Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ПРОГРАМНИЙ ЗАСІБ ДЛЯ ПРОГНОЗУВАННЯ ЗАМОВЛЕНЬ ТРАНСПОРТНОГО ЗАСОБУ НА ОСНОВІ АНАЛІЗУ СТАТИСТИКИ ПОПЕРЕДНІХ ЗАПИТІВ

Виконав: студент 2 курсу, групи 2КІ-22м
напряму підготовки (спеціальності)
123 — «Комп'ютерна інженерія»

_____ Черняховський І. Ю.

Керівник роботи: к.т.н., доц. каф. ОТ
_____ Городецька О.С.
«14» 12 2023 р.

Опонент: к.т.н., доц. каф. ПЗ
_____ Ракитянська Г.Б.
«15» 12 2023 р.

Допущено до захисту

Завідувач кафедри ОТ

_____ д.т.н., проф. Азаров О. Д.

«18» 12 2023 р.

ВНТУ – 2023

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 — Інформаційні технології
Спеціальність 123 — «Комп'ютерна інженерія»
Освітня програма — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
проф., д.т.н. О.Д. Азаров
«26» вересня 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

студенту Черняховському Ігорю Юрійовичу

1 Тема роботи «Програмний засіб для прогнозування замовлень транспортного засобу на основі статистики попередніх запитів», керівник роботи **Городецька Оксана Степанівна**, к.т.н., доцент, затверджені наказом вищого навчального закладу від 18 вересня 2023р. № 247

2 Строк подання студентом роботи 8 грудня 2023 року

3 Вихідні дані до роботи: виконати розробку програмного модуля генерації прогнозу попиту та тарифів на перевезення з використанням методів машинного навчання та штучних нейронних мереж.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз сучасних методів розробки, варіативний вибір засобів для розробки програмного забезпечення, вдосконалення методу побудови архітектури підсистеми прогнозування попиту, розробка програмного модуля генерації прогнозів попиту та тарифів, програмна реалізація та тестування.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема роботи алгоритму моделі XGBRegressor, діаграма розгортання системи, діаграма діяльності створення прогнозу, діаграма класів програмного модуля

6 Консультанти розділів роботи представлені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Городецька О.С., к.т.н., доц. каф. ОТ		
5	Небава М.І., к.е.н., професор кафедри ЕПВМ		

7 Дата видачі завдання 19.09.2023 р.

8 Календарний план наведено в таблиці 2.

Таблиця 2—Календарний план

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Підпис
1	Аналіз завдання. Вступ	04.10.2023	
2	Аналіз літературних джерел	18.10.2023	
3	Розробка технічного завдання	01.11.2023	
4	Розробка структури програмного модуля генерації прогнозів попиту та тарифів	15.11.2023	
5	Розробка програмного модуля генерації прогнозів попиту та тарифів	25.11.2023	
6	Попередній захист	16.10.2023	
7	Оформлення пояснювальної записки і презентації	02.12.2023	
8	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	07.12.2023	

Студент _____ Черняховський І. Ю.

Керівник роботи _____ Городецька О. С.

АНОТАЦІЯ

УДК 004.42

Черняховський І. Ю. Програмний засіб прогнозування замовлень транспортного засобу на основі аналізу статистики попередніх запитів. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна інженерія, Вінниця: ВНТУ, 2023. — 113 с. Укр. мова. Бібліогр.: 21 назв; рис.: 40 ; табл. 8.

У магістерській роботі представлено розробку програмного засобу для прогнозування замовлень транспорту, який використовує Python і Java для створення REST API та клієнтського додатку відповідно. Засіб базується на машинному навчанні та аналітиці, спрямованих на надання конкурентних переваг клієнтського додатку через впровадження додаткових затребуваних користувачами функцій. Вдосконалено метод побудови архітектури підсистеми прогнозування попиту, який враховує потоковий збір даних, кластер зберігання великих даних, контейнеризацію для зручного розгортання та безперервної інтеграції.

Розробку та підготовку до комерційного застосування було проведено з врахуванням сучасних трендів розробки програмного забезпечення, а також з увагою до надійності та безпеки програмного засобу. Тестування програмного продукту продемонструвало його ефективність, забезпечуючи достатню точність прогнозування. Економічний аналіз підтвердив конкурентоспроможність та економічну доцільність розробки та вдосконалення продукту.

Ключові слова: таксі, замовлення послуг перевезення, машинне навчання, REST API, Python, Java, Docker.

ABSTRACT

Chernyakhovskyi I. Software tool for forecasting orders of a transport vehicle based on the analysis of statistics of previous requests. Master's qualification work in specialty 123 — computer engineering, Vinnytsia: VNTU, 2023. — 113 p. Ukrainian. Bibliography: 21 titles; figures: 40; tables 8.

The thesis presents the development of software for forecasting transportation orders, which uses Python and Java to create a REST API and a client application, respectively. The project is based on machine learning and analytics aimed at providing competitive advantages to the client application through the introduction of additional features demanded by users. The method of building the architecture of the demand forecasting subsystem has been improved, which takes into account streaming data collection, big data storage cluster, containerization for convenient deployment and continuous integration.

The development and preparation for commercial use was carried out taking into account modern software development trends, as well as paying attention to the reliability and security of the software tool. Testing of the software product demonstrated its effectiveness, providing sufficient forecasting accuracy. The economic analysis confirmed the competitiveness and economic feasibility of the product development and improvement.

Keywords: taxi, ordering transportation services, machine learning, REST API, Python, Java, Doker.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

APK — Android Package (Формат файлу пакету для додатків Android)

ARIMA — Autoregressive Integrated Moving Average (Авторегресивна інтегрована ковзна середня)

CNN — Convolutional Neural Networks (Конволюційні нейронні мережі)

CRNN — Convolutional Recurrent Neural Networks (Згортково-рекурентні нейронні мережі)

DNN — Deep Neural Networks (Глибокі нейронні мережі)

FNN — Feedforward Neural Networks (Фідерні нейронні мережі)

GPS — Global Positioning System (Глобальна система позиціонування)

GRU — Gated Recurrent Unit (Градієнтне вицілювання)

HTTP — Hypertext Transfer Protocol (Протокол передачі гіпертексту)

IDE — Integrated Development Environment (Інтегроване середовище розробки)

JRE — Java Runtime Environment (Середовище виконання Java)

JSON — JavaScript Object Notation (Нотація об'єктів JavaScript)

JVM — Java Virtual Machine (Віртуальна машина Java)

LSTM — Long Short-Term Memory (Довго-короткострокова пам'ять)

MVP — Minimum Viable Product (Мінімально життєздатний продукт)

ReLU — Rectified Linear Unit (Виправлений лінійний елемент)

RFID — Radio-Frequency Identification (Радіочастотна ідентифікація)

RMSE — Root Mean Square Error (Середньоквадратична помилка)

RNN — Recurrent Neural Networks (Рекурентні нейронні мережі)

SQL — Structured Query Language (Мова структурованих запитів)

UML — Unified Modeling Language (Уніфікована мова моделювання)

WMS — Warehouse Management System (Система управління складом)

XSS — Cross-Site Scripting (Міжсайтовий скриптинг)

БД — База Даних

ГІС — Геоінформаційна система

ВСТУП

У сучасному технологічному світі, що стрімко розвивається, роль програмного забезпечення у сфері транспортних послуг, зокрема замовлення таксі, стає дедалі важливішою. Без ефективних систем управління та прогнозування попиту в роботі служб таксі виникають неефективність та неузгодженість. Однак використання програмного забезпечення, заснованого на обробці статистичних даних, може значно полегшити цей процес і забезпечити оптимальне використання ресурсів.

Традиційні способи бронювання, які вимагають зв'язку з оператором таксі або ходіння по місту в пошуках таксі, швидко втрачають актуальність з розвитком IT-індустрії. Споживачі віддають перевагу сервісам, прив'язаним до мобільних додатків. Основна причина цього - зручність. Додатки для таксі дозволяють повністю автоматизувати процес замовлення автомобіля, що призводить до економії коштів як для служби, так і для клієнта.

Прогнозування замовлень таксі вимагає комплексного підходу, який аналізує велику кількість даних, що стосуються попиту на послуги таксі в різних регіонах і в різний час доби, а також враховує різні фактори, такі як події, погода і свята. Програмне забезпечення для прогнозування замовлень повинно забезпечити збільшення ефективності управління автопарком підприємства, що надає послуги індивідуальних містких перевезень, та забезпечити його додаткові конкурентні переваги через впровадження затребуваних функцій клієнтського програмного забезпечення. Досягти цього можна, аналізуючи попередні дані та використовуючи їх для прогнозування майбутнього попиту.

Такий підхід не тільки допомагає підвищити ефективність роботи служб таксі, але й відкриває нові можливості для поліпшення мобільності в сучасних містах і надання користувачам зручного та швидкого доступу до транспортних послуг.

Мета дослідження — розширення функціональних можливостей програмного засобу забезпечення сервісів індивідуальних пасажирських

перевезень шляхом впровадження прогнозування попиту та тарифів на перевезення.

Задачі дослідження:

— аналіз існуючих додатків транспортних послуг та їх функціональних можливостей;

— створення моделі прогнозування з використанням машинного навчання та штучних нейронних мереж;

— вдосконалення методу побудови архітектури підсистеми прогнозування попиту;

— створення мобільного додатку та додавання функціональності генерування прогнозів попиту на послуги;

— тестування програмного засобу та оцінка його ефективності та відповідності сформульованим вимогам до програмного забезпечення.

Об'єкт дослідження — процес прогнозування замовлень транспортних засобів, зокрема таксі, з використанням статистичного аналізу попередніх запитів та інтеграцією із штучними нейронними мережами.

Предмет дослідження — методи та алгоритми аналізу даних, що можуть застосовуватись у програмному засобі для прогнозування замовлень транспортних засобів, та їх вплив на ефективність та якість транспортних послуг. Побудова клієнтського додатку з можливостями прогнозування попиту та цін.

Методи дослідження: статистичний аналіз даних, розробка алгоритмів, програмування, тестування програмного забезпечення.

Наукова новизна — вдосконалено метод побудови архітектури підсистеми прогнозування попиту, який, на відміну від існуючих враховує потоковий збір даних, кластер зберігання великих даних, контейнеризацію для зручного розгортання та безперервної інтеграції, що дозволило розширити функціональні можливості програмного засобу.

Практичне значення одержаних результатів — розроблений програмний засіб сприятиме підвищенню якості та доступності транспортних

послуг для користувачів, підвищить рівень сервісу та задоволеності користувачів сервісом, забезпечить більш ефективне управління ресурсами перевізників.

Апробація результатів магістерської кваліфікаційної роботи: зроблено доповідь на Міжнародній науково-практичній інтернет-конференції Молодь в науці: дослідження, проблеми, перспективи (МН-2024) [1].

Матеріали роботи доповідались та опубліковувались:

Програмний засіб для прогнозування замовлень транспортного засобу на основі аналізу статистики попередніх запитів / І. Ю. Черняховський, О. С. Городецька, Л. А. Савицька // Матеріали МНПІК Молодь в науці: дослідження, проблеми, перспективи (МН-2024), 2024 р. [Електронний ресурс] — https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/19539/16188.

1 АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ГАЛУЗІ ТА ШЛЯХІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Огляд ринку транспортних послуг та його особливості

Транспортна індустрія в умовах стрімкого розвитку технологій та змін споживацьких уподобань пройшла величезні трансформації, особливо в сегменті таксі. Огляд ринку таксі надає нам можливість розглянути ключові аспекти, які визначають сучасну динаміку цієї галузі.

Запровадження мобільних додатків та онлайн-платформ стало справжньою революцією у світі таксі. Пасажири тепер можуть легко та зручно замовляти транспорт, відстежувати рух автомобіля та здійснювати оплату, уникаючи нестачі готівки та непорозумінь з водіями. Сучасні додатки дозволяють не лише замовляти таксі, а й відстежувати рух транспортного засобу в реальному часі. Це надає пасажирам можливість точно визначити час прибуття та готуватися до поїздки. Мобільні додатки та онлайн-платформи для таксі перетворюють рутинні поїздки в справжню насолоду. Ці зручні інструменти зробили таксі доступним та ефективним, а їхні можливості продовжують розширюватися. У світі, де кожна мить має значення, ці технології змінюють спосіб, яким ми мандруємо містом та взаємодіємо з транспортними послугами.

Введення в гру технологічних гігантів, таких як Uber, Bolt та Lyft змінило баланс сил у ринку. Моделі перевезення, де один водій може обслуговувати кількох пасажирів одночасно, відкрили нові можливості та перетворили погляд на власність транспортних засобів. Технологічні гіганти не припиняють своєї роботи на рівні мобільних додатків. Вони внесли значний внесок у розвиток автономних транспортних засобів, електричних автомобілів та систем штучного інтелекту для оптимізації маршрутів та прогнозування попиту. Традиційні таксі-служби також відчувають вплив технологічних гігантів. Конкуренція призводить до необхідності модернізації та підвищення якості послуг, щоб зберегти та здобути нових клієнтів.

Конкуренція на ринку таксі заохочує компанії впроваджувати програми лояльності та різноманітні акції для залучення та утримання клієнтів. Користувачі можуть вигравати знижки, безкоштовні поїздки та інші бонуси, що стимулює їхню вірність вибраному сервісу. Умови конкуренції також стимулюють компанії до диференціації своїх послуг. Деякі фокусуються на розкішних автомобілях та ексклюзивному обслуговуванні, тоді як інші прагнуть до низьких цін та високої доступності. Користувацький досвід став важливим чинником конкурентоспроможності. Інтуїтивний мобільний інтерфейс, точність визначення місцезнаходження та швидке обслуговування стають ключовими аспектами, що привертають та утримують пасажирів. Конкуренція в галузі таксі не лише сприяє покращенню якості та доступності послуг, але й стимулює розвиток новаторських стратегій, сервісів та програм лояльності, роблячи таксі-індустрію ще більш привабливою для пасажирів.

Розгляд ринку таксі не повний без звернення до тенденцій сталого розвитку. Зростаючий інтерес до електромобілів та експерименти з автономними транспортними засобами стають частиною стратегій компаній, спрямованих на зменшення впливу на довкілля та оптимізацію вартості експлуатації флоту. Впровадження електромобілів та автономних транспортних засобів призводить до зміни моделі бізнесу у галузі таксі. Компанії шукають шляхи оптимізації вартості обслуговування та максимізації ефективності в цій сфері. Тенденції електромобілів та автономних транспортних засобів переписують правила гри у галузі таксі. Ці інновації визначають нові стандарти у сфері транспортних послуг та ставлять питання про призначення та майбутнє масових перевезень.

Ринок таксі — це динамічна система, що поєднує в собі технології, конкуренцію та взаємодію з спільнотою. Огляд його особливостей вказує на постійні зміни та необхідність адаптації для виживання та розвитку у сучасному світі транспортних послуг. Ринок таксі визначається не лише конкуренцією, але і неперервними змінами, що відбуваються під впливом технологічних інновацій та вимог споживачів. Галузь постійно пристосовується до нових реалій, і успішні

гравці — ті, хто активно впроваджує та адаптується до нових тенденцій, враховуючи висновки та виклики ринку.

1.2 Роль технологій у розвитку транспортних послуг

Індустрія транспортних послуг переживає період трансформацій, завдяки стрімкому розвитку технологій. Впровадження інновацій визначає новий порядок речей та взаємодії між водіями та пасажирами. Давайте глибше розглянемо роль технологій у розвитку цієї галузі та їхній вплив на спосіб, яким ми подорожуємо. Введення мобільних додатків в таксі-індустрію стало епохальним змінником. Пасажири тепер можуть замовляти таксі за декілька натискань на смартфоні. Це полегшує процес замовлення та дозволяє ефективно керувати флотом для водіїв та служб таксі. Використання геолокаційних систем дозволяє оптимізувати маршрути та зменшити час очікування для пасажирів. Алгоритми прогнозування попиту допомагають водіям максимізувати використання ресурсів, щоб забезпечити ефективний та швидкий транспорт. Впровадження безготівкових платежів забезпечує безпеку та зручність для пасажирів. Це важливий аспект у сучасному світі, де швидкі та безпечні транзакції стають визначальними. Використання аналітики та штучного інтелекту дозволяє збирати та аналізувати дані про використання транспорту. Це важливо для оптимізації флоту, передбачення попиту та розробки стратегій для поліпшення обслуговування.

Геолокаційні системи відіграють визначальну роль у перетворенні таксі-індустрії, надаючи безліч переваг для як пасажирів, так і водіїв. Давайте розглянемо, як ці технології оптимізують маршрути та зменшують час очікування, забезпечуючи більш зручний та ефективний процес пасажирських перевезень. Геолокаційні системи в сучасних таксі-додатках надають надзвичайну точність визначення місцезнаходження, що робить можливим ефективне взаємодію з пасажирами та оптимізацію роботи водіїв. Це сприяє швидкості та простоті процесу замовлення. Геолокаційні системи дозволяють автоматично обирати найоптимальніший маршрут для подорожі. Алгоритми

враховують показники трафіку в реальному часі, дорожні роботи та інші фактори, що можуть впливати на швидкість руху. Геолокаційні системи не лише оптимізують маршрути, але й забезпечують пасажирам безпеку. Інформація про місцезнаходження в реальному часі дозволяє відстежувати маршрут та подавати допомогу в разі потреби. Геолокаційні системи стали невід'ємною частиною розвитку таксі-індустрії, забезпечуючи ефективність та комфорт для користувачів. Ці технології не тільки роблять подорожі більш зручними, але й сприяють ефективній роботі таксі-сервісів, зменшуючи час очікування та оптимізуючи маршрути для швидших та ефективних подорожей.

Впровадження аналітики та штучного інтелекту в таксі-індустрію визначає нові стандарти ефективності та якості обслуговування. Давайте розглянемо, як ці технології впливають на передбачення попиту та оптимізацію роботи сервісів таксі. Аналітичні інструменти, підтримані штучним інтелектом, аналізують величезні обсяги даних про попит на послуги таксі. Вони враховують часові та сезонні зміни, локальні події та інші фактори для точного передбачення попиту на послуги перевезень. Штучний інтелект аналізує геолокаційні дані та інформацію про дорожні умови в реальному часі для оптимізації маршрутів. Це не лише зменшує час подорожі, але й збільшує плавність руху, що робить подорож більш комфортною. Системи аналітики можуть аналізувати попит та пропозицію для автоматичного налаштування тарифів у режимі реального часу. Це дозволяє підтримувати конкурентоспроможні ціни та стимулювати водіїв працювати в пікові часи. Штучний інтелект аналізує дані про користувачів, враховуючи їхню попередню історію поїздок та уподобання. Це дозволяє підлаштовувати сервіс до індивідуальних потреб та створювати персоналізовані пропозиції. Штучний інтелект може передбачити час тривалості поїздки, враховуючи різні параметри, такі як відстань, трафік, час доби та інші фактори. Це дозволяє пасажирам точно розраховувати час прибуття та водіям — планувати свій робочий графік. Аналітика та штучний інтелект перетворюють таксі звичайних перевезень у високотехнологічну галузь. Забезпечуючи точне

передбачення та оптимізацію, ці технології підвищують ефективність послуг та створюють персоналізовані рішення для кожного користувача.

Введення програм лояльності та систем знижок стає стратегічним кроком для таксі-сервісів, спрямованим на підвищення вірність клієнтів та стимулювання нових користувачів. Давайте розглянемо, як ці програми впливають на збереження та залучення клієнтів у таксі-індустрії. Програми лояльності, які передбачають нарахування бонусів або балів за кожну поїздку, стимулюють пасажирів обирати конкретний таксі-сервіс. Збір балів, які потім можна обміняти на знижки чи додаткові послуги, робить користування таксі більш вигідним та привабливим. Програми лояльності можуть передбачати систему поетапних знижок або відсоткових облікових ставок для постійних клієнтів. Чим частіше використовуєш послуги, тим більше вигоди та привілеї тобі надаються. Таксі-сервіси можуть запускати сезонні чи спеціальні пропозиції для клієнтів, які беруть участь у їхніх програмах лояльності. Це може включати знижки під час свят, безкоштовні поїздки на день народження чи інші ексклюзивні привілеї. Таксі-сервіс може співпрацювати з іншими компаніями для створення спільних програм лояльності. Знижки чи бонуси за використання послуг інших партнерів можуть розширити залучення та утримання клієнтів. Програми лояльності та знижок у таксі-сфері дозволяють компаніям залучати та утримувати клієнтів, роблячи користування послугами більш вигідним та приємним. Комплексний підхід, який враховує різні аспекти взаємодії з клієнтами, стає ключем до успішного збереження та приваблення нових користувачів. Технології відіграють вирішальну роль в сфері вантажних перевезень, роблячи їх більш ефективними, безпечними і економічно вигідними. Автоматизовані системи управління складами (WMS) та роботизоване обладнання допомагають автоматизувати підготовку і сортування вантажів, підвищуючи продуктивність і знижуючи ризик людської помилки GPS-слідкування та системи RFID дають можливість відслідковувати місцезнаходження вантажів в реальному часі, підвищуючи прозорість і безпеку доставки.

1.3 Огляд популярних програмних засобів

Uber — американський постачальник послуг мобільності [1]. Ця компанія розташована у Сан-Франциско і працює приблизно в 72 країнах і 10 600 містах.

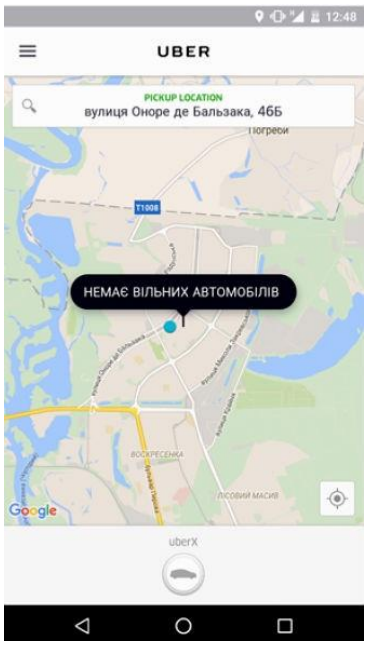


Рисунок 1.1 — Знімок екрану додатку

Його послуги включають виклики таксі, доставку їжі, посилок, кур’єрів, доставку вантажів, оренду електровелосипедів і скутерів, а також перевезення поромів у партнерстві з місцевими операторами.

Uber не володіє транспортними засобами, замість цього він стягує комісію за кожне бронювання. Тарифи встановлюються клієнтом заздалегідь, але можуть бути змінені за допомогою динамічної моделі ціноутворення на основі місцевого попиту та пропозиції на момент бронювання.

Bolt — естонська мобільна компанія [2], що надає послуги з використання автомобілів, мікромобілів, каршерингу та доставки їжі зі штаб-квартирою в Таллінні та працює в більш ніж 400 містах у більш ніж 45 країнах Європи, Африки, Західної Азії та Латинської Америки.

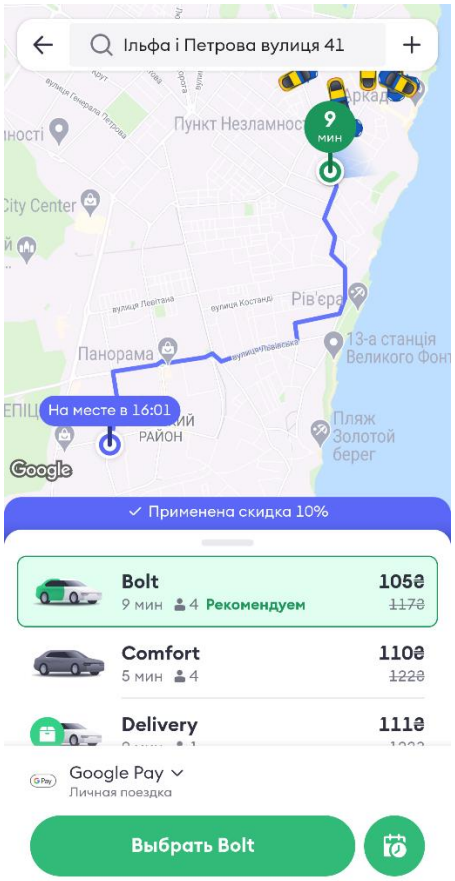


Рисунок 1.2 — Знімок екрану додатку Bolt

У компанії 100 мільйонів клієнтів по всьому світу і понад 2,5 мільйона водіїв, які використовують платформу Bolt Driver для подорожей. Bolt відзначається своїм підходом до інновацій та стратегією залучення клієнтів. З конкурентоспроможними цінами, розширеною функціональністю та програмами лояльності, Bolt виявляється привабливим вибором для тих, хто шукає зручні та доступні перевезення. Bolt заздалегідь показує приблизну вартість поїздки, що допомагає користувачам уникнути несподіваних витрат. Завдяки великій мережі водіїв, Bolt може забезпечувати швидке замовлення поїздок, знижуючи час очікування. Крім стандартних поїздок, Bolt часто пропонує різні види послуг, такі як доставка їжі, прокат електроскутерів та більш комфортабельні варіанти перевезення. Bolt інколи пропонує опції для замовлення електромобілів або інших екологічно чистих видів транспорту. Bolt може пропонувати знижки, бонуси за реферали або програми лояльності для постійних користувачів. Bolt

зазвичай надає інформацію про водія та можливість оцінювання поїздок, що сприяє підвищенню відповідальності та безпеки.

Lyft є Американським постачальником транспортних послуг [3], який розробляє, продає та керує мобільним додатком, який пропонує послуги таксі, оренду автомобілів, скутерів, велосипедів, автомобілів та доставку їжі. Він розташований у Сан-Франциско, Каліфорнія, і працює в 647 містах США. У цієї компанії немає автомобілів; Замість цього він стягує комісію за кожне бронювання. Тарифи встановлюються клієнтом заздалегідь, але можуть бути змінені за допомогою динамічної моделі ціноутворення на основі місцевого попиту та пропозиції на момент бронювання. Система Lyft пропонує чіткі та прозорі тарифи, які розраховуються перед підтвердженням поїздки. Це дозволяє користувачам заздалегідь знати вартість послуги. Lyft використовує систему зворотного зв'язку та рейтингів для оцінки якості обслуговування водіїв та пасажирів. Це сприяє створенню надійної та безпечної спільноти користувачів. Lyft відрізняється від конкурентів, таких як Uber, своєю фокусованістю на дружбу та підтримку користувачів. Вони вирізняються своєю дружелюбною атмосферою та можливістю водіїв більше взаємодіяти з пасажирами, створюючи більш особистий досвід. Крім того, Lyft активно працює над поліпшенням безпеки, забезпечуючи перевірку водіїв та використання заходів безпеки під час поїздок.

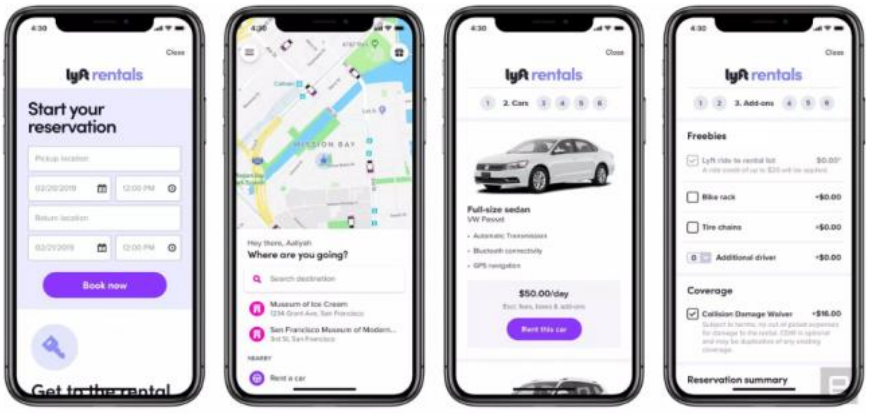


Рисунок 1.3 — Інтерфейс Lyft

Проаналізувавши 3 аналоги мобільних додатків для замовлення таксі, наведемо порівняльну таблицю переваг та недоліків цих програмних продуктів.

Таблиця 1.1 — Переваги та недоліки мобільних додатків «Uber», «Lyft», «Bolt»

Назва додатка	Переваги	Недоліки
Uber	Безготівкова система оплати, програма лояльності, широкий вибір послуг	Проблеми із безпекою, висока вартість за піком.
Lyft	Колективні поїздки прозорі тарифи.	Великий час очікування, додаток працює нестабільно.
Bolt	Зв'язок з оператором, конкурентні тарифи;	Збої додатку, проблеми з безпекою.

1.4 Базові поняття штучних нейронних мереж

Теорія нейронних мереж [4] включає в себе розуміння того, як мозок може навчатися та адаптуватися, використовуючи велику кількість простих обчислювальних одиниць, званих нейронами. Штучні нейронні мережі намагаються імітувати цей процес, створюючи математичні моделі для розв'язання складних задач обчислювального характеру.

Основною будівельною одиницею нейронної мережі є штучний нейрон, який називається перцептроном. Він отримує один або кілька входів (часто називаються синапсами у біологічних системах) і генерує вихід (званий аксоном у біологічних системах). Вхід кожного нейрона зазвичай множиться на ваги, які визначають важливість кожного сигналу, а потім сумується. Ця сума може бути модифікована за допомогою зсуву і після цього проходить через активаційну функцію, що визначає вихід нейрона.

Активаційна функція визначає, як нейрон має реагувати на суму вхідних сигналів. Ця функція може бути простою, як функція сходинки (все або нічого),

або складнішою, як сигмоїдна функція, гіперболічний тангенс або ReLU (Rectified Linear Unit).

Зворотний поширення помилки (Backpropagation) — алгоритм для навчання нейронних мереж шляхом коригування ваг зв'язків на основі градієнтів функції втрат. Градієнти обчислюються зворотнім проходженням через мережу.

Для навчання моделі використовують тренувальний набір даних, а для оцінки її продуктивності — валідаційний і тестовий набір даних.

Перенавчання (Overfitting) і недонавчання (Underfitting) представляють різний баланс у здатності моделі адаптуватися до навчальних даних та її здатності узагальнювати до нових даних [5].

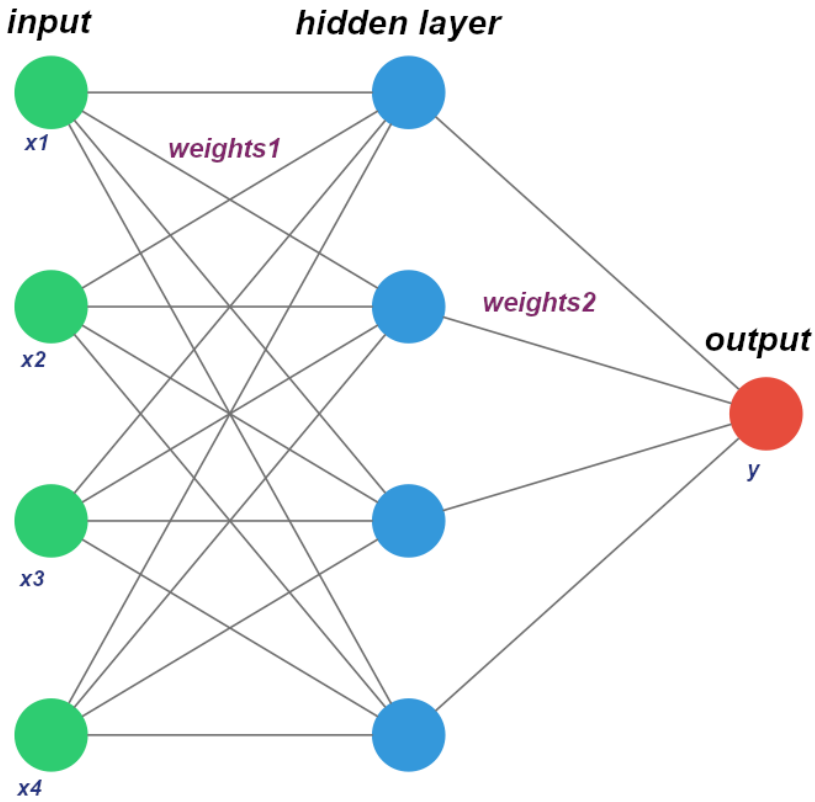


Рисунок 1.4 — Структура штучної нейронної мережі

Перенавчання виникає, коли модель стає надто специфічною до навчальних даних, що призводить до доброї продуктивності на навчальних

даних, але поганої на нових даних. В інших словах, модель "запам'ятовує" навчальні дані замість того, щоб "розуміти" їх.

Недонавчання виникає, коли модель надто спрощена або недостатньо навчена на навчальних даних, що призводить до поганої продуктивності як на навчальних, так і на нових даних. Модель не може виявити важливі закономірності в даних через її обмежену складність.

Архітектура мережі описує, як нейрони з'єднані між собою. Це може включати кілька шарів, кожен з яких складається з багатьох нейронів. Шари можуть бути:

— вхідним шаром (Input Layer): де дані вводяться в мережу;

— прихованими шарами (Hidden Layers): де відбувається обробка;

— вихідним шаром (Output Layer): де отримуються результати.

Нейронні мережі (Neural Networks) мають різні архітектури, призначені для різних завдань. Ось кілька ключових архітектур, які використовуються в глибокому навчанні:

Фідерні (Feedforward) нейронні мережі (FNN): Це найпростіша форма нейронної мережі, де дані переміщаються з вхідного шару через приховані шари до вихідного шару. Вони добре підходять для класифікації і регресії.

Зворотнє поширення (Backpropagation) нейронні мережі: Це розширення фідерних мереж, де використовується алгоритм зворотного поширення помилки для тренування мережі. Вони дозволяють вирішувати багато завдань машинного навчання, включаючи класифікацію, регресію та генерацію зображень.

Конволюційні нейронні мережі (CNN): Вони оптимізовані для обробки зображень і інших двовимірних даних. Вони використовують спеціалізовані шари згортки для виявлення важливих ознак у зображеннях.

Рекурентні нейронні мережі (RNN): Ця архітектура призначена для роботи з послідовними даними, такими як текст або часові ряди. Вони мають зв'язки між внутрішніми шарами, які дозволяють зберігати попередні стани та інформацію.

Довго-короткострокова пам'ять (LSTM) та Градієнтне вицілювання (GRU): Ці розширення RNN допомагають вирішити проблему зниклого градієнта і покращують здатність моделі зберігати довгострокову залежність в даних.

Autoencoders — нейронні мережі, які використовуються для зменшення розмірності даних та витягування важливих ознак. Вони використовуються для зменшення шуму в даних або для побудови рекомендаційних систем.

Глибокі нейронні мережі (DNN) — мережі мають багато прихованих шарів і використовуються для рішення складних завдань, таких як обробка природної мови, візуальне розпізнавання та генерація тексту.

Згортково — рекурентні нейронні мережі (CRNN) — ця архітектура поєднує конволюційні та рекурентні шари і дозволяє ефективно обробляти як зображення, так і послідовності даних.

Ці архітектури можуть бути комбіновані і налаштовані для різних задач. У кожної архітектури є свої переваги і недоліки, і вибір залежить від конкретної задачі та доступних даних.

Застосування нейронних мереж у аналізі великих даних приносить численні переваги. Нейронні мережі можуть самостійно вивчати та вдосконалювати свої прогнози на основі навчальних даних, без необхідності вручну програмувати правила або алгоритми. Нейронні мережі особливо ефективні в обробці неструктурованих даних, таких як зображення, аудіо та текст, що робить їх ідеальними для роботи з різноманітними наборами великих даних. Після навчання на достатньо великому наборі даних нейронні мережі здатні узагальнювати знання і робити достовірні прогнози на нових даних.

Нейронні мережі можуть виявляти складні, неявні зв'язки та залежності між змінними, які можуть бути не очевидними за традиційними статистичними методами. Нейронні мережі добре піддаються масштабуванню, що дозволяє аналізувати дедалі більші набори даних за допомогою додавання обчислювальних ресурсів. Вони можуть ефективно працювати з багатовимірними даними та виокремлювати складні шаблони.

Нейронні мережі можуть адаптуватися до змін у данних через свою гнучку структуру, яка дозволяє оновлювати ваги відповідно до нових тенденцій. Архітектура нейронних мереж природньо підтримує паралельну обробку, що дозволяє швидко обробляти великі об'єми даних. Нейронні мережі здатні обробляти невизначеність і неповноту в даних, використовуючи, наприклад, мережі, засновані на теорії нечітких множин. Здатність аналізувати великі даних робить нейронні мережі інструментом універсального застосування у різноманітних галузях, від фінансів до охорони здоров'я.

Нейронні мережі здатні вчитися на основі навчальних даних. Це означає, що вони можуть адаптуватися до нових завдань та отримувати знання зі змінних даних. Нейронні мережі відмінно працюють з неструктурованими даними, такими як зображення, аудіо та текст. Вони використовуються в розпізнаванні образів, обробці природної мови та аналізі великих об'ємів текстової інформації.

Нейронні мережі можуть узагальнювати знання і робити прогнози на нових даних після навчання на історичних даних. Вони здатні виявляти складні та неявні залежності в даних, які можуть бути важко визначити за допомогою традиційних методів. Нейронні мережі ефективно працюють з багатьма вимірними даними і великими об'ємами інформації. Архітектура нейронних мереж сприяє паралельній обробці, що дозволяє прискорювати аналіз великих обсягів даних за допомогою багатьох обчислювальних ресурсів. Також, вони здатні адаптуватися до змін у даних та оточенні, що робить їх важливими в реальних часах та змінних середовищах. Більше того, нейронні мережі можуть виявити та врахувати різні шаблони та закономірності в даних, навіть якщо вони складні і змінні. Розвиток глибокого навчання і штучних нейронних мереж дозволяє розробляти більш потужні та точні моделі для різних завдань

Нейронні мережі не лише стали важливим інструментом для аналізу великих даних, але й створюють нові можливості в різних сферах досліджень та розвитку технологій, залишаючись на передньому краї інновацій у сфері аналітики великих даних і машинного навчання.

1.5 Огляд сучасних регресійних моделей

Прогнозування у сфері транспортних послуг відіграє важливу роль для планування, управління трафіком, оптимізації маршрутів та загальної ефективності транспортних систем. Існують різні методи, які можуть використовуватися для прогнозування різних аспектів транспортних послуг [5].

ARIMA (Авторегресивні моделі зі стохастичною компонентою і ковзним середнім) — потужний метод для прогнозування часових рядів, який використовує попередні значення часового ряду для передбачення майбутніх значень, також у транспортних послугах ARIMA може бути використана для прогнозування попиту на послуги на підставі історичних даних.

Експоненційне згладжування — група методів для прогнозування часових рядів, які приділяють більше ваги більш новим даним. Це може бути корисним для короткострокового та середньострокового прогнозування трафіку або попиту на транспортні послуги.

Регресійний аналіз дозволяє встановлювати залежність між різними змінними та попитом на транспортні послуги. Наприклад, можна аналізувати вплив різних факторів, таких як ціни на пальне, населення, та інші на обсяги пасажиропотоку або кількість перевезень.

Random Forest — ансамблевий метод, який може бути використаний для прогнозування попиту на транспортні послуги. Він добре справляється з великою кількістю факторів та нелінійною залежністю між ними.

Градієнтний бустинг дозволяє покращити точність прогнозу шляхом комбінування багатьох моделей. В транспортних послугах — може бути корисним для вирішення складних завдань прогнозування.

Глибоке навчання та рекурентні нейронні мережі (RNN) можуть бути використані для прогнозування трафіку, часу подорожі та інших параметрів транспортних послуг. Вони особливо корисні для аналізу послідовних даних.

Методи оптимізації: лінійне та цілочисельне програмування можуть бути використані для оптимізації розкладів та маршрутів з метою максимізації використання ресурсів та мінімізації витрат.

Симуляційне моделювання — орієнтовані моделі та мікросимуляція можуть бути використані для моделювання поведінки окремих учасників транспортної системи, їх взаємодії та аналізу впливу різних стратегій управління.

Географічні інформаційні системи (ГІС) — використовуються для аналізу геопросторових даних, побудови маршрутів, визначення оптимальних локацій для транспортних послуг та аналізу географічних факторів, що впливають на попит.

2 ТЕОРЕТИЧНЕ ДОСЛІДЖЕННЯ, ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

2.1 Вибір моделей машинного навчання та структури нейронних мереж для прогнозування попиту

Спочатку розглянемо структуру даних, що є базисом для прогнозування попиту на послуги таксі у окремо взятій службі, що збирає та агрегує дані про поїздки.

Початковий датасет містить наступні стовпчики:

— datetime — дата та час з точністю до однієї години.;

— downtown — ознака середмістя або віддалені райони;

— temp — температура повітря, що була зареєстрована у відповідний період;

— public_event — ознака того, що відбувалися громадські/масові заходи;

— downfall — наявність опадів у відповідний період;

— trips — кількість поїздок, що були замовлені за період.

Датасет охоплює важливі фактори, які можуть впливати на кількість замовлень. Він містить записи за два повних роки з точністю до однієї години, що дає детальне розуміння сезонних та щоденних тенденцій. Стовпчик "дата та час" відображає часові рамки кожного запису. Ознака "downtown" розрізняє між районами, зокрема, між центральною частиною міста та віддаленішими районами, що може вказувати на різні моделі споживання таксі. "Temp" вказує на температуру повітря, яка може впливати на бажання людей використовувати таксі. "Public_event" є показником відбування громадських або масових заходів, що, як правило, збільшує споживання таксі. "Downfall" відзначає наявність опадів, що також може спонукати людей до використання таксі замість альтернативних видів транспорту. І, нарешті, "trips" — кількість поїздок, замовлених за вказаний період, яка є цільовою змінною і відображає фактичний попит на таксі. Загалом, ці дані забезпечують комплексний погляд на фактори,

що впливають на вибір таксі як транспортного засобу, і можуть бути використані для точного прогнозування попиту.

В якості регресійної моделі розглядатимемо алгоритм підвищення градієнтів, зокрема градієнтне підсилення на деревах рішень [6,7]. Основна ідея полягає у побудові ансамблю послідовних моделей, де кожна наступна модель виправляє помилки попередньої.

Нехай y_i є істинним значенням, а $\widehat{y_i}^{(t)}$ — прогнозованим значенням моделі на t — му кроці. Ціль алгоритму — мінімізувати різницю між цими значеннями.

Функція втрати $L(\theta)$ визначає різницю між фактичними та прогнозованими значеннями. Для регресії часто використовується квадратична функція втрат, яка виражається формулою (2.1)

$$L(\theta) = \sum_{i=1}^{n} \left(y_i - \widehat{y_i}^{(t)}\right)^2, \tag{2.1}$$

де n — кількість спостережень.

Модель градієнтного підсилення конструюється послідовно. На кожному кроці t будується нове дерево рішень f_t, яке прогнозує градієнт функції втрати по відношенню до попереднього прогнозу.

Прогноз на t-му кроці розраховується за формулою (2.2)

$$\widehat{y_i}^{(t)} = \widehat{y_i}^{(t-1)} + \eta \cdot f_t(x_i), \tag{2.2}$$

де η — швидкість навчання, яка контролює внесок кожного дерева.

На кожному кроці виконується оптимізація для визначення оптимальних параметрів дерева f_t.

Алгоритм XGBRegressor включає додаткові параметри регуляризації для запобігання перенавчанню. Регуляризація в XGBoost включає в себе два типи регуляризації:

— регуляризація L1 (lasso) додає штраф до абсолютних значень ваг;

— регуляризація L2 (гребінь) додає штраф до квадратів ваг.

Ці регуляризатори додаються до функції втрат, забезпечуючи баланс між здатністю моделі вловлювати складні закономірності в даних та запобіганням перенавчанню.

Альтернативою XGBRegressor може бути нейронна мережа, що відома як багатошаровий перцептрон. Мережа, як й багато інших типів, складається із вхідного шару, прихованих шарів та вихідного шару.

Вхідний шар — кількість нейронів відповідає кількості особливостей (фіч) вхідних даних.

Приховані шари — два або більше шари повнозв'язних нейронів, кількість нейронів у прихованих шарах залежить від складності задачі.

Вихідний шар — один нейрон, оскільки це задача регресії.

Вихідний сигнал — прогнозоване числове значення.

Функції активації, що використовуються:

— приховані шари ReLU (Rectified Linear Unit) або її варіації, як-от Leaky ReLU чи ELU;

— вихідний шар лінійна активація, оскільки виходом є неперервне числове значення.

Функція втрат визначається виразом (2.3)

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L\left(y^i, f\left(x^i, \theta\right)\right), \tag{2.3}$$

де N — кількість елементів навчальної вибірки;

θ — параметр відображення $f\ (x)$;

x^i— вектор ознак і-го зразке;

y^i— відповідне і-му зразку значення залежної змінної;

L — втрата функції втрат.

Є багато видів функцій втрат у навчанні з учителем, як-от квадрат евклідової відстані, кросентропія, втрата контрасту (contrast loss), завісні втрати (hinge loss), приріст інформації тощо. Але найчастіше використовується середньо – квадратичне відхилення (RMSE) [8].

Ця нейронна мережа може ефективно вирішувати задачі регресії, аналізуючи складні нелінійні взаємозв'язки в даних. Її гнучкість дозволяє адаптуватися до різноманітних наборів даних і завдань прогнозування.

2.2 Розвідка даних та створення моделі прогнозування

Відповідно до першого розділу, ключовою особливістю проекту є можливість прогнозування попиту на послуги перевезення. Тому доцільно розглянути розвідку даних та побудову самої моделі прогнозування до моделювання та проектування клієнтського додатку та серверного API, оскільки саме створення моделі визначає потрібні та суттєві дані для створення прогнозу, динамічність прогнозу та інші аспекти, пов’язані як с прогнозуванням, так й з використанням отриманого прогнозу в подальшому функціонуванні додатку.

Тут та далі будемо мати на увазі, що прогноз генерується у відносних одиницях, де за базовий рівень взято середнє значення попиту за останній період. Це дозволить на підставі отриманого прогнозу легше формувати прогноз тарифів у клієнтському додатку.

З ознак, що включені до датасету та перелічені в попередньому підрозділі, температуру зовнішнього середовища та кількість поїздок є чисельними ознаками, інші – категоріальними. Температуру зовнішнього середовища залишаємо без змін, кількість поїздок стандартизуємо, але в спосіб що відрізняється від стандартного – середнє значення приймаємо не за 0, а за 1.

Графік розподілу стандартизованих таким чином даних про кількість поїздок показано на рис. 2.1.

З візуального розподілу на рис. 2.1 можна зробити висновок, що стандартизовані дані в більшій своїй частині підпорядковуються нормально розподілу, але й чуттєві відхилення в діапазоні значень [0; 0.5]

Для вивчення впливу факторів часу доби, календарного сезону, можливості збільшення попиту у вихідні, святкові дні, час пік та, можливо, у високий туристичний сезон, виділимо додаткові ознаки з наявної вже інформації.

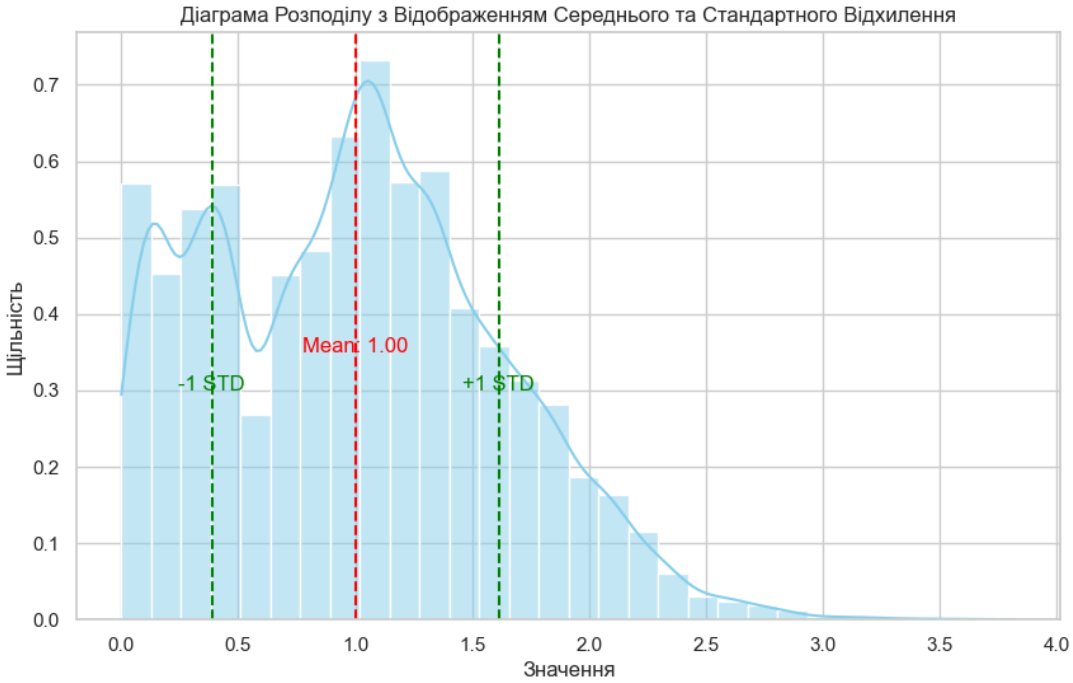


Рисунок 2.1 — Стандартизовані дані про кількість поїздок

Структура датасету після перетворень буде виглядати наступним чином:

— datetime – дата та час з точністю до однієї години;
— downtown – ознака середмістя – True, або віддалені райони - False;
— temp – температура повітря;
— public_event – ознака того, що відбувалися громадські/масові заходи;
— downfall – наявність опадів у відповідний період;
— month – місяць року;
— hour – година доби;

— season – сезон (весна, літо, осінь, зима);

— weekday – день тижня;

— week_number – номер тижня у році;

— day – день у місяці;

— trips – кількість поїздок, що були замовлені за період. Цільова змінна.

Розглянемо залежність кількості поїздок від часу доби, графічно вона показана на рис. 2.2. Зробимо це помісячно, врахувавши, що залежності можуть відрізнятись від пори року.

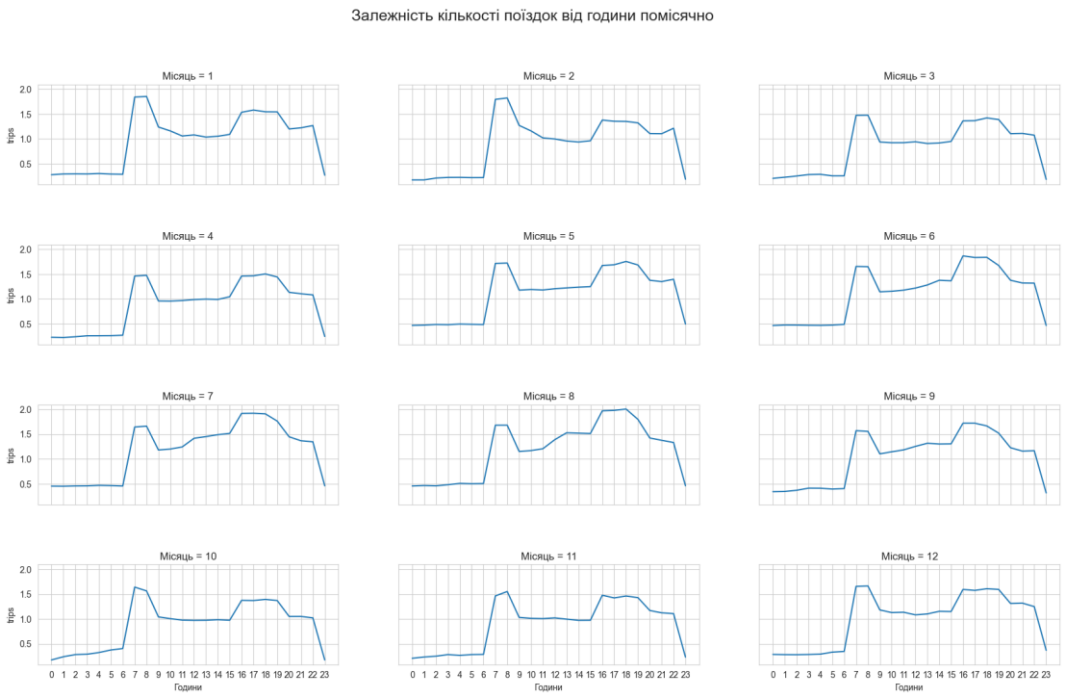


Рисунок 2.2 — Залежність кількості поїздок від часу доби (помісячно)

З графічної інтерпретації можна зробити висновок, що в кількості замовлень є піки, що відповідають певним годинам, й не залежать від місяця в році. З урахування того, що годину не можна вважати кількісною змінною для цієї задачі, розділимо добу на кілька періодів, які потім в практичній реалізації кодуватимемо з використанням OneHotEncoder [9].

Побудуємо аналогічні залежності, але замістимо місяць року на день тижня. Графічна інтерпретація залежностей показана на рис. 2.3.

З побудованих графіків можна зробити висновок, що день тижня впливає на розподіл кількості замовлень у вихідні дні не спостерігається суттєвих ранкових та вечірніх піків навантаження. Тому треба враховувати залежність від дня тижня. Оскільки дні тижня не є кількісними змінними (середа не менше суботи), тому перед побудовою моделі регресії необхідно буде знов таки використати OneHotEncoder. Використання OneHotEncoder для представлення днів тижня як категоріальних змінних у моделі регресії є логічним підходом. Цей метод дозволяє ефективно кодувати категорії днів тижня, не вводячи неправильної числової ієрархії. Під час використання OneHotEncoder важливо також враховувати, як це може вплинути на розмір та склад даних. При додаванні багатьох категоріальних змінних кількість колонок може зростати, що може вплинути на обчислювальні витрати та потреби у пам'яті. Також, деякі алгоритми можуть вимагати додаткових уваг при роботі з великою кількістю категорій.

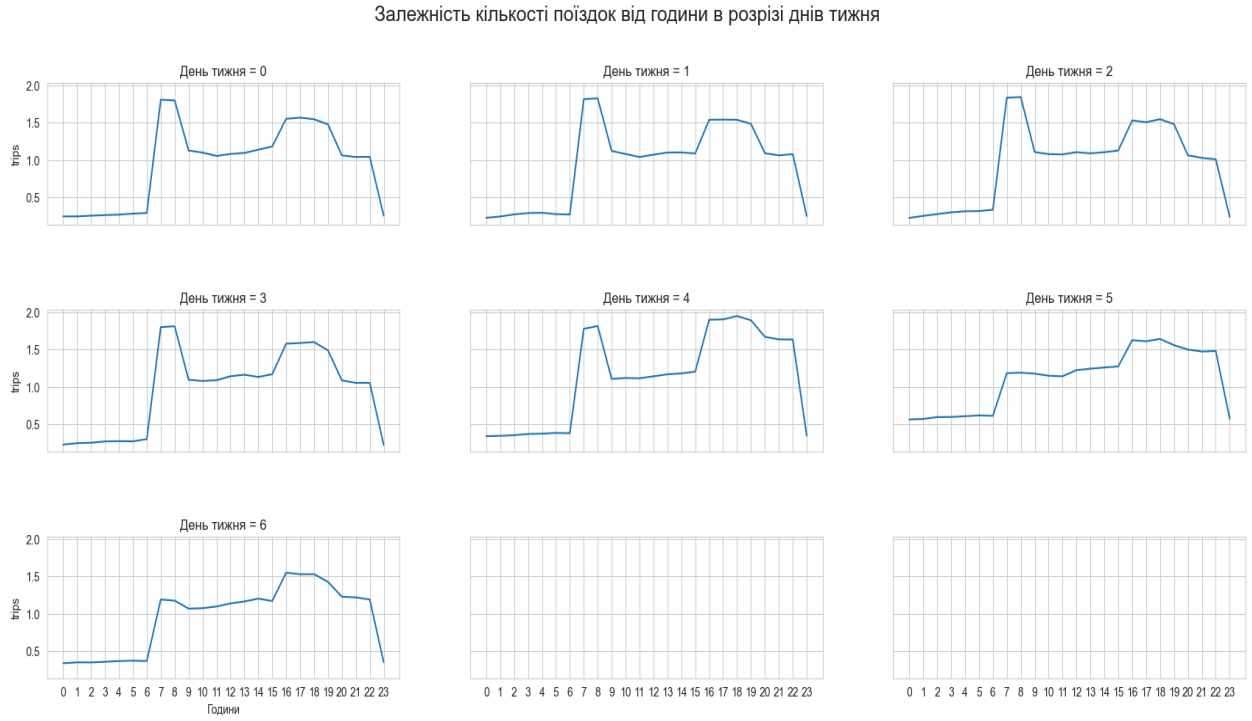


Рисунок 2.3 — Залежність кількості поїздок від часу доби (по днях тижня)

Вияснимо, чи мають вплив змінні downfall, та public_event, які присутні в датасеті. Графічно залежність від downfall показано на рис. 2.4.

Рисунок 2.4 — Графічна інтерпретація впливу змінної downfall

Можна зробити висновок, що вплив змінної downfall є, але треба з'ясувати, чому характер графіків, що відповідають літнім місяцям відрізняється від графіків для місяців інших пор року. Для цього порахуємо кількість міток down-fall в розрізі місяців року. Залежність графічно показана на рис. 2.5.

Рисунок 2.5 — Кількість міток downfall помісячно

Відмінність графіків для літнього періоду пояснюється суттєво меншою кількістю міток downfall у порівнянні з іншими порами року.

Побудуємо тепер аналогічні рис. 2.4 графіки, але для булевої змінної public_event. Залежності представлено на рис. 2.6.

Можна зробити висновок, що вплив змінної public_event є, а самі події відбувалися протягом періоду, що розглядається, у вечірній час. Це пояснює відсутність графіків для інших частин доби.

Тепер вивчимо залежність кількості поїздок від температури повітря, що присутня в таблиці даних. Графік залежності від температури представлено на рис. 2.7.

Залежність кількості поїздок від температури наявна: можна спостерігати пік на мінус 5, початок сталого зростання на +15, та різку зміну швидкості зростання на +27. Але треба вивчити залежність більш детально, оскільки різке зростання після +27 повністю пояснюється схильністю пасажирів перемикатися з громадського транспорту на таксі у період спеки. Але не мають настільки ж очевидного пояснення інші дві зміни динаміки.

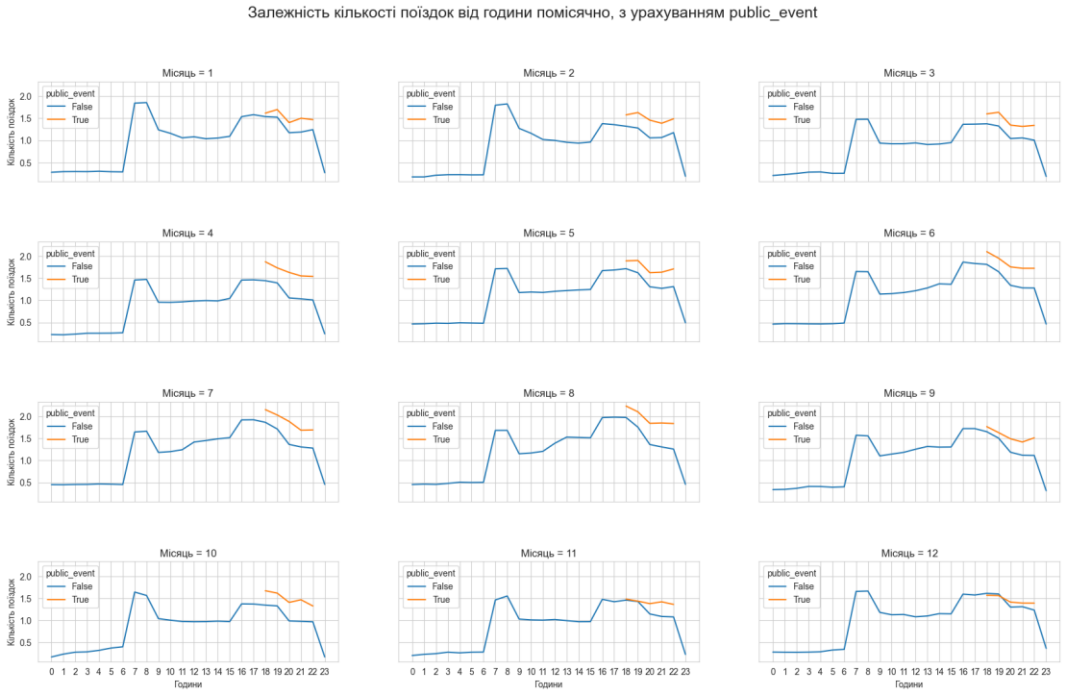


Рисунок 2.6 — Графічна інтерпретація впливу змінної public_events

Для цього для кожної пори року побудуємо погодинний графік середньої температури.

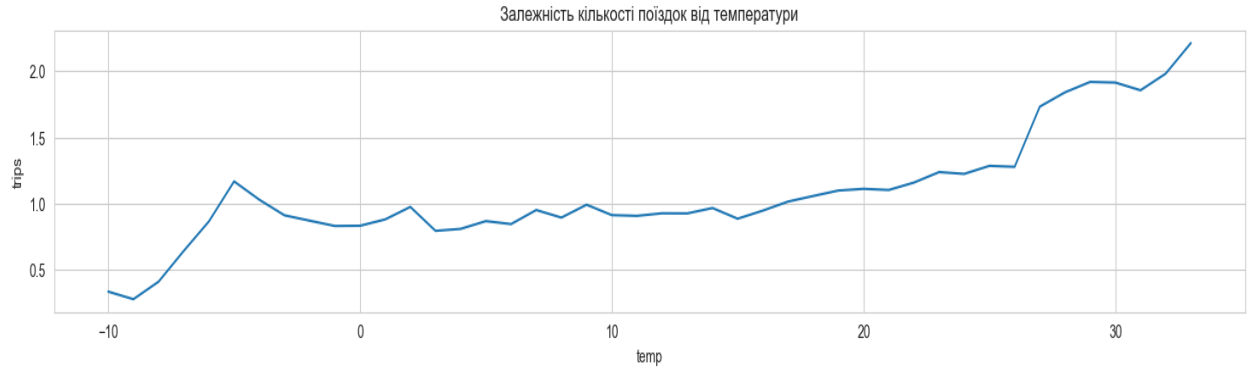


Рисунок 2.7 — Залежність середньої кількості поїздок від температури

З побудованої візуалізації можна зробити висновок, що менше мінус 5 днем температура може становити тільки взимку. Спробуємо побудувати залежність кількості поїздок від температури з розподілом по частинам доби, які було введено вище.

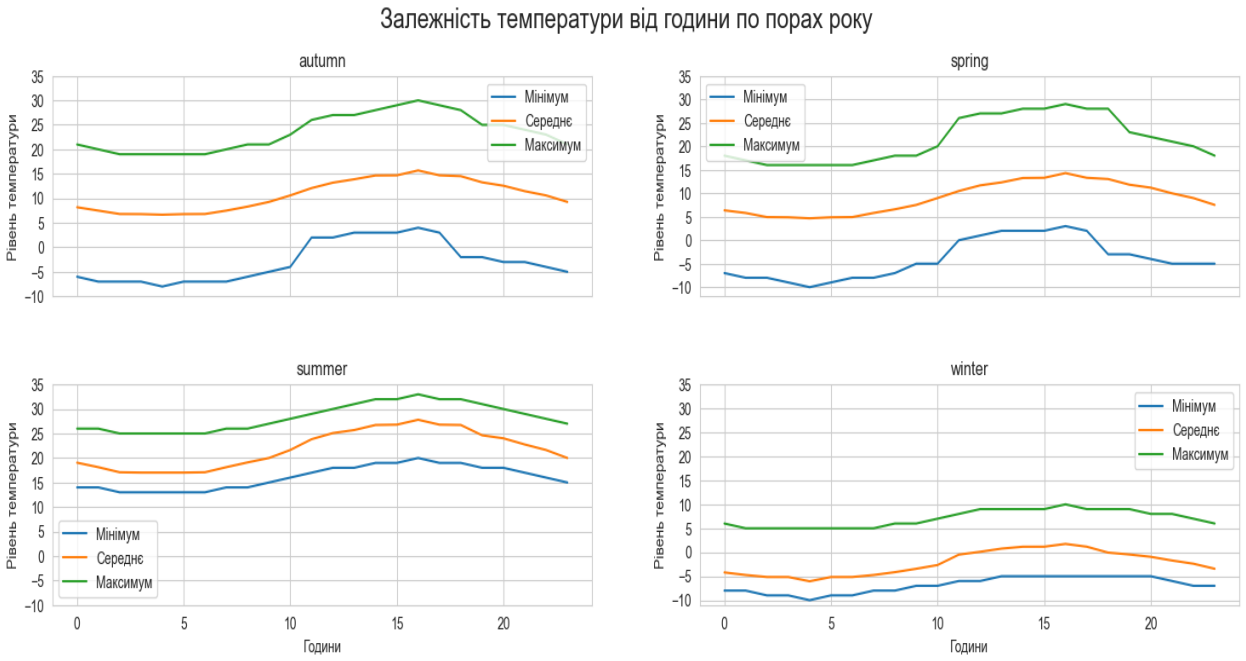


Рисунок 2.8 — Залежність температури від години по порах року.

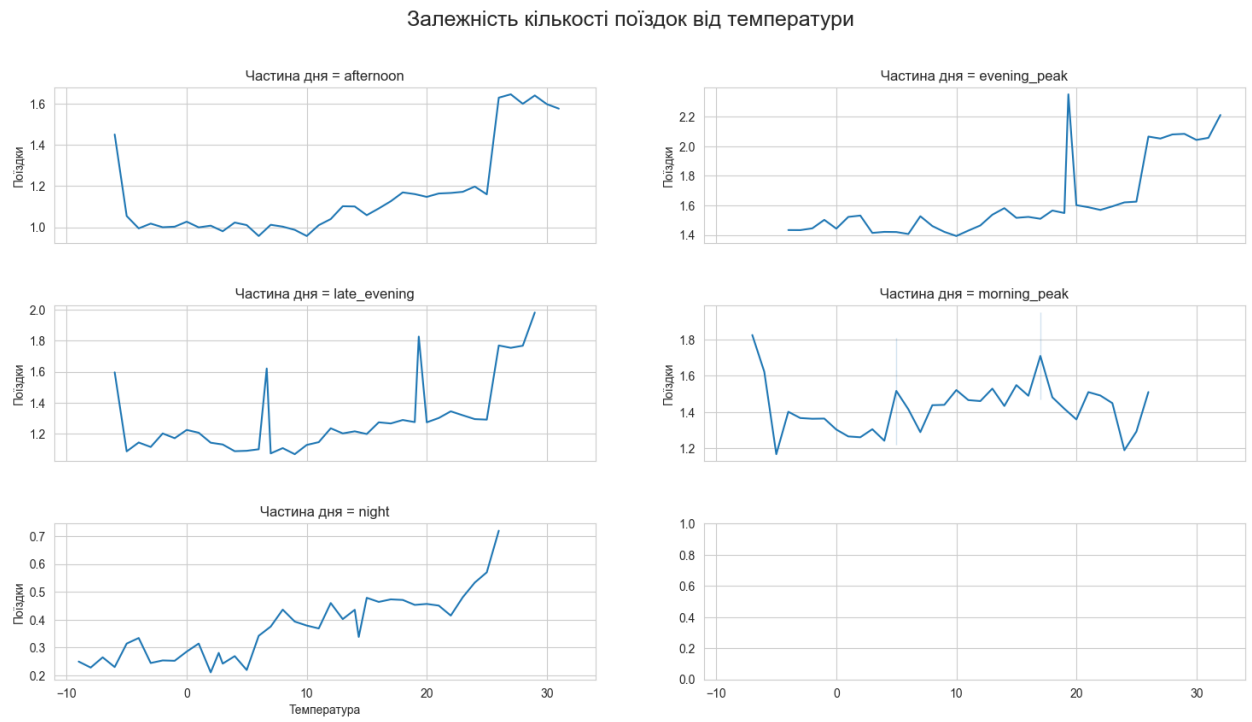


Рисунок 2.9 — Залежність кількості поїздок від температури.

З побудованих залежностей можна зробити висновок, що попит на послуги таксі, якщо нівелювати вплив коливань в залежності від години доби, різко збільшується при пікових температурах, як високих, так й низьких. Введемо показник extreme_temp, що прийматиме значення True для температур нижче мінус 3 та вище +26.

Додатково дослідимо середньодобову температуру по тижнях та місяцях. Результати агрегації даних показані на рис. 2.10.

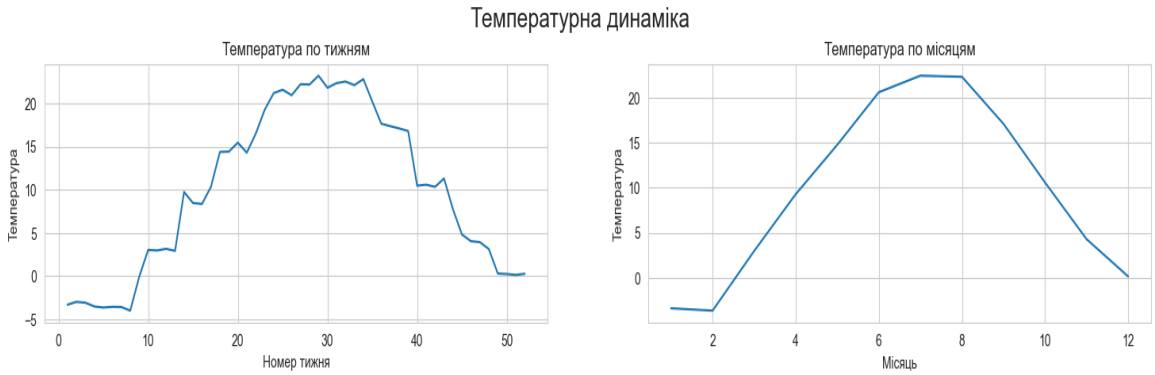


Рисунок 2.10 — Температурна динаміка

Спробуємо визначити, чи існують фактори сезонності для явища, що досліджується. Для цього необхідно нівелювати фактори екстремальних температур, часу доби тощо. Дослідимо попит в ненавантажений час (afternoon) за умови, що температура не перевищує +26 та не опускається нижче мінус 5 (цей діапазон було визначено вище). Результати експерименту показані на рис. 2.11.

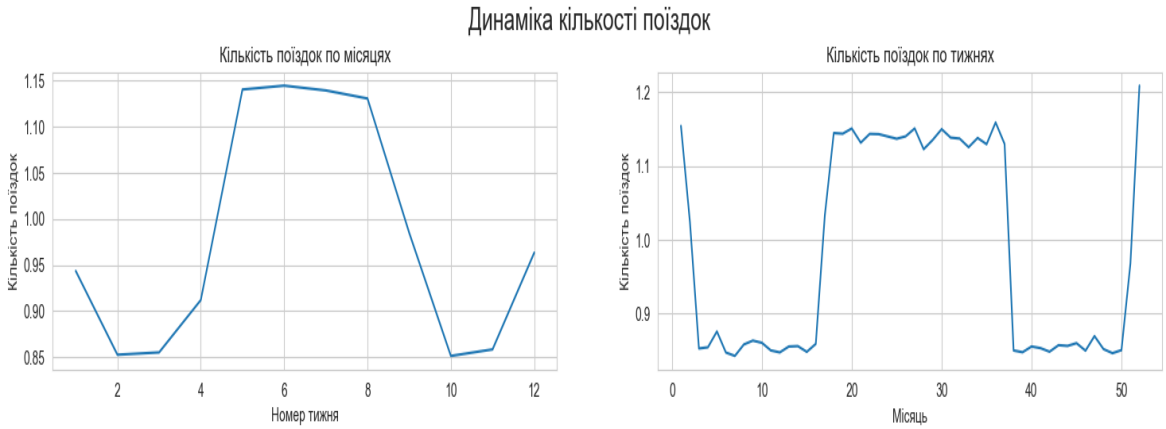


Рисунок 2.11 — Сезонна динаміка кількості поїздок

На графіку можна спостерігати сезонний сплеск, але ще є сплески на початку та кінці року, що може пояснюватись через сплеск на свята. Підвищений попит на свята є короткотривалим, тому треба зменшити крок з тижня до дня й досліджувати дні навколо традиційних свят. Візуалізація динаміки попиту на перевезенні в святкові періоди показана на рис. 2.12. Можна зробити висновок, що до періодів свят, що мають вплив на попит, варто віднести:

— від 1 січня до10 січня;

— від 7 березня до 9 березня;

— від 25 грудня до 26 грудня.

Ці дати відповідають різним святам та особливим подіям, які можуть впливати на мобільність та запит на послуги перевезення. Важливо враховувати ці періоди в аналізі та моделюванні попиту, оскільки вони можуть внести суттєвий внесок у загальний обсяг перевезень.

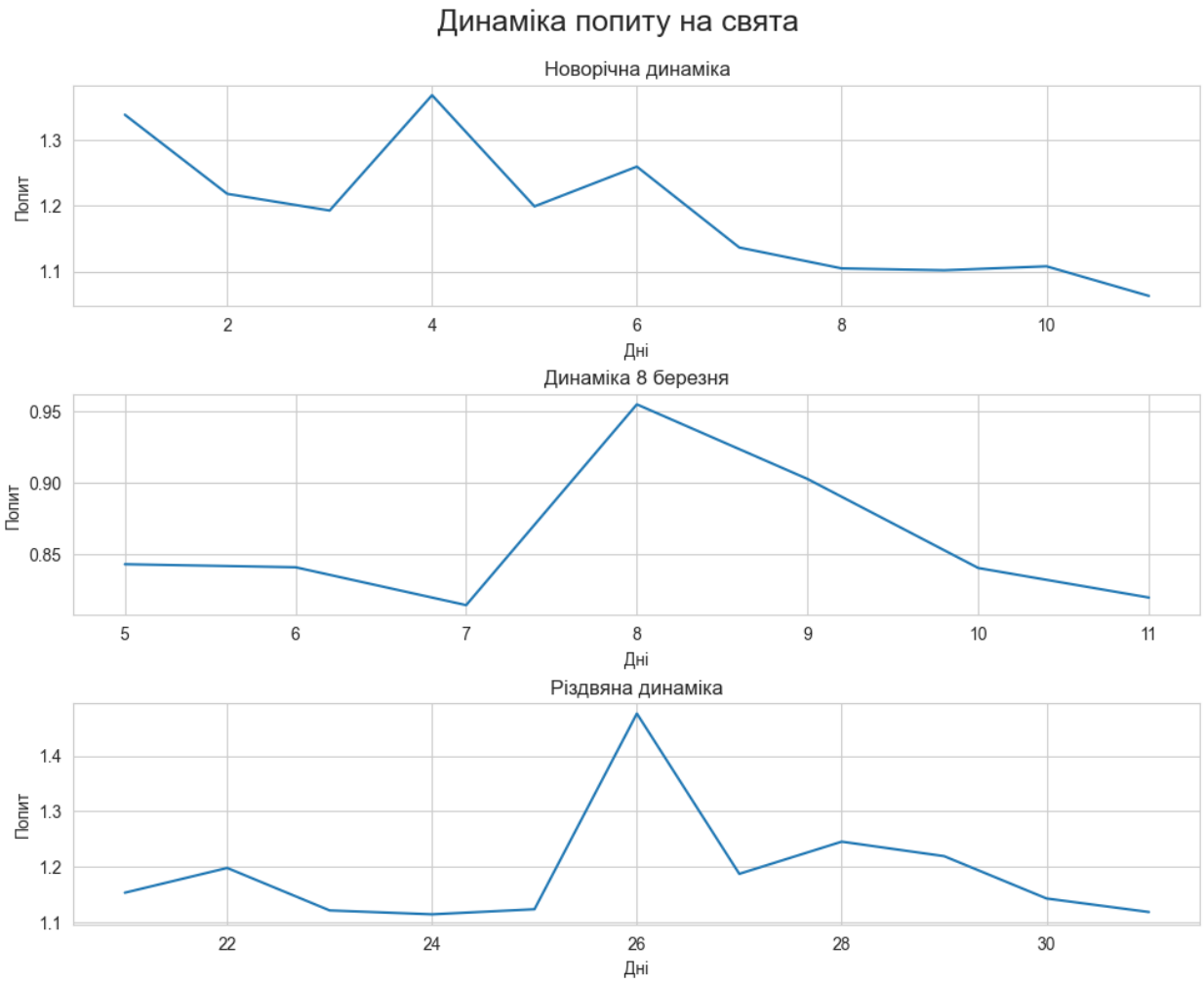


Рисунок 2.12 — Динаміка попиту на свята

Таким чином, регресійна модель, що буде створена на перетворених даних, враховуватиме наступні фактори:

— частину доби;
— день тижня;
— географію: середмістя / віддалені райони;
— опади;
— публічні заходи;
— зависокі або занизькі температури;
— високий (туристичний) сезон;
— свята.

Відповідно до розділу 2.1 було створено дві регресійні моделі – з використанням XGRegressor та штучної нейронної мережі. Результати навчання моделей показані на рис. 2.13 та рис. 2.14 відповідно.

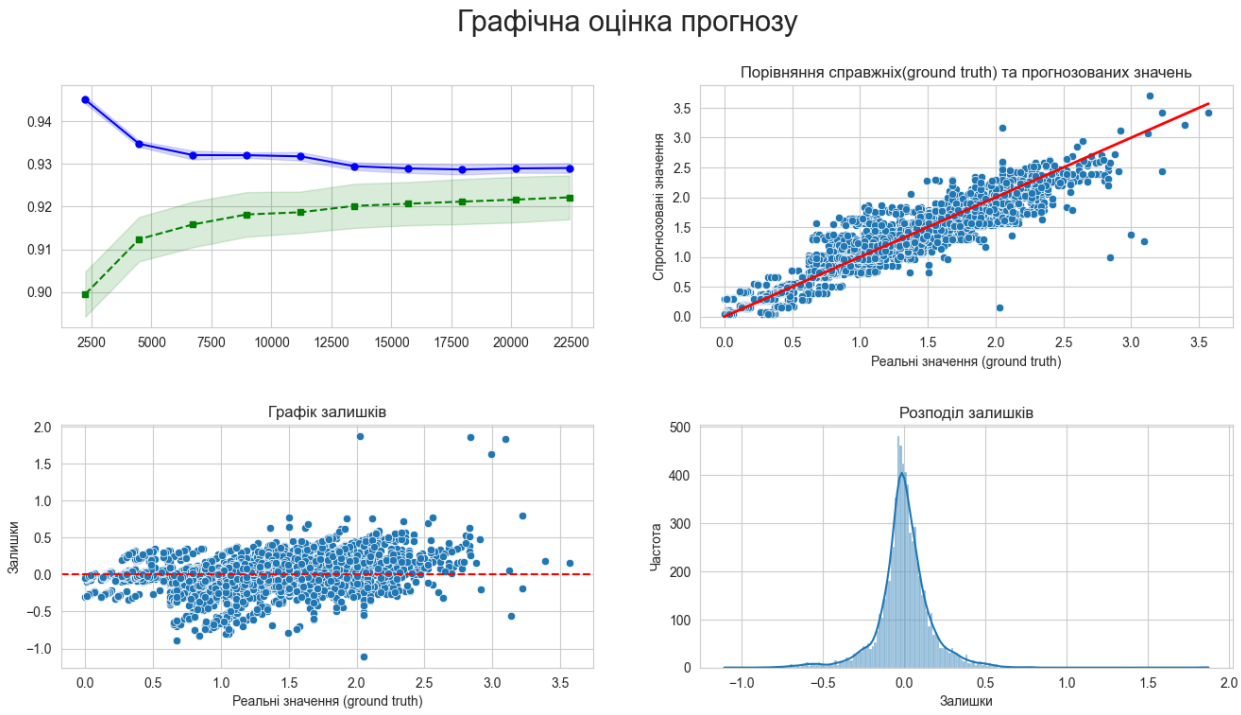


Рисунок 2.13 — Результат навчання моделі XGBRegressor

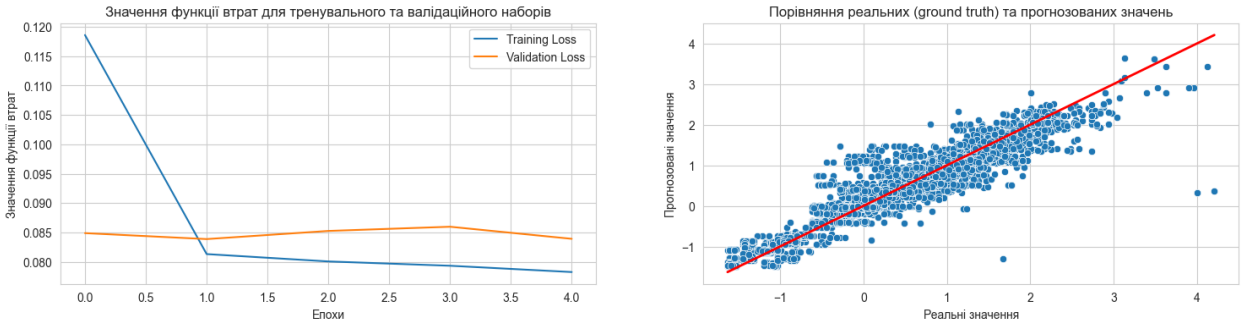


Рисунок 2.14 — Результат навчання штучної нейронної мережі

Оцінки за коефіцієнтом детермінації:

XGBRergressor – 0.923;

багатошаровий перцептрон – 0.92.

2.3 Вдосконалення методу побудови архітектури підсистеми прогнозування попиту

Використання регресійної моделі, особливо такої складної, як XGBRegressor або глибокої нейронної мережі, безпосередньо в мобільному додатку є непрактичним з кількох причин. По-перше, обчислювальна вимогливість таких моделей часто значно перевищує обчислювальні можливості стандартних мобільних пристроїв. Розрахунок прогнозів у реальному часі може призвести до значного зниження продуктивності додатку, споживання батареї та загального користувацького досвіду. Крім того, нейронні мережі та регресійні моделі вимагають значного обсягу пам'яті для зберігання ваг моделі, що може бути проблематичним для пристроїв з обмеженою пам'яттю. Тут мова йде саме про використання навчених попередньо моделей. Окрім того, подібна архітектура унеможливлює оперативне оновлення та донавчання моделей, оскільки на локальному пристрої усі необхідні дані відсутні й доведеться передавати файли досить великого обсягу – або окремо під час підключення пристрою до широкосмугового інтернету, або під час оновлення додатку.

Замість локального розгортання, більш оптимальним рішенням є використання сервера та віддаленої бази даних. Сервер може обробляти важкі обчислення, зберігати та аналізувати великі обсяги даних без впливу на продуктивність мобільного пристрою. Мобільний додаток у цьому випадку виступає як клієнт, що відправляє запити до сервера через API, отримуючи необхідні прогнози або відправляючи дані для аналізу та зберігання в базі даних. Цей підхід не тільки знижує навантаження на мобільний пристрій, але й спрощує процес оновлення та покращення моделі, оскільки всі зміни можуть бути впроваджені на сервері без необхідності оновлення самого мобільного додатку, тобто, необхідно оновлювати лише один екземпляр серверного застосунку замість великої кількості екземплярів клієнтських.

Діаграма розгортання UML [10] фокусується на фізичній структурі системи, включно з обладнанням і програмним забезпеченням.

Зв'язки та мережі ілюструють, як клієнтські пристрої пов'язані із серверами (через інтернет, внутрішні мережі тощо).

Діаграма розгортання для системи, що моделюється та проектується, подана на рис. 2.15.

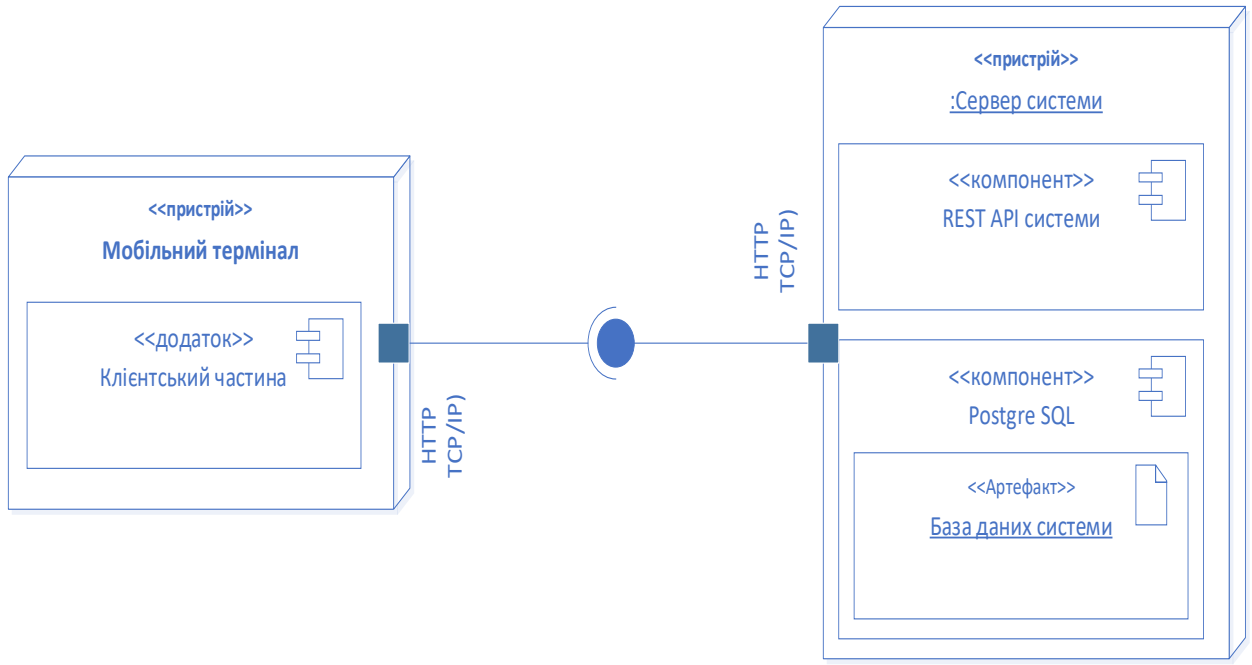


Рисунок 2.15 — Діаграма розгортання системи

Порушення звичайного порядку моделювання й проектування системи, де діаграма розгортання стає далеко не найпершою, пов’язано з тим, що система, що розробляється, складається з двох частин, що наглядно ілюструє рис. 2.15. Одна з частин системи клієнтський додаток, інша — серверна API. Для останньої до того ж слід використати відмінний клієнтського додатку порядок моделювання та проектування, оскільки віддалена API прогнозування не передбачає безпосередньої взаємодії з користувачем, а має лише контракт взаємодії з клієнтським додатком. Окрім REST API на сервері знаходиться база даних, яка призначена для накопичення та агрегації даних про поїздки. Система накопичення та агрегації даних знаходиться поза межами поточного розгляду — вважатимемо, що REST API має необхідні агреговані дані з бази даних провайдера послуг таксі.

2.4 Вимоги до програмного забезпечення, що розробляється

Оскільки система, що розробляється, складається з двох взаємодіючих компонентів, які були показані в попередньому підрозділі, подальші формулювання вимог до системи, моделювання та проектування будуть виконуватись у відповідних підрозділах для кожної підсистеми окремо. Серверна частина системи REST API буде мати можливість використання окремо від мобільного додатку із зрозумілим контрактом надсилання запитів та отримання відповідей.

Сформулюємо функціональні та нефункціональні вимоги до клієнтського мобільного додатку.

Реєстрація та авторизація користувачів:

— можливість створення нового користувацького облікового запису;

— авторизація із застосуванням логіну та паролю;

— відновлення доступу до облікового запису в разі забуття паролю;

Інтерфейс карти:

— відображення карти з можливістю вибору місцезнаходження;

— функція пошуку за адресою;

— відображення поточного місцезнаходження користувача.

Замовлення поїздки:

— вибір початкової та кінцевої точок поїздки, початкова точка є поточним місцезнаходженням, може бути обрана на карті;

— відображення приблизної вартості та часу поїздки.

Прогнозування попиту та тарифів:

— можливість перегляду прогнозу зміни тарифу на таксі на наступні 48 годин;

— вибір оптимального часу для поїздки на основі прогнозу.

— можливість створення замовлення таксі заздалегідь, виходячи з прогнозу тарифів та вибір дати та часу для відкладеного замовлення.

Продуктивність:

— висока швидкість роботи додатку;

— мінімальна затримка при завантаженні карт та обробці даних.

Надійність:

— забезпечення стабільної роботи додатку;

— автоматичне відновлення після помилок.

Безпека:

— захист користувацьких даних та інформації про транзакції;

— використання шифрування для передачі та зберігання даних.

Сумісність:

— підтримка різних версій Android;

— адаптація до різних розмірів екрану та роздільної здатності.

Інтерфейс користувача:

— інтуїтивно зрозумілий і простий у використанні інтерфейс.

Сформулюємо тепер вимоги до REST API, що генерує прогнози попиту на таксі у відповідь на запит, що містить уточнюючу інформацію.

Обробка запиті:

— прийом та обробка JSON-запитів із даними: початкова та кінцева дата, інтервал, події, інформація про центр міста;

— перевірка валідності та формату вхідних даних.

Інтеграція з БД:

— збереження та вибірка даних з Postgre SQL бази даних;

— ефективне управління підключеннями до бази даних.

Взаємодія з моделлю прогнозування:

— інтеграція з XGBRegressor для генерації прогнозів попиту;

— оптимізація викликів моделі для забезпечення швидкого відгуку.

Запити до сторонніх сервісів:

— взаємодія з веб-сервісами для отримання погодних даних;

— обробка та інтеграція погодних даних у прогнозування.

Формування відповідей:

— генерація погодинних прогнозів попиту;

— відправлення даних у форматі JSON у відповідь на запити.

Мова програмування та фреймворк:

— реалізація REST API на Python, в основі фреймворк FastAPI.

Безпека:

— імплементація механізмів аутентифікації та авторизації;

— захист від загальних видів атак (наприклад, SQL-ін'єкції, XSS).

Продуктивність і масштабованість:

— оптимізація продуктивності для швидкої обробки великої кількості запитів;

— можливість масштабування для обробки зростаючого обсягу даних та запитів;

— використання контейнеризації Docker;

— забезпечення стабільної та ефективної роботи в контейнеризованому середовищі.

2.5 Моделювання програмного забезпечення

Розглянемо основні кроки моделювання системи, проілюструвавши їх діаграмами в нотації UML 2.0 [10].

Контекстна діаграма — інструмент візуалізації, який використовується в системному аналізі та розробці програмного забезпечення для виявлення взаємодії системи із зовнішнім світом [10]. Вона показує систему як єдине ціле і контекст, у якому вона функціонує, включно із зовнішніми сутностями (як-от користувачі, зовнішні системи або процеси), які взаємодіють із нею. Контекстна діаграма є важливим інструментом в системному аналізі та розробці програмного забезпечення. Використовуючи цей метод візуалізації, ми можемо надати зрозуміле уявлення про те, як система взаємодіє з навколишнім середовищем. Зокрема, контекстна діаграма презентує систему як цілісний об'єкт, враховуючи при цьому всі складові та їх взаємодію в контексті зовнішнього оточення. На контекстній діаграмі чітко видно, як система взаємодіє із своїм оточенням, і це може включати в себе обмін інформацією, передачу

даних чи виклик зовнішніх подій. Контекстна діаграма системи, що розробляється, показана на рис. 2.16.

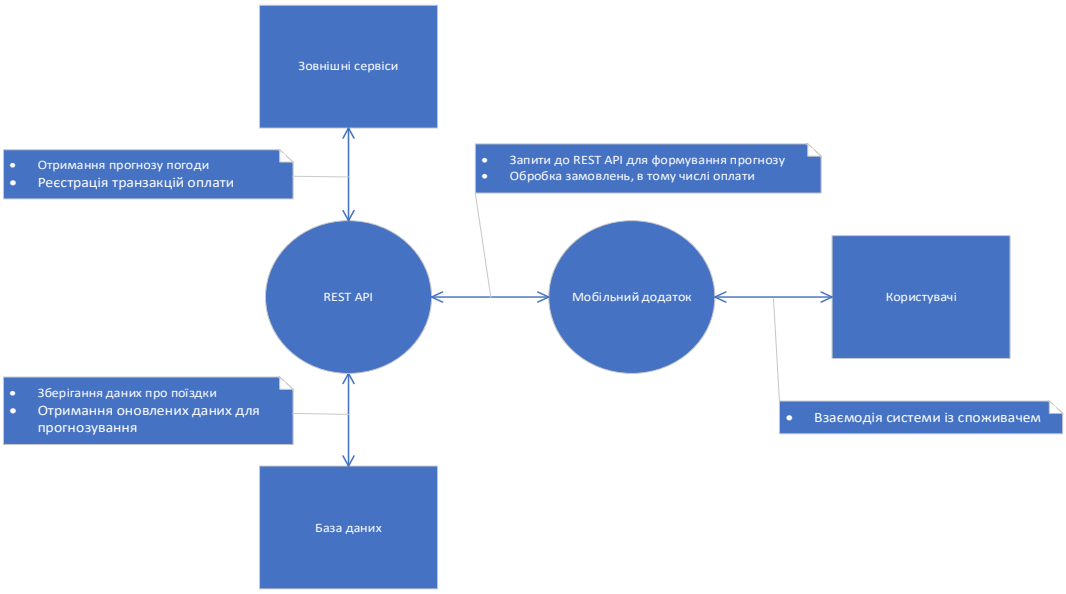


Рисунок 2.16 — Контекстна діаграма системи

На контекстній діаграмі нашої системи мобільного додатку для замовлення таксі в центрі знаходиться сервер, який відіграє ключову роль у всій архітектурі. Він забезпечує обробку запитів, які надходять від мобільного додатку, використовуваного користувачами для реєстрації, авторизації, планування поїздок та перегляду прогнозів попиту. Сервер також зв'язується з зовнішніми сервісами, отримуючи погодні дані для прогнозування попиту та обробляючи платіжні транзакції. Важливою частиною системи є база даних, яка використовується сервером для зберігання інформації про користувачів, історію поїздок, замовлення та інші необхідні дані для функціонування системи. Важливо відзначити, що сервер не лише виконує базові операції, такі як реєстрація, авторизація та обробка замовлень, але й активно взаємодіє з зовнішніми сервісами. Наприклад, використання погодних даних для прогнозування попиту та обробка платіжних транзакцій.

Мобільний додаток є основним інтерфейсом для користувачів, де вони можуть взаємодіяти з системою, виконуючи різноманітні дії, включаючи

замовлення таксі та перегляд графіків попиту. Взаємодія користувачів з мобільним додатком передбачає надсилання запитів на сервер, який в свою чергу обробляє ці запити, обмінюючись інформацією з базою даних та зовнішніми сервісами. Однією з ключових функцій мобільних додатків для замовлення таксі є можливість користувача вислати запит на послугу, вказавши місце призначення та інші необхідні параметри. Така структура забезпечує гнучку та ефективну взаємодію між різними компонентами системи, дозволяючи користувачам легко планувати поїздки, а системі — ефективно обробляти дані та забезпечувати високу якість послуг.

Сценарій використання (Use Case) — опис серії дій, що виконуються системою, які призводять до корисного результату для конкретного користувача або актора [10]. Він фокусується на взаємодії між користувачем і системою та описує, як користувач досягає конкретної мети за допомогою системи. Сценарії використання часто використовують для визначення функціональних вимог системи. Ці сценарії є важливим інструментом при визначенні функціональних вимог до системи. Вони дозволяють розкрити потреби та очікування користувачів, визначити ключові функції системи та визначити, як ці функції повинні взаємодіяти для досягнення конкретних цілей.

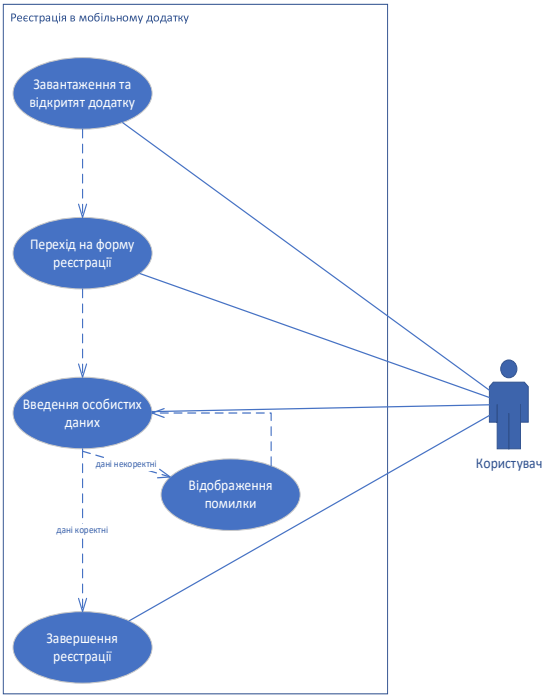


Рисунок 2.17 — Сценарій використання «Реєстрація»

Діаграма сценаріїв використання (Use Case Diagram) показує, які актори взаємодіють з якими сценаріями використання і як сценарії можуть бути пов'язані один з одним. Коротко опишемо основні сценарії використання системи та проілюструємо їх відповідними діаграмами.

Початок взаємодії з додатком користувача починається із реєстрації в додатку, що відображена діаграмою на рис. 2.17.

Далі розглядаємо замовлення поїздки в поточний момент. Сценарій замовлення поїздки є ключовим використанням додатку, і його графічне представлення дозволяє зрозуміти, як користувач може взаємодіяти з системою для отримання необхідних послуг. Розширення сценаріїв, такі як «Введення початкової адреси» та «Вибір адреси на карті», враховують різні ситуації та потреби користувачів. Це по статистиці використання додатків є основним сценарієм. Графічно він представлений на рис. 2.18. В сценарії існує два розширення «Введення початкової адреси» та «Вибір адреси на карті». Це обумовлено тим, що в базовому сценарії використання користувач замовляє поїздку з поточної адреси (що визначена за геопозицією) до деякої адреси, що введена у полі «Кінцева адреса». Але в певних ситуаціях, наприклад, для замовлення перевезення для родича, треба ввести в ручну початкову адресу, а у випадках якогось «складного» кінцевого пункту або кінцевого пункту, що не точно визначається адресою, користувач віддає перевагу вибору кінцевої точки на карті. Наприклад, такий сценарій обирають при поїздці до гаражу, автомобільної стоянки чи ринку замість жилого будинку. Діаграма сценаріїв використання ефективно відображає ключові етапи та можливості взаємодії користувачів з додатком, спрощуючи розуміння функціоналу та ідентифікацію можливих покращень. В цілому, діаграма сценаріїв використання служить важливим інструментом для розуміння та визначення ключових етапів взаємодії користувача з системою, що допомагає вдосконалити дизайн та функціонал додатку.

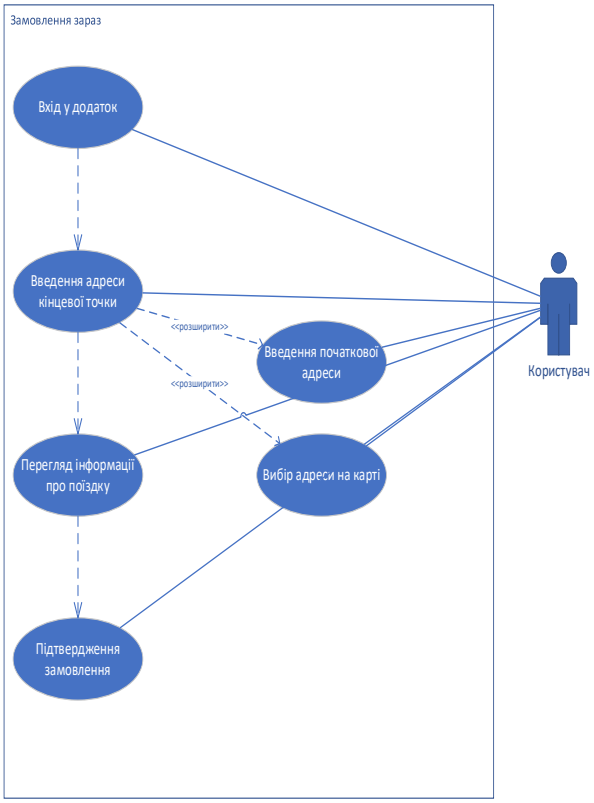


Рисунок 2.18 — Сценарій використання «Замовлення зараз»

Сценарій, що включає використання прогнозу попиту, розглядаємо в діаграмі на рис. 2.19.

Ключовими особливостями сценарію є те, що користувач переглядає прогноз динаміки попиту та обирає час відкладеного замовлення, зіставивши оптимальний час замовлення із своїми потребами – можливістю зачекати або перенести поїздку. Розширенням кроку перегляд прогнозу є зміна періоду прогнозу, оскільки початково є сенс надавати прогноз на 3-6-12 годин наперед, але в деяких випадках користувач цікавитиметься й поїздками наступного дня. Цей сценарій надає користувачеві більшу гнучкість та контроль над процесом замовлення, що підвищує його задоволеність від використання додатку та взаємодії з сервісом. Такі функціональні можливості сприяють покращенню загального користувацького досвіду та конкурентоспроможності сервісу на ринку.

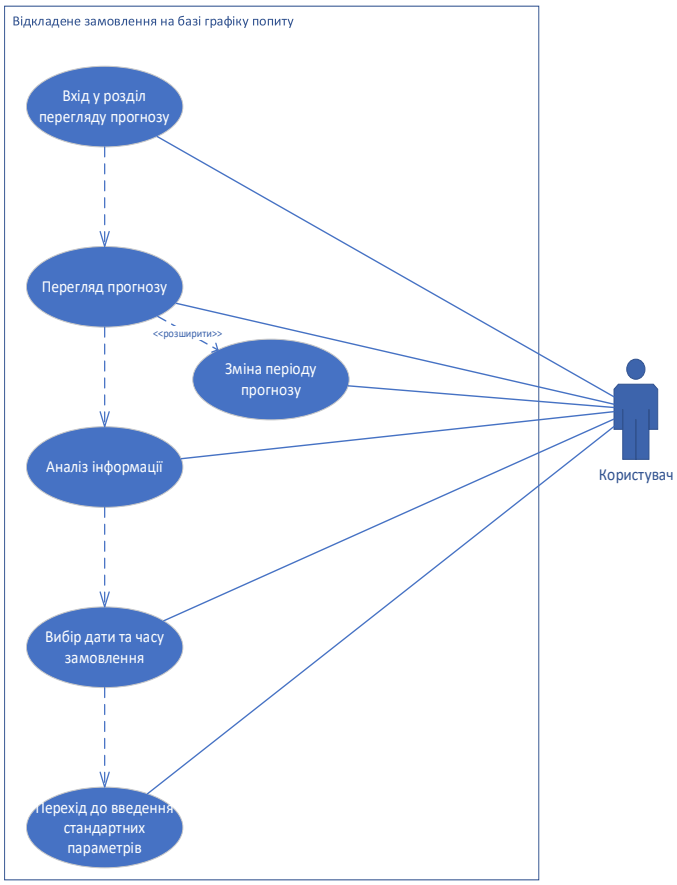


Рисунок 2.19 — Сценарій використання «Замовлення з врахуванням попиту»

Ключовим елементом сценарію є взаємодія користувача з прогнозом динаміки попиту на послуги та можливістю вибору часу відкладеного замовлення.

Після вибору часу відкладеного замовлення користувач переходить до заповнення інших обов’язкових даних (реквізитів) замовлення, що співпадає із попереднім розглянутим сценарієм «Замовлення зараз».

Діаграми діяльності (Activity Diagrams) в UML являють собою графічні уявлення робочих процесів і дій у межах системи [10].

Розглянемо діаграми активностей, що відповідають трьом основним сценаріям, що були проілюстровані. Діаграма активностей для сценарію реєстрації показана на рис. 2.20. До особистої інформація відносимо адресу електронної пошти, телефон та пароль, що створює користувач для подальшого входу в додаток. Валідуємо форму шляхом перевірки коректності форматів

реквізитів за допомогою шаблонів та відсилаємо дані на сервер. Сервер в свою чергу ініціює задачу надсилання підтвердження на електронну пошту.

Рисунок 2.20 — Діаграма діяльності «Реєстрація користувача»

Наступною діяльністю, що розглядається, є процес замовлення поїздки на поточний час. Діаграма діяльності представлена на рис. 2.21. Оскільки система, що будується з метою демонстрації можливостей та корисності прогнозування попиту, не має функціоналу визначення водія, призначення водія та оцінки часу прибуття замовленої автівки, для цих кроків використовуватимемо при реалізації програмного забезпечення заглушки, тобто імітуватимемо роботи цих алгоритмів за допомогою додавання фіксованого часу до поточного й відображення випадково одного з введених до системи водіїв.

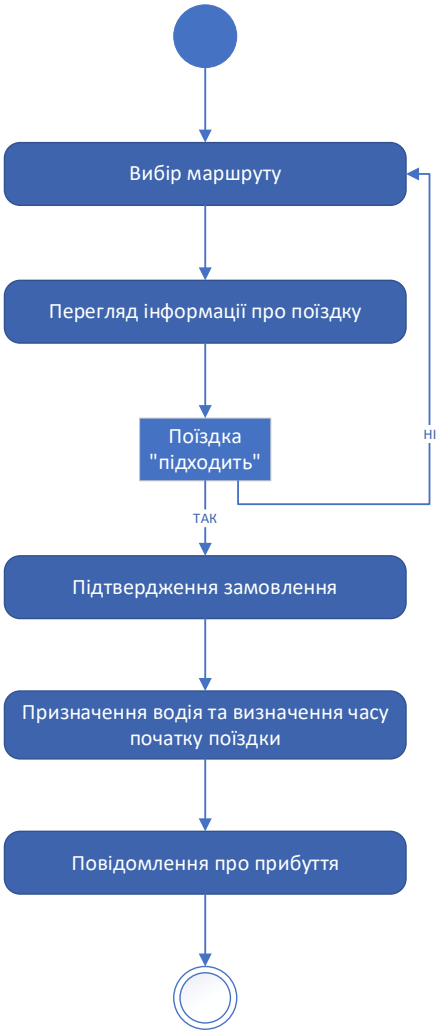


Рисунок 2.21 — Діаграма діяльності «Замовлення зараз»

Діаграма діяльності, що відповідає створенню відкладеного замовлення, показана на рис. 2.22. Діаграма відображає основні кроки та можливі альтернативні послідовності діяльності. У випадку невідповідності інтервалу прогнозування за замовчанням користувач змінює інтервал прогнозу й відповідним чином змінюється інформація про прогноз. Після перегляду актуального для потреб користувача прогнозу він приймає рішення відносно доцільності створення відкладеного замовлення. Прийнявши про це позитивне рішення визначатиме параметри замовлення, а в протилежному випадку просто перейде на головний екран додатку без уточнення та підтвердження відкладеного замовлення.

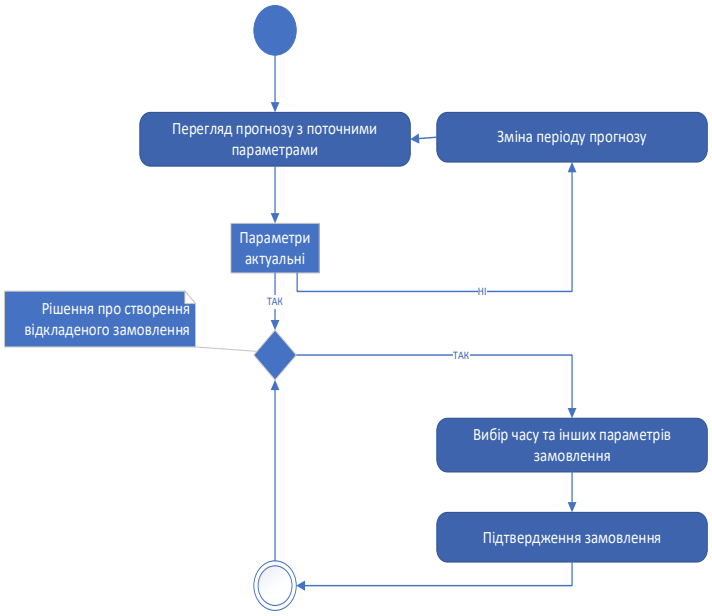


Рисунок 2.22 — Діаграма діяльності «Відкладене замовлення»

Також розглядаючи діаграми діяльності, створимо діаграму діяльності для серверної частини додатку «Створення прогнозу». Відповідна діаграма відображена на рис. 2.23.

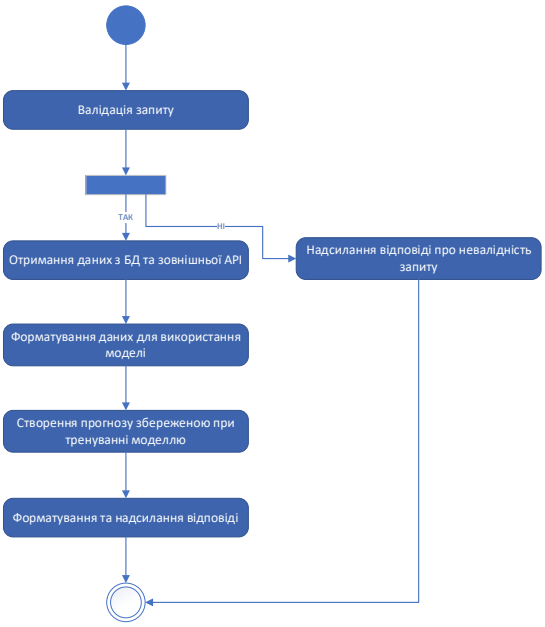


Рисунок 2.23 — Діаграма діяльності «Створення прогнозу

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ

3.1 Вибір інструментів реалізації

Звичайно, в контекст вибору інструментів реалізації також доцільно обґрунтовувати окремо реалізацію мобільного та серверного додатків.

Для створення API використано мову Python [11-13]. Python є популярним вибором для розробки REST API за рахунок:

— простоти та читабельності коду;
— великої спільноти та екосистеми;
— достатньої продуктивності;
— підтримки сторонніх бібліотек та інструментів;
— крос-платформеності;
— достатнього рівня забезпечення безпеки;
— розширюваності та масштабованості.

Для створення API використовується фреймворк FastAPI [14]. Він розроблений з акцентом на швидкість, продуктивність і автоматичну генерацію інтерактивної документації, що робить його чудовим вибором для розробки високопродуктивних і документованих API. Ось деякі ключові особливості та переваги FastAPI:

— швидке розгортання;
— асинхронність;
— автоматична валідація запитів;
— автоматична генерація документації;
— інтеграція з популярними базами даних;
— висока продуктивність.

Модуль, що використовує математичні моделі та машинне навчання також побудовано на Python, оскільки REST API з використанням Fast API фактично являє собою обгортку для використання математичних моделей.

Як мобільна платформа для створення MVP програми буде використаний Android. З цієї причини в якості мови реалізації додатку використано Java.

Java — об'єктно-орієнтована мова програмування загального призначення, заснована на класах, розроблена з меншими залежностями реалізації. Широко використовується для розробки програм у ноутбуках, центрах обробки даних, ігрових консолях, наукових суперкомп'ютерах, смартфонах, інтернеті речей, тощо [15, 16].

Важливою причиною величезної популярності Java є незалежність від платформи, так звана мультиплатформенна підтримка. Програми Java можуть виконуватись на різних машинах, якщо існує JRE (Java Runtime Environment).

Java є вибором багатьох розробників, тому що поєднує інновації зі стабільністю, і що дуже важливо, має повну або майже повну зворотну сумісність: код, написаний на перших версіях Java, як і раніше, буде запускатися на сучасних JVM (Java Virtual Machine).

Оскільки в об'єктах відсутні посилання на зовнішні дані, Java може похвалитися надійністю.

Особливістю Java є Багатопотоковість. Можливості багатопотоковості вбудовані прямо в мову Java. Це означає, що можна створювати інтерактивні та чуйні програми з кількома паралельними потоками активності.

Висока продуктивність. Архітектура Java призначена для зменшення витрат накладних під час використання. Концепція багатопоточності Java також збільшує швидкість виконання програм Java

Незалежність від платформи. Програми Java компілюються у формат байтового коду, який залежить від архітектури машини, але може бути легко переведений на конкретну машину з допомогою віртуальної машини Java (JVM). Це значна перевага при розробці аплетів або програм, що завантажуються з Інтернету та необхідні для роботи в різних системах.

Повна реалізація патернів ООП. Заснований на досвіді C++, Java розширює функціональні змогу повної реалізації патернів об'єктно — орієнтованого програмування.

Надійність та безпека. Код Java є надійним, оскільки проходити перевірку на етапі компіляції та складання програми до моменту запуску на віртуальній

машині. Він також надає концепт обробки винятків виявлення логічних помилок, які можуть призвести до збою системи. Безпека також у тому, що з конвертації коду між машинами відбувається перевірка коду до виконання.

Використання Java:

— розробка мобільних додатків для Android;

— створення корпоративного програмного забезпечення;

— значна частина промисловості програмного забезпечення у різних проявах;

— серверні технології Apache, JBoss, GlassFish.

В якості середовища розробки в роботі використовується Android — studio. Android-studio є офіційною IDE (скорочено англійською мовою, Integrated Development Environment) для платформи Android абсолютно безкоштовно через ліцензію Apache License 2.0 та кросплатформенність (Microsoft Windows, macOS та GNU/Linux).

Android-studio засновано на програмному забезпеченні IntelliJ IDEA JetBrains і було випущено як заміну Eclipse. Android studio є офіційною IDE для розробки програм Android.

Цей програмний продукт пропонує повний інструмент для розробки та налагодження програм для мобільної операційної системи Google.

З його допомогою можна виконувати редагування коду, налагодження, використовувати інструменти підвищення продуктивності. Продукт має гнучку систему компіляції та миттєве створення та розгортання, що дозволяє зосередитись саме на створенні програми.

Android-studio включає шаблони проектів та коду, які спрощують додавання усталених шаблонів, наприклад, бічна панель навігації та перегляд сторінки.

Серед основних характеристик продукту можна виділити:

— функції інтеграції ProGuard та підписи додатків;

— рендерінг у реальному часі;

— консоль розробника: поради щодо оптимізації, допомогу з перекладом, статистика використання;

— підтримка збирання на основі Gradle — система автоматичного складання, побудована на принципах Apache Ant та Apache Maven, але надає DSL мовами Groovy та Kotlin замість традиційної XML-подібної форми представлення конфігурації проекту;

— рефакторинг для Android та швидкі виправлення;

— багатий редактор макетів, який дозволяє користувачам перетягувати компоненти інтерфейсу користувача;

— інструменти Lint для визначення продуктивності, зручності використання, сумісності версій та інших проблем;

— шаблони для створення спільних макетів Android та інших компонентів;

— підтримка програмування програм для Android Wear;

— інтегрована підтримка Google Cloud Platform, що забезпечує інтеграцію з Google Cloud Messaging та App Engine;

— віртуальний пристрій Android, що використовується для запуску та тестування програм.

3.2 Розгортання програмного комплексу

На додаток до діаграми розгортання, що була розглянута в попередньому розділі через необхідність пояснення архітектури системи перед її моделюванням, необхідно обрати спосіб розгортання системи та пояснити фізичне розташування компонентів програмного комплексу, що розгортається.

Використання контейнеризації Docker [17] та розміщення REST API та бази даних PostgreSQL [18, 19] в двох окремих контейнерах має кілька значних переваг, як для тестування на локальній машині, так й у разі промислої експлуатації на віддаленому сервері чи на двох серверах. Перше та найголовніше, це підвищення надійності, оскільки окремі контейнери для API та бази даних забезпечують ізоляцію процесів, що зменшує ризик взаємного впливу

помилок або збоїв в одній системі на іншу. Таким чином, якщо виникає проблема з REST API, це не безпосередньо впливає на базу даних і навпаки.

Друга перевага полягає в гнучкості та масштабованості. Використання контейнерів дозволяє легко масштабувати кожен компонент незалежно. Наприклад, можна збільшити кількість контейнерів для обробки запитів REST API під час пікових навантажень, не торкаючись бази даних. Це забезпечує оптимальне використання ресурсів та підтримку високої продуктивності системи.

Третя перевага — спрощення процесу розгортання та тестування. Розміщуючи REST API та базу даних у окремих контейнерах, можна легко відтворити продуктивне середовище на локальній машині для тестування. Це забезпечує високу точність тестування, оскільки середовище розробника максимально наближене до середовища виробництва. Крім того, з допомогою технологій контейнеризації, таких як Docker, процес налаштування середовища стає більш стандартизованим та автоматизованим.

Четверта перевага — підвищення безпеки. Розділення REST API та бази даних на різні контейнери дозволяє більш точно налаштувати мережеві політики та контроль доступу для кожного компоненту. Це знижує ризик несанкціонованого доступу та можливість атак, оскільки для компрометації всієї системи зловмисникам потрібно проникнути в кілька ізольованих середовищ. Крім того, використання окремих контейнерів полегшує відстеження та аналіз безпекових подій.

Отже, розгортання серверної (віддаленої) частини програмного комплексу виконуємо за допомогою Docker, для збірки контейнеру з REST API — Docker build, для оркестрації контейнерів — Docker compose.

Користувацький додаток встановлюється на локальному пристрої користувача з операційною системою Android. Під час користувацького тестування відбувається встановлення додатку з файлу API, при промисловій експлуатації додаток розміщуються в Google Play.

Для розміщення додатку в Google Play, перш за все, потрібно створити обліковий запис розробника в Google Play Console. Це вимагає заплатити одноразовий внесок та надати необхідну інформацію про себе чи свою компанію. Після створення облікового запису, наступним кроком є підготовка додатку до публікації. Це включає налаштування деталей додатку, таких як назва, опис, зображення та іконки, а також визначення вікових обмежень та конфігурацій приватності.

Далі, необхідно завантажити APK або App Bundle файл додатку. Перед завантаженням слід переконатися, що додаток відповідає всім вимогам Google Play, включаючи політики змісту та технічні вимоги. Після завантаження файлу, треба налаштувати ціноутворення та доступність, вказуючи, чи буде додаток безкоштовним або платним, та в яких країнах він буде доступний.

Нарешті, перед публікацією додатку, важливо ретельно протестувати його в Google Play Console. Це дозволить виявити та виправити можливі помилки, а також забезпечити кращий досвід користувачів. Після завершення всіх цих кроків, можна надіслати додаток на розгляд. Google перевіряє додатки перед публікацією, щоб переконатися, що вони відповідають всім вимогам. Якщо додаток схвалений, він стає доступним в Google Play, де його можуть завантажити користувачі.

3.3 Діаграма компонентів

Розглянемо діаграму компонентів [10] для backendy системи, що показана на рис. 3.1. REST API складається з трьох основних модулів, що мають відокремлені функції та взаємодіють за визначеним контрактом. Це є виконанням принципу інкапсуляції об'єктно — орієнтованого підходу до реалізації програмного засобу.

Модуль main відповідає безпосередньо за функціонування REST API отримання запитів, формування та надсилання відповідей за цими запитами.

Модуль db відповідає за взаємодію з базою даних додатку, у тому числі за шифрування та порівняння паролів користувачів.

Зона відповідальності модулю ml — обробка даних та створення прогнозів попиту, які потім вертає у відгуку модуль main.

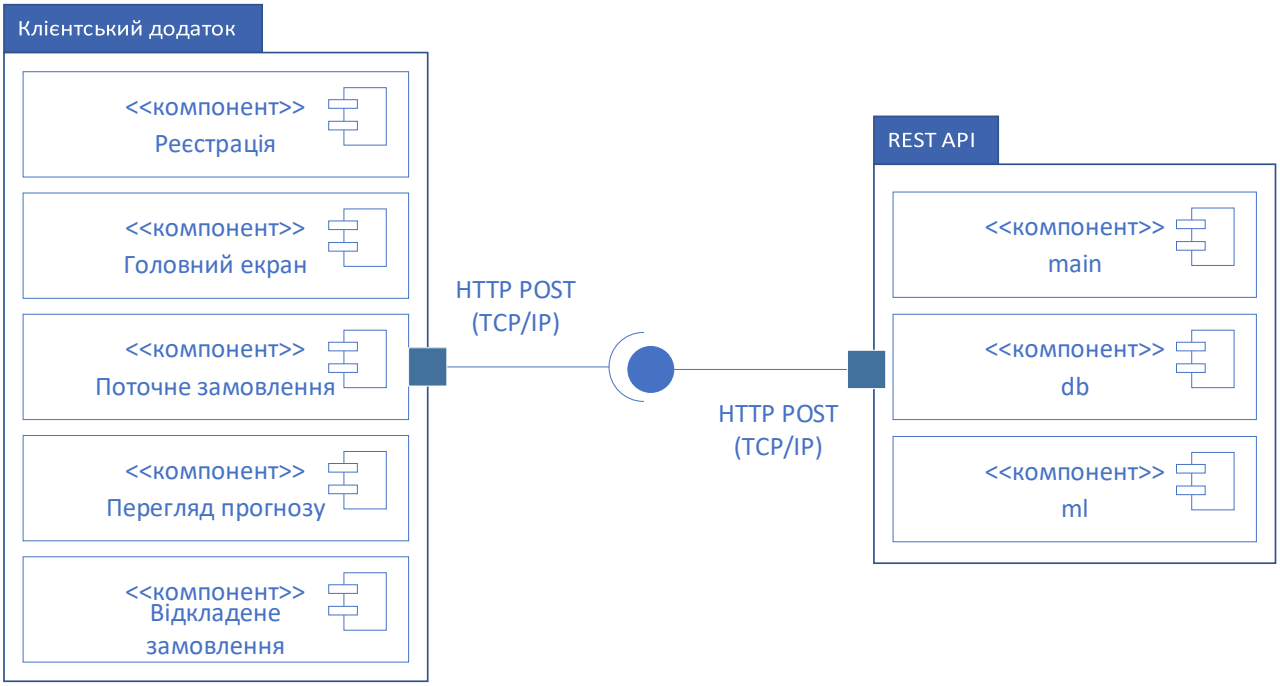


Рисунок 3.1 — Діаграма компонентів

Також на діаграмі компонентів відтворюємо архітектуру клієнтського додатку. Модулі клієнтського додатку відповідають сценаріям використання, що були розглянуті у другому розділі. Сценарій відкладеного замовлення реалізується двома модулями — перегляд прогнозу вартості та створення відкладеного замовлення.

Також створимо діаграму станів мобільного додатку [10]. Діаграму показано на рис. 3.2. Діаграму станів на практиці Android — додатку можна інтерпретувати як діаграму переході між формами (екранами) мобільного додатку. Перехід від одного стану додатку до іншого відбувається в результаті дії користувача або настання певної події. Так перехід від головного екрану з картою до замовлення поїздки або перегляду графіку попиту відбувається за вибором користувача, а перехід до стану «Настання часу відкладеної поїздки» відбувається відповідно до часу, тобто події «Настання визначеного моменту часу. Оскільки, як було зауважено вище, система не реалізує взаємодії з водіями,

то в практичній реалізації буде відсутній стан «Поїздка в процесі», а перехід до стану «Перегляд інформації про поїздку та водія» відбудеться через фіксований інтервал часу після «Замовлення поїздки» або «Настання часу відкладеної поїздки».

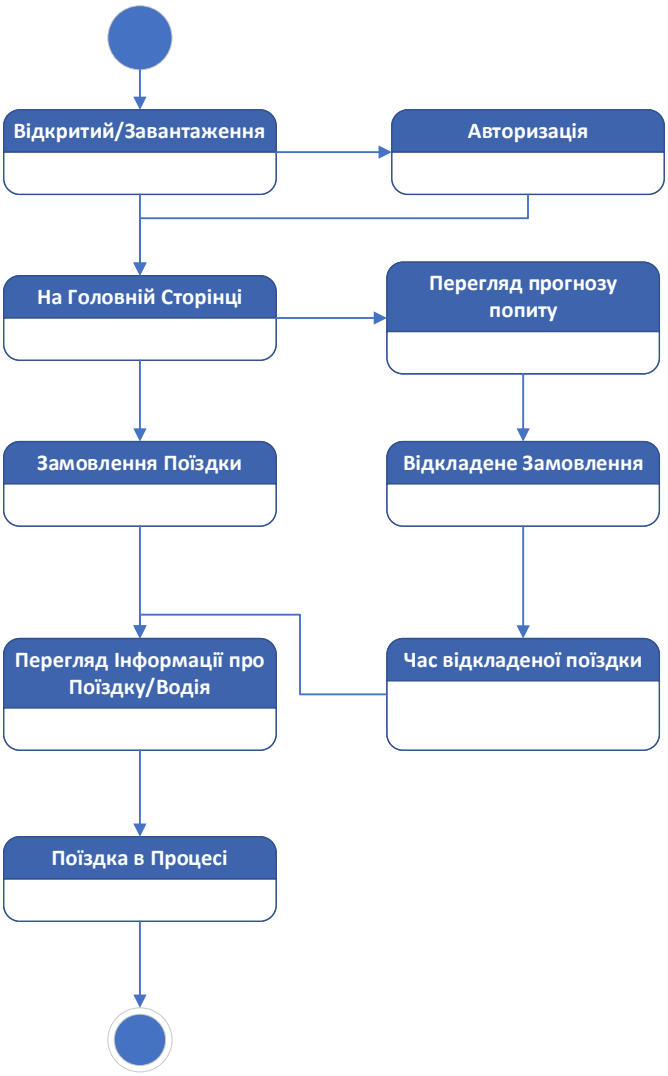


Рисунок 3.2 — Діаграма станів

Для REST API послідовність станів є лінійною та складається з отримання запиту, перевірки коректності запиту, формування та надсилання відповіді. Компоненти системи, що виконують відповідні операції були описані вище.

4 ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ ТА ТЕСТУВАННЯ

4.1 Методика тестування створеного програмного забезпечення

Unit-тестування REST API, написаного на Python [20], є базовим елементом забезпечення якості програмного продукту. Цей процес включає кілька важливих етапів:

— підготовка середовища тестування, а саме першим кроком є налаштування середовища для unit тестування, що включає створення віртуального середовища Python для ізоляції залежностей і налаштування необхідних інструментів тестування — unittest;

— розробка тестових випадків — наступним кроком є розробка конкретних тестових випадків, які перевіряють функціональність окремих компонентів API, що може включати тестування різних HTTP-методів (GET, POST, PUT, DELETE), перевірку відповіді сервера (коди статусу, формати відповідей) та обробку помилок;

— мокування зовнішніх залежностей для unit-тестування важливо ізолювати тести від зовнішніх залежностей, таких як бази даних або зовнішні сервіси, які можна здійснити за допомогою мокування (mocking) техніки створення імітаційних об'єктів, які імітують поведінку справжніх об'єктів;

— виконання тестів та аналіз результатів після написання тестових випадків їх необхідно виконати, щоб перевірити, чи працюють окремі частини API належним чином. У разі виявлення помилок, вони повинні бути виправлені, а тести повторно виконані.

Coverage у контексті unit-тестування відіграє ключову роль, оскільки покриття вказує на відсоток коду, який охоплюється тестами. Високий рівень покриття тестами є важливим, оскільки це забезпечує більшу впевненість у тому, що програмний код працює належним чином і помилки будуть виявлені на ранньому етапі. Максимальне покриття допомагає зменшити ризики у випуску продукту, забезпечує стабільність і надійність програмного забезпечення. Це

особливо важливо в складних системах, де непередбачені взаємодії між компонентами можуть призвести до критичних помилок.

Важливим кроком, що передує промисловій експлуатації системи, є інтеграція з системою безперервної інтеграції (CI): Для автоматизації процесу тестування та забезпечення стабільності коду, unit тести можуть бути інтегровані у систему безперервної інтеграції, яка автоматично виконує тести при кожному оновленні коду.

Unit тестування користувацького Android додатку значно відрізняється від тестування REST API. По-перше, воно включає тестування UI компонентів, к тому числі, перевірку правильності відображення елементів на екрані та їх взаємодії з користувачем. Також необхідно протестувати реакцію додатку на різні вхідні дані та стани, включаючи повороти екрану, переривання зв'язку, тощо. Для цього буде використано Espresso, інструмент який забезпечує більш широкі можливості для тестування, ніж стандартні бібліотеки unit-тестування для REST API. Окрім перелічених сценаріїв тестування, що значно перевищують за кількістю сценарії для REST API, важливою особливістю тестування Android додатку є тестування інтеграції між додатком та системними компонентами Android, що включають сервіси, бродкасти та інтенти.

Тестування проводиться в наступному порядку:

— Написання unit-тестів для індивідуальних елементів інтерфейсу та логіки обробки даних — включає перевірку правильності роботи форм, валідацію введених даних та обробку помилок.

— Основну частину тестування мережевих запитів виконаємо під час тестування REST API з використанням Postman, що буде описано в розгляді контрольного прикладу, більше того при тестуванні Android-додатку важливо переконатись, що додаток правильно обробляє відповіді API, включаючи випадки з помилками.

— Інтеграційне тестування перевіряє взаємодію між різними частинами додатку, включаючи мережеві запити.

— Тести користувацького інтерфейсу — використовуючи інструмент Espresso для автоматизації тестування користувацького інтерфейсу, перевіряємо навігацію між формами та взаємодію з ними.

— Перевірка на різних пристроях і під мережевими умовами — завершальним етапом перевірки, що на відміну від попередніх виконується на реальних пристроях замість емулятора, є тестування на декількох різних пристроях, що мають різні характеристики потужності, розміру екрану та версії ОС Android, більше того доцільне також тестування у різних мережевих умовах, щоб впевнитись в стабільній роботі додатку.

4.2 Розгляд контрольного прикладу.

Спочатку тестуємо створений програмний інтерфейс REST API, оскільки тестування додатку Android вимагає виконання запитів до функціонуючої API. Для цього REST API має бути протестований та розгорнутий в Docker контейнері відповідно до розділу 3.2. Також має функціонувати екземпляр бази даних Post-gre SQL.

В результаті збірки маємо два запущених контейнери, що входять до однієї групи, як показано на рис. 4.2. Перевіряємо коректність запуску контейнеру REST API також використовуючи Docker desktop. REST API працює коректно та готовий приймати запити, як це виглядає на рис. 4.3.

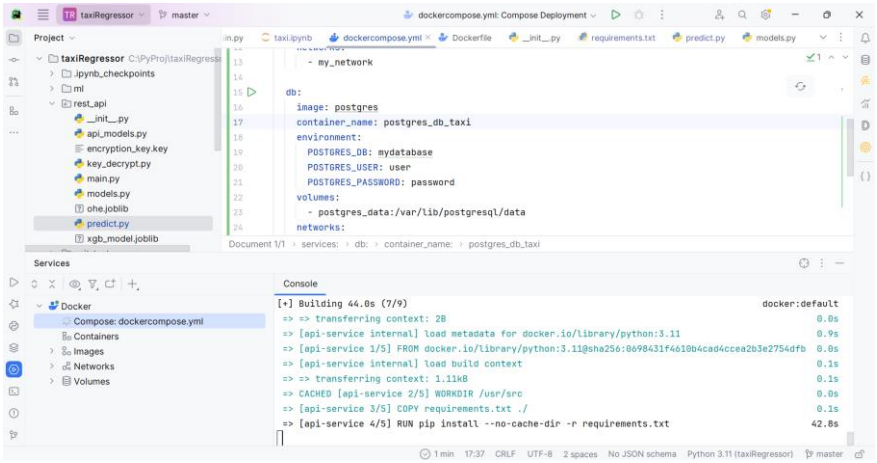


Рисунок 4.1 — Збірка та оркестрація контейнерів REST API

Проводимо збірку та оркестрацію контейнерів, використовуючи Docker build та Docker compose, в результаті чого маємо два запущених контейнери. Скріншот процесу збірки з використанням можливостей інтеграції PyCharm Pro показано на рис. 4.1.

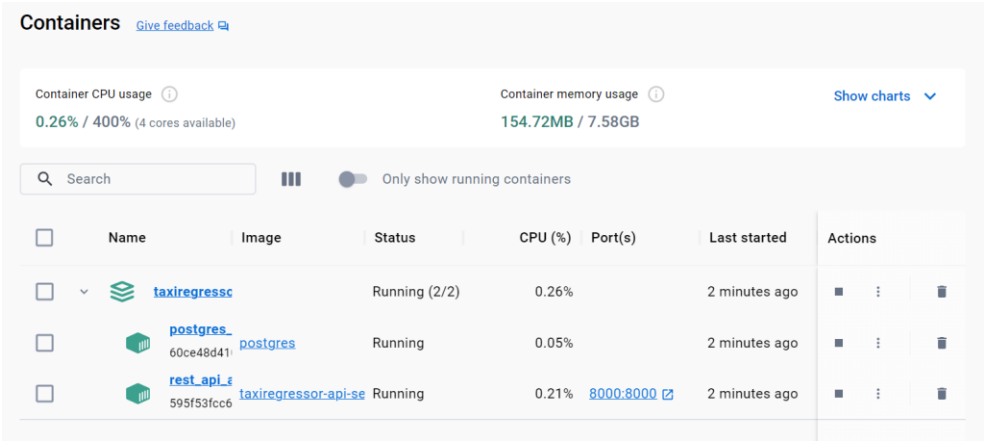


Рисунок 4.2 — Docker desktop

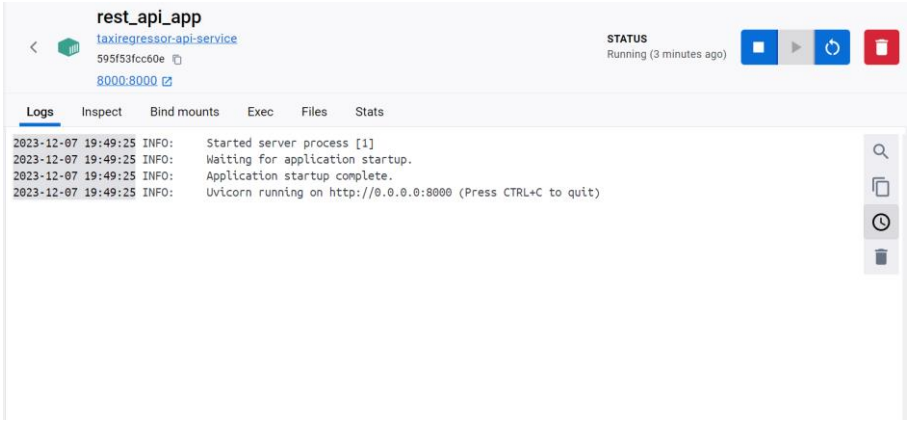


Рисунок 4.3 – Консоль контейнеру REST API

Далі проводимо функціональне тестування REST API з використанням спеціалізованого програмного забезпечення Postman. Postman — потужне програмне забезпечення для розробки API, яке дозволяє розробникам тестувати, використовувати, документувати та моніторити API. Воно надає інтуїтивно зрозумілий графічний інтерфейс для відправки запитів до API, перегляду відповідей та аналізу їхніх даних. Postman підтримує різні методи HTTP-запитів,

включаючи GET, POST, PUT та DELETE, та дозволяє тестувати веб-сервіси, перевіряючи їхню відповідність специфікаціям та вимогам.

Крім того, Postman пропонує можливості для автоматизації тестів та інтеграції з різними середовищами CI/CD, що робить його корисним інструментом для розробників на всіх етапах розробки API. Можливість збереження запитів та їхніх колекцій, а також спільна робота над ними в команді, є важливими перевагами, які роблять Postman популярним вибором серед професіоналів у галузі розробки програмного забезпечення.

Надсилаємо до API перший POST запит для реєстрації користувача. В подальшому необхідно разом із запитом надсилати HTTP Basic Credentials, оскільки створений REST API вимагає аутентифікації користувача.

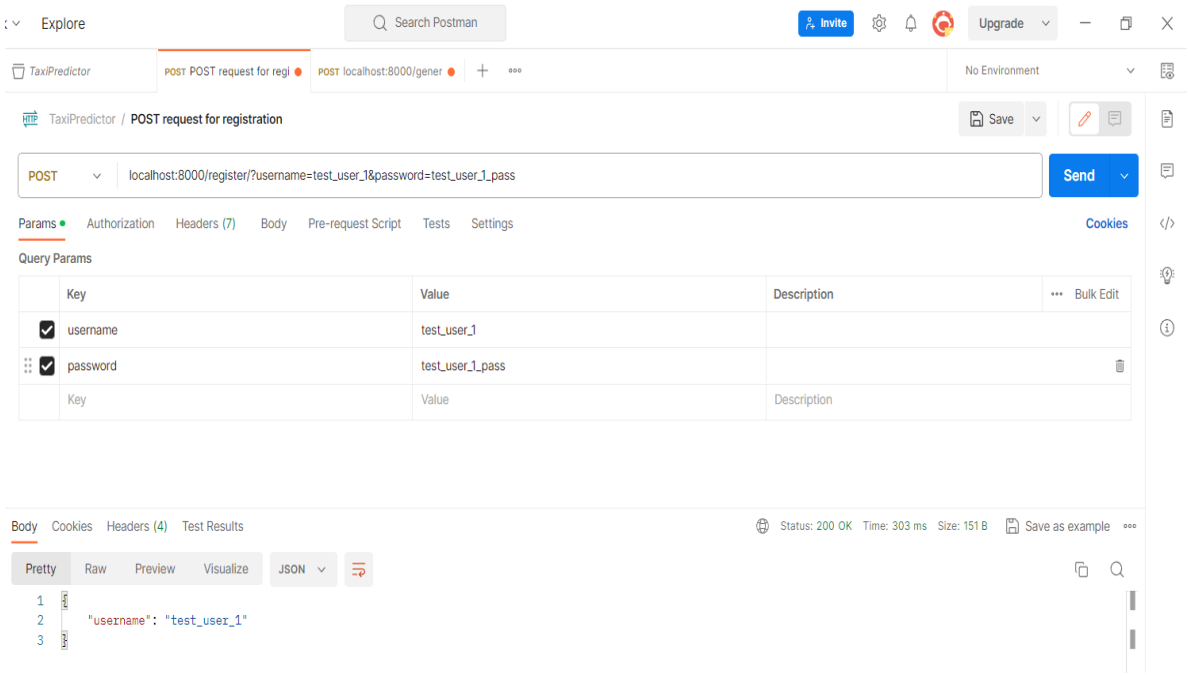


Рисунок 4.4 — Запит реєстрації користувача

В результаті отримуємо відповідь, що свідчить про успіх реєстрації користувача із заданим тестовим паролем.

Тестування основного запиту до REST API, в результаті якого повертається прогноз, показано на рис. 4.5.

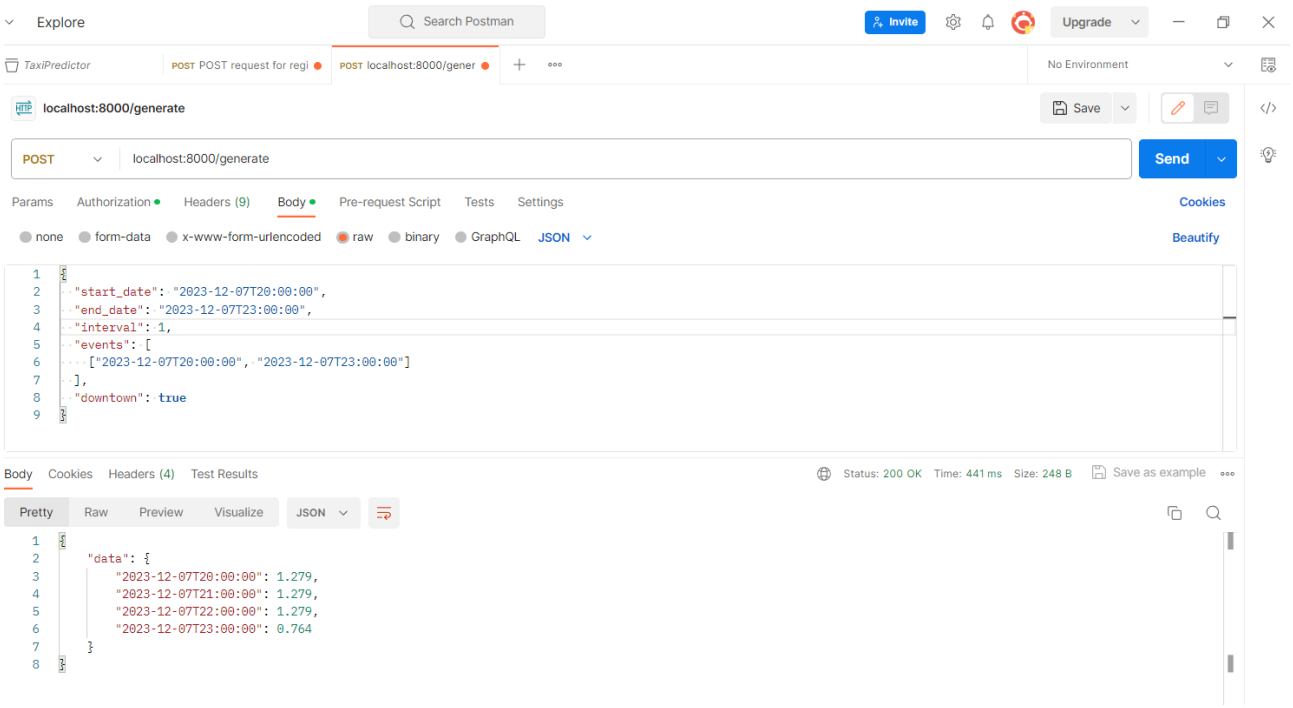


Рисунок 4.5 — Тестування запиту для отримання прогнозу

В запиті передаємо границі інтервалу прогнозу та інтервал між прогнозованими значеннями та вказуємо необхідні параметри. Створений сервіс автоматично отримує дані погоди за обраною локацією, використовуючи API OpenWeatherMap, та розраховує прогнозні дані з використанням моделі, що була розглянута в розділі 2. На підставі цього можемо зробити висновок, що робота створеного REST API є коректною та переходити до тестування користувацького додатку.

Перевірку користувацького додатку виконуємо відповідно до сценаріїв використання, що були розглянуті в другому розділі.

Першим з сценаріїв використання є реєстрація користувача. Екран реєстрації користувача наведено на рис. 4.6. На екрані користувач має можливість ввести адресу електронної пошти, пароль, що маскується, та надіслати дані на підтвердження реєстрації. Подальший вхід у додаток виконується з введеними реєстраційними даними для ідентифікації користувача.

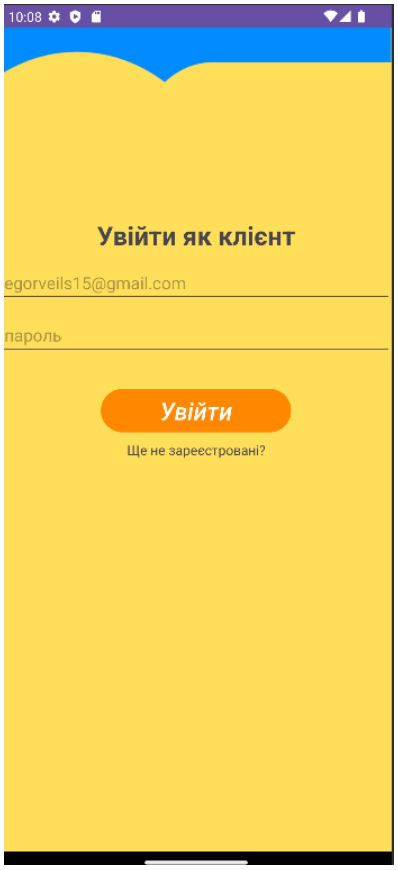
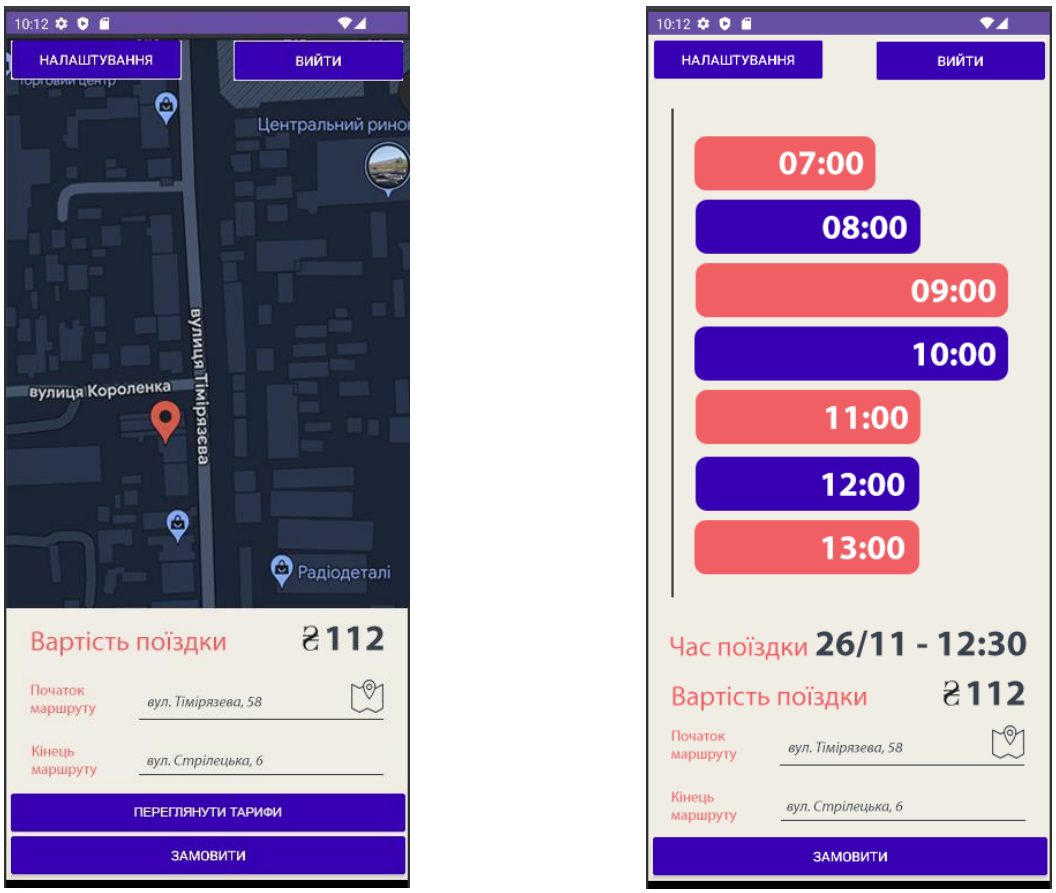


Рисунок 4.6 — Екран реєстрації користувача

Після реєстрації або входу користувач потрапляє на головний екран додатку, де можна обрати маршрут поточної поїздки у тому числі з використанням карти. Після вибору обох точок маршруту: початкової та кінцевої, користувач бачить поточну вартість перевезення й може або замовити поїздку зараз за поточною ціною, або переглянути прогноз вартості. Головний екран додатку показаний на рис. 4.7(а).

Натиснувши кнопку «Перегляд вартості» користувач від сценарію використання «Замовлення зараз» переходить до сценаріїв перегляду прогнозу вартості замовлення та створення відкладеного замовлення. Такий підхід до візуалізації прогнозу цін на поїздку забезпечує зручність користувачів та дозволяє додатку ефективно взаємодіяти з ними, забезпечуючи при цьому якість обслуговування та мінімізацію можливих конфліктів у комерційних питаннях. Екран перегляду прогнозу вартості, що поєднаний із створенням відкладеного замовлення, показаний на рис. 4.7(б). Перегляд прогнозу цін на поїздку

реалізований у вигляді стилізованої діаграми, де акцент зроблено на часу поїздки та графічному порівнянні вартості поїздки по годинах. Така реалізація з одного боку робить перегляд простим й зручним, не примушуючи вдивлятись в дрібні цифри на екрані смартфону, а з іншого боку відсутність точних цифр вартості поїздки дозволить уникнути непорозумінь під час комерційної експлуатації додатку, оскільки фактична вартість може відрізнятись від прогнозованої за низки причин.



а) головний екран

б) екран перегляду прогнозу

Рисунок 4.7 — Екрани додатку

4.3 Перевірка виконання вимог до програмного забезпечення.

Виконаємо перевірки виконання вимог до програмного забезпечення, що були сформульовані під час моделювання та проектування програмного забезпечення. Перевірку показано в таблиці 4.1 для створеного REST API, та в таблиці 4.2 для мобільного додатку.

Таблиця 4.1 — Перевірка виконання вимог до REST API.

№ з/п	Коротке формулювання вимоги	Шлях реалізації
1	Прийом та обробка JSON-запитів із даними	REST API приймає та обробляє запити у форматі JSON встановленої структури
2	Перевірка валідності та формату вхідних даних	Перевіряється валідність дат, що містить запит, формат вхідних даних перевіряється засобами Pydantic, що інтегровані в FastAPI.
3	Інтеграція з БД	Роботу з БД Postgre SQL інтегровано в модулю сервісу db. Використання ORM SQLAlchemy дозволяє за необхідності змінити систему управління базою даних.
4	Взаємодія з моделлю прогнозування	При запуску сервісу створюється екземпляр моделі, що використовує збережені при тренуванні моделі ваги та налаштування.
5	Запити до сторонніх сервісів	Погодні дані сервіс отримує шляхом HHTP запиту до OpenWeatherMap.
6	Формування відповідей	У відповідь формується прогноз відносного попиту із заданим у вхідному запиті часовим інтервалом. Відповідь

		сервісу надсилається в форматі JSON. Для форматування запиту також використовується модель Pydantic.
7	Мова програмування та фреймворк	Для створення сервісу використано Python та FastAPI відповідно до технічних вимог.

Продовження таблиці 4.1

8	Безпека	Реалізація безпекових механізмів обмежена базовою аутентифікацією користувача HTTP та збереженням облікових даних користувачів у зашифрованому вигляді. Для шифрування даних використовується алгоритм SHA-256.
9	Продуктивність і масштабованість	Масштабованість системи досягається за рахунок контейнеризації Docker, продуктивність за рахунок асинхронності та коректних налаштувань параметрів контейнерів.

Таблиця 4.2 — Перевірка виконання вимог до REST API.

№ з/п	Коротке формулювання вимоги	Шлях реалізації
1	Реєстрація та авторизація користувача	Реалізовано механізм реєстрації користувача за електронною поштою та паролем.

2	Відображення поточної позиції на карті, використання карти при створені замовлення.	На головному екрані додатку відображається карта з вказівкою, на поточну позицію користувача. Також певне місце на карті може бути обране під час створення замовлення як кінцева точка маршруту.
3	Замовлення поїздки	Реалізована механіка замовлення поїздки. Після введення початкової та кінцевої точок маршруту користувач може бачити орієнтовну вартість поїздки.

Продовження таблиці 4.2

4	Прогнозування коливань тарифу, спричинених зміною попиту.	З екрану перегляду маршруту та вартості користувач може перейти до форми перегляду коливань вартості поїздки, що показує графік вартості на підставі прогнозу, отриманого від API.
5	Продуктивність	Додаток використовує ресурси системи в оптимальний спосіб. Під час тестування перевірено відсутність витоку пам’яті, своєчасне видалення тимчасових файлів, тощо.
6	Надійність	Під час тестування виявлені та усунені недоліки, що приводили до збоїв в роботі програмного забезпечення.
7	Безпека	Додаток відповідає базовим вимогам до безпеки, що є необхідними для Android додатку та використовує безпекові

		протоколи Android. Разом з цим необхідне доопрацювання з метою підвищення рівня захисту особистих даних користувача.
8	Сумісність	Додаток протестований на різних пристроях, починаючи з версії операційної системи Android 9.0. Більш глибока зворотна сумісність не забезпечена.
9	Інтерфейс користувача	Інтерфейс користувача інтуїтивно зрозумілий, не має суттєвих відмінностей від звичного вже інтерфейсу додатків замовлення послу перевезення пасажирів.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку програмного засобу для прогнозування попиту на транспортні послуги, заснована на використанні нейронної мережі. Особливістю програми є створення інтегрованого рішення, яке враховує широкий спектр факторів, що впливають на затребуваність транспортних послуг, зокрема зміни погодних умов, що можуть суттєво впливати на рішення споживачів щодо вибору транспорту. Новизна полягає в створенні універсальної архітектури, яка забезпечує точність прогнозів та їх доступність для різних користувачів.

Аналогом може бути ANT-Logistics, приблизна ціна від 360 $ або 13095 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюють із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, увідповідності із табл. 5.1.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Крите-рій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продуктув реальних умовах
Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 5.1

Ринкові переваги					
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші,ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продуктутрохи кращі, ніжв аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих ком-паній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне не-значне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансуванн я є	Потрібні незначні фінансові ресурси. Джерела фінансуванн я є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні мате-ріали, що використовуютьсяу військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використову ються у виробництві

Продовження табл. 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних доку- ментів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	4	3	3
Наявність аналогів на ринку	4	4	4
Цінова політика	3	4	3
Технічні та споживчі властивості виробу	4	3	3
Експлуатаційні витрати	3	4	3
Ринок збуту	3	4	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	3	4	3
Матеріально-технічна база	3	3	4
Термін реалізації ідеї	4	4	3
Супровідна докумен5ація	4	3	4
Сума	42	42	44
Середньоарифметична сума балів	(44+43+41) / 3 = 42,66		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 - 10	Низький
11-20	Нижче середнього
21- 30	Середній
31 - 40	Вище середнього
41 -48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного програмного продукту є високим, що досягається за рахунок того, що програмний продукт є програмним засобом для прогнозування попиту на транспортні послуги, заснована на використанні нейронної мережі. Особливістю програми є створення інтегрованого рішення, яке враховує широкий спектр факторів, що впливають на затребуваність транспортних послуг, зокрема зміни погодних умов, що можуть суттєво впливати на рішення споживачів щодо вибору транспорту. Привабливість для споживачів полягає в отримуванні рекомендацій щодо найкращого часу для замовлення таксі, враховуючи поточний та прогнозований попит, а також альтернативні маршрути, що можуть знизити вартість поїздки та час у дорозі.

5.2 Прогнозування витрат на виконання науково-дослідної роботи.

Основна заробітна плата розробників розраховується за формулою

$$З_0 = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p} \qquad (5.1)$$

де М — місячний посадовий оклад конкретного розробника, грн.;

T_p — число робочих днів в місяці, 22 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	33199	1509,06	42	63380,52
Програміст	27555	1252,53	42	52606,26
Всього				115986,78

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

Додаткова заробітна плата розробників, які приймали участь в розробці обладнання розраховується як 14,3% від основної заробітної плати розробників та робітників:

$$З_д = \frac{З_о \cdot 14{,}3\ \%}{100\%} \quad (5.2)$$

$$З_д = (115986{,}78 \cdot 14{,}3\ \% / 100\ \%) = 16\,586{,}10\ (\text{грн.})$$

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = \frac{(З_о + З_д) \cdot 22\ \%}{100\%} \quad (5.3)$$

$$Н_з = (115986{,}78 + 16\,586{,}10) \cdot 22\ \% / 100\ \% = 29166\ (\text{грн.})$$

Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$\text{А} = \frac{\text{Ц}}{\text{Т}_\text{в}} \times \frac{t_{\text{вик}}}{12} \text{ (грн.)} \quad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років;

$t_{\text{вик}}$ — термін використання під час розробки, місяців.

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 23000 грн., термін його корисного використання згідно податкового законодавства — 2 роки, а термін його фактичного використання — 1,7 міс.

$$\text{А} = \frac{23000}{2} \times \frac{1{,}7}{12} = 1628{,}4 \text{ (грн.)}$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення.

Але, так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн. (Visual Studio Professional 1800 грн/міс, використовувався 1,82 місяці), то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $B_{нем.ак.}$

= 135040 грн.

Розрахунки заносимо до таблиці 5.5.

Таблиця 5.5 — Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн.
Комп'ютер та комп'ютерна периферія (HP Pavilion)	23000	2	1,7	1628,4
Офісне обладнання (меблі)	20600	4	1,7	757,576
Приміщення	1920000	20	1,7	13600
Всього				15985,96

Тарифи на електроенергію для непобутових споживачів

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподілелектроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати насилову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \qquad (5.5)$$

де В — вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,3 кВт;

Ф — фактична кількість годин роботи обладнання, годин;

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,9.

$$В_е = 0{,}9 \cdot 0{,}3 \cdot 8 \cdot 42 \cdot 6 = 544{,}32 \text{ (грн.)}$$

До статті «Інші витрати» належать витрати, які не знайшли відображенняу зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників

$$І_в = (З_0 + З_р) \cdot \frac{Н_{ів}}{100\%} \quad (5.6)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$І_в = 115986{,}78 \times \frac{59\%}{100\%} / = 68432{,}2 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.7)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$Н_{нзв} = 115986{,}78 * \frac{138\ \%}{100\%} = 160061{,}75 \text{ (грн.)}$$

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$В_{заг} = 115986,78 + 16\,586,10 + 29166 + 15985,96 + 13540 + 544,3 + 68432,2 + 160061,75 = 432503,11 \text{ грн.}$$

Загальні витрати на науково – дослідну діяльність

Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються ЗВ, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн) \%} \quad (5.8)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = 432503,11 / 0,5 = 865\,006,22 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить

потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково- технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її

можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N \cdot Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \quad (5.9)$$

де $\pm \Delta Ц_0$ — зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_0$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, Цо = Цб ± ΔЦо;

Цб — вартість програмного продукту у році до впровадження результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість і ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333;

р — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 5000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 1000 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року — на 2000

шт., протягом другого року — на 4000 шт., протягом третього року на 6000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0 \times 1000 + (5000 + 1000) \times 2000) \times 0{,}8333 \times 0{,}27) \times (1 - 0{,}18) = \\ = 2{,}213{,}911{,}44 \text{ грн.}$$

$$\Delta П_2 = (0 \times 1000 + (5000 + 1000) \times (2000 + 4000) \times 0{,}8333 \times 0{,}27) \times \\ \times (1 - 0{,}18) = 6641734{,}32 \text{ грн.}$$

$$\Delta П_3 = (0 \times 1000 + (5000 + 1000) \times (2000 + 4000 + 6000) \times 0{,}8333 \\ \times 0{,}27) * (1 - 0{,}18) = 22139114{,}4 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 22139114,4 грн.

Розраховуємо приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.10)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково- технічної) роботи, грн;

Т — період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}05...0{,}15$;

t — період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

$$ПП = (2{,}213{,}911{,}44/(1+0{,}1)^1) + (6641734{,}32/(1+0{,}1)^2) + (1328346{,}64/(1+0{,}1)^3) =$$

$$= 2\,012\,646{,}76 + 6\,716\,651{,}60 + 12794402{,}19 = 24141914{,}14 \text{ грн.}$$

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \times ЗВ, \quad (5.11)$$

де $k_{\text{інв}}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію і це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{\text{інв}}$*=2...5*, але може бути і більшим;

ЗВ − загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 \times 865006{,}22 = 1730012{,}44 \text{ грн.}$$

Тоді абсолютний економічний ефект Е*абс* або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП - PV, \quad (5.12)$$

$$Е_{абс} = 24141914{,}14 - 1730012{,}44 = 22411901{,}7 \text{ грн.}$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.13)$$

$Т_ж$ — життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{(1 + 22411901{,}7/1730012{,}44} - 1 = 1{,}407$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \quad (5.14)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d = (0,09…0,14);

f — показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{min} = 0{,}14 + 0{,}1 = 0{,}24.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даноїнаукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового

проектуінвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \quad (5.15)$$

$$T_{ок} = \frac{1}{1{,}407} = 0{,}7 \text{ р.}$$

Оскільки $T_{ок} <$ 3-х років, а саме термін окупності рівний 0,7 роки, то фінансування даної наукової розробки є доцільним.

Отже сума витрат на розробку програмного продукту складає 865006 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є високо конкурентоспроможним. Період окупності складе близько 0,7 роки.

ВИСНОВКИ

Результатом виконання роботи є всебічний аналіз клієнтських додатків компаній, що надають послуги міських пасажирських перевезень, а також моделювання, проектування, програмна реалізація та тестування програмного засобу підтримки замовлень перевезень, що має вбудований модуль прогнозування вартості перевезення. Програмна система складається з клієнтської частини у вигляді Android додатку, та REST API, що розгортається на сервері. В роботі послідовно враховані усі етапи розробки: від аналізу предметної області та створення концепції до тестування та оцінки економічної доцільності, що відтворює повний цикл створення реальних програмних комплексів.

В першому розділі роботи проведений багатобічний аналіз предметної області, що виявив значущі трансформації в секторі таксі, які є результатом стрімкого розвитку цифрових технологій. Детально розглянуті різноманітні аспекти цієї індустрії, включаючи зміни у споживацьких вподобаннях, нові підходи до взаємодії з клієнтами та зростання значимості мобільних додатків. Особлива увага приділена впливу технологій на організацію та управління послугами таксі, з акцентом на важливість аналітики даних та автоматизованого прогнозування. Ретельний огляд сучасних технологічних рішень у цій галузі показав, що використання алгоритмів машинного навчання, особливо штучних нейронних мереж, є перспективним напрямом для підвищення ефективності та оптимізації витрат.

Другий розділ присвячений аналізу чинників, що змінюють попит на послуги міських пасажирських перевезень легковим транспортом, а також побудовано модель прогнозування попиту з використанням штучних нейронних мереж та машинного навчання. Проведено оцінку точності кожного з методів, та надано перевагу використанню машинного навчання для створення прогнозу попиту в системі. Прогнозування попиту є базою для відтворення прогнозованої динаміки вартості перевезення за певним маршрутом в клієнтському додатку, що

також розробляється в роботі. В розділі запропоновано та обґрунтовано використання архітектури, що складається з клієнтського додатку та серверної частини, причому генерація прогнозів опрацьовується саме серверною частиною через високу потребу в ресурсах для функціонування моделей машинного навчання та необхідність збирання та агрегація даних в потоковому режимі. Також в розділі розглянуто сценарії використання системи, сформульовану функціональні та нефункціональні вимоги до клієнтського додатку та технічні вимоги до REST API.

Третій розділ роботи фокусується на виборі технологій та підходів для реалізації проекту. Використання Python та Java виявилося вдалою стратегією, оскільки ці мови забезпечують необхідну гнучкість, продуктивність та надійність. Особливо важливим є використання FastAPI для розробки REST API, що дозволило оптимізувати процес розробки та забезпечити високу швидкість обробки запитів. Також значний внесок у проект внесла контейнеризація за допомогою Docker, яка забезпечує легкість розгортання та стабільність середовища.

Четвертий розділ описує планування та виконання тестування програмного забезпечення, зокрема unit-тестування, що відіграло ключову роль у забезпеченні якості та надійності системи. Використання інструментів, як-от Postman для тестування REST API та Espresso для тестування Android-додатку, дозволило глибоко аналізувати роботу кожного компоненту системи, виявляючи та усуваючи помилки ще на ранніх етапах розробки. Саме це стало гарантією, що кінцевий продукт буде стабільним та надійним у використанні. До розділу також включено розгляд контрольного прикладу для кожного з компонентів системи, що дозволило висвітлити практичні аспекти користування програмною системою, що розроблена. Також важливими аспектом є питання безпеки, оскільки програмний продукт збирає та обробляє велику кількість даних. У роботі були враховані сучасні вимоги до безпеки даних, що гарантує захист особистої інформації користувачів та відповідність до нормативних вимог.

Економічний аналіз, проведений у розділі 5, вказує на ефективність інвестицій у розробку даного проекту. Витрати на розробку та маркетинг були ретельно проаналізовані, що дозволяє зробити висновок про економічну доцільність проекту. Маркетинговий аналіз підтвердив потенціал програмного продукту на ринку, з огляду на зростаючий попит на рішення у сфері транспорту та логістики.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Програмний засіб для прогнозування замовлень транспортного засобу на основі аналізу статистики попередніх запитів / І. Ю. Черняховський, О. С. Городецька, Л. А. Савицька // Матеріали МНПІК Молодь в науці: дослідження, проблеми, перспективи (МН-2024), 2024 р. [Електронний ресурс] — https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/19539/16188.

2. Uber [Електронний ресурс]. Режим доступу: https://www.uber.com/ua/uk/- Назва з екрану.

3. Bolt [Електронний ресурс]. Режим доступу: https://bolt.eu/uk-ua/ - Назва з екрану.

4. Lyft [Електронний ресурс]. Режим доступу: https://www.uber.com/ua/ - Назва з екрану.

5. Jones H., “An Essential Beginners Guide to Artificial Neural Networks and Their Role in Machine Learning and Artificial Intelligence”. CreateSpace Independent Publishing Platform, 2018, ISBN: 1647481023, стор. 76.

6. Teoh T., Rong Z., “Artificial Intelligence with Python”. Springer Nature, Singapore, 2022, ISBN: 9789811686153. 336 pages.

7. Tianqi C., Guestrin C., «XGBoost: A Scalable Tree Boosting System». The 22nd ACM SIGKDD International Conference, 2016. URL: https://arxiv.org/abs/1603.02754

8. Song T. та Hu T. «Research on XGboost academic forecasting and analysis modelling». The Second International Conference on Physics, Mathematics and Statistics, Hangzhou, China, 2019

9. Guolin K , Meng Q, Finley T., Wang T., Chen W., Ma W., Ye W., Liu T.-Y., «LightGBM: A Highly Efficient Gradient Boosting Decision Tree». Advances in Neural Information Processing System, № 30, 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

10. Coelho L., Richert W., “Building Machine Learning Systems with Python”. Packt, 2018, ISBN: 1782161406. 290 pages.

11. Miles R., Hamilton K., “Learning UML 2.0: A Pragmatic Introduction to UML”, 1st edition, O’Reilly media, 2006, ISBN: 0596009828. 290 pages.

12. Michael Dawson, Programming with Python, Course Technology, 2020, ISBN: 9781435455009, 455 pages.

13. Zed Shaw, Learn Python 3 the Hard Way, Course Technology, 2017, ISBN: 9780134692883, стор. 320.

14. Python [Електронний ресурс]. Режим доступу: https://docs.python.org/3/- Назва з екрану.

15. FastAPI [Електронний ресурс]. Режим доступу: https://fastapi.tiangolo.com/- Назва з екрану.

16. Developers [Електронний ресурс]. Режим доступу: https:// developer.android.com/ - Назва з екрану.

17. Java Is the Language of Possibilities [Електронний ресурс] – Режим доступу: https://www.oracle.com/java/technologies/ - Назва з екрану.

18. Docker [Електронний ресурс]. Режим доступу: https://www.docker.com/ - Назва з екрану.

19. PostgreSQL [Електронний ресурс]. Режим доступу: https://www.postgresql.org/ - Назва з екрану.

20. Simon Riggs, Gianni Ciolli, PostgreSQL 14 Administration Cookbook, Packt Publishing, 2022, ISBN: 9781803248974, стор. 608.

21. Васильєв О., Програмування мовою Python, Богдан, 2020. ISBN: 9789661056113. Стор. 504.

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
проф., д.т.н. О. Д. Азаров
«__» _____2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Програмний засіб для прогнозування замовлень транспортного засобу на основі аналізу статистики попередніх запитів»

08-54.МКР.046.00.000 ПЗ

Науковий керівник: к.т.н., доц. каф. ОТ
_____ Городецька О.С.
«____» _____ 2023 р.

Магістрант групи 2КІ-21м
____ Черняховський І. Ю.

ВНТУ 2023

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність дослідження пов’язана з постійною необхідністю підвищення конкурентної спроможності операторів міських індивідуальних пасажирських перевезень шляхом вдосконалення сервісу, у тому числі розширення функціональності клієнтських додатків.

1.2 Наказ про затвердження теми магістерської кваліфікаційної роботи.

2 Мета і призначення МКР

2.1 Мета роботи — розширення функціональних можливостей програмного засобу забезпечення сервісів індивідуальних пасажирських перевезень шляхом впровадження прогнозування попиту та тарифів на перевезення.

2.2 Призначення розробки — вдосконалення методу побудови архітектури підсистеми прогнозування попиту, який враховує потоковий збір даних, кластер зберігання великих даних, контейнеризацію для зручного розгортання та безперервної інтеграції, що дозволило розширити функціональні можливості програмного засобу.

3 Вихідні дані для виконання МКР

Виконати розробку програмного модуля генерації прогнозу попиту та тарифів на перевезення з використанням методів машинного навчання та штучних нейронних мереж, обґрунтувати вибір моделей дослідити та порівняти їх ефективність.

4 Вимоги до виконання МКР

МКР повинна задовольняти такі вимоги:

— запропонувати програмний модуль генерації прогнозів попиту та тарифів на перевезення;

— розробити алгоритм та структуру для програмного модуля генерації

прогнозів попиту та тарифів на перевезення;

— розробити структуру програмного модуля;

— забезпечити зручний та легкий для користувача графічний інтерфейс клієнтської частини додатку;

— результат роботи, а саме програмний модуль прогнозування попиту та тарифів на перевезення та клієнтський додаток, що використовує дану функіональність.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ	01.09.21	02.09.21	Вступ
2	Аналіз літературних джерел	03.09.21	09.09.21	Розділ 1
3	Розробка технічного завдання	10.09.2021	24.09.21	Технічне завдання
3	Розробка структури програмного модуля генерації прогнозів попиту та тарифів на перевезення	25.09.21	08.10.21	Розділ 2, розробка структури
4	Розробка програмного модуля генерації прогнозів попиту та тарифів на перевезення	11.10.21	29.10.21	Розділ 3, розробка програми
5	Практична реалізація, результати.	01.11.21	14.11.21	Розділ 4
6	Розробка економічної частини	15.11.21	30.11.21	Розділ 5
7	Оформлення пояснювальної записки	02.12.21	15.12.21	ПЗ, презентація

6 Матеріали, що подаються до захисту МКР

До захисту МКР подаються: пояснювальна записка МКР, ілюстративні та графічні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук рецензента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів розрахункової та графічної документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Едина система конструкторської документації. Основні написи»;

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

Діаграма діяльності створення прогнозу

Рисунок Б.1 — Діаграма діяльності створення прогнозу

ДОДАТОК В

Структура моделі XGBRegressor

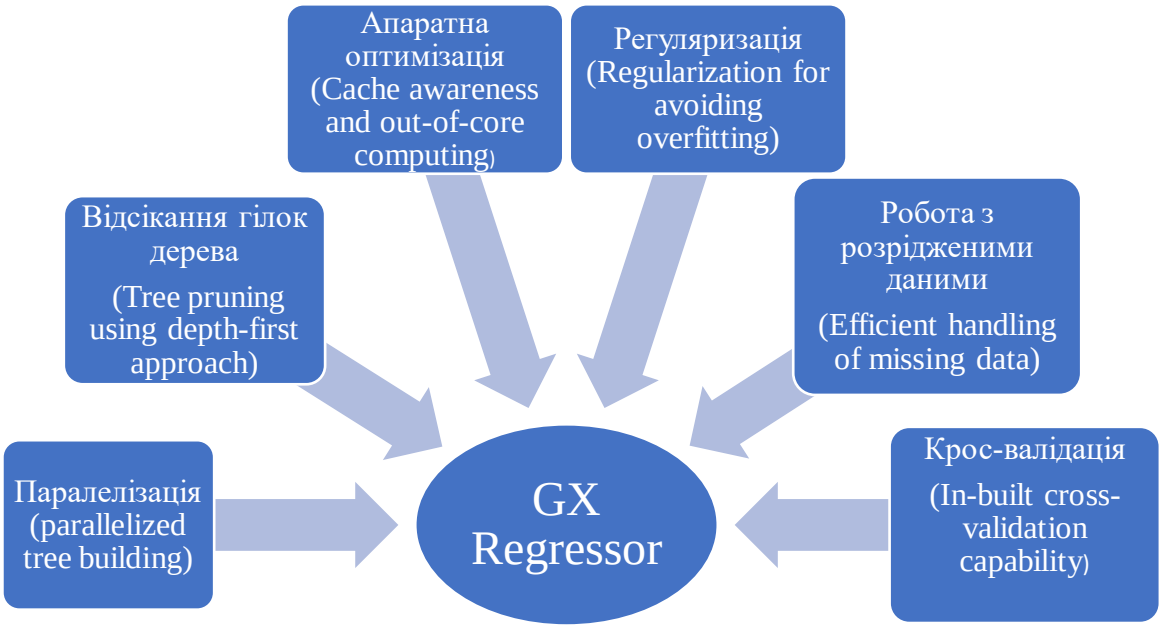


Рисунок В.1 — Структура моделі XGBRegressor

ДОДАТОК Г

Діаграма класів програмного модуля

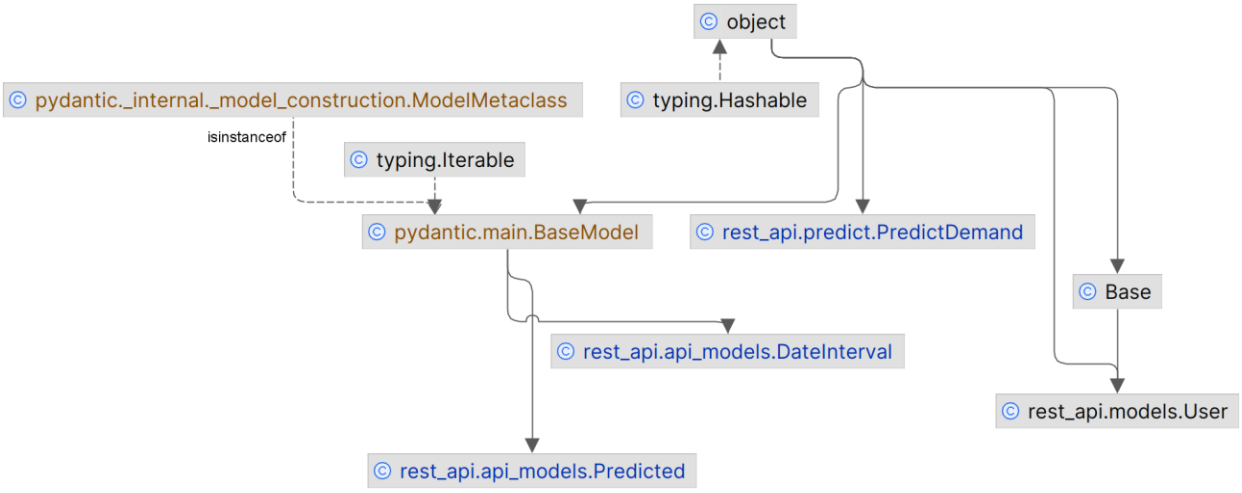


Рисунок Г.1 — Діаграма класів програмного модуля

ДОДАТОК Д

Діаграма розгортання системи

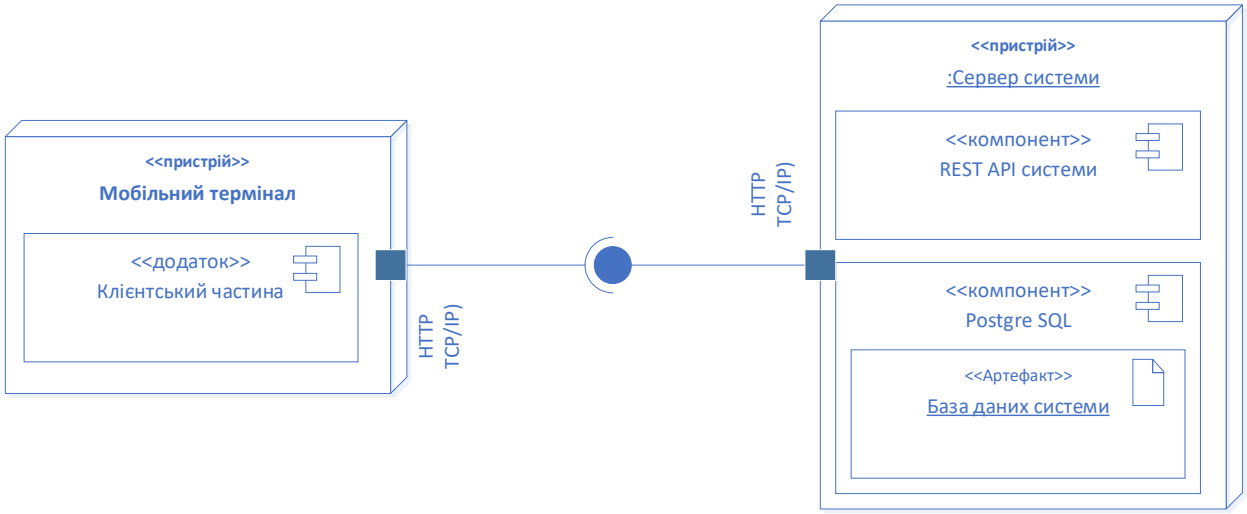


Рисунок Д.1 — Діаграма розгортання системи

ДОДАТОК Е

Лістинг програмного модуля

```
from datetime import datetime

from pydantic import BaseModel, Field
from typing import List, Tuple, Dict


class DateInterval(BaseModel):
    """
    Вхідні дані для створення прогнозу
    start_date: початковий момент для створення прогнозу
    end_date: кінцевий момент створення прогнозу
    interval: періодичність прогнозу в годинах
    events: інтервалу часу, коли відбувається проведення публічних заходів, що враховується в моделі під час
            створення прогнозу
    downtown: булева ознака, що прогноз створює прогноз для центральної частини міста, оскільки модель розрізняє
              центральну частину міста та інші при створенні прогнозу

    """
    start_date: datetime = Field(..., example="2023-01-01T09:00:00")
    end_date: datetime = Field(..., example="2023-01-10T17:00:00")
    interval: int = Field(..., gt=0, example=1)
    events: List[Tuple[datetime, datetime]] = Field(default=[], example=[("2023-01-02T10:00:00", "2023-01-02T12:00:00"),
                                                                          ("2023-01-05T15:00:00", "2023-01-05T18:00:00")]
                                                    )
    downtown: bool


class Predicted(BaseModel):
    data: Dict[datetime, float]



from cryptography.fernet import Fernet
```

```
def decrypt_api_key(encrypted_key, key_file_path):

    with open(key_file_path, "rb") as key_file:
        key = key_file.read()

    cipher = Fernet(key)
    decrypted_key = cipher.decrypt(encrypted_key)
    return decrypted_key.decode()



from fastapi import Depends, HTTPException, status
from fastapi import FastAPI
from fastapi.security import HTTPBasicCredentials, HTTPBasic
from passlib.context import CryptContext
from sqlalchemy.orm import Session

from .api_models import DateInterval, Predicted
from .models import User, SessionLocal
from .predict import PredictDemand

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")


def get_db():
    """
    Генератор сесій БД
    :return: екземпляр сесії БД
    """
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()


def get_password_hash(password):
    """
    Метод, що хешує пароль
    :param password: пароль, що вказаний символами
    :return: хешоване значення паролю
    """
    return pwd_context.hash(password)
```

```
def verify_password(plain_password, hashed_password):
    """
    Зіставлення символьного та хешованого паролю
    :param plain_password: пароль символами
    :param hashed_password: хешований пароль
    :return: булевий результат зіставлення
    """
    return pwd_context.verify(plain_password, hashed_password)


app = FastAPI()
security = HTTPBasic()


@app.post("/register/")
async def register_user(username: str, password: str, db: Session = Depends(get_db)):
    """
    Ендпоінт реїстрації користувача
    """
    hashed_password = get_password_hash(password)
    db_user = User(username=username, hashed_password=hashed_password)
    db.add(db_user)
    db.commit()
    db.refresh(db_user)
    return {"username": username}


@app.get("/alive")
async def alive():
    """
    Ендпоінт перевірки статусу сервісу
    """
    return {"status": "OK"}


def verify_credentials(credentials: HTTPBasicCredentials, db: Session) -> bool:
    """
    Метод перевірки даних авторизації користувача
    :param credentials: дані авторизації користувача
    :param db: екземпляр сесії БД, де зберігаються дані користувачів
    :return: булевий результат перевірки
    """
```

```
    username = credentials.username
    password = credentials.password

    db_user = db.query(User).filter(User.username == username).first()

    if db_user:
        return verify_password(password, db_user.hashed_password)

    return False


@app.post("/generate/", response_model=Predicted)
async def generate_prediction(data: DateInterval,
                              credentials: HTTPBasicCredentials = Depends(security),
                              db: Session = Depends(get_db)):
    """
    Ендпоінт створення передбачення за заданим інтервалом
    """

    if not verify_credentials(credentials, db):
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Incorrect username or password",
            headers={"WWW-Authenticate": "Basic"},
        )
    start, end = data.start_date, data.end_date

    if start > end:
        raise HTTPException(status_code=400, detail="Start dt must be before end dt")

    return PredictDemand.get(data)



from sqlalchemy import Column, Integer, String, create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

SQLALCHEMY_DATABASE_URL = "sqlite:///./users.db"
Base = declarative_base()


class User(Base):
```

```
    __tablename__ = "users"

    id = Column(Integer, primary_key=True, index=True)
    username = Column(String, unique=True, index=True)
    hashed_password = Column(String)


engine = create_engine(SQLALCHEMY_DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base.metadata.create_all(bind=engine)



import pathlib
from datetime import datetime
from datetime import timedelta, time
from typing import Tuple, Dict, List
import numpy as np

import requests
from numpy.typing import NDArray
import pandas as pd
import joblib

from .key_decrypt import decrypt_api_key
from .api_models import DateInterval, Predicted

CITY_NAME = "Odesa"
API_KEY =
"gAAAAABlXzax7YQ2zLqQ6JSA8rlVvqiAOW071uhH5pbIfcg1dY0JYpS9vYQ
RzVs4UlqhR_Bj9aX6nDL2sg2cT5D_XesLqlP7jTzWOCne80pVK9oTap6tHoQH
eMwyq2kO643LVcPnz140"

CURRENT_DIR = pathlib.Path(__file__).parent.resolve()
CYPHER_FILE = CURRENT_DIR/ 'encryption_key.key'
OHE_PATH = CURRENT_DIR / 'ohe.joblib'
XGB_PATH = CURRENT_DIR / 'xgb_model.joblib'

"""
Необхідний формат таблиці для формування передбачення

['day_part', 'weekday', 'downtown', 'downfall', 'public_event', 'extreme_temp',
'high_season', 'holiday']
```

```
"""


class PredictDemand:
    _ohe = joblib.load(OHE_PATH)
    _xgb_model = joblib.load(XGB_PATH)

    @classmethod
    def get(cls, data: DateInterval) -> Predicted:
        """
        Отримання передбачення по вхідних даних.
        Обгортка внутрішнього методу _get()
        :param data:
        :return: екземпляр Predicted
        """
        prediction_args = [data.start_date, data.end_date, data.interval, data.events, data.downtown]
        prediction_args += [cls._get_weather_data(data.start_date, data.end_date, data.interval)]
        return Predicted(data=cls._get(*prediction_args))

    @classmethod
    def _get(cls,
             start_date: datetime,
             end_date: datetime,
             interval: int,
             events: List[Tuple[datetime, datetime]],
             downtown: bool,
             weather_data: Dict[datetime, Tuple[NDArray[int], NDArray[bool]]]) -> Dict[datetime, int]:
        """
        Основний метод отримання передбачення по вхідних даних.
        Виконує перетворення вхідних даних на формат, аналогічний формату,
         що був використаний при навчанні моделі.
        Після отримання dataframe в необхідному форматі застосовується збережений при навчанні OneHotEncoder.
        На підготовлених даних застосовується збережений екземпляр XGBRegressor
        :param start_date: початок інтервалу прогнозування
        :param end_date: кінець інтервалу прогнозування
        :param interval: крок інтервалу прогнозування
        :param events: громадські заходи (інтервали часу)
        :param downtown: середмістя / віддалені райони
```

```
        :param weather_data: дані про погоду
        :return: передбачення, що є словником: ключі - момент часу в форматі ISO8601,
                                               значення - передбачення кількості поїздок
        """
        df = cls._get_df_for_prediction(start_date, end_date, interval, events, downtown, weather_data)

        X_initial_values = df[['day_part', 'weekday', 'downtown', 'downfall', 'public_event',
                               'extreme_temp', 'high_season', 'holiday']].values
        X_values = cls._ohe.transform(X_initial_values)

        predicted_demand = cls._xgb_model.predict(X_values)
        predicted_dict = {dt: np.round((np.round(prediction,
                                                 0)) / 140, 3)
                          for dt, prediction in zip(df['datetime'].values,
                                                    predicted_demand)}
        return predicted_dict

    @classmethod
    def _get_df_for_prediction(cls,
                               start_date: datetime,
                               end_date: datetime,
                               interval: int,
                               events: List[Tuple[datetime, datetime]],
                               downtown: bool,
                               weather_data: Dict[datetime, Tuple[NDArray[int], NDArray[bool]]]) -> pd.DataFrame:
        """
        Метод по наявних даним створює таблицю, що є ідентичною за структурою таблиці,
        що використовувалася при прогнозуванні. Це дозволить перетворити її аналогічним чином та отримати
        передбачення за заздалегідь навченими на великому датасеті ідентичної структури.
        :param start_date: початок інтервалу прогнозування
        :param end_date: кінець інтервалу прогнозування
        :param interval: крок інтервалу прогнозування
        :param events: громадські заходи (інтервали часу)
        :param downtown: середмістя / віддалені райони
        :param weather_data: дані про погоду
        :return: датафрейм формату, що необхідний для прогнозування
        """
```

```
        time_points = cls._generate_date_range(start_date, end_date, interval)
        df = pd.DataFrame({'points': time_points})
        df['day_part'] = df['points'].apply(lambda x: cls._time_of_day(x))
        df['weekday'] = df['points'].apply(lambda x: x.weekday())
        df['downtown'] = downtown
        df['downfall'] = df['points'].apply(lambda x: weather_data[x][1] if x in
weather_data else None)
        df['public_event'] = df['points'].apply(lambda x:
cls._is_datetime_in_intervals(x, events))
        df['temp'] = df['points'].apply(lambda x: weather_data[x][0] if x in
weather_data else None)
        df['extreme_temp'] = df['temp'].apply(lambda x: (x < -3) | (x > 26))
        df['high_season'] = df['points'].apply(lambda x: cls._tourist_season_range(x))
        df['holiday'] = df['points'].apply(lambda x: cls._is_date_in_holiday_range(x))
        df['datetime'] = df['points'].apply(lambda x: pd.to_datetime(x).isoformat())
        df.drop(columns=['temp', 'points'], inplace=True)
        return df

    @classmethod
    def _is_date_in_holiday_range(cls,
                                  dt: datetime):
        """
        Визначає, чи належить день до сезонів свят.
        :param dt: екземпляр datetime
        :return: булева ознака належності до сезонів свят
        """
        january_range = (datetime(dt.year, 1, 1), datetime(dt.year, 1, 10))
        march_range = (datetime(dt.year, 3, 7), datetime(dt.year, 3, 9))
        december_range = (datetime(dt.year, 12, 25), datetime(dt.year, 12, 26))

        if january_range[0] <= dt <= january_range[1] or \
                march_range[0] <= dt <= march_range[1] or \
                december_range[0] <= dt <= december_range[1]:
            return True
        else:
            return False

    @classmethod
    def _tourist_season_range(cls, dt: datetime) -> bool:
        """
        Визначає, чи належить день до туристичного сезону.
        :param dt: екземпляр datetime
        :return: булева ознака належності до туристичного сезону
        """
```

```
        date_only = dt.date()

        april_25 = date_only.replace(month=4, day=25)
        september_15 = date_only.replace(month=9, day=15)
        december_20 = date_only.replace(month=12, day=20)
        january_10 = date_only.replace(month=1, day=10)

        if april_25 <= date_only <= september_15 or december_20 <= date_only or
date_only <= january_10:
            return True
        else:
            return False

    @classmethod
    def _time_of_day(cls, dt: datetime) -> str:
        """
        Визначає частину дня за шаблоном, аналогічним до того,
        що використовувався при навчанні моделі
        :param dt: екземпляр datetime, що визначає момент часу
        :return: символьне значення частини дня
        """

        morning_peak_start = time(7, 0)
        morning_peak_end = time(11, 0)
        afternoon_end = time(16, 0)
        evening_peak_end = time(19, 30)
        late_evening_end = time(23, 0)

        current_time = dt.time()

        if morning_peak_start <= current_time < morning_peak_end:
            return "morning_peak"
        elif morning_peak_end <= current_time < afternoon_end:
            return "afternoon"
        elif afternoon_end <= current_time < evening_peak_end:
            return "evening_peak"
        elif evening_peak_end <= current_time < late_evening_end:
            return "late_evening"
        else:
            return "night"

    @classmethod
    def _is_datetime_in_intervals(cls,
                                  target_datetime: datetime,
```

```
                                intervals: List[Tuple[datetime, datetime]]) -> bool:
        """
        Перевіряє, чи потрапляє завданий момент часу в один із заданих інтервалів часу.
        :param target_datetime: момент часу, для якого визначається належність
        :param intervals: інтервали часу, до яких визначається належність
        :return: True, якщо завданий інтервал потрапляє в один з інтервалів переліку
        """
        for start, end in intervals:
            if start <= target_datetime <= end:
                return True
        return False

    @classmethod
    def _round_down_to_hour(cls,
                            dt: datetime) -> datetime:
        """
        Округляє datetime до найближчої години вниз.
        :param dt: екземпляр datetime
        :return: закруглення datetime
        """
        return dt.replace(minute=0, second=0, microsecond=0)

    @classmethod
    def _generate_date_range(cls,
                             start_date: datetime,
                             end_date: datetime,
                             interval_hours: int) -> List[datetime]:
        """
        Генерує список datetime від start_date до end_date з інтервалом interval_hours.
        :param start_date: початковий момент часу
        :param end_date: кінцевий момент часу
        :param interval_hours: інтервал
        :return: повертає згенерований перелік моментів часу
        """
        start_date_rounded = cls._round_down_to_hour(start_date)
        end_date_rounded = cls._round_down_to_hour(end_date)

        current_date = start_date_rounded
        dates = []
        while current_date <= end_date_rounded:
            dates.append(current_date)
```

```
            current_date += timedelta(hours=interval_hours)

        return dates

    @classmethod
    def _get_weather_data(cls,
                          start_time: datetime,
                          end_time: datetime,
                          generation_interval_hours) -> None | Dict[datetime,
Tuple[NDArray[int], NDArray[bool]]]:
        """
        Метод отримує прогноз погоди по місту, що потрібний для формування
передбачення попиту на послугу
        :param generation_interval_hours:
        :param start_time: початковий момент для прогнозу
        :param end_time: кінцевий момент для прогнозу
        :return: масив NumPy значень температури та масив NumPy булевих
значень,
         що визначають наявність опадів
        """

        decrypted_key = decrypt_api_key(API_KEY.encode(),
                                        pathlib.Path(__file__).parent.resolve()
/
"encryption_key.key")

        if end_time - start_time > timedelta(hours=48):
            return None

        url
=
f"http://api.openweathermap.org/data/2.5/forecast?q={CITY_NAME}&appid={d
ecrypted_key}&units=metric"
        response = requests.get(url)
        data = response.json()

        weather_data, response_data = {}, {}

        for item in data['list']:
            item_time = datetime.fromtimestamp(item['dt'])
            temp = item['main']['temp']
            precip = 'rain' in item or 'snow' in item
            weather_data.update({item_time: (temp, precip)})

        current_time = start_time.replace(minute=0, second=0, microsecond=0)
        while current_time <= end_time:
```

```
        filtered_weather_data = list(filter(lambda x: x[0] >= current_time, weather_data.items()))
        response_data.update({current_time: filtered_weather_data[0][1]})
        current_time += timedelta(hours=generation_interval_hours)

    return response_data



from unittest import TestCase
from rest_api.predict import PredictDemand

from datetime import datetime, timedelta


class TestWeatherRequest(TestCase):

    def setUp(self):
        self.interval = [datetime.now(), datetime.now() + timedelta(hours=24)]
        self.weather_response_mock = {datetime(2023, 11, 23, 18, 0): (5.43, False),
                                      datetime(2023, 11, 23, 21, 0): (5.44, False),
                                      datetime(2023, 11, 24, 0, 0): (5.42, False),
                                      datetime(2023, 11, 24, 3, 0): (4.97, False),
                                      datetime(2023, 11, 24, 6, 0): (5.54, False),
                                      datetime(2023, 11, 24, 9, 0): (9.65, False),
                                      datetime(2023, 11, 24, 12, 0): (10.2, False),
                                      datetime(2023, 11, 24, 15, 0): (8.96, False),
                                      datetime(2023, 11, 24, 18, 0): (8.21, False)}

    def test_weather_request(self):
        weather_data = PredictDemand._get_weather_data(*self.interval, 3)
        pass

    def test_get_df_for_prediction(self):
        actual = PredictDemand._get(start_date=datetime(2023, 11, 23, 18, 0),
                                    end_date=datetime(2023, 11, 24, 12, 0),
                                    interval=3,
                                    events=[(datetime(2023, 11, 23, 18, 0),
                                             datetime(2023, 11, 23, 21, 0))],
                                    downtown=False,
                                    weather_data=self.weather_response_mock)
        pass
```

```
version: '3'

services:
  api-service:
    build: .
    container_name: rest_api_app
    environment:
      PORT: 8000
    ports:
      - "8000:8000"
    command: uvicorn rest_api.main:app --host 0.0.0.0 --port 8000
    networks:
      - my_network

  db:
    image: postgres
    container_name: postgres_db_taxi
    environment:
      POSTGRES_DB: mydatabase
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - my_network

networks:
  my_network:

volumes:
  postgres_data:
```

```
FROM python:3.11

WORKDIR /usr/src

COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

COPY ./rest_api /usr/src/rest_api
```

```
import uvicorn

if __name__ == "__main__":
    uvicorn.run("rest_api.main:app", host="0.0.0.0", port=8000, reload=True)
```

ДОДАТОК Ж

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Програмний засіб для прогнозування замовлень транспортного засобу на основі статистики попередніх запитів

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 96,6% Схожість 3,4%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Черняховський І.Ю.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Городецька О. С.
(підпис) (прізвище, ініціали)