НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет програмних систем та прикладної математики
Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
«___»_____2026 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Системне програмування та спеціалізовані комп'ютерні системи»
спеціальності 123 «Комп'ютерна інженерія»
на тему: «Інформаційна система оптимізації логістичних маршрутів»

Виконав:
студент 4 курсу, групи КВ-21
Петринка Олександр Ігорович _____

Керівник:
ст. викладач каф. СПіСКС, д.ф.
Радченко Костянтин Олександрович _____

Консультант з нормоконтролю:
доц. каф. СПіСКС, к.т.н., доц.
Клятченко Ярослав Михайлович _____

Рецензент:
ст. викладач каф. обчислювальної техніки ФІОТ, д.ф.
Коренко Дмитро Володимирович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент_____

Київ – 2026 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРОГРАМНИХ СИСТЕМ ТА ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Освітньо-професійною програма

«Системне програмування та спеціалізовані

комп'ютерні системи»

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис)
"___" _____2025 р.

ЗАВДАННЯ
на дипломний проєкт студента
Петринки Олександра Ігоровича

1. Тема проєкту «Інформаційна система оптимізації логістичних маршрутів», керівник проєкту ст. викладач каф. СПіСКС, д.ф. Радченко Костянтин Олександрович, затверджена наказом по університету від «28» травня 2026 р. № 1984-С
2. Термін подання студентом проєкту _____
3. Вихідні дані до проєкту: див. Технічне завдання
4. Зміст пояснювальної записки:
   1) аналіз проблеми ефективного управління логістичними процесами та маршрутизації;
   2) математичне моделювання задач маршрутизації та вибір методів оптимізації;
   3) проектування архітектури інформаційної системи моніторингу та управління логістичними ресурсами;
   4) реалізація системи та аналіз отриманих результатів.
5. Перелік графічного матеріалу:
   1) схема бази даних системи;
   2) алгоритм динамічної переоптимізації маршрутів;
   3) діаграма взаємодії компонентів системи.

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я. М., доцент кафедри СПіСКС, к.т.н., доцент		

7. Дата видачі завдання «__» _____ 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення літератури за тематикою роботи	14.11.2025	
2	Розроблення та узгодження технічного завдання	27.11.2025	
3	Аналіз існуючих програм для аналізу енергоспоживання	29.12.2025	
4	Розроблення структури інформаційної системи	05.02.2026	
5	Розробка серверної частини системи	16.03.2026	
6	Розроблення клієнтських інтерфейсів	20.04.2026	
7	Тестування алгоритмів маршрутизації	21.05.2026	
8	Підготовка матеріалів першого розділу дипломного проєкту	25.05.2026	
9	Підготовка матеріалів другого розділу дипломного проєкту	29.05.2026	
10	Підготовка матеріалів третього розділу дипломного проєкту	01.06.2026	
11	Підготовка матеріалів четвертого розділу дипломного проєкту	02.06.2026	
12	Оформлення технічної документації та графічної частини	03.06.2026	
13	Попередній огляд матеріалів дипломної роботи на кафедрі	04.06.2026	

Студент Олександр ПЕТРИНКА

Керівник проєкту Костянтин РАДЧЕНКО

АНОТАЦІЯ

Дипломний проєкт включає пояснювальну записку (54 с., 13 рис., 1 табл., список використаної літератури з 15 найменувань, 3 додатки).

Об’єкт розробки – Інформаційна система оптимізації логістичних маршрутів.

Розроблена система дозволяє розв’язувати задачі маршрутизації транспортних засобів з урахуванням часових вікон, робочих графіків водіїв та вантажопідйомності, забезпечуючи автоматичне перерахування маршрутів у реальному часі.

У ході розробки:

– проведено аналіз сучасних методів вирішення логістичних задач та обґрунтовано необхідність використання автоматизованих систем;
– обґрунтовано доцільність використання алгоритмів оптимізації та обрано інструментарій Google OR-Tools;
– побудовано архітектуру системи з використанням FastAPI та React;
– реалізовано програмну систему, що інтегрує картографічний рушій OSRM та розширення PostGIS для роботи з геоданими;
– проведено тестування, яке показало, що система швидко змінює маршрути при появі нових замовлень та точно передає місцезнаходження водія.

Середовище розробки – Visual Studio Code, Docker.

Ключові слова: оптимізація маршрутів, логістика, FastAPI, PostGIS.

ANNOTATION

The diploma project includes an explanatory note (54 p., 13 fig., 1 tables, list of used literature of 15 titles, 3 appendices).

The object of development is an Information System for logistics route optimization.

The developed system allows solving vehicle routing problems (VRP) considering time windows, driver schedules, and vehicle capacity, providing automatic route recalculation
in real-time.

During the development:

– an analysis of modern methods for solving logistics problems was carried out and the necessity of using automated systems was substantiated;
– the feasibility of using optimization algorithms was substantiated and the Google OR-Tools toolkit was selected;
– the system architecture was built using FastAPI and React;
– a software system was implemented that integrates the OSRM routing engine and PostGIS extension for working with geodata;
– testing was conducted, which showed that the system quickly changes routes when new orders appear and accurately transmits the driver's location.

Development environment – Visual Studio Code, Docker.

Keywords: route optimization, logistics, FastAPI, PostGIS.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045430.002 ТЗ	Інформаційна система			
			оптимізації логістичних			
			маршрутів.			
			Технічне завдання			
	А4	ІАЛЦ.045430.003 ТП	Інформаційна система			
			оптимізації логістичних			
			маршрутів.			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045430.004 ПЗ	Інформаційна система			
			оптимізації логістичних			
			маршрутів.			
			Пояснювальна записка			
	А4	ІАЛЦ.045430.001 Д1	Схема масового імпорту			
			даних.			
			Структурна схема.			

Змін.	Арк.	№ докум.	Підпис	Дата	***ІАЛЦ.045480.001 ОА***		
Розробив		Петринка О. І.			Інформаційна система оптимізації логістичних маршрутів **Опис альбому**	Літ.	Аркуш / Аркушів
Перевірив		Радченко К.О.					
Консульт.						КПІ ім. Ігоря Сікорського, ФПСПМ КВ-21	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	І	Інформаційна система	1		
			оптимізації логістичних			
			маршрутів.			
			Схема бази даних.			
			Структурна схема			
	А4	І	Інформаційна система			
			оптимізації логістичних			
			маршрутів.			
			Алгоритм оптимізації.			
			Блок-схема алгоритму			
	А4	І	Інформаційна система			
			оптимізації логістичних			
			маршрутів.			
			Алгоритм реоптимізації.			
			Блок-схема алгоритму			
		Диск CD-ROM	Текст пояснювальної			
			з			
			Графічний матеріал			

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Інформаційна система оптимізації логістичних маршрутів».

Галузь застосування: транспортна логістика, сервіси доставки, кур’єрські служби, управління постачаннями, автоматизована система управління флотом.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проєктування на здобуття другого (магістерського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є створення легкої у використанні інформаційної системи, що дозволяє здійснювати динамічну оптимізацію логістичних маршрутів з урахуванням часових обмежень та параметрів транспорту, а також виконувати аналітичну оцінку ефективності використання ресурсів автопарку в реальному часі.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література з транспортної логістики та методів оптимізації, документація бібліотеки Google OR-Tools, офіційна документація фреймворків FastAPI та React, технічні специфікації картографічного рушія OSRM, електронні ресурси та фахові публікації в мережі Інтернет.

5.ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється:

- забезпечення кросплатформенності за рахунок використання Docker-контейнеризації;
- реалізація динамічної оптимізації маршрутів на базі алгоритмів Google OR-Tools;
- підтримка масового імпорту замовлень через файли формату CSV;
- забезпечення автоматичного перерахунку маршрутів при зміні складу замовлень;
- наявність окремого додатка для водіїв з підтримкою офлайн-режиму;
- зручність користування системою завдяки сучасній темній темі та інтуїтивній навігації;

5.2 Вимоги до апаратного забезпечення:

- центральний процесор з тактовою частотою не менше 2.0 ГГц (мінімум 4 ядра);
- обсяг оперативної пам’яті не менше 8 ГБ;
- вільне місце на жорсткому диску – не менше 2 ГБ;
- наявність постійного доступу до мережі Інтернет для роботи картографічних сервісів;

5.3 Вимоги до програмного та апаратного забезпечення користувача:

- Операційна система: будь-яка ОС з підтримкою сучасних браузерів (Chrome, Firefox, Safari).

6.ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1	Вивчення літератури за тематикою роботи	14.11.2025
2	Розроблення та узгодження технічного завдання	27.11.2025
3	Аналіз існуючих програм для маршрутизації	29.12.2025
4	Розроблення структури інформаційної системи	05.02.2026
5	Розробка серверної частини системи	16.03.2026
6	Розроблення клієнтських інтерфейсів	20.04.2026
7	Тестування алгоритмів маршрутизації	21.05.2026
8	Підготовка матеріалів першого розділу дипломного проєкту	25.05.2026
9	Підготовка матеріалів другого розділу дипломного проєкту	29.05.2026
10	Підготовка матеріалів третього розділу дипломного проєкту	01.06.2026
11	Підготовка матеріалів четвертого розділу дипломного проєкту	02.06.2026
12	Оформлення технічної документації та графічної частини	03.06.2026
13	Попередній огляд матеріалів дипломної роботи на кафедрі	04.06.2026

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість	№	Примітки
	А4	ІАЛЦ.045430.004 ПЗ	Інформаційна система			
			оптимізації логістичних			
			маршрутів.			
			Пояснювальна записка			
	А4	І	Масовий імпорт даних.			
			Структурна схема			
	А4	І	Схема бази даних.	1		
			Структурна схема			
	А4	І	Алгоритм оптимізації.			
			Блок-схема алгоритму			
	А4	І	Алгоритм реоптимізації.	1		
			Блок-схема алгоритму			
		Диск CD-ROM	Текст пояснювальної			
			з			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	

ІАЛЦ.045430.003 ТП

Розробив		Петринка О.І.			
Перевірив		Радченко К.О.			
Консульт.					
Н. контроль		Клятченко Я.М.			
Зав. каф.		Романкевич В.О.			

Інформаційна система оптимізації логістичних маршрутів
Відомість технічного проєкту

Літ.	Аркуш	Аркушів

КПІ
ім. Ігоря Сікорського,
ФПСПМ КВ-21

Пояснювальна записка

до дипломного проєкту

на тему: «Інформаційна система оптимізації логістичних маршрутів»

Київ – 2025

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

API (англ. Application Programming Interface) – інтерфейс взаємодії програмних компонентів.

CSV (англ. Comma-Separated Values) – текстовий формат зберігання табличних даних.

FMS (англ. Fleet Management System) – система управління автопарком та водіями.

GPS (англ. Global Positioning System) – система супутникового визначення координат.

JWT (англ. JSON Web Token) – стандарт безпечної передачі даних для авторизації.

OR-Tools – бібліотека Google для розв’язання задач оптимізації маршрутів (VRP).

OSRM (англ. Open Source Routing Machine) – рушій для побудови дорожніх маршрутів.

PostGIS – розширення СУБД PostgreSQL для роботи з геоданими.

PWA (англ. Progressive Web App) – веб-додаток з функціями мобільної програми та GPS.

SPA (англ. Single Page Application) – веб-додаток, що працює без перезавантаження сторінок.

VRP (англ. Vehicle Routing Problem) – математична задача побудови оптимальних маршрутів.

ВСТУП

Станом на сьогодні електронна комерція розвивається надто стрімкими темпами. Дивлячись на ситуацію в Україні, і світі загалом, а саме: зростання вартості пального, зміна екологічних норм та нестабільна політична ситуація-логістика дедалі більше відіграє ключову роль. Транспортні витрати становлять суттєву частку собівартості продукції, при цьому значна частина ресурсів витрачається неефективно через недосконале планування маршрутів, відсутність автоматизованого контролю за рухом транспорту або несвоєчасне реагування на зміну дорожніх умов.

Дуже замітно це в Україні. Через воєнні дії, зміну звичних транспортних коридорів та пошкодження дорожньої інфраструктури виникає критична потреба у гнучкому та швидкому плануванні перевезень. У цих умовах ефективне управління автопарком має відігравати важливу роль у підтримці економічної стабільності та забезпеченні безперебійного постачання товарів першої необхідності.

Світовий досвід показує , що застосування програмних рішень для аналізу геопросторової інформації в парі з оптимізацією підвищує ефективність доставки та знижує витрати. Проте, велика частка українських підприємств ще не використовує інформаційні системи, які виконують динамічне планування маршрутів. А тим паче з врахуванням різноманітних критеріїв, таких як: час доставки, розташування водіїв, максимальна вага авто при повному навантаженні.

Відповідно, розробка спеціалізованого програмного забезпечення для автоматизації логістичних потреб є дуже актуальною. Такі системи мають забезпечити прокладання найефектившіних маршрутів з урахуванням часу та надмірного споживання палива та інших важливих критеріїв.

АНАЛІЗ ПРОБЛЕМИ ПРОБЛЕМИ ОПТИМІЗАЦІЇ ЛОГІЧНИХ МАРШРУТІВ

1.1 Актуальність автоматизації логістичних маршрутів

У сучасному світі активного розвитку цифрових систем логістика перестала бути тільки допоміжним чинником діяльності підприємства. Навпаки, вона є одним з ключових чинників конкурентності і ланкою стабільної роботи та доходу. Зростастання використання електронної комерції та цифровізація бізнесу висувають нові вимоги організації перевезень. Вони повинні враховувати швидкість доставки, мінімазацію витрат і точність виконання замовлень.

Аналітики свідчать про стабільний зріст електронної комерції в Україні. Труднощі з економічним становищем не зупиняють роботу. Прогнозується подальше зростання обсяг онлайн-продажів впродовж наступних років. Внаслідок цього логічним є суттєве збільшення кількості адресних доставок. Це створює додаткові навантаження на логістичні фірми та системи. Через такі умови звичайні підходи щодо планування перевезень стають малоефективними та не можуть забезпечити необхідного рівня адаптипності.

Особливо важким і найбільш дорогим етапом всього логістичного процесу є фінальний етап. Це остання стадія доставки товару від складу або центру розподілення посилок до кінцевого адресата. За приблизними оцінками міжнародних логістичних компаній - витрати на цей етап можуть перевищувати понад половину всіх витрат на транспортування. Це можна пояснити необхідністю обробки великої кількості адресних замовлень. Також непередбачуваною дорожньою обстановкою у містах, проблеми з паркуванням на необхідністю дотримання приблизного часу доставки.

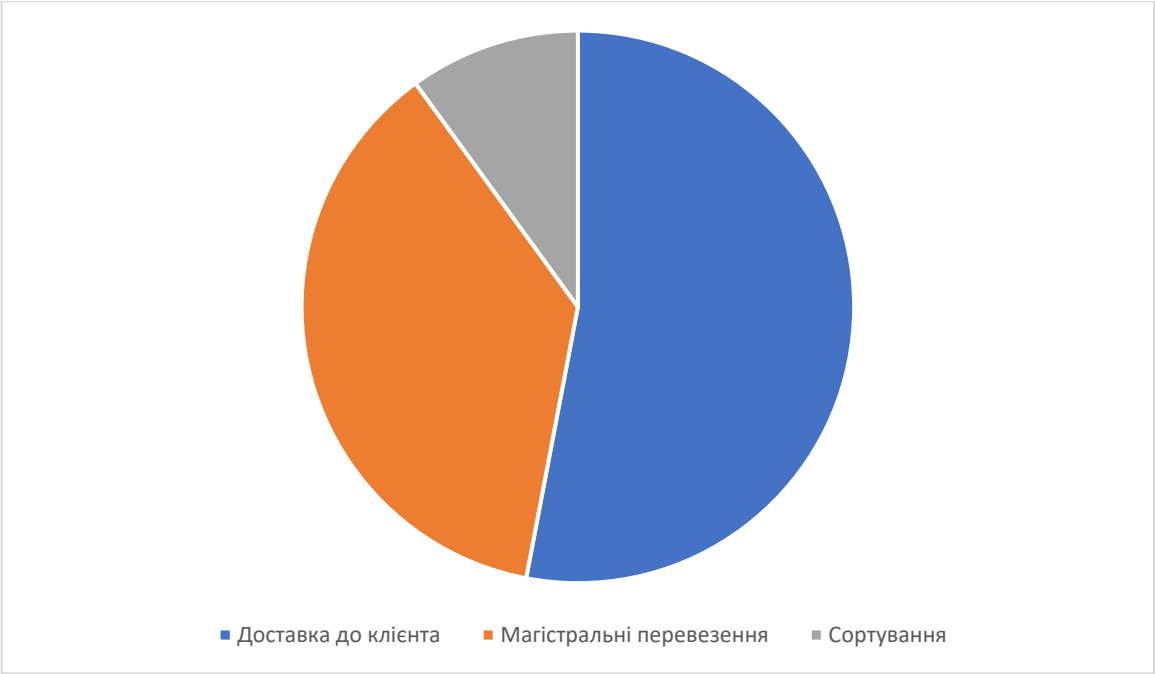


Рисунок 1.1 - Структура витрат у логістиці

З точки зору комп'ютерної інженерії та теорії алгоритмів, планування марш рутів вручну для великої кількості проміжних точок доставки стає практично неможливим. Задача прокладання оптимального плану маршрутів належить до класу NP – складності. Вона формалізується як задача маршрутизації транспортних засобів. Якщо для 5 точок доставки максимальна кількість можливих шляхів становить 5!(120 варіацій), то для 10 точок – вже аж понад 3,6 мільйонів варіантів (10!). У реальних логістичних системах кількість таких адрес може сягати десятків, сотень або навіть тисяч. Для 100 клієнтів число всіх можливих комбінацій буде дорівнювати 100!. Вручну перебрати таку кількість звичайно не можливо ніяк. За подібних умов людина не зможе ефективно провести аналіз всіх варіацій маршрутів та знайти найбільш оптимальне рішення за короткий термін. У кінцевому результати ручне планування навіть при малій кількості точок приводить до появи надлишкового пробігу. Це призводить до надлишкового обслуговування авто, збільшення витрат на амортизацію, пальне та суттєве збільшення часу доставки.

В Україні питання автоматизації логістики має особливе значення. Повномастабні бойові дії спричинили суттєві пошкодження транспортної інфраструктури - ускладнили або замінили повністю привичні маршрути перевізників та змушують підтриємців постійно шукати вихід та адаптуватися до чергових обмежень. Додатковими серйозними проблемами стали нестабільність, інші військові конфлікти, дефіцит та суттєве подорожчання пального, обмеження пересування у нічний час доби, а також потреба врахування графіків відключення електроенергії на складських приміщеннях. Через виникнення таких умов навіть невелике скорочення пробігу автомобіля стає суттєвою економією на паливі та амортизації. Для сучасного бізнесу це є необхідністю. Через неефективні маршрути доставки дорожчає кожен товар, що відчуває кожна людина.

Дедалі більшої уваги привертає до себе екологічний норми і їх стандарти. Згідно до європейського курсу розвитку України - підприємства поступово починають переходити до стабільного зменшення викидів шкідливих газів. Логістична сфера є одним з найбільш суттєвих джерель забруднення навколишнього середовища. Це стається через величезні об’єми викидів. Використання інформаційних систем маршрутизації дозволяє зменшити загальний кілометраж поїздок, скоротити час робти двигунів при простої та заторах. Це дає змогу знизити викиди вуглекислого газу приблизно до 25 % , що в свою чергу дає змогу наблизитись українським компаніям до екологічних норм, збільшує їх конкурентноспроможність та привабливість інвесторів.

Головна мета полягає в переході від звичайного статичного методу планування маршрутів до динамічних систем. У більшості звичайний логістих систем планування маршрутних листів відбувається єдиноразова. В основному ввечері або перед початком зміни. Ця схема більше не працює, тому що актуальність виконання замовлення або адреси постійно змінюються. Поява нових термінових заявок, скасування доставки з боку клієнта або виникнення заторів на дорогах вимагають використовувавати алгоритми швидкого перерахунку шляху наживо.

1.2 Огляд методів розв’язання задачі машрутизації транспортних засобів

Основою будь-якої автоматизованої системи управління логістикою є математична постановка задачі маршрутизації транспортних засобів. Вперше ця задача була представлена у 1959 році двома науковцями – Джорджем Данцігом та Джоном Рамсером. Сформулювано було для оптимізації перевезення паливних ресурсів до мереж автозаправних станцій. З тих пір звичайна задача комівояжера значно розширилась та модифікувалась в цілий клас складних задач комбінаторики. Вони на сьогоднішній день є обов’язковою складовою досліждень у сфері комп’ютерних наук та інформаційних досліджень.

У класичному тлумаченні задача маршрутизації описується за допомогою теорії графів. Логістична система моделюється у вигляда графа, який складається з вершин (вузлів) і ребер, які їх з’єднують. Головним вузлом є склад або логістичний центр (депо), з якого виконується виїзд автомобілів. Решта вершин слугують точкам доставки, тобто клієнтів . Кожне з ребер графу має певну «вагу», яка може описувати відстань між точками, час дороги або вартість затрат на паливо.

Основна мета оптимізаційної задачі полягає у формуванні набору маршрутів автомобілів, які починаються та завершуються в одному місці – депо, обходять усіх клієнтів і забезпечують мінімізацію витрат на виконання всіх етапів логістичних задач.

Щоб знайти оптимальне рішення необхідно врахувати декілька важливих факторів, які показують реальні умови компаній. У межах даного дослідження розглядається ускладнений варіант задачі, яка одночасно враховує два основних критерії - максимальна вага автомобіля при повному навантаженні та час доставки, в межах якого має здійснюватись обслуговування клієнта.

Перше з розглянутих обмежень відповідає задачі маршрутизації з врахуванням вантажопідйомності транспортних засобів. В цих умовах кожна одиниця автопарку має фіксований фізичний ліміт завантаження. Він може вимірюватись як і в фактичній вазі, так і в об’ємі або кількості палет різних

розмірів, які може вмістити в себе автомобіль. Під час розробки маршруту програма повинна на кожному етапі накопичувати сумарне навантаження всіх запланованих грузів. Якщо додавання чергового замовлення приведе до перебільшення допустимих норм, система буде вимушена або завершити даний маршрут з поверненням авто до логістичного центру для перевантаження, або формувать інший новий маршрут із залученням до роботи іншого транспорту.

Друге обмеження трансмформує задачу у варіацію VRP із часовими проміжками. В такому випадку кожен замовник додає допустимий для доставки час обслуговування. Врахування даного критерію значно ускладнює весь процес. У ситуації надто раннього прибуття автомобіля водій очікує початку дозволеного інтервалу, що зменшить ефективність роботи. Запізнення призводить до порушення умов обслуговування, що може спричинити репутаційні втрати або ж штрафні санкції, в залежності від політики компанії.

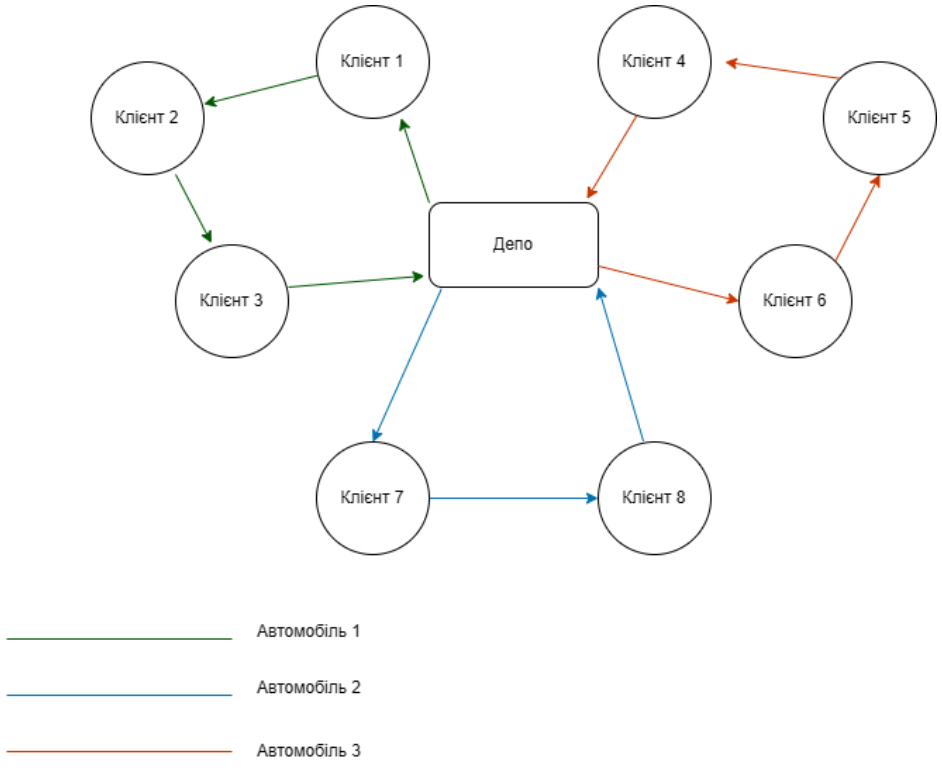


Рисунок 1.2 - Концептуальна графова модель маршрутизації транспортних засобів (VRP)

З позиції теорії алгоритмів та обчислювальної складості задача VRP в ідноситься до класу NP - складних. Це означає, що при прямому переборі всіх можливих варіацій маршрутів обчислювальні витрати зростають надто швидко. Логічно, що знахождення ідеального рішення при надто великих наборах даних стає майже неможливим у короткий термін.

Саме тому в практичних застосуваннях використовують різні підходи до вирішення задачі, які умовно поділень на три групи: точні методи, евристичні алгоритми та метаевристичні підходи. Точні методи гарантують оптимальність, але обчислення є дорогими. Евристики дозволяють швидко отримувати наближені рішення. Метаевристики поєднують елементи різних підходів, що забезпечує баланс між якість і часом обчислення.

1.2.1 Точні методи розв'язання

Точні методи, а саме алгоритм відсікаючих площин, грунтується на повному аналізі простору можливих рішень. У процесі роботи вони формують дерево маршрутів і поступово виключають ті гілки, для яких можна довести, що вони не містять оптимального розв'язку. Цей процес дозволяє звузити область пошуку до декількох потенційно оптимальних варіантів.

Точні алгоритми мають основну перевагу - є гарантія знаходження глобального оптимального рішення задачі. Але практичне використання обмежується через надмірну обчислювальну складність. Для задач середніх розмірів, наприклад до 100 точок, повний перебір може забирати надто багато часових ресурсів – від декількох годин до днів. Саме через цей чинник в умовах реальної роботи такі методи є малопридатними до практичного використання.

1.2.2 Евристичні алгоритми

Для подолання обмежень точних методів у реальній роботі були розроблені евристичні алгоритми. Вони не забезпечують знаходження найкращого маршруту, але можуть отримати прийнятні рішення за значно менший час. Це робить їх актуальними для використання в компаніях, де швидкість займає ключову роль.

Найбільш відомим є алгоритм «найближчого сусіда». Принцип досить інтуїтивно зрозумілий: авто вирушає з точки відправлення і на кожному етапу обирає найближчого ще не відвіданого клієнта, поки дозволяє обмеження вантажопідйомності. Хоча реалізація є простою і швидкість прийнятною, даний метод часто призводить до збиткових маршрутів, особливо на завершальних стадіях поїздки. Саме тоді автомобіль повинен їхати до останніх клієнтів, де відстань може бути значною.

Більш ефективним методом є алгоритм Кларка-Райта, який грунтується на ідеї поступового об’єднання маршрутів. Початоково кожен клієнт розглядається як окремий маршрут, що виконується з логістичного центру. Далі система робить прорахунок потенційної економії від об’єднання точок в один маршрут. Рахується різниця в витратах, яка виходить внаслідок заміни двох окремих маршрутів одним спільним. На основі цього показника алгоритм послідовно об’єднує найвигідніші комбінації.

1.2.3 Метаевристичні алгоритми

Метаевристичні методи оптимізації становлять найбільш розвинений клас алгоритмів розв’язання важких комбінаторних задач. Вони поєднують евристики для формування початкового допустимого рішення з подальшими ітеративними процедурами його вдосконалення. Основна перевага в їх можливості ефективно аналізувати рішення і уникати застягання у локальних оптимумах. Це стається коли алгоритм зупиняє покращення, уникаючи того, що глобально кращі варіації ще є.

Серед найбільш поширених можна виділити такі підходи:

Генетичні алгоритми: імітують механізми природної еволюції. У цьому методі кожен шлях подається у вигляді закодованої «хромосоми». Далі відбувається популяції різних рішень, які обмінюються фрагментами маршрутів та випадковими змінаим порядку клієнтів . На кожній ітерації відбираються найкращі варіанти, що мають мінімальну загальну вартість шляху, і саме їх використовують для генерації наступно го покоління.

Алгоритми імітації відпалу: базується на аналогії з фізичним процесом охолодження матеріалів. Його особливістю є ймовірне п рийняття навіть гірших рішень на початкових етапах прокладання. Це дає зможу виходити з локальних мінімумів і аналізувати більш е можливих комбінацій. По мірі «охолодження» системи ймовірність прийняття гірших рішень поступово меншає, і алгоритм переходить до стабілізації найкращого найденого шляху.

Алгоритм пошуку із заборонами (Tabu Search): Використовує механізм короткострокової пам'яті для уникнення циклів однакових рішень. У процесі оптимізації зміни шляху записуються у списку заборон, що тимчасово блокує їх хворотне виконання. Це дає змогу алгоритму не виконувати повторних аналізів неефективних конфігурацій і сприяє більш обширному пошуку.

У межах розробленої програми застосовується алогритмічне ядро Google OR-Tools , яке реалізує гібридний підхід до оптимізації. На стартовому етапі використовують швидкі евристики для побудови початкового допустимого маршруту з урахуванням обмежень грузопідйомності і часових меж. Далі запускається процедура локального покращення, яка виконує велику кількість ітеративних змін шляху. Їх головна ціль - покрокове зменшення фінальної вартості рішення. Така комбінація дає змогу досягти високої якості планування при збереженні мінімального часу обчислень. Це є надто важливим для інтеграції з високонавантаженими бекенд-системами.

1.3 Аналіз підходів до динамічної реоптимізації маршрутів

Історично системи керування автопарками базувались на основі статичного підходу маршрутизації. Процес був простий - вкінці робочого дня або вночі диспетчер передавав усі замовлення на наступний термін до системи оптимізації. В результаті формувались фіксовані листи для кожного з транспортних засобів , які після получення результату практично ніяк не змінювались. Це та модель, яка відповідає класичній статичній задачі маршрутизації. Вона ефективна у передбачуваних умовах , коли наприклад є постійні клієнти і все відомо зазделегідь. Але цей підхід має суттєві недоліки у

сучасному світі, вихідні дані можуть швидко втратити актуальність або маршрут може бути вже неактуальним після короткого часу виїзду автомобіля із депо.

Це зумовило розвиток нового класу задач – динамічної маршрутизації авто. Головна її особливість, що повний набір даних може бути не відомим на момент планування. Інформація може надходити поступово, паралельно з виконанням доставки. В таких умовах система повинна не лише будувати початковий план, але й змінювати його відповідно до нових вимог і точок.

Згідно наукових досліджень рівень складості динамічної маршрутизації оцінюється за допомогою окремого показника – ступеня динамічності. Ця метрика визначається як співвідношення кількості замовлень, які не були відомі раніше, до всієї кількості оброблених. Високе його значення показує про значний рівень невизначеності та потребу системи на швидку реакцію на зміни.

Чим ближчим є ступінь динамічності до одиниці, тим вищими стають вимоги до програми, продуктивності алгоритмів та ресурсів системи. В таких ситуаціях програма має забезпечити не тільки швидкі рішення, але й безперервну адаптації без порушення вже відомих операцій.

У реальних процесах виділяють такі події:

- Надходження нових термінових замовлень. У процесі виконання роботи можуть додаватись замовлення, наприклад експрес-доставка. В даній ситуації система повинна швидко додати нву точку до уже сформованого маршруту, таким чином, щоб не порушити часових обмежень інших клієнтів.
- Анулювання замовлень. покупець може скасувати замовлення вже після того, як автомобіль вирушив у дорогу. В таких випадках система має онововити план перевезення і прибрати цю точку зі шляху, для уникнення зай вих витрат по часу, амортизації і пальному.
- Затримки під час обслуговування. водій може витрати більше часу на видачу замовлення через непередбачувані події. Система має вміти коригувати маршрут при таких ситуаціях.

- Транспортні форс-мажори. дорожні затори, аварії, ремонти дороги, залізничні переїзди, поломка авто та інші події. В таких випадках система має виконати перерозподіл невиконаних замовлень між іншим доступним транспортним засобом.

Принцип зафіксованих зупинок (Committed Stops)

Одним з найскладніших аспектів проєктування є забезпечення правильної взаємодії між реальними діями водія та роботою алгоритму. Хоча програма і може перебудовувати маршрут зразу, інколи виникають ситуації, коли водій вже прямує до певної точки, тоді різка зміна шляху може бути незручною, або невідповідати правилам дорожнього руху .

Для стабільної роботи при динамічних змінах програмна архітектура повинна відрізнятись від звичайний рішень. Ключовою складовою роботи системи динамічної оптимізації є постійний потік геоданих, отриманих у процесі телеметрії транспортного засобу. Мобільний застосунок водія використовує GPS - модуль смартфона для автоматичного збору координат у фоновому режимі. Ці дані передаються на центральний сервер з певною частотою.

Серверна частина системи створена із застосуванням фреймворку FastAPI на мові програмування Python . У випадках виникнення тригерної ситуації, наприклад відмова від замовлення, бекенд здійснює послідовністю операцій для оновлення поточного стану систему і запуску модуля реоптимізації маршруту.

Система виконує запит до бази даних PostgreSQL та зчитує останню з доступних даних про місцезнаходження всіх транспортних засобів автопарку .

Відокремлення виконаних доставок, а також заявок, які перебувають у статусі виконання

Використовуючи реальну дорожню графову карту, OSRM швидко обраховує оновлені матриці відстаней і часу між теперішніми координатами авто та всіма точками доставок.

Після отримання матриць дані рухаються до Google OR-Tools , яке робить повторний розподіл замовлень між авто. Під час оптимізації програма контролює дотримання обмежень - часу і ваги.

Фінальний результат водії отримують у своєму застосунку. В результаті цього можливо одразу бачити нову послідовність точок доставки.

У результаті даний спосіб дає змогу змінювати процес маршрутизації із статичної процедури у постійний адаптивний механізм. Це забезпечує високу надійність розробки до змін та дає змогу якісно виконувати свої обов'язки перед клієнтами з максимально ефективним використанням ресурсів.

1.4 Порівняльний аналіз розробленої системи із аналогами

Динамічний розвиток логістики та сервісів доставки спричинив появі великої кількості програмних рішень. Переважна частина сучасних платформ функціонує за моделлю SaaS , надаючи компанії доступ до хмарних сервісів для управління автомобілями.

Хоч вони і мають великий обсяг функцій , більша частина комерційних рішень мають багато недоліків. Для деяких категорій підприємств такі системи можуть бути економічно збитковими або бути неадаптованими до локальних особливостей. З основних мінусів варто відмінити високу ціну підписки, залежність від закордонної інфраструктури та складність інтерграції.

З метою обгрунтування необхідності розробки власної системи було виконано аналіз лідерів міжнародних платформ маршрутизації , а саме Onfleet і Routific.

1.4.1 Аналіз платформи Onfleet (США)

Платформа Onfleet є одним з провідних рішень на глобальному ринку програмного забезпечення для організації доставки та керування транспортними операціями. Основна увага у системі надається забезпеченню прозорості процесів і підвищенню якості взаємодії між диспетчером, водієм і кінцевим клієнтом. Серед функції платформи можна відмітити інтерфейс з підтримкою відстеження авто в режимі реального часу, мобільні застосунки для водіїв і автоматичне надсилання сповіщень отримувачам.

Використання Onfleet має декілька суттєвих недоліків для нашого ринку. Одним з ключових є висока вартість: навіть базовий тарифний план передбачає

суттєвих витрат (особливо для малого бізнесу, який може навіть її не окупити) , а також обмеження по кількості виконаних доставок. Також компанії-користувачі не отримують доступ до внутрішньої логіки алгоритмів або вихідного коду, а всі дані зберігаються у зовнішній хмарній платформі.

Для українського бізнесу це створює додаткові ризики пов'язаних з безпекою інформації та конфіденційністю даних. Передача інформації про клієнтів, адрес та іншої важливої інформації може бути небезпечною для деяких компаній. На рисунку 1.4 продемонстрований приклад інтерфейсу диспетчерської панелі системи Onfleet.

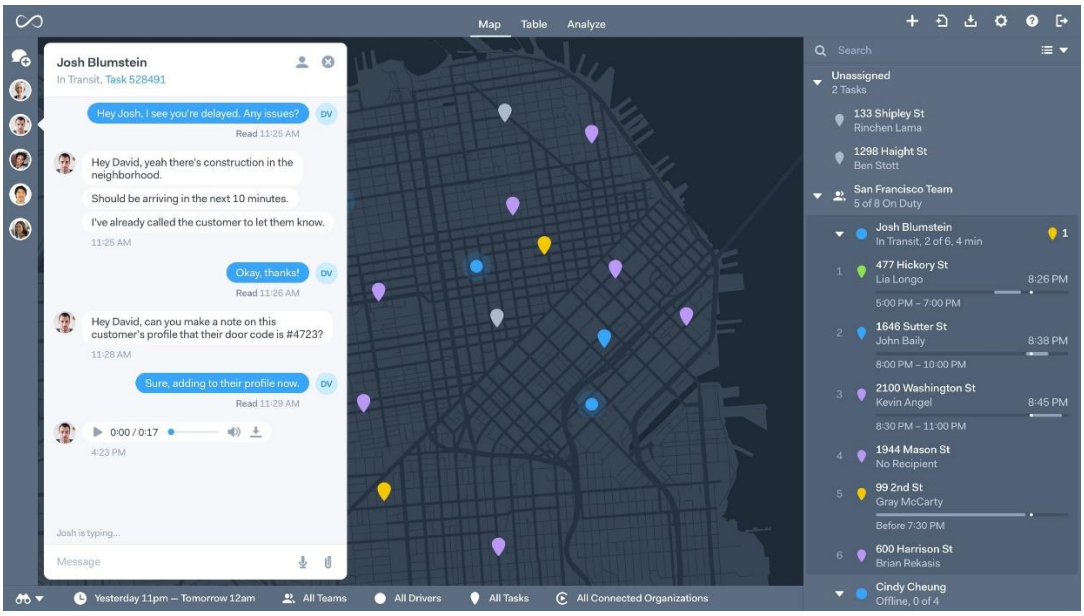


Рисунок 1.4 - Інтерфейс робочого вікна диспетчера у системі Onfleet

1.4.2 Аналіз платформи Routific (Канада)

Платформа Routific є рішенням для оптимізації транспортних маршрутів. Основна увага в ньому надана інтелектуальним алгоритмам маршрутизації. Система націлена в основному на малий та середній бізнес, які хочуть скоротити пробіг автомобілів і підвищити ефективність роботи. Однією з ключових переваг цієї платформи є якісне алгоритмічне ядро, призначене для задач прокладання шляхів з часовими проміжками.

Разом із тим проведений аналіз показав, що функціонал та можливості є обмеженими. Архітектура платформи більше розрахована на сценарії , коли список замовлень формують до початку робочого дня. У інших випадках, коли заявки надійдуть уже під час виконня шляху, механізми перерахунку працюють менш ефективно в порівнянні з сучасними системами.

Додатковим фактором є те, що система використовує комерційні картографічні сервіси, наприклад Maxbox , що створює залежність від сторонніх API та додає витрати на обробку запитів. У результаті частина з цих витрат фактично лягає на плечі кінцевого користувача платформи. На рисунку 1.5 представлено приклад інтерфейсу побудови маршрутів у систему Routific.

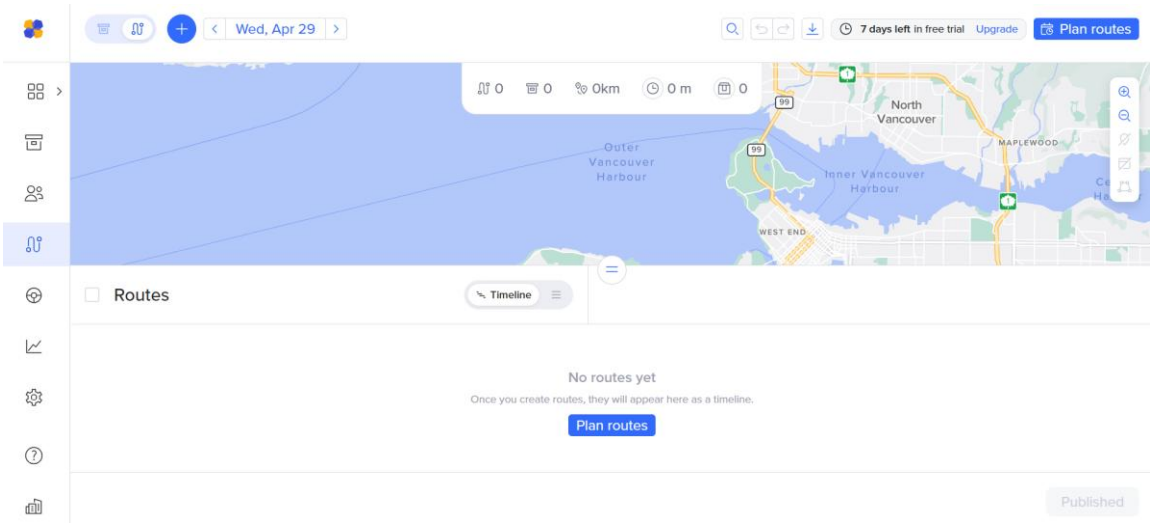


Рисунок 1.5 - Меню оптимізації маршрутів у системі Routific

1.4.3 Обґрунтування переваг розробленої системи

Розроблена в межах даного дипломного проєкту інформаційна система створювалась з метою усунення основник недоліків існуючих комерційних платформ. Ключова увага під час проєктування було зроблено на забезпеченні максимальної технічної незалежності.

Головною особливістю системи є використання власного локального сервера Open Source Routing Machine (OSRM). Цей підхід дасть змогу виконувати велику кількість запитів до картографічної інфраструктури без зайвих витрат на комерційні API сторонніх компаній. Програма реалізована за

допомогою бібліотеки Google OR-Tools. Вона є одним із найбільших відомих інструментів для розв'язку комбінаторних задач і забезпечує високу ефективність прокладання шляхів.

Окрему увагу під час розробки було мобільному інструментарію водія. На відміну від численних аналогів, що потребують встановлення застосунків, у системі використано технологію Progressive Web App (PWA). Це забезпечує можливість швидкого доступу до шляхів через любий сучасний веббраузер без необхідності встановлення додаткового програмного забезпечення. Цей спосіб є особливо важливим при використанні кур'єрами власних мобільних пристроїв.

Для детального зіставлення функціональних та технічних характеристик розглянутих платформ у роботі сформовано порівняльну таблицю розглянутих аналогів 1.1

Таблиця 1.1 – Порівняльна таблиця розглянутих аналогів

Параметр порівняння	Onfleet	Routific	Розроблена система
Карти	Платне API	Платне API	Безкоштовно
Тип розгортання	Тільки хмарне	Тільки хмарне	Локальне
Додаток водія	Нативний (IOS/Android)	Нативний (IOS/Android)	Веб-додаток
Динамічна реоптимізація	Підтримується	Обмежена	Повна
Безпека даних	Залежність від провайдера	Залежність від пройдера	Повний контроль і БД
Вартість	Висока щомісячна плата	Середня щомісячна плата	Мінімальна

На підставі проведеного аналізу зроблено висновок, що розробка власної інформаційної системи оптимізації логістих маршрутів з використання сучасних Open-Source рішень є економічно вигідною. Цей спосіб дає змогу уникнути

залежності від дорогих комерційних платформ, забезпечуючи при цьому високий рівень функціональності.

Створена програма може ефективно адаптуватися до специфічних умов функціонування українських логістичних підприємств. Вона буде враховувати нестабільність транспортного середовища, зміну шляху та необхідність в швидкому реагуванні на несподівані події.

1.5 Постановка задачі

Через проведений аналіз математичних моделей задачі маршрутизацій транспортних засобів та вивчення недоліків існуючих рішень було продумано концепцію та вимоги до інформаційної системи.

Об’єктом дослідження є процес управління логістичними маршрутами автопарку в умовах динамічної зміни точок доставки, зокрема при зміні дорожньої обстановки та додаванні нових замовлень.

Предметом досліження є математичні моделі, алгоритми оптимізації, побудовані на базі відкритих технологій. Система має забезпечити незалежність від комерційних платних продуктів, автоматизований перерахунок шляхів у режимі реального часу та вміти адаптуватись до змін середовища.

Для досягнення поставленої цілі необхідно виконати декілька основних задач, а саме:

Провести глибокий аналіз задачі логістичної оптимізації, досліження класичних підходів до розв’язання оптимізації логістичних маршрутів з урахуванням часових меж та пояснити вибір стеку для розробки системи.

Розробити архітектуру системи, яка буде мати серверний модуль, р еляційну базу даних з підтримкою геопросторових даних, картографічний рушій і застосунки.

Розробити модуль алгоритмізації на основі бібліотеки Google OR-Tools для формування початкових маршрутів та їх подальшої оптимізації з врахуванням максимальної ваги авто при завантаженні і часових меж.

Інтегрувати системи з локальним сервером Open Source Routing Machine (OSRM) для отримання матриць часу і відстаней без використання сторонніх API сервісів.

Розробити модуль динамічної реоптимізації шляхів, який може швидко поміняти логістичні плани у відповідь на додавання або видалення замовлення з врахуванням механізму зафіксованих зупинок.

Реалізувати вебінтерфейс диспетчера та застосунок для водіїв.

Виконати тестування програмного забезпечення на тестових наборах даних, перевірити ефективність алгоритмів та економічну вигідність.

Висновки до розділу

У першому розділі пояснювальної записки виконано детальний аналіз проблематики управління логістичними маршрутами автопарку в реальних умовах роботи бізнесу.

Було пояснено актуальність цієї розробки. Показано, що активний розвиток електронної комерції , здорожчання палива і труднощі дорожньої інфраструктури пов’язаних з воєнним станом в Україні, значно підвищують необхідність у використанні розумних систем логістики. Користування такими рішеннями дає змогу бізнесу знизити витрати та стабільно працювати.

Далі було розглянуто математичну базу задачі маршрутизації, яка знаходиться в основі сучасних систем управління перевезеннями. Під час аналізу було відзначено, що через високу складність задачі найбільш ефективним підходом є метаевристичні алгоритми та комбіновані методи оптимізації. Виходячи з цього, було обгрунтовано використання бібліотеки Google OR-Tools як основного алгоритмічного ядра системи.

Також було детально розглянуто концепцію динамічної реоптимізації маршрутів. Доведено, що система логістичного керування має підтримувати перерахунок маршруту «на льоту» з визначеними зупинками , відслідковувати місцезнаходження транспорту.

Проведений аналіз вже існуючих рішень - платформи Onfleet та Routific. Результатом цього є виявлення низки обмежень таких платформ - велика плата за підписку, залежність від комерційного API та відсутність локального розгортання системи. Це доводить актуальність створення власної інформаційної системи оптимізації логістичних маршрутів на основі технологій - FastAPI , Postgre SQL , PostGIS та інших.

Отже, було визначено основний перелік вимог до майбутньої програми та завдань, необхідних для її реалізації.

МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ, ОБГРУТУВАННЯ ТЕХНОЛОГІЧНОГО СТЕКУ

2.1 Математична модель задачі VRP з урахуванням часових вікон

Для побудови ефективної та масштабованої інформаційної системи управління логістичними процесами першочерговим завданням є формалізація предметної області. Перехід від абстрактного опису бізнес-процесів доставки до чіткої логіко-математичної моделі дозволяє застосувати сучасні підходи комбінаторної оптимізації. Розроблювана система орієнтована на автоматизацію та оптимізацію процесів доставки до клієнта. Фундаментальну математичну основу обрано класичну задачу маршрутизації транспортних засобів із обмеженнями вантажопідйомності та часовими вікнами.

На противагу традиційним підходам, де логістична задача розглядається виключно як статична система складних лінійних рівнянь, у межах цього проєкту застосовано структурно-алгоритмічну модель. Такий підхід дозволяє детально описати особливості практичної реалізації моделі засобами сучасних високопродуктивних програмованих солверів та відобразити специфіку інтеграції математичного ядра з геопросторовими сервісами обробки просторових даних у реальному часі.

Логістична мережа моделюється як зважений орієнтований граф. У цій моделі виділяються 2 типи вершин. Перший – це вузли-депо, які є складами або логістичними центрами, звідки автомобілі починають свій рух, і куди повертаються після завершнення роботи. Другим типом є клієнти, які позначені як точки доставки замовлень.

Між вершинами графа формуються орієнтовані ребра, які описують можливі маршрути рухи транспортних засобів. Кожне з ребер описане набором параметрів, які визначають його «вагу» для алгоритму. Основними метриками є:

1. Відстань у різних одиницях вимірювання.
2. Час переміщення між вузлами, яка вимірюється у різних одиницях та залежить від стану дорогу. Формування точних значень цих параметрів

покладається на окремий мікросервіс OSRM, який обчислює матриці відстаней і часу для кожної з вершин графа на основі реальної транспортної інфраструктури.

Для запуску процесу оптимізації програма формує певний набір початкових даних. Вони використовуються ядром під час пошуку оптимального маршруту.

Кожна точка доставки як складова вектору попиту має певне значення пріоритету, що відображає обсяг замовлення. Цей параметр поданий у вигляди ваги, кількості або об’єму продукції.

Для кожного вузла визначається часовий інтервал, де задаються допустимі межі доставки. Додатково враховують параметр часу обслуги, який описує тривалість видачі замовлення і його розвантаження.

Модель містить множину доступних автомобілів, для кожного з яких є обмеженння по вазі товару, а також межі часу робочої зміни водія. Ці параметри враховуються алгоритмом при створенні шляхів та розподілі продукції між автомобілями.

У процесі оптимізації система визначає значення набору змінних, які описують склад майбутнього маршруту. Основою моделі є бінарні змінні, що показують факт руху автомобіля між двома вузлами графа. Для кожної точки визначається чи буде певний транспортний засіб здійснювати перехід від одного клієнта до іншого в межах даного маршруту.

Також алгоритм створює набір часових змінних, які визначають приблизний момент прибуття автомобіля до кожної з точок. На основі цих даних система створює детальний графік виконання маршруту та перевіряє це з обмеженнями віж клієнтів.

Основною ціллю моделі є мінімалізація загальних операційних витрат логістичної системи. У рамках розробленого програмного забезпечення це завдання виконується через зменшення сумарної довжини маршрутів або загального часу пересування усіх транспортних засобів.

Під час пошуку розв’язку алгоритм аналізує велику множину комбінацій маршрутів та обирає той, який забезпечить найменше значення вартості перевезень при дотриманні всіх критеріїв.

Для того щоб розв’язок моделі був правильний він включає набір категоричних логічних та інших заборон. Вони гарантують, що ці шляхи можуть бути застосовані в реальних умовах використання автомобіля.

У випадках порушень навіть однієї з заборон програмний комплекс позначає такий маршрут як непридатний та виключає його з подального розгляду. Це дає змогу алгоритму працювати тільки з тими рішеннями, які є можливими для реального використання:

1. Обмеження обов'язковості та унікальності відвідування. Система гарантує, що кожна з активних точок буде включена до маршруту тільки раз. Алгоритм виключає можливість пропуску проміжної точки або призначенні такої декільком трансортним засобам одночасно.
2. Обмеження нерозривності маршруту (Flow Conservation). Дане правило контролює логічну послідовність переміщення автомобіля в межах дороги. Якщо транспорт прибуває до певної точки, він повинен її залишити та продовжити переміщення далі відповідно до побудованого графа. Це дозволить уникнути ситуацій некоректного завершення маршруту або його розрив. Автомобілі завершують маршрут тільки в депо.
3. Обмеження місткості автотранспорту (Capacity Constraints). У процесі створення шляху програма обраховує суму ваги товарів або кількості, прив’язаних за певним рейсом. Дане значення не повинне перевищувати допустимі ліміти.
4. Обмеження часових вікон (Time Windows Constraints). Для кожного з замовників враховується допустимий часовий проміжок розвантаження.
5. Запобігання ізольованим підмаршрутам (Subtour Elimination). Це спеціальне алгоритмічне правило запобігає формуванню окремих замкнених циклів, які не пов’язані з основним депо. Через це всі

маршрути утворюють одну цілісну струкруту та відповідають вимогам реального процесу перевезення.

2.2 Порівняльний аналіз методів оптимізації, бібліотека Google OR-Tools

Одним з ключових факторів розроблення є вибір ефективного алгоритмічного підходу для розв’язку задач оптимізації. Як було встановлено раніше, задача характеризується надзвичайно високою обчислювальною складістю.

Загалом, це означає що зі збільшенням кількості точок доставки та транспортних хасобів кількість можливих маршрутів зростає суттєво. У реальних логістичних системах, де одночасно використовується десятки авто і обробляються сотні замовлень, повний перебір усіх можливих варіантів стає неможливим.

Для вибору оптимального алгоритмічного ядра розроблювальної системи треба зробити порівняльний аналіз зазначених підходів з урахуванням сцефіки динамічної транспортної логістики . Окрему вагу варто приділити таким характеристикам, як швидкодія, маштабованість, здатність працювати в режимі реального часу та ефективність обробки великих даних.

Аналіз підходів до розв'язання задачі VRP

Точні методи (Exact Methods). До категорії точних методів відносяться алгоритми, які забезпечують глобальний оптимальний розв’язок проблеми. Найбільш відомі підходи серед них є метод гілок і меж, алгоритм відсікаючих площин, метод генерації стовпців. Основна перевага таких методів це гарантоване отримання найбільш ефективного розв’язання.

Проте, реально використання таких методів для планування маршрутів сильно обмежуються їх обчислювальною складністю. По мірі зростання кількості точок доставки час розрахунку збільшується кратно. Метод варто розлядати максимум до 50-100 вузлів.

Для нашої системи, яка має перераховувати і змінювати шлях одразу, використовувати такі підходи є поганим рішенням. Це буде надто довго.

Евристики (Constructive Heuristics). Евристичні алгоритми орієнтовані на отримання ефективно допустимого розв'язку задачі. Вони не гарантують найкращого з можливих розв'язків, але розв'язок буде оптимальним і швидким. Побудова рішення відбувається покроково. Найбільш популярні з таких є: алгоритм Кларка-Райта та різні варіації алгоритмів вставки. Їх основною перевагою є швидкість роботи, навіть для великих масивів даних маршрут може бути сформовано за долі секунди.

Метаевристики (Metaheuristics). Метаевристичні алгоритми показують себе як найефективніші і є найбільш технологічно сучасним підходом до розв'язання задач оптимізації у промислових масштабах. Метаевристики не формують маршрут безпосередньо, а виступають у ролі високорівневого механізму керування процесом пошуку оптимального вирішення.

Принцип роботи полягає в тому, що на початку створюється початковий допустимий шлях, потім алгоритм робить його покращення завдяки локальним змінам. Серед них варто відмітити - перестановку вузлів, заміну ребер і зміни порядку доставки. Це дозволяє поступово знижувати вартість маршруту та знаходити більш якісні шляхи в порівнянні з іншими методами.

Обґрунтування вибору Google OR-Tools

Власноручна реалізація алгоритмів є надто складним процесом. Цей процес часто виходить за межу розробки прикладних комерційних систем та належить до сфери наукових досліджень. Додатковим недоліком є те, що самостійно реалізовані алгоритми на мові Python не здатні забезпечити достатню продуктивність з великими даними.

Взявши це все до уваги, для реалізації математичного ядра системи було вибрати спеціалізовану бібліотеки Google OR-Tools. Це відкритий програмний комплекс, який розроблений Google AI , що фактично став стандартом у сфері комбінаторної оптимізації.

Вибір Google OR-Tools пояснюється такими перевагами:

Висока продуктивність : бібліотеки розроблена на мові C++, що забезпечує високу швидкість розрахунків і ефективне використання ресурсів. Також, для розробників доступні зручні Python-декоратори.

Спеціалізований модуль VRP Routing Solver: Google OR-Tools має окремий модуль маршрутизації, орієнтований саме на задачі VRP . Він має вбудовану підтримку багатьох логістичних обмежень - робота з кількома депо, часові проміжки та інші обмеження.

Гнучка система розв'язання : бібліотека дає змогу реалізувати гібридний підхід оптимізацій. Через налаштування парамкетрів девелопер може управляти алгоритмами. У цьому проєкті для швидкого отримання першого допустимого маршруту використовується конструктивна евристики PATH_CHEAPEST_ ARC , яка допомагає програмі практично одразу формувати початковий план доставки.

Керування часом пошуку : У системах динамічної логістики варто уникати пошуку ідеального рішення через значний проміжок часу розрахунку . Бібліотека дозволяє задавати параметр обмеження часу виконання. У розробленій системі він становить 30 секунд. Протягом цього інтервалу алгоритм виконує локальний пошук і поступово покращує маршрут, після чого видає найкращий знайдений варіант.

2.3 Обгрунтування вибору FastAPI та PostgreSQL для бекенду

Після визначення математичної моделі та вибору ядра Google OR-Tools наступає етап проєктування потужної, масштабованої на надійної серверної частини. Бекенд у нас виконує роль центрального вузла, що забезпечує взаємодію між базою даних, модулем оптимізації та клієнтськими застосунками.

Система орієнтована на реоптимізацію в реальному часі. Відповідно через це до серверної частини вимоги стають більшими через стійкість до навантажень. Вона повинна ефективно обробляти велику кількість запитів від водіїв і диспетчерів, забезпечуючи мінімальні затримки .

Вибір мови програмування та веб-фреймворку

Класично для розробки корпоративних логістичних систем застосовуються мови програмування Java або C#. В межах даного проєкту ключим фактором стала наяність прямої інтеграції з бібліотекою Google OR-Tools, що визначило мову Python як основну.

Висока продуктивність (ASGI): Завдяки використанню асинхронної серверної архітектури , FastAPI забезпечує обробку великої кількості одночасних підключень. Це дозволяє ефективно отримувати потокові GPS-дані телеметрії від усіх водіїв автопарку без значних затримок і втрати продуктивності системи.

Вбудована валідація даних (Pydantic): У логістичних системах точність вхідної інформації є критично важливою. Особливо це важливо для географічних координат та параметрів замовлень. FastAPI використовує Pydantic для автоматичної валідації та конвертації типів даних на етапі прийому запиту, перед виконанням бізнес-логіки. Це дозволяє відсіювати некоректні або неповні запити та зменшувати кількість помилок у подальшій обробці даних.

Вибір системи управління базами даних (СУБД)

Під час роботи наша система працює з великими об’ємами даних. Дані мають виражену реляційну природу та пов’язані зв’язки між собою (водій, авто, замовлення, клієнти, маршрути). Застосування нереляційних баз даних, як MongoDB , для таких завдань є недоцільним. NoSQL не здатний забезпечити необхідної транзакційної надійності та дотримуватися ACID.

Тому за основу було вибрано базу даних PostgreSQL. Плюсами є безкоштовна ліцензія, інтуївна зрозумілість, а також наявність просторового розширення PostGIS і індексів GiST.

PostGIS перетворює PostgreSQL на повноцінну просторову базу даних , що дає змогу добре працювати з координатами:

Типи просторових даних: Координати зберігаються не просто в форматі (широта, довгота). Вони мають спеціальний тип даних - GEOMETRY .

Просторові індекси (GiST): GiST індекс дає змогу базі даних максимально швидко видавати результат запиту до бази даних. Це дає змогу практично одразу отримувати потрібний нам результат, не чекаючи довгий час на обробку запиту.

Просторові функції: Мають змогу розраховувати пряму відстань, належність точки доставки до певної зони. Обчислюється за допомогою SQL - запитів.

Для зв’язку бекенду з базою даних було обрано бібліотеку SQLAlchemy.

Взаємодія цих сучасних технологій забезпечує надійний на продуктивний фундамент.

Висновки до розділу

У другому розділі дипломного проєкту було розглянуто як і алгоритми, так і обрано технології для розробки системи .

Було описано математичну модель логістичої мережі та описано задачу маршрутизації з обмеженнями. Головна мета визначена як зменшення загального пробігу авто і економія ресурсів. Основні обмеження: транспорт не можна перевантажувати, маршрут має бути послідовним, замовлення мають бути доставлені вчасно.

Проведено аналіз алгоритмів оптимізації . Доведено, що точні прорахунки є надто збитковими по часу. Тому математичним ядром обрано бібліотеку Google OR-Tools. Вона містить швидкі алгоритми з суворим обмеженням часу на розв’язок задачі. Це дає змогу отримувани якісні результати, а архітектура на мові C ++ забезпечує швидкість роботи системи.

Було обгрунтовано вибір технологічний стеку. Основним інструментом розробки обрано фреймворк FastAPI . Для цілісності, впорядкованості та роботи з просторовими даними обрано базу даних PostgreSQL з розширенням PostGIS.

Підсумовуючи, в цьому розділі розглянуто технічну базу проєкту. Визначені моделі, алгоритми та інструменти.

3. ПРОЕКТУВАННЯ ТА АЛГОРИТМІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Розробка архітектури системи на основі Docker

Відповідно до сучасних вимог програмного забезпечення, потрібно розробити легко масштабовану систему. Вона повинна швидко розгортатись на нових пристроях та бути надійними до збою. Звичайний підхід, за якого всі компоненти програми (база даних, веб-сервер, бекенд) встановлені безпосередньо на операційну систему має суттєву кількість недоліків - конфлікти версій бібліотек, відсутність ізоляції середовищ та складість міграції.

Для уникнення подібних пробленим під час розробки інформаційної системи логістичних маршрутів було приняте рішення використовувати технології контейнеризації Docker та Docker Compose . Ця технологія дає змогу розділити складі системи на незалежні модулі. Кожен з них виконується у ізольованому середовищі (контейнер) і має всі необхідні для роботи залежності.

Відповідно до конфігураційного файлу docker-compose.yml, система складається з наступних ключових сервісів:

- База даних. Контейнер на основі образу PostgreSQL. Відповідає за зберігання даних. Для їх збереження при перезапуску чи оновленні контейнера використовується технологія Docker Volumes .
- Сервіс OSRM. Контейнер, який виконує роль високопродуктивного математичного бекенду для розрахунку матриць відстаней і часу.
- Backend. Основний вузол програми, який иістить розроблений додаток. Контейнер збирається з унікального Dockerfile. Він має в собі всю бізнес-логіку системи: автентифікацію, керування замовленнями, виклик Google OR-Tools та взаємодію з базою даних через SQLAlchemy. Налаштований на автоматичний перезапуск у разі збою.
- Frontend. Контейнери, які відповідають за роботу візуальних інтерфейсів. Для робочої версії програми весь код інтерфейсів

стискається та оптимізується. Для миттєвого відкривання у своїх браузерах використовується швидкий та надійний веб-сервер Nginx .

Docker Compose створює ізольовану віртуальну мережу. Це його важлива перевага. Завдяки цьому сервіси можуть спілкуватися між собою внутрішними DNS. Наприклад, бекенд-сервер звертається до бази даних за адресою хоста 5432, а до маршрутизатора 5000. На зовнішньому рівні відкриваються лише необхідні користувачам порт. В той час сама база даних залишається повністю прихованою від зовнішнього впливу.

Для спрощення розробки та розгортання було створено набір скриптів, який мінімузіє людський фактор при налаштування середовища:

- init-osrm.sh Автоматизує процес підготовки картографічних даних. Завантажує карту, обробляє граф доріг і зберігає готові бінарні файли у директорію osrm-data . Це дає змогу уникати нестачу оперативної пам’яті під час основного запуску.
- deploy.sh – це головний файл розгортання системи на новому пристрої. Послідовно ініціалізує змінні середовища з файлу .env, запускає процес збірки образів та підіймає весь комплекс мікросервісів командою docker-compose up -d.

Такий підхід робить систему платформонезалежною. Система може бути розгорнута на будь-якому хмарному сервері за короткий проміжок часу, вимагаючи мінімальних конфігурацій.

3.2 Алгоритм інтеграції з OSRM для розрахунку матриць відстаней

Для прокладання ефективних маршрутів недостатньо відстань від точки до точки напряму. Реальний рух автомобіля обмежений дорожнім покриттям та його станом, правилами дорожнього руху та транспортними розв’язками. Основним компонентом системи, що відповідає за розрахунок реальних дорожнії метрик , є OSRM.

OSRM виступає в ролі сервера обробки координат. Він працює з графом доріг, отриманих з бази OpenStreetMap . У розробленій системі інтеграція з

OSRM реалізована через спеціалізований модуль geospatial.py на стороні бекенду.

Для того, щоб бібліотека Google OR-Tools могла знайти нормальний маршрут, їй необхідно знати відстань та час подорожі між кожною парою точок. Цей процес реалізованим таким чином:

- Збір координат. Система отримує список актуальних замовлень та координати з бази даних.
- Формування запиту. Бекенд надсилає масив координат до OSRM. Цей модуль замість побудови одного маршруту розраховує шляхи між усіма надісланими точками.
- Отримання та обробка матриці. OSRM повертає мартицю. В ній кожний елемент містить час подорожі або відстань від точки А до точки В.
- Типіцізація та нормалізація. Google OR-Tools працює з цілочисельними даними. Відповідно отримані від OSRM значення приводяться до типу Integer для уникнення помилок.

Візуалізація маршрутів на карті

OSRM використовується не тільки для розрахунку матриць відстаней. А також для отримання точної геометрії вже побудованих маршрутів. Після того, як оптимізатор визначив послідовність точок, система переходить до наступного етапу.

Алгоритм запиту геометрії передбачає передачу впорядкованого списку коорінат конкретного маршруту і отримання відповіді у форматі GeoJSON .

Збереженення отриманих результатів у базу даних (формат WKT) для подальшої візуалізаці на карту в інтерфейсі диспетчера за допомогою бібліотеки MapLibre GL .

Переваги обраного підходу

Локальне розгортання OSRM у Docker-контейнері дає системі декілька суттєвих переваг порівняно з використанням зовнішніх сервісів:

- Швидкодія. Запити в мережах однієї віртуальної мережі Docker відбуваються майже миттєво.

- Відсутність витрат. Система не вимагає платити за кожну розраховану координату. Це робить її економічно вигідною для бізнесу з великою кількістю замовлень.
- Автономність. Розрахунки можуть проводитися без підключення до інтернету, за умови вже завантажених карт.

Розроблений алгоритм інтеграції забезпечує систему точними картографічними даними, які є основною правильних розрахунків.

3.3 Проєктування реляційної структури бази даних

Основою інформаційної системи є надійна реляційна структура збереження даних. Вона спроєктована в 3-й нормальній формі (3NF) для уникнення дублювання інформації. Взаємодія з базою даних реалізована за допомогою сучасного асинхронного ORM-фреймворку SQLAlchemy 2.0. Фізична структура бази даних розділена на п'ять основних логічних сутностей: користувачі (Users), транспортні засоби (Vehicles), замовлення (Orders), маршрути (Routes) та телеметрія (Vehicle Positions).

Таблиця користувачів (Users) зберігає облікові дані та ідентифікатор користувача. Окрім стандартних полів (електронна пошта, хешований пароль, ПІБ), таблиця містить перелічуваний тип ролей (Dispatcher, Driver, Admin) для розмежування прав доступу. Для водіїв передбачено поля shift_start та shift_end, що фіксують графік робочої зміни у форматі UTC. Також водії мають зовнішній ключ assigned_vehicle_id, який встановлює зв'язок один до одного між водієм та конкретним транспортним засобом.

Таблиця автопарку (Vehicles) описує технічні та фінансові характеристики транспорту. Поле vehicle_type класифікує машину (Truck, Van, Motorcycle). Критично важливими для процесу оптимізації є поля capacity_kg (максимальна вантажопідйомність) та економічні показники cost_per_km і cost_per_hour. Їх алгоритм використовує для розрахунку вартості використання кожної машини. Географічне розташування початкового депо автомобіля зберігається у просторовому полі start_location типу Geometry('POINT', srid=4326).

Таблиця замовлень (Orders) зберігає всю необхідну інформацію про посилку та вимоги клієнта. Адресна інформація дублюється у двох форматах: текстовому (для відображення в інтерфейсі) та просторовому (поле location для оптимізатора). Додатково таблиця містить параметри weight_kg (вага вантажу), service_time_minutes (час, необхідний водію на розвантаження) та часові вікна time_window_start або time_window_end, які виступають жорсткими математичними обмеженнями при генерації рейсу.

Таблиці маршрутизації (Routes та RouteSteps) реалізують зв'язок один до багатьох. Таблиця Routes зберігає загальні агреговані метрики побудованого рейсу - прив'язку до автомобіля, заплановану дату, сумарну дистанцію (total_distance_km), загальну тривалість та фінальну розраховану вартість виконання (total_cost). Геометрія всього маршруту зберігається у полі типу LINESTRING. Дочірня таблиця RouteSteps деталізує маршрут покроково. Вона містить зовнішній ключ на замовлення (order_id), порядковий номер зупинки (sequence_order), тип кроку (виїзд з депо або доставка) та розрахунковий час прибуття (ETA).

Таблиця телеметрії (Vehicle Positions) забезпечує зберігання історичних та поточних GPS-даних. Фіксує точні географічні координати автомобіля, швидкість руху та часову мітку для відображення реального місцезнаходження транспорту на карті диспетчера.

3.4 Деталізація структури бази даних

Для забезпечення надійного зберігання інформації та підтримки цілісності даних (Referential Integrity) розроблено реляційну схему бази даних. Архітектура сховища побудована з урахуванням нормалізації до третьої нормальної форми (3NF), що виключає надмірність та різноманітних аномалій.

Нижче наведено детальну специфікацію ключових відношень (таблиць) бази даних інформаційної системи:

1. Відношення users (Користувачі системи)

Таблиця призначена для зберігання облікових даних та ідентифікації персоналу компанії.

id (Integer, Primary Key) - унікальний ідентифікатор користувача;

email (String, Unique, Index) - електронна пошта для авторизації;

full_name (String) - повне ім'я співробітника;

hashed_password (String) - криптографічний хеш пароля (алгоритм bcrypt);

is_active (Boolean) - прапорець активності облікового запису (за замовчуванням True);

role (Enum) - рівень доступу користувача. Можливі значення: DISPATCHER, DRIVER, ADMIN;

shift_start та shift_end (Time) - часові межі робочої зміни водія у форматі UTC;

assigned_vehicle_id (Integer, Foreign Key) - зовнішній ключ, що вказує на закріплений транспортний засіб із таблиці vehicles.

2. Відношення vehicles (Транспортні засоби)

Таблиця містить техніко-економічні характеристики автопарку, необхідні для роботи математичного солвера.

id (Integer, Primary Key) - унікальний ідентифікатор автомобіля;

name (String) - диспетчерська назва борту (наприклад, "Van-01");

vehicle_type (Enum) - тип кузова: TRUCK, VAN або MOTORCYCLE;

capacity_kg (Float) - максимальна допустима вантажопідйомність у кілограмах;

cost_per_km (Float) - фінансова вартість одного кілометра пробігу;

cost_per_hour (Float) - фінансова вартість однієї години роботи даного автомобіля;

start_location (Geometry('POINT', 4326)) - геопросторові координати початкового депо автомобіля у форматі PostGIS.

3. Відношення orders (Замовлення та точки доставки)

Таблиця агрегує всю вхідну інформацію про посилки.

id (Integer, Primary Key) - системний номер замовлення;

customer_name (String) - ім'я отримувача;

address (String) - текстовий опис адреси доставки;

location (Geometry('POINT', 4326)) - просторові координати клієнта;

weight_kg (Float) - фізична вага відправлення;

service_time_minutes (Integer) - нормативний час на обслуговування (вивантаження) клієнта;

time_window_start та time_window_end (Time) - межі жорсткого часового вікна доставки;

status (Enum) - поточний стан: PENDING, ASSIGNED, IN_TRANSIT, COMPLETED, FAILED.

4. Відношення routes та route_steps (Маршрути та їхні кроки)

Ці таблиці пов'язані відношенням один до багатьох і зберігають результати роботи алгоритму оптимізації.

routes.id (Integer, Primary Key) - ідентифікатор згенерованого рейсу;

routes.vehicle_id (Integer, Foreign Key) - посилання на автомобіль, що виконує рейс;

routes.scheduled_date (DateTime) - запланована дата виконання;

routes.total_distance_km та routes.total_duration_minutes (Float) - агреговані метрики маршруту;

routes.total_cost (Float) - загальна фінансова вартість виконання рейсу;

routes.geometry (Geometry('LINESTRING', 4326)) - закодована полілінія всього шляху для відображення на карті;

route_steps.id (Integer, Primary Key) - унікальний номер кроку;

route_steps.order_id (Integer, Foreign Key) - посилання на конкретне замовлення з таблиці orders;

route_steps.sequence_order (Integer) - порядковий номер зупинки в рейсі (0, 1, 2...);

route_steps.step_type (String) - тип кроку: depot_start, delivery або depot_end;

route_steps.eta (DateTime) - розрахунковий час прибуття автомобіля (Estimated Time of Arrival).

5. Відношення vehicle_positions (Телеметрія автомобілів)

Таблиця призначена для збору GPS-даних від мобільних додатків водіїв у реальному часі.

id (Integer, Primary Key) - унікальний ідентифікатор запису;

vehicle_id (Integer, Foreign Key) - посилання на автомобіль, що відправив координати;

location (Geometry('POINT', 4326)) - точні поточні геопросторові координати;

timestamp (DateTime) - точний час фіксації телеметрії;

speed_kmh (Float) - поточна швидкість руху автомобіля;

heading (Float) - напрямок руху у градусах (0-360) для коректного позиціювання маркера на карті.

3.5 Специфікація програмних інтерфейсів

Взаємодія між клієнтськими додатками (фронтендом) та серверною частиною здійснюється через HTTP-протокол за архітектурою REST. Кожен ендпоінт приймає та повертає дані у форматі JSON. Розроблений API можна логічно розділити на мікросервіси авторизації, маршрутизації та аналітики.

1. Модуль авторизації та безпеки (Auth API)

Ендпоінт: POST /api/v1/auth/login

Призначення: Автентифікація користувача в системі.

Тіло запиту: Об'єкт LoginRequest, що містить поля email та password.

Відповідь системи (200 OK): Повертає об'єкт Token із згенерованим access_token та типом bearer. Термін дії токена визначається налаштуваннями ACCESS_TOKEN_EXPIRE_MINUTES.

Обробка помилок: У разі невірного пароля система повертає 401 Unauthorized.

Ендпоінт: GET /api/v1/auth/me

Призначення: Отримання інформації про поточну сесію. Вимагає валідного JWT-токена в заголовку Authorization.

2. Модуль оптимізації та планування (Optimization API)

Ендпоінт: POST /api/v1/optimization/run

Призначення: Запуск асинхронного фонового процесу генерації маршрутів для всіх замовлень у статусі PENDING.

Тіло запиту: Об'єкт OptimizationRequest, що містить scheduled_date (дату рейсу), масив vehicle_ids та масив цільових order_ids.

Ендпоінт не блокує потік очікуванням розрахунків. Він передає задачу до BackgroundTasks, де створюється окрема сесія СУБД AsyncSessionLocal, ініціюється виклик до OSRM для отримання матриць, після чого алгоритм Google OR-Tools шукає оптимальний розв'язок.

Відповідь системи (202 Accepted): Повертає статус {"status": "accepted", "message": "Optimization started in background"}.

Ендпоінт: POST /api/v1/optimization/run-sync

Призначення: Синхронний запуск оптимізації (переважно для режиму тестування та дебагу). Сервер утримує з'єднання відкритим, поки математичний модуль повністю не сформує маршрути.

Відповідь системи (200 OK): Повертає масив створених маршрутів із прив'язаними автомобілями та детальними кроками (route_steps).

Ендпоінт: POST /api/v1/optimization/reoptimize

Призначення: Тригер динамічної реоптимізації на основі поточних GPS-координат водіїв. Алгоритм бере за стартову точку депо локацію «Наступної зупинки» транспортного засобу.

3. Аналітичний модуль (Analytics API)

Ендпоінт: GET /api/v1/optimization/stats

Призначення: Розрахунок показників логістичної та фінансової ефективності в реальному часі.

Алгоритм проходить циклом по всіх активних маршрутах. Для порівняння (baseline) генерується радіальний сценарій: обчислюється відстань за допомогою функції geodesic (від депо до клієнта і назад помножена на 2). Далі обчислюється

час у дорозі, виходячи з середньої швидкості 30 км/год та 15-хвилинного сервісного часу на точку.

Відповідь системи (200 OK): Повертає складний JSON-об'єкт, що містить вузли optimized (реальні показники системи), baseline (показники без оптимізації) та savings (абсолютну економію в кілометрах і грошах, а також відносні відсотки percent_distance та percent_cost).

3.6 Проєктування інтерфейсу користувача

Інтерфейс користувача розроблювальної системи спроєктований з урахуванням специфіки роботи логістичних компаній, де важлива швидкість сприйняття інформації та зручність управління. Система містить два основних клієнтських дод атки: додаток диспетчера та водія.

Додаток диспетчера реалізований на базі бібліотеки React. Основною концепцією дизайну було обрану сучасну темну тему. Це зроблено для ефективної праці через менше зорове навантаження.

Ключові сторінки інтерфейсу диспетчера:

- Замовлення. Центральна панель для управління замовленнями . Підтримує масовий імпорт даних за допомогою CSV-файлів. Диспетчер може власноруч призначати замовленню конкретному водію. Це спонукає систему знову перераховувати маршрут.
- Маршрути. Головна робоча область візуалізації результату. Завдяки бібліотеці MapLibre GL відображається точна геометрія маршрутів. Вона отримана від OSRM. Диспетчер бачить послідовність точок відвідування та місцезнаходження автомобіля в реальному часі.
- Автопарк. Модуль для управління транспортними засобами та водіїв, де налаштовуються основні параметри - робочі години, вантажопідйомність та амортизація.
- Оптимізація. Панель для запуску переоптимізації автопарку та перегляду ефективності побудованих планів.

Мобільна частина системи розроблена як PWA (Progressive Web App). Такий підхід дозволяє розробити додаток, який поєднує доступність веб-технологій із можливостями нативного програмного забезпечення:

- Offline support. Водій має доступ до свого маршрутного листа навіть за умов нестабільності зв’язку.
- GPS-трекінг. Координати водія регулярно передаються на бекенд, що дозволяє диспетчеру бачити актуальний стан перевезень
- Управління статусами. Водій відмічає кожну точку статусом «виконано», що автоматично перераховує залишок дистанції та часу прибуття.

Інтерфейс водія максимально адаптовано для використання «на ходу»: великі кнопки, контрастні шрифти та пряма інтеграція з картографічними сервісами.

Висновки до розділу

У третьому розділі було виконано практичну реалізацію інформаціної системи. За результатами виконаної роботи розроблено надійну мікросервісну архітектуру. Застовування методу контейнеризації Docker та Docker Compose дозволило розділити систему на незалежні модулі. Це гарантує відсутність кофліктів і можливість розгортання системи на практично будь-якому сервері. Розроблений алгоритм взаємодії з локальним OSRM дає змогу системі одразу розраховувати матриці відстаней на основі реальних доріг.

Спроєктовано клієнтські додатки. Створено два спеціалізованих інтерфейси - веб-панель диспетчера та мобільний додаток водія. Панель створена для контролю автопарку на реоптимізації маршрутів, а мобільний клієнт підтримує офлайн-роботу, збір геопозиції та оновлення статусів доставки.

Результатами цього розділу є успішне перетворення розроблених раніше моделей та алгоритмів у повноцінний програмний продукт. Розроблена система повністю відповідає поставленому технічному завданню та є готовою до використання у комерційний та власних цілях.

4. АПРОБАЦІЯ РОБОТИ СИСТЕМИ ТА АНАЛІЗ ФУНКЦІОНАЛЬНОЇ РОБОТОЗДАТНОСТІ

4.1 Організація робочого середовища та умови експлуатації

Для забезпечення безперебійної роботи системи визначено мінімальні системні вимоги до апаратного забезпечення користувачів. Так як Google OR-Tools та просторова база даних функціонують на віддаленому сервері, вимоги до клієнтських пристроїв є мінімальними.

Робоче місце диспетчера – персональний комп'ютер або ноутбук із процесором частотою від 1.6 ГГц, від 4 ГБ оперативної пам'яті та сучасним веб-браузером (Google Chrome, Mozilla Firefox, Safari). Підключення до мережі Інтернет зі швидкістю не менше 5 Мбіт/с для плавного завантаження інтерактивних карт MapLibre GL.

Термінал водія – смартфон під управлінням операційної системи Android 10+ або iOS 14+ з підтримкою технології PWA (Progressive Web App), вбудованим апаратним GPS-модулем для збору телеметрії та доступом до мобільного інтернету (3G/4G/LTE).

4.2 Перевірка роботи веб-інтерфейсу диспетчера

Функціональне тестування веб-додатка диспетчера розпочинається з етапу авторизації.

Користувач вводить свої облікові дані (електронну пошту та пароль), після чого система перевіряє права доступу та перенаправляє його на головну інформаційну панель (Dashboard).

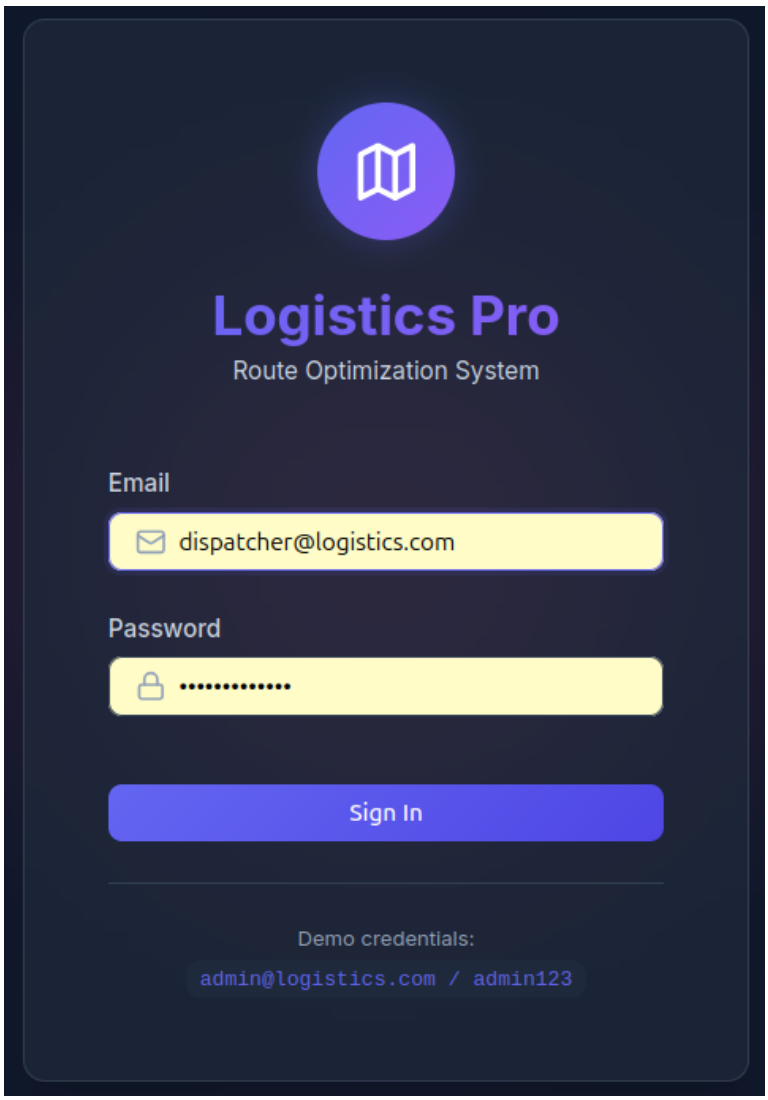


Рисунок 4.1 - Вікно авторизації

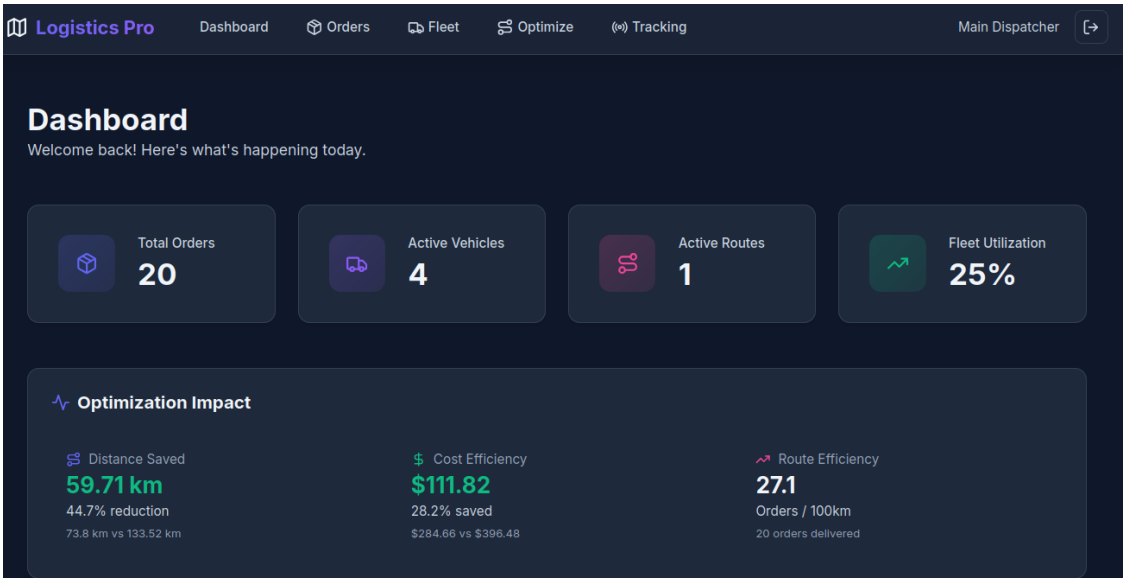


Рисунок 4.2 - Головна панель диспетчера

Основний сценарій роботи диспетчера передбачає управління масивом замовлень. Під час випробовувань було успішно протестовано функцію масового завантаження замовлень.

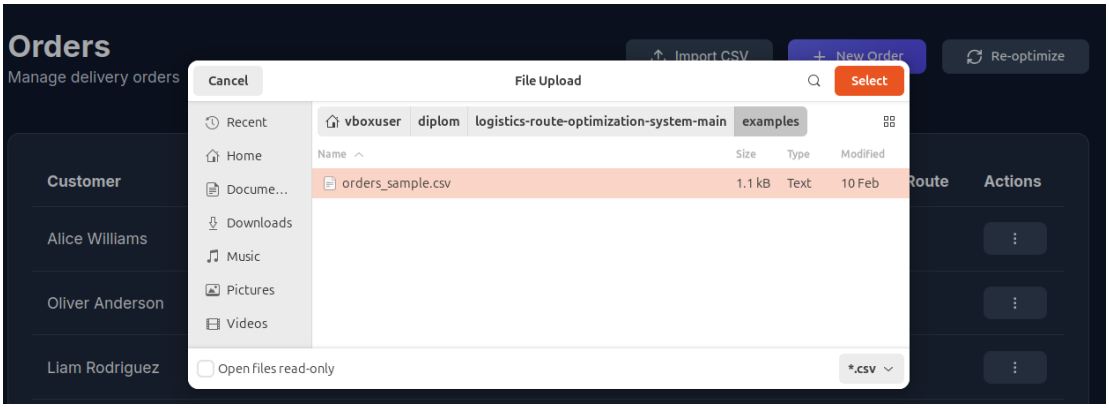


Рисунок 4.3 - Вікно масового імпорту замовлень за допомогою CSV-файлу

Диспетчер завантажує CSV-файл, що містить імена клієнтів, текстові адреси, вагу вантажу та часові вікна доставки. Фронтенд-додаток валідує файл і відправляє його на бекенд, де відбувається автоматичне геокодування адрес і збереження точок у базу даних зі статусом PENDING.

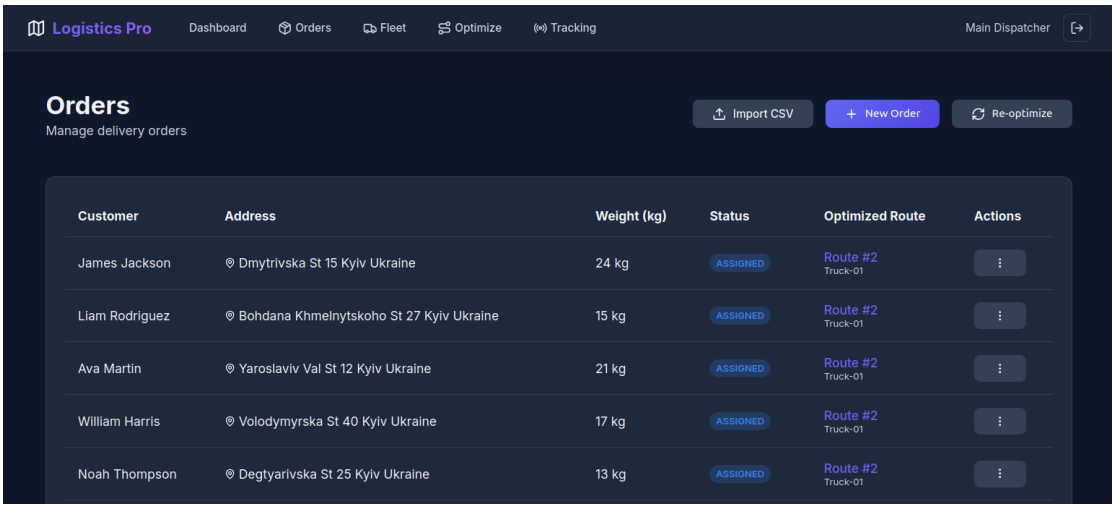


Рисунок 4.4 - Головна робоча панель диспетчера з переліком активних замовлень

Ключовим елементом інтерфейсу є модуль "Маршрути" (Routes). Після завантаження замовлень диспетчер обирає доступні транспортні засоби з бази автопарку та натискає кнопку ініціалізації розрахунку.

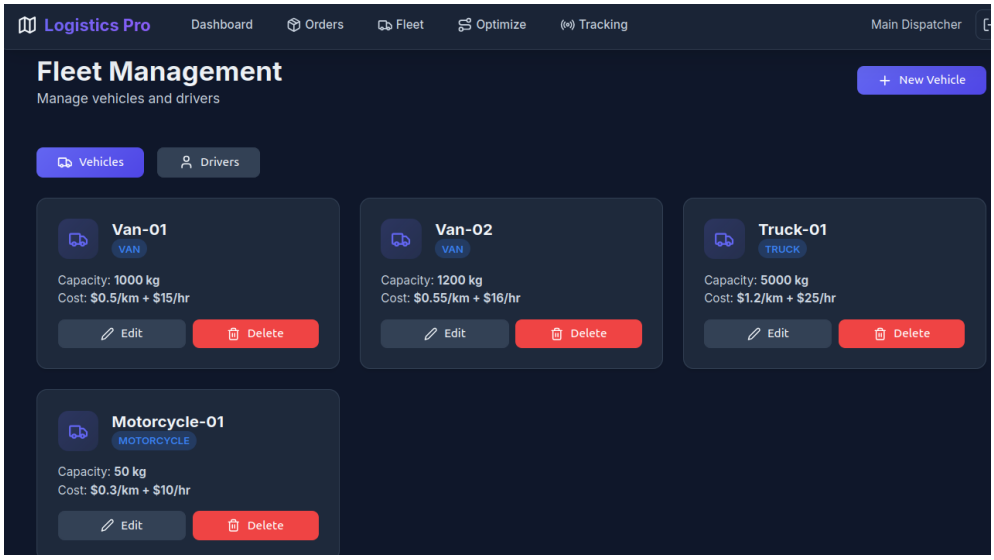


Рисунок 4.5 - Вікно управління транспортом і персоналом

Інтерфейс переходить у стан очікування, поки фоновий процес на сервері виконує оптимізацію за допомогою OSRM та OR-Tools.

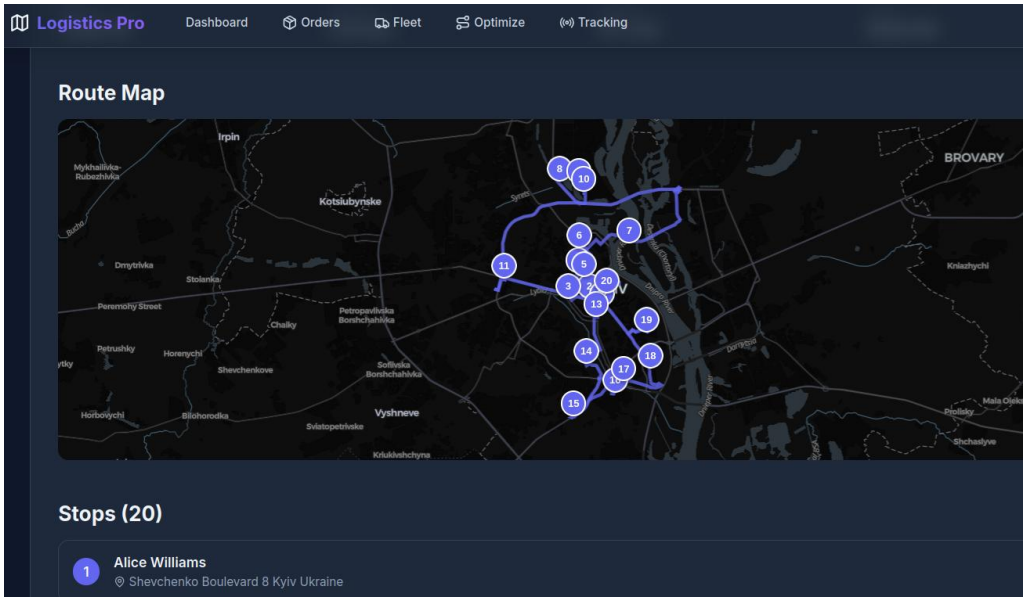


Рисунок 4.6 - Візуалізація згенерованих логістичних маршрутів на інтерактивній карті

4.3 Перевірка роботоздатності мобільного додатку водія

Оскільки додаток реалізовано за технологією PWA, він не потребує встановлення через класичні магазини (Google Play чи App Store). Водій просто додає ярлик системи на головний екран смартфона з браузера.

Після авторизації водій отримує доступ до свого індивідуального маршрутного листа, згенерованого диспетчером. Інтерфейс оптимізовано для мобільних пристроїв - маршрут відображається у вигляді послідовного списку зупинок.

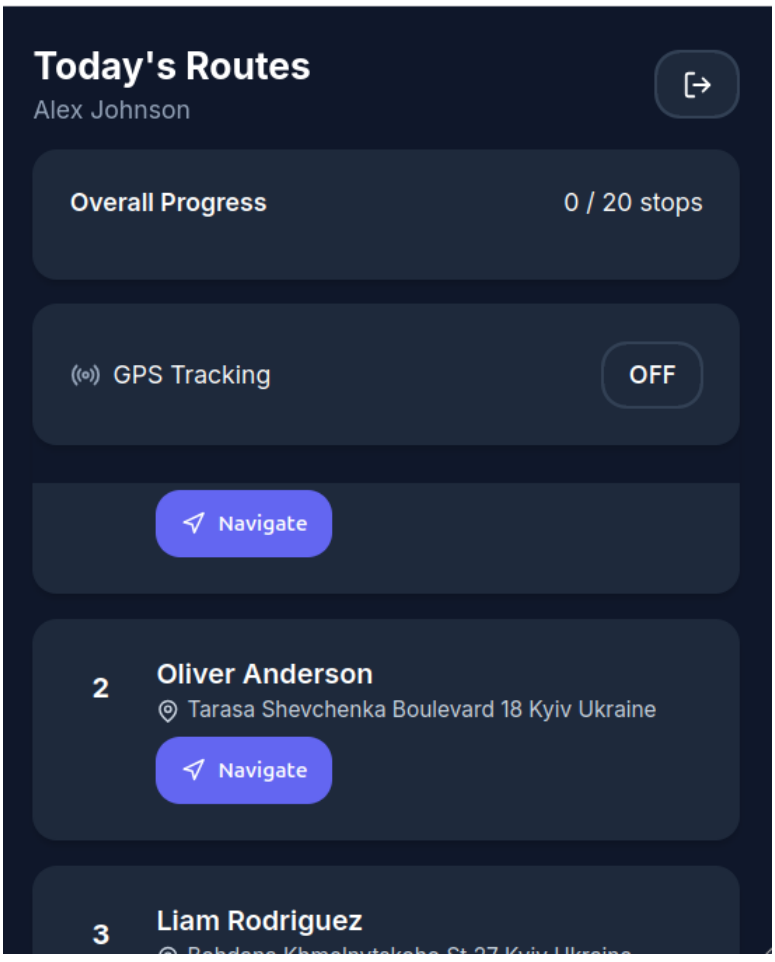


Рисунок 4.7 - Маршрутний лист водія

У ході випробувань було протестовано такі функціональні можливості:

При натисканні на кнопку "Прокласти маршрут" додаток успішно використовує механізм Deep Linking для передачі точних координат клієнта у штатний картографічний додаток смартфона.

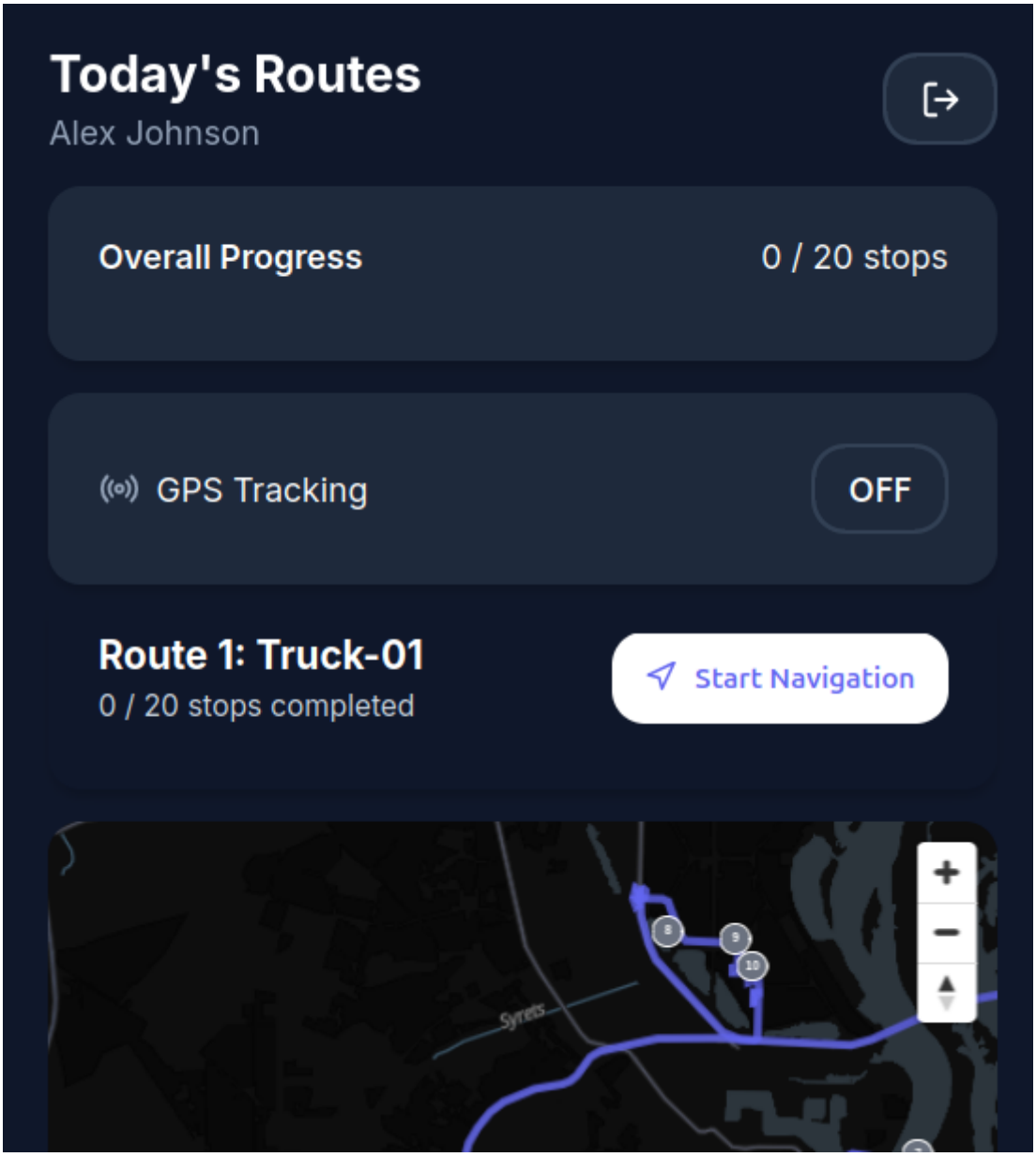


Рисунок 4.8 - Інтерфейс водія з кнопкою навігації

Після прибуття на точку водій натискає кнопку "Виконано". Ця дія миттєво відправляє HTTP-запит на сервер, змінюючи статус замовлення в базі даних.

Система успішно запитує дозволи на використання геолокації пристрою (HTML5 Geolocation API) і здатна періодично передавати координати автомобіля на бекенд для відображення реальної позиції на карті диспетчера.

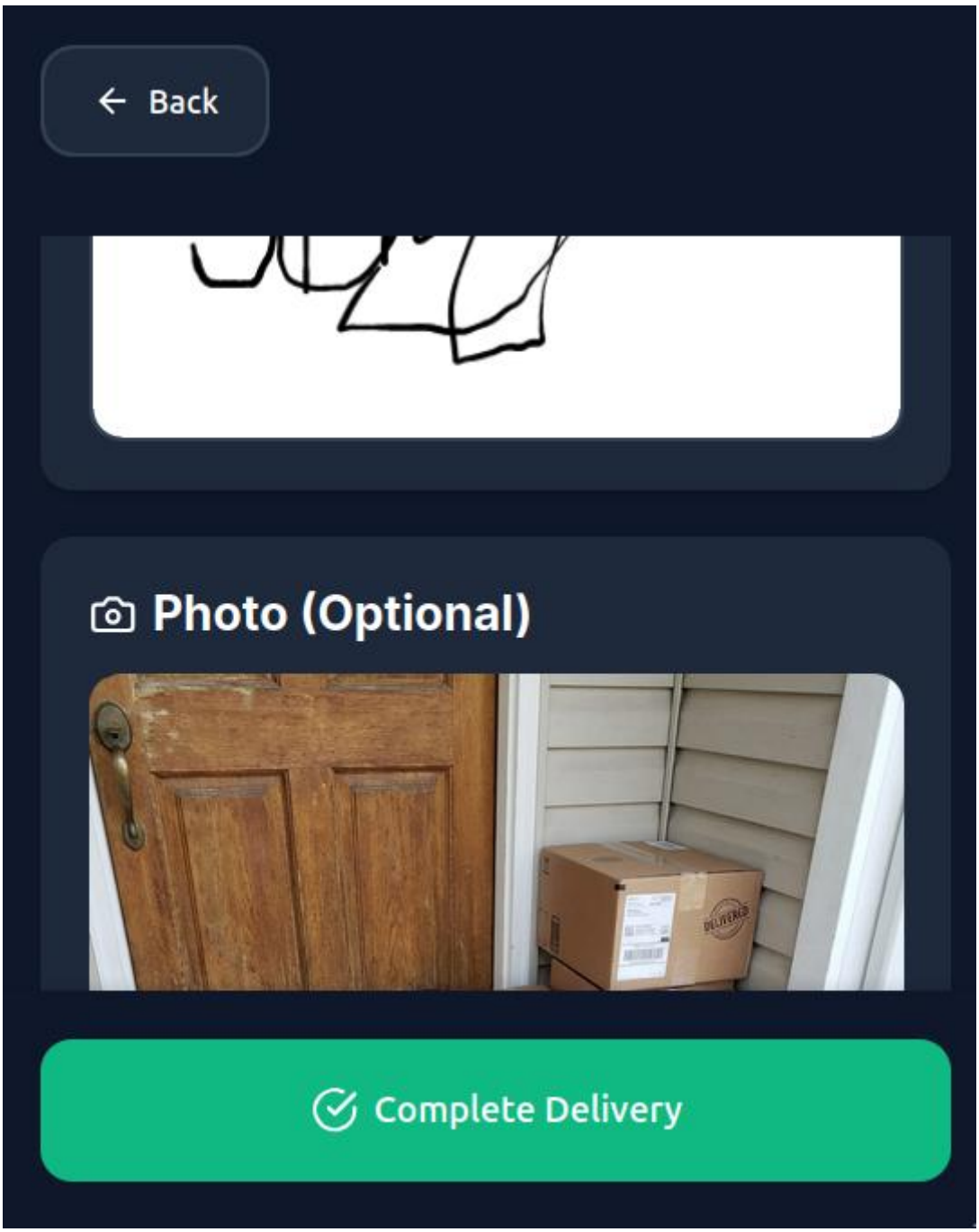


Рисунок 4.9 - Інтерфейс мобільного додатка водія (управління статусами)

4.4 Тестування життєвого циклу логістичного процесу

Для остаточного підтвердження працездатності розробленого програмного забезпечення було проведено комплексне тестування. Воно імітує повний робочий день логістичної компанії.

Послідовність успішно пройденого наскрізного сценарію:

1. Ініціалізація: Диспетчер імпортує 20 нових замовлень. Перевіряє список доступних автомобілів та водіїв.
2. Глобальне планування: Диспетчер ініціює оптимізацію. Алгоритм розбиває 20 точок на маршрути, закріплюючи їх за водіями. Статус замовлень змінюється на ASSIGNED.
3. Виконання: Водій відкриває свій додаток, бачить список замовлень і починає доставку.
4. Реоптимізація: Під час виконання маршруту клієнт скасовує замовлення. Диспетчер в інтерфейсі вручну видаляє цю точку з черги та натискає кнопку реоптимізації. Система перераховує маршрут для водія. Практично миттєво послідовність точок буде оновлено згідно нових вимог.
5. Завершення: Водій послідовно розвозить всі замовлення і відзначає їх як «COMPLETED». Диспетчер може відслідковувати виконані замовлення.

Тестування показало, що вся система функціонує злагоджено та забезпечує цілісність даних на всіх етапах життєвого циклу замовлення.

Висновки до розділу

У четвертому розділі було проведено тестування інформаційної системи та перевірено її працездатність на рівні клієнтських інтерфейсів.

Показано, що розроблений веб-додаток диспетчера забезпечує зручне та безвідмовне управління логістичними процесами. Підтримує масове завантаження даних та забезпечує швидку відмальовку оптимізованих маршрутів на електронній карті. Мобільний додаток водія успішно виконує функції PWA-додатка. Додаток забезпечує двосторонню синхронізацію статусів доставки та збір GPS-телеметрії. Наскрізне тестування життєвого циклу замовлення, включно зі сценарієм динамічної реоптимізації маршруту в реальному часі, підтвердило повну функціональну готовність програмного забезпечення до практичного використання у бізнес-середовищі.

ВИСНОВКИ

У результаті виконання дипломного проєкту було успішно розроблено інформаційну систему оптимізації логістичних маршрутів. Створений програмний продукт дозволяє повністю автоматизувати процес диспетчеризації. Він спрощує планування об'їзду точок та моніторингу доставки вантажів у режимі реального часу.

Відповідно до поставленого технічного завдання та мети роботи, було досягнуто наступних результатів:

- Проведено системний аналіз та математичне моделювання предметної області.
- Досліджено сучасні підходи до вирішення задачі маршрутизації транспортних засобів. Логістичну мережу формалізовано у вигляді зваженого орієнтованого графа. Створена структурна модель дозволила алгоритмічно врахувати ключові бізнес-обмеження - вантажопідйомність автомобілів, жорсткі часові вікна обслуговування клієнтів та доступність автопарку.
- Для знаходження оптимальних маршрутів інтегровано промислову бібліотеку Google OR-Tools. Використано гібридний підхід з двох етапів. На першому етапі конструктивна евристика миттєво генерує базовий допустимий розв'язок, а на другому - алгоритм керованого локального пошуку ітеративно покращує його, мінімізуючи загальний кілометраж. Для отримання високоточних дистанцій налаштовано інтеграцію з локальним геосервером OSRM (Open Source Routing Machine). Це забезпечило побудову матриць відстаней з урахуванням реальної топології дорожньої мережі.
- Спроєктовано відмовостійку серверну архітектуру (Backend). Бекенд системи реалізовано мовою Python з використанням сучасного асинхронного фреймворку FastAPI. Застосування концепції неблокуючого введення-виведення та фонових задач дозволило

забезпечити високу пропускну здатність платформи. Впроваджено надійну підсистему аутентифікації та авторизації на базі стандартів OAuth2 та безстанних JWT-токенів із криптографічним хешуванням паролів алгоритмом bcrypt.

- Реалізовано надійну дата-логічну модель для роботи з просторовими даними. Як основне сховище даних розгорнуто реляційну СУБД PostgreSQL. Завдяки використанню спеціалізованого просторового розширення PostGIS реалізовано нативне збереження географічних координат та швидку індексацію маршрутів за допомогою GiST-індексів. Взаємодію бекенду з базою даних організовано через асинхронний драйвер asyncpg та ORM-бібліотеку SQLAlchemy 2.0.

Створено кросплатформені клієнтські інтерфейси (Frontend). Для різних ролей користувачів розроблено ізольовані клієнтські додатки.

Веб-панель диспетчера створена у форматі Single Page Application (SPA) на базі бібліотеки React. Інтерфейс підтримує масовий імпорт замовлень, інтерактивне візуальне відображення маршрутів за допомогою MapLibre GL та управління станами замовлень.

Мобільний додаток водія розроблено за технологією Progressive Web App (PWA). Цо дозволяє використовувати додаток на смартфонах без встановлення з сторонних ресурсів. Додаток забезпечує доступ до маршрутного листа, збір телеметрії (GPS-координат) через HTML5 Geolocation API та інтеграцію зі штатними навігаторами пристрою.

Експериментальні дослідження підтвердили високу алгоритмічну ефективність та стабільність розробленої системи. Аналіз оптимізаційних показників довів здатність алгоритму зменшувати сумарний кілометраж автопарку. Використання технології контейнеризації Docker дозволило повністю ізолювати всі мікросервіси та забезпечити легке кросплатформене розгортання проєкту.

Розроблена інформаційна система оптимізації логістичних маршрутів є повністю завершеною. Вона є стійкою до навантажень - програмним продуктом

індустріального рівня. Використання сучасного стеку технологій (Python, FastAPI, PostgreSQL/PostGIS, OR-Tools, React, Docker) гарантує легке масштабування системи. Результати роботи мають високу практичну цінність і можуть бути інтегровані в діяльність служб доставки, поштових операторів та e-commerce підприємств для підвищення ефективності використання транспортних ресурсів з метою мінімазіції витрат.

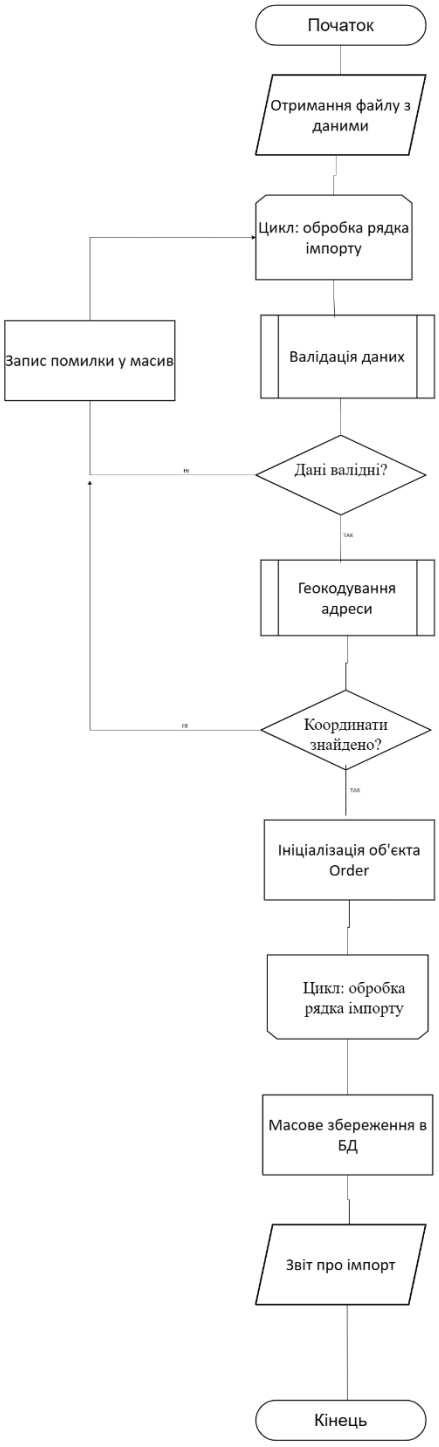
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dantzig G., Ramser J. The Truck Dispatching Problem. Management Science. 1959. Vol. 6, No 1, pp. 80-91.
2. Toth P., Vigo D. Vehicle Routing: Problems, Methods, and Applications. Society for Industrial and Applied Mathematics. 2014. Vol. 1, 463 p.
3. Braysy O., Gendreau M. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. Transportation Science. 2005. Vol. 39, No 1, pp. 104-118.
4. Laporte G. The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research. 1992. Vol. 59, No. 3. P. 345–358.
5. Gendreau M., Potvin J.-Y. Handbook of Metaheuristics. 3rd ed. Springer, 2019. 855 p.
6. Google Developers. Vehicle Routing Problem | OR-Tools. URL: https://developers.google.com/optimization/routing/vrp
7. Project OSRM. Open Source Routing Machine API Documentation. URL: http://project-osrm.org/docs/v5.24.0/api/
8. Obe R. O., Hsu L. S. PostGIS in Action, Third Edition. Manning Publications, 2021. 600 p.
9. PostgreSQL Global Development Group. PostgreSQL 15 Documentation. URL: https://www.postgresql.org/docs/15/
10. Ramírez S. FastAPI Documentation. URL: https://fastapi.tiangolo.com/
11. Masse M. REST API Design Rulebook. O'Reilly Media, 2011. 118 p.
12. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). Internet Engineering Task Force (IETF), RFC 7519, 2015. URL: https://datatracker.ietf.org/doc/html/rfc7519
13. Hardt D. The OAuth 2.0 Authorization Framework. Internet Engineering Task Force (IETF), RFC 6749, 2012. URL: https://datatracker.ietf.org/doc/html/rfc6749

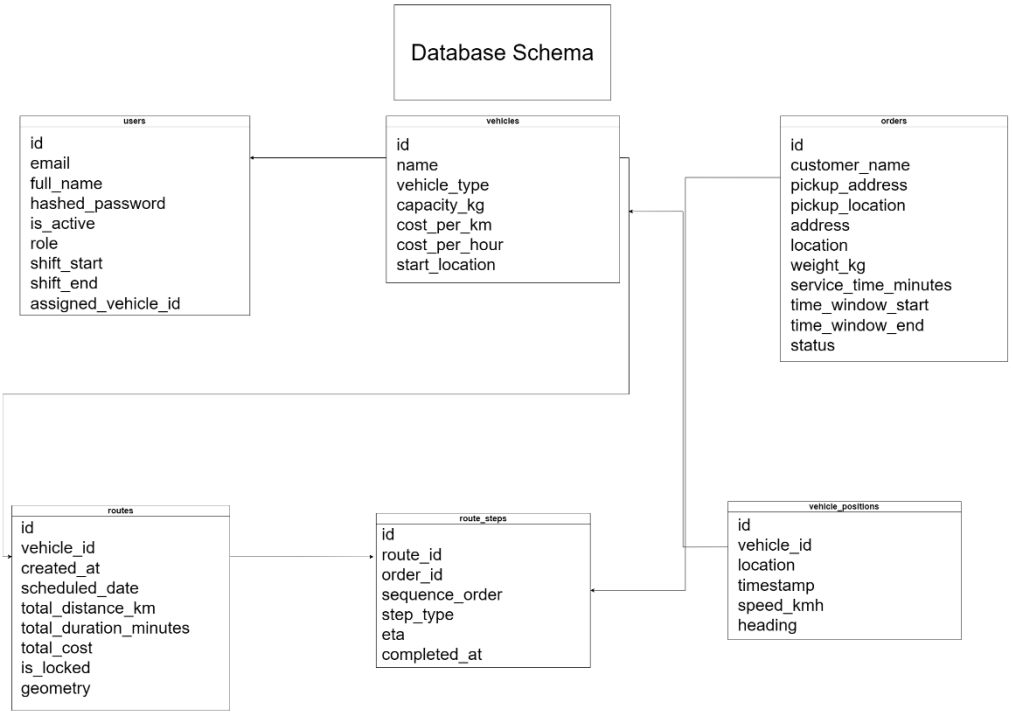
14. Docker Inc. Docker Documentation. URL: https://docs.docker.com/
15. Meta Platforms, Inc. React – A JavaScript library for building user interfaces. URL: https://react.dev/

ДОДАТКИ

ДОДАТОК 1
Копії графічних матеріалів



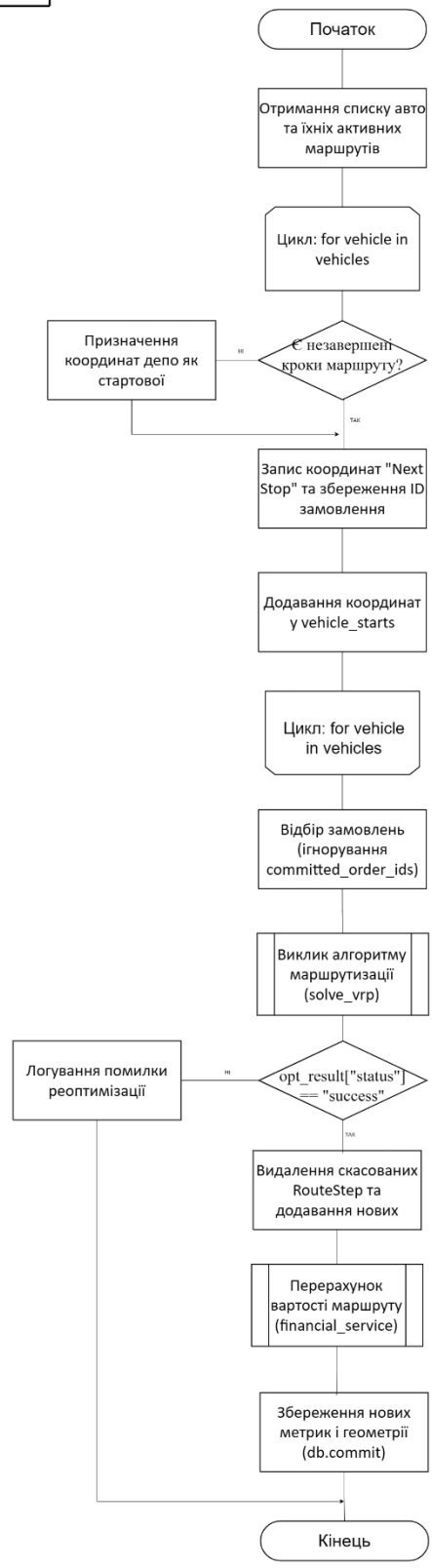
Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045430.001 Д1		
Розробив		Ляшина			Масовий імпорт даних. Схема алгоритму	Літ.	Аркушів 1
Перевірив		Рагімов				Аркуш 1	
Н. контр.		Клочкова				КПІ ім. І. Сікорського ФІПМ, КВ-21	
Затв.		Романкевич					



Змн.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045430.002 Д2		
Розробив		Петренко			Схема бази даних системи	Літ.	Аркуш	Аркушів
Перевірив		Рогожин					1	1
Н. контр.		Ключенко				КПІ ім. І. Сікорського ФІОМ, КВ-21		
Затв.		підпис						

ІАЛЦ.467200.003 Д3

						ІАЛЦ.045430.003 Д3	
Змін.	Арк.	№ докум.	Підпис	Дата			
Розробив		Петренко			Алгоритм оптимізації	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Радченко				КПІ ім. І. Сікорського ФІПМ, КВ-21	
Н. контр.		Костенко					
Затв.		Романенко					



Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045430.004 Д4
Розробив		Фурман			Алгоритм реоптимізації.
Перевірив		Рагулін			Літ. / Аркуш 1 / Аркушів 1
Н. контр.		Калганов			КПІ ім. І. Сікорського ФІМ, КВ-21
Затв.		Романенко			