Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Системотехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Дослідження підходів до проєктування систем організації перевезень попутним транспортом
(тема)

Виконав: студент 2 курсу, групи СПРм-22-1
Варданян К.А.
(прізвище, ініціали)

Спеціальність 122 – Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова програма
Системне проектування
(повна назва освітньої програми)

Керівник професор Міщеряков Ю.В.
(посада, прізвище,ініціали)

Допускається до захисту

Зав. кафедри _____
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Системотехніки

Рівень вищої освіти другий(магістерський)

Спеціальність 122 – Комп'ютерні науки
(код і повна назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___»_____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІНУ РОБОТУ РОБОТУ

студентові Варданян Карен Артуровичу
(прізвище,ім'я, по батькові)

1. Тема роботи Дослідження підходів до проєктування систем організації перевезень попутним транспортом
затверджена наказом по університету від 01.04 2024 р. № 259СТ

2.Термін подання студентом роботи до екзаменаційної комісії 22.06. 2024р

3.Вихідні дані до роботи: Проаналізувати існуючі підходи до проєктування систем організації перевезень попутним транспортом. Визначити ключові фактори, які впливають на ефективність та успішність таких систем. Розробити рекомендації щодо вдосконалення та оптимізації систем організації перевезень попутним транспортом. Запропонувати прототип системи організації перевезень попутним транспортом, адаптованої до українського ринку. Визначити ключові фактори, які впливають на ефективність та успішність таких систем.

4.Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ. 4.2 Аналіз предметної області. 4.2.1 Огляд і аналіз сучасного стану проблеми. 4.2.2 Аналіз предметної галузі, що визначає специфіки використання сервісу. 4.2.3 Визначення сфери застосування системи. 4.2.4 Огляд існуючих систем. 4.2.5 Постановка задачі. 4.3 Розробка вимог до системи. 4.3.1 Розробка системних вимог. 4.3.2 Обґрунтування вибору напрямку досліджень. 4.3.3 Розробка функціональних вимог. 4.3.4 Аналіз підходів до проектування систем організації перевезень попутним транспортом. 4.3.5 Розробка діаграми прецедентів UseCase. 4.4 Опис прийнятих

проектних рішень. 4.4.1 Обґрунтування вибору мови програмування. 4.4.1.1 Основні критерії вибору мови програмування. 4.4.2 Визначення алгоритмів пошуку попутників. 4.4.3 Результати дослідження алгоритмів пошуку співпадаючих маршрутів. 4.4.4 Обґрунтування вибору API. 4.4.4.1 Чому варто вибрати Google Maps Directions API. 4.4.5 Обґрунтування вибору СУБД. 4.4.6 Логічне та фізичне моделювання даних системи. 4.4.6.1 Вирішення питань високого навантаження бази даних. 4.4.6.2 Загальна архітектура. 4.4.6.3 Переваги такої системи. 4.4.6.4 Недоліки такої системи. 4.4.6.5 Переваги такої архітектури. 4.4.6.6 Недоліки такої архітектури. 4.4.7 Інтеграція з зовнішніми системами. 4.4.8 Обґрунтування вибраної архітектури. 4.4.9 Архітектурна схема мультитенатного сервісу пошуку автомобільних попутників. 4.4.10 Вплив на майбутнє розширення та підтримку системи. 4.4.11 Рекомендації для подальшого розвитку проекту. 4.5 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА. 4.5.1 Тестування сервісу пошуку автомобільних попутників. 4.6 Висновки.

5.Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 5.1 Схеми алгоритмів пошуку попутників, 5.2 Схема мультитенатної системи пошуку попутників, 5.3 Фізична модель бази даних, логічна модель бази даних, діагарми розгортання.

6.Консультанти розділів роботи(п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Аналіз предметної області			
Опис прийнятих проектних рішень			

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів	Приміт ка
1.	*Отримання завдання атестаційної роботи*	*21.03.24*	
2.	*Аналіз предметної області*	*26.03 — 15.04.24*	
3.	*Аналіз існуючих систем*	*17.04 — 21.04.24*	
4.	*Вибір архітектури системи*	*2.04 – 09.05.24*	
5.	*Вибір середовища розробки для розробки програмного забезпечення*	*10.05 – 18.05 24*	
6.	*Розробка програмного забезпечення*	*18.05 — 27.05.24*	
7.	*Реалізація серверної частини системи*	*28.05 — 30.05.24*	

8.	*Тестування програмного забезпечення*	*01.07 — 12.06.24*	
9.	*Оформлення пояснювальної записки*	*12.06 — 14.06.24*	
10.	*Оформлення презентації*	*15.06.22- 16.06.2024*	
11.	*Представлення на рецензування*	*17.06.2024*	
12.	*Представлення атестаційної роботи в ЕК*	*22.06.2024*	

Дата видачі завдання _____ 20___ р.

Студент  _____
(підпис)

Керівник роботи _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка звіту з практики: 102 с., 4 ч., 5 табл., 34 рис., 2дод., 26 джерел.

БАЗА ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ, MSSQL, КОРИСТУВАЛЬНИЦЬКИЙ ІНТЕРФЕЙС, СИСТЕМА ОРГАНІЗАЦІЇ ПЕРЕВЕЗЕНЬ.

Головним предметом дослідження є аналіз та оцінка різних підходів до проєктування систем організації перевезень попутним транспортом. Це включає вивчення технологічних, бізнесових, юридичних, психологічних, екологічних та соціальних аспектів, що впливають на розробку та впровадження таких систем. Дослідження спрямоване на розуміння вимог та потреб користувачів, використання сучасних інформаційних технологій для ефективного управління перевезеннями, а також розробку рекомендацій для оптимального використання ресурсів та покращення якості послуг.

Об'єктом дослідження є процес проєктування та розробки систем організації перевезень попутним транспортом. Дослідження включає аналіз різних підходів до створення таких систем, враховуючи технологічні, бізнесові, юридичні, соціальні та екологічні аспекти. Об'єкт дослідження також охоплює вивчення потреб користувачів, оптимальних стратегій маршрутизації, ефективних методів співпраці між водіями та пасажирами, а також розробку інноваційних рішень для підвищення якості та ефективності системи організації попутних перевезень.

Мета досліджень: пошук оптимальних підходів до проєктування систем організації перевезень попутним транспортом. Оптимальність полягає у тому що система має бути ефективною у використанні ресурсів, таких як час, гроші а також вона повинна бути стабільною та надійною у будь-яких умовах експлуатації.

Під час виконання роботи було спроектовано та реалізовано серверну частину системи з організації перевезень попутним транспортом та пошуку автомобільних попутників.

Було використано моделювальний підхід для розробки системи організації перевезень попутним транспортом, що включає використання комп'ютерних моделей, які дозволять проаналізувати різні аспекти такої системи. Це включає моделювання користувачів, транспортних засобів, маршрутів, економічної ефективності, трафіку та навантаження на дороги.

ABSTRACT

Thesis contains: 102 pages, 34 images, 5 tables, 2 applications, *26* sources. Graphic part of the thesis contains __ posters.

DATABASE, DATABASE MANAGEMENT SYSTEM (DBMS), MSSQL, USER INTERFACE, TRANSPORTATION ORGANIZATION SYSTEM.

The main subject of the research is the analysis and evaluation of various approaches to designing systems for organizing ridesharing transportation. This includes studying technological, business, legal, psychological, environmental, and social aspects that influence the development and implementation of such systems. The research aims to understand user requirements and needs, utilize modern information technologies for efficient transportation management, and develop recommendations for optimal resource utilization and service improvement.

The object of the research is the process of designing and developing systems for organizing ridesharing transportation. The research encompasses analyzing various approaches to creating such systems, considering technological, business, legal, social, and environmental aspects. The research object also involves studying user needs, optimal routing strategies, effective collaboration methods between drivers and passengers, and developing innovative solutions to enhance the quality and efficiency of ridesharing organization systems.

Research goal: To search for optimal approaches to designing systems for organizing ridesharing transportation.

During the study, the server-side of the ridesharing transportation organization system and carpooling search system were designed and implemented. A modeling approach was used to develop the ridesharing transportation organization system, involving the utilization of computer models to analyze various aspects of such a system. This includes modeling users, transportation vehicles, routes, economic efficiency, traffic, and roadloads.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

URL – uniform resource locator.
БД – база даних
ІС – інформаційна система
СУБД – система керування базами даних
ER – Entity-Relation
ID – identifier
IDEF0 – нотація опису бізнес-процесів
IDEF1X – методологія розробки моделей даних
API – application programming interface
KNN – k Nearest Neighbor
ОС – операційн система

ЗМІСТ

ВСТУП

За останні роки доступ до Інтернету став масовим, що робить інформаційні системи невід'ємною частиною повсякденного життя для багатьох людей. Це означає, що користувачі мають можливість швидко та зручно отримувати доступ до різноманітної інформації та використовувати різноманітні сервіси, які полегшують їхнє життя. Наприклад, вони можуть швидко знаходити відповіді на свої питання, замовляти товари та послуги онлайн, спілкуватися з іншими людьми, та багато іншого. Це дозволяє економити час і зусилля, роблячи роботу більш ефективною та зручною для користувачів.

У розробці інформаційної системи виконується автоматизація різних бізнес-процесів з метою зробити використання системи зручним, ефективним та швидким для користувача. Для створення такої системи можуть застосовуватися різні підходи надання послуг, один з яких - Restful підхід.

Використання архітектури Rest у програмному забезпеченні дозволяє уникнути зайвих внутрішніх прошарків. Кожна одиниця інформації однозначно визначається унікальним ідентифікатором, що називається URL адресою, і має строго визначений формат. Використання Rest архітектури значно спрощує підтримку вже існуючих або автоматизованих бізнес-процесів, оскільки ключові процеси описуються в рамках одного блоку, який відповідає за роботу певної частини сервісу і може бути повторно використаний або модифікований без впливу на решту системи.

Веб-системи, побудовані на цій архітектурі, легкі у використанні і доступні для будь-якого користувача в повсякденному житті. Вони не вимагають встановлення додаткового програмного забезпечення і не надмірно навантажують ресурси пристроїв. Це робить онлайн сервіси дуже популярними, оскільки всі обчислювальні операції відбуваються на сервері, а користувачам лише потрібен доступ до Інтернету.

Сервіс з пошуку попутників базується на веб-технологіях і використовує принципи RESTful архітектури для забезпечення ефективної комунікації між клієнтами та сервером. Використання REST дозволяє створювати API, які забезпечують доступ до ресурсів за допомогою стандартних HTTP-методів, таких як GET, POST, PUT та DELETE. Це дозволяє зручно і ефективно обмінюватися даними між клієнтською та серверною стороною системи.

У контексті дослідження підходів до проєктування систем організації перевезень попутним транспортом, важливо дослідити різні методи пошуку попутників та поїздок для користувачів з різними ролями.

У рамках дослідження підходів до проєктування систем організації перевезень попутним транспортом, важливо ретельно проаналізувати різноманітні методи пошуку попутників та поїздок, які забезпечують ефективність та зручність користувачам з різними ролями. Це може включати розгляд алгоритмів пошуку оптимальних маршрутів та співпраці між водіями та пасажирами з урахуванням їхніх індивідуальних потреб і вимог. Також важливо вивчити різні моделі фільтрації та сортування результатів пошуку, які забезпечують користувачам можливість швидкого та зручного вибору оптимальних варіантів поїздок. Такий детальний аналіз різних методів пошуку дозволить визначити оптимальний підхід до проєктування системи організації перевезень попутним транспортом, який задовольнятиме потреби користувачів та забезпечить ефективну роботу сервісу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд і аналіз сучасного стану проблеми

У рамках дослідження підходів до проєктування систем організації перевезень попутним транспортом, першочерговою проблемою є аналіз поточного стану транспортної інфраструктури та визначення потреб користувачів у зручних і ефективних засобах пересування. Це включає дослідження різноманітних методів пошуку та організації поїздок для різних категорій користувачів, таких як власники автомобілів, які шукають попутників, або особи, що потребують транспортування. Розглядаються можливості використання веб-додатків та електронних ресурсів для надання інформації про поїздки та забезпечення можливостей замовлення перевезень.

Додатково, важливо дослідити ефективність різних підходів до організації системи попутних перевезень. Це охоплює вивчення технологічних аспектів, таких як використання RESTful архітектури програмного забезпечення для створення ефективних та легко масштабованих веб-сервісів. Аналізується можливість використання унікальних ідентифікаторів (URL-адрес) для однозначної ідентифікації інформації та впровадження строго визначеного формату даних. Розглянуто також методи оптимізації системи, включаючи моделювання користувачів, транспортних маршрутів, економічної ефективності та навантаження на дороги.

Крім того, проведено аналіз попиту на послуги перевезень попутним транспортом та оцінка їх впливу на транспортну систему міста. Це включає вивчення використання транспортних засобів, розподіл пасажиропотоків та визначення потенційних переваг і обмежень використання системи організації попутних перевезень. Наприклад, аналізується співвідношення між користувачами особистих автомобілів та користувачами громадського транспорту у конкретному місті.

Усе це дозволяє зрозуміти ключові вимоги та потреби користувачів, розробити ефективні та зручні рішення для організації перевезень попутним транспортом та підвищити загальну якість та доступність транспортних послуг для мешканців міста.

Загалом, більшість мешканців великих міст України використовують громадський транспорт на щоденній основі. Середня статистика використання громадського транспорту може бути такою:

– Київ: Понад 1 мільйон пасажирів користуються метро, тролейбусами, трамваями та маршрутками щодня;

– Львів: На кожного мешканця Львова припадає близько 0,6 поїздок громадським транспортом щоденно;

– Одеса: У середньому 0,8-1 мільйона осіб користуються громадським транспортом щодня.

Переваги використання сервісу пошуку попутників для мешканців України включають:

– економія часу і коштів: Подорожування автомобільним попутництвом може бути швидшим та економічнішим у порівнянні з громадським транспортом або таксі, особливо на довгі відстані;

– зручність і комфорт: Подорожування автомобілем з попутником дозволяє користувачам мати більше комфорту та зручності порівняно з громадським транспортом;

– гнучкість в плануванні маршрутів: Сервіс пошуку попутників надає можливість користувачам вибирати оптимальний маршрут і час відправлення, що відповідає їхнім потребам і розкладу;

– сприяння екологічним ініціативам: Спільне використання автомобілів допомагає зменшити викиди шкідливих речовин в атмосферу та сприяє збереженню довкілля.

Додатково, сервіс пошуку попутників може сприяти підвищенню безпеки під час подорожей. Ось деякі аспекти, які стосуються безпеки:

– перевірка користувачів: Багато сервісів пошуку попутників надають можливість перевірки профілів користувачів, включаючи відгуки та рейтинги, що може сприяти вибору надійного попутника;

– поділ витрат: Подорож з попутником може бути безпечнішою, оскільки в разі необхідності можна розділити витрати на пальне і інші витрати, що зменшує ризик непередбачених витрат;

– спільні подорожі: Подорожувати в компанії із попутником може бути безпечніше для людини, що шукає автівку для поїздки, оскільки це зменшує ризик від непередбачених ситуацій через те що громадським транспортом

користується велика кількість людей, а поїздки попутним транспортом є менш масовими;

– екстрені випадки: Під час подорожі з попутником, який вже є знайомим або якому довіряєте, може бути легше і швидше отримати допомогу у випадку екстрених ситуацій.

Отже, сервіс пошуку попутників може бути вигідним для мешканців України з точки зору зручності і ефективності подорожей, а також з точки зору збереження ресурсів та безпеки для пасажира, який шукає автівку.

1.2 Аналіз предметної галузі, що визначає специфіки використання сервісу

Сервіс пошуку попутників у сфері організації перевезень попутним транспортом виступає як посередник між власниками автівок, що шукають попутників, та особами, які потребують транспортування. Його функціональність спеціально розроблена для використання різними категоріями користувачів, кожна з яких має свої унікальні можливості. Посередницька роль сервісу полягає в обмеженні доступу користувачів до певних функцій в залежності від їхньої ролі. Наприклад, лише власники автівок можуть створювати поїздки, тоді як інші користувачі можуть лише приєднуватись до існуючих поїздок.

Законодавство України, зокрема Конституція та Закон про автомобільний транспорт, регулює діяльність сервісу та його користувачів. Ці нормативні акти встановлюють права та обов'язки учасників перевезень, забезпечують безпеку та захист прав користувачів. Державний контроль та регулювання забезпечують виконання встановлених правил і стандартів у сфері автомобільного транспорту, що сприяє якості та безпеці послуг.

Розглянуті нормативні акти не лише визначають права та обов'язки учасників, але й сприяють розвитку автомобільного транспорту, включаючи регулювання персоналу та діяльності автомобільних перевізників. Цей підхід сприяє створенню стабільної та безпечної системи організації перевезень попутним транспортом, що відповідає потребам користувачів та вимогам законодавства.

1.3 Визначення сфери застосування системи

Розроблювана система для організації перевезень попутним транспортом має на меті створення ефективного механізму пошуку автомобільних послуг для різних категорій користувачів. Вона буде використовуватися як посередник між власниками автівок, що пропонують послуги перевезень, та пасажирами, які потребують транспортування. Це означає, що обидва типи користувачів будуть активно взаємодіяти з системою, використовуючи її функціонал.

Власники автівок матимуть можливість створювати поїздки, визначаючи такі параметри, як дата, час відправлення, маршрут та кількість доступних місць у своїх автомобілях. Пасажири, з свого боку, зможуть переглядати доступні поїздки, аналізувати їх параметри та обирати найзручніші для них варіанти для реєстрації. Така взаємодія дозволить забезпечити ефективну координацію між власниками автомобілів і пасажирами.

Крім того, система буде включати в себе адміністративну функціональність, яка дозволить контролювати безпеку та виконання правил сервісу з боку посередників, які можуть виступати в ролі адміністраторів. Це означає, що адміністратори системи будуть відповідальні за розгляд скарг та врегулювання конфліктів, а також за забезпечення безпеки усіх учасників.

У контексті законодавчого середовища України, система повинна дотримуватися відповідних нормативних актів, зокрема Конституції та законів про автомобільний транспорт. Ці закони встановлюють права та обов'язки учасників перевезень, а також забезпечують захист прав користувачів та безпеку їхніх персональних даних.

Одним з основних засобів реалізації системи може стати створення веб-сайту, який забезпечить швидкий, зручний та безпечний пошук попутників. Він дозволить користувачам здійснювати пошук онлайн, надавати необхідну інформацію про поїздки та обробляти великий обсяг даних. Такий підхід також забезпечить високий рівень доступності системи для різних категорій користувачів та зробить процес організації перевезень більш ефективним і зручним.

1.4 Огляд існуючих систем

Зростання швидкості життя та постійний рух у містах створюють потребу у нових, інноваційних підходах до транспортування. Наприклад, із поширенням агрегаторів таксі на ринку з'явилася багатошарова мережа додатків, що діють як посередники між пасажирами та водіями.

Підвищення цін на пальне та послуги таксі призводить до постійного зростання тарифів на перевезення, що зменшує попит на такі послуги. Це спонукає до пошуку більш доступних та економічних альтернатив.

Наразі існують обмежені можливості для переміщення в межах міста. Більшість існуючих систем пошуку попутників спрямовані на міжміські або міжнародні поїздки, залишаючи без уваги потреби у міському транспортуванні. Це породжує потребу в розробці системи спрямованої саме на пошук попутників для міських подорожей.

Враховуючи те, що багато людей щодня подорожують в межах міста, створення такого сервісу є вкрай актуальним. Він не лише дозволить користувачам зручно організовувати свої міські поїздки, заощаджуючи час і гроші, але також стимулюватиме водіїв отримувати додатковий прибуток, пропонуючи свої послуги для спільних поїздок у місті.

На сьогоднішній день в Україні існують різноманітні сервіси, що дозволяють шукати попутників для поїздок:

BlaBlaCar - це один з найвідоміших сервісів попутництва, який дозволяє знаходити попутників для поїздок по Україні та за її межами.

Uklon - хоча цей сервіс в основному спеціалізується на послугах таксі, він також має функціонал пошуку попутників для спільних поїздок.

Waybro - це сервіс, спеціалізований на організації попутництва, який дозволяє знаходити попутників для поїздок по всій країні.

BananaCar - ще один український сервіс, спрямований на забезпечення спільних поїздок, який надає можливість знаходити попутників для подорожей в межах України.

OLX - популярний онлайн-ресурс для розміщення оголошень про різноманітні товари та послуги. Багато користувачів використовують OLX для розміщення індивідуальних об'яв про попутництво.

Ці сервіси надають можливість користувачам знаходити співмандрівників для комфортних та економічних поїздок по Україні.

BlaBlaCar, відомий у всьому світі сервіс попутництва, має лідируюче положення на українському ринку. Його інтуїтивно зрозумілий інтерфейс, доступний як на веб-платформі, так і у мобільному додатку, приваблює звичайних користувачів. Україна має понад 5 мільйонів активних користувачів, які володіють автомобілями або шукають транспорт. Крім того, BlaBlaCar пропонує можливості для перевезення тварин, розширюючи свої функції.

Проте у системі є певні недоліки, які можуть стати перешкодою для користувачів у великих містах. Наприклад, система дозволяє створювати заявки лише для поїздок між містами всередині країни або за її межами, але відсутня можливість знайти попутника для поїздки в межах міста. Крім того, відсутня автоматична геолокація місця початку подорожі, а фільтрація здійснюється обмежено лише за рейтингом користувачів та водіїв.

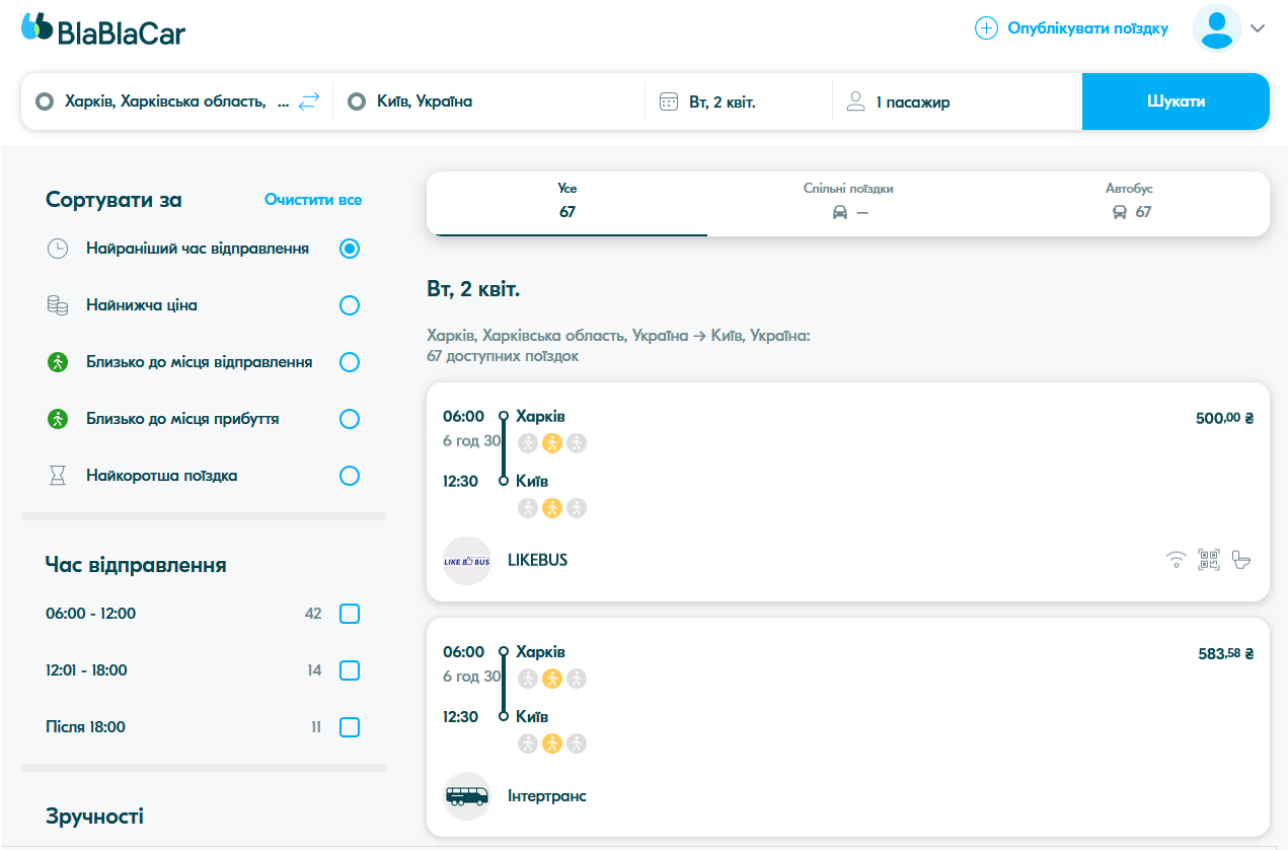


Рисунок 1.1 - Користувацький інтерфейс сайту BlaBlaCar

Поміж інших недоліків слід зазначити відсутність можливості попереднього перегляду коментарів інших користувачів, які вже мали досвід поїздок з певним

водієм. На додаток, володар автомобіля не може визначити деталі щодо багажу або вказати його кількість.

Неспроможність забезпечити негайний та надійний транспорт може виникнути через скасування поїздок водіями у останній момент або їх запізнення, що може спричинити невдоволення та затримки для пасажирів. Невідповідність очікуванням також може виникнути, оскільки умови подорожі можуть не відповідати очікуванням будь-якої зі сторін, особливо через обмін інформацією про вимоги та очікування через профілі та повідомлення. Безпека та довіра також можуть стати питанням, бо хоча BlaBlaCar надає деякі заходи безпеки, існує ризик зустрічі з ненадійними або небезпечними особами. Затримки та зміни маршрутів також можуть вплинути на планування подорожі через можливість непередбачених обставин з боку водіїв або пасажирів. Платіжні питання та вартість можуть виникнути через несправедливість у розділі витрат, а також з питань прозорості та чесності щодо оплати. Крім того, обмежена доступність BlaBlaCar у деяких регіонах або країнах може ускладнити пошук попутників або доступність поїздок. Хоча BlaBlaCar може бути зручним і ефективним сервісом для багатьох подорожуючих, важливо усвідомлювати ці потенційні недоліки та ризики при використанні попутництва. На Рисунку 1.1 наведено приклад користувацького інтерфейсу сайту BlaBlaCar.

Uklon, хоч і є популярним сервісом для замовлення таксі, має свій функціонал пошуку попутника але він має свої недоліки.

Обмежений функціонал для пошуку попутників: У порівнянні з іншими сервісами пошуку попутників, Uklon має обмежені можливості для організації спільних поїздок. Відсутність спеціальних інструментів або розділів для пасажирів та водіїв, які шукають попутників на спільний маршрут, ускладнює процес знаходження попутників.

Відсутність системи рейтингування та відгуків: Uklon не має системи, яка дозволяла б користувачам залишати відгуки чи оцінювати один одного. Відсутність цієї функції може зменшити рівень довіри між користувачами та ускладнити вибір попутників.

Обмеженість можливостей налаштування параметрів пошуку: Користувачі Uklon можуть бути обмежені у налаштуванні параметрів пошуку, таких як час відправлення, маршрут та інші умови подорожі. Це може ускладнити процес знаходження попутників, оскільки користувачі не можуть точно визначити свої умови перевезення.

Недостатня відкритість та прозорість: Недостатня відкритість та прозорість щодо інформації про попутників та їхній досвід подорожі може створювати невизначеність та ризики для користувачів. Відсутність можливості перегляду коментарів та відгуків від інших користувачів може ускладнити процес прийняття рішення щодо попутників.

Недостатня геолокація та деталізація маршрутів: Uklon може не надавати достатньої інформації щодо геолокації та деталей маршруту, що може ускладнити процес знаходження попутників та планування подорожі.

Сервіс BananaCar, який позиціонується як другий за популярністю серед користувачів в Україні, має свої особливості порівняно з BlaBlaCar. На відміну від останнього, BananaCar відзначається можливістю зберігання даних про кожну поїздку, що дозволяє вирішувати будь-які суперечки або проблеми, які можуть виникнути під час виконання заявки на поїздку. Багато користувачів використовують цей сервіс саме для поїздок за кордон, враховуючи його привабливість у цьому вимірі. Інтерфейс BananaCar відмінно адаптований для користувача, що робить взаємодію з платформою зручною та легкою. Однак, основним недоліком є обмежена кількість опцій для фільтрації поїздок, що ускладнює пошук попутників для внутрішніх поїздок в Україні. Брак можливості залишити відгук після завершення поїздки позбавляє користувачів можливості оцінити своїх попутникыв, що може вплинути на рішення інших користувачів щодо вибору попутника.

Сервіс BananaCar надає можливість додавати поїздку як водію так і шукати існуючі поїздки для попутника. На рисунк 1.2 наведено интерфейс пошуку поїздки.

Сервіс Waybro, хоча новий на ринку, вже відзначається популярністю серед мешканців великих міст. Цей сервіс дозволяє знаходити попутників для поїздок в межах міста, що є важливою перевагою для багатьох користувачів. Однак, варто відзначити, що обмежена кількість критеріїв для фільтрації поїздок може викликати незручності. Наприклад, відсутність можливості вказати кількість вільних місць у автомобілі, наявність багажу та його об'єм, а також можливість перевезення тварини. Відсутність можливості використовувати функцію геолокації для точного визначення місця знаходження попутника також є суттєвим недоліком, що може призвести до затримок та непорозумінь між користувачами.

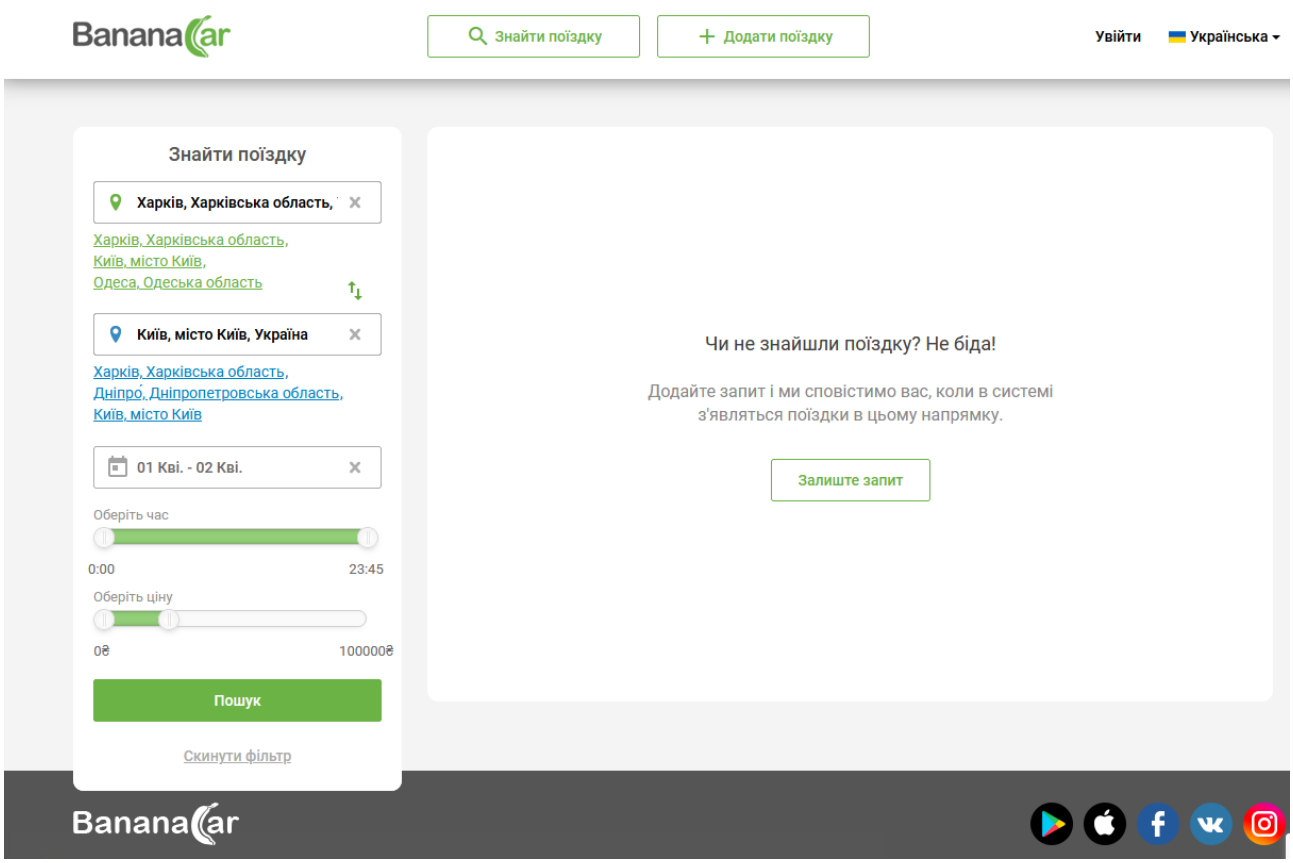


Рисунок 1.2 - Користувацький інтерфейс сайту BananaCar

Сервіс Waybro - це новий сервіс, спрямований на полегшення пошуку попутників для подорожей всередині міста. Він відрізняється від інших сервісів своєю спрямованістю на внутрішньоміські поїздки, що робить його особливо популярним серед мешканців великих міст, де швидке та зручне пересування є важливим.

Однак, не дивлячись на свою популярність, Waybro має деякі обмеження та недоліки. Наприклад, сервіс пропонує обмежений набір критеріїв для пошуку попутників або водіїв, що може обмежити можливості користувачів. Відсутність можливості вказати додаткові параметри, такі як кількість вільних місць у автомобілі, наявність багажу та його об'єм, можливість перевезення тварини або здійснення зупинки у певному місці для попутника, може бути недоліком для користувачів, які мають специфічні потреби.

Також важливим недоліком є відсутність функції геолокації, яка дозволяє точно визначити місце знаходження попутника або водія. Це може ускладнити процес знаходження та підтвердження заявки на поїздку. Крім того, такі сервіси мають ризик стати місцем для шахрайства, оскільки вони не завжди перевіряють

дозволи на перевезення та не надають рейтингових систем для оцінки користувачів або перегляду відгуків.

Не забуваючи про вищезгадані недоліки, слід пам'ятати, що Waybro може бути корисним інструментом для тих, хто шукає попутників для внутрішньоміських поїздок, проте користувачам варто бути обережними та обдуманими при використанні цього сервісу.

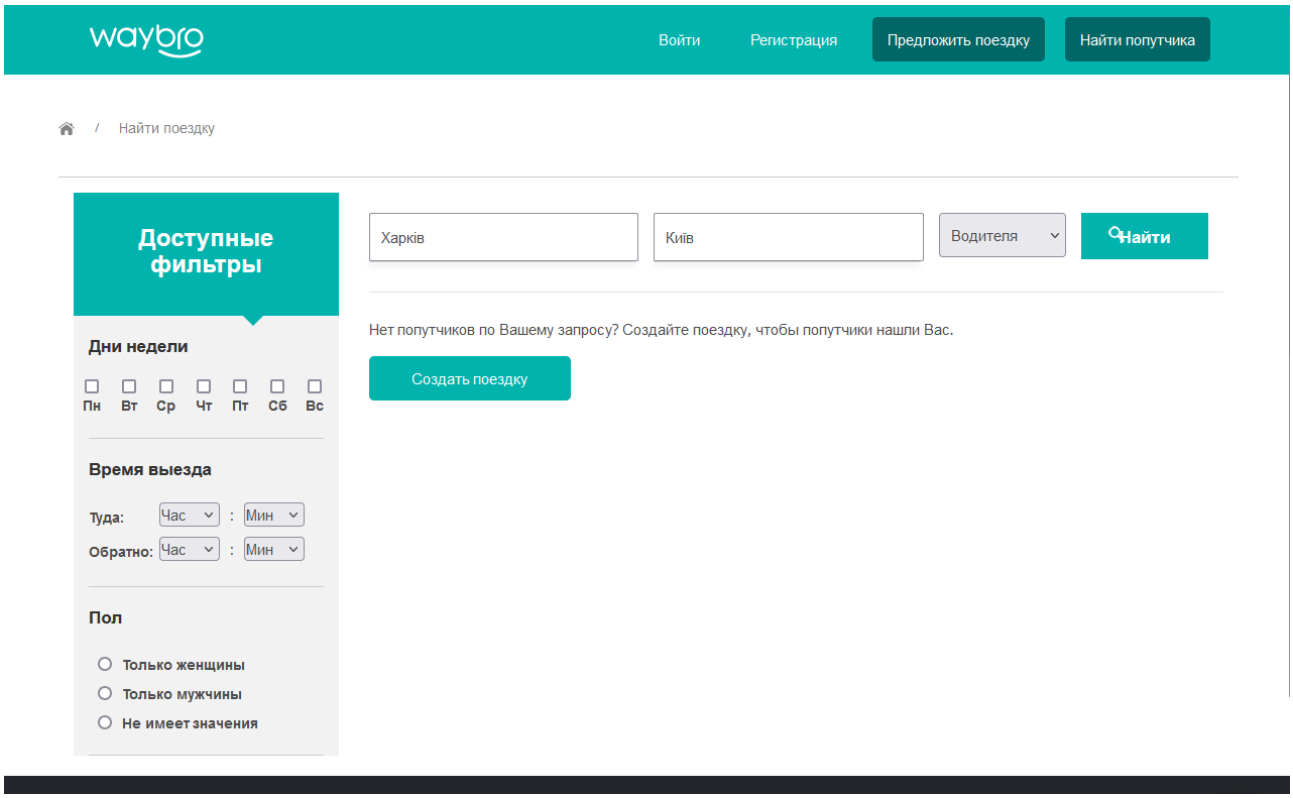


Рисунок 1.3 - Користувацький інтерфейс сайту Waybro

Шукати попутників через інформаційні системи, які дозволяють розміщати оголошення, є найбільш ризикованим варіантом серед сервісів пошуку попутників. У цьому випадку фізична особа розміщає оголошення, і немає можливості перевірити, наскільки надійна ця людина або чи має вона дозвіл на перевезення пасажирів.

Подорожі через сайти оголошень можуть бути пов'язані з різними ризиками, які важливо усвідомлювати перед початком подорожі. Одним з основних ризиків є непередбачуваність, оскільки відсутність чітких угод щодо часу відправлення, місця зустрічі та маршруту може призвести до непередбачуваних ситуацій. Крім того, поїздка через сайти оголошень може бути пов'язана з ризиком пошкодження майна, відсутністю страхування та

небезпекою для безпеки у разі зустрічі з ненадійними особами. Невідповідність очікуванням та непередбачені витрати також можуть стати проблемою під час поїздки. Окрім цього, важливо враховувати ризик інформаційної безпеки, оскільки введення особистих даних на сайті оголошень може призвести до їх зловживання або порушення конфіденційності. Нарешті, невдале зіставлення інформації в оголошеннях також може викликати проблеми під час пошуку попутників. Такі ризики варто усвідомлювати та приймати відповідні заходи обережності перед початком подорожі через сайти оголошень. Крім того, відсутні рейтингові системи, які дозволяють переглядати рейтинг користувача або переглядати позитивні та негативні відгуки, залишені іншими користувачами.

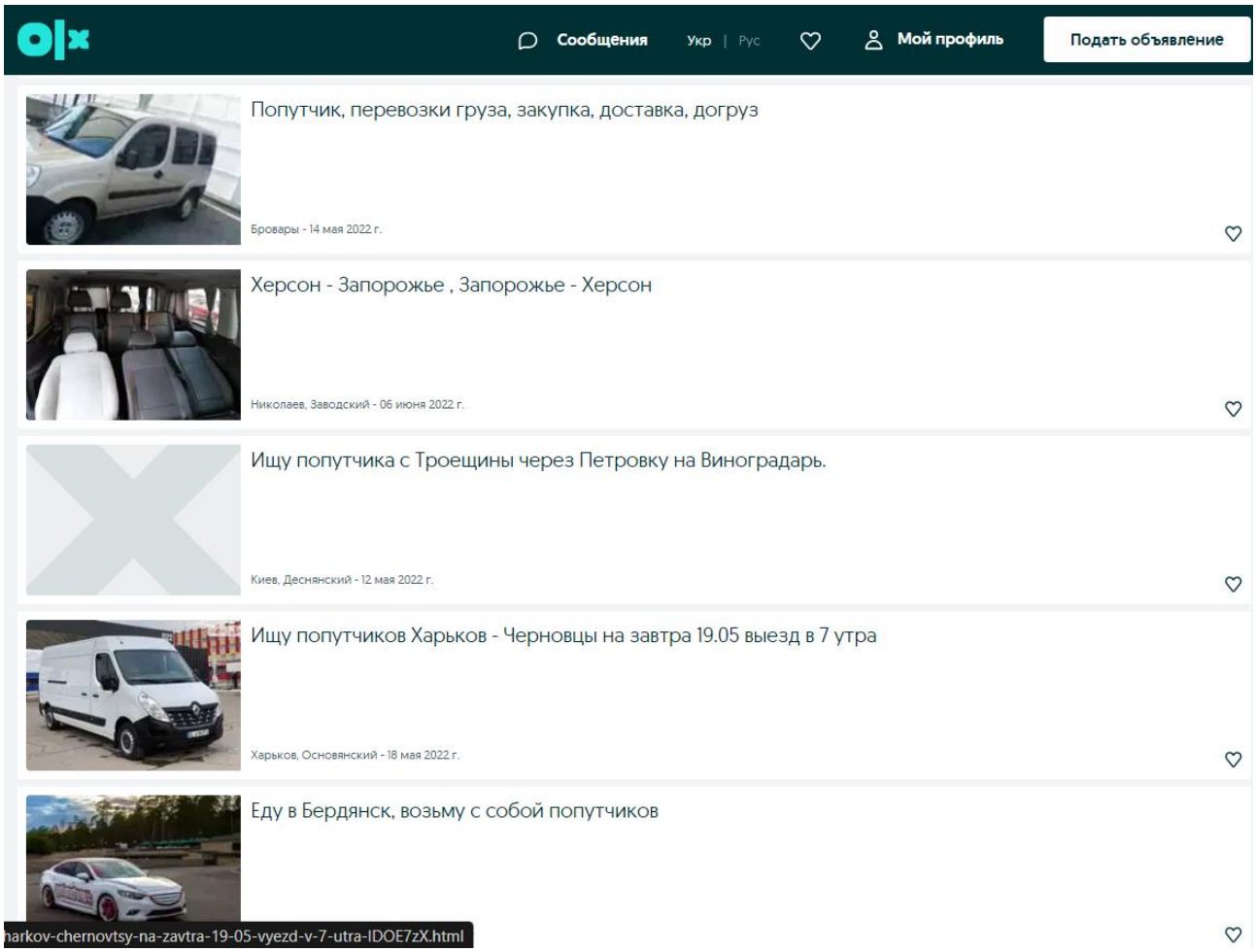


Рисунок 1.4 - Користувацький інтерфейс сайту OLX

На прикладі інтерфейсу відомого сервісу для розміщення оголошень OLX показано список знайдених оголошень про пошук попутників на рисунку 1.4

Інші, сервіси з пошуку попутників вимагають надавати документи про проходження технічного оглядку автівки. Використовуючи сервіси об’яв, не можливо перевірити технічний стан автівки, тому це велика загроза для життя попутника. Перелічені вище проблеми для користувачів є великими недоліками. Також недоліком цього варіанту пошуку попутнику є те що, необхідність переїхати із однієї точки міста у інше може виникнути спонтанно у короткий проміжок часу. Тому шукати на сторінках з об’явами підходящу поїздку може зайняти тривалий час. Це трапляється через неоптимізованість пошуку, тому що не існує параметрів для фільтрації поїздок.

Інші сервіси пошуку попутників, на відміну від списків, часто вимагають надання документів про пройдений технічний огляд автомобіля. Оскільки користувачі в таких сервісах не мають можливості перевірити технічний стан автомобіля перед поїздкою, це може становити серйозну загрозу для безпеки попутників. Перераховані вище проблеми є значними недоліками для користувачів. Додатково, недоліком такого типу пошуку попутників є те, що потреба в поїздці з однієї точки міста в іншу може виникнути спонтанно і в короткий строк. Тому пошук відповідної поїздки на сторінках з оголошеннями може зайняти значний час через відсутність оптимізованих параметрів фільтрації поїздок.

Додатково було знайдено важливий недолік, що є у кожній системі з розглянутих систем , або варіантів пошуку попутника. Жодна з систем не надає можливості автоматичного пошуку попутників за наданими критеріями пошуку поїздки. І цей недолі є, як у володарів автівок так і у попутників. Автоматичний підбір є зручним для користувачів і дозволяє шукати поїздку у реальному часі, а не робити заказ поїздки заздалегідь через домовленість між користувачами системи.

Використання такого варіанту пошуку попутників дозволить зекономити час користувачів, що дозволить зробити пошук більш швидким та зручним. Автоматичний підбір попутників може відкрити безліч можливостей для користувачів. Основна перевага полягає у тому, що вони можуть шукати поїздки у реальному часі, що робить процес більш гнучким та адаптивним до їхніх потреб. Крім того, відпадає необхідність робити попередні домовленості між учасниками системи, що заощаджує час і спрощує взаємодію.

Автоматизація цього процесу не лише дозволяє користувачам швидко знаходити попутників, але й забезпечує їхню безпеку та комфорт. Завдяки

встановленим критеріям та фільтрам користувачі можуть знаходити попутників, які відповідають їхнім вимогам щодо безпеки, спільних інтересів або інших чинників, що важливі для них. Це робить подорожі більш приємними та зменшує ризик непередбачених ситуацій під час поїздки.

Крім того, автоматичний підбір попутників може стимулювати спільне використання транспорту та зменшення кількості приватних автомобілів на дорогах. Це може призвести до зменшення заторів, викидів вуглекислого газу та сприяти створенню більш сталого середовища.

Отже, автоматизований процес пошуку попутників не лише зробить подорожі зручнішими та ефективнішими для користувачів, але й сприятиме зменшенню негативного впливу на довкілля та розвитку більш сталої та екологічно свідомої транспортної системи.

1.5 Постановка задачі

Головною метою дослідження є аналіз та вивчення різних підходів до проєктування систем організації перевезень попутним транспортом. Основною задачею є розробка та аналіз компонентів інформаційних систем, спрямованих на ефективний пошук та організацію поїздок з використанням попутного транспорту. Система, розроблюється, має застосовувати передові технології для автоматизації пошуку та організації поїздок, забезпечуючи зручний та ефективний процес користування для усіх учасників.

Дослідження передбачає вивчення можливостей та переваг різних підходів до створення таких систем, включаючи аналіз існуючих платформ та методів їхнього функціонування. Результатом дослідження буде розробка оптимальної концепції системи організації перевезень попутним транспортом, яка враховуватиме потреби користувачів і забезпечуватиме найвищий рівень зручності та ефективності.

Система також має передбачати можливість ознайомлення нових користувачів з інструкцією користування перед початком використання, а також забезпечити авторизацію для власників автівок та користувачів, які шукають попутників. При цьому необхідно врахувати безпеку та надійність системи, а також забезпечити зручний інтерфейс для користувачів з різним рівнем технічної підготовки.

Система пошуку попутників повинна включати такі ролі для користувачів:

– користувач без реєстрації (Unregistered User);
– зареєстрований користувач (Registered User);
– модератор (Moderator).

Основні задачі дослідження цієї роботи включають такі етапи:

Аналіз існуючих підходів до організації перевезень попутним транспортом:

Розробка моделі системи: Створення концептуальної моделі системи організації перевезень попутним транспортом, включаючи архітектурні принципи, базові функціональні можливості та основні компоненти. Що включає опис взаємодії між компонентами системи, включаючи протоколи обміну даними та логіку переходів між станами системи. Детальний аналіз вимог до системи. Нарешті, розробка дизайну користувацького інтерфейсу та вибір оптимального стеку технологій завершують створення концептуальної моделі системи організації перевезень попутним транспортом.

Визначення технологічних підходів: Вибір технологій та інструментів розробки, які найбільш підходять для реалізації системи організації перевезень попутним транспортом, з урахуванням масштабу проекту і потреб користувачів.

Проведення експериментів та аналіз результатів: Практичне тестування розробленої системи, залучення користувачів до взаємодії з нею та збір фідбеку для виявлення сильних та слабких сторін системи.

2 РОЗРОБКА ВИМОГ ДО СИСТЕМИ

2.1 Розробка системних вимог

При розробці будь-якого програмного продукту важливим етапом є процес проектування. Це ключовий момент, коли визначаються вимоги до системи, яку планують створити. У випадку інформаційної системи для пошуку автомобільних попутників цей етап не менш важливий. Визначення системних вимог є кроком, який передбачає встановлення функціональних та нефункціональних характеристик майбутньої системи. Оскільки ця інформаційна система призначена для автоматизації процесу пошуку попутників, важливо чітко визначити, які функції вона повинна виконувати та які вимоги повинні бути задоволені.

Дослідження підходів до проєктування систем організації перевезень попутним транспортом передбачає детальне розглядання функціоналу серверної частини системи. Однією з головних функцій сервера є можливість створення заявок на поїздку. Важливо забезпечити можливість подання заявки на поїздку як для зареєстрованих користувачів, так і для незареєстрованих. Поїздка має бути створена після заповнення обов'язкових полів у відповідній формі, яка містить інформацію, необхідну для формування заявки. Крім того, користувач повинен мати можливість додати більш детальну інформацію про поїздку, що допоможе відібрати найбільш підходящого попутника. Ця інформація повинна включати додаткові деталі про маршрут поїздки, які можуть співпадати з маршрутом водія повністю або частково.

Серверна частина системи повинна надавати можливість зареєстрованим користувачам, які є власниками автомобілів, створювати поїздки. Власник автомобіля є обов'язково зареєстрованим користувачем, і для нього повинна бути реалізована можливість створення поїздки. Поїздка буде створена після заповнення необхідних полів у відповідній формі. Крім того, серверна частина системи повинна вміти шукати попутників, маршрут яких співпадає з маршрутом власника автомобіля. Після надходження певної кількості заявок, система має відшукати попутників для користувачів.

Система повинна підтримувати режим роботи для багатьох користувачів. У майбутньому може бути розроблений мобільний додаток для платформи

інформаційної системи. Щодо надійності системи, список вимог до безпеки включає наступне:

– забезпечення захисту даних для уникнення несанкціонованого доступу;

– виявлення некоректних даних при їх введенні з подальшим повідомленням користувачу;

– перевірка введених даних на коректність, щоб уникнути неправильного введення користувачем;

– у випадку намагання видалити дані, система має запропонувати можливість відмови видалення та вимагати додаткового підтвердження користувача для видалення.

2.2 Обґрунтування вибору напрямку досліджень

Аналіз методів організації перевезень попутним транспортом у наукових дослідженнях та практичних реаліях:

У наукових дослідженнях та практичних реаліях існує ряд методів організації перевезень попутним транспортом, які варіюються за складністю, використовуваною технологією та ефективністю. До цих методів входять:

Автоматизовані платформи для організації попутних перевезень: Існують веб-платформи та мобільні додатки, які забезпечують зручний та ефективний спосіб пошуку та організації попутних перевезень.

Спільні транспортні сервіси: Ці сервіси надають можливість спільного використання автівок між користувачами, що сприяє ефективнішому використанню транспортних ресурсів та зменшенню заторів на дорогах.

Аналіз цих методів у наукових дослідженнях та практичних реаліях дозволяє визначити їхні переваги, недоліки та області застосування.

Серед вимог поставлених для системи пошуку автомобільних попутників є певні критерії оцінки якості системи. Перелік вимог наведено нижче.

Автоматизовані платформи можуть прискорити процес знаходження попутників та організації поїздки, проте деякі системи можуть вимагати більше часу для реєстрації та вибору оптимального маршруту. Спільні транспортні сервіси можуть бути швидкими у використанні, оскільки вони дозволяють безпосередньо шукати доступні автомобілі. Тому швидкість пошуку попутника або поїдки має бути великою і це є важливим критерієм.

Автоматизовані платформи можуть забезпечувати більшу надійність у виборі попутників з високим рейтингом та перевіреними автомобілями. Спільні транспортні сервіси можуть стикатися з проблемами, такими як затримки або відмови з боку водіїв, що може впливати на якість поїздки.

Автоматизовані платформи можуть надавати більше можливостей для персоналізації та вибору умов поїздки, таких як музика, кондиціонер та інші функції. Спільні транспортні сервіси можуть бути менш зручними у використанні, оскільки вони можуть бути залежними від доступності автомобілів у конкретний час. Тому широкий вибір фільтрації критеріїв пошуку дозволить знайти найбільш підходящу по якості для користувача поїздку.

Тому система що розроблюється має відповідати критеріям якості наведеним вище. Щоб надавати користувачам надійний і зручний у використані сервіс.

2.3 Розробка функціональних вимог

Таблиця 1.1 Перелік та опис функціональних вимог для систем пошуку автомобільних попутників

Назва вимоги	Функціональні вимоги
Реєстрація користувача	Система повинна надавати можливість реєстрації для користувачів, які бажають створити обліковий запис.
	Користувач повинен мати можливість увійти в систему після реєстрації за допомогою ідентифікаційних даних (логін, пароль)
Створення поїздки	Користувачі, що володіють автомобілем, повинні мати можливість створювати заявки на поїздки.
	Під час створення поїздки, водії повинні вказувати інформацію про маршрут, дату та час відправлення, кількість доступних місць, вартість поїздки тощо.

Продовження таблиці 1.1

Назва вимоги	Функціональні вимоги
Пошук поїздки	Користувачі, які шукають поїздку, повинні мати можливість переглядати доступні поїздки за різними критеріями, такими як маршрут, дата, час, вартість тощо.
	Система повинна забезпечити можливість фільтрації поїздок з урахуванням вказаних користувачем параметрів.
Бронювання місця	Користувачі, які знайшли підходящу поїздку, повинні мати можливість бронювати доступне місце у автомобілі водія.
	Підтвердження бронювання повинно надходити як водієві, так і пасажиру.
Управління бронюваннями	Водії та пасажири повинні мати можливість переглядати та управляти своїми бронюваннями, включаючи скасування, зміну дати або часу поїздки тощо.
Оплата поїздки	Система повинна підтримувати можливість оплати поїздки онлайн, використовуючи різні платіжні методи.

2.4 Аналіз підходів до проектування систем організації перевезень попутним транспортом

При проектуванні і створенні системи організації перевезені попутним транспортом можна виділити 5 основних підходів щодо проектування подібних систем.

При централізованому підході всі операції керуються з одного центрального сервера або групи серверів. Цей підхід забезпечує високу контрольованість і координацію всіх процесів.

Переваги:

– Єдиний контроль: Легко керувати і моніторити систему з одного центру.

– Швидкість обробки даних: Оперативна обробка запитів і надання відповідей користувачам.

– Безпека: Легше впроваджувати політики безпеки і захисту даних.

Недоліки:

– Централізація ризиків: Вразливість до збоїв у роботі центрального сервера.

– Масштабованість: Важче масштабувати систему для обробки великої кількості користувачів.

Серед відомих компаній, що використовують централізовану модель для управління замовленнями поїздок, маршрутизацією і платіжними операціями є Uber.

Другий підхід у проектувані систем це проектування децентралізованої системи. У децентралізованій системі кожен учасник або група учасників може керувати своєю частиною системи незалежно від центрального сервера.

Переваги:

– Стійкість до збоїв: Відсутність єдиного центру знижує ризик повного збоїв.

– Масштабованість: Легше додавати нових учасників без перевантаження центрального сервера.

– Прозорість: Використання блокчейна забезпечує прозорість і надійність записів.

Недоліки:

– Керованість: Складніше управляти і контролювати систему.

– Безпека: Питання безпеки і захисту даних можуть бути складнішими через розподілений характер системи.

Третім варіант проектування подібних систем є гібридним. Гібридна система поєднує елементи як централізованих, так і децентралізованих систем. Центральний сервер може керувати основними процесами, тоді як деякі функції можуть бути децентралізованими. Гібридними є системи, де центральний сервер управляє основними функціями, такими як обробка платежів і безпека, а децентралізовані вузли обробляють менш критичні задачі, як-от маршрутизація.

Переваги:

– Гнучкість: Поєднує переваги обох підходів.

– Масштабованість і надійність: Децентралізовані елементи забезпечують надійність, а центральний сервер дозволяє легко керувати системою.

– Оптимізація ресурсів: Можливість оптимізувати використання ресурсів залежно від поточних потреб.

Недоліки:

– Складність: Більш складна архітектура системи, що потребує ретельного планування і реалізації.

– Вартість: Можливо, вища вартість впровадження і підтримки.

Також можна використовувати Peer-to-Peer (P2P) підхід для системи перевезення попутним транспортом. У P2P системі користувачі взаємодіють безпосередньо один з одним без централізованого посередника. Цей підхід може використовуватися для організації поїздок, де водії і пасажири знаходять один одного за допомогою P2P мережі.

Переваги:

– Безпосередня взаємодія: Знижує залежність від центрального сервера.

– Висока масштабованість: Кожен новий користувач стає частиною мережі, підвищуючи її загальну потужність.

– Зменшення витрат: Відсутність необхідності в центральному сервері може знизити витрати на інфраструктуру.

Недоліки:

– Безпека і довіра: Складніше забезпечити безпеку і довіру між користувачами.

– Контроль якості: Відсутність централізованого контролю може ускладнити забезпечення якості послуг.

П’ятим варіантом для створення подібних систем створення централізованої Saas системи.

У централізованій SaaS системі всі операції і дані зберігаються і обробляються на централізованому сервері або групі серверів.

Переваги:

– Єдиний контроль: Просте управління і моніторинг всієї системи з одного центру.

– Швидкість обробки: Ефективне управління ресурсами і швидка обробка запитів.

– Безпека: Легше впроваджувати політики безпеки і захисту даних.

Недоліки:

– Ризики відмови: Якщо центральний сервер вийде з ладу, вся система може припинити роботу.

– Масштабованість: Може бути складніше масштабувати систему для обслуговування великої кількості користувачів.

SaaS (Software as a Service) — це модель розповсюдження програмного забезпечення, при якій користувачі отримують доступ до програм через інтернет, зазвичай за підпискою. Програмне забезпечення розміщується на серверах провайдера, а користувачі взаємодіють з ним через веб-браузери або клієнтські додатки. SaaS є частиною більш широкої категорії хмарних обчислень.

Saas система включає хостінг на стороні провайдера. Програмне забезпечення розміщується на серверах провайдера, а не на локальних машинах користувачів. А також провайдер відповідає за підтримку, оновлення і безпеку програмного забезпечення.

Серед переваг використання цього підходу також можна віднести той факт, що користувачі отримують доступ до програмного забезпечення через інтернет, зазвичай через веб-браузер. Тому не вимагається установка додаткового ПЗ на локальні машини користувачів.

Saas підхід базується на моделі підписки. Сервіс буде продаватись за підпискою, з можливістю щомісячної або річної оплати. Це дозволяє користувачам уникати значних початкових витрат на придбання програмного забезпечення.

Цей підхід підтримує мультиорендність (мульті-тенантність). Один сервіс пошуку попутним транспортом зможе обслуговувати багатьох користувачів або організацій одночасно, з ізоляцією даних між орендарями. Кожен користувач або організація має власний доступ до даних і налаштувань.

Основні компоненти Saas систем це: інфраструктура, платформа та програмне забезпечення.

До інфраструктури відносяться сервери та сховища на який хоститься програмне забезпечення. А також мережева обладнання, що забезпечує доступ до інтернету і мережеву взаємодію.

До платформи відноситься ОС, на якій запускається програмне забезпечення. А також СУБД, для збереження даних користувача.

Програмне забезпечення включає в себе API та сервіси, які забезпечують функціональність додатку і інтеграцію з іншими системами, а також веб-додатки або клієнтські додатки для взаємодії з користувачами.

Перелік переваг централізованого Saas підходу:

– Відсутність початкових витрат на придбання програмного забезпечення і обладнання.

– Зменшення витрат на технічне обслуговування і підтримку.

– Легкість додавання нових користувачів і збільшення ресурсів відповідно до потреб.

– Висока доступність і стійкість до збоїв.

– Можливість доступу до програмного забезпечення з будь-якого місця з інтернет-з’єднанням.

– Підтримка різних пристроїв і платформ

– Автоматичне оновлення програмного забезпечення без потреби втручання користувачів.

– Завжди актуальна версія програмного забезпечення з новими функціями і виправленнями помилок.

Недоліки централізованого Saas підходу:

– Необхідність постійного інтернет-з'єднання для доступу до програм.

– Можливі проблеми з продуктивністю при низькій швидкості інтернету.

– Передача і зберігання конфіденційних даних на сторонніх серверах може викликати занепокоєння щодо безпеки.

– Необхідність ретельного вибору надійного провайдера SaaS.

– SaaS рішення можуть бути менш гнучкими в порівнянні з локальними рішеннями.

– Обмежені можливості для індивідуалізації програмного забезпечення під конкретні потреби організації.

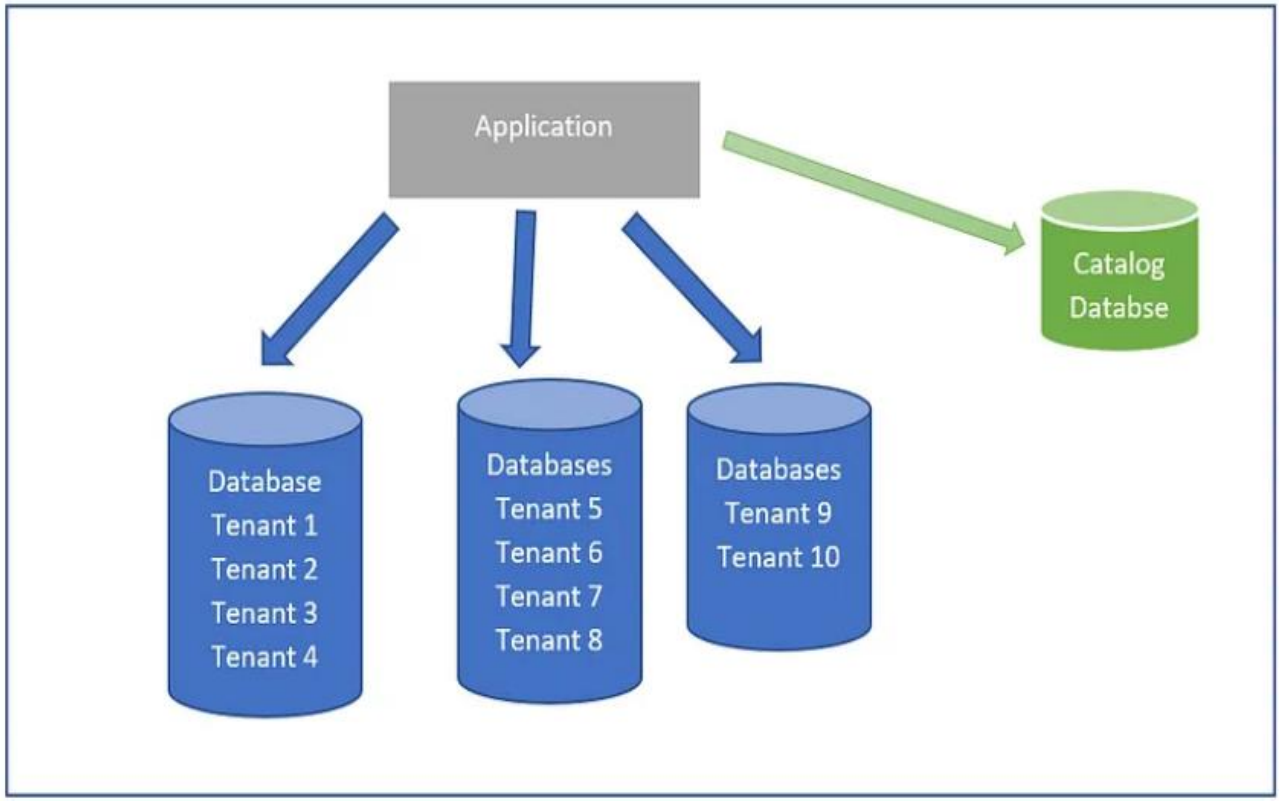


Рисунок 2.8 Схема Saas сервісу з спільною багатотенантною базою даних

На рисунку 2.8 наведено Saas сервісу з використанням з спільної багатотенантної бази даних. У цій моделі один екземпляр додатка обслуговує безліч клієнтів, а одна база даних містить дані для багатьох клієнтів. Це найекономічніший спосіб розробки SaaS-додатків. Крім того, у цій моделі легко розгортати нові функції. Але з іншого боку, ви втратите ізоляцію орендарів, масштабованість, можливість налаштування, моніторинг продуктивності та обслуговування. Оскільки всі орендарі використовують один і той самий обчислювальний ресурс для бази даних, ви не можете обмежити обчислювальні ресурси для кожного орендаря.

Таким чином, велике навантаження на одного орендаря буде сильно впливати на продуктивність інших орендарів. Крім того, у ваших реляційних таблицях має бути стовпець для ідентифікації орендаря, і вам слід фільтрувати всі дані за ідентифікатором орендаря при запиті з бази даних.

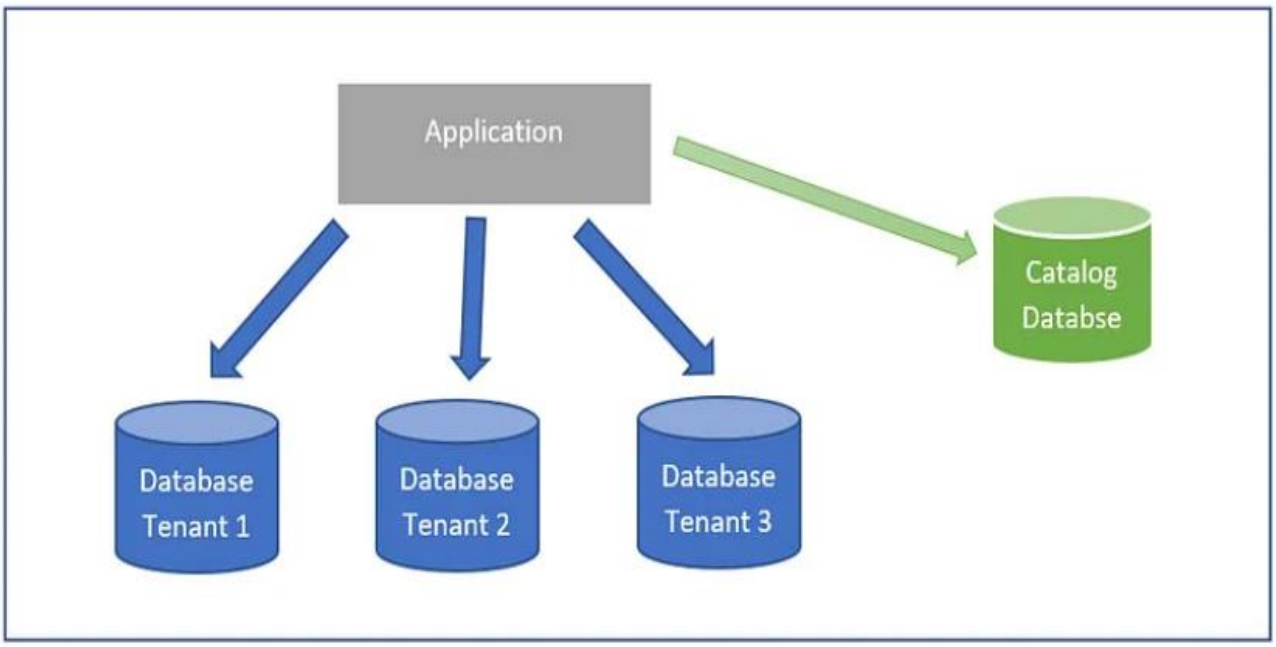


Рисунок 2.9 Схема Saas сервісу базою даних на тенант

На рисунку 2.9 наведено зі схемою Saas системи з базою даних на кожний тенант. У цій моделі один екземпляр додатка буде використовуватися кількома орендарями. Однак для кожного орендаря буде розгорнуто окрему базу даних. У цій моделі зберігається сувора ізоляція клієнтів, оскільки кожен клієнт має свою окрему базу даних. Проте додаток необхідно масштабувати по вертикалі або по горизонталі, щоб задовольнити зростаючий попит.

База даних каталогу зберігає відповідність між клієнтом і базою даних. Коли в додаток надходить запит (припустимо, у вас є серверна частина без збереження стану), вам необхідно визначити точного орендаря, від якого надійшов запит, потім знайти правильну базу даних у каталозі і динамічно побудувати рядок підключення, щоб з'єднатися з базою даних орендарів. У порівнянні з попередньою моделлю, у цій моделі менше проблем з обслуговуванням, оскільки вам не потрібно розгортати додаток для кожного орендаря окремо. Однак слід бути обережним при налаштуванні конкретного орендаря, оскільки інші орендарі також використовують ту саму бізнес-логіку в додатку.

Підсумовуючи можна зрозуміти, що використання централізованої Saas моделі для проектування системи з пошуку автомобільних попутників є абсолютно виправданим і доречним.

2.5 Розробка діаграми прецедентів UseCase

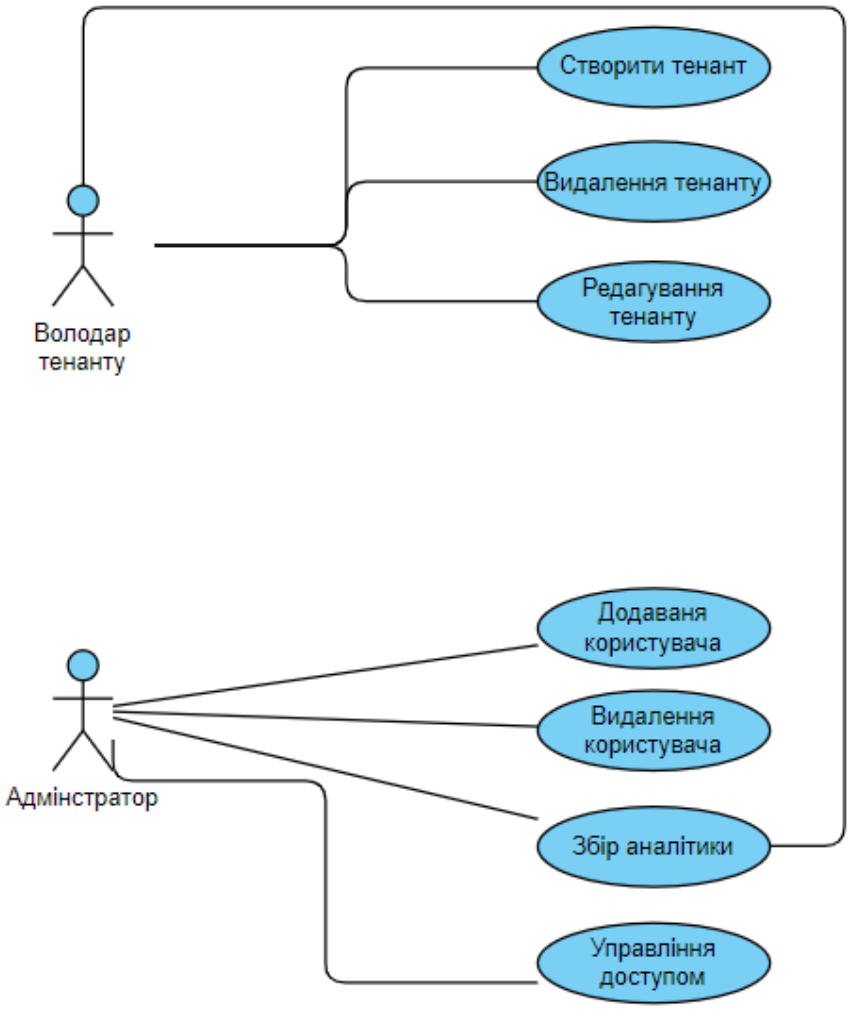


Рисунок 2.10 – Use-Case діаграма сервісу для сторони менджмента

На рисунку 2.10 наведено,що у системі, що проектується на адміністративній частині, реалізовано два типи користувачів:

– Володар тенанту;

– Адміністратор.

Для володар тенанту визначено такі бізнес функції: створення тенанту, видалення тенанту, редагування тенанту.

Адміністратор має доступ для таких бізнес функцій: ручне додавання користувачів у систему, видалення користувачів, збір аналітики та управління доступом.

На рисунку 2.11 наведено Use-Case діаграму сервісу автоматизованого пошуку автомобільних попутників.

У системі, що проектується, реалізовано три типи користувачів:

– гість;

– пасажир;

– водій.

Для не авторизованих користувачів передбачена можливість створити обліковий запис.

Для пасажирів створено функціональні можливості, необхідні для опису компонентів інформаційної системи з пошуку автомобільних попутників. Пасажири можуть переглядати список заявок на поїздки, створювати нові заявки, обирати поїздки та підтверджувати заявки.

Деякі функціональні можливості є спільними для пасажирів і водіїв.

Для водіїв передбачені специфічні функції, такі як створення поїздки та введення даних про автомобіль. Водії також можуть обирати поїздки та підтверджувати заявки, що є спільними функціями з пасажирами.

Наведені функції реалізують усю бізнес-логіку, що вимагається від системи пошуку автомобільних попутників.

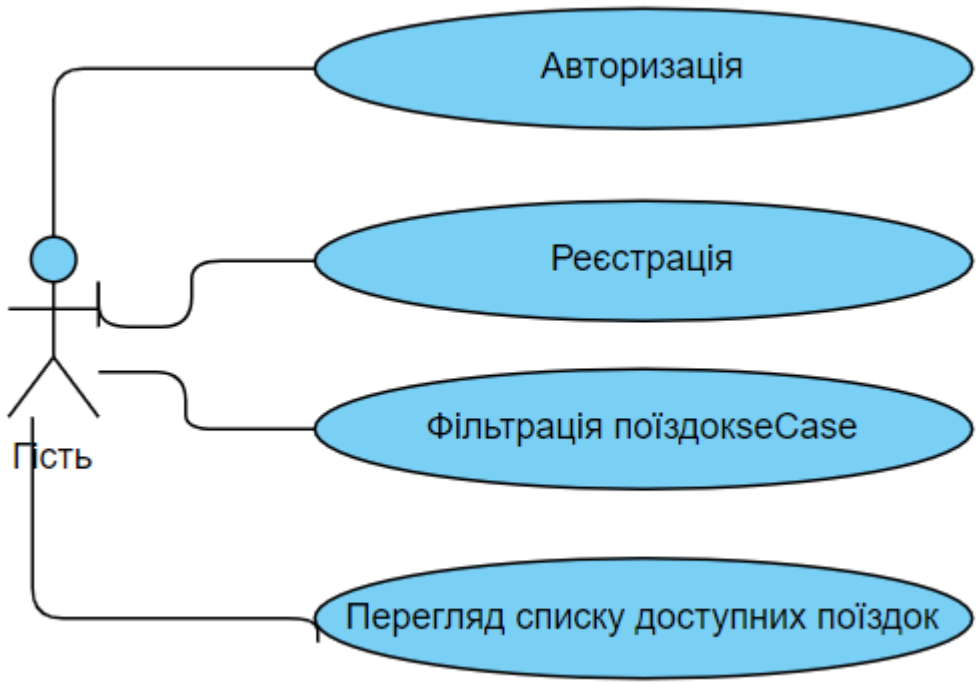


Рисунок 2.11 – Use-Case діаграма функцій доступних користувачам з роллю гість

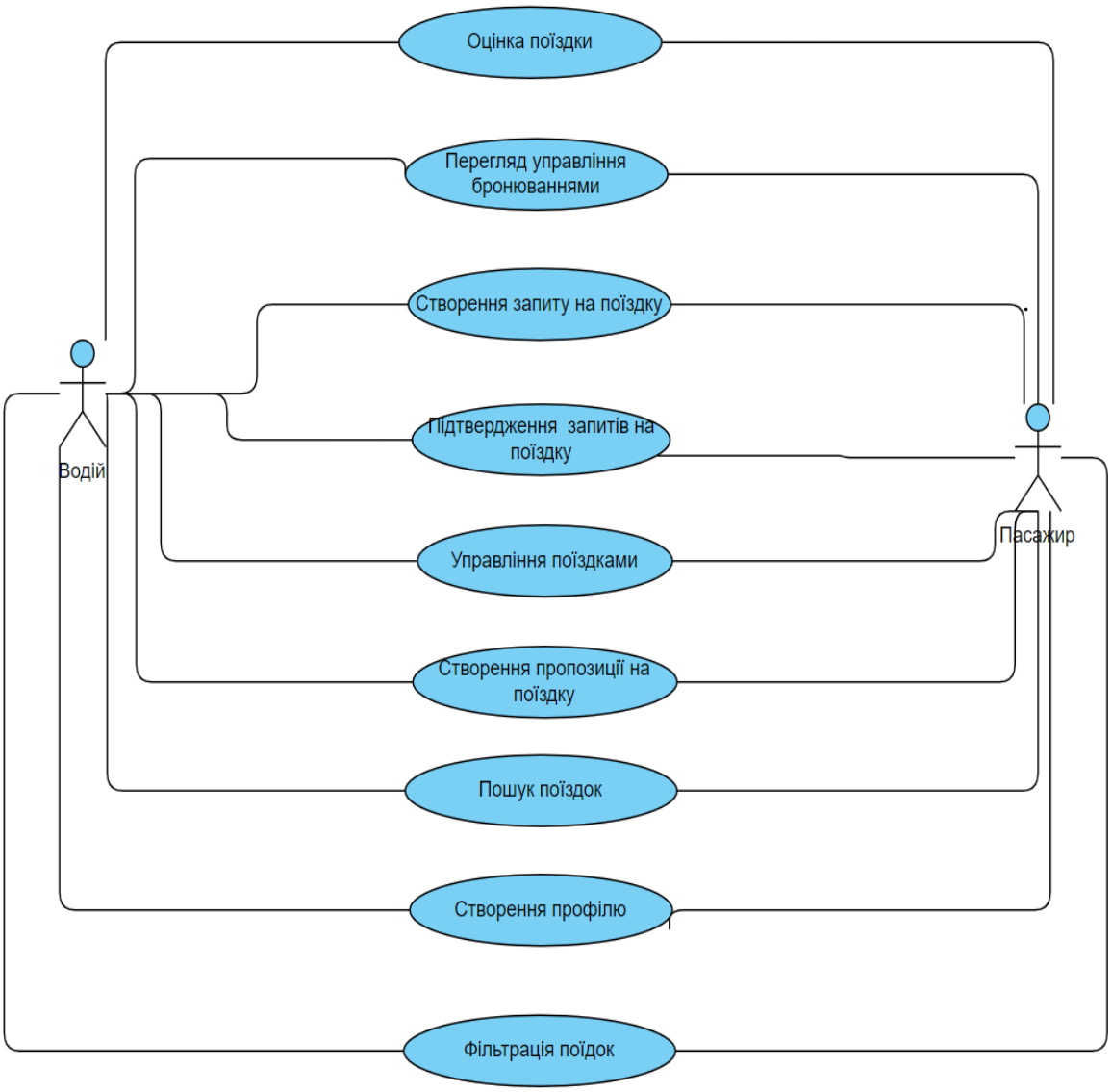


Рисунок 2.12 – Use-Case діаграма функцій доступних користувачам системи

Гість це неавторизований користувач або, користувач, який не створив обліковий запис. Будь-який користувач з такою роллю має обмежений доступ до функцій системи. Вищє наведено перелік доступних функцій:

– авторизація;
– реєстрація;
– перегляд списку доступних поїздок;
– фільтрація поїздок.

Користувачу з роллю пасажир та водій мають доступ до основних функцій сервісу. А саме для користувачів з роллю водій було визначено такі функції:

– підтвердження запитів на поїздку:
– управління поїздками:

– створення поїздок;
– управління бронюванням;
– оцінка поїздки;
– оцінка поїздки фільтрація поїздок.

Користувачу з роллю пасажир є доступ до таких операцій у системі пошуку автомобільних попутників:

– фільтрація поїздок;
– створення профілю;
– оцінка;
– перегляд і управління бронюванням;
– створення запиту на поїздку.

3 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

3.1 Обґрунтування вибору мови програмування

В процесі розробки сервісу пошуку автомобільних попутчиків виникла необхідність вибору мови програмування та платформи, які забезпечать високу продуктивність, масштабованість, надійність і зручність розробки. Після детального аналізу існуючих варіантів, було прийнято рішення використовувати платформу .NET Core.

3.1.1 Основні критерії вибору мови програмування

Платформа .NET є продуктивною і кросплатформною. Ця високопродуктивна платформа дозволяє обробляти запити користувачів у реальному часі, що є критично важливим для сервісу пошуку автомобільних попутників.

Також для цього сервісу є важливим масштабування. Можливість ефективного масштабування проекту з урахуванням зростання кількості користувачів та обсягів даних.

Також серед переваг обраної платформи є то, що вона підтримує асинхронність. Будь-який сучасний сервіс має обробляти багато запитів одночасно, не блокуючи ресурси системи.

Захист даних користувачів та стабільність роботи є невідємною частиною будь-якого нового сервісу, а з точки зору роботи з даними ця платформа додає багато інструментів для безпечної роботи з даними.

Однією з головних переваг .NET Core є його кросплатформеність. Додаток може бути розгорнутий на Windows, Linux або macOS, що дає гнучкість у виборі інфраструктури для розгортання.

.NET Core відомий своєю високою продуктивністю завдяки компіляції JIT (Just-In-Time) та потужному середовищу виконання CLR (Common Language Runtime). Він дозволяє ефективно використовувати апаратні ресурси, що важливо для обробки великих обсягів даних та швидкої реакції на запити користувачів.

Також .NET Core підтримує мікро сервісну архітектуру, що дозволяє легко масштабувати додаток горизонтально. Це особливо важливо для сервісу пошуку попутчиків, де кількість запитів може значно змінюватись.

Мова програмування C# та платформа .NET Core також надають потужні інструменти для асинхронного програмування, що дозволяє обробляти багато запитів одночасно, не блокуючи виконання інших операцій.

Також .NET Core надає вбудовані механізми безпеки, такі як захист від SQL-ін'єкцій, валідація введених даних та шифрування. Це допомагає забезпечити високий рівень захисту даних користувачів.

Враховуючи всі наведені критерії, платформа .NET Core була обрана для розробки сервісу пошуку автомобільних попутчиків. Вона забезпечує високу продуктивність, масштабованість, безпеку та кросплатформеність, що є критично важливим для успіху проекту.

3.2 Визначення алгоритмів пошуку попутників

Під час пошуку алгоритмів алгоритму для порівняння маршрутів було розглянуто декілька алгоритмів, які можуть бути використані для пошуку схожих маршрутів за координатами. Деякі з найпоширеніших алгоритмів включають:

Алгоритм k-найближчих сусідів (kNN): Цей алгоритм шукає k найближчих точок до даної точки в просторі координат. За допомогою цього методу можна знайти схожі маршрути, порівнюючи їхні координати.

Алгоритм згортання шляхів (Path Convolution Algorithm): Цей метод згортає шляхи в просторі координат у граф, а потім використовує алгоритми пошуку шляхів або алгоритми порівняння графів для пошуку схожих маршрутів.

Алгоритм динамічного програмування (Dynamic Programming): Цей метод може бути використаний для порівняння декількох маршрутів за їхньою схожістю. Він розбиває маршрути на підрядки та обчислює схожість між ними, що дозволяє визначити, наскільки два маршрути подібні один до одного.

Геометричні алгоритми: Деякі алгоритми використовують геометричні особливості маршрутів, такі як форма, довжина, кутові звороти тощо, для визначення їхньої схожості.

Кожен алгоритм має свої переваги та недоліки. Нижче наведено порівняння алгоритмів:

Переваги алгоритму k-найближчих сусідів:

– простота реалізації та роботи;

– відносно ефективний для невеликих наборів даних;

– може бути застосований для пошуку схожих маршрутів на основі координат.

Недоліки алгоритму k-найближчих сусідів:

– чутливість до вибору параметру k – кількості найближчих сусідів;

– вимагає великої кількості обчислень для великих наборів даних;

– не дуже ефективний для маршрутів з великою кількістю точок.

Алгоритм згортання шляхів дозволяє враховувати не тільки розташування точок, але й структуру маршруту, а також може бути ефективним для маршрутів зі складною топологією. Але головними недоліками цього алгоритму є його складність реалізації та роботи, особливо для великих масивів даних, що у вимагає великих обчислювальних потужностей.

Для пошуку попутника може бути використаний алгоритм динамічного програмування. Який дозволяє ефективно порівнювати маршрути за їхньою схожістю. А також надає можливість врахування різноманітних параметрів маршрутів, таких як довжина, кутові звороти тощо. Але він також вимагає високої складності обчислень, особливо для великих масивів даних. А також він чутливим до параметрів і вимагає ретельного підбору параметрів та налаштувань.

Серед варіантів алгоритмів динамічного програмування було знайдено LCS алгоритм. Цей метод найбільшої спільної підпослідовності (Longest Common Subsequence, LCS) – це алгоритм динамічного програмування, який використовується для знаходження найбільшої спільної підпослідовності між двома послідовностями. В контексті пошуку попутника, ми можемо розглядати маршрути як послідовності координат точок, і застосовувати метод LCS для знаходження схожості між маршрутом користувача і маршрутом попутника.

Ось як відбувається використання методу LCS для пошуку попутника:

а) Підготовка даних: Початково, обидва маршрути (маршрут користувача і маршрут попутника) повинні бути перетворені на послідовності координат точок. Це може бути зроблено шляхом розбиття маршрутів на окремі точки.

б) Створення матриці: Створюється матриця розміром (m+1) на (n+1), де m – довжина послідовності першого маршруту, а n – довжина послідовності другого маршруту.

в) Обчислення LCS: Використовуючи метод динамічного програмування, заповнюється матриця LCS. Кожен елемент матриці представляє собою довжину найбільшої спільної підпослідовності між підпослідовностями, які складаються з елементів з першого маршруту і перші і елементів з другого маршруту.

г) Відновлення LCS: За допомогою заповненої матриці, можна відновити найбільшу спільну підпослідовність. Це робиться шляхом просування від кінця матриці до початку і визначення, які елементи були використані для побудови LCS.

д) Аналіз результатів: Після відновлення найбільшої спільної підпослідовності можна здійснити аналіз результатів для визначення схожості маршрутів. Наприклад, можна порівняти довжину знайденої LCS з загальною довжиною маршрутів, або використовувати інші метрики для визначення ступеня схожості.

В результаті використання методу LCS можна визначити, наскільки схожі маршрути користувача і попутника на основі їхніх координат. Чим більша спільна підпослідовність, тим більш схожі маршрути, і це може вказувати на потенційну можливість для спільної поїздки.

Схема пошуку поїздки з клієнтської сторони.

На етапі підготовки даних необхідно підготувати дані та сформувати датасет поїздки. Для цього необхідно отримати набір точок маршруту. Координати маршруту будуть отримані зі сторонньої API (Google maps, 2Gis).

У відповідь API повертає дані про маршрут, включаючи детальну інформацію про кожен етап шляху, такі як початкові та кінцеві точки етапів, а також полілінії, які описують кривизну маршруту.

Етап підготовки даних включає у себе такі кроки:

Витяг поліліній. Google Maps Directions API повертає маршрути у вигляді поліліній, які представляють собою закодовані рядки, що описують безліч точок маршруту. Ці полілінії враховують усі повороти, вигини та інші особливості маршруту.

Декодування поліліній. Для декодування поліліній використовується спеціальний алгоритм. У результаті декодування виходить набір координат

(широта і довгота) для кожної точки маршруту. Ці координати точно відображають шлях, включаючи всі кривизни і повороти доріг.

Далі виконується додаткова інтерполяція для більшої точності. Він включає такі шаги.

Інтерполяція точок. Для підвищення точності і деталізації маршруту можна додатково розбити ділянки між уже отриманими точками на дрібніші сегменти. Наприклад, можна додати проміжні точки через кожні 10 метрів. Це особливо корисно для довгих прямих ділянок, де проміжні точки можуть бути далеко одна від одної.

Інтерполяція точок маршруту. Для кожної пари послідовних точок маршруту обчисліть проміжні точки за допомогою інтерполяції по великим колам.

Формула інтерполяції по великим колам:

Обчислюється центральний кут d між двома точками P0 P1:

$$\text{Cos(d)}=\sin(\phi 0)\cdot\sin(\phi 1)+\cos(\phi 0)\cdot\cos(\phi 1)\cdot\cos(\lambda 1-\lambda 0).$$

Далі використовується параметр інтерполяції t, що змінюється від 0 до 1, щоб обчислити проміжні точки:

$$A = \frac{\sin((1-t)\cdot d)}{\sin(d)}$$
$$B = \frac{\sin(t\cdot d)}{\sin(d)}$$

.

Далі обчислюються проміжні координати (x,y,z):

$$x=A\cdot\cos(\phi 0)\cdot\cos(\lambda 0)+B\cdot\cos(\phi 1)\cdot\cos(\lambda 1),$$

$$\phi=\arctan2(z,x2+y2)y=A\cdot\cos(\phi 0)\cdot\sin(\lambda 0)+B\cdot\cos(\phi 1)\cdot\sin(\lambda 1),$$

$$z=A\cdot\sin(\phi 0)+B\cdot\sin(\phi 1),$$

На останньому етапі виконується перетворення з декартової системи назад в широту і довготу:

$$\phi = \arctan 2(z, \sqrt{x^2 + y^2})$$
$$\lambda = \arctan 2(y, x)$$

.

На основі сформованих відстаней обчислюються координати додаткових проміжних точок. Для цього використовується метод лінійної інтерполяції: обчислюється частка відстані між початковою та кінцевою точками, і на цій основі визначаються координати нових точок.

Після підготовки датасету знань на першому етапі пошуку попутника буде використано алгоритм пошуку попутника за допомогою фільтрації початкових та кінцевих точок маршруту. Цій підхід дозволить уникнути використання додаткових алгоритмів, які можуть вплинути на швидкість пошуку попутника. Але цей підхід також має свої певні недоліки. Вони полягають у тому що ймовірність того, що поїздку буде знайдено не велика. Мета алгоритму полягає у тому, що кожна k-та X координата буде порівняна з координатами створеного маршруту. Якщо співпадаючі початкові точки маршрутів збіглися. То буде виконано пошук по кінцевим точкам і у випадку якщо кінцева і початкова точка лежить у створеному маршруті, то користувачу буде запропоновано цю поїздку. Цей підхід дозволить уникнути використання зовнішніх алгоритмів, що має вплинути на швидкість пошуку поїздки.

У випадку, якщо поїздку не було знайдено за допомогою алгоритму з порівняння координат маршруту. То у такому випадку буде використано алгоритм k-найближчих сусідів.

На попередньому етапі обробки даних необхідно відфільтрувати отриманий масив точок. Для цього має бути обрано шаг k який дозволить фільтрувати не всі точки, а лиши кожну k-ату точку маршруту. Це дозволить пришвидшити час обчислення необхідний на пошук поїздки, а також зменшити навантаження на сервер, що має виконувати всі розрахунки.

Перший етап алгоритму kNN – обчислення відстані від цільового об’єкта (який потрібно класифікувати) до кожного з об’єктів навчального набору (вже позначених якимось класом).

Для цього для кожного маршруту в наборі обчислюється відстань до цільового об’єкта. Це може бути відстань Євкліда, або манхеттенська. Кожен об’єкт в наборі має певні ознаки або атрибути, які використовуються для обчислення цієї відстані.

Після того, як відстань від цільового об’єкта до кожного об’єкта набору обчислена, можна перейти до наступного етапу алгоритму kNN.

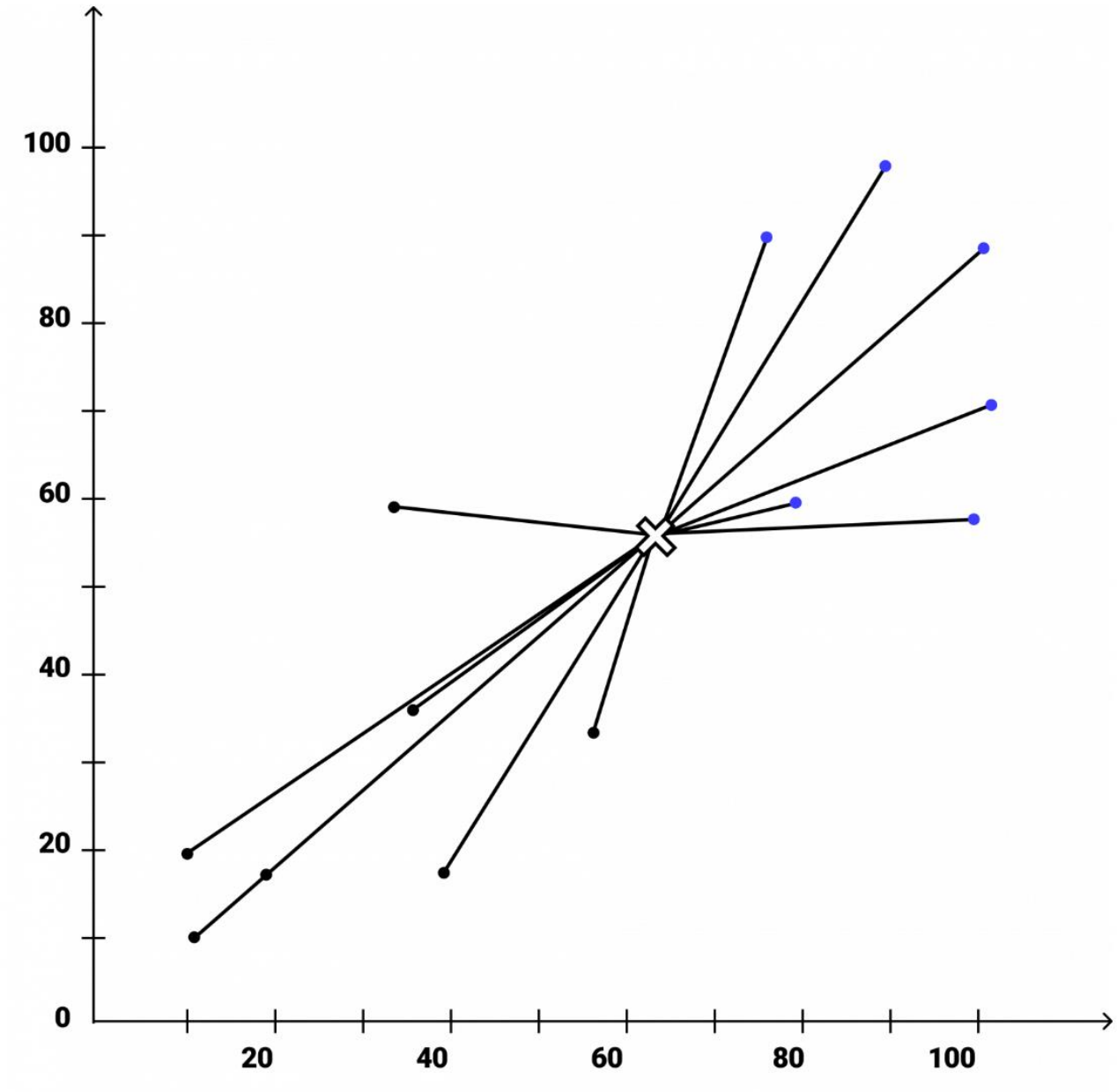


Рисунок 3.1 Розрахунок відстані до цільового маршруту

На другому етапі шляхом ітеративного процесу підбирається оптимальне значення k на основі точності прогнозів для кожного з обраних k. І саме на цьому етапі виконується підбір маршруту що співпадає за координатами, а також є можливість шукати маршрут за приближенням до існуючого маршруту.

Цей підхід суттєво відрізняється від першого запропонованого тим, що є коефіцієнт наближення, що дозволить збільшити ймовірність успішного пошуку поїздки. А також за допомогою цього підходу існує можливість запропонувати декілька маршрутів для поїздки.

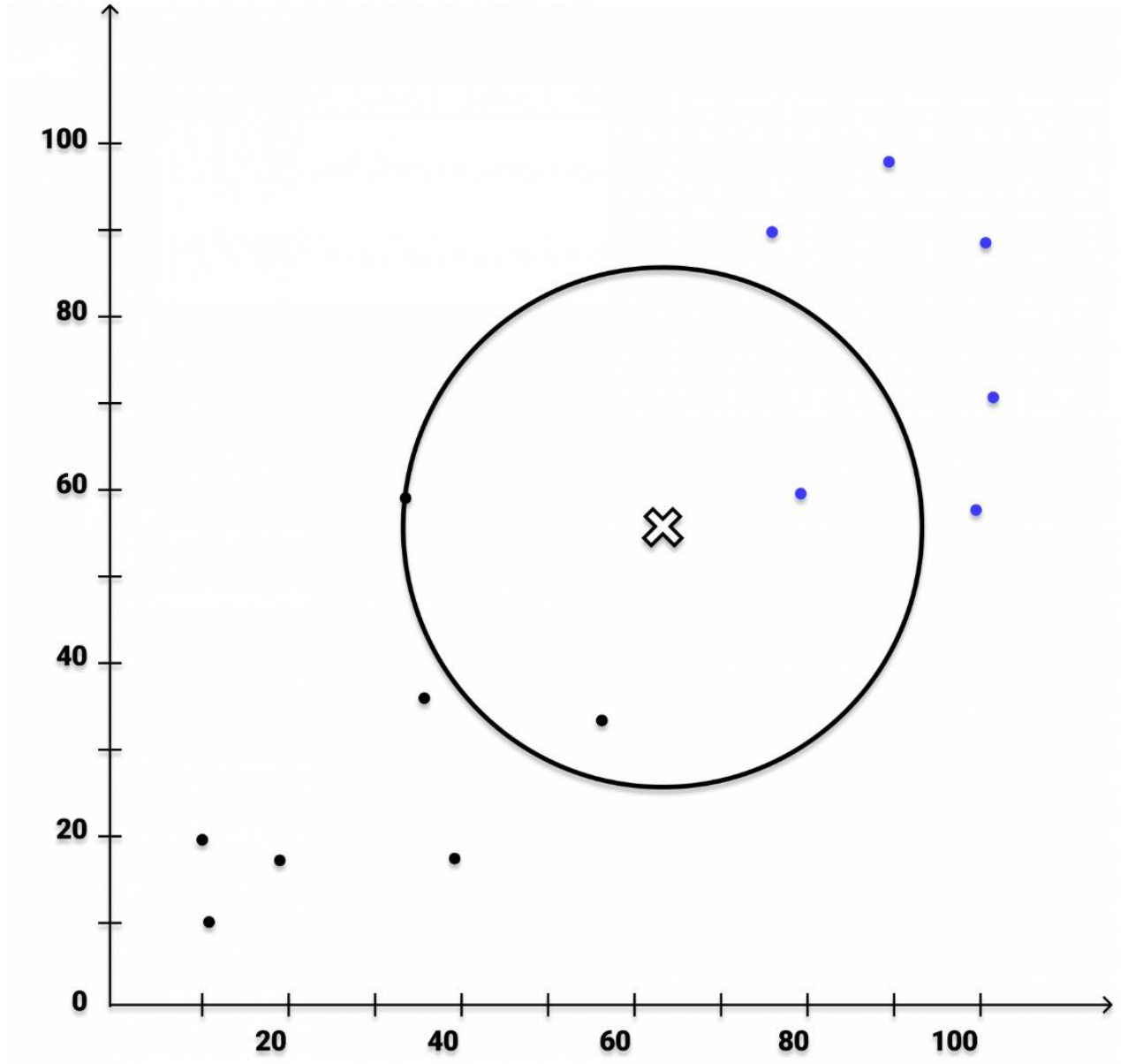


Рисунок 3.2 Вибір маршрут точки яких співпадають

Після виконання фільтрації підбору попутників з алгоритмом k-найближчих сусідів було сформовано набір маршрутів які частково, або повністю співпадають з маршрутом по якому виконується пошук. Цей алгоритм дозволить знаходити одного, або декількох попутників. Також фільтрація маршрутів за допомогою цього дозволить шукати замовлення на перевезення посилок, або передач.

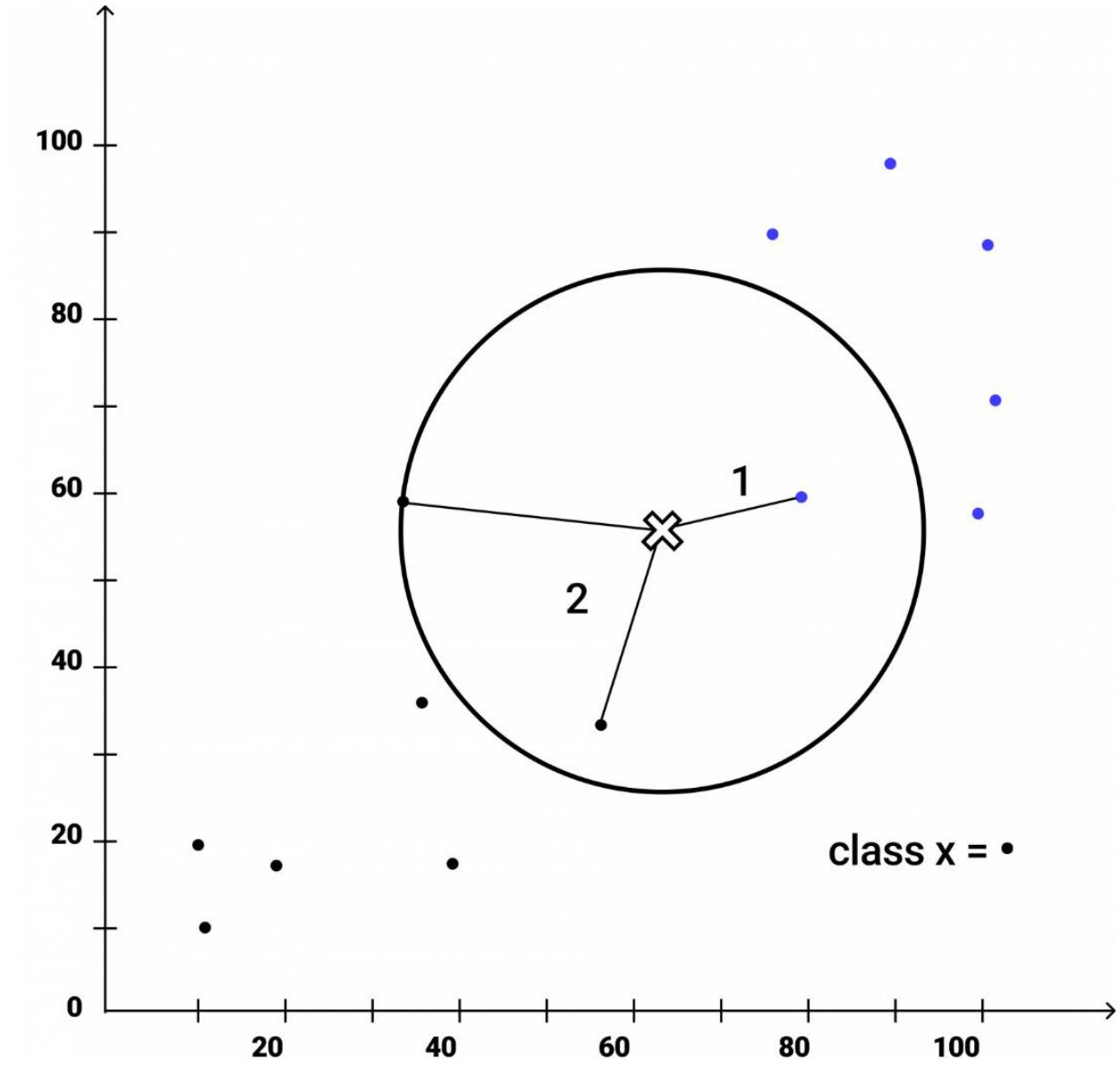


Рисунок 3.3 Приклад пошуку поїздки з відхиленням від головного маршруту

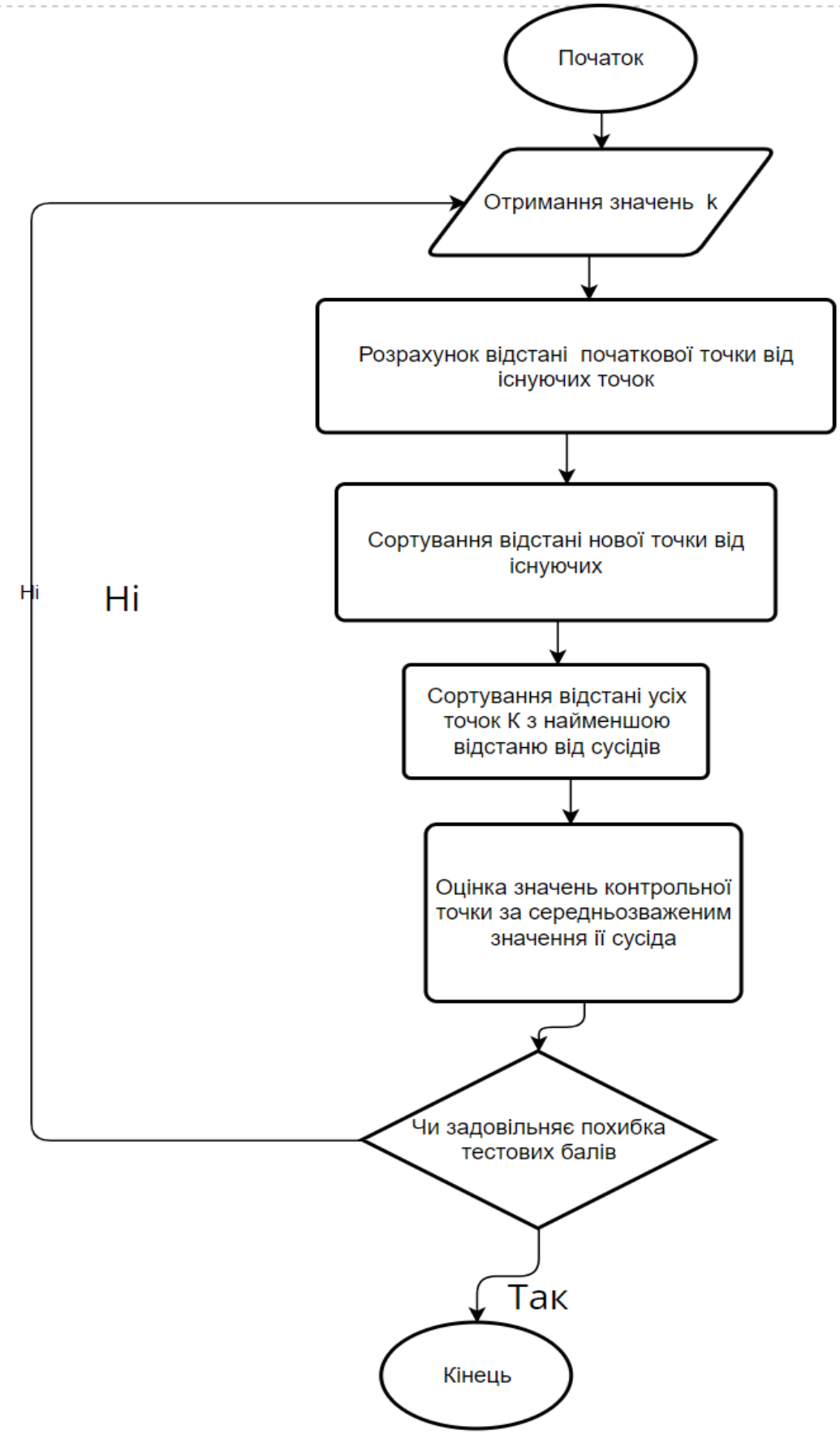


Рисунок 3.4 Схема алгоритму k-найближчих сусідів

На рисунку 3.3 можна побачити приклад знайденої поїздки для користувача, що розглядає варіант з відхилення від обраного. Пропонується водію відхилитися від маршруту у точку зустрічі з попутником. А також цей випадок працює у зворотному варіанті. Коли попутнику може бути запропоновано вийти у точку для зустрічі з водієм. Можливості відхилення від маршруту, або підбору точки зустрічі, реалізується за допомогою параметрів, що надаються користувачу під час створення поїздки для сторони водія, а також для додаткова фільтрація існує для попутника, що шукає автівку за для поїздки. На рисунку 3.4 наведено схему алгоритму пошуку маршруту за допомого алгоритму k-найближчих сусідів

На рисунку 3.5 наведено схему, що описує власний алгоритм пошук співпадаючого маршруту на основі даних введеного маршруту та маршрутів, що вже створені в системі.

На першому етапі пошуку поїздки у сервіс, що відповідає за пошук поїздки

Виконується отримання даних зі стороні користувача на автоматичний пошук поїздки. Для виконання автоматичного пошуку необхідно надати інформацію про поїздку. Обов’язковими даними початкове та кінцеве місце поїздки, а також час у який необхідно знайти поїздку. Якщо дані пройшли валідацію, то робиться запит до сторонньої API. У створеній системі запит робиться до Google Maps API.

Обраний API отримує початкове та кінцеве місце поїздки. Я якості відповіді на запит отримується побудований маршрут та координати що належать до цього маршруту. У випадку, якщо дані не надійшли, то сервіс пошуку припиняє роботу, через відсутність необхідних даних. У випадку, якщо дані було отримано, то сервіс продовжує свою роботу.

На наступному етапі на основі отриманих координат сервіс отримує координати вже існуючих поїздок, що частково співпадають з маршрутом, що введено на пошук. Цей етап потрібен для того, щоб не фільтрувати усі активні поїздки у системі.

На наступному етапі, якщо за отриманими координатами не біло знайдено поїздку зі схожими координатами, то користувачу пропонується залишити заявку на поїздку. Для того, щоб у випадку появи нової поїздки у системі, то користувачам буде запропоновано створити спільну поїздку. У випадку коли спільні маршрути було знайдено, то сервіс виконує пошук найбільш близьких маршрутів за допомогою перебору початкових і кінцевих точок наданої поїздки

та координат вже створених поїздок. На цьому етапі виконується порівняння координат існуючих поїздок з координатами початкових і кінцевих точок маршруту на який виконано запит.

Якщо співпадаючих маршрутів знайдено декілька, то перевага віддається тому, що максимально, або повність покриває маршрут на який робиться запит.

Після завершення пошуку, сервіс відправляє результат іншому сервісу, що запропонує поїздку підтвердити поїздку користувачу, або відхилити, та виконати пошук повторно за для знаходження нової поїздки.

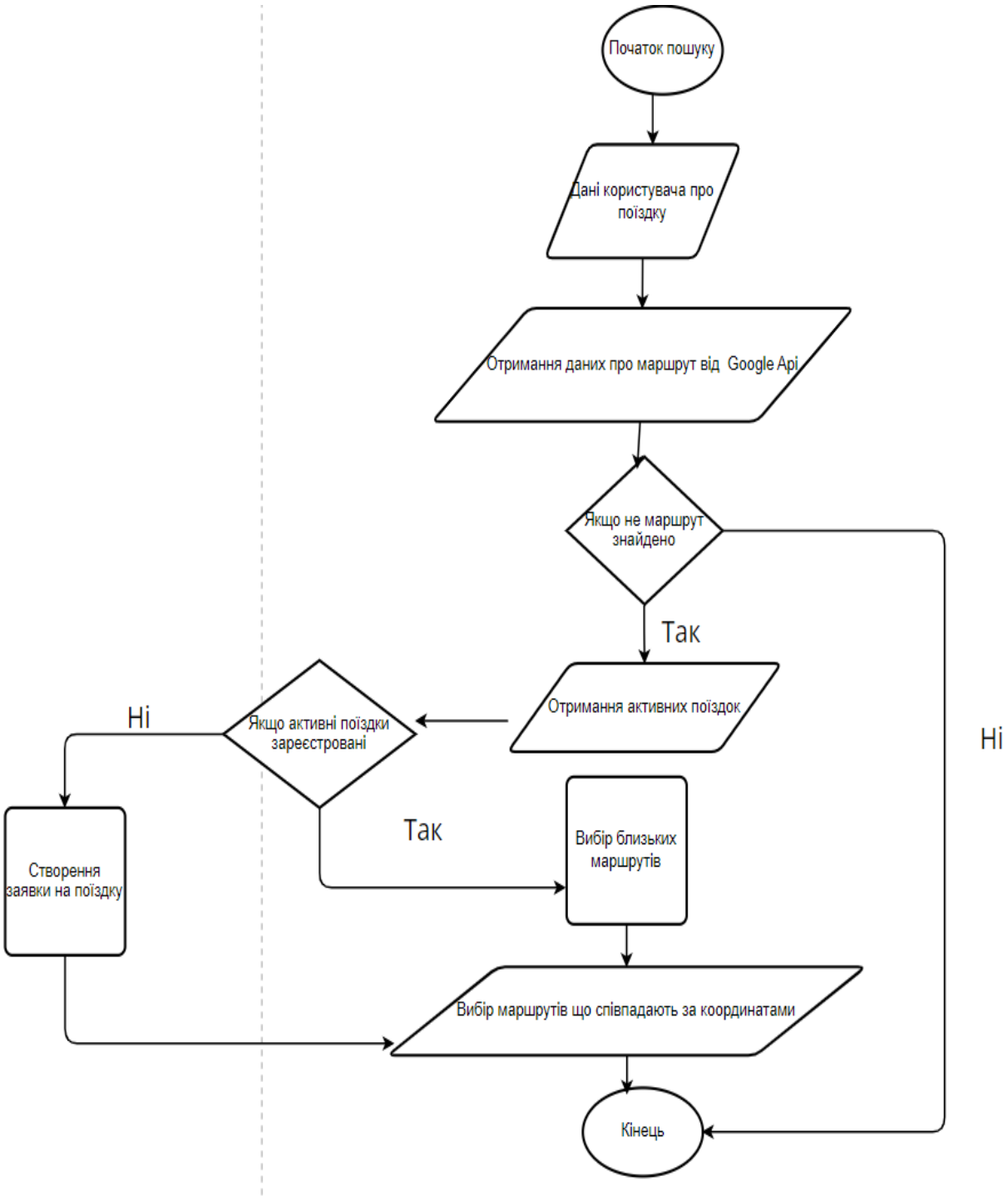


Рисунок 3.5 Алгоритм пошуку співпадаючих маршруту

3.3 Результати дослідження алгоритмів пошуку співпадаючих маршрутів.

На рисунку 3.4 та у таблиці 3.1 наведено результати пошуку маршруту для попутників за допомогою використання алгоритмі перебору маршрутів на основі маршруту на який робиться запит на пошук попутника.

Таблиця 3.1 Результат пошуку маршруту

Маршрут	Кількість координат	Час пошуку у мс
Довжина маршруту 1 – Км	28	110
Довжина маршруту 2 - Км	44	153
Довжина маршруту 3 - Км	54	171
Довжина маршруту 5 - Км	82	211
Довжина маршруту 10 – Км	198	299
Довжина маршруту 20 – Км	218	305
Довжина маршруту 50 – Км	414	330
Довжина маршруту 100 - Км	656	353
Довжина маршруту 160 – Км	736	380
Довжина маршруту 490 -Км	960	460
Довжина маршруту 980 -Км	1764	595

Рисунок 3.6 Діаграма часу виконання запитів

На рисунку 3.6 наведено результати роботи алгоритму пошуку маршруту для поїздки. Використано алгоритм перебору координат поїздок, що наближені до координат до поїздки на яку виконується запит. Алгоритмом було виконано порівняння на співпадіння існуючих у системі активних поїздок з поїздкою, що було введено під час створення заявки на поїздку.

На діаграмі можна побачити, що час виконання запиту залежить від кількості координат серед яких виконується пошук. Чим більше координат тим

більше часу необхідно на пошук поїздки. Кількість координат залежить від довжини маршруту поїздки. Навіть після проведеної інтерполяції координат маршруту в результаті, якої кількість координат було зменшено, можна побачити, що час стрімко зростає. Додатково на швидкість виконання пошуку може впливати час на запит до бази даних на отримання даних на існуючі поїздки, а також час відповіді зі сторони Google Maps API. На швидкість запиту на зовнішньої API вплинути неможливо.

Але цей алгоритм має суттєві переваги при виконанні пошуку на короткі дистанції у яких буде мало координат. Можна побачити, що необхідний час на знаходження поїздки досить маленький. Тому використання цього алгоритму для коротких поїздок є обґрунтованим рішенням.

Але використання цього алгоритму для пошуку поїздки на великі відстані є недоцільним тому що зі збільшенням довжини маршруту час необхідний на пошук співпадаючого маршруту збільшується.

У таблиці 3.2 наведено дані, що показується залежність довжини маршруту і кількості координат та часу на виконання пошуку поїздки за допомого алгоритму к найближчих сусідів.

Таблиця 3.2 Діаграма часу виконання запитів за допомого алгоритму к найближчих сусідів

Маршрут	Кількість координат	Час пошуку у мс
Довжина маршруту 1 – Км	18	143
Довжина маршруту 2 - Км	36	176
Довжина маршруту 3 - Км	58	191
Довжина маршруту 5 - Км	72	248
Довжина маршруту 8 - Км	116	271
Довжина маршруту 10 – Км	130	299

Продовження таблиці 3.2

Маршрут	Кількість координат	Час пошуку у мс
Довжина маршруту 1 – Км	18	143
Довжина маршруту 2 - Км	36	176
Довжина маршруту 3 - Км	58	191

Довжина маршруту 5 - Км	72	248
Довжина маршруту 8 - Км	116	271
Довжина маршруту 10 – Км	130	299

Рисунок 3.6 Діаграма часу виконання запитів за допомогою алгоритму k-найближчих сусідів

У таблиці 2.2 та на рисунку 3.6 представлені результати пошуку схожих маршрутів з використанням алгоритму k-найближчих сусідів (k-Nearest Neighbor, k-NN). Для перевірки ефективності алгоритму було виконано пошук по 14 маршрутах з різною довжиною та різними початковими точками. Перед запуском алгоритму дані пройшли попередню обробку, включаючи нормалізацію, фільтрацію та інтерполяцію, як описано в попередніх розділах.

Інтерпольовані дані маршрутів та їх координати були використані для забезпечення більшої точності результатів. Це дозволило алгоритму k-NN працювати з рівномірно розподіленими точками, що покращило загальну якість пошуку.

На рисунку 3.6 показано, що алгоритм k-найближчих сусідів демонструє поступове збільшення часу, необхідного для пошуку маршруту. Це збільшення

часу стає більш помітним при використанні алгоритму для коротких відстаней, де кількість координат є меншою. У таких випадках, час пошуку виявляється досить великим у порівнянні з алгоритмом порівняння точок маршруту.

Проте, при використанні алгоритму k-найближчих сусідів для пошуку маршрутів на великі відстані, наприклад, між містами, час виконання пошуку зростає не так стрімко. В деяких випадках час пошуку навіть може зменшуватися. Це пояснюється тим, що при великих відстанях алгоритм k-NN ефективніше знаходить схожі маршрути, використовуючи більшу кількість даних для аналізу.

Далі було проведено дослідження роботу алгоритму згортання маршруту (Path Folding Algorithm). Схему алгоритму наведено на рисунку 2.7. Алгоритм складається з наступних етапів:

– Початок роботи алгоритму;

– На етапі попередньої обробки даних здійснюється підготовка маршрутів. Це включає нормалізацію даних, доповнення відсутніх точок або інші необхідні дії для приведення даних до єдиного формату;

– Далі будується матриця відстаней. На цьому етапі обчислюються відстані між усіма точками двох маршрутів, після чого будується відповідна матриця відстаней. Можуть використовуватися різні метрики, такі як евклідова відстань або відстань Хаверсина;

– Далі необхідно провести ініціалізація матриці вартості. На цьому етапі Створюється та ініціалізується початкова матриця вартості, де кожен елемент відповідає вартості переходу між точками маршрутів;

– Заповнення матриці вартості. На цьому етапі матриця вартості заповнюється з використанням рекурентної формули, що дозволяє мінімізувати загальну вартість переходів. Формула враховує мінімальні вартості переходів між сусідніми точками маршрутів;

– Пошук оптимального шляху. Визначається оптимальний шлях через матрицю вартості, який мінімізує загальну вартість переходів. Цей шлях представляє найкраще вирівнювання між двома маршрутами;

– Пост обробка результатів: Здійснюється аналіз та інтерпретація отриманих результатів. Отримані дані передаються наступному сервісу що відповідає за представлення даних у доступному користувачу вигляді;

– Завершення роботи алгоритму.

На етапі побудови матриці відстаней для розрахунку відстаней було використано формулу гаверсинуса.

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d = R \cdot c$$

,

де:

- $\phi 1$ і $\phi 2$ – широти двох точок (в радіанах)
- $\lambda 1$ і λ – довготи двох точок (в радіанах) ;
- $\Delta\phi = \phi 2 - \phi 1$ – різниця широт;
- $\Delta\lambda = \lambda 2 - \lambda 1$ – різниця довгот;
- R – радіус Землі (середнє значення $R \approx 6371$км) ;
- d – відстань між двома точками по поверхні сфери.

На етапі заповнення матриці вартості використовується рекурентна формула. Ця формула є важливий компонент алгоритму, що дозволяє обчислити мінімальну вартість переходів.

$$C(i,j) = dist(i,j) + \min(C(i-1,j), C(i,j-1), C(i-1,j-1)),$$

де C(i,j)– вартість шляху до точки (i,j) а dist(i,j)– відстань між точками і та j.

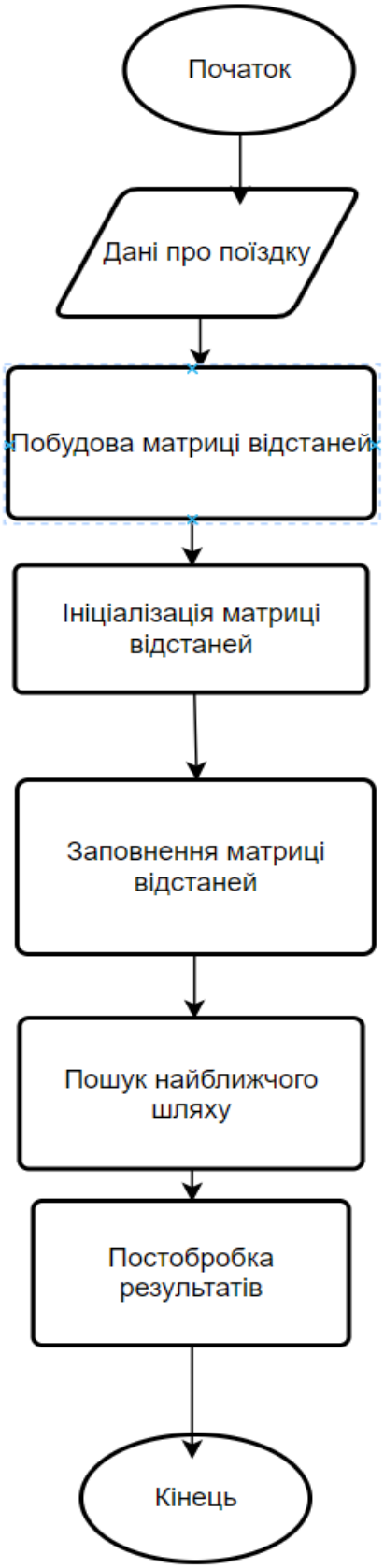


Рисунок 3.7 Схема алгоритму згортання маршруту

Таблиця 3.3 Діаграма часу виконання запитів за допомого алгоритму згортання маршруту

Маршрут	Кількість координат	Час пошуку у мс
Довжина маршруту 1 – Км	18	64
Довжина маршруту 2 - Км	36	89
Довжина маршруту 3 - Км	58	133
Довжина маршруту 5 - Км	72	125
Довжина маршруту 8 - Км	116	117
Довжина маршруту 10 – Км	130	131
Довжина маршруту 15 – Км	182	198
Довжина маршруту 20 – Км	218	287
Довжина маршруту 40 – Км	394	415
Довжина маршруту 50 – Км	484	479
Довжина маршруту 75 - Км	658	612
Довжина маршруту 100 - Км	726	783
Довжина маршруту 120 - Км	794	821
Довжина маршруту 160 – Км	918	1105
Довжина маршруту 460 -Км	1552	1634
Довжина маршруту 496 -Км	1754	1801

На рисунку 3.7 представлені експериментальні результати роботи алгоритму згортання маршрутів для знаходження спільного маршруту поїздки. У таблиці 3.3 наведені дані про відстань маршрутів та кількість координат після проведення інтерполяції.

Виходячи з отриманих даних, можна зробити висновок, що алгоритм згортання маршрутів демонструє високу ефективність. Ця ефективність проявляється у відносно швидкому знаходженні спільного маршруту на коротких відстанях до 15 кілометрів. Проте, зі збільшенням довжини маршруту час пошуку суттєво зростає. Це збільшення часу безпосередньо пов'язане з кількістю координат: чим більше координат, тим більше часу необхідно для пошуку маршруту.

Цей недолік виникає через те, що алгоритм згортання маршрутів включає кілька етапів, кожен з яких потребує значних арифметичних обчислень. Кожен

крок алгоритму додає до загального часу виконання, що робить процес пошуку більш тривалим при великій кількості координат.

Таким чином, використання цього алгоритму є виправданим для пошуку маршрутів на короткі відстані до 15-20 кілометрів, де він демонструє високу ефективність і швидкість. Однак, при довших маршрутах ефективність знижується через збільшення часу обробки, що варто враховувати при плануванні його застосування.

Рисунок 3.7 Діаграма часу виконання запитів за допомогою алгоритму згортання маршруту

Отримані дані надають докладний огляд ефективності трьох різних алгоритмів пошуку маршрутів для поїздок: алгоритму перебору маршрутів, алгоритму k-найближчих сусідів і алгоритму згортання маршрутів. Нижче подано детальні висновки:

Ефективність на коротких відстанях:

– Алгоритм перебору маршрутів: Швидкість пошуку маршрутів для коротких відстаней (до 15 км) є задовільною, але може бути повільною для

довших маршрутів через експоненційне зростання часу зі збільшенням кількості координат.

– Алгоритм k-найближчих сусідів: Показує високу швидкість пошуку на коротких відстанях, з мінімальним зростанням часу зі збільшенням довжини маршруту.

– Алгоритм згортання маршрутів: Ефективний для коротких відстаней, але вимагає значного обсягу обчислень та часу для довших маршрутів.

Зростання часу зі збільшенням довжини маршруту:

– Алгоритм перебору маршрутів: Показує найбільше зростання часу пошуку зі збільшенням довжини маршруту через необхідність перебору всіх можливих комбінацій координат.

– Алгоритм k-найближчих сусідів: Демонструє менше зростання часу порівняно з алгоритмом перебору маршрутів, але все ж відчутне при збільшенні довжини маршруту.

– Алгоритм згортання маршрутів: Також показує зростання часу пошуку зі збільшенням довжини маршруту, але менше порівняно з алгоритмом перебору маршрутів.

Підвищення продуктивності:

– Алгоритм згортання маршрутів: Може підвищити продуктивність на коротких відстанях, але потребує уваги при плануванні застосування для довших маршрутів через зростання часу обробки.

– Інші фактори: Обробка даних та запити до зовнішнього API (наприклад, Google Maps API) також можуть впливати на швидкість пошуку маршрутів.

Таким чином, вибір оптимального алгоритму для пошуку маршрутів повинен базуватися на конкретних потребах системи, враховуючи довжину маршруту, кількість координат та потреби у швидкості та продуктивності.
У випадку, коли від системи вимагається більша точність знайдених маршрутів незалежно від відстані, то краще використовувати алгоритму згортання маршруту. Але коли від системи вимагаються висока продуктивність, то краще використовувати декілька алгоритмів, які будуть спрацьовувати в залежності від кількості координат та довжини маршруту. У такому варінті використання краще

обирати серед алгоритму фільтрації маршрутів для поїздок на короткі відстані, та алгоритму k найближчих сусідів (kNN).

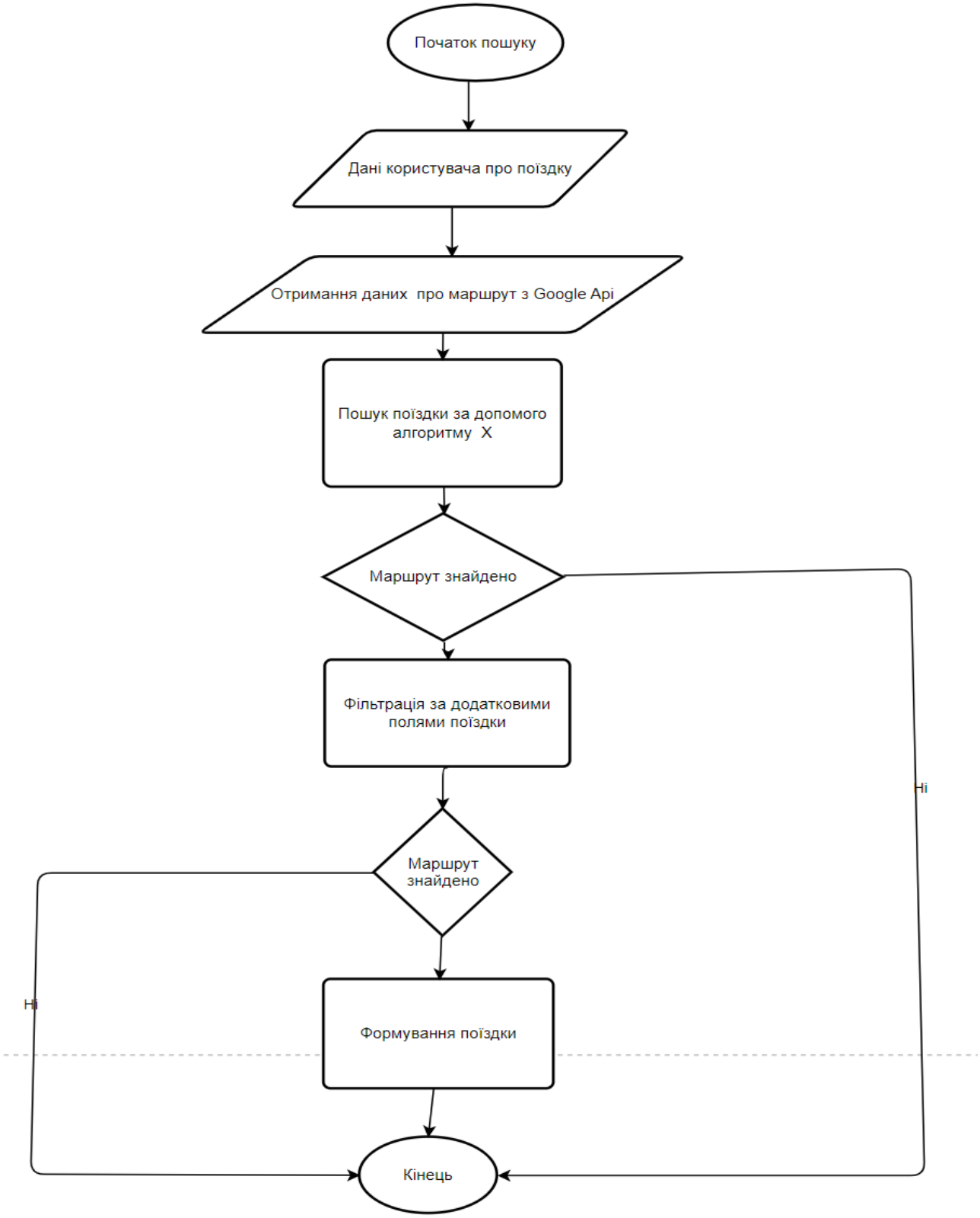


Рисунок 3.8 Діаграма часу виконання запитів за допомогою алгоритму згортання маршруту

На рисунку 3.8 наведено блок схему загальної схеми алгоритму пошуку автомобільних попутників у с системі. На першому етапі отримуються дані про поїздку від користувача. Далі початкова і кінцева точка маршруту передається у Google Api. Після обробки даних на сторонній API отримуються координати маршруту.

На наступному етапі отриманий масив координати передається обраному алгоритму пошуку співпадаючих поїздок. Цей алгоритм відповідає за пошук необхідної поїздки за отриманими координатами. У випадку коли співпадаючі маршрути не було знайдено, то пошук припиняється. Якщо співпадаючі маршрути було знайдено, то далі виконується фільтрація за додатковим параметрами, що можуть бути вказані під час створення поїздки або заявки на поїздку.

Якщо після додаткової фільтрації поїздку було знайдено, то користувачу буду запропоновано підтвердити поїздку або обрати альтернативну якщо знайдено декілька варіантів. Але якщо, поїздку не було знайденоб то користувачеві буде запропоновано створити заявку на поїздку.

3.4 Обґрунтування вибору API

Під час пошуку Api для сервісу пошуку автомобільних попутників було розглянуто декілька варіантів. Але Here Maps Routing API та Mapbox Directions API показали найкращі результати за швидкістю отримання відповіді на запит у порівнянні з OpenRouteService API.

Google Maps Directions API - це один з найпопулярніших API для отримання маршруту між двома або більше точками.

Api від Google показує досить ретельно кожен крок маршруту, містить докладні інструкції, відстань, тривалість та координати. Також він пропонує інтеграцію з іншими сервісами Google, такими як Places API для пошуку цікавих місць по маршруту. Також дає інформація про поточний стан трафіку, затори та можливі затримки. Додатково є можливість оптимізувати маршрут з кількома зупинками.

Переваги використання Google Maps Directions API:

– Висока точність даних завдяки великій базі користувачів та постійно оновлюваній інформації;

– Надання детальних інструкцій для кожного кроку маршруту;

– Підтримка різних видів транспорту та інтеграція з іншими API Google.

Недоліки використання Google Maps Directions API:

– Висока вартість використання при великій кількості запитів;

– Обмеження на кількість запитів, що можуть бути зроблені за день.

Mapbox Directions API - це інструмент для отримання маршруту між точками, що базується на відкритих даних OpenStreetMap.

Особливості:

– Є підтримка автомобільних, пішохідних, велосипедних маршрутів.

– Існує можливість налаштовувати маршрути за допомогою власних правил та обмежень.

– Інтеграція з інструментами для створення інтерактивних карт.

– Підтримка повернення координат у форматі GeoJSON, що спрощує інтеграцію з іншими інструментами.

Переваги:

– Широкі можливості налаштування маршрутів та інтеграції з іншими інструментами Mapbox.

– Доступніші тарифи порівняно з Google Maps.

– Використання даних OpenStreetMap, які постійно оновлюються спільнотою.

Недоліки:

– Може бути менш точною в деяких регіонах порівняно з Google Maps.

– Менше даних про трафік у реальному часі.

3.4.1 Чому варто вибрати Google Maps Directions API

Точність даних для сервісу пошуку автомобільних попутників і докладність інструкцій є критичними для вашого сервісу, а також потрібна інтеграція з іншими сервісами Google, то Google Maps Directions API буде найкращим вибором. Він забезпечить високу якість маршрутів, актуальну інформацію про трафік та простоту інтеграції, що важливо для створення надійного і зручного для користувачів сервісу пошуку автомобільних попутчиків.

3.5 Обґрунтування вибору СУБД

Вибір системи управління базами даних (СУБД) є дуже важливим кроком під час розробки сервісу пошуку автомобільних попутників. Відповідна СУБД має забезпечити високий рівень продуктивності, надійність, масштабованість і функціональність для ефективного оброблення даних. Після ретельного аналізу різних варіантів було прийнято рішення використовувати PostgreSQL як основну СУБД. Це рішення базується на порівнянні переваг і недоліків різних СУБД, а також на специфічних вимогах до проекту.

PostgreSQL є потужною реляційною системою управління базами даних (СУБД) з відкритим кодом, яка пропонує широкий спектр функцій та можливостей. Її основною перевагою є відкритий код, що дозволяє користувачам вільно використовувати, змінювати і розширювати систему відповідно до своїх потреб. Висока продуктивність PostgreSQL досягається завдяки оптимізації запитів, ефективному управлінню індексами та використанню сучасних алгоритмів для обробки даних.

PostgreSQL також відзначається високою масштабованістю, що дозволяє їй ефективно працювати з великими обсягами даних і великою кількістю одночасних користувачів. Система підтримує горизонтальне масштабування і реплікацію, що робить її придатною для використання в великих проектах. Крім того, PostgreSQL пропонує багатий набір функцій, включаючи підтримку складних запитів, транзакцій, тригерів, збережених процедур та розширень, що додають додаткові можливості.

Також ця СУБД має додаткове розширення PostGis. PostGIS - це розширення для бази даних PostgreSQL, яке дозволяє зберігати та опрацьовувати географічні та геопросторові дані. У сервісі пошуку попутників, PostGIS може бути використано для низки цілей:

– Збереження геоданих користувачів. PostGIS дозволяє зберігати географічні дані про користувачів, такі як їхні маршрути, місцезнаходження та інші важливі геопросторові атрибути;

– Аналіз та обробка геоданих: PostGIS надає широкий спектр геопросторових функцій, таких як визначення відстані між точками, знаходження найближчих об'єктів, обчислення зон доступності та багато інших,

що дозволяє виконувати різноманітні аналітичні операції над геоданими користувачів.

– Відображення геоданих на карті: PostGIS може бути використано для відображення географічних об'єктів на карті, що дозволяє візуалізувати маршрути користувачів та їхнє місцезнаходження для полегшення вибору попутників.

– Розрахунок відстаней та часу подорожі: За допомогою PostGIS можна розраховувати відстані між точками на маршруті, а також оцінювати час подорожі, що дозволяє користувачам отримувати більш точну та інформативну інформацію про потенційних попутників.

Microsoft SQL Server є комерційною реляційною СУБД, відомою своєю інтеграцією з продуктами Microsoft, такими як Azure і Visual Studio. Вона забезпечує високу продуктивність завдяки оптимізації запитів і підтримці індексів, що дозволяє швидко обробляти великі обсяги даних. Тісна інтеграція з іншими продуктами Microsoft робить SQL Server привабливим вибором для користувачів екосистеми Microsoft, забезпечуючи безшовну роботу з іншими інструментами та сервісами.

SQL Server отримує надійну підтримку і регулярні оновлення від Microsoft, що забезпечує стабільність, безпеку і додавання нових функцій. Однак висока вартість ліцензії може бути значним фактором для невеликих компаній або стартапів, які шукають більш економічні рішення. Крім того, SQL Server краще працює в середовищі Microsoft, що може обмежувати гнучкість використання в інших середовищах.

Реляційні бази даних краще підходять для сервісу пошуку попутників порівняно з нереляційними базами даних з-за потреби в точності, цілісності та структурованості даних. У такому сервісі, де інформація про користувачів, їх маршрути та взаємозв'язки важливі, реляційні бази даних надають зручний механізм для зберігання та управління цими даними, забезпечуючи точність та консистентність інформації. Крім того, вони мають потужні мови запитів, такі як SQL, що дозволяють ефективно виконувати різноманітні запити та аналізувати дані.

У порівнянні з нереляційними базами даних, де структура даних може бути менш структурованою та менше контролюватися, реляційні бази даних забезпечують більшу гнучкість у моделюванні складних взаємозв'язків між

даними та можливість здійснення складних запитів. Вони також гарантують високий рівень цілісності даних, що є критичним для забезпечення надійності та безпеки інформації про маршрути та користувачів у сервісі пошуку попутників. Таким чином, реляційні бази даних є більш підходящим вибором для цього конкретного випадку в порівнянні з нереляційними базами даних.

3.6 Логічне та фізичне моделювання даних системи

Нижче на рисунку 3.9 наведено опис полів для діаграми, що наведено на рисунку 3.8

а) Organization – таблиця організація. Вона містить інформацію про тенант організації. Нижче наведено п***о***ля цієї таблиці:

– Id: uuid - Первинний ключ. Унікальний ідентифікатор організації.

– Title: Назва організації (string, обов'язкове поле). Вказує на офіційну назву організації.

– Country: Країна організації (string, обов'язкове поле). Вказує на країну, де зареєстрована організація.

– Phone: Номер телефону організації (string?, необов'язкове поле). Контактний номер для зв'язку з організацією.

– Email: Електронна пошта організації (string?, необов'язкове поле). Контактна електронна пошта для комунікації.

– Address: Адреса організації (string, обов'язкове поле). Фізична адреса організації.

– ContactPerson: Контактна особа організації (string, обов'язкове поле). Особа, з якою можна зв'язатися щодо питань організації.

– RegistrationDate: Дата реєстрації організації (DateTime, обов'язкове поле). Дата, коли організація була офіційно зареєстрована.

– IsActive: Чи активна організація (bool, обов'язкове поле, за замовчуванням true). Вказує на статус активності організації.

б) Таблиця: Car (Автомобіль). Містить детальну інформацію про автомобіль доданий у систему. Нижче наведено перелік полів цієї таблиці:

– Id: uuid - Первинний ключ. Унікальний ідентифікатор автомобіля.

– CarModel: Модель автомобіля (string, обов'язкове поле).

– CarName: Назва автомобіля (string, необов'язкове поле).

– CarType: Тип автомобіля (наприклад, седан, позашляховик) (CarType?, необов'язкове поле).

– CarNumber: Реєстраційний номер автомобіля (string, обов'язкове поле).

– Manufacturer: Виробник автомобіля (string, обов'язкове поле).

– Year: Рік випуску автомобіля (int, обов'язкове поле).

– Color: Колір автомобіля (string, необов'язкове поле).

– EngineType: Тип двигуна автомобіля (string, необов'язкове поле).

– Mileage: Пробіг автомобіля (double?, необов'язкове поле).

– Price: Ціна автомобіля (decimal?, необов'язкове поле).

– RegistrationDate: Дата реєстрації автомобіля (DateTime?, необов'язкове поле).

– VIN: VIN (Vehicle Identification Number) автомобіля (string, необов'язкове поле).

– TransmissionType: Тип трансмісії (механічна, автоматична) (string, необов'язкове поле).

– FuelType: Тип палива (бензин, дизель, електрика) (string, необов'язкове поле).

– SeatingCapacity: Кількість місць для сидіння (int?, необов'язкове поле).

– HorsePower: Потужність двигуна в кінських силах (int?, необов'язкове поле).

– OrganizationId: uuid - Зовнішній ключ. Ідентифікатор організації, якій належить автомобіль.

в) Таблиця: Drive (Поїздка). У цієї таблиці міститься інформація про поїздки створену або виконану у системі пошуку автомобільних попутників. Нижче наведено перелік полів цієї таблиці:

– Id: uuid - Первинний ключ. Унікальний ідентифікатор поїздки.

– DeparturePlace: Місце відправлення поїздки (string, обов'язкове поле).

– ArrivalPlace: Місце прибуття поїздки (string, обов'язкове поле).

– DriveStartTime: Час початку поїздки (DateTime, обов'язкове поле).

– DriveEndTime: Час закінчення поїздки (DateTime, обов'язкове поле).

– PlaceAmount: Кількість місць для пасажирів (int, обов'язкове поле).

– IsBaggageAvailable: Наявність можливості перевезення багажу (bool, обов'язкове поле).

– Price: Вартість поїздки (double, обов'язкове поле).

– DriverId: Унікальний ідентифікатор водія (Guid, обов'язкове поле).

– Driver: Об'єкт водія (Driver, обов'язкове поле).

– CarId: Унікальний ідентифікатор автомобіля (Guid, обов'язкове поле).

– Car: Об'єкт автомобіля (Car, обов'язкове поле).

– Passengers: Колекція пасажирів (ICollection<Passenger>?, необов'язкове поле).

– RouteCoordinates: Координати маршруту (List<Point>?, необов'язкове поле).

– OrganizationId: Унікальний ідентифікатор організації (Guid, обов'язкове поле).

– Organization: Об'єкт організації (Organization, обов'язкове поле).

г) Таблиця: Driver (Водій). Ця таблиця містить інформацію про водія, що створив обліковий запис у системі пошуку автомобільних попутників. Нижче наведено перелік полів ціє сутності:

– Id: uuid - Первинний ключ. Унікальний ідентифікатор водія.

– DriverName: Ім'я водія (string, обов'язкове поле).

– LicenseNumber: Номер водійського посвідчення (string, обов'язкове поле).

– ContactNumber: Контактний номер телефону водія (string, обов'язкове поле).

– Address: Адреса водія (string?, необов'язкове поле).

– ExperienceYears: Кількість років досвіду водія (int?, необов'язкове поле).

– OrganizationId: uuid - Зовнішній ключ. Унікальний ідентифікатор організації, до якої належить водій.

д) Таблиця: DriveRequest (Заявка на поїздку). Ця сутність описує заявку, на поїздку, що створює користувач системи з роллю пасажир. Нижче наведено перелік полів цієї сутності:

– Id: uuid - Первинний ключ. Унікальний ідентифікатор заявки.

– RequestDate: Дата заявки (DateTime?, необов'язкове поле).

– Status: Статус заявки (string?, необов'язкове поле).

– Message: Повідомлення в заявці (string?, необов'язкове поле).

– DeparturePlace: Місце відправлення поїздки (string, обов'язкове поле).

– ArrivalPlace: Місце прибуття поїздки (string, обов'язкове поле).

– DriveStartTime: Час початку поїздки (DateTime, обов'язкове поле).

– DriveEndTime: Час закінчення поїздки (DateTime, обов'язкове поле).

– PassengerId: Унікальний ідентифікатор пасажира (Guid, зовнішній ключ, обов'язкове поле).

– DriveId: Унікальний ідентифікатор поїздки (Guid?, зовнішній ключ, необов'язкове поле).

– Drive: Об'єкт поїздки (Drive?, необов'язкове поле).

– OrganizationId: Унікальний ідентифікатор організації (Guid, зовнішній ключ, обов'язкове поле).

е) Таблиця: Passenger (Пасажир). Ця таблиця описує сутність пасажир. Нижче наведено перелік полів:

– Id: uuid - Первинний ключ. Унікальний ідентифікатор пасажира.

– Name: Ім'я пасажира (string, обов'язкове поле). Вказує на ім'я пасажира.

– Email: Електронна пошта пасажира (string, обов'язкове поле). Контактний e-mail пасажира.

– PhoneNumber: Номер телефону пасажира (string, обов'язкове поле). Контактний номер телефону пасажира.

– DriveId: Унікальний ідентифікатор поїздки (Guid?, зовнішній ключ, необов'язкове поле). Вказує на поїздку, якщо вона є.

– OrganizationId: Унікальний ідентифікатор організації (Guid, зовнішній ключ, обов'язкове поле). Вказує на організацію, до якої належить пасажир.

ж) Таблиця: Point (Точка). Ця сутність описує координату, що має створена поїздка:

– Id: uuid - Первинний ключ. Унікальний ідентифікатор точки.

– Latitude: Широта точки (double, обов'язкове поле).

– Longitude: Довгота точки (double, обов'язкове поле).

– DriveId: Унікальний ідентифікатор поїздки (Guid?, зовнішній ключ, необов'язкове поле). Вказує на унікальний ідентифікатор поїздки, до якої належить ця точка.

– Опис зв’язків між таблицями наведено нижче:

– Organization має зв'язок один-до-багатьох з таблицями Car, Driver, Drive, DriveRequest, Passenger.

– Drive має зв'язок один-до-багатьох з таблицею DriveRequest.

- DriveRequest має зв'язок один-до-багатьох з таблицею Point.
- Car має зв'язок один-до-багатьох з таблицею Drive.
- Driver має зв'язок один-до-багатьох з таблицею Drive.
- Passenger має зв'язок один-до-багатьох з таблицею DriveRequest

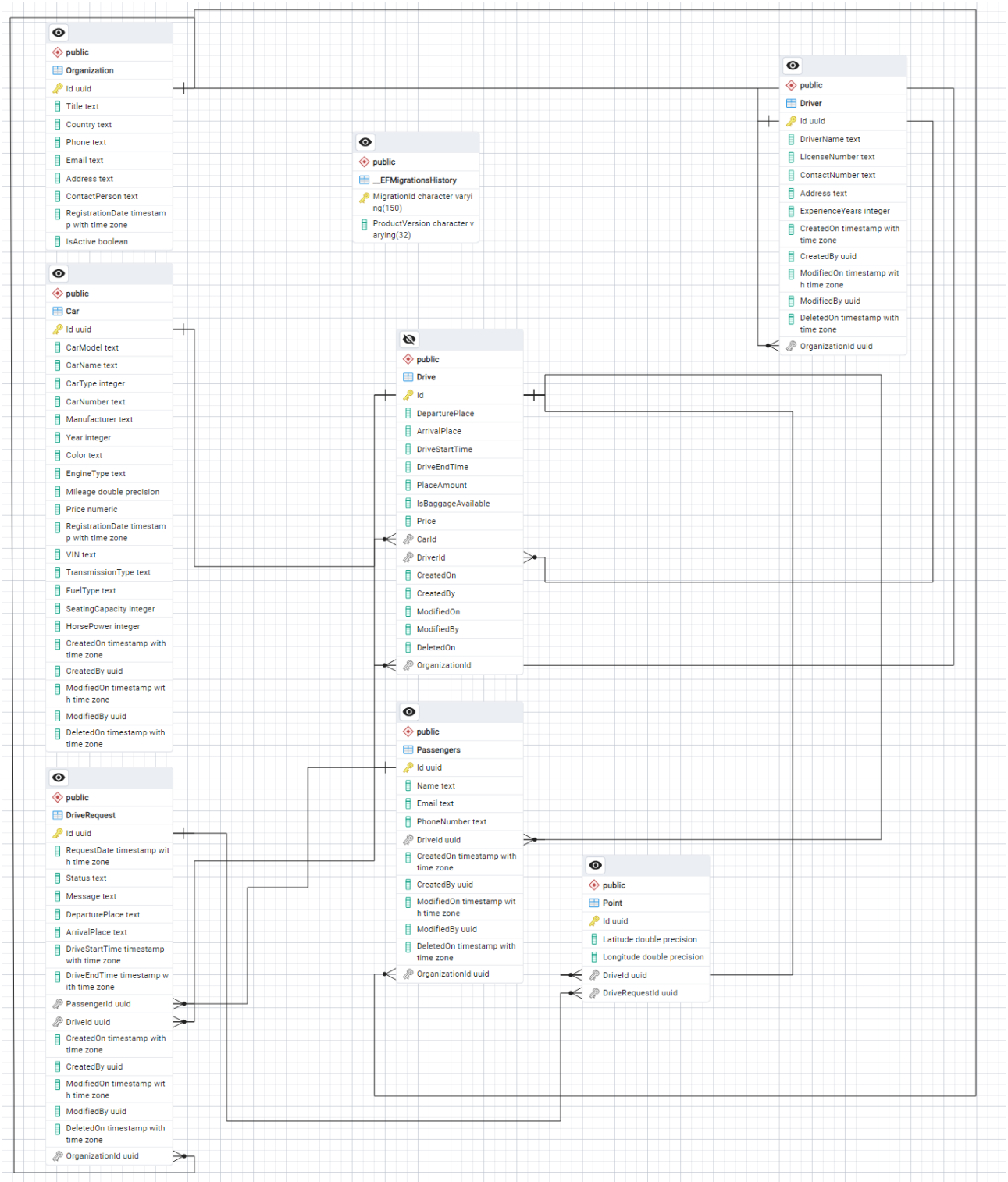


Рисунок 3.9 – Схема фізичної моделі даних для сервісу «З пошуку автомобільних попутників»

Створена фізична модель бази даних для системи пошуку автомобільних попутників є важливою для кількох аспектів роботи з реляційними системами. Вона забезпечує чітке уявлення про те, як дані будуть зберігатися, включаючи типи даних, розміри полів та індекси. Вона допомагає зрозуміти взаємозв'язки між таблицями, що є критичним для забезпечення цілісності даних та оптимізації продуктивності системи. Знання фізичної структури також дозволяє ефективно виконувати запити та визначати необхідність індексів, що сприяє уникненню надмірності даних. Крім того, модель служить важливою документацією, яка полегшує підтримку та розвиток системи, а також визначає правила для міграцій і оновлень баз даних, забезпечуючи безперервну роботу без втрати даних.

На рисунку 3.9 наведено логічну схему бази даних для сервісу пошуку автомобільних попутників. Логічна модель бази даних для системи пошуку автомобільних попутників виконує кілька важливих функцій:
Модель дозволяє чітко визначити, які дані будуть зберігатися, як вони будуть організовані, і як вони взаємодіють між собою. Це допомагає уникнути дублювання даних і забезпечує цілісність інформації.

а) За допомогою логічної моделі можна оптимізувати запити до бази даних, що забезпечує швидкий доступ до необхідної інформації. Це важливо для швидкого пошуку доступних поїздок, водіїв і пасажирів.

б) Модель допомагає встановити правила для введення даних, такі як унікальність, обов'язковість полів та референтну цілісність. Це гарантує, що всі дані в базі є точними і взаємопов'язаними.

в) Логічна модель дозволяє легко розширювати систему, додаючи нові сутності або атрибути без порушення існуючої структури. Це важливо для розвитку системи та впровадження нових функцій.

г) Чітка структура бази даних дозволяє легше реалізувати механізми контролю доступу та захисту даних. Можна визначити, які користувачі матимуть доступ до певної інформації, що підвищує загальну безпеку системи.

д) Добре спроектована логічна модель полегшує аналіз даних та створення звітів. Це дозволяє адміністраторам системи отримувати цінну інформацію про роботу сервісу, поведінку користувачів та інші ключові показники.

е) Завдяки швидкому доступу до даних та їх правильній організації користувачі можуть швидко знаходити необхідну інформацію про поїздки, що підвищує задоволеність від використання системи.

Загалом, ця логічна модель забезпечує чітку організацію даних, що дозволяє сервісу пошуку автомобільних попутників ефективно обробляти запити, координувати поїздки, підтримувати зв'язок між водіями та пасажирами, а також надавати надійну та зручну платформу для користувачів. Завдяки такій моделі, сервіс може забезпечити високий рівень обслуговування, безпеки та задоволення потреб користувачів, сприяючи зручності та економічній вигоді від спільних поїздок.

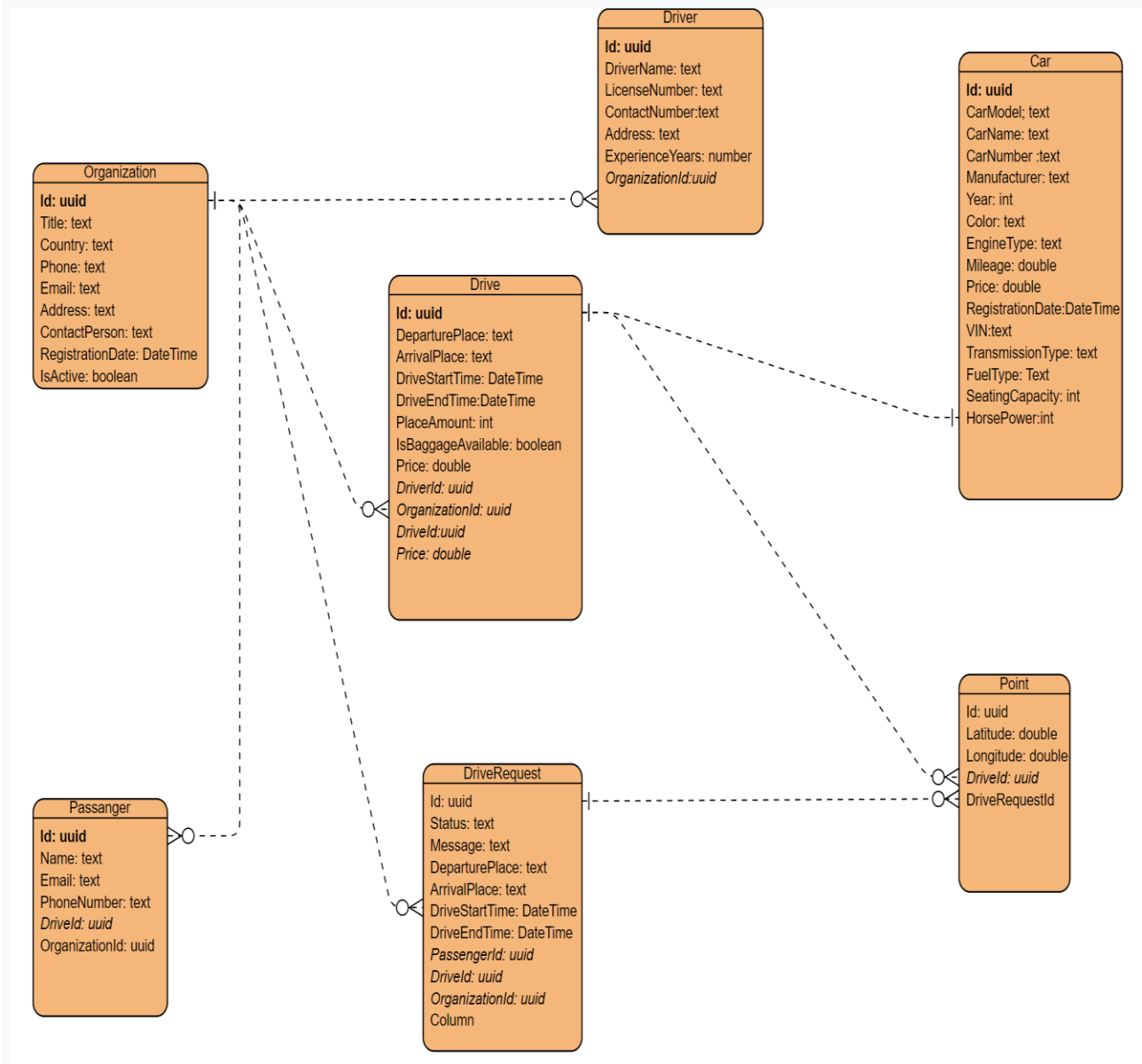


Рисунок 3.10 – Логічна модель даних для сервісу «З пошуку автомобільних

попутників»

Після того, як було проведено фізичне моделювання та побудови відповідної схеми був визначено перелік сутностей та їх характеристик:

3.6.1 Вирішення питань високого навантаження бази даних

У мультитенантній системі передбачено два види підписки:

Спільна база даних для кількох тенантів (Shared Database): Декілька організацій використовують одну спільну базу даних.

Окрема база даних для кожної організації (Dedicated Database): Кожна організація має власну окрему базу даних.

Для забезпечення високої продуктивності та надійності системи необхідно розробити стратегії та архітектуру для вирішення питань високого навантаження на базу даних.

У роботі розглянуто різні підходи, щодо зменшення часу виконання запитів у системі:

а) Шардінг

– Shared Database: Використання горизонтального шардінгу, розподіляючи дані різних тенантів по різних шардінгових базах даних.

– Dedicated Database: Необхідно налаштувати кожну базу даних для окремого тенанта, що дозволить розподілити навантаження між різними базами даних.

б) Кешування

– Використання in-memory кешування (наприклад, Redis) для зменшення кількості запитів до бази даних.

– Кешування часто запитуваних даних або результатів складних запитів.

в) Індексація

– Оптимізація індексів для прискорення запитів.

– Використання композитних індексів для часто використовуваних комбінацій колонок у WHERE та JOIN умовах.

г) Реплікація

– Налаштування реплікації бази даних для підвищення доступності та розподілу навантаження на читання.

– Використання реплік для виконання запитів на читання, залишаючи основну базу даних для запису.

д) Планувальник запитів

– Використання планувальника запитів для оптимізації складних запитів.

– Розподіл складних запитів на декілька простих запитів з проміжним збереженням результатів.

е) Підтримка горизонтального масштабування

– Додавання нових серверів баз даних в залежності від збільшення навантаження.

– Використання контейнеризації (наприклад, Docker) для легкого масштабування баз даних.

На рисунку 3.11 наведено діаграму розгортання. Ця діаграма розгортання показує архітектуру системи з використанням Load Balancer, Application Server з Sharding Proxy та кількома шардованими базами даних.

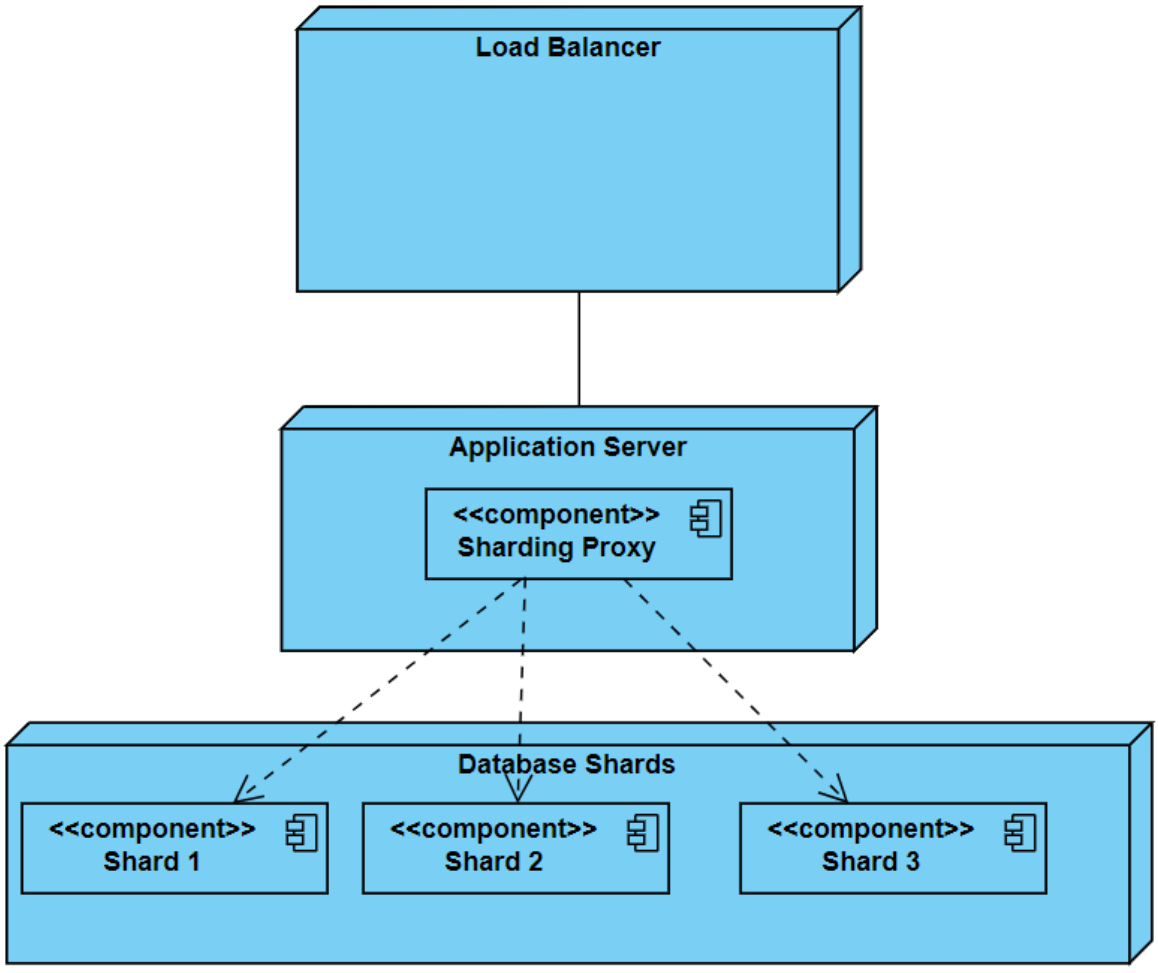


Рисунок 3.11 Діаграма розгортання для спільної бази даних

Нижче наведено опис кожного елемента на діаграмі:

а) Load Balancer (Розподільник навантаження):

– Тип: Node (вузол).

– Опис: Компонент, який приймає всі вхідні запити від клієнтів і розподіляє їх між екземплярами сервера додатків. Це допомагає забезпечити балансування навантаження та підвищити доступність і надійність системи.

б) Application Server (Сервер додатків):

– Тип: Node (вузол).

– Опис: Сервер, який обробляє бізнес-логіку додатку. Він приймає запити від Load Balancer і направляє їх до відповідних компонентів для подальшої обробки.

– Компонент Sharding Proxy. Проксі-сервер, який розподіляє запити до відповідних шард-баз даних. Він забезпечує логіку шардінгу, що дозволяє розподілити дані між кількома базами даних для підвищення продуктивності та масштабованості.

в) Database Shards (Шардовані бази даних):

– Тип: Node (вузол) для сервера баз даних, який містить кілька шард-баз даних.

– Опис: Кожен шард є окремою базою даних, яка зберігає частину загальних даних системи. Це дозволяє розподілити навантаження на базу даних та забезпечити горизонтальну масштабованість.

3.6.2 Загальна архітектура

– Load Balancer отримує запити від клієнтів та розподіляє їх на Application Server.

– Application Server містить Sharding Proxy, який обробляє запити та визначає, до якого шару бази даних направити запит.

– Sharding Proxy взаємодіє з відповідними Shard базами даних для виконання запитів та повернення результатів.

– Shard 1, Shard 2, Shard 3 є компонентами баз даних, що розподіляють дані системи для забезпечення продуктивності та масштабованості.

Ця архітектура допомагає розподілити навантаження, забезпечити високу доступність та підвищити продуктивність системи шляхом розподілу даних на кілька баз даних.

3.6.3 Переваги такої системи

– Розподіл даних між кількома шардами зменшує навантаження на кожен окремий шард, що покращує загальну продуктивність системи.

– Легко додавати нові шарди для обробки збільшених обсягів даних і навантаження без впливу на існуючі дані.

– Якщо один з шардів виходить з ладу, інші шарди можуть продовжувати працювати, забезпечуючи безперебійну роботу системи.

– Шардинг дозволяє ефективно обробляти великі обсяги даних, розподіляючи їх між кількома базами даних.

3.6.4 Недоліки такої системи

– Управління кількома шардами потребує складної логіки для розподілу і синхронізації даних, що може ускладнити розробку та підтримку системи.

– Розподілені транзакції між шардами можуть бути складними для реалізації і можуть вплинути на цілісність даних.

– Забезпечення консистентності даних між різними шардами може бути складним завданням, особливо при оновленнях даних.

– Може виникати додаткова затримка через необхідність маршрутизації запитів через Sharding Proxy і розподіл їх між шардами.

– Якщо кілька тенантів використовують одну базу даних, то час обробки запитів може збільшуватися при високому навантаженні, що знижує продуктивність системи для всіх тенантів.

Ця архітектура підходить для систем з не великим обсягом даних і не дуже високим навантаженням, де важливо забезпечити високу продуктивність і масштабованість. Однак складність управління шардами і розподіл транзакцій між ними може вимагати додаткових зусиль з розробки та підтримки системи.

На рисунку 3.12 наведено діаграми розгортання мультитенантної системи пошуку автомобільних попутників.

Ця діаграма показує архітектуру розгортання системи пошуку автомобільних попутників, побудованої за моделлю мультитенантної системи, де для кожного орендаря створюється окрема база даних.

Компоненти діаграми

а) Load Balancer він відповідає за балансування навантаження розподіляє вхідні запити користувачів між кількома серверами додатків для забезпечення рівномірного розподілу навантаження та високої доступності системи.

б) Application Server це сервер додатків обробляє запити, надані Load Balancer. Включає логіку мультитенантності для визначення, до якої бази даних орендаря слід звернутися для обробки конкретного запиту. Multi-Tenant-Logic це логіка мультитенантності забезпечує розподіл запитів до відповідних баз даних орендарів на основі ідентифікаторів орендарів або інших параметрів.

в) Tenant N Database Server Сервер бази даних для першого орендаря. Містить всі дані, пов'язані з Tenant 1.

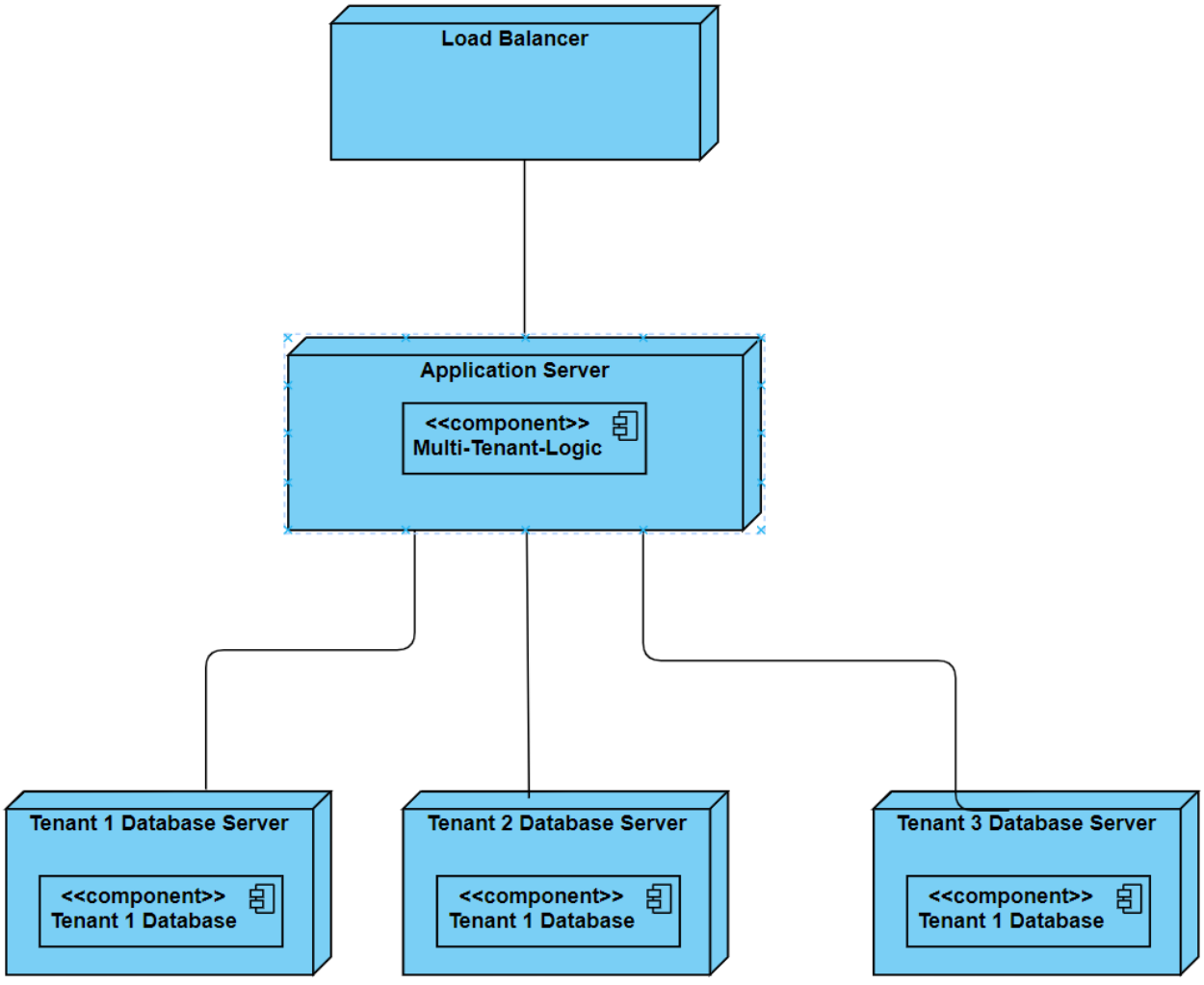


Рисунок 3.12 Діаграма розгортання мультитенантної архітектури для системи пошуку автомобільних попутників

3.6.5 Переваги такої архітектури:

– Окремі бази даних для кожного орендаря забезпечують кращу ізоляцію даних, що підвищує безпеку і дозволяє уникнути випадкового доступу до даних інших орендарів.

– Можливість додавання нових баз даних орендарів при збільшенні кількості клієнтів без суттєвого впливу на існуючих користувачів.

– Оскільки кожна база даних обслуговує одного орендаря, це дозволяє оптимізувати продуктивність запитів і зменшити навантаження на кожну окрему базу даних.

3.6.6 Недоліки такої архітектури

– Збільшується складність управління і моніторингу при зростанні кількості орендарів.

– Необхідність утримання окремих серверів баз даних для кожного орендаря може збільшити витрати на інфраструктуру.

– При великій кількості орендарів може виникнути складність у масштабуванні та балансуванні навантаження між серверами баз даних.

Ця архітектура підходить для сценаріїв з високими вимогами до безпеки та ізоляції даних, що робить її оптимальним вибором для системи пошуку автомобільних попутників, яка обслуговує декількох орендарів.

3.7 Інтеграція з зовнішніми системами

Сервіс пошуку автомобільних попутників пов’язан з декількома зовншніми системами, а саме:

– Взаємодія з картографічними сервісами;

– Інтеграція з платіжними системами;

– Інтеграція з системами електронної пошти(SMPT сервіси);

– Інтеграція з соціальними мережами.

Сервіс пошуку автомобільних попутників має інтеграцію з картографічним сервісом. Інтеграція з картографічними сервісом, такими як

Google Maps, дозволяє користувачам переглядати маршрути, розраховувати відстані та час подорожі, що полегшує їм вибір потенційних попутників. Та отримувати дані координат маршруту для пошуку поїздки.

Інтеграція з системами електронної пошти дозволяє надсилати користувачам сповіщення про нові подорожі або важливі оновлення через електронну пошту. Також цей сервіс використовується для реєстрації нових користувачів.

Інтеграція з платіжними системами, такими як PayPal або Stripe, дозволяє користувачам легко та безпечно оплачувати послуги, такі як замовлення поїздок або підписка на преміум-плани.

Також можлива інтеграція з соціальними мережами, такими як Facebook або Twitter, надає можливість знаходити попутників серед друзів та знайомих, а також ділитися своїми маршрутами та досвідом подорожей.

3.8 Обґрунтування вибраної архітектури

Для розробки сервісу пошуку автомобільних попутників було обрано варіант централізованої Saas системи. Нижче наведено обґрунтування вибору Software as a Service (SaaS) моделі для сервісу пошуку попутників, а також кілька ключових аспектів ,що обґрунтують обраний вибір:

Централізована SaaS архітектура дозволяє користувачам отримувати доступ до сервісу через інтернет з будь-якого пристрою та місця. Це робить використання системи максимально зручним для користувачів, оскільки вони можуть працювати з нею з будь-якої точки світу.

У централізованій архітектурі системи, усі технічні питання, такі як розгортання, оновлення, резервне копіювання та безпека, здійснюються централізовано зі сторони постачальника послуги. Це відбирає відповідальність за інфраструктуру та технічні аспекти управління від користувачів, звільняючи їх від додаткових витрат часу та ресурсів.

Централізована SaaS модель дозволяє швидко розгортати нові версії програмного забезпечення та оновлення централізовано для всіх користувачів. Це забезпечує усім користувачам найновіші функції та покращення без необхідності ручного оновлення або розгортання.

Централізована SaaS модель зменшує витрати на обладнання та обслуговування інфраструктури, оскільки ресурси спільно використовуються між багатьма користувачами. Це дозволяє знизити початкові витрати на впровадження та забезпечити більш привабливі умови користувачам.

Централізована архітектура дозволяє легко масштабувати ресурси та забезпечувати високу доступність сервісу. Це досягається шляхом використання розподілених систем та резервного копіювання даних, що забезпечує неперервну роботу системи при будь-яких обставинах.

Хоча централізована модель Software as a Service (SaaS) має численні переваги, вона також має свої недоліки:

– Залежність від постачальника послуг: Користувачі повністю залежать від постачальника SaaS, оскільки всі аспекти інфраструктури, включаючи безпеку, надійність та оновлення, контролюються ним. Це може створити проблеми, якщо постачальник виявиться ненадійним або вирішить припинити надання послуг.

– Безпека та конфіденційність даних: При передачі та зберіганні даних на стороні постачальника SaaS можуть виникнути проблеми з безпекою та конфіденційністю. Користувачам не завжди відомо, де саме зберігаються їхні дані та як вони захищені.

– Обмежена гнучкість та настроюваність: Оскільки інфраструктура управляється постачальником послуг, користувачі мають обмежені можливості щодо гнучкого налаштування системи під свої потреби. Це може стати перешкодою для організацій з унікальними вимогами або специфічними процесами.

– Відмова від доступу до даних: У випадку, якщо користувач вирішить припинити використання послуг SaaS, він може стикнутися з проблемою з доступом до своїх даних. Постачальники можуть мати обмеження щодо експорту або перенесення даних у інші системи.

– Загрози щодо неперервності роботи: Навіть якщо постачальник гарантує високу доступність, існує ризик виникнення проблем з неперервністю роботи через технічні або природні катастрофи, атаки зломщиків або інші фактори, що можуть вплинути на доступність послуг.

Отже, вибір централізованої SaaS архітектури пов'язаний з рядом ризиків та обмежень, які потрібно врахувати під час проектування та розвитку системи.

Вибір централізованої моделі Software as a Service (SaaS) для системи пошуку попутників має більше переваг, ніж недоліків, оскільки вона забезпечує зручний доступ, економічну ефективність та швидке впровадження інновацій. Цей підхід виявляється кращим за інші моделі, оскільки він сприяє зростанню бізнесу та задоволенню потреб користувачів, забезпечуючи високий рівень доступності, безпеки та гнучкості використання.

3.9 Архітектурна схема мультитенатного сервісу пошуку автомобільних попутників

Сервіс пошуку автомобільних попутників передбачає інтеграцію водіїв та пасажирів, що мають схожі маршрути, для спільних поїздок. Для забезпечення надійної, масштабованої та ефективної роботи цього сервісу обрана централізована SaaS система. Нижче детально описано серверну архітектуру цієї системи.

Архітектура сервера включає в себе такі компоненти Web Api. Authentication Service, User Profile Service , Ride Matching Service, Ride Management Service, Notification Service

Web Api виконує такі функції:

– Приймає всі запити від клієнтів веб-інтерфейсу та перенаправляє їх до відповідних сервісів;

– Виконує аутентифікацію та авторизацію користувачів;

– Відповідає за пошук та підбір оптимальних варіантів попутників;

– Використовує алгоритми машинного навчання для аналізу даних та визначення найкращих збігів;

– Забезпечує балансування навантаження та обмеження швидкості запитів.

Authentication Service відповідає за:

– Реєстрацію управління сесіями користувачів;

– Авторизацію користувачів;

– Використовує протоколи JWT для безпечної аутентифікації.

User Profile Service відповідає за такі функції:

– Зберігає та управляє профілями користувачів, включаючи особисту інформацію, фотографії та рейтинги;

– Підтримує функціонал редагування та оновлення профілів.

Також на сервері розташовано бази даних організацій, що створили тенант. Реляційна база даних зберігає структуровані дані, такі як профілі користувачів, маршрути поїздок та платіжні дані.

Використання такого підходу повторює хмарний підхід, але в якості ресурсів, що надають користувачеві використовуються сервера компаній, що реалізує цей сервіс автоматизованого пошуку автомобільних попутників.

Централізована SaaS система для сервісу пошуку автомобільних попутників забезпечує масштабованість, надійність та гнучкість. Основною перевагою такої архітектури є можливість легкого масштабування системи відповідно до зростання кількості користувачів та навантаження. Завдяки використанню хмарних сервісів, система може автоматично адаптуватися до змін у трафіку, гарантуючи безперебійність роботи та швидкий час відгуку.

На додаток, інтеграція різних мікросервісів дозволяє легко оновлювати та розширювати функціонал системи. Кожен мікросервіс може розроблятися, тестуватися та розгортатися незалежно від інших, що зменшує ризики виникнення помилок та дозволяє швидко впроваджувати нові функції. Це також сприяє покращенню безпеки системи, оскільки проблеми в одному мікросервісі не впливають на інші.

Гнучкість системи дозволяє швидко реагувати на змінні потреби ринку та користувачів. Нова функціональність може бути додана з мінімальними зусиллями, а існуючі компоненти можуть бути легко оновлені або замінені без переривання роботи сервісу. Це забезпечує високий рівень задоволення користувачів, оскільки сервіс залишається стабільним та інноваційним.

Надійність системи забезпечується використанням резервного копіювання даних, моніторингом продуктивності та автоматичним відновленням після збоїв. Це гарантує, що користувачі завжди можуть покластися на сервіс, навіть у разі непередбачуваних проблем.

Завдяки такій архітектурі, сервіс пошуку автомобільних попутників може ефективно забезпечувати високоякісне обслуговування користувачів, підтримуючи стабільність, продуктивність та безпеку. Це дозволяє компанії

фокусуватися на інноваціях та розвитку, знаючи, що їх технічна інфраструктура готова підтримати будь-які майбутні виклики.

3.10 Вплив на майбутнє розширення та підтримку системи

Вибір централізованного варінту архітектури має багато переваг, які дозволяють реалізувати можливість розширення та масштабування системи у майбутньому.

SaaS архітектура дозволяє легко масштабувати систему відповідно до зростання кількості користувачів та обсягу даних. Постачальники SaaS забезпечують еластичну інфраструктуру, яка автоматично адаптується до збільшення навантаження. Це означає, що система може обробляти різні рівні трафіку без значних змін у конфігурації або додаткових інвестицій в апаратне забезпечення. Для сервісу пошуку попутників, де очікується постійний приріст користувачів, така масштабованість є критично важливою, оскільки дозволяє підтримувати високу продуктивність і швидкість роботи навіть при значному збільшенні кількості одночасних запитів.

Централізована SaaS модель дозволяє швидко розгортати нові функції та оновлення для всіх користувачів одночасно. Це досягається через централізоване управління оновленнями, що мінімізує час простою та необхідність ручного втручання з боку користувачів. Нові функціональні можливості можуть бути додані без значних затримок, що дозволяє постійно покращувати сервіс та відповідати на зворотний зв'язок від користувачів. Наприклад, можна швидко інтегрувати нові алгоритми пошуку попутників або оптимізації маршрутів, забезпечуючи користувачів найсучаснішими інструментами для покращення їхнього досвіду.

У централізованій SaaS моделі вся відповідальність за технічне обслуговування, резервне копіювання, безпеку та виправлення помилок лежить на постачальнику послуг. Це знижує навантаження на внутрішні IT-ресурси організації та дозволяє зосередитися на основному бізнесі. Постачальники SaaS також забезпечують постійний моніторинг системи та управління продуктивністю, що допомагає швидко виявляти та вирішувати потенційні проблеми. Наприклад, якщо виникає технічний збій або вразливість, постачальник може оперативно втрутитися і мінімізувати вплив на користувачів.

Постачальники SaaS зазвичай пропонують високий рівень безпеки, який включає шифрування даних, багатофакторну аутентифікацію та регулярні аудити безпеки. Вони також забезпечують відповідність різним галузевим стандартам та регулятивним вимогам, що важливо для обробки особистої інформації користувачів. Наприклад, у випадку сервісу пошуку попутників, конфіденційність даних користувачів є пріоритетом, і постачальники SaaS можуть гарантувати захист цих даних відповідно до стандартів GDPR або інших регуляторних вимог.

SaaS рішення дозволяють використовувати сучасні технології та інструменти, які можуть бути швидко інтегровані в систему. Це означає, що нові сервіси та компоненти можуть бути легко додані, розширюючи функціональні можливості сервісу. Наприклад, інтеграція з іншими популярними сервісами, такими як платіжні системи, сервіси картографії або аналітичні інструменти, може бути здійснена без значних зусиль, що забезпечує додаткову гнучкість та функціональність.

Завдяки централізованому управлінню інфраструктурою, SaaS модель знижує витрати на апаратне забезпечення, ліцензії на програмне забезпечення, енергоспоживання та персонал для технічного обслуговування. Це дозволяє організаціям перенаправляти ресурси на розвиток нових функцій та поліпшення користувацького досвіду. Наприклад, кошти, які зазвичай витрачаються на підтримку серверів та мережевої інфраструктури, можуть бути використані для проведення маркетингових кампаній або розробки нових функціональних можливостей для покращення сервісу.

Вибір централізованої моделі SaaS для сервісу пошуку попутників забезпечує значні переваги для майбутнього розширення та підтримки системи. Ця архітектура надає можливість легко масштабувати ресурси, швидко впроваджувати нові функції, знижувати витрати на технічне обслуговування, забезпечувати високий рівень безпеки та відповідність стандартам, а також інтегрувати сучасні технології. Незважаючи на наявні ризики, такі як залежність від постачальника послуг та потенційні проблеми з безпекою даних, переваги централізованої SaaS моделі роблять її оптимальним вибором для розвитку та підтримки ефективного і зручного сервісу пошуку попутників.

3.11 Рекомендації для подальшого розвитку проекту

Для забезпечення подальшого успішного розвитку сервісу пошуку попутників, необхідно зосередити увагу на таких ключових напрямках:

Оптимізація алгоритмів пошуку: Постійне вдосконалення алгоритмів пошуку попутників є критично важливим для покращення точності та швидкості знаходження відповідних маршрутів. Використання методів машинного навчання та аналізу великих даних допоможе оптимізувати процеси зіставлення водіїв та пасажирів на основі їхніх маршрутів, уподобань та історичних даних. Це дозволить підвищити якість підбору попутників та задоволеність користувачів.

Покращення користувацького інтерфейсу: Регулярне оновлення дизайну та функціональності користувацького інтерфейсу має забезпечити інтуїтивно зрозумілий та привабливий вигляд сервісу. Врахування зворотного зв’язку від користувачів, проведення тестувань та аналіз поведінки користувачів допоможуть визначити області для покращення. Важливо забезпечити зручну навігацію, швидкий доступ до основних функцій та приємний користувацький досвід.

Інтеграція з додатковими сервісами: Розширення функціональності платформи шляхом інтеграції з іншими популярними сервісами, такими як платіжні системи, сервіси картографії, соціальні мережі та служби безпеки, створить додаткові зручності для користувачів. Наприклад, інтеграція з платіжними системами дозволить здійснювати безпечні та зручні транзакції, а використання сервісів картографії забезпечить точне визначення маршрутів та місць зустрічі.

Забезпечення високої безпеки даних: Безпека та конфіденційність даних користувачів є пріоритетними для сервісу пошуку попутників. Впровадження сучасних технологій шифрування, багатофакторної аутентифікації та регулярні аудити безпеки допоможуть захистити особисту інформацію користувачів від несанкціонованого доступу та витоку. Постійне оновлення політик безпеки та навчання персоналу дозволить підтримувати високий рівень захисту даних.

Масштабування інфраструктури: Планування та реалізація заходів щодо масштабування інфраструктури дозволять забезпечити стабільну роботу сервісу при збільшенні кількості користувачів. Використання хмарних технологій

забезпечить еластичність та адаптивність системи до змін у навантаженні, знижуючи ризик простоїв та втрату продуктивності. Регулярний моніторинг і оптимізація ресурсів допоможуть підтримувати високу якість обслуговування.

Розширення географії покриття: Поступове розширення географії покриття сервісу є важливим етапом розвитку. Проведення маркетингових досліджень допоможе визначити найбільш перспективні ринки для виходу. Важливо враховувати локальні особливості, потреби користувачів та конкурентне середовище для ефективного планування експансії. Розробка стратегії виходу на нові ринки дозволить забезпечити успішне впровадження сервісу в нових регіонах.

Аналіз та використання зворотного зв'язку: Постійний аналіз зворотного зв'язку від користувачів є ключовим для ідентифікації слабких місць та можливостей для покращення. Регулярні опитування, аналіз відгуків та поведінки користувачів допоможуть виявити проблемні зони та розробити відповідні заходи для їх усунення. Впровадження змін на основі зворотного зв'язку сприятиме підвищенню задоволеності користувачів та покращенню загальної якості сервісу.

Запровадження нових функцій та сервісів: Розширення функціональності платформи за рахунок введення нових корисних функцій допоможе залучити нових користувачів та утримати існуючих. Наприклад, впровадження рейтингової системи для водіїв та пасажирів, системи бонусів та винагород, інтеграція з програмами лояльності та іншими додатковими сервісами створять додаткову цінність для користувачів. Це сприятиме покращенню їхнього досвіду та підвищенню залученості.

Підвищення якості обслуговування клієнтів: Забезпечення високої якості обслуговування клієнтів є важливим елементом успіху сервісу. Впровадження сучасних технологій підтримки користувачів, таких як чат-боти, інтеграція з популярними месенджерами та цілодобова підтримка, дозволять швидко та ефективно вирішувати проблеми користувачів. Це підвищить рівень задоволеності клієнтів та сприятиме формуванню позитивного іміджу сервісу.

Пошук нових партнерств та співпраця: Встановлення партнерських відносин з іншими компаніями та сервісами може забезпечити нові можливості для користувачів та розширити функціональність платформи. Співпраця з локальними транспортними компаніями, туристичними агентствами, сервісами доставки та іншими релевантними партнерами може значно підвищити цінність

сервісу та залучити нових користувачів. Партнерства можуть також сприяти розвитку нових бізнес-моделей та додаткових джерел доходу.

Дотримання цих рекомендацій сприятиме сталому розвитку сервісу пошуку попутників, підвищенню його конкурентоспроможності та забезпеченню високого рівня задоволеності користувачів.

4 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

4.1 Тестування сервісу пошуку автомобільних попутників

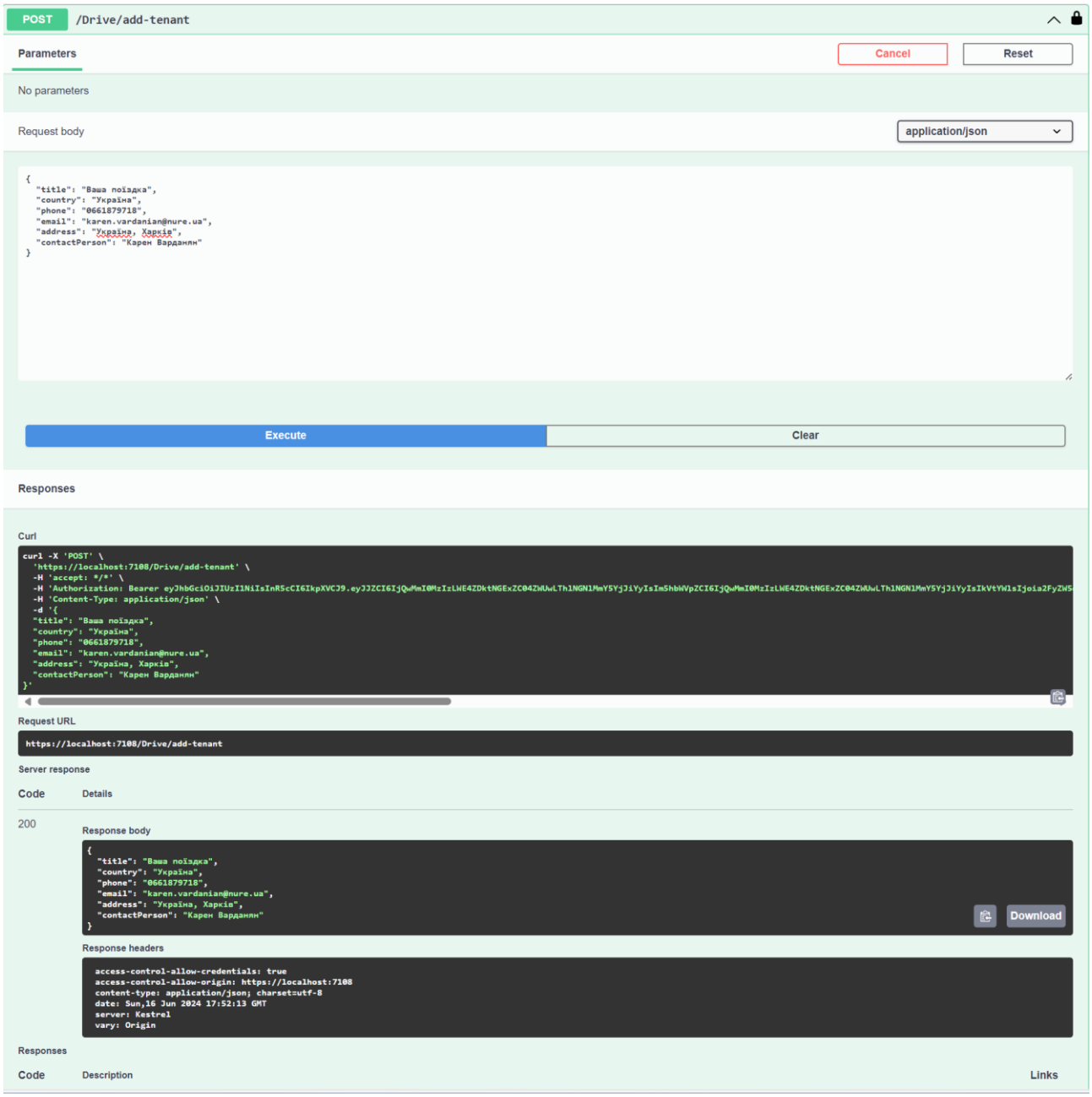


Рисунок 4.1 – Приклад створення організація у сервісі пошуку автомобільних попутників

На рисунку 4.1 наведено приклад створення тенанту у сервісі пошуку автомобільних попутників. Під час створення користувачеві необхідно передати такі дані:

– title: Назва організації;

– country: Країна;

– phone: Номер телефону;

– email: Електронна пошта;

– address: Адреса ;

– contactPerson: Контактна особа.

У тілі запиту передана вся необхідна інформація про нову організацію, включаючи назву, країну, контактні дані та інформацію про контактну особу.

Коли запит на створення нового тенанта успішно виконаний, а сервер підтвердив успішне додавання нового тенанта, повернувши ті ж самі дані у відповіді.

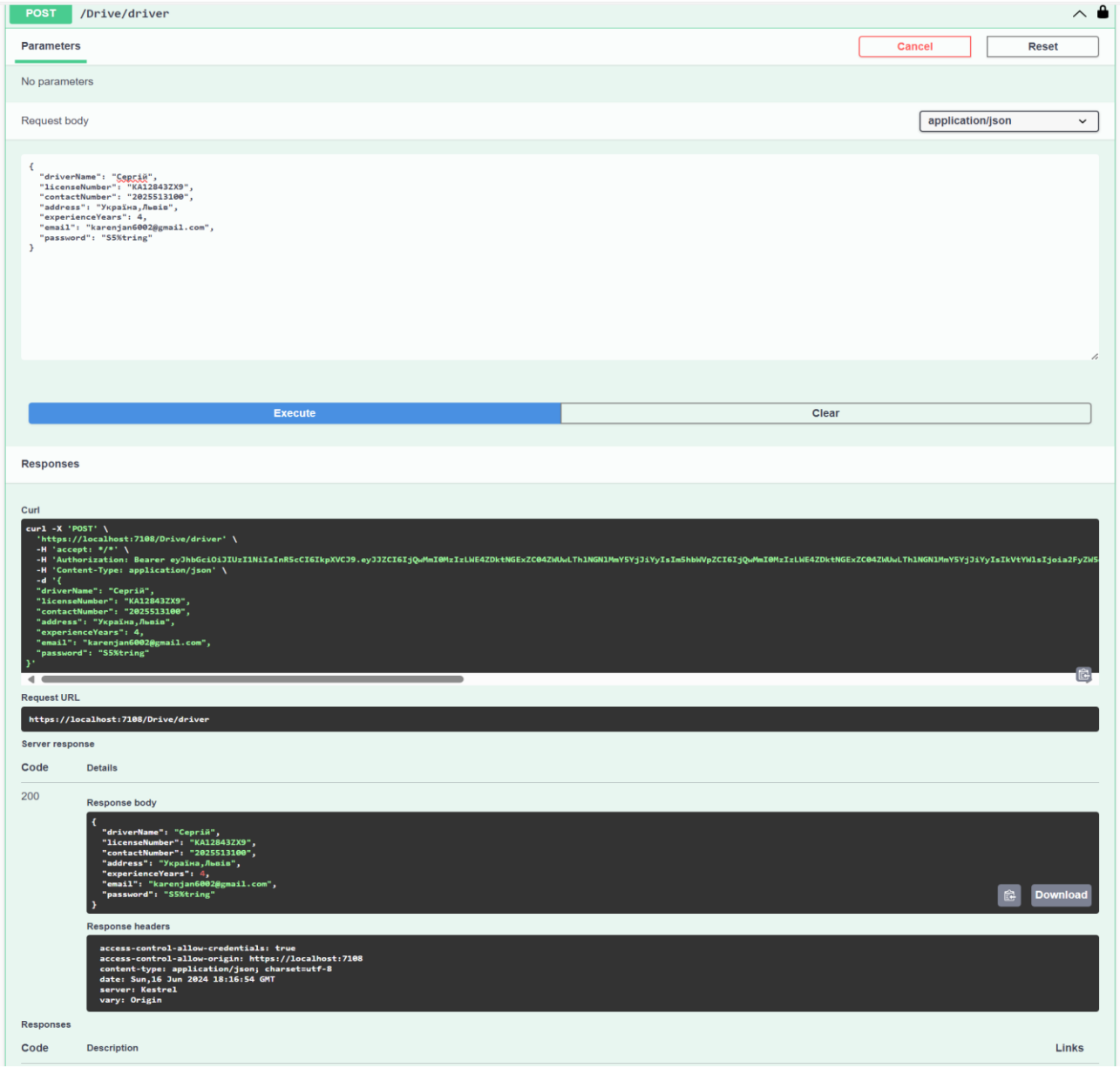


Рисунок 4.2 – Приклад додавання водія у систему пошуку автомобільних попутників

На рисунку 4.2 наведено приклад додавання водія у сервісі пошуку автомобільних попутників. Під час створення користувачеві необхідно передати такі дані:

- driverName: ім'я водія;
- licenseNumber: номер ліцензії водія;
- contactNumber: контактний номер;
- address: адреса;
- experienceYears: кількість років досвіду водія;
- email: електронна пошта;
- password: пароль.

У тілі запиту передана вся необхідна інформація про водія, включаючи контактні дані та інформацію про контактну особу. Коли запит на додавання нового водія успішно виконаний, а сервер підтвердив успішне додавання нового водія, повернувши ті ж самі дані у відповіді.

На рисунку 4.3 демонструє процес додавання даних про автівку у сервіс з пошуку автомобільних попутників. Для успішного додавання даних автівки у ситему необхідно надати таку інформацію:

- carModel: модель автомобіля;
- carName: назва автомобіля;
- carType: тип автомобіля;
- carNumber: номер автомобіля;
- manufacturer: виробник автомобіля;
- year: рік випуску автомобіля;
- color: колір автомобіля;
- engineType: тип двигуна;
- mileage: пробіг автомобіля;
- price: ціна автомобіля;
- registrationDate: дата реєстрації автомобіля;
- VIN: ідентифікаційний номер транспортного засобу;
- transmissionType: тип трансмісії;
- fuelType: тип палива;
- seatingCapacity: кількість сидячих місць;
- horsePower: потужність двигуна.

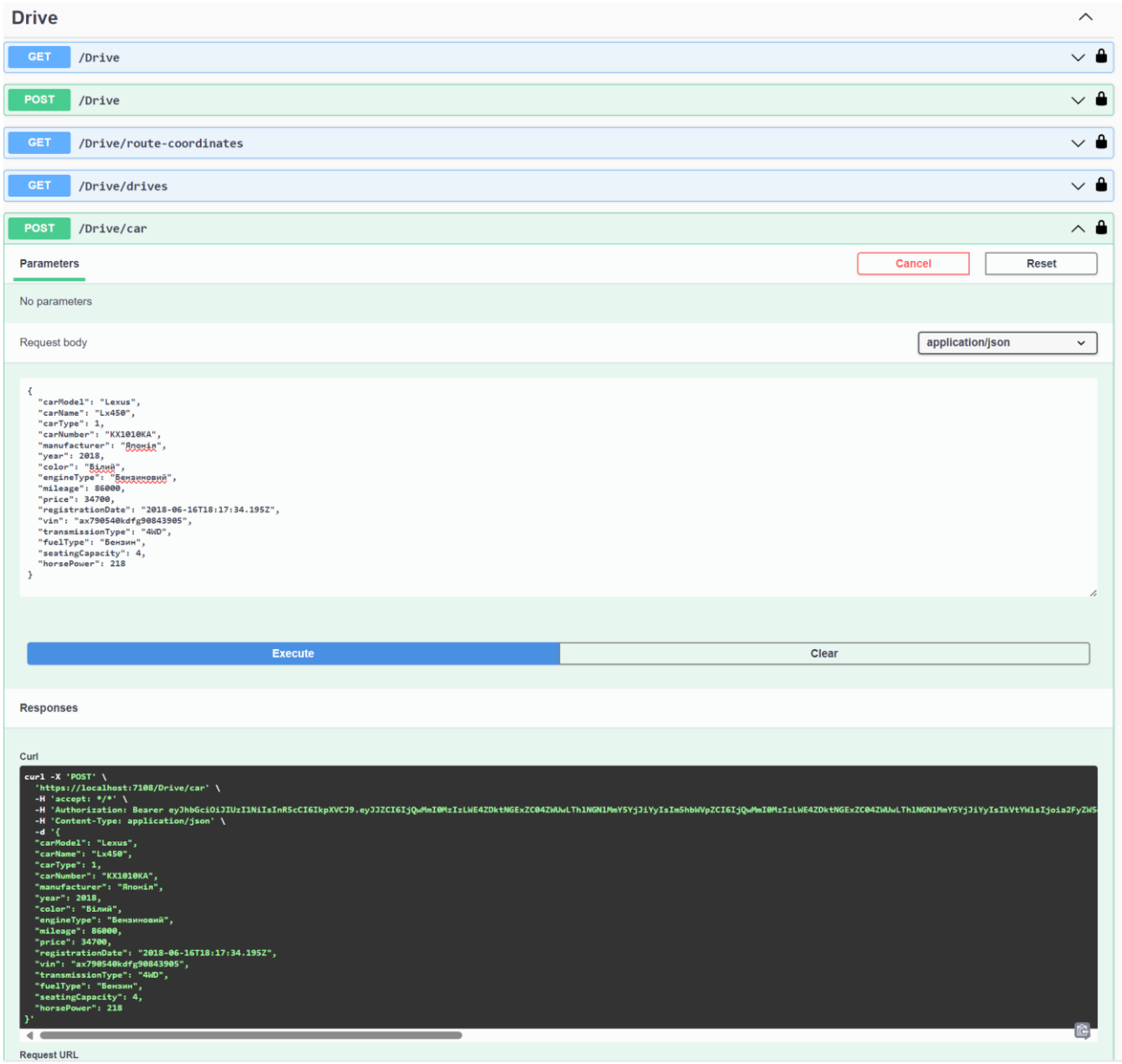


Рисунок 4.3 – Приклад додавання автівки у систему пошуку автомобільних попутників

Коли запит на додавання нової автівки успішно виконаний. У тілі запиту передана вся необхідна інформація про автівки, включаючи її характеристику. Сервер підтвердив успішне додавання нового водія, повернувши ті ж самі дані у відповіді.

На рисунку 4.4 демонструється процес створення нової поїздки в сервісі пошуку автомобільних. Детальний опис полів необхідних для створення поїздки наведено нижче.

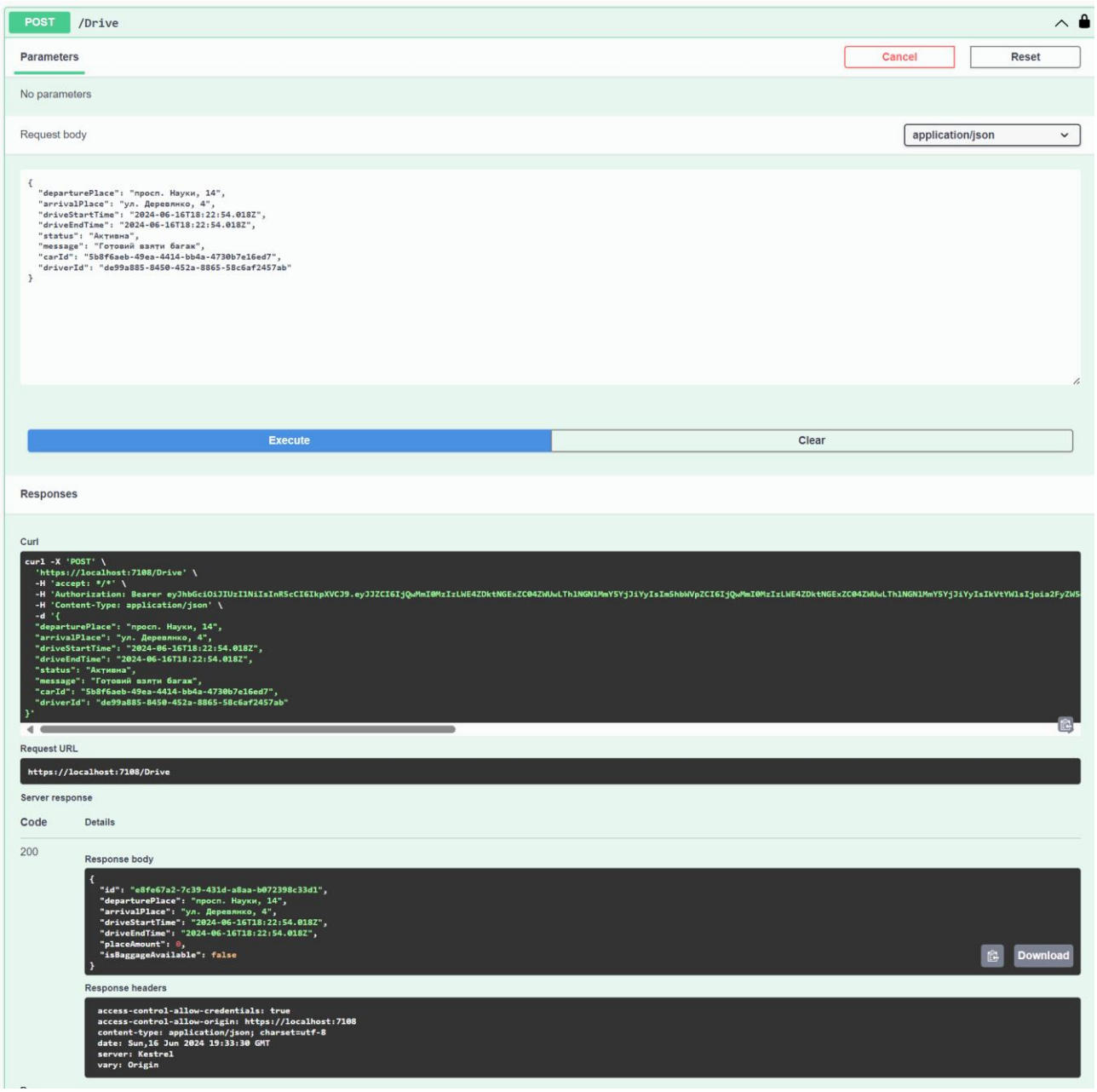


Рисунок 4.4 – Приклад створення поїздки у системі пошуку автомобільних попутників

Перелік полів, що необхідно заповнити під час створення поїздки:

– departurePlace: місце відправлення;
– arrivalPlace: місце прибуття;
– driveStartTime: час початку поїздки;
– driveEndTime: час закінчення поїздки;
– status: статус поїздки;
– message: повідомлення;
– carId: ідентифікатор автомобіля;
– organizationId: ідентифікатор організації.

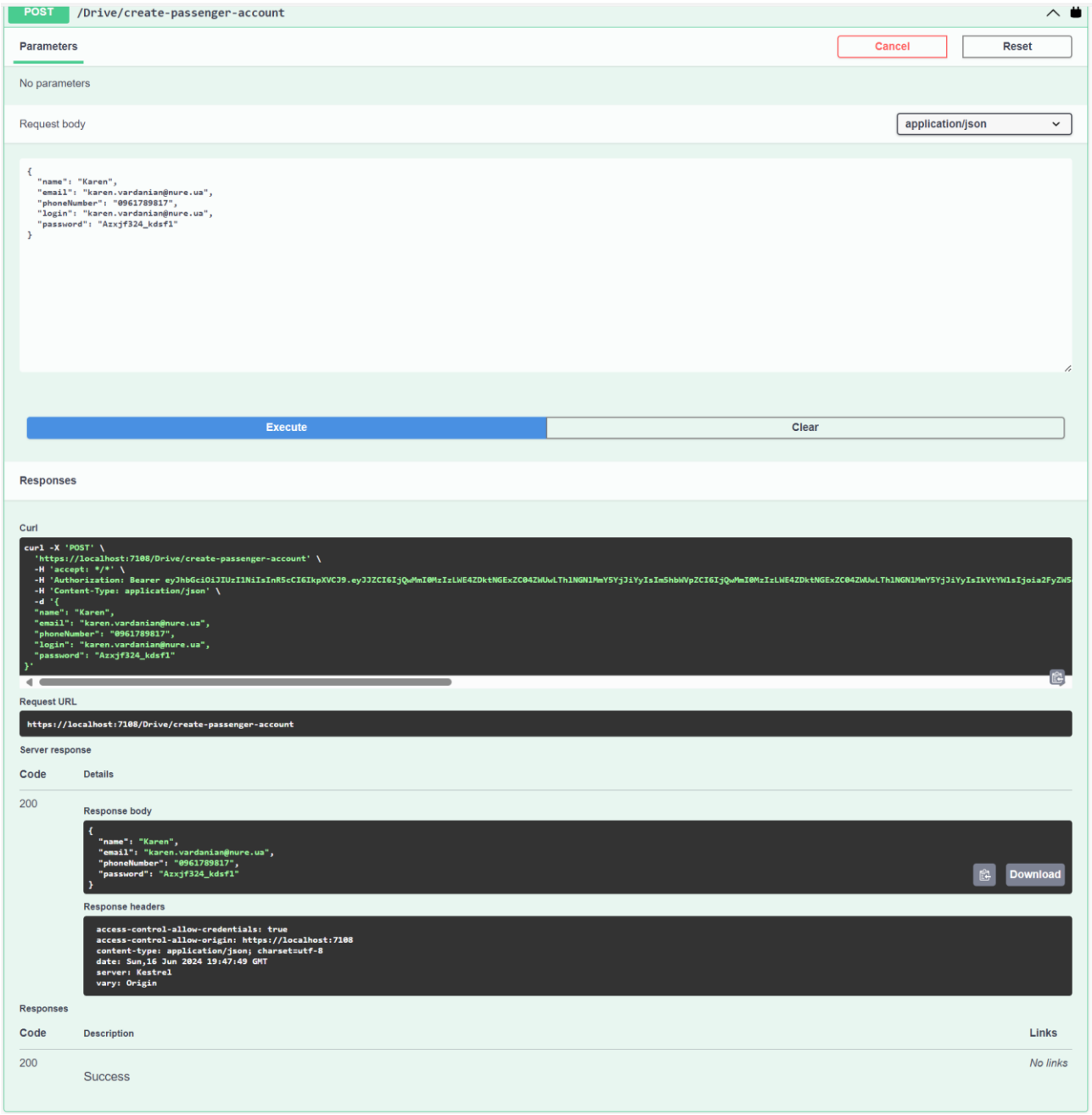
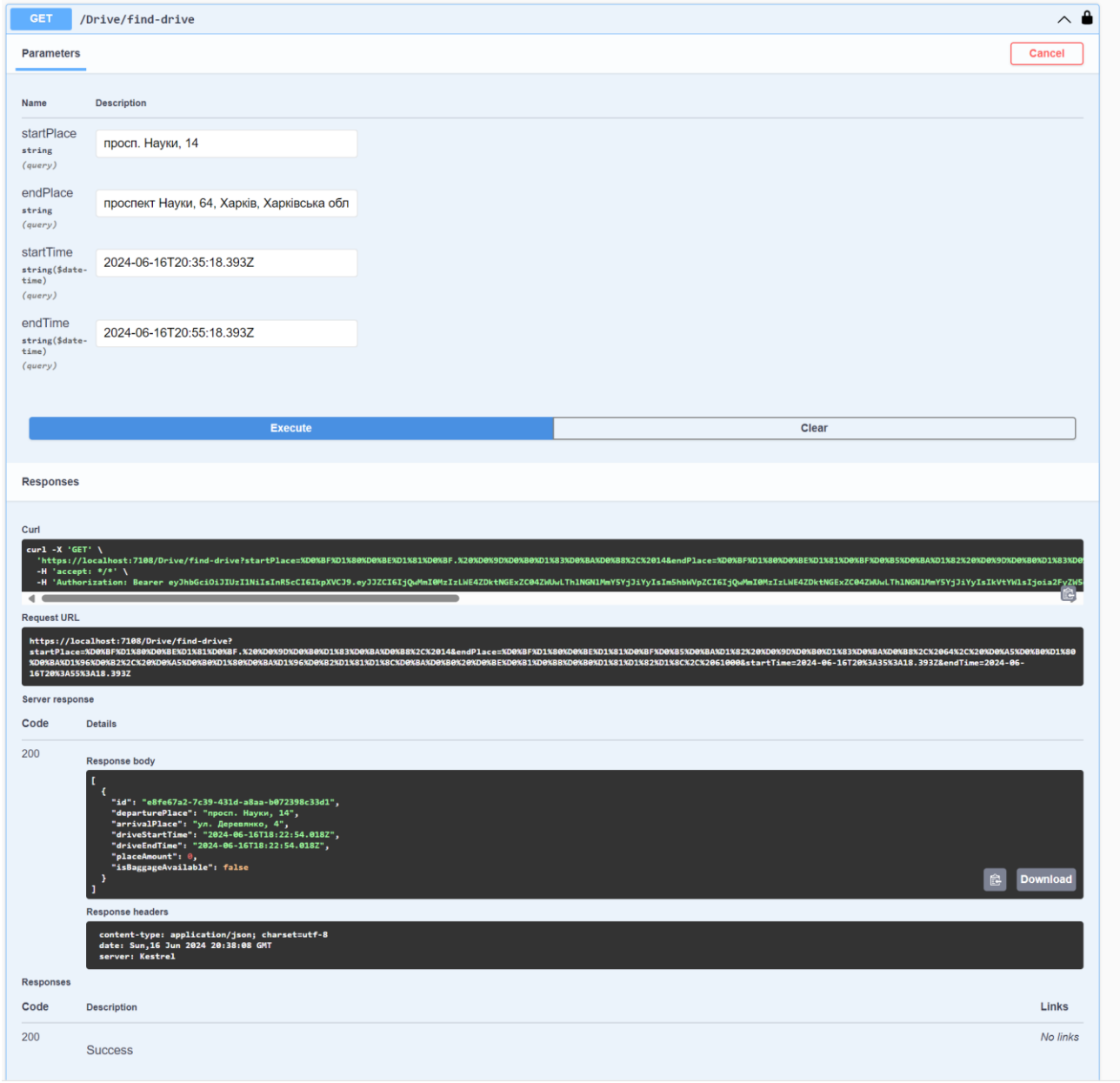


Рисунок 4.5 – Приклад створення профілю пасажира у сервісі автомобільних попутників

На рисунку 4.5 показується процес створення профілю пасажира в сервісі пошуку автомобільних. Детальний опис полів необхідних для створеня профілю наведено нижче:

– name: ім’я пасажира;
– phone: телефон пасажира;
– email: електронна пошта;
– password: пароль.

Коли запит на додавання нового профілю у системі успішно виконаний. У тілі запиту передана вся необхідна інформація про людину. Сервер підтвердив успішне додавання нового пасажира, повернувши ті ж самі дані у відповіді.



На рисунку 4.6 – Прилад пошуку автомобільного попутника

На цьому рисунку є кілька полів для введення даних, які використовуються для пошуку поїздки в сервісі автомобільних попутників. Нижче наведено пояснення значення кожного з полів:

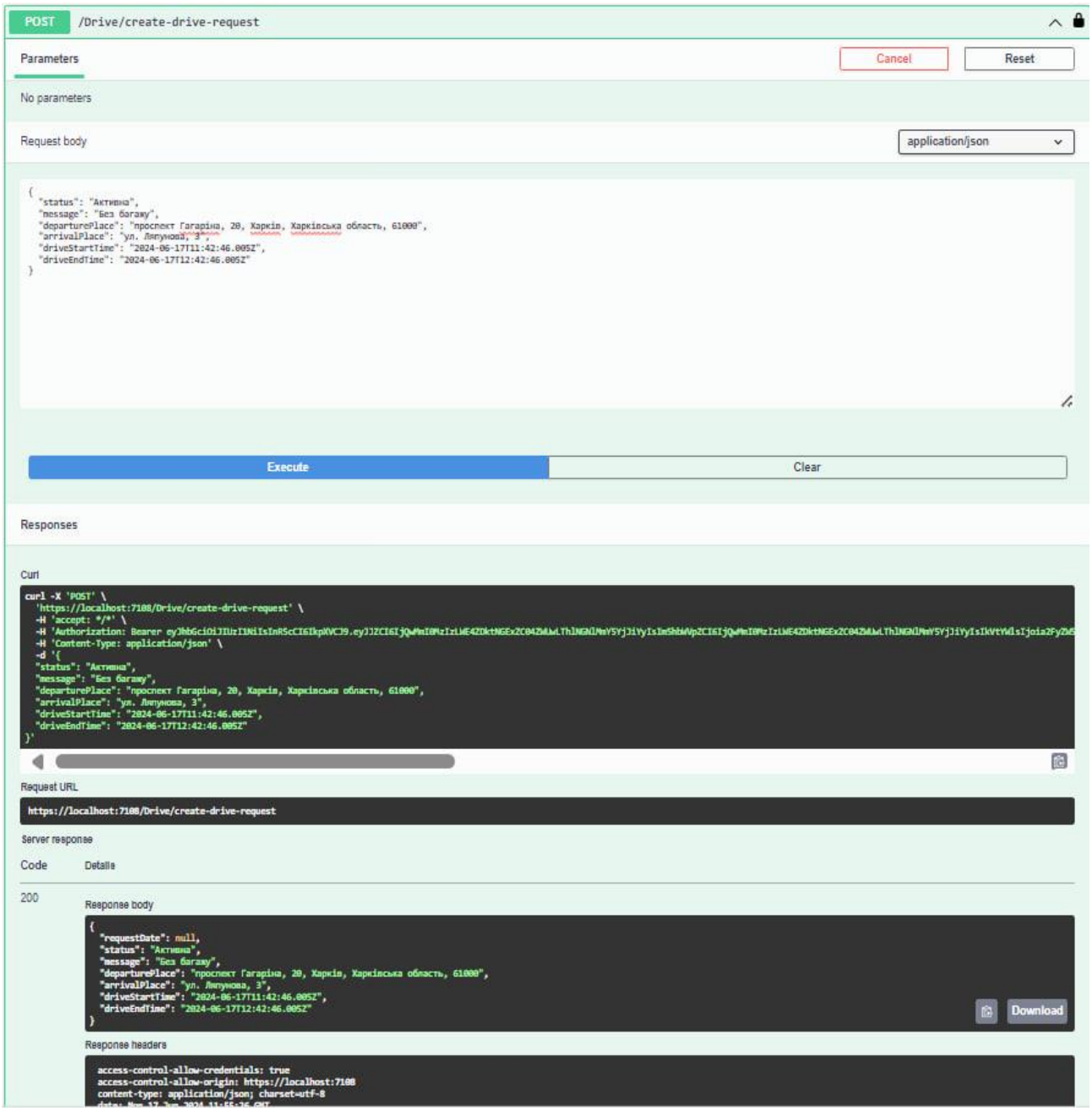
– startPlace (початкове місце) - це поле для введення адреси або місця, звідки пасажир планує розпочати свою поїздку;

– endPlace (кінцеве місце) - це поле для введення адреси або місця, куди пасажир планує прибути;

– startTime (час початку) - це поле для введення дати та часу, коли пасажир хоче розпочати поїздку;

– endTime (час закінчення) - це поле для введення дати та часу, до якого пасажир хоче завершити поїздку.

Ці поля використовуються для визначення параметрів запиту до API, що дозволяє знайти поїздки, які відповідають введеним критеріям. Після того коли запит на пошук маршруту у системі успішно виконаний, то далі показується відповідь отримана з серверу. У тілі відповіді отриманій з серверу показується уся детальна інформація про поїздку.



На рисунку 4.7 – Прилад пошуку автомобільного попутника

На рисунку 4.7 наведено приклад створення заявки на поїздку зі сторони користувача з роллю пасажир у сервісі пошуку автомобільних попутників. Під час створення такої заявки користувачеві необхідно передати такі дані:

– departurePlace: місце відправлення;
– arrivalPlace: місце прибуття;
– driveStartTime: час початку поїздки;
– driveEndTime: час закінчення поїздки;
– status: статус поїздки;
– message: додаткове повідомлення;
– message: повідомлення.

У тілі запиту передана вся необхідна інформація про нову організацію, включаючи назву, країну, контактні дані та інформацію про контактну особу. Коли запит на створення нової заявки на поїздку успішно виконаний, а сервер підтвердив успішне додавання нової заявки на поїздку, то він повертає ті ж самі дані у відповіді.

На зображенні 4.8 показується результати виконання HTTP GET запиту до сервісу пошуку автомобільних попутників. Відповідь сервера містить наступну інформацію у форматі JSON:

– responseCode: 200, що свідчить про успішне виконання запиту.
– message: ‘success’, підтвердження того, що операція пройшла без помилок.
– Дані про доступні пропозиції поїздок включають атрибути, такі як id, fromLocation, toLocation, departureTime, які описують індивідуальні пропозиції поїздок.

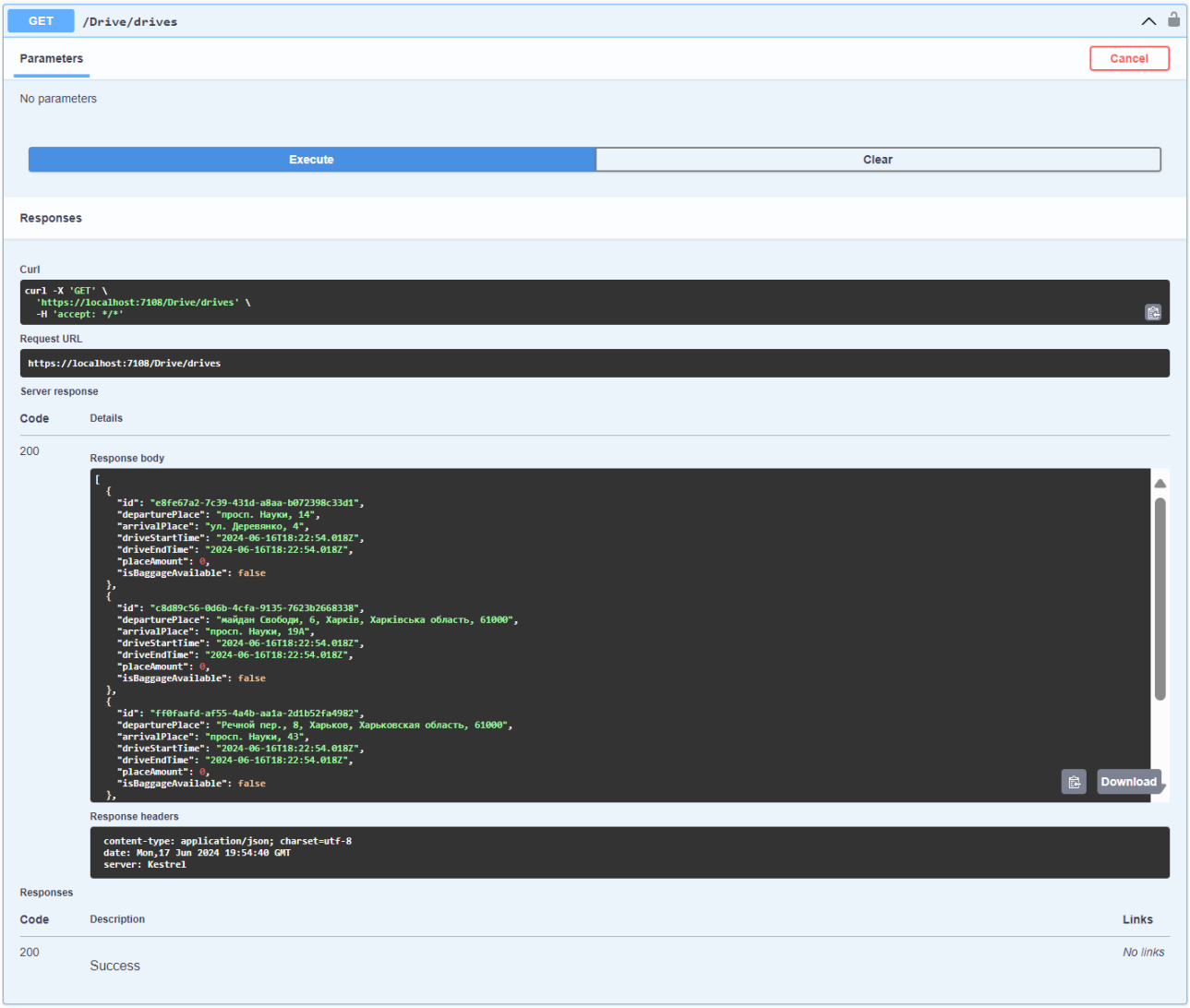
Ця інформація може бути корисною для розробників, які тестують або взаємодіють з такими сервісами програмно.

– id: Унікальний ідентифікатор пропозиції поїздки;
– departurePlace: Місце відправлення поїздки;
– arrivalPlac: Місце прибуття поїздки;
– departureTime: Час відправлення поїздки;
– deprtureEndTime: Час завершення поїздки;

– placeAmount: Кількість доступних місць у автівці;

isBagageAvailiable: булева змінна, що відповідає за можливість взяти багаж.

Ці поля дозволяють користувачам сервісу отримувати детальну інформацію про доступні пропозиції поїздок і вибирати найбільш підходящий варіант для спільних поїздок.



На рисунку 4.8 – Прилад формату даних про поїздку, що показується користувачам

ВИСНОВКИ

У даній кваліфікаційній роботі була всебічно досліджена проблема організації пошуку автомобільних попутників з метою створення ефективної, безпечної та зручної системи для координації спільних поїздок. Це дослідження є надзвичайно актуальним в умовах сучасного світу, де транспортні витрати та завантаженість доріг зростають, а необхідність зменшення викидів вуглекислого газу стає все більш нагальною. Крім того, соціальна взаємодія та комунікація між людьми набувають нового значення у світлі розвитку технологій спільного використання ресурсів.

Основною метою роботи було розробити та впровадити систему для пошуку автомобільних попутників, яка забезпечувала б швидкий та зручний пошук попутників для спільних поїздок на автомобілі. Задля досягнення цієї мети були поставлені наступні завдання: проведення аналізу існуючих рішень та визначення основних вимог до системи, проектування архітектури системи, розробка бази даних, серверної частини та клієнтських додатків, проведення комплексного тестування системи, впровадження системи та її оцінка за визначеними критеріями.

На початковому етапі дослідження був проведений аналіз існуючих рішень для пошуку автомобільних попутників. Було виявлено, що більшість сучасних систем мають такі проблеми, як низька продуктивність під високим навантаженням, недостатня безпека даних та складність використання. На основі цього аналізу були сформовані вимоги до нової системи, яка мала б уникнути цих недоліків.

Було спроектовано архітектуру системи, яка включає три основні компоненти: база даних, серверну частину та клієнтські додатки. Розробка бази даних забезпечила зберігання інформації про користувачів, поїздки, автомобілі та організації. Серверна частина була реалізована з використанням мікросервісної архітектури, що дозволяє легко масштабувати систему. Клієнтські додатки були створені з акцентом на зручність та простоту використання.

Після завершення розробки було проведено комплексне тестування системи. Включалися модульні, інтеграційні та функціональні тести, які показали, що система відповідає всім визначеним вимогам і працює стабільно

навіть під високим навантаженням. Система була успішно впроваджена та оцінена за кількома критеріями, включаючи продуктивність, безпеку даних та зручність використання. Всі ці критерії були задоволені, що свідчить про високу якість розробленого продукту.

Розроблена система для пошуку автомобільних попутників дозволяє водіям створювати поїздки та вказувати деталі маршруту, пасажирам – шукати поїздки за різними критеріями та подавати заявки на участь у поїздках, а організаціям – керувати своїми водіями та автомобілями. Система забезпечує безпечне зберігання даних та захист від несанкціонованого доступу. Ця система є ефективним інструментом для координації спільних поїздок, допомагаючи зменшити транспортні витрати, знизити навантаження на дороги та сприяти екологічній стійкості. Крім того, вона сприяє підвищенню соціальної взаємодії між людьми, які використовують спільні поїздки.

У майбутньому можливе розширення функціоналу системи, додавання підтримки різних мов, інтеграція з платіжними системами для автоматизації оплати поїздок, а також використання машинного навчання для покращення рекомендацій та пошуку поїздок. Це дозволить зробити систему ще більш зручною та корисною для користувачів. Таким чином, розроблена система є важливим кроком у напрямку створення зручного та екологічного транспорту майбутнього, який сприятиме підвищенню якості життя людей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ebbinghaus H. Memory: A contribution to experimental psychology. New York : Teachers College, Columbia University, 1913. 123 p.
2. Leitner S. So lernt man lernen. Der Weg zum Erfolg. Herder, Freiburg, 2003. 317 S.
3. Account of research leading to the SuperMemo method [Електронний ресурс] – Режим доступу до ресурсу: https://www.supermemo.com/en/archives1990-2015/english/ol/beginning.
4. Application of a computer to improve the results obtained in working with the SuperMemo method [Електронний ресурс] – Режим доступу до ресурсу: https://www.supermemo.com/en/archives1990-2015/english/ol/sm2.
5. Studying – Anki Manual [Електронний ресурс] – Режим доступу до ресурсу: https://docs.ankiweb.net/studying.html#learningrelearning-cards.
6. Jain, S. and Karbari, S.R. Single Page Reactive Application using Angular Spring MVC and Rest API. *International Journal of Engineering and Advanced Technology*. 2020. Vol. 9, no. 5. P. 478–480. DOI: https://doi.org/10.35940/ijeat.e9494.069520.
7. Choose Between Traditional Web Apps and Single Page Apps (SPAs) [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps.
8. SOFTECH INC WALTHAM MA. Integrated Computeraided Manufacturing (ICAM) Architecture Part II. Volume IV. Function Modeling Manual (IDEF0). Fort Belvoir, VA : Defense Technical Information Center, 1981.
9. Object Management Group. Unified Modeling Language 2.5.1 [Електронний ресурс] – Режим доступу до ресурсу: https://www.omg.org/spec/UML/2.5.1/PDF.

10. Hafeez, A. Importance and Impact of Class Diagram in Software Development. *Indian Journal of Science and Technology*. 2019. Vol. 12, no. 25. P. 1–4. DOI: https://doi.org/10.17485/ijst/2019/v12i25/145739.

11. Booch G. Object-Oriented analysis and design with applications. 3rd ed. San Jose : Addison-Wesley, 2007. 720 p.

12. Deitel H., Deitel P. Visual C# 2010 how to program. Pearson Education, Limited, 2011. 992 p.

13. Рейтинг мов програмування 2022. [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/lenta/articles/language-rating-2022/.

14. Welcome to IdentityServer4 [Електронний ресурс] – Режим доступу до ресурсу: https://identityserver4.readthedocs.io/en/latest/.

15. ASP.NET Core Blazor [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-6.0.

16. Entity Framework Core [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/ef/core/.

17. Behjat, N. A Comparison between Relational Databases and NoSQL Databases. *International Journal of Trend in Scientific Research and Development*. 2018. Vol. 2, no. 3. P. 845–848. DOI: https://doi.org/ 10.31142/ijtsrd11214.

18. Ben-Gan I. Microsoft SQL server 2008: T-SQL fundamentals. Redmond, Wash : Microsoft Press, 2009. 394 p.

19. Chen P. P. S. The entity-relationship model: Toward a unified view of data. Cambridge, Mass : M.I.T. Center for Information Systems Research, 1977. 36 p.

20. IIBA. A guide to the business analysis body of knowledge. International Institute of Business Analysis, 2015. 512 p.

21. Chen P. P. S. The entity-relationship model: toward a unified view of data. Cambridge, Mass : M.I.T. Center for Information Systems Research, 1977. 36 p.

22. Hoffer J. A. Modern database management. 9th ed. Upper Saddle River, NJ : Pearson/Prentice Hall, 2009. 690 p.

23. What spaced repetition algorithm does Anki use? [Електронний ресурс] – Режим доступу до ресурсу: https://faqs.ankiweb.net/what-spaced-repetition-algorithm.html.

24. FuzzySharp [Електронний ресурс] – Режим доступу до ресурсу: https://nanonets.com/blog/fuzzy-matching-fuzzy-logic/.

25. A Comprehensive Guide to Fuzzy Matching/Fuzzy Logic [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/JakeBayer/FuzzySharp.

26. Варданян К.А. DEVELOPMENT OF COMPONENTS OF THE INFORMATION SYSTEM FOR FINDING A PASSENGER. International conference «GLOBAL SCIENCE: PROSPECTS AND INNOVATIONS», V International scientific and practical conferece (Liverpool,28030 December 2023, UK).194-197 pages.

27. Варданян К.А. Дослідження підходів до проєктування систем організації перевезень попутним транспортом. 28-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті»: зб. матеріалів форуму. Т. 6. Конференція «Інформаційні інтелектуальні системи» (м. Харків, 16-18 квітня 2024р.). Харків, 2024. 238-242с.