

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

**«Система оптимізації логістичних маршрутів
для вантажних перевезень»**

Виконав студент групи ІПЗк-23-1
БІЛОТІЛ Ярослав Олександрович

Керівник роботи:
ЄФРЕМОВ Юрій Миколайович

Рецензент:
САВІЦЬКИЙ Роман Святославович

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БІЛОТІЛ Ярослав Олександрович**

Керівник роботи: **ЄФРЕМОВ Юрій Миколайович**

Тема роботи: **«Система оптимізації логістичних маршрутів для вантажних перевезень»**, затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: система логістичних перевезень вантажів; процеси та інструменти оптимізації маршрутів транспортування вантажів.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю.М.	27.02.2025р.	27.02.2025р.
2	Єфремов Ю.М.	27.02.2025р.	27.02.2025р.
3	Єфремов Ю.М.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Пояснювальна записка до бакалаврської кваліфікаційної роботи на тему:
«Система оптимізації логістичних маршрутів для вантажних перевезень»

Пояснювальна записка містить 75 сторінок, 31 рисунок, 23 таблиці, 39 використаних джерел та 1 додаток.

Метою роботи є створення вебсистеми для автоматизованого планування логістичних маршрутів із урахуванням витрат, часу доставки та дорожньої ситуації. У роботі проаналізовано сучасні підходи до оптимізації маршрутів (алгоритми Dijkstra, A*, транспортна задача) та реалізовано програмне рішення з використанням Flask, Google Maps API і бібліотеки PuLP.

Система підтримує ролеву авторизацію (адміністратор, диспетчер, водій), має модулі керування маршрутами, замовленнями, транспортом і аналітикою. Інтерфейс дозволяє формувати маршрути, зберігати їх історію, враховувати обмеження транспорту й трафік.

Запропоноване рішення підвищує ефективність перевезень, знижує витрати та покращує керування логістикою.

КЛЮЧОВІ СЛОВА: ЛОГІСТИКА, ОПТИМІЗАЦІЯ МАРШРУТІВ, ВАНТАЖНІ ПЕРЕВЕЗЕННЯ, GOOGLE MAPS, ТРАНСПОРТНА ЗАДАЧА, ІНФОРМАЦІЙНА СИСТЕМА.

					ІПЗ.КР.Б – 121-25 - ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Білотіл Я.О.			Система оптимізації логістичних маршрутів для вантажних перевезень Пояснювальна записка			Літ.	Арк.	Аркуші	
Перевір.		Єфремов Ю.М.								3	94
Т. контр.								Житомирська політехніка, група ІПЗк-23-1			
Н. контр.											
Затверд.											

ABSTRACT

Explanatory note to the bachelor's qualification work on the topic: "Logistics route optimization system for freight transportation".

The explanatory note contains 75 pages, 31 figures, 23 tables, 39 sources used, and 1 appendix.

The purpose of the work is to create a web system for automated planning of logistics routes taking into account costs, delivery time and traffic situation. The work analyzes modern approaches to route optimization (Dijkstra, A* algorithms, transport problem) and implements a software solution using Flask, Google Maps API and the PuLP library.

The system supports role-based authorization (administrator, dispatcher, driver), has modules for managing routes, orders, transport and analytics. The interface allows you to create routes, store their history, take into account transport restrictions and traffic.

The proposed solution increases transportation efficiency, reduces costs and improves logistics management.

KEYWORDS: LOGISTICS, ROUTES OPTIMIZATION, FREIGHT TRANSPORTATION, GOOGLE MAPS, TRANSPORTATION PROBLEM, INFORMATION SYSTEM.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ МАРШРУТІВ ДЛЯ ВАНТАЖНИХ ПЕРЕВЕЗЕНЬ	10
1.1. Постановка задачі	10
1.2. Аналіз аналогів програмного продукту	12
1.3. Вибір архітектури системи.....	18
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	21
Висновки до розділу 1	24
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ МАРШРУТІВ ДЛЯ ВАНТАЖНИХ ПЕРЕВЕЗЕНЬ	25
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	25
2.2. Розробка бази даних системи.....	35
2.3. Проектування та реалізація алгоритмів роботи системи	38
2.4. Реалізація функціоналу системи оптимізації логістичних маршрутів	40
Висновки до розділу 2	44
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З РЕАЛІЗОВАНОЮ СИСТЕМОЮ	45
3.1. Порядок встановлення та налаштування параметрів логістичної системи.....	45
3.2. Опис інтерфейсу вебзастосунку LogiRoute	47
Висновки до розділу 3	71

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ВИСНОВКИ 72

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 73

ДОДАТКИ..... 76

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API — Application Programming Interface — інтерфейс прикладного програмування, що дозволяє взаємодіяти між клієнтською частиною та сервером.

JWT — JSON Web Token — стандарт безпечної передачі даних між клієнтом і сервером, який використовується для автентифікації користувачів.

CRUD — Create, Read, Update, Delete — основні операції над даними, що реалізовані в системі для управління користувачами, маршрутами, транспортом тощо.

DB — Database — база даних, що використовується для зберігання інформації про користувачів, маршрути, транспортні засоби, замовлення тощо.

GPS — Global Positioning System — глобальна система позиціонування, яка використовується для відстеження транспорту на мапі.

HTML — HyperText Markup Language — мова розмітки, що використовується для побудови структури веб-інтерфейсу системи.

CSS — Cascading Style Sheets — каскадні таблиці стилів, які використовуються для оформлення інтерфейсу системи.

JS — JavaScript — мова програмування, що реалізує логіку дій на стороні клієнта (в тому числі обробку форм, взаємодію з мапою тощо).

SQLAlchemy — ORM-бібліотека для Python, яка використовується у серверній частині для зручної роботи з базою даних.

PuLP — бібліотека Python для лінійного програмування, використовується для реалізації алгоритму оптимізації транспортної задачі.

Flask — мікрофреймворк для Python, який використовується для створення серверної частини вебдодатку.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми полягає у тому, що у сучасному світі транспортна логістика відіграє важливу роль у забезпеченні ефективного переміщення товарів між пунктами призначення. Зростання обсягів перевезень, конкуренція між компаніями та вимоги до оперативності змушують підприємства шукати рішення, які дозволяють автоматизувати процеси та зменшити витрати. Одним з таких рішень є оптимізація логістичних маршрутів.

Класичні методи планування маршрутів зазнають труднощів при врахуванні великої кількості факторів: дорожні умови, затори, аварії, обмеження по транспорту, терміни доставки тощо. Водночас сучасні інформаційні технології дозволяють створювати інтелектуальні системи, які здатні автоматично будувати оптимальні маршрути з урахуванням багатьох змінних і реальних умов.

Метою даної роботи є розробка вебзастосунку для побудови та оптимізації логістичних маршрутів вантажних перевезень. Система повинна автоматизувати процес планування маршрутів, забезпечити облік транспорту, призначення водіїв і контроль виконання, а також надавати можливість реагування на зміни дорожньої ситуації.

У межах кваліфікаційної роботи було спроектовано та реалізовано вебзастосунок, який включає побудову маршрутів на основі Google Maps API з урахуванням трафіку та подій на дорогах, оптимізацію логістичних маршрутів за допомогою транспортної задачі, реалізованої через бібліотеку PuLP, підтримку трьох ролей користувачів (адміністратор, диспетчер, водій) з різними функціональними можливостями, функціонал для створення, збереження, редагування і перегляду маршрутів, замовлень та транспортних засобів, а також систему авторизації на основі JWT і зручний інтерфейс із використанням HTML, CSS (Bootstrap) і JavaScript.

Для досягнення поставленої мети було вирішено низку завдань: проведено аналіз існуючих методів оптимізації маршрутів і аналогічних

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

програмних рішень; обрано архітектуру, інструменти та засоби реалізації системи; розроблено алгоритм оптимізації на основі транспортної задачі; реалізовано інтерфейс для трьох типів користувачів; створено модулі побудови маршруту, збереження історії, управління замовленнями та транспортом; протестовано роботу системи в реальних сценаріях.

Об’єкт дослідження — процес організації логістичних перевезень, що охоплює планування, контроль та виконання доставок вантажів між пунктами призначення з урахуванням часу, витрат, доступних ресурсів і дорожньої ситуації. До процесу входять операції з маршрутизації, диспетчеризації, вибору транспорту та взаємодії між учасниками логістичного ланцюга.

Предмет дослідження — веборієнтована інформаційна система, що забезпечує оптимізацію логістичних маршрутів для вантажних перевезень шляхом автоматизованого розрахунку оптимальних шляхів, врахування транспортних обмежень, часу доставки, трафіку та витрат. Система також включає функції керування маршрутами, замовленнями, транспортними засобами, користувачами та аналітичними даними.

Методи дослідження, використані в роботі: об’єктно-орієнтоване проектування, системний аналіз, побудова математичних моделей (транспортна задача), робота з API Google Maps, веброзробка із використанням Flask, HTML/CSS/JavaScript, створення REST API, робота з базами даних MySQL.

Практичне значення проєкту полягає в тому, що розроблена система може бути застосована логістичними компаніями малого та середнього бізнесу для оптимізації доставки, зниження витрат та підвищення якості обслуговування клієнтів. Платформа легко масштабується та може бути адаптована під потреби конкретного підприємства або галузі. Її використання сприяє цифровій трансформації логістичних процесів і підвищенню конкурентоспроможності компанії на ринку.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ МАРШРУТІВ ДЛЯ ВАНТАЖНИХ ПЕРЕВЕЗЕНЬ

1.1. Постановка задачі

У сучасних умовах динамічного розвитку галузі вантажних перевезень особливо актуальним є завдання підвищення ефективності процесу планування маршрутів доставки товарів. Необхідність оперативного і точного визначення оптимальних маршрутів обумовлена постійним зростанням обсягів перевезень, зростаючими вимогами до швидкості і надійності доставки, а також необхідністю мінімізації витрат на транспортування. Сучасна логістика стикається з низкою складних факторів, які впливають на якість планування маршрутів, серед яких стан дорожньої інфраструктури, наявність заторів, транспортні обмеження, вартість пального, часові вікна доставки та багато інших. Врахування цих факторів у ручному режимі є надзвичайно складним, тому постає завдання автоматизації процесу оптимізації маршрутів шляхом створення спеціалізованого програмного комплексу.

У межах даного проєкту було розроблено веборієнтовану систему для оптимізації логістичних маршрутів вантажних перевезень. Головна мета цієї системи — надати користувачам зручний інструмент для побудови оптимального маршруту з урахуванням часу, відстані, витрат на транспортування та поточної ситуації на дорогах. Система дозволяє вводити пункти відправлення та призначення, будувати маршрут на інтерактивній мапі з урахуванням трафіку та потенційних перешкод, обчислювати орієнтовну вартість доставки, а також зберігати маршрути в базу даних. Особливістю є підтримка ролей користувачів — водій, диспетчер, адміністратор — із відповідним рівнем доступу.

Для реалізації поставлених задач використовувалися сучасні вебтехнології. Серверна частина побудована на Flask (Python), що забезпечує взаємодію з базою даних MySQL, реалізацію API, обробку запитів та

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

авторизацію через JWT. На стороні клієнта використовувалися HTML, CSS і JavaScript, а також інтеграція з Google Maps API для побудови маршрутів, відображення заторів, дорожніх подій і обрахунку відстані. Інтерфейс системи є адаптивним та інтуїтивно зрозумілим для користувача.

Система підтримує реєстрацію користувачів і вхід з урахуванням ролі. Після входу водій бачить лише призначені йому маршрути та їх деталі, але не може змінювати статус маршруту. Диспетчер і адміністратор мають доступ до історії маршрутів усіх користувачів, можуть змінювати статус доставки, створювати замовлення, додавати транспорт, а також керувати обліковими записами. Для маршрутизації використано можливості Google Maps Directions API, що дозволяє враховувати дорожні умови в реальному часі. Крім того, реалізовано облік вартості маршруту за заданими тарифами.

Було створено сторінки маршрутизації з формами введення точок, вибору водія і транспорту, сторінку історії маршрутів із таблицею, фільтрацією та пошуком, а також інтерфейс керування транспортними засобами. Кожна дія в системі передбачає логічний контроль доступу залежно від ролі користувача, що забезпечує цілісність та безпеку даних. Маршрути зберігаються в базі з урахуванням параметрів: координати початку й кінця, список точок, відстань, орієнтовний час, ціна, статус виконання, водій, транспортний засіб та дата створення.

Особливу увагу приділено можливості розширення системи. Було закладено основу для додавання модуля обліку подій на дорозі з API HERE або Google Traffic, що дозволяє динамічно реагувати на зміни дорожньої ситуації. Розроблено REST API для взаємодії з клієнтською частиною, що дозволяє в майбутньому реалізувати мобільний застосунок або підключення до сторонніх сервісів.

Таким чином, у результаті розробки було реалізовано повноцінну вебсистему для побудови й керування логістичними маршрутами, яка враховує сучасні вимоги до зручності, масштабованості та точності. Така

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

система дозволяє логістичним компаніям автоматизувати процес планування, покращити контроль за виконанням перевезень і підвищити загальну ефективність роботи.

1.2. Аналіз аналогів програмного продукту

Для розробки ефективної системи оптимізації логістичних маршрутів необхідно проаналізувати існуючі безкоштовні або умовно безкоштовні програмні продукти та веб-сервіси, які надають подібний функціонал. Це дозволить виявити їхні переваги, недоліки та визначити унікальні аспекти майбутньої розробки.

1. Google Maps Platform

Опис:

Google Maps Platform надає інструменти для побудови маршрутів з урахуванням реального часу, заторів і різних типів транспорту. Користувачі можуть прокладати маршрути між багатьма точками, отримувати інформацію про трафік і прогнозований час доставки. Основним недоліком є обмежена кількість точок у маршруті (до 10) та відсутність спеціалізації для вантажних перевезень, таких як обмеження по вазі чи габаритах транспорту.

Google Maps Platform активно використовується у багатьох логістичних і транспортних рішеннях завдяки своїй надійності, швидкості та широкій підтримці API. Вона легко інтегрується з вебзастосунками, що дозволяє автоматизувати процеси побудови маршрутів і відображення інформації про доставку. Незважаючи на деякі обмеження, платформа залишається одним із найпопулярніших рішень для візуалізації маршрутів у реальному часі.

Зовнішній вигляд:

Інтерфейс Google Maps Platform є інтуїтивно зрозумілим, з інтерактивною картою, на якій можна прокладати маршрути, переглядати трафік у реальному часі та отримувати детальні інструкції щодо навігації.

Вигляд сайту, побудованого на Google Maps Platform зображено на рисунку 1.1.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

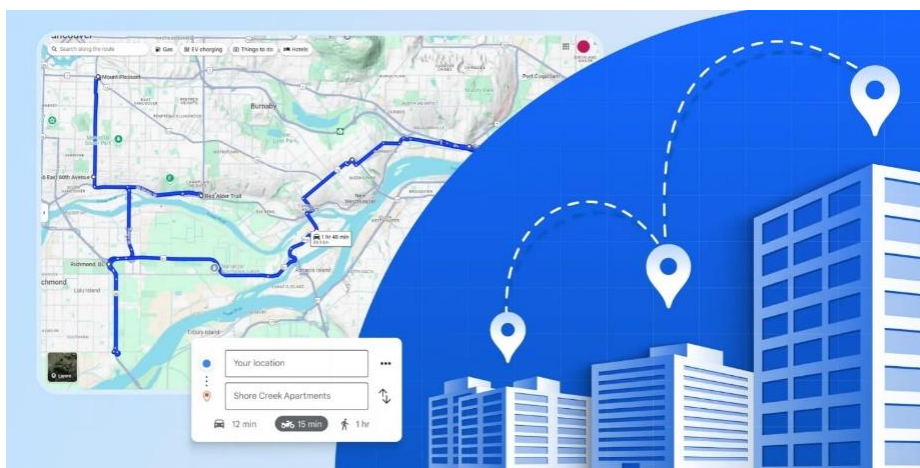


Рис. 1.1 – Вигляд сайту, побудованого на Google Maps Platform

2. HERE WeGo

Опис:

HERE WeGo — безкоштовний веб-сервіс і мобільний додаток для планування маршрутів, який дозволяє прокладати маршрути з урахуванням дорожніх умов і різних видів транспорту. Сервіс підтримує офлайн-карти та розрахунок часу маршруту. Основним недоліком є відсутність функцій для врахування специфіки вантажних перевезень та обмежені можливості для оптимізації багатозупинкових маршрутів.

Сервіс орієнтований переважно на приватних користувачів і пасажирські перевезення, а не на професійну логістику. Проте його можливість роботи в офлайн-режимі робить його корисним для навігації в зонах зі слабким покриттям. HERE WeGo має відкрите API, але воно менш популярне та менш гнучке порівняно з Google Maps.

Зовнішній вигляд:

Інтерфейс HERE WeGo сучасний та зручний, з можливістю перегляду маршруту, вибору типу транспорту та отримання детальної інформації про подорож.

Інтерфейс сервісу HERE WeGo зображено на рисунку 1.2.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

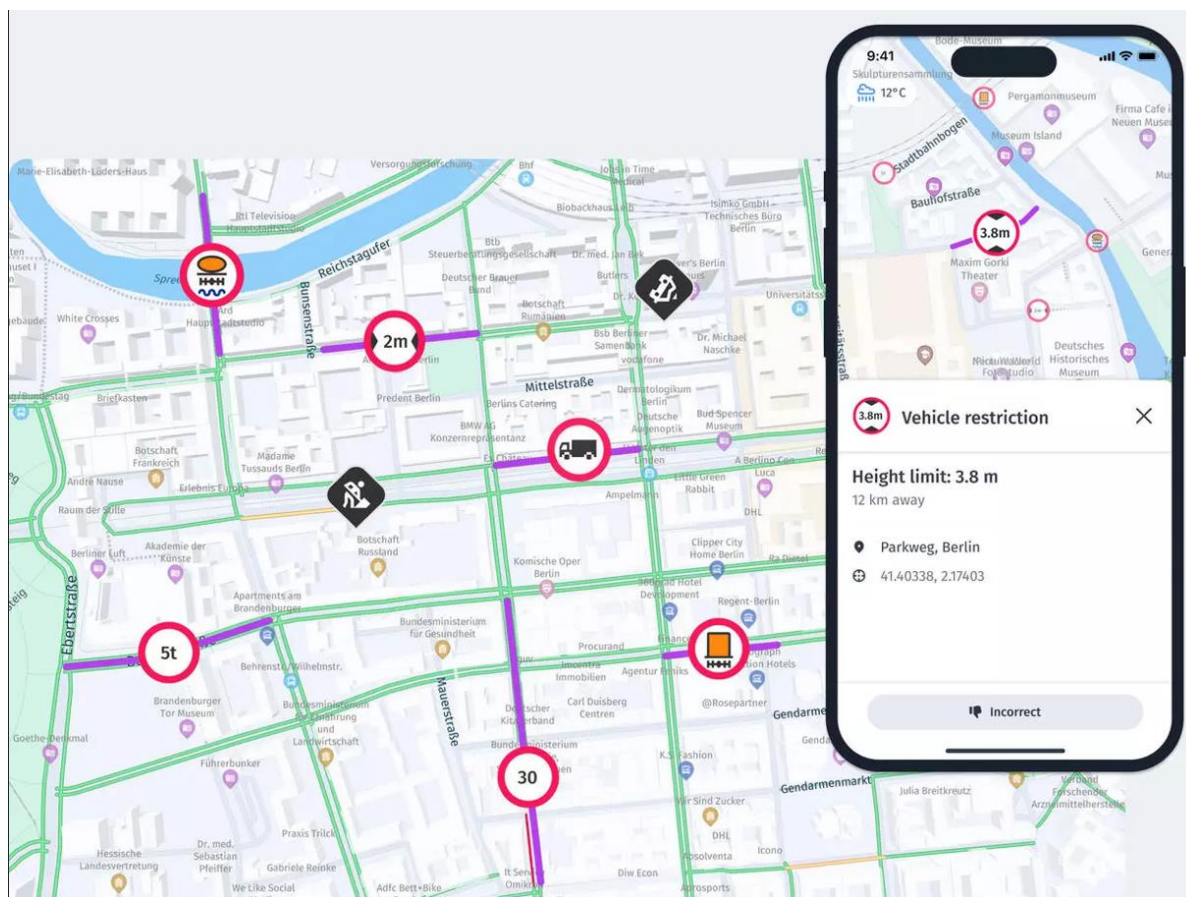


Рис. 1.2 - Вигляд сайту HERE WeGo

3. OpenRouteService

Опис:

OpenRouteService — безкоштовна платформа з відкритим вихідним кодом, що базується на даних OpenStreetMap. Вона дозволяє створювати маршрути з урахуванням пішохідних, велосипедних і автомобільних шляхів. Також доступна опція маршрутизації для вантажівок, що враховує обмеження по висоті, вазі тощо. Недоліком є менш детальна карта порівняно з Google Maps і менш зручний інтерфейс для кінцевого користувача.

Зовнішній вигляд:

Інтерфейс OpenRouteService функціональний, з можливістю вибору типу транспорту, додавання кількох точок маршруту та налаштування параметрів маршрутизації.

Інтерфейс платформи OpenRouteService зображено на рисунку 1.3.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

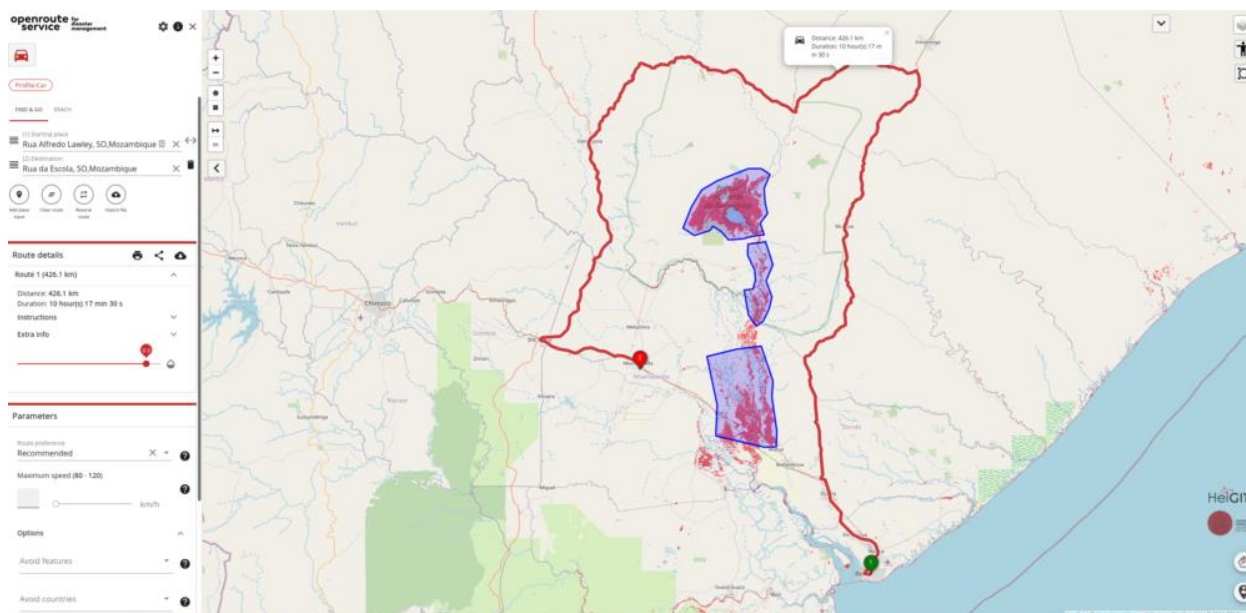


Рис. 1.3 – Інтерфейс OpenRouteService

4. GraphHopper

Опис:

GraphHopper Maps — безкоштовний веб-додаток з відкритим кодом, який використовує власний маршрутизатор на основі OpenStreetMap. Сервіс дозволяє планувати маршрути для різних видів транспорту (автомобіль, велосипед, пішохід), враховувати обмеження (наприклад, типи доріг або заборони руху) та аналізувати витрати на подорож. GraphHopper широко використовується для вбудовування у власні рішення завдяки відкритому API.

Його гнучкість та можливість налаштування параметрів маршрутизації робить сервіс особливо привабливим для розробників логістичних рішень. Разом з тим, для використання розширених функцій може знадобитися базове розуміння роботи з API та конфігурацією серверної частини.

Система включає основні функції, зокрема планування маршрутів з урахуванням обмежень, підтримку різних типів транспорту, а також відкрите API для інтеграції з іншими інформаційними системами.

Зовнішній вигляд:

Інтерфейс мінімалістичний і подібний до звичних картографічних сервісів.

Інтерфейс сайту GraphHopper зображено на рисунку 1.4.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

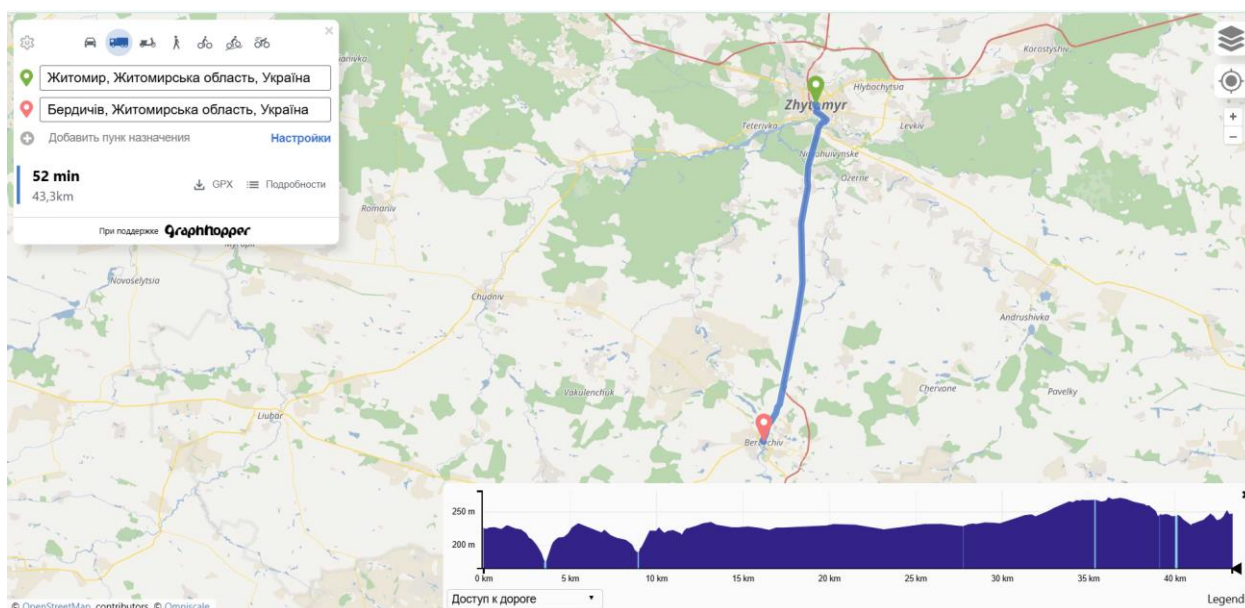


Рис. 1.4 – Вигляд програми GraphHooper

5. OSRM (Open Source Routing Machine)

Опис:

OSRM Demo — веб-демо одного з найшвидших у світі рушіїв пошуку маршруту на базі OpenStreetMap. Система дозволяє розраховувати маршрути для автомобільного, велосипедного та пішохідного транспорту, підтримує налаштування маршрутів і доступна як open-source рішення для впровадження у власні програмні продукти.

Серед основних функцій системи — дуже швидкий розрахунок маршрутів, підтримка кількох профілів транспорту, а також можливість локального розгортання на власному сервері.

Сервіс підходить для інтеграції у високонавантажені логістичні платформи, де важлива швидкість обробки запитів. Завдяки відкритому коду його можна адаптувати під потреби конкретного проєкту. Для повноцінного впровадження потрібні певні технічні навички — зокрема налаштування серверної інфраструктури та попередня підготовка картографічних даних.

Зовнішній вигляд:

Інтерфейс простий, з можливістю додавання кількох точок маршруту.

Інтерфейс системи зображено на рисунку 1.5.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

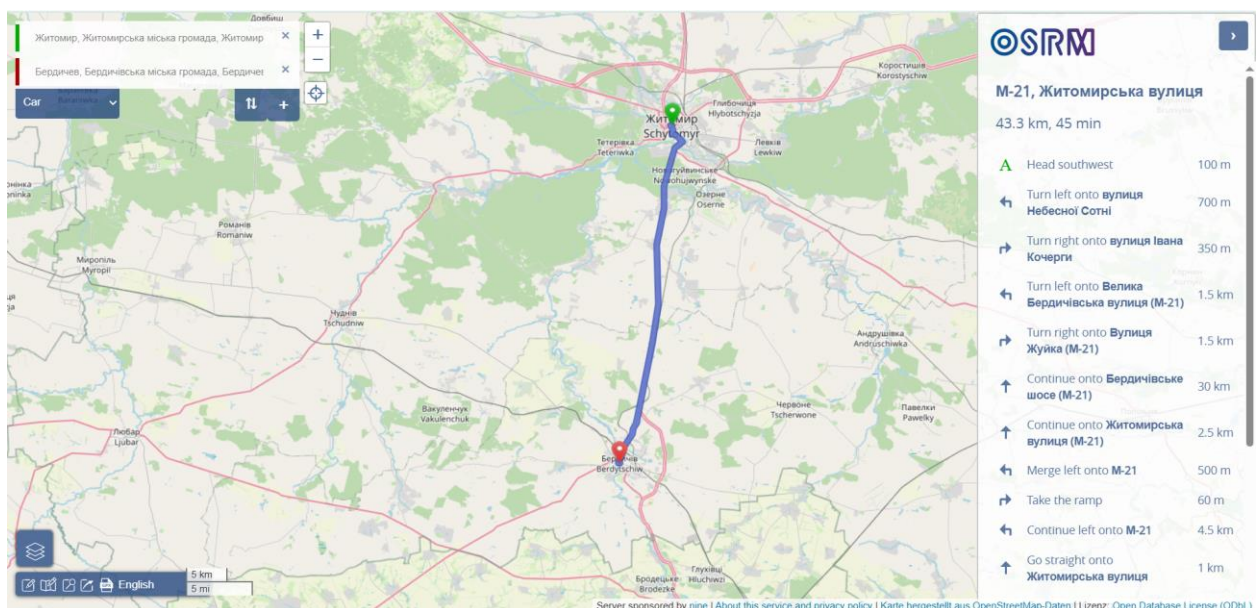


Рис. 1.5 – Вигляд програми OSRM

Порівняльна характеристика аналогів наведена у таблиці 1.1:

Таблиця 1.1

Таблиця порівняльної характеристики аналогів

Назва продукту	Тип транспорту	Підтримка динамічного трафіку	Інтеграція з API
Google Maps	Автомобіль, пішохід, велосипед, громадський транспорт	Так	Так (Google Maps API)
OpenRouteService	Автомобіль, пішохід, велосипед, вантажівки, інвалідні візки	Частково (дані заторів не завжди оновлюються в реальному часі)	Так (OpenRouteService API)
GraphHooper	Автомобіль, пішохід, велосипед	Ні	Так (GraphHooper API)
OSRM (Open Service Routing Machine)	Автомобіль, пішохід, велосипед	Ні	Так (OSRM API)
HERE WeGo	Автомобіль, пішохід, велосипед, громадський транспорт	Так	Так (HereRouting API)
LogiRoute (розроблена система)	Вантажний транспорт	Так (через Google Maps Traffic API)	Так (Google Maps API, REST API до Flask backend)

Назва продукту	Врахування обмежень транспорту	Відкритий код / Безкоштовність	Інтерфейс українською
Google Maps	Частково (заборони руху, типи доріг)	Безкоштовно (з обмеженнями API), не open-source	Так
OpenRouteService	Так (обмеження по висоті, вазі, типу дороги тощо)	Безкоштовно (API з обмеженнями), open-source	Ні
GraphHooper	Так	Безкоштовно (open-source)	Ні
OSRM (Open Service Routing Machine)	Так (можливість налаштувати профілі транспорту)	Безкоштовно (open-source)	Ні
HERE WeGo	Так	Безкоштовно (API частково платний), не open-source	Так
LogiRoute (розроблена система)	Так (можна враховувати обмеження по вазі, статус дороги, тип маршруту)	Відкритий код (на рівні дипломного проєкту), безкоштовна	Так (уся система українською мовою)

1.3. Вибір архітектури системи

Проектування програмного забезпечення вимагає ретельного вибору архітектури, яка забезпечить гнучкість, масштабованість, надійність і зручність підтримки майбутньої системи оптимізації логістичних маршрутів. У даному розділі виконано аналіз найбільш поширених архітектурних патернів, на основі якого обґрунтовано вибір архітектурного рішення для розробки системи.

Огляд архітектурних патернів

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

Для розробки систем, що працюють із великими обсягами даних, взаємодіють із зовнішніми сервісами та мають високі вимоги до доступності, зазвичай застосовуються такі архітектурні патерни:

Клієнт-серверна архітектура (Client-Server) — це класична модель, у якій клієнтська частина запитує ресурси або послуги у серверної частини. Така архітектура дозволяє централізовано керувати даними та логікою програми, забезпечує розмежування між інтерфейсом користувача та серверною логікою. Недоліком є можлива перевантаженість сервера при великій кількості запитів.

Трирівнева архітектура (Three-Tier Architecture) — ця модель розділяє програму на три рівні: презентаційний (інтерфейс користувача), логічний (обробка бізнес-логіки) та рівень даних (сховище даних). Така структура покращує масштабованість і спрощує обслуговування. Вона є розширенням клієнт-серверної моделі та краще підходить для складних систем.

Мікросервісна архітектура (Microservices Architecture) — цей патерн передбачає поділ системи на невеликі незалежні сервіси, кожен із яких виконує окрему функцію. Мікросервіси взаємодіють між собою через API. Ця архітектура забезпечує масштабованість, стійкість до збоїв окремих компонентів і гнучкість розробки. Недоліком є ускладнення розгортання та адміністрування системи.

Однорівнева архітектура (Monolithic Architecture) — весь додаток розробляється як єдиний програмний продукт, у якому всі компоненти інтегровані разом. Такий підхід швидкий на етапі розробки прототипів, однак має обмежену масштабованість і складний у підтримці при розростанні системи.

Архітектура REST API (RESTful Architecture) — цей підхід орієнтований на створення вебсервісів, які працюють через HTTP-протокол. REST дозволяє створювати легкі клієнт-серверні взаємодії, де ресурси представляються у вигляді URL і обробляються стандартними методами (GET, POST, PUT,

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

DELETE).

Обґрунтування вибору архітектури

Враховуючи специфіку розроблюваної системи оптимізації логістичних маршрутів, яка передбачає інтеграцію з картографічними сервісами (Google Maps API), обробку даних користувачів, маршрутів і замовлень у реальному часі, взаємодію з різними ролями користувачів та забезпечення безпеки доступу, доцільно було обрати трирівневу клієнт-серверну архітектуру у поєднанні з REST API. Це дозволяє легко реалізувати розділення відповідальностей, забезпечити гнучкість при подальшому масштабуванні, інтегруватися з зовнішніми сервісами, а також створити адаптивний інтерфейс для різних пристроїв.

У системі реалізовано клієнтську частину на основі HTML, CSS, JavaScript з використанням Bootstrap. Серверна логіка побудована на Flask (Python), що дозволило реалізувати REST API для обробки запитів, авторизації, розрахунку вартості маршруту та роботи з транспортною задачею. База даних реалізована на MySQL, де зберігаються маршрути, користувачі, замовлення, транспортні засоби, а також історія змін і статусів.

Опис обраної архітектури системи

На основі проведеного аналізу було реалізовано архітектуру, яка включає наступні компоненти:

Front-end — вебінтерфейс користувача, реалізований на HTML, CSS, JavaScript. Інтерфейс включає сторінки для створення маршрутів (routing.html), керування транспортом (vehicles.html), історії маршрутів (history.html), профілю користувача та інших. Через інтерфейс здійснюється введення точок маршруту, відображення результатів оптимізації, взаємодія з мапою Google Maps, візуалізація подій на дорогах і контроль стану доставки.

Back-end — серверна частина реалізована на Flask (Python). Вона обробляє запити клієнта, виконує логіку авторизації (JWT), маршрутизації, збереження даних у базу, реалізує алгоритм розв'язання транспортної задачі

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

для оптимізації вартості доставки. Інтеграція з Google Maps API дозволяє враховувати реальні дорожні умови, будувати точні маршрути та отримувати інформацію про тривалість, відстань, трафік тощо.

Database — база даних на MySQL з реляційною структурою. У ній зберігається вся інформація, пов'язана з користувачами, маршрутами, статусами, замовленнями та транспортом. Завдяки структурованим таблицям реалізовано логіку прив'язки замовлення до водія, маршруту до замовлення, обліку транспортних одиниць тощо.

Інтеграційні сервіси — взаємодія з зовнішніми сервісами здійснюється через REST API. Основна інтеграція реалізована з Google Maps Platform (Directions, Distance Matrix, Places API) для побудови маршрутів, автозаповнення адрес, врахування дорожньої ситуації. Передбачено можливість підключення додаткових API (наприклад, для отримання інформації про погодні умови або аварії).

Архітектура системи дозволяє масштабувати окремі компоненти залежно від навантаження. Наприклад, при зростанні кількості запитів можна виділити окремий сервер для обробки API або перенести обчислення оптимізації на окремий мікросервіс. Такий підхід забезпечує гнучкість та адаптивність системи до змін і розширення функціоналу.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки програмного комплексу, що дозволяє ефективно оптимізувати логістичні маршрути для вантажних перевезень, були обрані перевірені, сучасні й доступні інструменти. Основна мета вибору технологій полягала у забезпеченні зручності розробки, простоти масштабування, легкості підтримки та інтеграції з зовнішніми сервісами. Усі інструменти адаптовані для розробки вебзастосунку, який працює у форматі клієнт-серверної архітектури, з можливістю інтерактивної побудови маршрутів, авторизації користувачів, збереження даних у базу, а також відображення

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

даних у вигляді карти з маршрутом і точками доставки.

Серверна частина (Back-end) реалізована за допомогою фреймворку Flask на мові програмування Python. Flask — це легкий мікрофреймворк, який ідеально підходить для створення REST API. Завдяки йому було реалізовано всю серверну логіку: авторизацію користувачів через JWT, обробку запитів з фронтенду, взаємодію з базою даних, формування маршрутів, а також логіку розрахунку вартості доставки. Python обрано через його простоту, велику кількість готових бібліотек (зокрема для роботи з транспортними задачами) та добру інтеграцію з системами штучного інтелекту або оптимізації, якщо така інтеграція буде потрібна в майбутньому.

Клієнтська частина (Front-end) створена на основі стандартних вебтехнологій — HTML5, CSS3 та JavaScript. Ці технології дозволили побудувати зручний інтерфейс для взаємодії з користувачем. Для адаптивної верстки застосовано фреймворк Bootstrap, що дало змогу забезпечити коректне відображення елементів на різних пристроях — від ПК до планшетів. Усі дії на клієнтській частині обробляються асинхронно за допомогою JavaScript, що забезпечує швидку реакцію системи без перезавантаження сторінок. У системі реалізовано інтерфейс для створення маршруту, перегляду збережених маршрутів, керування замовленнями, транспортом та профілем користувача.

Базу даних реалізовано на системі керування базами даних MySQL. Вона забезпечує надійне збереження даних про користувачів, маршрути, замовлення, транспортні засоби, статуси виконання доставки та інші пов'язані об'єкти. MySQL обрано через її високу продуктивність, простоту використання, гнучкість і підтримку у Flask через SQLAlchemy.

Для картографічної частини системи було використано API Google Maps. Вибір саме Google Maps API обумовлений високою точністю карт, підтримкою української мови, наявністю інформації про дорожній трафік, затори, обмеження та аварії. У системі використано декілька модулів цього

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

API: Directions API — для побудови маршруту між точками, Distance Matrix API — для розрахунку часу і відстані, Places API — для автозаповнення адрес при введенні. Це дозволяє користувачеві швидко і зручно формувати маршрут, бачити його на карті, переглядати деталі доставки, а також враховувати ситуацію на дорогах.

Усі запити між клієнтською частиною і сервером реалізовані через REST API. Це дозволяє легко масштабувати систему, підключати нові клієнтські інтерфейси, мобільні додатки або сторонні сервіси. Крім того, REST API забезпечує чітке розділення логіки та зручну підтримку.

Для розміщення вебзастосунку використовується VPS-сервер на базі операційної системи Linux (наприклад, Ubuntu). Такий варіант хостингу дозволяє повністю контролювати середовище виконання, встановлювати необхідні бібліотеки, оновлювати компоненти системи та забезпечувати безперервну роботу сервера. В якості платформи для розміщення розглядаються українські хостинг-провайдери з підтримкою Flask, Python та MySQL, що дозволяє уникнути складнощів з локалізацією та юридичною відповідністю.

Орієнтовні вимоги до апаратного забезпечення включають наявність двоядерного процесора з тактовою частотою не менше 2.0 ГГц, щонайменше 4 ГБ оперативної пам'яті (залежно від кількості одночасних користувачів), дискового простору від 50 ГБ для зберігання бази даних, логів, маршрутів і картографічних даних, а також стабільного інтернет-з'єднання з високою пропускною здатністю для забезпечення доступу до API Google Maps та обробки запитів у режимі реального часу.

Такі характеристики дозволяють забезпечити стабільну роботу серверної частини, швидку обробку запитів і комфортну роботу користувачів із системою в реальному режимі. У разі збільшення навантаження систему можна масштабувати горизонтально (додавати сервери) або виділити окремий сервер під інтенсивну обробку маршрутів. Обрані інструменти і технології

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

підтвердили свою надійність під час тестування системи на прикладних даних та демонструють хорошу продуктивність, сумісність і гнучкість для подальшого розвитку програмного продукту.

Висновки до розділу 1

У ході аналізу методів оптимізації логістичних маршрутів для вантажних перевезень було підтверджено актуальність автоматизації процесу планування маршрутів з урахуванням великої кількості факторів, таких як стан дорожньої інфраструктури, транспортні обмеження, вартість пального, часові вікна доставки та інші умови, що змінюються в реальному часі. Розглянуто основні підходи до вирішення задач маршрутизації, зокрема алгоритми пошуку найкоротших шляхів (Дейкстри, A^*), евристичні та метаевристичні методи, а також елементи динамічного програмування.

Проведений огляд сучасних сервісів та програмних продуктів (Google Maps Platform, HERE WeGo, OpenRouteService, GraphHopper, OSRM) дозволив виявити їх сильні сторони та обмеження, що, у свою чергу, дало змогу сформулювати вимоги до власної системи. Було обґрунтовано вибір архітектурного підходу, інструментів реалізації, базових алгоритмів та сервісів для інтеграції.

У результаті аналізу сформовано концепцію створення програмного комплексу, який дозволяє автоматизовано формувати оптимальні логістичні маршрути з урахуванням реальних обмежень, інтегруватися з картографічними сервісами (Google Maps API), забезпечити зручний вебінтерфейс, підтримку ролей користувачів та точні розрахунки відстані, часу і вартості доставки. Отримані результати стали основою для реалізації функціональної вебсистеми, опис якої подано у наступних розділах роботи. Система враховує не лише географічні координати, а й специфіку вантажу та транспортного засобу, що дозволяє підвищити точність планування. Запропонований підхід може бути адаптований до різних сфер логістики та масштабів підприємства.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ОРІЄНТОВАНОЇ СИСТЕМИ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ МАРШРУТІВ ДЛЯ ВАНТАЖНИХ ПЕРЕВЕЗЕНЬ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Аналіз вимог до системи

Метою створеної системи є автоматизована побудова та оптимізація логістичних маршрутів для вантажних перевезень з урахуванням параметрів вантажу, характеристик транспорту, дорожніх обмежень та умов реального часу. Система орієнтована на диспетчерів, які формують маршрути доставки, водіїв, які отримують завдання, та адміністраторів, які налаштовують і супроводжують систему.

Функціональні вимоги

Система включає реєстрацію та авторизацію користувачів із розподілом ролей (адміністратор, диспетчер, водій), можливість додавання, редагування та прив'язки транспортних засобів до користувачів, створення замовлень із визначенням точки відправлення, доставки, водія та транспортного засобу, розрахунок оптимального маршруту за допомогою Google Maps API з урахуванням дорожніх умов, відображення маршруту на інтерактивній мапі, збереження історії маршрутів і статусів доставки, перегляд маршруту водієм та оновлення статусу замовлення, а також генерацію звітів і статистики для адміністратора.

Нефункціональні вимоги

Система повинна мати швидкий час відгуку (до 3 секунд на побудову маршруту), адаптивний веб-інтерфейс для різних пристроїв, забезпечувати безпечний обмін даними через JWT-аутентифікацію, захист від SQL-ін'єкцій та XSS. Робота системи має бути стабільною в режимі 24/7, з можливістю резервного копіювання даних.

Системні вимоги

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

Система оптимізації логістичних маршрутів вимагає підтримки операційних систем Windows, Linux (Ubuntu) і macOS для клієнтської частини. Серверна частина реалізована мовою Python із використанням фреймворку Flask та бібліотек Flask-JWT, Flask-CORS, SQLAlchemy і PuLP. Вебінтерфейс створено на основі HTML5, CSS3, JavaScript і Bootstrap 5. Для побудови маршрутів та взаємодії з картою використовується Google Maps API (Directions API, Places API, Traffic Layer). Як СУБД застосовано MySQL. Рекомендовані апаратні ресурси сервера: не менше 4 ГБ оперативної пам'яті та 10 ГБ вільного місця на диску. Система повинна функціонувати при стабільному інтернет-з'єднанні зі швидкістю не нижче 10 Мбіт/с. Забезпечено захист передавання даних через HTTPS, а також реалізовано механізми безпеки від SQL-ін'єкцій, XSS та CSRF-атак. Вебдодаток сумісний з браузерами Google Chrome, Mozilla Firefox, Microsoft Edge і Safari. Реалізовано функцію автозбереження змін маршруту та статусів, а також передбачено регулярне резервне копіювання бази даних MySQL.

Основні користувачі системи та їх функції наведені у таблиці 2.1.

Таблиця 2.1

Основні користувачі системи

№	Користувач	Опис ролі та функцій
1	Диспетчер-логіст	Планує маршрути доставки, створює та редагує замовлення, закріплює транспорт і водіїв, переглядає історію маршрутів та звіти.
2	Адміністратор	Управляє користувачами (створення, редагування, видалення), налаштовує довідники, має повний доступ до всіх функцій.
3	Водій (користувач)	Переглядає призначені йому маршрути, бачить деталі перевезення, змінює статус маршруту (наприклад, «виконано»).

Сценарії використання системи (Use Cases) описані нижче, а діаграма зображена на рисунку 2.1.

UC1 — Авторизація користувача (через JWT)

UC2 — Додавання нового транспортного засобу (диспетчер/адміністратор)

UC3 — Додавання нового вантажу або замовлення (диспетчер)

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

UC4 — Створення нового маршруту доставки (диспетчер)

UC5 — Автоматизована побудова маршруту за допомогою Google Maps API (з урахуванням обмежень, відстані, часу)

UC6 — Візуалізація маршруту на інтерактивній карті (Leaflet / Google Maps)

UC7 — Збереження маршруту в базу даних (із даними про користувача, транспорт, дату, статус)

UC8 — Перегляд історії маршрутів, фільтрація, експорт звітів (диспетчер, адміністратор)

UC9 — Управління користувачами: реєстрація, редагування, зміна ролей (адміністратор)

UC10 — Перегляд маршруту водієм і оновлення статусу (водій)

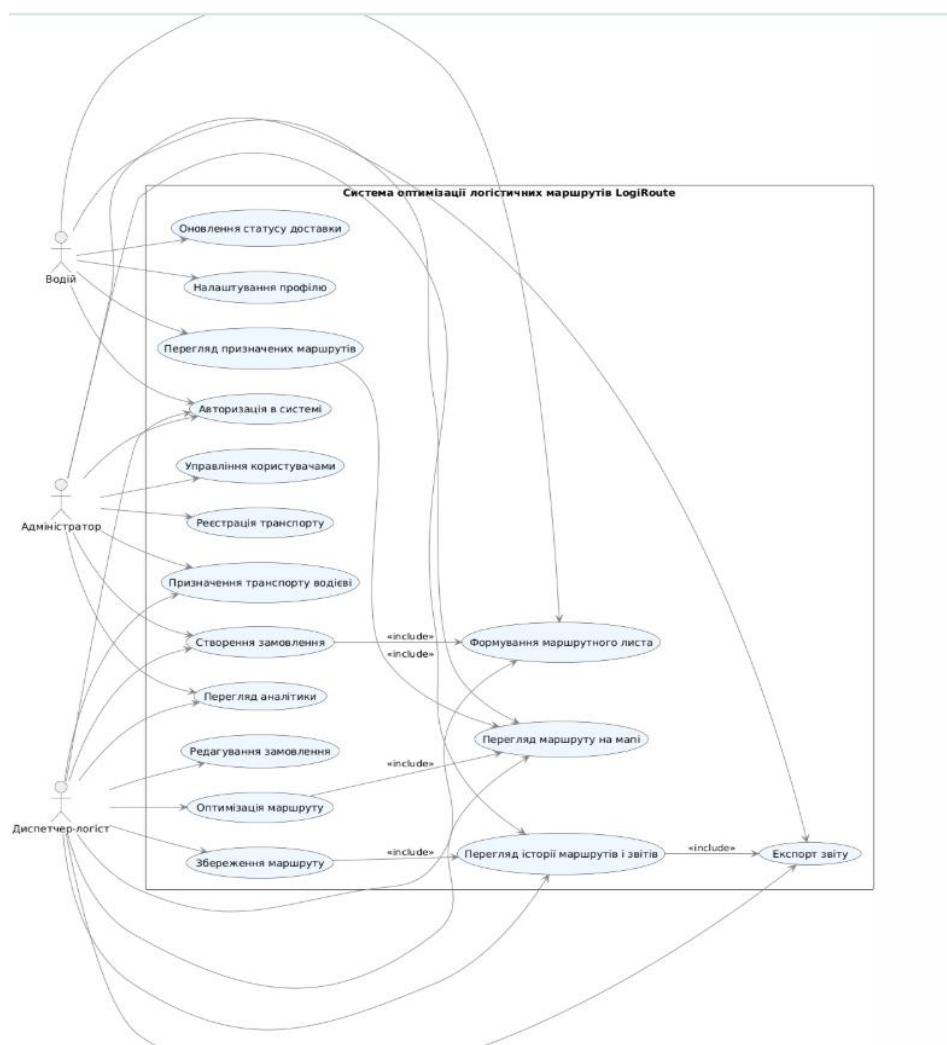


Рис. 2.1 – Діаграма варіантів використання системи

Опис діаграми варіантів використання наведено в таблицях 2.2 – 2.17.

Таблиця 2.2

Опис варіанту використання «Авторизація в системі»

Назва варіанту використання	Авторизація в системі
Основні актори	Адміністратор, Диспетчер-логіст, Водій
Короткий опис	Користувач вводить логін і пароль. Система перевіряє облікові дані та надає доступ до функцій відповідно до ролі.

Таблиця 2.3

Опис варіанту використання «Управління користувачами»

Назва варіанту використання	Управління користувачами
Основні актори	Адміністратор
Короткий опис	Адміністратор створює, редагує або видаляє облікові записи користувачів, задає ролі (диспетчер, водій) та налаштовує доступ.

Таблиця 2.4

Опис варіанту використання «Реєстрація транспорту»

Назва варіанту використання	Реєстрація транспорту
Основні актори	Адміністратор
Короткий опис	Додає нові транспортні засоби до системи, вказуючи тип, вантажопідйомність, номер, доступність тощо.

Таблиця 2.5

Опис варіанту використання «Створення замовлення»

Назва варіанту використання	Створення замовлення
Основні актори	Диспетчер-логіст
Короткий опис	Диспетчер вносить нове замовлення на перевезення, включаючи дані про вантаж, маршрут, водія і терміни доставки.

Таблиця 2.6

Опис варіанту використання «Редагування замовлення»

Назва варіанту використання	Редагування замовлення
Основні актори	Диспетчер-логіст
Короткий опис	Диспетчер може змінити деталі вже створеного замовлення — маршрут, водія, час, тип транспорту або вантажу.

Таблиця 2.7

Опис варіанту використання «Оптимізація маршруту»

Назва варіанту використання	Оптимізація маршруту
Основні актори	Диспетчер-логіст
Короткий опис	Система автоматично генерує найкращий маршрут для доставки вантажу з урахуванням часу, відстані, трафіку та параметрів вантажу.

Таблиця 2.8

Опис варіанту використання «Перегляд маршруту на карті»

Назва варіанту використання	Перегляд маршруту на карті
Основні актори	Диспетчер-логіст, Водій
Короткий опис	Відображення маршруту на інтерактивній мапі з позначенням початкової і кінцевої точки, зупинок та обмежень.

Таблиця 2.9

Опис варіанту використання «Збереження маршруту»

Назва варіанту використання	Збереження маршруту
Основні актори	Диспетчер-логіст
Короткий опис	Після формування маршруту диспетчер зберігає його в системі для подальшого призначення водієві та виконання рейсу.

Таблиця 2.10

Опис варіанту використання «Перегляд історії маршрутів та звітів»

Назва варіанту використання	Перегляд історії маршрутів та звітів
Основні актори	Адміністратор, Диспетчер-логіст
Короткий опис	Отримання звітів про виконані маршрути з деталізацією — час, витрати, відстань, статус доставки тощо.

Таблиця 2.11

Опис варіанту використання «Перегляд призначених маршрутів»

Назва варіанту використання	Перегляд призначених маршрутів
Основні актори	Водій
Короткий опис	Водій переглядає маршрути, призначені диспетчером, з усією необхідною інформацією для виконання перевезення.

Таблиця 2.12

Опис варіанту використання «Оновлення статусу доставки»

Назва варіанту використання	Оновлення статусу доставки
Основні актори	Водій
Короткий опис	Після завершення рейсу водій фіксує статус доставки (виконано, затримано, скасовано) для збереження в історії.

Таблиця 2.13

Опис варіанту використання «Призначення транспорту водієві»

Назва варіанту використання	Призначення транспорту водієві
Основні актори	Адміністратор / Диспетчер
Короткий опис	Адміністратор або диспетчер призначає конкретний транспорт для певного водія при створенні або редагуванні замовлення.

Таблиця 2.14

Опис варіанту використання «Формування маршрутного листа»

Назва варіанту використання	Формування маршрутного листа
Основні актори	Диспетчер-логіст / Водій
Короткий опис	Система автоматично формує маршрутний лист з усією інформацією про перевезення: маршрут, вантаж, час, контактні дані.

Таблиця 2.15

Опис варіанту використання «Перегляд аналітики»

Назва варіанту використання	Перегляд аналітики
Основні актори	Адміністратор / Диспетчер
Короткий опис	Користувач переглядає аналітичні графіки та статистику про виконані перевезення, затримки, завантаження транспорту тощо.

Таблиця 2.16

Опис варіанту використання «Налаштування профілю»

Назва варіанту використання	Налаштування профілю
Основні актори	Користувач
Короткий опис	Користувач змінює персональні налаштування акаунта, такі як ім'я, email, мова інтерфейсу, фото профілю тощо.

Таблиця 2.17

Опис варіанту використання «Експорт звіту»

Назва варіанту використання	Експорт звіту
Основні актори	Диспетчер-логіст / Адміністратор
Короткий опис	Користувач експортує звіт про перевезення у форматі Excel або PDF для подальшої обробки або звітності.

Вимоги користувачів до системи визначають основні функції, необхідні для ефективної роботи кожної ролі, та описані у таблиці 2.18 згідно з їхніми повноваженнями та очікуваним функціоналом.

Таблиця 2.18

Вимоги користувачів

№	Тип користувача	Вимога	Опис вимоги
1	Диспетчер-логіст	Побудова оптимального маршруту	Користувач вводить дані про вантаж і точки маршруту (відправлення/призначення), система генерує оптимальний маршрут доставки.
2	Диспетчер-логіст	Перегляд історії маршрутів	Диспетчер має доступ до архіву виконаних маршрутів із деталями перевезень для аналітики та звітності.
3	Адміністратор	Управління обліковими записами	Адміністратор створює, редагує або видаляє користувачів та керує їхніми правами доступу.
4	Адміністратор	Створення та редагування маршрутів	Має змогу вручну створювати маршрути або змінювати вже збережені, встановлювати час, точки та інші параметри.
5	Користувач (водій)	Перегляд маршрутів і вантажів	Користувач переглядає маршрути, до яких його призначено, а також супровідну інформацію щодо вантажу.
6	Користувач (усі ролі)	Зміна облікових даних	Користувач має можливість оновлювати свої персональні дані, e-mail, пароль тощо.
7	Диспетчер-логіст	Формування звітів про виконані маршрути	Має можливість створювати звіти про перевезення: маршрут, час, витрати, водій, вантаж.
8	Адміністратор	Видалення неактуальних маршрутів	Адміністратор може видаляти застарілі або некоректні маршрути для оптимізації бази.

9	Адміністратор	Налаштування ролей і прав доступу	Можливість встановлення або зміни рівня доступу для кожного користувача відповідно до ролі.
10	Диспетчер-логіст	Отримання сповіщень про зміни маршрутів	Налаштування системних повідомлень у разі зміни статусу маршруту, доставки чи непередбачених подій.

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать програмну реалізацію додатку. На рисунку 2.2 наведено діаграму класів, яка використовуються для його роботи.

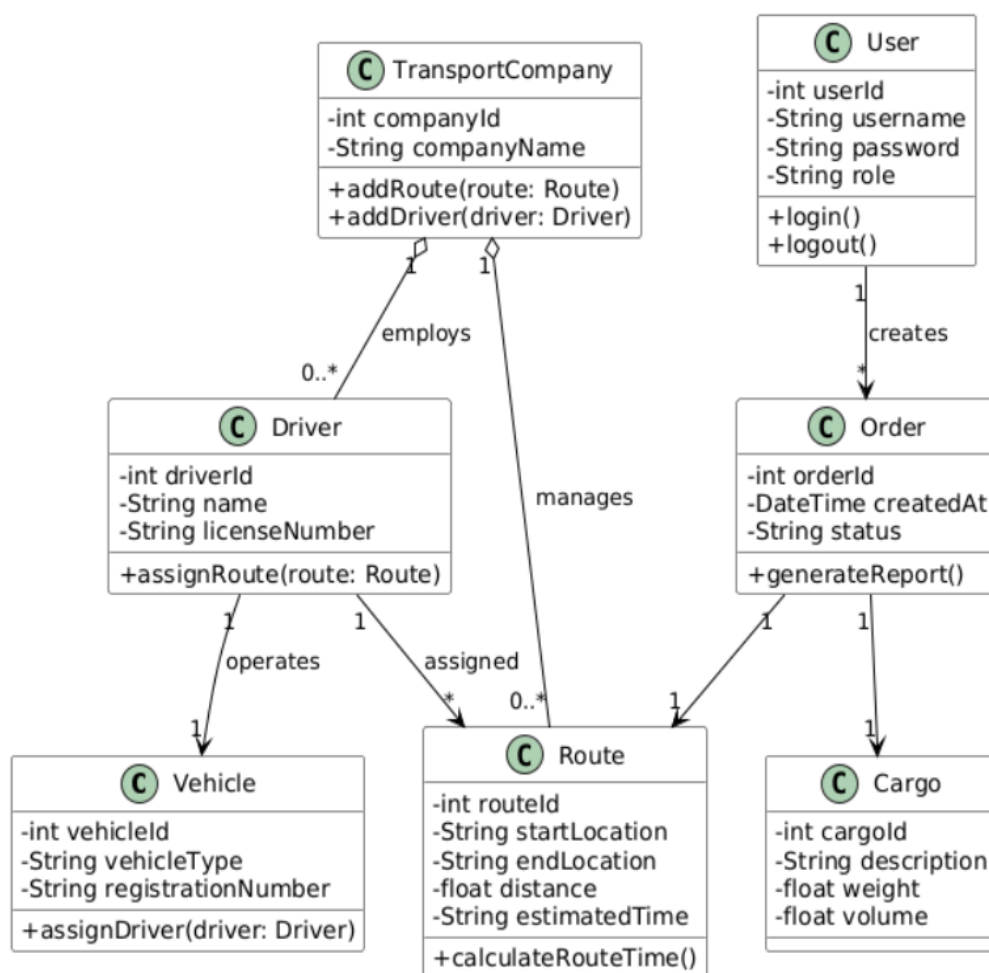


Рис. 2.2 – Діаграма класів системи

Опис діаграми класів:

На діаграмі класів зображено основні сутності системи оптимізації логістичних маршрутів LogiRoute, їхні атрибути, методи та зв'язки між ними.

Діаграма забезпечує структурне уявлення про об'єкти, які використовуються в системі, та їх взаємодію.

Клас User представляє користувача системи, який має змогу входити до системи, працювати з інтерфейсом та виконувати дії відповідно до своєї ролі (адміністратор, диспетчер-логіст, водій). Клас містить атрибути `userId`, `username`, `password`, `role`, а також методи `login()` і `logout()`.

Клас Driver описує водія, який виконує маршрути. До складу класу входять атрибути `driverId`, `name`, `licenseNumber` та метод `assignRoute(route)`, що дозволяє призначити маршрут конкретному водієві. Водій пов'язаний з транспортним засобом і виконує маршрут.

Клас Vehicle відповідає за опис транспортного засобу. Містить атрибути `vehicleId`, `vehicleType`, `registrationNumber` та метод `assignDriver(driver)` для закріплення водія за транспортом.

Клас Route відповідає за зберігання інформації про маршрут доставки: `routeId`, `startLocation`, `endLocation`, `distance`, `estimatedTime`. Має метод `calculateRouteTime()` для обчислення часу проходження маршруту. Кожен маршрут пов'язаний із замовленням і виконується водієм на транспортному засобі.

Клас Order описує замовлення, що створюється користувачем для виконання вантажного перевезення. Має атрибути `orderId`, `createdAt`, `status` і метод `generateReport()`. Замовлення пов'язується з маршрутом і вантажем.

Клас Cargo (вантаж) зберігає відомості про перевезення: `cargoId`, `description`, `weight`, `volume`. Кожне замовлення включає одне перевезення.

Клас TransportCompany представляє компанію-перевізника. Містить ідентифікаційні дані (`companyId`, `name`) та методи `addRoute(route)` і `addDriver(driver)` для управління маршрутами та водіями.

Взаємозв'язки між класами:

Клас TransportCompany пов'язаний із класами Driver та Route, оскільки компанія наймає водіїв і створює маршрути.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Driver пов'язаний із Vehicle — кожен водій закріплений за транспортним засобом.

Користувач (User) створює об'єкти Order, які в свою чергу пов'язані з маршрутами (Route) та вантажами (Cargo).

Один транспортний засіб може використовуватись у кількох замовленнях.

Діаграма класів забезпечує логічну структуру моделі предметної області, дозволяє визначити зв'язки між ключовими компонентами та є основою для розробки бази даних і програмної логіки системи.

2.2. Розробка бази даних системи

Для забезпечення зберігання та обробки інформації про вантажі, маршрути, транспортні засоби, водіїв і замовлення у системі оптимізації логістичних маршрутів було спроектовано реляційну базу даних. Основною метою створення бази даних є централізоване зберігання структурованої інформації, яка необхідна для ефективної взаємодії користувачів із системою та підтримки ключових бізнес-процесів перевезень.

Структура бази даних забезпечує можливість збереження даних про користувачів, замовлення на перевезення, характеристики вантажів, маршрути, транспортні засоби, водіїв, а також дозволяє встановлювати взаємозв'язки між цими об'єктами. Така організація дозволяє гарантувати цілісність інформації та забезпечити швидкий доступ до даних, що є критично важливим для побудови оптимальних логістичних маршрутів у режимі реального часу.

Модель бази даних розроблена з урахуванням принципів нормалізації, що дозволяє уникнути дублювання інформації та підвищити надійність системи. У структурі передбачено шість основних таблиць:

Users — таблиця, яка містить дані про користувачів системи: ідентифікатор користувача, ім'я користувача, пароль, роль (адміністратор, диспетчер, водій).

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Orders — таблиця, у якій зберігається інформація про оформлені заявки на перевезення, включаючи дату створення, статус виконання, пов'язаний маршрут та користувача, який створив замовлення.

Cargo — таблиця з інформацією про вантажі, які перевозяться: тип вантажу, вага, габарити, ідентифікатор замовлення.

Vehicles — таблиця, що містить відомості про транспортні засоби: тип, номерний знак, вантажопідйомність, стан.

Routes — таблиця, де фіксуються заплановані маршрути: початкова та кінцева точки маршруту, відстань, орієнтовний час у дорозі.

Drivers — таблиця з інформацією про водіїв: ім'я, номер водійського посвідчення, пов'язані транспортні засоби та маршрути.

Усі таблиці пов'язані між собою за допомогою зовнішніх ключів, що забезпечує логічну цілісність даних та уможливорює побудову складних запитів для аналітики або оптимізації логістичних рішень.

Опис таблиць наведено у таблицях 2.19 - 2.22:

Таблиця 2.19

Опис таблиці user

Поле	Тип	Призначення
id	INT	Унікальний ідентифікатор користувача
username	VARCHAR(100)	Ім'я користувача
email	VARCHAR(100)	Електронна адреса
password_hash	VARCHAR(255)	Захешований пароль
role	ENUM	Роль користувача (admin, dispatcher, driver)
photo	VARCHAR(255)	Шлях до фото профілю
created_at	DATETIME	Дата створення облікового запису
updated_at	DATETIME	Дата останнього редагування профілю
last_login	DATETIME	Дата останнього входу в систему

Таблиця 2.20

Опис таблиці order

Поле	Тип	Призначення
id	INT	Унікальний ідентифікатор заявки
user_id	INT	Ідентифікатор користувача (диспетчера)
vehicle_id	INT	Призначене авто
pickup	VARCHAR(255)	Точка відправлення
delivery	VARCHAR(255)	Точка доставки
weight	INT	Вага вантажу
type	VARCHAR(50)	Тип вантажу
deadline	DATE	Кінцевий термін доставки
status	VARCHAR(50)	Статус заявки

Опис таблиці vehicle:

Таблиця 2.21

Опис таблиці vehicle

Поле	Тип	Призначення
id	INT	Унікальний ідентифікатор транспортного засобу
name	VARCHAR(100)	Назва/марка транспортного засобу
type	VARCHAR(50)	Тип транспортного засобу
number_plate	VARCHAR(20)	Державний номер
capacity	INT	Вантажопідйомність (у кг)
status	VARCHAR(50)	Статус: «в роботі», «очікує» тощо
user_id	INT	Зовнішній ключ на користувача (водія)

Таблиця 2.22

Опис таблиці route

Поле	Тип	Призначення
id	INT	Унікальний ідентифікатор маршруту
user_id	INT	Диспетчер, який створив маршрут
vehicle_id	INT	Призначений транспорт
start_points	TEXT	Початкові точки маршруту
end_points	TEXT	Кінцеві точки маршруту
cargo_type	VARCHAR(50)	Тип вантажу (truck, container тощо)

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

weight	INT	Вага вантажу
volume	INT	Об'єм вантажу
category	VARCHAR(50)	Категорія вантажу (general, oversized)
departure	DATETIME	Час відправлення
distance	DECIMAL(10,2)	Відстань у км
duration	VARCHAR(50)	Тривалість маршруту
cost	DECIMAL(10,2)	Вартість доставки
created_at	DATETIME	Дата створення маршруту
status	VARCHAR(50)	Статус маршруту

2.3. Проектування та реалізація алгоритмів роботи системи

У системі оптимізації логістичних маршрутів LogiRoute реалізовано низку алгоритмів, що забезпечують її функціональну цілісність та ефективність. Центральне місце посідає алгоритм розрахунку оптимального маршруту. Його основою є адаптований під задачі логістики алгоритм Дейкстри, який не лише враховує довжину маршруту, а й аналізує ряд важливих параметрів — дорожню інфраструктуру, транспортні обмеження, поточний стан трафіку, часові вікна доставки та інші фактори. Такий підхід дозволяє формувати як найкоротші, так і найшвидші або економічно вигідні маршрути. Система підтримує побудову маршрутів з кількома точками доставки, що особливо важливо при обслуговуванні кількох замовлень одним транспортним засобом.

Під час створення замовлення користувач вводить необхідні параметри, зокрема адресу завантаження й розвантаження, тип і характеристики вантажу. На основі цих даних спрацьовує алгоритм валідації, який перевіряє правильність заповнення, коректність значень, доступність транспортних засобів і відповідність маршруту технічним параметрам вантажу. Якщо все коректно, система запускає механізм автоматичного призначення водія та транспортного засобу. У цьому процесі враховується не лише доступність техніки та персоналу, а й відповідність за габаритами, вантажопідйомністю,

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

поточним завантаженням і розташуванням, що дозволяє ефективно використовувати ресурси підприємства.

Окремо варто згадати реалізований алгоритм моніторингу стану перевезення. Система дозволяє відстежувати зміну статусу маршруту — від створення до завершення доставки, із фіксацією ключових часових міток. Це дає змогу диспетчерам в режимі реального часу бачити, коли транспорт виїхав, прибув до місця призначення або завершив рейс. Водії мають змогу підтвердити виконання замовлення у своєму інтерфейсі, що одразу оновлює статус у системі. Така прозорість процесу логістики дозволяє краще контролювати ефективність перевезень та вчасно реагувати на можливі затримки.

Уся взаємодія між модулями системи реалізована через чітко визначені інтерфейси — наприклад, після авторизації користувача перевіряється актуальність даних, здійснюється пошук вільного транспорту, викликається оптимізація маршруту та, у разі успіху, збереження замовлення до бази даних. Кожен алгоритм реалізовано у вигляді окремого методу в логіці серверної частини, що забезпечує зручність у тестуванні та масштабуванні. Такий підхід дозволяє легко оновлювати або вдосконалювати логіку без втрати стабільності роботи всієї системи. Побудовані діаграми діяльності й послідовності чітко демонструють, як саме відбувається взаємодія між компонентами LogiRoute, і слугують наочною ілюстрацією її внутрішньої архітектури.

Узгоджена взаємодія всіх модулів забезпечує стабільність і передбачуваність роботи системи в різних сценаріях. Архітектура дозволяє швидко адаптувати рішення під нові вимоги або змінити окремі компоненти без порушення загальної логіки. Такий підхід є основою для подальшого розвитку системи та її інтеграції в реальні бізнес-процеси.

Діаграма послідовності системи зображена на рисунку 2.3, діаграма активності — на рисунку 2.4.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

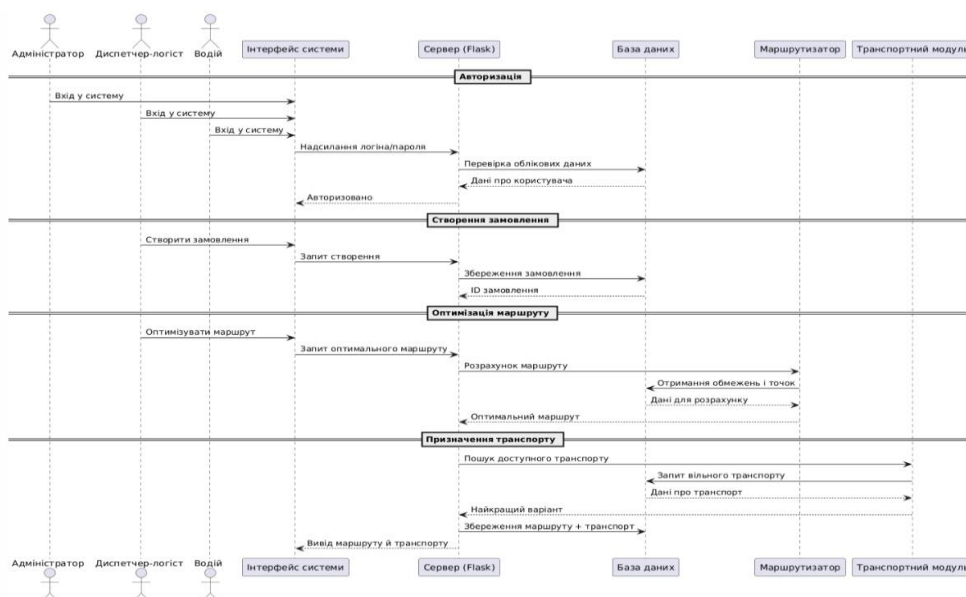


Рис. 2.3 – Діаграма послідовності системи



Рис. 2.4 – Діаграма активності системи

2.4. Реалізація функціоналу системи оптимізації логістичних маршрутів

У процесі розробки системи оптимізації логістичних маршрутів було реалізовано ключовий функціонал, який охоплює всі основні етапи логістичного процесу: від створення замовлення до моніторингу його виконання. Система побудована на архітектурі «клієнт–сервер» з використанням бази даних та зовнішніх API (зокрема, Google Maps API) для маршрутизації, а також передбачає інтеграцію алгоритмів оптимізації на серверній частині.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

Після авторизації користувач має змогу створити нове замовлення на перевезення. У формі створення замовлення користувач заповнює ключові поля: пункт відправлення, пункт призначення, опис вантажу, його вагу, об'єм, контактну інформацію. Система автоматично перевіряє валідність введених даних, наявність вільного транспорту та водіїв, а також доступність доставки в межах заданих параметрів. Після цього дані зберігаються в базу даних, і замовлення переходить до обробки.

Далі активується алгоритм оптимізації маршруту, реалізований на основі модифікованої транспортної задачі з урахуванням реального трафіку, відстані та часу доставки. В результаті система генерує оптимальний маршрут і призначає водія та транспортний засіб, виходячи з їхньої доступності, графіку роботи та технічних характеристик (вантажопідйомність, об'єм тощо).

У кабінеті диспетчера відображається повна інформація про всі замовлення та маршрути. Адміністратор або диспетчер може змінювати статус маршруту (наприклад: “очікує виконання”, “в дорозі”, “виконано”), а також редагувати чи скасовувати маршрути. Водій зі свого акаунта бачить лише призначені йому маршрути з можливістю перегляду на мапі та підтвердження виконання доставки.

Кожен маршрут візуалізується на інтерактивній мапі з нанесенням усіх точок маршруту, відстані, приблизного часу доставки, вартості та маркерів подій на дорогах (дорожні роботи, аварії тощо). Усі дані зберігаються у базі та доступні в історії для перегляду та аналізу.

Для кожної ролі передбачений свій інтерфейс та набір функцій. Адміністратор має доступ до всіх даних та може керувати користувачами, транспортом і замовленнями. Диспетчер відповідає за логістику маршрутів і контроль виконання. Водій бачить лише свої маршрути, без можливості змінювати дані інших користувачів. Така рольова модель забезпечує безпеку, зручність і відповідальність кожної сторони.

Збереження даних здійснюється в реляційній базі MySQL, де кожне

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

замовлення пов’язане з вантажем, маршрутом, водієм і транспортом. Оптимізаційний алгоритм працює на backend-сервері через бібліотеку PuLP на Python, що дозволяє автоматично знаходити найекономічніші варіанти розподілу ресурсів.

Система також реалізує функціонал аналітики, що дозволяє переглядати загальну статистику по маршрутах, кількість виконаних перевезень, середню вартість доставки, середній час виконання, кількість затримок тощо. Це дає змогу компанії оцінити ефективність логістики та приймати обґрунтовані рішення щодо її покращення.

Таким чином, реалізований функціонал повністю охоплює основні логістичні процеси — від створення і оптимізації маршруту до його виконання та аналізу, що робить систему ефективним інструментом для сучасних вантажних перевезень.

Узагальнюючи, було побудовано діаграму компонентів системи оптимізації логістичних маршрутів (рис. 2.5), яка демонструє основні функціональні частини платформи та взаємозв’язки між ними. Вона ілюструє, як організована взаємодія між модулями оформлення замовлень, обробки маршрутів, управління транспортними засобами, користувачами та їх ролями, а також як система взаємодіє з зовнішніми сервісами — такими як картографічні сервіси, API дорожньої ситуації та блок обчислення оптимальних маршрутів.

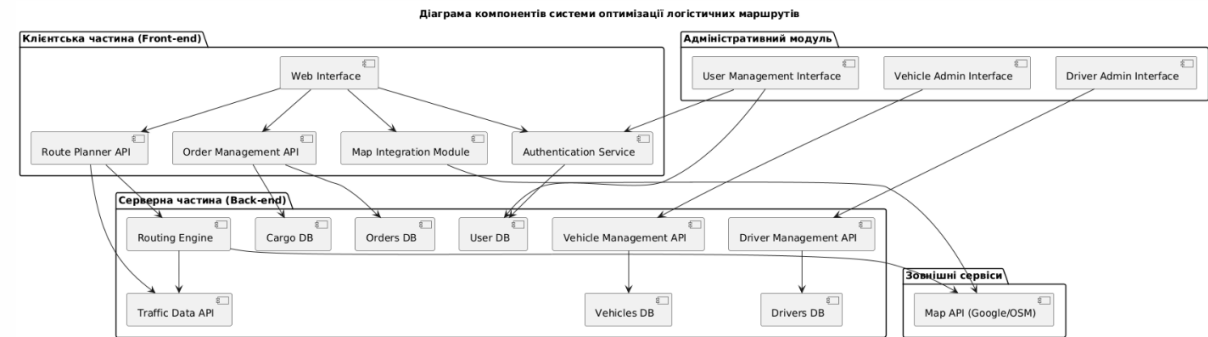


Рис. 2.5 – Діаграма компонентів системи

Опис основних компонентів системи:

User Interface (Інтерфейс користувача) – компонент, який забезпечує взаємодію користувачів із системою через вебінтерфейс. Тут відображаються маршрути, замовлення, статуси доставки, форма входу/реєстрації, доступ до функцій відповідно до ролі (водій, диспетчер, адміністратор).

Authentication Module (Модуль автентифікації) – компонент, який відповідає за реєстрацію, вхід та перевірку прав доступу користувачів. Працює з JWT-токенами для захисту API та контролю ролей.

Routing Optimizer (Оптимізатор маршрутів) – центральний логічний компонент, який виконує побудову оптимального маршруту на основі вхідних даних (точки маршруту, вага вантажу, обмеження, трафік). Реалізує транспортну задачу з використанням бібліотек оптимізації (наприклад, PuLP).

Map & Traffic Service (Сервіс карти та трафіку) – компонент, що взаємодіє з Google Maps API для побудови маршруту, візуалізації його на карті, обчислення часу та відстані, а також для отримання інформації про трафік і події на дорогах.

Database (База даних) – компонент зберігання всіх сутностей системи: користувачі, маршрути, замовлення, вантажі, транспорт, статуси. Реалізовано у вигляді реляційної СУБД.

Route Management (Керування маршрутами) – компонент, що обробляє створення, збереження, редагування та призначення маршрутів. Працює з замовленнями, водіями, транспортом, об'єднує дані з бази та оптимізатора.

Monitoring & Tracking Module (Модуль моніторингу) – відповідає за відстеження виконання маршруту в реальному часі. Показує диспетчерам і водіям актуальне місцезнаходження, прогнозований час прибуття та контрольні точки маршруту.

Admin Dashboard (Адміністративна панель) – компонент, який забезпечує доступ адміністратору до управління користувачами, транспортом, замовленнями, а також перегляду аналітики та статусів усіх маршрутів.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

Висновки до розділу 2

У другому розділі здійснено проектування архітектури інформаційної системи оптимізації логістичних маршрутів. На основі аналізу вимог визначено основні компоненти системи, ролі користувачів (адміністратор, диспетчер, водій), а також побудовано діаграми варіантів використання, класів, активності, послідовності та компонентів, що дозволило сформувати цілісну модель функціонування. Спроектовано реляційну базу даних із чітко визначеними сутностями: користувачі, маршрути, замовлення, транспортні засоби, водії та вантажі.

Реалізовано ключові алгоритми, зокрема побудову оптимального маршруту з урахуванням дорожнього трафіку, транспортних обмежень і характеристик вантажу. Також створено механізми перевірки даних, автоматичного призначення транспорту та водія, оновлення статусів і збору аналітики. Завдяки модульному підходу логіку алгоритмів реалізовано як окремі компоненти, що спрощує підтримку та розвиток системи.

Система взаємодіє з Google Maps API для побудови маршрутів і врахування дорожньої ситуації в реальному часі. Проведене тестування підтвердило стабільну роботу алгоритмів і відповідність функціональним вимогам. Таким чином, другий розділ заклав технічну основу для створення гнучкої, масштабованої та функціонально повноцінної системи управління логістичними маршрутами.

Таким чином, другий розділ заклав основу для створення гнучкої та ефективної системи управління логістичними маршрутами. Запропоновані архітектурні рішення, алгоритми та структура даних забезпечують узгоджену роботу всіх компонентів, підтримку ролей користувачів та автоматизацію ключових процесів. Система готова до впровадження в реальних умовах і може адаптуватися до змін у логістичних вимогах підприємства.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З РЕАЛІЗОВАНОЮ СИСТЕМОЮ

3.1. Порядок встановлення та налаштування параметрів логістичної системи

Розгортання системи оптимізації логістичних маршрутів передбачає встановлення серверної частини, клієнтського інтерфейсу та налаштування зв'язку з базою даних і зовнішніми сервісами, зокрема Google Maps API.

Система реалізована як веб-застосунок, що складається з трьох основних компонентів: клієнтської частини (frontend), яка забезпечує взаємодію користувача з системою через вебінтерфейс; серверної частини (backend), яка відповідає за бізнес-логіку, обробку запитів та взаємодію з базою даних; а також бази даних MySQL, у якій зберігається інформація про користувачів, маршрути, транспортні засоби, замовлення та інші пов'язані дані.

Для коректної роботи системи необхідно виконати кілька послідовних кроків: налаштувати серверну частину, розгорнувши Flask-додаток, встановивши всі залежності через `pip install -r requirements.txt` і вказавши змінні середовища (ключ безпеки, URI бази даних, Google API ключ); підключити MySQL-сервер, створивши базу даних і таблиці згідно з описаною структурою (Users, Routes, Orders тощо); розгорнути клієнтську частину, реалізовану на HTML, CSS (Bootstrap) і JavaScript, з використанням шаблону layout.html та інтеграцією AJAX-запитів до backend; підключити Google Maps API для побудови маршрутів, пошуку адрес та відображення трафіку, вказавши API-ключ у відповідному JavaScript-файлі (app.js) і активувавши сервіси Directions API, Places API та Traffic Layer; налаштувати систему авторизації на основі JWT, з присвоєнням кожному користувачу відповідної ролі (адміністратор, диспетчер, водій), яка визначає доступ до функціоналу.

На рисунку 3.1 наведено діаграму розгортання системи, яка ілюструє взаємодію основних її компонентів.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

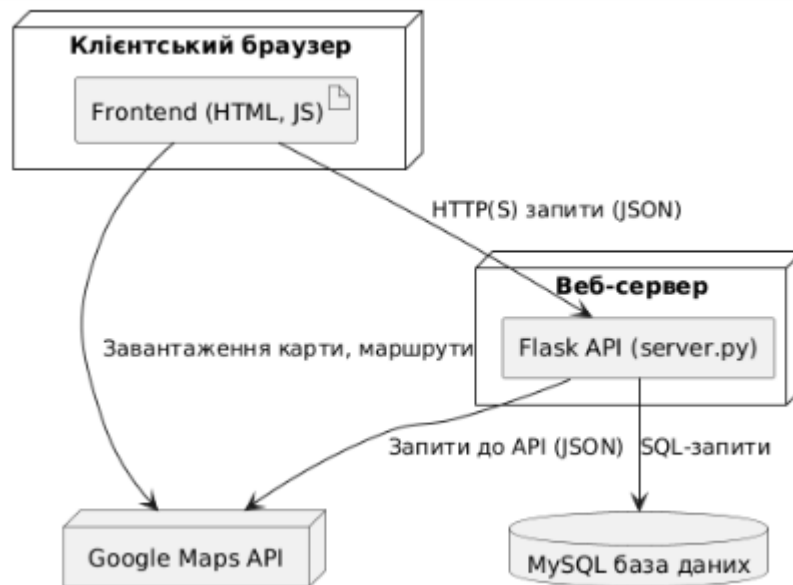


Рис. 3.1 – Діаграма розгортання системи

На діаграмі розгортання (рис. 3.1) зображено основні компоненти логістичної системи та способи їх взаємодії. Система реалізована за клієнт-серверною архітектурою, де користувачі взаємодіють із клієнтською частиною через веббраузер, а всі запити обробляються сервером застосунку.

Frontend представлений у вигляді вебінтерфейсу, що розгорнутий на вебсервері (наприклад, через Flask або окремо через Nginx). Він відповідає за візуалізацію інтерфейсу, відображення мапи, таблиць, форм введення, а також за взаємодію з користувачем відповідно до його ролі.

Backend реалізовано у вигляді REST API на основі Python (Flask). Він приймає запити від клієнта, виконує бізнес-логіку, взаємодіє з базою даних і зовнішніми сервісами. Відповіді формуються у форматі JSON та передаються на frontend.

База даних (MySQL) зберігає структуровану інформацію про користувачів, маршрути, транспорт, замовлення, ролі та логістичні операції. Сервер додатку взаємодіє з нею за допомогою ORM або SQL-запитів.

Google Maps API — зовнішній сервіс, який використовується для побудови маршрутів, автозаповнення адрес, відображення трафіку,

розрахунку відстані та часу. Клієнтська частина на пряму взаємодіє з Google Maps через JavaScript-бібліотеки, а також може надсилати запити через backend.

Користувачі системи (адміністратори, диспетчери, водії) отримують доступ до функцій відповідно до своєї ролі. Авторизація реалізована за допомогою JWT, який зберігається на клієнті після входу в систему.

Такий підхід дозволяє легко масштабувати систему, оновлювати окремі її компоненти та інтегрувати нові функції без зміни загальної архітектури.

3.2. Опис інтерфейсу вебзастосунку LogiRoute

Розроблена система оптимізації логістичних маршрутів призначена для автоматизації процесів, пов'язаних із плануванням вантажних перевезень. Вона дозволяє адміністраторам, диспетчерам та водіям ефективно взаємодіяти між собою, а також здійснювати управління маршрутами, транспортними засобами, замовленнями і доставками. Основною метою системи є зменшення витрат на транспортування, підвищення точності та швидкості обробки замовлень, а також оптимізація використання транспортних ресурсів.

Користувачі системи мають змогу проходити реєстрацію та авторизацію, створювати замовлення на перевезення, отримувати та призначати маршрути, контролювати статус доставки та аналізувати виконання по кожному маршруту. Ролі користувачів — адміністратор, диспетчер і водій — мають різний рівень доступу, що дозволяє гнучко розмежовувати функціональні можливості та зберігати безпеку роботи з даними.

Сторінка реєстрації є першим кроком користувача до повноцінної роботи з системою. Вона відкривається автоматично або за посиланням із головної сторінки, якщо користувач ще не має облікового запису. Інтерфейс сторінки виконано у мінімалістичному стилі: центрально розташована форма містить три поля — ім'я користувача, електронну пошту та пароль. Кожне з полів має валідацію: наприклад, електронна пошта має відповідати загальному

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

формату адреси, а пароль — бути не коротшим за певну мінімальну кількість символів. Введені дані перевіряються ще до надсилання — користувач бачить підказки, якщо щось введено некоректно.

Після натискання кнопки «Зареєструватися» спрацьовує JavaScript-функція, яка формує POST-запит до серверного API. На сервері перевіряється, чи ще не зареєстрований обліковий запис з такою електронною адресою. Якщо ні — створюється новий запис у базі даних: ім'я користувача, email та хешований пароль. Користувач автоматично отримує роль за замовчуванням (наприклад, водій або диспетчер), якщо це не змінюється адміністратором. У випадку успішної реєстрації користувача може бути перенаправлено на сторінку входу або одразу на головну сторінку, де він уже авторизований. Якщо ж email уже існує в базі, форма показує відповідне повідомлення про помилку, без перезавантаження сторінки.

Завдяки клієнтській валідації, асинхронній перевірці даних та чіткому зворотному зв'язку користувач швидко розуміє, що необхідно виправити або як завершити реєстрацію. Паролі не зберігаються у відкритому вигляді, а обмін даними відбувається по захищеному каналу. Сторінка є базовою, але дуже важливою частиною всієї системи, бо з неї починається взаємодія з усіма іншими можливостями.

Сторінка реєстрації користувача зображена на рисунку 3.2.

Рис. 3.2 – Сторінка реєстрації користувача

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

Сторінка входу в систему логістики вантажних перевезень дозволяє авторизованим користувачам отримати доступ до функціоналу відповідно до своєї ролі — адміністратора, диспетчера або водія. Вона є стартовою точкою роботи в системі, і без проходження авторизації жодна з основних сторінок не доступна.

Інтерфейс сторінки містить два основні поля для введення електронної пошти та пароля, а також кнопку для входу. У разі неправильного введення даних (наприклад, помилки в логіні або паролі) користувачу одразу виводиться повідомлення про невірні облікові дані без перезавантаження сторінки, що реалізовано за допомогою асинхронного запиту до сервера.

Система не реалізує функціонал додаткового захисту від багаторазових неправильних спроб входу, таких як блокування акаунта, CAPTCHA або обмеження кількості спроб — тобто при бажанні користувач може багаторазово вводити неправильні дані. Водночас, основна мета авторизації — надати доступ лише тим користувачам, які вже зареєстровані в системі й мають відповідні облікові записи.

Після успішного входу система перевіряє роль користувача і автоматично перенаправляє його на відповідну сторінку: адміністратора — до панелі керування, диспетчера — до диспетчерського інтерфейсу, а водія — до розділу з призначеними маршрутами.

Сторінка входу в систему зображена на рисунку 3.3.

Регістрація'."/>

Рис. 3.3 — Сторінка входу в систему

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Головна сторінка системи є точкою входу до платформи LogiRoute для всіх користувачів. Її основна мета — надати короткий огляд можливостей системи та дозволити швидко перейти до потрібного розділу. Сторінка відкривається автоматично, якщо користувач уже авторизований, або вручну — за прямим посиланням.

Інтерфейс головної сторінки виконано у сучасному стилі з акцентом на простоту і зручність навігації. У верхній частині розташовано фіксований хедер, який містить логотип системи зліва, а також навігаційне меню — посилання на основні розділи: «Маршрути», «Аналітика», «Замовлення», «Транспорт», «Історія» та «Профіль». Залежно від ролі користувача — водій, диспетчер чи адміністратор — частина пунктів у меню може бути прихована або недоступна. Наприклад, водій не має доступу до сторінок аналітики чи редагування транспорту.

З правого боку хедера розташована іконка користувача, при натисканні на яку з'являється випадаюче меню з пунктами «Налаштування акаунта» та «Вийти з акаунта». Крім того, поруч з аватаркою відображається ім'я користувача, що покращує впізнаваність і персоналізацію інтерфейсу.

Основна частина сторінки (контентна область) у поточній версії слугує як шаблон — з нею майже не взаємодіють напряму. Вона не містить динамічного вмісту, проте закладена як основа для майбутнього наповнення — наприклад, короткий дашборд або новини про оновлення системи. Таким чином, головна сторінка виступає не лише як привітання, але й як центральна навігаційна точка, з якої користувач може швидко потрапити до будь-якого функціонального модуля системи.

Завдяки простому макету, логічному розміщенню елементів і адаптації до ролей, ця сторінка виконує важливу роль у забезпеченні зручної навігації в межах системи.

Головна сторінка системи зображена на рисунку 3.4.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

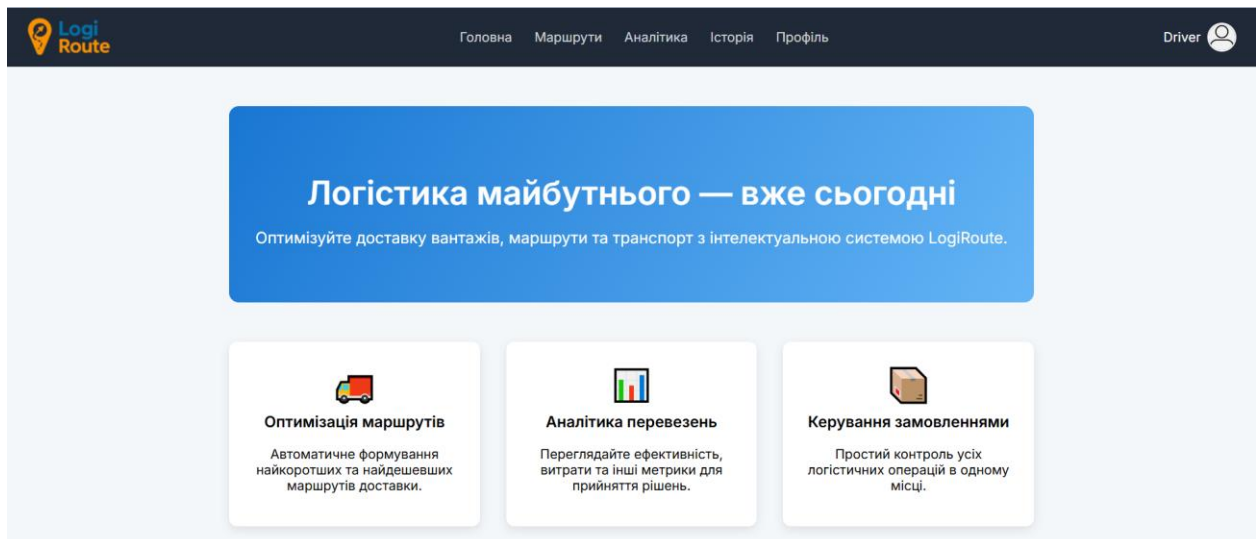


Рис. 3.4 - Головна сторінка системи

Сторінка аналітики доставки призначена для наочного представлення статистичних даних, що стосуються виконаних або запланованих доставок у логістичній системі. Вона доступна користувачам із роллю адміністратора або диспетчера та дозволяє швидко оцінити ефективність роботи транспорту, завантаженість маршрутів і загальні показники перевезень.

Інтерфейс сторінки побудований за сучасними принципами візуалізації: у верхній частині — заголовок сторінки з написом "Аналітика доставки", а нижче — блоки з графіками та короткою статистикою. Дані на сторінці можуть змінюватися динамічно, залежно від реальних замовлень та маршрутів, які обробляються системою.

Основна частина інтерфейсу складається з декількох графіків. Наприклад, один із них може показувати кількість доставок по днях тижня або по місяцях — таким чином, користувач одразу бачить, у які дні виконується більше перевезень. Інший графік — кругова діаграма — може демонструвати відсоткове співвідношення доставок за категоріями (експрес, стандарт, великогабаритна тощо), що дозволяє оцінити розподіл навантаження по типах послуг.

Під кожним графіком або на бічній панелі можуть бути короткі підсумкові цифри: загальна кількість доставок, середній час виконання доставки, кількість водіїв на маршрутах, рівень заповненості транспорту. Всі

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

ці дані беруться із бази даних системи, фільтруються на серверній частині та передаються на клієнтську через API. Для відображення статистики використовується бібліотека Chart.js, яка дозволяє зручно інтегрувати графіки у сторінку та динамічно оновлювати їх при зміні даних.

Логіка роботи сторінки аналітики побудована так, щоб при кожному вході дані були актуальні. Якщо адміністратор або диспетчер змінює статус доставки або додає новий маршрут — статистика автоматично оновлюється або принаймні доступна для повторного завантаження.

Ця сторінка є корисною не лише для аналізу поточних результатів, а й для прийняття рішень — наприклад, у плануванні додаткових рейсів у пікові дні або зміні типу транспорту залежно від обсягів доставки.

Сторінка аналітики доставки зображена на рисунках 3.5-3.6.

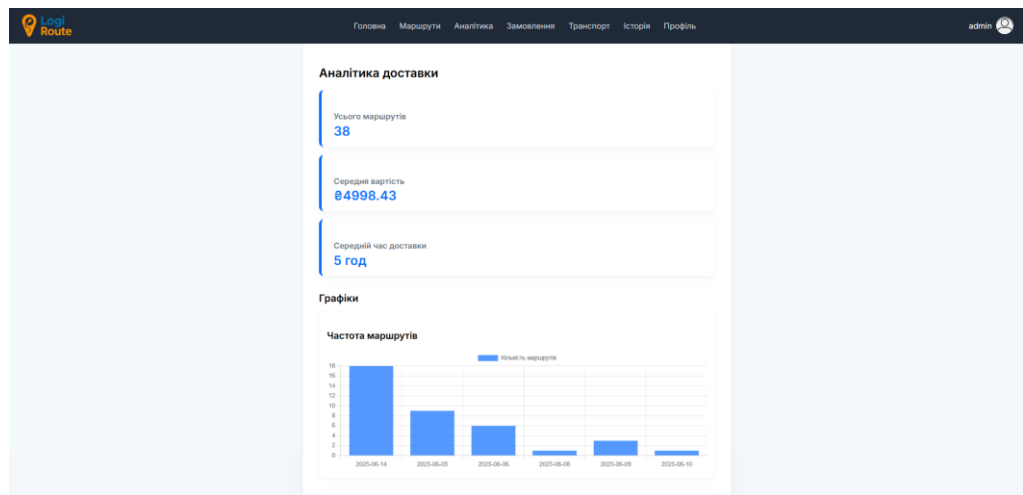


Рис. 3.5 — Сторінка аналітики доставки (А)

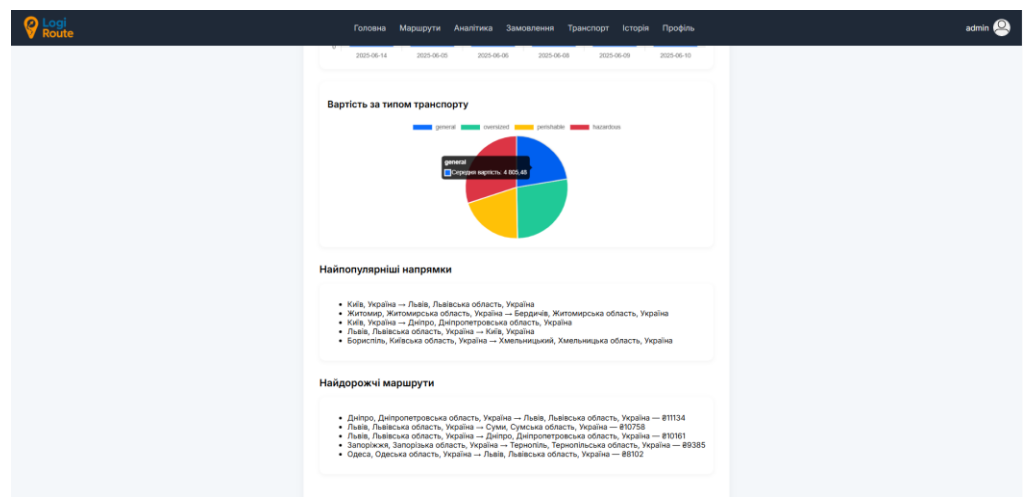


Рис. 3.6 — Сторінка аналітики доставки (В)

Сторінка аналітики перевезень є інструментом для аналізу діяльності транспортного парку системи в розрізі виконаних рейсів, завантаженості транспорту та ефективності використання ресурсів. Вона надає адміністраторам та диспетчерам можливість бачити загальну картину роботи перевізників, виявляти вузькі місця в логістиці та приймати обґрунтовані рішення щодо оптимізації маршрутів і транспорту.

Інтерфейс сторінки оформлений у стилі, подібному до аналітики доставки, що забезпечує візуальну послідовність та зручність користування. У верхній частині сторінки розміщений заголовок з назвою «Аналітика перевезень», а під ним — серія графіків, діаграм і статистичних блоків.

Один із головних графіків — це стовпчикова діаграма, яка показує кількість перевезень у розрізі днів або тижнів. Вона дозволяє виявити пікові навантаження на транспортну систему. Інший графік може представляти дані про кількість перевезених замовлень кожним транспортним засобом, що допомагає зрозуміти, наскільки рівномірно розподіляється робота між автомобілями.

Окремий блок може відображати середню відстань одного маршруту, кількість порожніх пробігів або витрати пального — все це дозволяє аналізувати економічну ефективність перевезень. Важливим елементом сторінки є діаграма типів транспорту, яка показує, які категорії автомобілів використовуються найчастіше (вантажівки, мікроавтобуси тощо) та скільки рейсів було виконано кожною з них.

Дані для графіків та блоків збираються з бази даних системи, де зберігається інформація про маршрути, виконані замовлення, призначення водіїв та використаний транспорт. Через API ці дані надсилаються на клієнтську частину, де за допомогою Chart.js формуються візуалізації.

Користувач не взаємодіє з цією сторінкою активно — вона є в першу чергу аналітичним інструментом. Проте, при наявності ролі диспетчера або адміністратора, можна відкрити додаткові фільтри — наприклад, перегляд

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

даних по окремому транспортному засобу або в межах певного діапазону дат.

Сторінка аналітики перевезень допомагає оцінити ефективність логістичних операцій у динаміці, а також виступає як звітний інструмент для керівництва підприємства.

Сторінка аналітики перевезень зображена на рисунку 3.7.

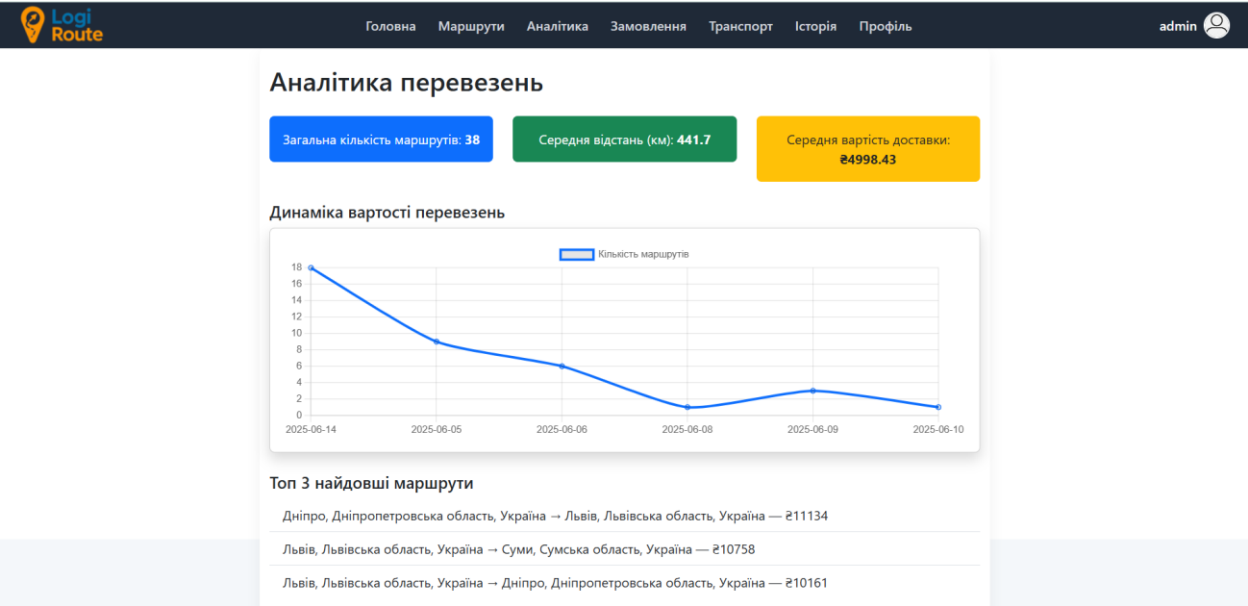


Рис. 3.7 — Сторінка аналітики перевезень

Сторінка «Історія маршрутів» призначена для відображення всіх маршрутів, які були створені, призначені або виконані в системі. Це універсальний інструмент як для водія, так і для адміністратора або диспетчера, однак обсяг інформації, який бачить користувач, залежить від його ролі.

Після входу користувача у систему та переходу на сторінку історії, на екрані з'являється таблиця з усіма маршрутами. Основна структура таблиці включає наступні поля: ID маршруту, адреса відправлення, адреса прибуття, загальна відстань, тривалість маршруту, вартість перевезення, статус маршруту, дата створення. Для адміністратора та диспетчера додатково доступні поля «Призначений водій» і «Транспортний засіб», а також кнопки керування (наприклад, редагування статусу маршруту).

Сторінка містить пошуковий рядок — за допомогою нього користувач може знайти маршрути за адресою, ID, статусом або датою. Окрім цього, є

можливість фільтрувати маршрути за статусом: «Заплановано», «Виконується», «Завершено», «Скасовано».

Для кожного маршруту можна переглянути деталі. Після натискання на рядок або спеціальну кнопку відкривається окреме вікно чи сторінка, де видно карту з відображенням маршруту, точки старту й фінішу, список усіх проміжних зупинок (якщо були), повну інформацію про транспорт та водія, а також — дії, які виконувались під час маршруту (наприклад, зміна статусу або вручення вантажу).

Для адміністратора або диспетчера доступна можливість змінити статус маршруту (наприклад, вручну позначити маршрут як виконаний у випадку збоїв автоматичного оновлення), або видалити помилково створений маршрут.

Для водіїв ця сторінка є довідником — він може переглянути, які маршрути вже були виконані, в яких датах, скільки часу тривала доставка тощо. Це зручно, наприклад, для звітності або для аналізу особистої продуктивності.

Таким чином, сторінка історії маршрутів є ключовим інструментом для моніторингу роботи логістичної системи, забезпечуючи прозорість, простоту доступу до інформації та зручність навігації для користувача будь-якої ролі.

Сторінка з історією маршрутів зображена на рисунках 3.8-3.9, модальне вікно для окремого маршруту — на рисунку 3.10.

Історія маршрутів				
Пошук...				
#	Початкова точка	Кінцева точка	Відстань (км)	Дата
1	Бердичів, Житомирська область, Україна	Житомир, Житомирська область, Україна	41.7	15.06.2025
2	Житомир, Житомирська область, Україна	Бердичів, Житомирська область, Україна	63.5	15.06.2025
3	Коломия, Івано-Франківська область, Україна	Миколаїв, Миколаївська область, Україна	778	15.06.2025
4	Київ, Україна	Дніпро, Дніпропетровська область, Україна	495.8	15.06.2025
5	Івано-Франківськ, Івано-Франківська область, Україна	Запоріжжя, Запорізька область, Україна	1009.3	15.06.2025
6	Мукачєво, Закарпатська область, Україна	Луцьк, Волинська область, Україна	382.4	15.06.2025

Рис. 3.8 — Історія маршрутів (А)

Історія маршрутів

Бердичів

#	Початкова точка	Кінцева точка	Відстань (км)	Дата	
1	Бердичів, Житомирська область, Україна	Житомир, Житомирська область, Україна	41.7	15.06.2025	
2	Житомир, Житомирська область, Україна	Бердичів, Житомирська область, Україна	63.5	15.06.2025	

Рис. 3.9 — Історія маршрутів (В)

Маршрут

Початкова точка

Київ, Україна

Кінцева точка

Дніпро, Дніпропетровська область, Україна

Відстань

495,8

Вартість

7252,87

Статус

Очікує

Рис. 3.10 — Модальне вікно для окремого маршруту

Сторінка призначених маршрутів є однією з найважливіших частин системи для водіїв. Саме тут вони отримують перелік маршрутів, які були призначені їм диспетчером або адміністратором. Ця сторінка забезпечує зручний доступ до інформації про майбутні рейси, дозволяє переглядати деталі маршруту, а також підтверджувати або завершувати виконання завдань (залежно від реалізованого функціоналу).

Після входу до системи з роллю "водій", користувач автоматично або

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

через меню переходить до сторінки з призначеними маршрутами. Інтерфейс сторінки — простий і інтуїтивно зрозумілий. У верхній частині розміщено заголовок «Мої маршрути», а нижче — таблиця або картки із детальною інформацією про кожен маршрут.

Кожен маршрут у системі містить такі дані: унікальний ідентифікатор маршруту, дату та час виїзду, список адрес (пунктів відправлення та доставки), загальну відстань, тривалість у годинах або хвилинах, стан маршруту (наприклад, «Очікує», «Виконується», «Завершено»), а також призначений транспортний засіб.

Крім текстової інформації, іноді передбачене відображення маршруту на карті — у вигляді інтеграції з Google Maps або іншим картографічним сервісом. Це допомагає водію попередньо ознайомитися з напрямком та особливостями дороги.

Якщо передбачено активну взаємодію, то водій може натискати на кнопку «Переглянути» для відкриття детальної сторінки маршруту. Там можуть бути доступні додаткові функції: завантажити маршрутний лист, підтвердити старт виконання або завершення, залишити коментар.

Сторінка також може містити візуальні індикатори або статусні мітки, які дозволяють швидко орієнтуватися у пріоритетності чи терміновості виконання завдань. За необхідності її функціонал може бути розширено для роботи в мобільній версії, що особливо зручно для водіїв під час руху.

Уся інформація на сторінці формується динамічно — для кожного користувача виводяться тільки ті маршрути, які призначені саме йому. Це забезпечується через backend-запит, який фільтрує маршрути за `user_id`, отриманим з токена авторизації.

Ця сторінка суттєво спрощує логістику для водія та робить виконання замовлень чітко контрольованим. Завдяки ній кожен учасник перевезення знає, що і коли йому потрібно робити, що зменшує ризики непорозумінь і підвищує ефективність роботи логістичної системи загалом.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

Сторінка призначених маршрутів зображена на рисунку 3.11.

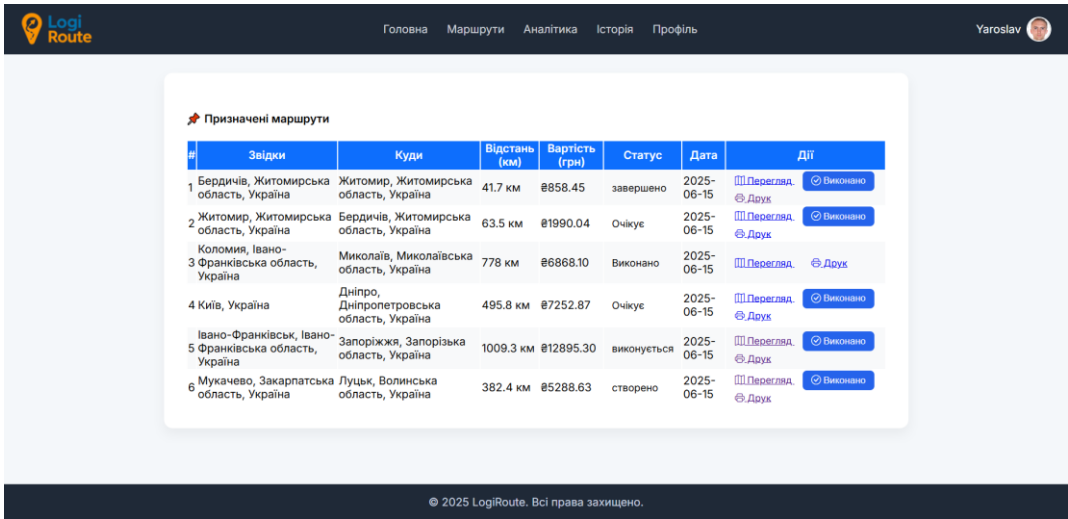


Рис. 3.11 — Сторінка призначених маршрутів

З сторінки призначених маршрутів можна переглянути інформацію про окремий маршрут. Ця сторінка відкривається, коли користувач на сторінці історії маршрутів натискає на певний маршрут або кнопку «Деталі». Основне призначення сторінки — надати повну інформацію про один конкретний маршрут у зручному та візуально зрозумілому вигляді.

У верхній частині сторінки зазвичай відображається заголовок із номером маршруту або його ID. Нижче розміщується інтерактивна карта (Google Maps або Leaflet), на якій видно маршрут — точка старту, кінцева точка та, якщо є, проміжні пункти. Кожна точка позначена маркером, а сам маршрут з'єднаний лінією. Над картою або збоку може бути панель з кнопками — наприклад, для оновлення маршруту або повернення назад.

Під картою знаходиться розширена інформація про маршрут: адреса відправлення, адреса призначення, розрахована відстань, приблизна тривалість у годинах і хвилинах, обрана дата виконання, статус маршруту («Очікує», «Призначено», «Виконується», «Завершено» тощо).

Також подається інформація про водія, який призначений на цей маршрут: ім'я, прізвище, контактний номер або ID. Якщо маршрут уже виконувався або завершений — виводяться також дата початку та завершення, коментарі або позначки, які міг залишити водій.

Для адміністратора і диспетчера доступне додаткове поле зі статусом, який можна змінити прямо на сторінці (наприклад, із «Очікує» на «Виконується»). Для водія поля залишаються лише в режимі перегляду без можливості редагування. Таким чином забезпечується контроль доступу та прозорість взаємодії.

Ця сторінка є важливою для оперативного аналізу конкретного маршруту — її використовують як для перевірки стану доставки, так і для внесення змін (якщо це дозволено), аналізу помилок чи затримок, а також — як підтвердження для звітності чи внутрішнього моніторингу.

Сторінка перегляду окремого маршруту зображена на рисунку 3.12.

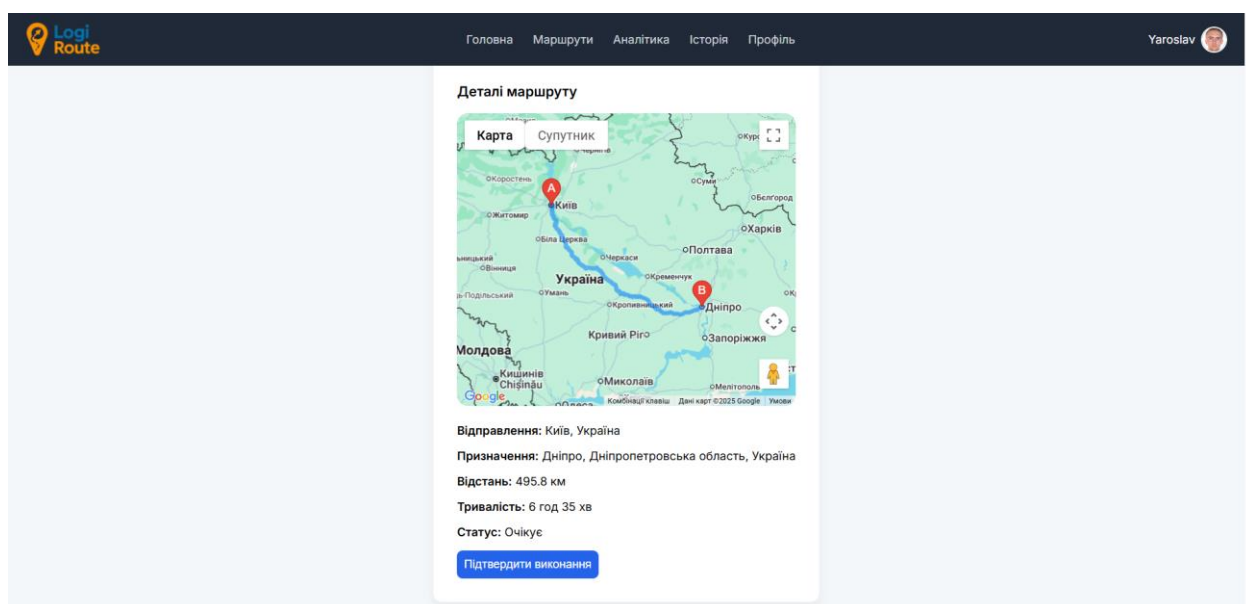


Рис. 3.12 — Сторінка перегляду окремого маршруту

Сторінка формування маршрутного листа — це інструмент, який дає змогу диспетчеру або адміністратору зібрати всю необхідну інформацію про заплановану доставку в структурованому форматі. Цей маршрутний лист може використовуватись водієм під час виконання рейсу або як звітний документ про логістичну операцію.

Після переходу на цю сторінку (наприклад, зі списку маршрутів або після натискання кнопки «Створити лист») відкривається форма, в якій уже відображаються попередньо заповнені диспетчером дані. У верхній частині

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

відображається заголовок — «Маршрутний лист» — та кнопка для збереження або друку листа.

Основна форма поділена на логічні блоки: інформація про маршрут, де вказується унікальний номер маршруту, дата його формування, ім'я користувача, що його створює; точка відправлення та доставки — зазначаються точні адреси (взято з основного маршруту) із вказанням часу виїзду, орієнтовного часу прибуття, типу вантажу тощо; транспортний засіб та водій — вказується тип транспортного засобу, його номерний знак, ім'я водія, номер телефону для зв'язку; інформація про вантаж — маса, об'єм, кількість одиниць, спосіб упаковки, специфіка (наприклад, небезпечний, крихкий, охолоджуваний); підпис відповідальних осіб — місце для підписів диспетчера, водія та отримувача, що підтверджує доставку.

Під час заповнення поля підтягуються автоматично, якщо маршрут вже існує. Усі введені дані можна відредагувати до моменту збереження. Користувач також має змогу згенерувати PDF-версію маршрутного листа для друку або завантаження — це реалізовано через інтеграцію з відповідною бібліотекою на фронтенді.

Маршрутний лист після формування зберігається в базі даних або може бути прикріплений до конкретного маршруту для подальшого використання. У разі потреби маршрутний лист може бути оновлений або повторно сформований після зміни маршруту чи замовлення. Крім того, система підтримує контроль версій, що дозволяє відстежувати зміни у документі. Наявність електронного маршрутного листа зменшує потребу в паперовому документообігу та мінімізує ризики втрати інформації. Це особливо важливо при масштабуванні логістичних процесів і роботі з великою кількістю перевезень.

Ця сторінка покликана зробити роботу з маршрутами більш формалізованою та документованою. Вона дозволяє не лише спростити комунікацію між диспетчером і водієм, а й забезпечує юридичну або

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

організаційну фіксацію кожного логістичного завдання.

Сторінка формування маршрутного листа зображена на рисунку 3.13.

Маршрутний лист
15.06.2025, 17:42:11

Дані водія

Ім'я	Yaroslav
Автомобіль	Mercedes-Benz Atego
Номерний знак	AM9373BC

Деталі маршруту

Початкова точка	Житомир, Житомирська область, Україна
Кінцева точка	Бердичів, Житомирська область, Україна
Відстань	63.5 км
Тривалість	1 год 18 хв
Статус	виконується

Інформація про вантаж

Тип вантажу	Вантажівка
Категорія	Загальні
Вага	26900 кг
Об'єм	50 м³
Дата відправлення	16 червня 2025 р. о 07:00

Підпис водія

Рис. 3.13 — Сторінка формування маршрутного листа

Панель диспетчера є однією з центральних сторінок для керування логістичними процесами в системі. Саме тут диспетчер виконує свої ключові функції: переглядає всі маршрути, створює нові завдання, призначає водіїв та транспорт, а також відстежує статус виконання кожного маршруту.

Інтерфейс сторінки побудований таким чином, щоб надати диспетчеру повну картину логістичної ситуації. У верхній частині зазвичай розміщений заголовок сторінки — «Панель диспетчера», а нижче — функціональні блоки та таблиці. У більшості випадків виводиться список усіх наявних маршрутів у вигляді таблиці з наступними стовпцями: ID маршруту, пункти відправлення й прибуття, призначений водій, призначений транспорт, статус маршруту, дата та час.

Поруч або над таблицею розташовані кнопки для створення нового маршруту, фільтрації або пошуку. Наприклад, диспетчер може швидко відсортувати маршрути за статусом («очікує призначення», «виконується», «завершено») або за конкретним водієм.

Коли диспетчер натискає на кнопку створення маршруту, відкривається форма, у якій необхідно вказати пункти старту і фінішу, дату та час відправлення, обрати призначеного водія зі списку доступних користувачів, вибрати транспортний засіб із зареєстрованого парку, а також за потреби додати додаткові коментарі або інструкції.

Після заповнення цієї форми маршрут зберігається в базі даних, і автоматично стає доступним для водія в його особистому кабінеті.

Також панель диспетчера дає можливість редагувати статуси маршрутів — наприклад, якщо потрібно вручну позначити маршрут як «виконано» або «скасовано». Це особливо корисно у випадку форс-мажорів або змін у плануванні.

Окрім цього, на сторінці може бути доступ до аналітичної інформації: загальна кількість маршрутів, кількість активних водіїв, завантаження транспорту тощо — у вигляді компактних віджетів або графіків.

Таким чином, панель диспетчера є багатофункціональним інструментом для планування, контролю та оперативного управління перевезеннями в системі. Вона дозволяє централізовано контролювати логістику та швидко реагувати на зміни в процесі доставки.

Панель диспетчера зображена на рисунку 3.14.

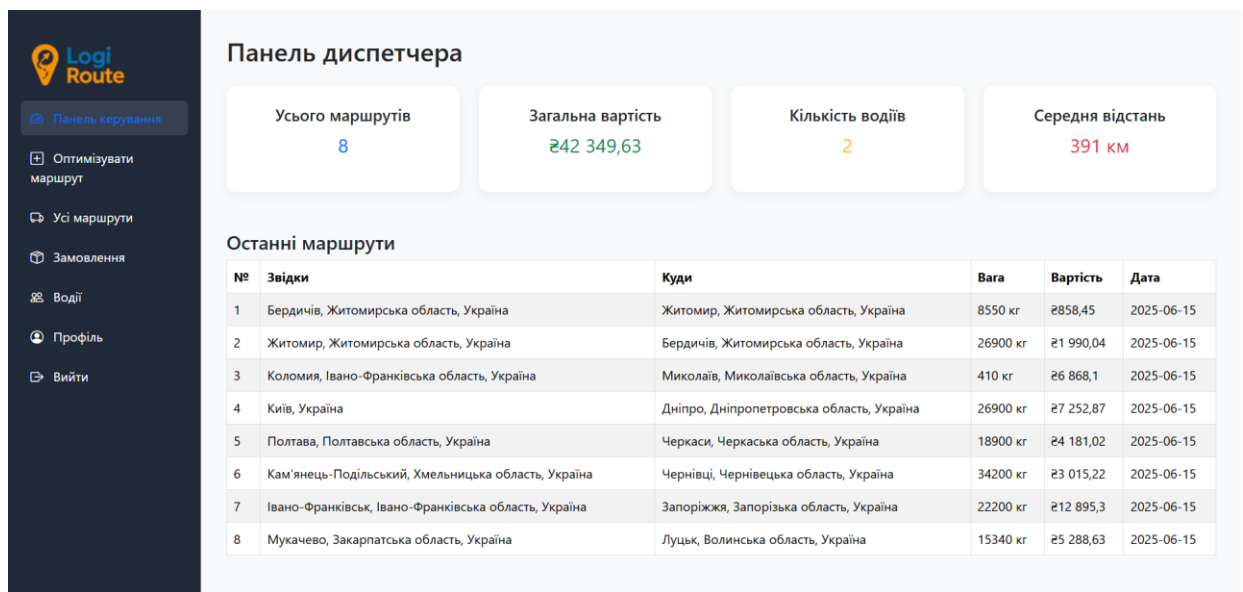


Рис. 3.14 — Панель диспетчера

Ця сторінка є центральною для диспетчера та адміністратора у процесі керування логістикою. Вона відкривається з навігаційного меню або після авторизації, якщо користувач має відповідну роль. Сторінка дозволяє переглядати всі маршрути, які вже були створені, а також керувати ними — створювати нові, редагувати існуючі, змінювати статус або видаляти маршрути.

У верхній частині розміщується заголовок «Список маршрутів» та додаткові елементи інтерфейсу, зокрема кнопка для створення нового маршруту, поле пошуку та фільтри (наприклад, за статусом, датою або водієм). Це дозволяє швидко знайти потрібний маршрут серед великої кількості. Основна частина сторінки — це таблиця, де кожен рядок відповідає окремому маршруту. У таблиці зазвичай подано такі стовпці: ID маршруту, дата створення, адреса відправлення, адреса доставки, призначений водій (ПІБ або ID), статус маршруту, приблизна відстань та час у дорозі, а також кнопки для перегляду, редагування або видалення запису.

При натисканні на кнопку перегляду користувач переходить на сторінку деталей маршруту (описану раніше). Кнопка редагування відкриває форму з попередньо заповненими даними маршруту. Видалення маршруту супроводжується підтвердженням, щоб уникнути випадкових дій.

Крім цього, сторінка динамічно оновлюється після змін — наприклад, після додавання нового маршруту чи редагування старого. Завдяки цьому диспетчер може працювати в реальному часі, не перевантажуючи сторінку вручну.

Цей інтерфейс є критично важливим у системі, адже дозволяє швидко бачити загальну картину перевезень, приймати рішення, делегувати задачі та координувати логістику без зайвих кроків. Простий і зрозумілий візуальний стиль та структурована інформація роблять цю сторінку зручною для щоденного використання в умовах високої динаміки.

Сторінка зі списком маршрутів зображена на рисунку 3.15.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

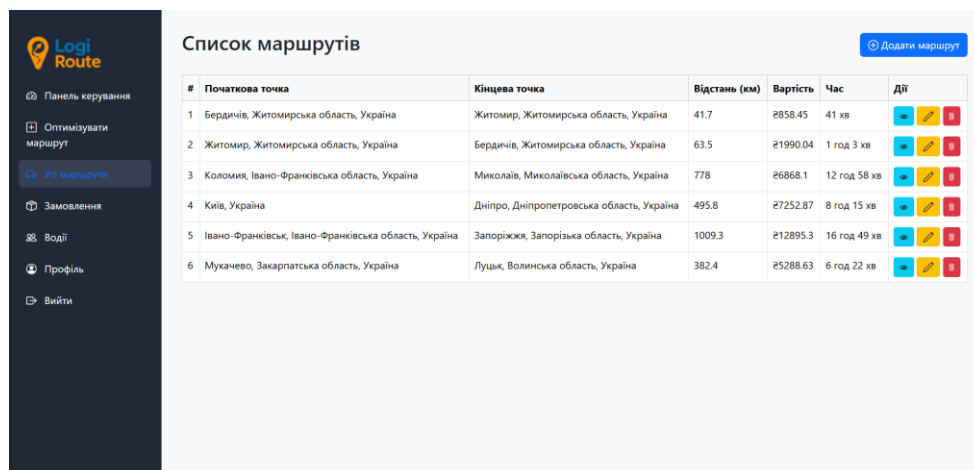


Рис. 3.15 — Сторінка зі списком маршрутів

Сторінка оптимізації маршрутів є однією з найважливіших у системі, оскільки саме тут відбувається основна робота з формуванням маршрутів доставки. Вона доступна авторизованим користувачам з відповідними ролями — переважно диспетчерам або адміністраторам. Інтерфейс сторінки побудований на основі Google Maps і дозволяє вводити точки відправлення та призначення, будувати маршрут між ними, розраховувати його довжину, орієнтовний час у дорозі та вартість перевезення.

На мапі користувач бачить поточну карту України (або іншої території), може вводити адреси вручну або натискати по мапі для додавання точок. Поля для вводу адрес мають автозаповнення завдяки використанню Google Places Autocomplete, що пришвидшує процес і знижує кількість помилок. Після вибору початкової та кінцевої точки відображається маршрут на мапі, а також блок з результатами: довжина маршруту в кілометрах, орієнтовний час у дорозі та розрахована вартість доставки.

Користувач також може натиснути кнопку «Оптимізувати», щоб передати координати на сервер і виконати розрахунок через власний backend-алгоритм, який підтримує як побудову одного маршруту, так і оптимізацію кількох (тобто для транспортної задачі). У разі багатоточкової доставки алгоритм розраховує найефективніший варіант перевезення з урахуванням заданих параметрів (наприклад, кількість пунктів призначення та місткість транспорту).

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Після побудови маршруту є можливість зберегти його в систему. Дані маршруту (точки, довжина, час, вартість) надсилаються на сервер і фіксуються в базі даних. Збережені маршрути потім доступні на сторінці «Історія маршрутів», де з ними можуть працювати інші ролі — наприклад, водій бачить свій призначений маршрут, а диспетчер може змінити його статус.

Крім того, при побудові маршруту враховується ситуація на дорогах: на мапі автоматично активується шар з дорожнім трафіком (через Google Traffic Layer). А після побудови маршруту з'являються також події на маршруті — такі як аварії або ремонти, які підтягуються з API і відображаються маркерами тільки після оптимізації.

Інтерфейс адаптований під різні розміри екранів, що дозволяє працювати зі сторінкою як на комп'ютерах, так і на планшетах. За потреби передбачено можливість редагування вже побудованого маршруту до моменту збереження, а також очистки карти для початку нового планування. Такий підхід значно підвищує гнучкість роботи диспетчера у динамічному середовищі.

Ця сторінка поєднує в собі простоту використання та ефективний алгоритмічний функціонал. Вона дозволяє диспетчерам і адміністраторам оперативно планувати логістику перевезень, бачити реальні умови на дорогах і одразу зберігати маршрути для подальшого використання.

Сторінка оптимізації маршруту зображена на рисунках 3.16-3.17.

The screenshot shows a web application interface for route optimization. At the top, there is a navigation bar with links: Головна, Маршрути, Аналітика, Замовлення, Історія, Профіль, and a user profile icon labeled 'dispatcher'. The main content area is titled 'Оптимізація логістичних маршрутів'. It contains several input fields and buttons: 'Початкові точки: Київ, Україна' with a '+ Додати початкову точку' button; 'Кінцеві точки: Львів, Львівська область,' with a '+ Додати кінцеву точку' button; 'Тип транспорту: Вантажівка' with a dropdown arrow; 'Вага вантажу (кг): 15000'; 'Об'єм вантажу (м³): 30'; 'Тип вантажу: Звичайний' with a dropdown arrow; and 'Дата і час відправлення: 15.06.2025 19:00' with a calendar icon. At the bottom of the form is a large blue button labeled 'Розрахувати маршрут'.

Рис. 3.16 — Сторінка оптимізації маршруту (А)

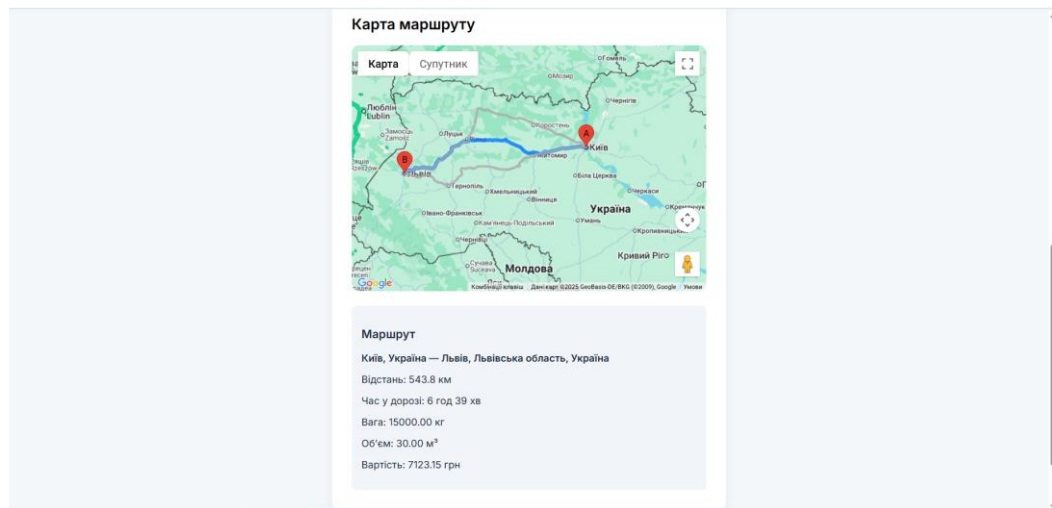


Рис. 3.17 — Сторінка оптимізації маршруту (В)

Сторінка налаштувань акаунта призначена для перегляду та редагування персональної інформації користувача. Доступ до неї має кожен зареєстрований користувач незалежно від ролі — адміністратор, диспетчер чи водій. Потрапити на сторінку можна через меню профілю у верхній панелі інтерфейсу: при натисканні на іконку користувача відкривається випадаючий список із пунктами «Налаштування акаунта» та «Вийти з акаунта».

Після переходу на сторінку налаштувань, користувач бачить зручну форму, в якій відображається його поточна інформація: ім'я користувача, email, роль. Усі поля, окрім ролі, можна редагувати. Користувач може оновити своє ім'я або електронну пошту, після чого натиснути кнопку збереження. Валідація даних проводиться як на клієнтській стороні, так і на сервері — система перевіряє коректність email, а також чи не використовується вже така адреса іншим користувачем.

Після успішного оновлення інформації користувачу показується відповідне повідомлення (наприклад, “Зміни збережено успішно”), і дані оновлюються без перезавантаження сторінки. У разі помилки, наприклад, при спробі ввести вже зареєстровану адресу електронної пошти — система одразу повідомить про це.

Пароль на цій сторінці змінити не можна — для цього передбачено окремий функціонал або модальне вікно (яке можна буде реалізувати в

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

майбутньому).

Ця сторінка має простий та інтуїтивно зрозумілий інтерфейс. Основна її логіка полягає в тому, щоб дати користувачу змогу підтримувати актуальність своїх персональних даних у системі, що особливо важливо в умовах багаторольової логістичної структури, де комунікація та призначення маршруту базуються на ідентифікаторах користувача.

Сторінка з налаштуваннями акаунта зображена на рисунку 3.18.

Налаштування акаунта

Ім'я користувача

Yaroslav

Електронна пошта

rk8813728@gmail.com

Фото профілю

Вибрати файл Файл не вибрано

Новий пароль

(не обов'язково)

Зберегти зміни

Рис. 3.18 — Сторінка з налаштуваннями акаунта

Сторінка панелі адміністратора відкривається одразу після входу в систему, якщо користувач має роль admin. Вона є центральною керуючою точкою для всіх адміністративних функцій у системі. У верхній частині інтерфейсу — фіксований заголовок із логотипом LogiRoute, а праворуч — кнопка профілю з іменем користувача, яка відкриває меню з пунктами "Налаштування акаунта" та "Вийти з акаунта". Завдяки цьому адміністратор має швидкий доступ до свого облікового запису незалежно від розділу, в якому він працює.

Зліва розміщена бічна навігаційна панель. Вона містить посилання на всі доступні адміністратору функції: "Головна", "Маршрути", "Користувачі", "Аналітика", "Замовлення", "Транспорт", "Історія", "Профіль". Кожен із пунктів веде на відповідну сторінку, де можна здійснювати перегляд, редагування чи створення відповідних даних. При цьому для активної

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

сторінки навігаційний пункт підсвічується, що дозволяє легко орієнтуватися у системі.

Центральна частина сторінки виконує роль інформаційної панелі. Тут може відображатися короткий огляд системи — наприклад, кількість активних користувачів, маршрутів, замовлень, статус транспорту чи загальна кількість доставок. Дані можуть подаватися у вигляді карток або графіків — залежно від реалізації. Якщо на даний момент не реалізовано аналітику, виводиться просто вітальний блок із підтвердженням ролі адміністратора та закликком обрати дію в меню.

Ця сторінка є головною для адміністратора, тому через неї він може перейти до всіх ключових операцій — керування обліковими записами, транспортом, призначення маршрутів, перегляд замовлень, створення або оновлення записів у системі. Логіка роботи базується на ролях: тільки адміністратор бачить цю сторінку, інші ролі не мають до неї доступу. Будь-які дії, які виконує адміністратор (наприклад, створення нового користувача або редагування статусу маршруту), виконуються через асинхронні запити до API, з подальшим оновленням інтерфейсу без перезавантаження сторінки.

Таким чином, панель адміністратора — це стартова сторінка для всіх функцій керування, яка надає централізований доступ до даних та дозволяє швидко й ефективно адмініструвати систему.

Панель адміністратора зображена на рисунку 3.19.

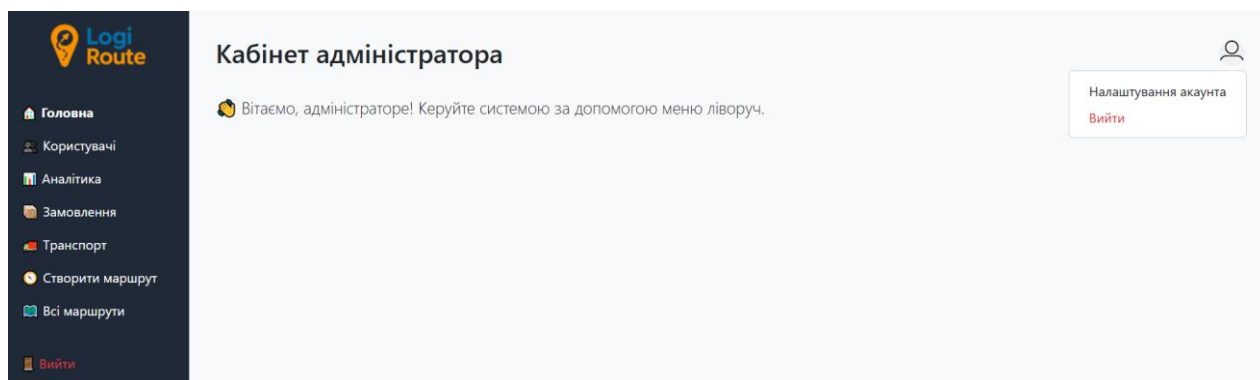


Рис. 3.19 - Панель адміністратора

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінка зі списком користувачів призначена для адміністраторів системи. Саме вони мають повні права на перегляд, редагування та управління обліковими записами інших користувачів. Потрапити на цю сторінку можна через навігаційне меню ліворуч у панелі адміністратора, де пункт «Користувачі» є одним із ключових.

Інтерфейс сторінки представляє собою зручну табличну форму, в якій відображаються всі зареєстровані в системі користувачі. Для кожного користувача в таблиці показано: ім'я, email, роль у системі (адміністратор, диспетчер або водій), а також додаткові керуючі елементи. Якщо користувач — водій, адміністратор може призначити йому транспорт або змінити його роль. У правій частині кожного рядка передбачено кнопки редагування та видалення — проте вони можуть бути обмежені для інших адміністраторів, аби уникнути ситуацій самовидалення.

Також передбачено поле пошуку над таблицею, яке дозволяє швидко знайти потрібного користувача за ім'ям або email. У результаті введення даних таблиця фільтрується у режимі реального часу без перезавантаження сторінки.

При редагуванні користувача відкривається модальне вікно або окрема форма, в якій адміністратор може змінити електронну пошту, ім'я або роль. Усі зміни надсилаються через запит до API, і база оновлюється без необхідності перезавантаження всієї сторінки. У разі успішного оновлення — система повідомляє про це. Якщо виникла помилка (наприклад, дубль email), користувач отримує повідомлення з поясненням.

Сторінка дозволяє підтримувати порядок у системі та забезпечує контроль за обліковими записами. Саме завдяки їй можна відслідковувати активних користувачів, видаляти непотрібних, змінювати ролі або реагувати на проблеми із входом.

Сторінка зі списком користувачів системи зображена на рисунку 3.20.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

Користувачі системи					
#	Ім'я	Роль	Email	Статус	Дії
1	Yaroslav	driver	yaroslav@gmail.com	Активний	Редагувати Видалити
2	dispatcher	dispatcher	dispatcher@gmail.com	Неактивний	Редагувати Видалити
3	admin	admin	admin@gmail.com	Активний	Редагувати Видалити
4	Driver	driver	driver@gmail.com	Неактивний	Редагувати Видалити

Рис. 3.20 — Сторінка зі списком користувачів системи

Сторінка керування транспортом є ключовою для адміністраторів та диспетчерів системи, оскільки дозволяє контролювати наявні транспортні одиниці, додавати нові, редагувати або видаляти існуючі. Цей функціонал особливо важливий для логістичної координації, адже до кожного маршруту повинен бути прив’язаний конкретний транспортний засіб.

На сторінку можна перейти через навігаційне меню зліва, де вона позначена пунктом «Транспорт». Після завантаження сторінки користувач бачить таблицю з переліком усіх зареєстрованих транспортних засобів. У таблиці відображаються такі характеристики: номер транспорту (реєстраційний або внутрішній ідентифікатор), тип транспорту (вантажівка, мікроавтобус тощо), модель, вантажопідйомність, призначений водій (якщо він уже прив’язаний) та статус (вільний, у рейсі, недоступний тощо).

Над таблицею розміщено кнопку «Додати транспорт». Після натискання на неї відкривається форма реєстрації нового транспортного засобу. У цій формі необхідно вказати ключові характеристики транспортного засобу, а також

(за потреби) одразу обрати користувача з роллю «водій», якому буде закріплений транспорт. Якщо обраний водій уже має прив’язаний транспорт, система попередить про це.

Під час редагування запису відкривається окрема форма, в якій можна змінити модель, номер, вантажопідйомність або перевести транспорт у інший статус. Видалення транспорту можливе лише в тому випадку, якщо він не призначений до активного маршруту. Ці обмеження запобігають логічним помилкам у роботі всієї системи.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

Також для диспетчерів ця сторінка дозволяє швидко отримати список доступного транспорту при формуванні нових маршрутів або замовлень.

Сторінка суттєво підвищує ефективність управління логістикою підприємства, надаючи єдину точку контролю за всім транспортом, який використовується у перевезеннях.

Сторінка реєстрації та перегляду транспорту зображена на рисунку 3.21.

Рис. 3.21 — Сторінка реєстрації та перегляду транспорту

Висновки до розділу 3

У цьому розділі розглянуто інтерфейс користувача, порядок роботи системи та процес її встановлення і розгортання. Інтерфейс адаптовано під ролі користувачів (адміністратор, диспетчер, водій), кожна сторінка має чітке функціональне призначення, а взаємодія з елементами системи реалізована інтуїтивно й логічно.

Описано основні етапи роботи — від авторизації та перегляду маршрутів до керування транспортом і замовленнями. Продемонстровано інтеграцію з модулем оптимізації маршрутів, аналітичним блоком та формуванням маршрутних листів. Також наведено порядок встановлення системи на сервер, включно з налаштуванням бази даних, backend та frontend частини.

Загалом підтверджено, що інтерфейс системи є зручним, адаптивним і стабільним. Тестування показало відсутність критичних помилок, а використання сучасних вебтехнологій забезпечує коректну роботу у популярних браузерах.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено веб-орієнтовану інформаційну систему для оптимізації логістичних маршрутів вантажних перевезень. Запропонована система дозволяє автоматизувати ключові процеси планування, призначення та моніторингу маршрутів із урахуванням поточних обмежень і логістичних параметрів.

На основі аналізу предметної області було виявлено проблемні аспекти у плануванні логістичних маршрутів, зокрема — відсутність інтегрованих засобів оптимізації та обмежений контроль за виконанням маршрутів у реальному часі. Розроблена система вирішує ці завдання шляхом впровадження функціоналу динамічного формування маршрутів, ролевої авторизації користувачів, управління транспортними засобами, водіями, замовленнями та інтеграції з картографічними сервісами.

У ході реалізації програмного комплексу були впроваджені такі функції: створення та оптимізація маршрутів на основі даних користувача; обробка замовлень; призначення транспортних засобів; облік історії маршрутів; ведення аналітики перевезень. Кожна функція реалізована у відповідності до поставлених завдань і підтверджена результатами тестування.

Практична цінність розробленої системи полягає у можливості її впровадження в логістичних компаніях та службах доставки з метою підвищення ефективності планування маршрутів, зменшення витрат на перевезення та покращення обслуговування клієнтів. Система має гнучку архітектуру та масштабованість, що дозволяє її адаптувати під потреби конкретного підприємства.

У перспективі подальших досліджень можливим є розширення функціоналу системи шляхом впровадження алгоритмів машинного навчання для прогнозування заторів, а також інтеграції з мобільними додатками для оперативного зв'язку з водіями в реальному часі.

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України «Про автомобільний транспорт» від 05.04.2001 № 2344-III [Електронний ресурс]. – Режим доступу: <https://zakon.rada.gov.ua/laws/show/2344-14>
2. Закон України «Про транспорт» від 10.11.1994 № 232/94-ВР [Електронний ресурс]. – Режим доступу: <https://zakon.rada.gov.ua/laws/show/232/94>
3. Постанова КМУ від 25.12.1996 № 1548 «Про затвердження правил перевезень вантажів автомобільним транспортом в Україні» [Електронний ресурс]. – Режим доступу: <https://zakon.rada.gov.ua/laws/show/1548-96-п>
4. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. – К.: ДП «УкрНДНЦ», 2016. – 16 с.
5. Вовк В.І. Інтелектуальні транспортні системи: навч. посібник. – Львів: Видавництво Львівської політехніки, 2021. – 168 с.
6. Шевченко В.О. Оптимізація логістичних маршрутів в системах перевезень: монографія. – Київ: НТУ, 2020. – 243 с.
7. Білик Л.Я. Логістика: теорія та практика. – Тернопіль: ТНЕУ, 2019. – 312 с.
8. Череп А.В. Логістика. – Київ: Центр учбової літератури, 2018. – 252 с.
9. Степаненко І.В. Основи транспортної логістики: навч. посіб. – Харків: ХНАДУ, 2019. – 208 с.
10. Кірсанов Д.І. Моделювання транспортних систем: навч. посіб. – Краматорськ: ДДМА, 2021. – 197 с.
11. Сайт Державної служби України з безпеки на транспорті [Електронний ресурс]. – Режим доступу: <https://dsbt.gov.ua>
12. Сайт Міністерства інфраструктури України [Електронний ресурс]. – Режим доступу: <https://mtu.gov.ua>
13. Сайт УкрАвтоДор [Електронний ресурс]. – Режим доступу: <https://ukravtodor.gov.ua>

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

14. Коваленко Ю.В. Програмне забезпечення логістичних систем. – Київ: КНЕУ, 2020. – 174 с.
15. Губська Л.С. Електронна логістика: сучасний стан та перспективи розвитку // Економіка і регіон. – 2020. – №4. – С. 92–97.
16. Нікіфорова І.А. Алгоритми оптимізації маршрутів доставки вантажів // Вісник ХНУ імені Каразіна. – 2021. – №35. – С. 45–53.
17. Козак Ю.Г. Стратегії розвитку транспортної логістики України. – Київ: УДУЕУ, 2019. – 236 с.
18. Бібліотека електронних підручників НТУ [Електронний ресурс]. – Режим доступу: <https://lib.ntu.edu.ua>
19. Google Maps Platform documentation [Електронний ресурс]. – Режим доступу: <https://developers.google.com/maps>
20. Державна служба статистики України. Офіційна статистична інформація [Електронний ресурс]. – Режим доступу: <https://www.ukrstat.gov.ua>
21. Flask – офіційна документація [Електронний ресурс]. – Режим доступу: <https://flask.palletsprojects.com>
22. SQLAlchemy documentation [Електронний ресурс]. – Режим доступу: <https://www.sqlalchemy.org>
23. Leaflet documentation [Електронний ресурс]. – Режим доступу: <https://leafletjs.com>
24. Bootstrap 5 documentation [Електронний ресурс]. – Режим доступу: <https://getbootstrap.com>
25. PuLP: Python Linear Programming Modeler [Електронний ресурс]. – Режим доступу: <https://coin-or.github.io/pulp>
26. Звіт про транспортну інфраструктуру України / Мінінфраструктури, 2023 [Електронний ресурс]. – Режим доступу: <https://mtu.gov.ua/news>
27. Statista – Freight transportation data in Ukraine [Електронний ресурс]. – Режим доступу: <https://www.statista.com>
28. Андрієнко В.І. Системи управління перевезеннями: навч. посіб. –

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

Вінниця: ВНТУ, 2021. – 190 с.

29. OpenStreetMap Documentation [Електронний ресурс]. – Режим доступу:

<https://wiki.openstreetmap.org>

30. Ткаченко О.П. Сучасні інформаційні системи в логістиці. – Харків: УПА, 2022. – 151 с.

31. Микитенко В.О. Інформаційні технології в управлінні логістичними процесами: навч. посіб. – Дніпро: ДНУ, 2020. – 204 с.

32. Чорна В.А. Сучасні підходи до моделювання маршрутів у логістиці // Вісник Національного авіаційного університету. – 2021. – №3. – С. 85–91.

33. Міжнародний союз автомобільного транспорту (IRU). Аналітичні огляди логістики [Електронний ресурс]. – Режим доступу: <https://www.iru.org>

34. Стратегії цифрової трансформації в транспорті // Аналітична записка Міністерства цифрової трансформації України, 2022 [Електронний ресурс]. – Режим доступу: <https://thedigital.gov.ua>

35. Курило С.О. Прогнозування логістичних потоків з урахуванням просторово-часових обмежень // Науковий вісник ДонНТУ. – 2022. – №2. – С. 112–118.

36. Європейська логістична асоціація (ELA). Ініціативи щодо сталого розвитку перевезень [Електронний ресурс]. – Режим доступу: <https://www.elalog.eu>

37. Транспортна стратегія України на період до 2030 року: офіційний документ [Електронний ресурс]. – Режим доступу: https://mtu.gov.ua/files/transport_strategy_2030.pdf

38. Савчук Л.І. Моделювання логістичних систем в умовах цифрової трансформації: монографія. – Чернівці: ЧНУ, 2021. – 188 с.

39. Український логістичний портал: новини, аналітика, статті [Електронний ресурс]. – Режим доступу: <https://logistua.com>

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

Додаток 1

Лістинг програмного коду файлу проєкту server.py

```
from flask import Flask, request, jsonify,
render_template, redirect, url_for

from flask_sqlalchemy import SQLAlchemy

from flask_cors import CORS

from werkzeug.security import
generate_password_hash, check_password_hash

from werkzeug.utils import secure_filename

import jwt

import datetime

import os

from dotenv import load_dotenv

load_dotenv()

app = Flask(__name__)

CORS(app)

app.config['SECRET_KEY'] =
os.getenv("SECRET_KEY")

app.config['SQLALCHEMY_DATABASE_URI']

I'] = os.getenv("MYSQL_URI")

app.config['SQLALCHEMY_TRACK_MODIFI
CATIONS'] = False

app.config['UPLOAD_FOLDER'] =
os.path.join('static', 'uploads')

if not
os.path.exists(app.config['UPLOAD_FOLDER']
):

os.makedirs(app.config['UPLOAD_FOLDER'])

db = SQLAlchemy(app)

class User(db.Model):

    id = db.Column(db.Integer,
primary_key=True)

    username = db.Column(db.String(80),
unique=True, nullable=False)

    email = db.Column(db.String(191),
unique=True, nullable=False)

    password_hash = db.Column(db.String(255),
```

		Білотіл Я.О.			ІІЗ.КР.Б. – 121-25 - ІІЗ	Арк.
		Єфремов Ю.М.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

nullable=False)

role = db.Column(db.Enum('admin',
'dispatcher', 'driver'), default='driver')

photo = db.Column(db.String(255), default="")

created_at = db.Column(db.DateTime,
default=datetime.datetime.utcnow)

updated_at = db.Column(db.DateTime,
default=datetime.datetime.utcnow,
onupdate=datetime.datetime.utcnow)

last_login = db.Column(db.DateTime)

class Route(db.Model):

id = db.Column(db.Integer,
primary_key=True)

user_id = db.Column(db.Integer,
db.ForeignKey('user.id'))

vehicle_id = db.Column(db.Integer,
db.ForeignKey('vehicle.id'))

start_points = db.Column(db.Text,
nullable=False)

end_points = db.Column(db.Text,
nullable=False)

cargo_type = db.Column(db.String(50))

weight = db.Column(db.Float)

volume = db.Column(db.Float)

category = db.Column(db.String(50))

departure = db.Column(db.String(50))

distance = db.Column(db.Float)

duration = db.Column(db.String(50))

cost = db.Column(db.Float)

status = db.Column(db.String(50),
default='Очікує')

created_at = db.Column(db.DateTime,
default=datetime.datetime.utcnow)

vehicle = db.relationship('Vehicle',
backref='routes')

user = db.relationship('User', backref='routes')

class Order(db.Model):

id = db.Column(db.Integer,
primary_key=True)

pickup = db.Column(db.String(100),
nullable=False)

delivery = db.Column(db.String(100),
nullable=False)

weight = db.Column(db.Float, nullable=False)

type = db.Column(db.String(50),
nullable=False)

		Білогіл Я.О.			ІІЗ.КР.Б. – 121-25 - ІІЗ	Арк.
		Єфремов Ю.М.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```
deadline = db.Column(db.String(20),
nullable=False)
```

```
status = db.Column(db.String(50),
default="Очікує")
```

```
user_id = db.Column(db.Integer,
db.ForeignKey('user.id'))
```

```
vehicle_id = db.Column(db.Integer,
db.ForeignKey('vehicle.id'))
```

```
user = db.relationship('User', backref='orders')
```

```
vehicle = db.relationship('Vehicle',
backref='orders')
```

```
def to_dict(self):
```

```
    return {
```

```
        "id": self.id,
```

```
        "pickup": self.pickup,
```

```
        "delivery": self.delivery,
```

```
        "weight": self.weight,
```

```
        "type": self.type,
```

```
        "deadline": self.deadline,
```

```
        "status": self.status,
```

```
        "user_id": self.user_id,
```

```
        "vehicle_id": self.vehicle_id
```

```
    }
```

```
class Vehicle(db.Model):
```

```
    id = db.Column(db.Integer,
primary_key=True)
```

```
    user_id = db.Column(db.Integer,
db.ForeignKey('user.id'))
```

```
    name = db.Column(db.String(100),
nullable=False)
```

```
    type = db.Column(db.String(50),
nullable=False)
```

```
    number_plate = db.Column(db.String(20),
unique=True, nullable=False)
```

```
    capacity = db.Column(db.Integer,
nullable=False)
```

```
    status = db.Column(db.String(50), default='В
роботі')
```

```
    user = db.relationship('User',
backref='vehicles')
```

```
    def to_dict(self):
```

```
        return {
```

		Білотіл Я.О.			ІІЗ.КР.Б. – 121-25 - ІІЗ	Арк.
		Єфремов Ю.М.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "id": self.id,
        "user_id": self.user_id,
        "name": self.name,
        "type": self.type,
        "number_plate": self.number_plate,
        "capacity": self.capacity,
        "status": self.status
    }

    @app.route('/')
    def index(): return render_template('login.html')

    @app.route('/registration')
    def registration(): return
    render_template('registration.html')

    @app.route('/admin-profile')
    def admin_profile(): return
    render_template('admin-profile.html')

    @app.route('/analytic-delivery')
    def analytic_delivery(): return

    render_template('analytic_delivery.html')

    @app.route('/analytic-transportation')
    def analytic_transportation(): return
    render_template('analytic_transportation.html')

    @app.route('/assigned-routes')
    def assigned_routes(): return
    render_template('assigned-routes.html')

    @app.route('/dashboard')
    def dashboard(): return
    render_template('dashboard.html')

    @app.route('/history')
    def history(): return
    render_template('history.html')

    @app.route('/local')
    def local(): return render_template('local.html')

    @app.route('/profile')
    def profile(): return
    render_template('profile.html')

```

		Білотіл Я.О.			ІІЗ.КР.Б. – 121-25 - ІІЗ	Арк.
		Єфремов Ю.М.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@app.route('/routes-management')

def routes_management(): return
render_template('routes-management.html')

@app.route('/routes')

def routes(): return
render_template('routes.html')

@app.route('/routing')

def routing(): return
render_template('routing.html')

@app.route('/route-details')

def route_details(): return
render_template('route-details.html')

@app.route('/route-sheet')

def route_sheet(): return render_template('route-
sheet.html')

@app.route('/settings')

def settings(): return
render_template('settings.html')

@app.route('/users')

def users(): return render_template('users.html')

@app.route('/vehicles')

def vehicles(): return
render_template('vehicles.html')

@app.route('/orders')

def orders(): return
render_template('orders.html')

def get_user_from_token():

    auth_header =
request.headers.get('Authorization')

    if not auth_header or not
auth_header.startswith('Bearer '):

        return None

    token = auth_header.split(' ')[1]

    try:

        decoded = jwt.decode(token,
app.config['SECRET_KEY'],
algorithms=["HS256"])

        return User.query.get(decoded['id'])

    except:

```

		Білотіл Я.О.			ІІЗ.КР.Б. – 121-25 - ІІЗ	Арк.
		Єфремов Ю.М.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return None

@app.route('/register', methods=['POST'])

def register():

    data = request.get_json()

    hashed_password =
generate_password_hash(data['password'],
method='pbkdf2:sha256')

    new_user = User(

        username=data['username'],

        email=data['email'],

        password_hash=hashed_password,

        role=data.get('role', 'driver')

    )

    try:

        db.session.add(new_user)

        db.session.commit()

        return jsonify({'message': 'Користувача та
транспорт зареєстровано'}), 201

    except Exception as e:

```

```

        db.session.rollback()

        return jsonify({'error': 'Помилка
реєстрації', 'details': str(e)}), 400

@app.route('/login', methods=['POST'])

def login():

    data = request.get_json()

    user =
User.query.filter_by(email=data['email']).first()

    if not user or not
check_password_hash(user.password_hash,
data['password']):

        return jsonify({'error': 'Невірна пошта або
пароль'}), 401

    user.last_login = datetime.datetime.utcnow()

    db.session.commit()

    token = jwt.encode({

        'id': user.id,

        'role': user.role,

        'exp': datetime.datetime.utcnow() +
datetime.timedelta(hours=1)

    }, app.config['SECRET_KEY'],
algorithm='HS256')

```

		Білотіл Я.О.			ІІЗ.КР.Б. – 121-25 - ІІЗ	Арк.
		Єфремов Ю.М.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        'username': u.username,

        'email': u.email,

        'role': u.role,

        'last_login': u.last_login.isoformat() if
u.last_login else None

    } for u in users])

    'photo': user.photo,

    'user_id': user.id

}))

@app.route('/api/users', methods=['GET'])

def get_users():

    user = get_user_from_token()

    if not user or user.role not in ['admin',
'dispatcher', 'driver']:

        return jsonify({'error': 'Unauthorized'}),
401

    user_data = User.query.get(user_id)

    if not user_data:

        return jsonify({'error': 'Користувача не
знайдено'}), 404

    return jsonify({

        'id': user_data.id,

```

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        'username': user_data.username,

        'email': user_data.email,

        'role': user_data.role

    })

    if User.query.filter((User.username ==
username) | (User.email == email)).first():

        return jsonify({'error': 'Користувач із
таким іменем або поштою вже існує'}), 400

    hashed_password =
generate_password_hash(password,
method='pbkdf2:sha256')

@app.route('/api/users', methods=['POST'])

def create_user():

    user = get_user_from_token()

    if not user or user.role not in ['admin',
'dispatcher']:

        return jsonify({'error': 'Unauthorized'}),
401

    data = request.get_json()

    username = data.get('username')

    email = data.get('email')

    password = data.get('password')

    role = data.get('role', 'driver')

    if not all([username, email, password]):

        return jsonify({'error': 'Всі поля
обов'язкові'}), 400

    new_user = User(

        username=username,

        email=email,

        role=role,

        password_hash=hashed_password

    )

    try:

        db.session.add(new_user)

        db.session.commit()

        return jsonify({'message': 'Користувач
створений успішно'}), 201

    except Exception as e:

        db.session.rollback()

        return jsonify({'error': 'Помилка

```

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

```
збереження', 'details': str(e))), 500
```

```
@app.route('/me', methods=['GET'])
```

```
def get_me():
```

```
    token = request.headers.get('Authorization',
    "").split(' ')[-1]
```

```
    try:
```

```
        decoded = jwt.decode(token,
app.config['SECRET_KEY'],
algorithms=['HS256'])
```

```
        user = User.query.get(decoded['id'])
```

```
        return jsonify({
```

```
            'username': user.username,
```

```
            'email': user.email,
```

```
            'role': user.role,
```

```
            'photo': user.photo
```

```
        })
```

```
    except:
```

```
        return jsonify({'error': 'Недійсний
токен'}), 401
```

```
@app.route('/update-profile', methods=['PUT'])
```

```
def update_profile():
```

```
    token = request.headers.get('Authorization',
    "").split(' ')[-1]
```

```
    try:
```

```
        decoded = jwt.decode(token,
app.config['SECRET_KEY'],
algorithms=['HS256'])
```

```
        user = User.query.get(decoded['id'])
```

```
        username = request.form.get('username')
```

```
        email = request.form.get('email')
```

```
        password = request.form.get('password')
```

```
        file = request.files.get('photo')
```

```
        if username: user.username = username
```

```
        if email: user.email = email
```

```
        if password:
```

```
            user.password_hash =
generate_password_hash(password,
method='pbkdf2:sha256')
```

```
        if file:
```

```
            filename =
secure_filename(file.filename)
```

```
            filepath =
os.path.join(app.config['UPLOAD_FOLDER'],
filename)
```

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ІЗ	Арк.
		Єфремов Ю.М.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

file.save(filepath)

        user.photo = f'/static/uploads/{filename}'

    db.session.commit()

    return 'Профіль оновлено успішно'

except Exception as e:

    return jsonify({'error': str(e)}), 500

@app.route('/api/routes', methods=['GET'])

def get_user_routes():

    user = get_user_from_token()

    if not user:

        return jsonify({'error': 'Unauthorized'}),
401

    if user.role in ['admin', 'dispatcher']:

        routes =
Route.query.order_by(Route.created_at.desc()).a
ll()

    else:

        routes =
Route.query.filter_by(user_id=user.id).order_by
(Route.created_at.desc()).all()

    return jsonify([
        {
            'id': r.id,
            'origin': r.start_points,
            'destination': r.end_points,
            'distance': r.distance,
            'weight': r.weight,
            'cost': r.cost,
            'status': r.status,
            'date': r.created_at.strftime('%Y-%m-%d'),
            'duration': r.duration,
            'user_id': r.user_id,
            'vehicle_id': r.vehicle_id,
            'driver_name': r.user.username if r.user else
None,
            'vehicle_name': f'{r.vehicle.name}
({r.vehicle.number_plate})' if r.vehicle else
None
        } for r in routes])

@app.route('/api/routes/<int:route_id>',
methods=['GET', 'PUT', 'DELETE'])

def manage_route(route_id):

    user = get_user_from_token()

```

		Білотіл Я.О.			ІІЗ.КР.Б. – 121-25 - ІІЗ	Арк.
		Єфремов Ю.М.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if not user: return jsonify({'error':
'Unauthorized'}), 401

```

```

route = Route.query.get_or_404(route_id)

```

```

if request.method == 'GET':

```

```

return jsonify({

```

```

'id': route.id,

```

```

'user_id': route.user_id,

```

```

'vehicle_id': route.vehicle_id,

```

```

'origin': route.start_points,

```

```

'destination': route.end_points,

```

```

'distance': route.distance,

```

```

'cost': route.cost,

```

```

'status': route.status,

```

```

'duration': route.duration,

```

```

'date': route.created_at.strftime('%Y-%m-
%d'),

```

```

'cargo_type': route.cargo_type,

```

```

'category': route.category,

```

```

'weight': route.weight,

```

```

'volume': route.volume,

```

```

'departure': route.departure,

```

```

'driver_name': route.user.username if
route.user else None,

```

```

'vehicle_name': route.vehicle.name if
route.vehicle else None

```

```

})

```

```

elif request.method == 'PUT':

```

```

data = request.get_json()

```

```

route.start_points = data.get('origin',
route.start_points)

```

```

route.end_points = data.get('destination',
route.end_points)

```

```

route.distance = float(data.get('distance',
route.distance))

```

```

route.cost = float(data.get('cost',
route.cost))

```

```

route.status = data.get('status', route.status)

```

```

route.user_id = data.get('user_id',
route.user_id)

```

```

route.vehicle_id = data.get('vehicle_id',
route.vehicle_id)

```

```

db.session.commit()

```

		Білотіл Я.О.			ІІЗ.КР.Б. – 121-25 - ІІЗ	Арк.
		Єфремов Ю.М.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return jsonify({'message': 'Оновлено
успішно'})

    elif request.method == 'DELETE':

        db.session.delete(route)

        db.session.commit()

        return jsonify({'message': 'Видалено'})

@app.route('/save-route', methods=['POST'])

def save_route():

    user = get_user_from_token()

    if not user:

        return jsonify({'error': 'Невірний або
відсутній токен'}), 401

    data = request.get_json()

    try:

        new_route = Route(

            user_id=user.id,

            start_points=',
'.join(data.get('start_points', [])),

            end_points=', '.join(data.get('end_points',
[])),

            cargo_type=data.get('cargo_type'),

            weight=float(data.get('weight', 0)),

```

```

            volume=float(data.get('volume', 0)),

            category=data.get('category'),

            departure=data.get('departure'),

            distance=float(data.get('distance', 0)),

            duration=data.get('duration'),

            cost=float(data.get('cost', 0))

        )

        db.session.add(new_route)

        db.session.commit()

        return jsonify({'message': 'Маршрут
збережено успішно'})

    except Exception as e:

        db.session.rollback()

        return jsonify({'error': 'Не вдалося
зберегти маршрут', 'details': str(e)}), 500

@app.route('/api/orders', methods=['GET'])

def get_orders():

    orders = Order.query.all()

    return jsonify([order.to_dict() for order in
orders])

```

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```
@app.route('/api/orders', methods=['POST'])
```

```
def create_order():
```

```
    user = get_user_from_token()
```

```
    if not user:
```

```
        return jsonify({'error': 'Unauthorized'}),
```

```
401
```

```
    if user.role not in ['admin', 'dispatcher']:
```

```
        return jsonify({'error': 'Лише
```

```
адміністратор або диспетчер можуть
```

```
створювати замовлення'}), 403
```

```
    data = request.get_json()
```

```
    required_fields = ['pickup', 'delivery', 'weight',  
'type', 'deadline', 'user_id', 'vehicle_id']
```

```
    for field in required_fields:
```

```
        if field not in data or data[field] is None:
```

```
            return jsonify({'error': f'Поле '{field}' є  
обов'язковим'}), 400
```

```
    order = Order(  
  
    pickup=data['pickup'],  
  
    delivery=data['delivery'],
```

```
    weight=data['weight'],
```

```
    type=data['type'],
```

```
    deadline=data['deadline'],
```

```
    status=data.get('status', 'Очікує'),
```

```
    user_id=data['user_id'],
```

```
    vehicle_id=data['vehicle_id']
```

```
)
```

```
    db.session.add(order)
```

```
    db.session.commit()
```

```
    return jsonify(order.to_dict()), 201
```

```
@app.route('/api/orders/<int:order_id>',  
methods=['PUT'])
```

```
def update_order(order_id):
```

```
    data = request.get_json()
```

```
    order = Order.query.get_or_404(order_id)
```

```
    order.status = data.get('status', order.status)
```

```
    db.session.commit()
```

```
    return jsonify(order.to_dict())
```

		Білотіл Я.О.			ІІЗ.КР.Б. – 121-25 - ІІЗ	Арк.
		Єфремов Ю.М.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

@app.route('/api/orders/<int:order_id>',
methods=['DELETE'])

def delete_order(order_id):

    order = Order.query.get_or_404(order_id)

    db.session.delete(order)

    db.session.commit()

    return jsonify({'message': 'Видалено'})

```

```

**v.to_dict(),

"username": v.user.username if v.user
else None

}

for v in vehicles

])

```

```

@app.route('/api/vehicles', methods=['GET'])

```

```

def get_vehicles():

    user = get_user_from_token()

    if not user:

        return jsonify({'error': 'Unauthorized'}),
401

```

```

if user.role in ['admin', 'dispatcher']:

    vehicles = Vehicle.query.all()

else:

    vehicles =
Vehicle.query.filter_by(user_id=user.id).all()

```

```

return jsonify([

{

```

```

@app.route('/api/vehicles', methods=['POST'])

```

```

def create_vehicle():

    user = get_user_from_token()

    if not user or user.role not in ['admin',
'dispatcher']:

        return jsonify({'error': 'Доступ дозволено
лише адміністратору або диспетчеру'}), 401

```

```

data = request.get_json()

```

```

vehicle = Vehicle(

    user_id=data['user_id'],

    name=data['name'],

```

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

type=data['type'],

number_plate=data['number_plate'],

capacity=data['capacity'],

status=data.get('status', 'В роботі')

)

try:

    db.session.add(vehicle)

    db.session.commit()

    return jsonify(vehicle.to_dict()), 201

except Exception as e:

    db.session.rollback()

    return jsonify({'error': str(e)}), 500

@app.route('/api/vehicles/<int:vehicle_id>',
methods=['GET'])

def get_vehicle_by_id(vehicle_id):

    user = get_user_from_token()

    if not user:

        return jsonify({'error': 'Unauthorized'}),
401

    vehicle = Vehicle.query.get(vehicle_id)

    if not vehicle:

        return jsonify({'error': 'Транспорт не
знайдено'}), 404

    return jsonify(vehicle.to_dict())

@app.route('/api/vehicles/<int:vehicle_id>',
methods=['DELETE'])

def delete_vehicle(vehicle_id):

    vehicle =
Vehicle.query.get_or_404(vehicle_id)

    db.session.delete(vehicle)

    db.session.commit()

    return jsonify({'message': 'Транспорт
видалено'})

from functools import wraps

def token_required(f):

    @wraps(f)

    def decorated(*args, **kwargs):

        auth_header =
request.headers.get('Authorization')

```

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ІЗ	Арк.
		Єфремов Ю.М.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if not auth_header or not
auth_header.startswith('Bearer '):

        return jsonify({'error': 'Токен не
надано'}), 401

        token = auth_header.split(' ')[1]

        try:

            data = jwt.decode(token,
app.config['SECRET_KEY'],
algorithms=["HS256"])

            current_user = User.query.get(data['id'])

            if not current_user:

                return jsonify({'error': 'Користувача
не знайдено'}), 404

            except Exception as e:

                return jsonify({'error': 'Недійсний
токен', 'details': str(e)}), 401

        return f(current_user, *args, **kwargs)

    return decorated

@app.route('/api/routes/<int:id>',
methods=['PATCH'])

@token_required

def update_route_status(current_user, id):

```

```

        data = request.json

        route = Route.query.get(id)

        if not route:

            return jsonify({'error': 'Маршрут не
знайдено'}), 404

        if route.user_id != current_user.id and
current_user.role == 'driver':

            return jsonify({'error': 'Доступ
заборонено'}), 403

        route.status = data.get('status', route.status)

        db.session.commit()

        return jsonify({'message': 'Статус
оновлено'})

@app.route('/api/analytics/routes')

def analytics_routes():

    user = get_user_from_token()

    if not user:

        return jsonify({'error': 'Unauthorized'}),
401

    if user.role == 'driver':

        routes =

        Route.query.filter_by(user_id=user.id).all()

```

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

else:

routes = Route.query.all()

if not routes:

return jsonify({

"total_routes": 0,

"average_cost": 0,

"average_distance": 0,

"average_time": 0,

"routes_chart": {"labels": [], "data": []},

"cost_chart": {"labels": [], "data": []},

"top_directions": [],

"top_costly": []

})

total_routes = len(routes)

avg_cost = round(sum(r.cost for r in routes if
r.cost is not None) / total_routes, 2)

avg_distance = round(sum(r.distance for r in
routes if r.distance is not None) / total_routes, 1)

durations = []

for r in routes:

if r.duration and 'год' in r.duration:

try:

durations.append(int(r.duration.split()[0]))

except:

continue

avg_time = int(sum(durations) /
len(durations)) if durations else 0

date_counts = {}

for r in routes:

date = r.created_at.strftime("%Y-%m-%d")

date_counts[date] = date_counts.get(date,
0) + 1

type_costs = {}

for r in routes:

if r.category and r.cost:

type_costs.setdefault(r.category,
[]).append(r.cost)

direction_counts = {}

for r in routes:

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        key = (r.start_points, r.end_points)

        direction_counts[key] =
direction_counts.get(key, 0) + 1

        top_directions =
sorted(direction_counts.items(), key=lambda x: -
x[1][:5]

        top_costly = sorted(routes, key=lambda r:
r.cost or 0, reverse=True)[:5]

return jsonify({

    "total_routes": total_routes,

    "average_cost": avg_cost,

    "average_distance": avg_distance,

    "average_time": avg_time,

    "routes_chart": {

        "labels": list(date_counts.keys()),

        "data": list(date_counts.values())

    },

    "cost_chart": {

        "labels": list(type_costs.keys()),

        "data": [round(sum(vals) / len(vals), 2)
for vals in type_costs.values()]

    }

})

@app.route('/sheet-html')

def route_sheet_html():

    route_id = request.args.get('id')

    if not route_id:

        return "ID маршруту не надано", 400

    route = Route.query.get(route_id)

    if not route:

        return "Маршрут не знайдено", 404

    driver = User.query.get(route.user_id)

    return render_template('route-sheet.html',
route=route, user=driver)

with app.app_context():

```

		Білотіл Я.О.			ІІЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```
db.create_all()
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True, port=5000)
```

		Білотіл Я.О.			ІПЗ.КР.Б. – 121-25 - ПЗ	Арк.
		Єфремов Ю.М.				94
Змн.	Арк.	№ докум.	Підпис	Дата		