse route = toRouteResponse(delivery.getRoute());

        if (route.getLoadPercent() >= 100) {
        score += 18;
        reasons.add("Маршрут завантажений на " + route.getLoadPercent() + "%");
        } else if (route.getLoadPercent() >= 80) {
        score += 10;
        reasons.add("Маршрут наближається до перевантаження");
        }

        if ("Критичний".equalsIgnoreCase(delivery.getPriority())) {
        score += 18;
        reasons.add("Критичний пріоритет клієнта");
        } else if ("Високий".equalsIgnoreCase(delivery.getPriority())) {
        score += 10;
        reasons.add("Високий пріоритет клієнта");
        }

        if (delivery.getVehicle().getStatus() == VehicleStatus.MAINTENANCE) {
        score += 20;
        reasons.add("Транспорт у сервісному статусі");
        }

        if (delivery.getOriginWarehouse().getCurrentLoadPercent() >= 80
        ||
delivery.getDestinationWarehouse().getCurrentLoadPercent() >= 80) {
        score += 12;
        reasons.add("Один зі складів завантажений понад 80%");
        }

        if (reasons.isEmpty()) {
        reasons.add("Ознак критичного ризику не виявлено");
        }

        int normalizedScore = Math.min(100, score);
        String level = riskLevel(normalizedScore);

        return new DeliveryRiskResponse(
        toDeliveryResponse(delivery),
        normalizedScore,
        level,
        reasons,
        recommendationFor(level)
        );
        }

private String riskLevel(int score) {
        if (score >= 80) {
        return "CRITICAL";
        }
        if (score >= 60) {
        return "HIGH";
        }
        if (score >= 35) {
        return "MEDIUM";
        }
        return "LOW";
        }

private String recommendationFor(String level) {
        return switch (level) {
        case "CRITICAL" -> "Потрібно терміново змінити маршрут або транспорт.";
        case "HIGH" -> "Рекомендується перевірити маршрут та складські навантаження.";
        case "MEDIUM" -> "Потрібен додатковий контроль логіста.";
default -> "Доставка виконується у нормальному режимі.";
        };
        }
```

SecurityConfig.java

```
        package com.example.logitrack.config;

        import com.example.logitrack.security.JwtAuthenticationFilter;
        import com.example.logitrack.security.UserDetailsServiceImpl;
        import lombok.RequiredArgsConstructor;
        import org.springframework.context.annotation.*;
        import org.springframework.security.authentication.AuthenticationManager;
        import
```

```
org.springframework.security.authentication.dao.DaoAuthenticationProvider;
	import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;
	import org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;
	import org.springframework.security.config.annotation.web.builders.HttpSecurity;
	import org.springframework.security.config.http.SessionCreationPolicy;
	import org.springframework.security.crypto.bcrypt.*;
	import org.springframework.security.crypto.password.PasswordEncoder;
	import org.springframework.security.web.*;
	import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;
	import org.springframework.web.cors.*;

	import java.util.List;

@Configuration
@EnableMethodSecurity
@RequiredArgsConstructor
public class SecurityConfig {

   private final JwtAuthenticationFilter jwtAuthenticationFilter;
   private final UserDetailsServiceImpl userDetailsService;

   @Bean
   public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
      return http
            .csrf(csrf -> csrf.disable())
            .cors(cors -> cors.configurationSource(corsConfigurationSource()))
            .sessionManagement(session ->

session.sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            )
            .authorizeHttpRequests(auth -> auth

.requestMatchers("/api/auth/**").permitAll()
                  .requestMatchers("/api-docs/**", "/swagger-ui/**", "/swagger-ui.html").permitAll()
                  .anyRequest().authenticated()
            )

.authenticationProvider(authenticationProvider())
            .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class)
            .build();
   }

   @Bean
   public DaoAuthenticationProvider authenticationProvider() {
      DaoAuthenticationProvider provider = new DaoAuthenticationProvider(userDetailsService);
      provider.setPasswordEncoder(passwordEncoder());
      return provider;
   }

   @Bean
   public PasswordEncoder passwordEncoder() {
      return new BCryptPasswordEncoder();
   }

   @Bean
   public AuthenticationManager authenticationManager(
         AuthenticationConfiguration configuration
   ) throws Exception {
      return configuration.getAuthenticationManager();
   }

   @Bean
   public CorsConfigurationSource corsConfigurationSource() {
      CorsConfiguration config = new CorsConfiguration();

      config.setAllowedOrigins(List.of(
            "http://localhost:3000",
            "http://localhost:3001",
            "http://localhost:3004",
            "http://localhost:5173",
            "http://127.0.0.1:3000",
            "http://127.0.0.1:3001",
            "http://127.0.0.1:3004",
            "http://127.0.0.1:5173"
      ));

      config.setAllowedMethods(List.of("GET", "POST", "PUT", "PATCH", "DELETE", "OPTIONS"));
      config.setAllowedHeaders(List.of("*"));
      config.setAllowCredentials(true);

      UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
      source.registerCorsConfiguration("/**", config);

      return source;
   }
}
```