МАГІСТЕРСЬКА РОБОТА

МР. ІПм- 33.00.00.000 ПЗ

***Група ІПм*-23-2**

КісільВолодимир

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Кісіль Володимир Володимирович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Моделі та методи оптимізації та автоматизації логістики перевезень на основі засобів штучного інтелекту

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Кісіль В.В.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Лютак Ігор Зіновійович, д.т.н., професор

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Нормоконтроль

доц. Вовк Р.Б.

(посада) (підпис) (дата) (ініціали та прізвище)

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

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2024 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Кісілю Володимиру Володимировичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Моделі та методи оптимізації та автоматизації логістики перевезень на основі засобів штучного інтелекту ”

керівник проекту (роботи) Лютак Ігор Зіновійович, д.т.н., професор

затверджені наказом закладу вищої освіти від “ 22 ” листопада 2024 р. № 781/7

2. Строк подання студентом проекту (роботи) 15 грудня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування інформаційних та програмних технологій певного класу

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Огляд сучасного стану оптимізації логістики перевезень
2. Створення бота для взаємодії з логістами
3. Розробка моделей оптимізації та автоматизації логістики перевезень
4. Апробація та результати експериментів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Загальна схема процесу визначення у формі діаграми діяльності мови UML (рис. 1.1, ст. 21)
2. Сучасні методи оптимізації витрат (рис. 1.2, ст. 23)
3. Використання HTTP-методів (рис. 2.1, ст. 33) .
4. Проблеми маршрутизації транспортних засобів (рис. 3.1, ст. 63)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2024 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Огляд сучасного стану оптимізації логістики перевезень	01.08.2024	виконано
2	Розробка моделей оптимізації та автоматизації логістики перевезень	21.08.2024	виконано
3	Розробка бота для взаємодії з користувачами	30.10.2024	виконано
4	Створення інструментів взаємодії з ботом	12.11.2024	виконано
5	Створення документації для бота	15.11.2024	виконано
6	Реалізація оплати за використання бота	11.12.2024	виконано
7	Розміщення бота на хостингу та його запуст	12.12.2024	виконано
8	Затвердження пояснювальної записки роботи завідувачем кафедри	15.12.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 93 с., 15 рис., 40 джерел.

Тема: Моделі та методи оптимізації та автоматизації логістики перевезень на основі засобів штучного інтелекту

Об'єкт дослідження: Система безпеки даних сайтів на можливість парсінгу даних з них.

Мета роботи: Розвиток сучасних логістичних систем, особливо в умовах збільшення обсягів даних та потреби в автоматизації процесів, висуває нові вимоги до оптимізації перевезень.

Предмет дослідження: Можливість зчитування даних та їх подальше використання.

Результати дослідження:

Проаналізовано сайти з інформацією на предмет перевезення, знайдемо можливість зчитування інформації з них в обхід антипарсерам та маскування процесів під користувача. Розроблело телеграм бота для відображення необхідної інформації логістам та інструкцію по зваємодії з ним.

Висновок:

В результаті, був розроблений бот який помагає логістам швидко та оперативно відслідковувати інтересуючий їх товар та можливість його перевезення.

ЛОГІСТИКА ПЕРЕВЕЗЕНЬ, ШТУЧНИЙ ІНТЕЛЕКТ, ОПТИМІЗАЦІЯ МАРШРУТІВ, АВТОМАТИЗАЦІЯ ЛОГІСТИЧНИХ ПРОЦЕСІВ, ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ, СИСТЕМА УПРАВЛІННЯ ПЕРЕВЕЗЕННЯМИ, АНАЛІТИКА ДАНИХ, CRM-СИСТЕМА, ВЕБ-ЗАСТОСУНОК, АЛГОРИТМИ МАШИННОГО НАВЧАННЯ, ХМАРНІ ТЕХНОЛОГІЇ.

ANNOTATION

Master's thesis: 93 p., 15 figures, 40 sources.

Theme: Models and methods of optimisation and automation of transport logistics based on artificial intelligence

Object of research: Security system of data sites for the possibility of parsing data from them.

Purpose of the study: The development of modern logistics systems, especially in the context of increasing data volumes and the need for process automation, puts forward new requirements for transport optimisation.

Subject of research: The ability to read data and its further use.

Research results:

We analysed websites with information on transportation, found the possibility of reading information from them bypassing antiparsers and disguising processes as a user. We developed a telegram bot to display the necessary information to logisticians and instructions for interacting with it.

Conclusion:

As a result, we developed a bot that helps logisticians quickly and efficiently track the goods they are interested in and the possibility of their transportation.

TRANSPORTATION LOGISTICS, ARTIFICIAL INTELLIGENCE, ROUTE OPTIMISATION, AUTOMATION OF LOGISTICS PROCESSES, INFORMATION TECHNOLOGY, TRANSPORTATION MANAGEMENT SYSTEM, DATA ANALYTICS, CRM SYSTEM, WEB APPLICATION, MACHINE LEARNING ALGORITHMS, CLOUD TECHNOLOGIES.

ЗМІСТ

Стр.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД - база даних

ІС - інформаційна система

ІТ - інформаційні технології

ООП - об’єктно-орієнтоване програмування

ОС - операційна система

ПЗ - програмне забезпечення

VRP - Vehicle Routing Problem (Проблема маршрутизації транспортних засобів)

ВСТУП

Актуальність роботи

Розвиток сучасних технологій стимулює автоматизацію та оптимізацію багатьох сфер людської діяльності, включно з логістикою. Логістика є одним із ключових елементів глобальної економіки, адже вона забезпечує ефективний рух товарів між виробниками та споживачами. Однак, у сучасному світі логістика стикається з численними викликами, серед яких складність управління великими обсягами даних, підвищення вимог до швидкості та точності перевезень, а також необхідність зменшення витрат і підвищення ефективності процесів. Усе це створює потребу у використанні інноваційних підходів, таких як штучний інтелект (ШІ).

Штучний інтелект відкриває нові можливості для вирішення складних задач у логістиці, включаючи прогнозування попиту, оптимізацію маршрутів, управління транспортними засобами та автоматизацію взаємодії з клієнтами. Використання засобів ШІ дозволяє не лише значно зменшити час обробки даних, але й підвищити точність прийняття рішень, адаптуючи їх до умов, що постійно змінюються.

Застосування таких технологій в контексті логістичних перевезень є особливо актуальним, враховуючи динамічний характер цього сектора.

У роботі представлено проект, який включає:

- Телеграм-бот, що інформує логістів про нові вантажі (лоуди) на платформі Central Dispatch та дозволяє створювати фільтри для автоматизованого отримання релевантної інформації.
- Веб-сайт, який аналізує всі доступні вантажі з урахуванням індивідуальних параметрів логіста, таких як місце знаходження, бажаний маршрут, характеристики транспортного засобу тощо.

Реалізація даного проекту демонструє, як засоби штучного інтелекту можуть сприяти підвищенню ефективності логістичних процесів, знижуючи навантаження на логістів та забезпечуючи оптимізацію витрат.

У роботі будуть розглянуті такі аспекти:

- Теоретичні основи автоматизації логістики за допомогою ШІ.
- Опис розробленого проекту та його функціональних можливостей.

- Аналіз ефективності впровадження даних рішень у реальних умовах.

Таким чином, дана магістерська робота покликана не лише продемонструвати можливості використання сучасних технологій у логістиці, але й запропонувати практичне рішення для оптимізації логістичних перевезень, що може бути впроваджене у реальних бізнес-процесах.

Порівняння роботи з відомими розв’язаннями проблеми

Розв'язання задач автоматизації логістики на основі штучного інтелекту вже активно досліджується і впроваджується в сучасних системах. Серед найвідоміших підходів можна виділити наступні:

Сучасні TMS-системи, такі як SAP Transportation Management або Oracle Transportation Management, пропонують потужні інструменти для управління перевезеннями. Вони забезпечують інтеграцію даних, оптимізацію маршрутів та відстеження вантажів у реальному часі. Проте, ці системи часто мають високу вартість впровадження та потребують значних ресурсів для адаптації під конкретні потреби бізнесу.

Qguar TMS - система, що здійснює планування, моніторинг і розрахунок транспортних процесів. Відстеження партій товарів у дистриб'юторському ланцюзі та широкі можливості для розрахунку вартості транспортування, є тільки основними можливостями системи. Перевагою Qguar TMS можна назвати тісний зв'язок з іншими системами Qguar, а особливо з Qguar WMS (або з іншою системою цього класу). Qguar TMS дає змогу керувати небезпечними вантажами та розраховувати логістичні операції. Оптимальне планування маршрутів, використовуючи методи штучного інтелекту, об'єднання перевезень, перевантаження і багато інших функцій, роблять цю систему потужним інструментом не тільки для диспетчерів[1].

Різноманітні стартапи та технологічні компанії, такі як Convoy та Uber Freight, використовують алгоритми машинного навчання для підбору вантажів, прогнозування попиту та оптимізації маршрутів. Хоча ці підходи демонструють високу ефективність, їх впровадження вимагає значних обсягів історичних даних та складних моделей, що не завжди доступно для малих і середніх компаній.

В основу проекту покладено технологічні рішення, які містяться в інтерфейсах прикладного програмування (API). Учасниками є Uber Freight, постачальник цифрової мережі вантажних перевезень Convoy і постачальник вантажних та інтермодальних перевезень JB Hunt.

За словами експертів, новий підхід та створені новітні стандарти мають підвищити ефективність транспортної логістики, а також суттєво скоротити витрати часу і грошей на автомобільні перевезення.

Вже у першому кварталі 2023 року ініціатори проекту мають намір презентувати початковий набір стандартів та зразки нових документів, які можна буде використовувати на важковаговому автотранспорті. Завдяки універсальним стандартам для API можна спростити процедуру пошуку замовлень на перевезення вантажів, а також ефективно боротися з затримками у дорозі, у тому числі при завантаженні та розвантаженні машин. Переваги відчують усі учасники процес, у тому числі вантажовідправники та отримувачі, стверджують представники групи.

Три компанії для досягнення своєї мети створили Консорціум стандартів розкладу (SSC) та запросили до співпраці інших гравців у галузі, таких як транспортні брокери, 3PL-оператори та постачальники транспортних і складських систем управління.

Минулого року Фонд незалежної асоціації водіїв-власників провів опитування, яке показало, що водії витрачають до 33% свого можливого часу за кермом у простоях. А у звіті Міністерства транспорту США за 2018 рік вказано, що простої призвели до скорочення річного доходу автотранспортної галузі на 1,1-1,3 мільярда доларів США.

Білл Дрігерт, співзасновник і керівник відділу операцій Uber Freight, сказав: «Якщо ми не узгодимо стандарти, ми створимо більше роботи для всіх у найближчі роки. Усі виграють, якщо ми зможемо узгодити спільні способи взаємодії»[2].

Інтеграція чат-ботів у логістичні процеси стає все популярнішою. Наприклад, Amazon використовує ботів для автоматизації підтримки клієнтів і відстеження замовлень. Такі рішення добре підходять для автоматизації взаємодії з користувачами, але їх функціональність обмежується рамками конкретних задач.

Розроблений у межах цієї роботи телеграм-бот та веб-сайт поєднують переваги зазначених підходів, пропонуючи:

Легкість впровадження, що робить систему доступною для малого та середнього бізнесу.

Високий рівень персоналізації за рахунок можливості створення індивідуальних фільтрів.

Інтеграцію з існуючими платформами, такими як Central Dispatch, без необхідності значних витрат на адаптацію.

Таким чином, запропоноване рішення відрізняється простотою, гнучкістю та доступністю, що дозволяє вирішувати широкий спектр задач у логістиці, адаптуючись до потреб користувачів.

Мета і задачі дослідження

Метою магістерської роботи є розвиток сучасних логістичних систем, особливо в умовах збільшення обсягів даних та потреби в автоматизації процесів, висуває нові вимоги до оптимізації перевезень. Головною метою даного дослідження є розробка моделей і методів, які дозволяють забезпечити ефективність та автоматизацію логістичних перевезень на основі засобів штучного інтелекту.

Для досягнення поставленої мети визначено такі основні задачі:

1. Аналіз сучасного стану автоматизації логістичних процесів:
   - Вивчення існуючих рішень у сфері логістики з використанням ШІ;
   - Виявлення переваг та недоліків цих рішень;
   - Визначення ключових викликів і проблем, які залишаються невирішеними.
2. Розробка концепції оптимізації логістики перевезень:
   - Формалізація задачі оптимізації перевезень з урахуванням особливостей галузі;
   - Визначення параметрів, які впливають на вибір маршрутів і вантажів (відстань, витрати, час тощо);
   - Створення адаптивного підходу, який враховує змінні умови.

3. Розробка та впровадження практичних рішень:

- Створення телеграм-бота для оперативного інформування логістів;
- Розробка веб-сайту для аналізу та обробки великого обсягу даних про вантажі;
- Інтеграція розроблених інструментів із платформою Central Dispatch.

4. Аналіз ефективності запропонованих рішень:

- Оцінка точності та швидкості роботи створених моделей;
- Порівняння з існуючими підходами за ключовими показниками ефективності;
- Визначення економічних переваг використання розробленої системи.

5. Рекомендації щодо впровадження:

- Розробка рекомендацій для бізнесу з впровадження розроблених інструментів;
- Визначення можливостей масштабування системи та її адаптації для інших галузей.

Виконання цих задач дозволяє досягти гармонійного поєднання теоретичних і практичних аспектів автоматизації логістики. Результати дослідження стануть основою для створення ефективних рішень, здатних задовольнити потреби логістичних компаній у сучасних умовах ринку.

Об’єктом дослідження є моделі та алгоритми оптимізації і автоматизації логістичних процесів перевезення вантажів з використанням засобів штучного інтелекту.

Предметом дослідження є інформаційні технології оптимізації логістики, інтеграція алгоритмів штучного інтелекту для прийняття рішень у перевезеннях вантажів і забезпечення ефективності взаємодії між компонентами логістичних систем.

Методи дослідження. Для визначення вимог до розробки програмного забезпечення було використано аналітичний підхід до існуючих рішень у сфері логістики, побудову математичних моделей для оптимізації маршрутів і управління перевезеннями, моделювання процесів із застосуванням алгоритмів штучного інтелекту та експериментальну перевірку роботи розробленої системи.

Наукова новизна одержаних результатів.

Виконано аналіз сучасних методів автоматизації логістичних процесів і алгоритмів штучного інтелекту, що дозволило запропонувати нову архітектуру та алгоритми для оптимізації маршрутів перевезень. Це забезпечило підвищення ефективності планування маршрутів і зменшення витрат

Практичне значення одержаних результатів

Результати дослідження мають суттєве практичне значення, оскільки дозволяють автоматизувати процеси логістики перевезень, зменшити витрати часу на пошук оптимальних рішень і підвищити ефективність роботи логістичних компаній. Запропоновані моделі та інструменти, зокрема телеграм-бот і веб-сайт, забезпечують швидке опрацювання великих обсягів даних, адаптацію до змінюваних умов і покращення прийняття рішень у реальному часі.

Особистий внесок

1. Запропонований автором підхід до пошуку та обробки інформації у логістивній сфері;

2. Приведена реалізація алгоритмів, що виконують швидку та масивну обробку даних, одержаних як результат парсінгу сайтів з відповідною інформацією

Структура та обсяг магістерської роботи

Магістерська робота викладена на 93 сторінках друкованого тексту, який складається із вступу, чотирьох розділів, висновків, списку використаних джерел (40 найменування). Робота містить 15 рисунки

РОЗДІЛ 1

ОГЛЯД СУЧАСНОГО СТАНУ ОПТИМІЗАЦІЇ ЛОГІСТИКИ ПЕРЕВЕЗЕНЬ

1.1 Аналіз проблем в організації логістики перевезень

1.1.1 Недостатня швидкість та час доставки

Логістика та транспортування є двома найважливішими аспектами управління ланцюгом поставок. Якщо логістика виконується неналежним чином, це може спричинити вузьке місце у всьому ланцюжку поставок і порушити потік товарів. Бізнес постраждає від втрати репутації, якщо логістика не буде виконана належним чином.

Транспортна логістика – це функціональна сфера логістики, яка займається управлінням руху матеріальних потоків в процесі їх переміщення від постачальника до кінцевого споживача. Основна мета транспортної логістики полягає в організації такої схеми переміщення вантажів, яка б забезпечувала надійність, вчасність та безпечність їх поставки.

Транспортна логістика є важливою для підприємства, тому щозлагоджена система логістики дає змогу швидко й ефективно доставляти продукти в різні пункти призначення.

Надійна логістика може збільшити вартість бізнесу, тому що споживачі готові платити більше за продукти, доставлені швидко та якісно.

Добре організована система транспортної логістики може допомогти підприємствам зменшити загальні витрати. Така система логістики дозволяє підприємствам доставляти продукцію ефективніше та уникати затримок.

Ще одна перевага наявності ефективної транспортної логістичної системи полягає в тому, що вона може надати конкурентну перевагу над конкурентами.

Транспортна логістика в сучасний час перебуває під величезним тиском, тому існують певні проблеми:

1. Якість транспортного обслуговування.

2. Швидкість і час доставки.

3. Неефективне використання маршрутів доставки продукції від виробника до споживача.

4. Незадовільний стан автомобільних доріг.

5. Високий рівень фізичного і морального зносу рухомого складу транспорту.

6. Втрати від простою в очікуванні завантаження/розвантаження транспортного засобу.

7. Втрати від неефективної роботи

Перспективами транспортної логістики та вирішенням деяких з цих проблем може бути оптимізація маршрутів доставки за допомогою аналізу даних. Завдяки координуванню вхідних замовлень і прогнозуванню умов дорожнього руху можна скоротити витрати.

Також використовуючи дані для прийняття обґрунтованих рішень щодо оптимізації маршруту, можна значно збільшити бюджет логістики. Зменшуючи пробіг, зменшується споживання палива.

Завдяки оптимізації маршруту можна максимізувати ефективність наявного автопарку та водіїв в організації.

Оновлення автоматизації здатне покращити як швидкість, так і ефективність, водночас борючись із кадровими проблемами та зростаючим попитом на товари.

Для економії палива та меншої кількості переміщень транспорту можливим варіантом є пошук вантажу на зворотній напрямок, щоб вантажівка не їхала порожньою.

Україна повинна долучитися до поліпшення загальноєвропейського логістичного простору (зокрема, логістичних центрів), що включає у себе поліпшення показників енергоефективності транспортних засобів; оптимізацію функціонування мультимодальних логістичних схем; більш ефективне використання логістичної інфраструктури за рахунок удосконаленого управління перевезеннями, складування та інформаційних систем; оптимізацію потужності для задоволення зростаючого попиту на логістику України та регіонів ЄС.

1.1.2 Підвищення цін на пальне

Проблеми, з якими сьогодні стикаються вантажні перевезення, численні та різноманітні. Деякі з цих перешкод включають недостатнє фінансування та координацію між державним і приватним секторами. Інші включають відсутність даних про транспортні питання, недостатню координацію між державами та обмежене суспільне визнання ролі транспорту в економічному розвитку.

Серед найпоширеніших проблем у транспорті сьогодні – зростання вартості палива, недостатня видимість, збільшення заторів і викидів вуглецю. Вирішення цих проблем вимагає багатогранного підходу та інноваційних технологій. Крім того, процес, ймовірно, затягнеться на роки.

Вартість палива

Зростання вартості палива змушує перевізників підвищувати ціни або нести збитки. Це впливає на всю транспортну галузь, включаючи вантажовідправника, одержувача та перевізника. Зростаючі витрати на паливо впливають як на постачальників, так і на виробників, оскільки витрати на вантажні перевезення вищі. Вищі ціни на паливо також впливають на ціни товарів та підвищують темпи інфляції. Ось чому вирішення проблеми зростання вартості палива є критичною проблемою для транспортної галузі.

Відсутність видимості

Сьогодні 75% вантажовідправників вважають видимість вантажу одним із найважливіших питань у ланцюжку постачання після зниження витрат і швидкості. Завдяки видимості в реальному часі компанії можуть швидко реагувати на потенційні збої в ланцюжку постачання. У минулому відстеження та моніторинг вантажу були можливі лише тоді, коли він досяг певної віхи. Але сьогодні видимість є обов’язковою умовою для ефективності ланцюжка поставок.

Без видимості компанії страждають від неефективності та несуть непотрібні витрати. Без даних у реальному часі вантажовідправники не можуть керувати статусом своїх відправлень. Неможливість контролювати та відстежувати витрати на вантажні перевезення безпосередньо впливає на кінцевий результат.

Відсутність видимості може вплинути на весь ланцюжок поставок, включаючи операції розподілу. Наприклад, несвоєчасне постачання може призвести до необхідності простою виробничих ліній на фабриці, дефіциту запасів або коригування персоналу на вантажних платформах. Це також може мати негативний вплив на продуктивність інших частин ланцюга постачання. В результаті, 54% від 3PLs повідомили про втрату бізнесу через відсутність видимості.

Затори та викиди вуглецю

Бюджетне управління Конгресу зазначає, що затори є однією з найбільших проблем, з якими сьогодні стикається індустрія вантажних перевезень. Порівняно з ситуацією десятирічної давнини, загальна кількість часу, який потрібен поїздам, щоб працювати на повну потужність, становить тепер у п'ять разів вище ніж у 1982 році. Кілька факторів сприяли цій збільшеній затримці, включаючи різке зростання міжнародної торгівлі, як наслідок, дефіцит вантажівок і збільшення рівня пасажиропотоку.

Затори впливають на всіх – вони знижують продуктивність працівників і шкідливі для навколишнього середовища. Крім того, що затори знижують продуктивність, вони також призводять до збільшення забруднення. Незважаючи на те, що це очевидна проблема, вона не отримала належної уваги з боку влади та громадськості.

1.2 Традиційні методи оптимізації логістичних процесів

1.2.1 Процес визначення і вирішення завдань оптимізації

Загалом процес постановки і визначення завдань оптимізації можна подати у формі взаємопов’язаних етапів, на кожному з яких виконуються певні дії, спрямовані на побудову та подальше використання інформаційно-логічних моделей систем. Характерною особливістю цього процесу є його циклічність або ітеративність, що відображає сучасні вимоги щодо аналізу та проєктування складних систем.

Окремими етапами процесу визначення та розв’язання завдань оптимізації такі:

1. Аналіз проблемної ситуації.
2. Побудова математичної моделі.
3. Аналіз моделі.
4. Вибір методу та засоби вирішення.
5. Виконання чисельних розрахунків.
6. Аналіз результатів розрахунків.
7. Застосування результатів розрахунків.
8. Корекція та доопрацювання моделі.

Конкретний зміст кожного з етапів залежить від специфічних особливостей розв’язуваних завдань оптимізації в тій чи іншій проблемній області. До того ж кожен новий цикл процесу визначення і розв’язання задачі ініціюється етапом аналізу проблемної ситуації, у чому виявляється реалізація вимоги проблемно-орієнтованого підходу щодо побудови й використання інформаційно-логічних моделей систем для розв’язання завдань оптимізації.

Аналіз проблемної ситуації. Одним з основних принципів системного моделювання є проблемна орієнтація процесів побудови та використання моделей. Інакше кажучи, модель конкретної системи будується в контексті вирішення певної проблеми або досягнення певної мети. Головне призначення першого етапу – логічне осмислення конкретної проблеми в контексті методології системного моделювання. До того ж виконується аналіз всіх наявних ресурсів (матеріальних, фінансових, інформаційних та інших), необхідних для побудови моделі, її використання та реалізації отриманих результатів із метою вирішення наявної проблеми.

У разі відсутності необхідних ресурсів на даному етапі може бути прийняте розв’язання або про звуження (зменшення) розв’язуваної проблеми, або взагалі про відмову від використання засобів системного моделювання. На цьому етапі також виконується аналіз вимог, що висуваються в тій чи іншій формі до результату вирішення проблеми.

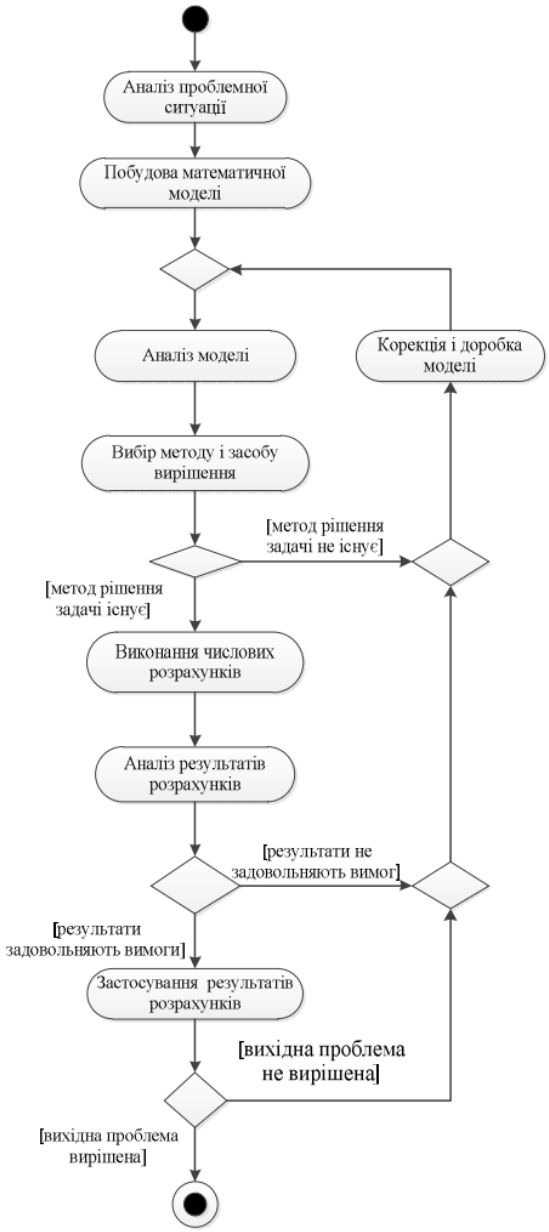

Рис. 1.1. Загальна схема процесу визначення та розв’язання завдань оптимізації у формі діаграми діяльності мови UML

1.2.2 Економічні критерії оптимізації логістичних процесів

Основними економічними критеріями оптимізації логістичних процесів є витрати та прибуток. Усі інші є різного ступеня похідними від цих основних критеріїв. Наприклад, собівартість, наведені витрати, частка ринку тощо.

У сучасних умовах постає питання щодо сучасних методів оптимізації витрат, що застосовуються в умовах вітчизняних підприємств. Але кожен із них має свої переваги й недоліки та може бути застосований за певних умов.

Наприклад, одним із найефективніших методів виявлення резервів зниження собівартості продукції є функціонально-вартісний аналіз (далі –ФВА). Функціонально-вартісний аналіз – комплексний системний аналіз діяльності підприємства, його складників (технічних і технологічних, маркетингових, фінансових, збутових підрозділів, управлінських функцій), а також оцінка ефективності використання витрат на реалізацію кожної з цих функцій з метою виявлення неефективних, нераціональних витрат, існуючих внутрішніх резервів і розроблення програм підвищення ефективності діяльності та збільшення прибутковості підприємства.

Основні переваги ФВА:

- більш чітке вивчення вартості продукції, що унеможливлює прийняття обґрунтованих стратегічних рішень щодо призначення цін на продукцію, вибору між можливостями виготовляти самостійно та купувати напівфабрикати;
- зрозумілість функцій, за допомогою яких підприємствам вдається приділяти ваги підвищенню ефективності високовартісних операцій тощо.

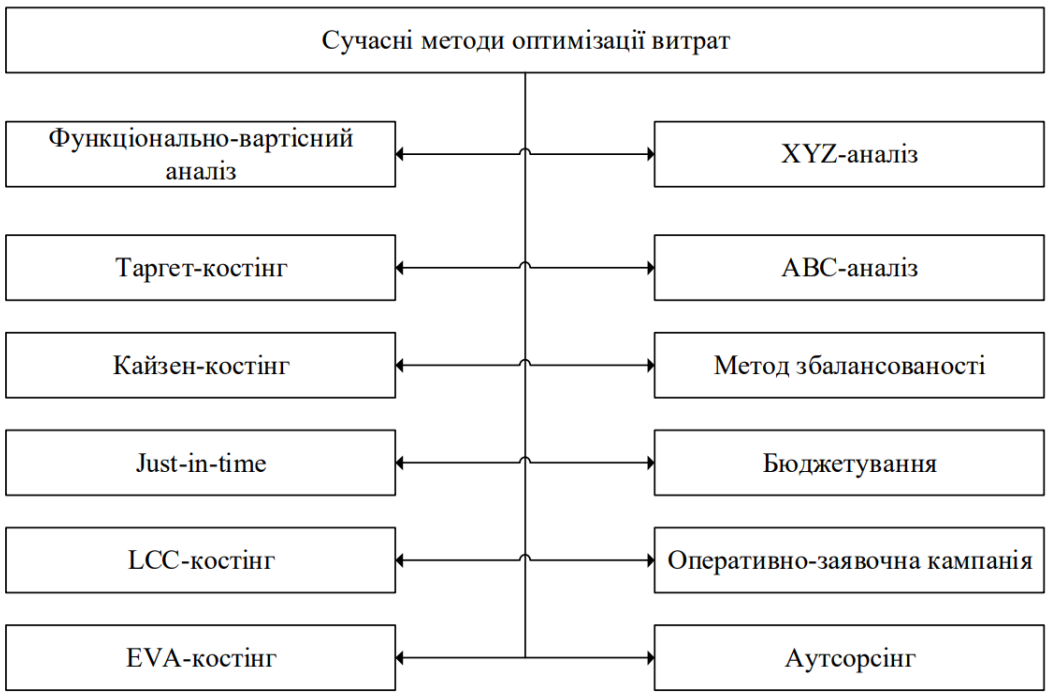

Рис. 1.2. Сучасні методи оптимізації витрат

1.3 Використання штучного інтелекту у логістиці: сучасний стан і перспективи

1.3.1 Оптимізація маршрутів

Оптимізація Маршрутів

ШІ дозволяє розробляти оптимальні маршрути доставки, враховуючи трафік, погодні умови, стан доріг і навіть час доби. Це допомагає зменшити час доставки, витрати на пальне та підвищити точність обслуговування клієнтів.

Приклад: Компанії, які використовують алгоритми ШІ для маршрутів, зменшують витрати на транспорт до 15-20%.

Управління Запасами

ШІ допомагає аналізувати дані продажів, прогнозувати попит і автоматизувати поповнення запасів. Це особливо важливо для складів і роздрібних компаній, де зменшення надлишкових товарів означає скорочення витрат.

Цікаво знати: Завдяки ШІ компанії знижують ризик дефіциту товарів на 35%.

Відстеження Вантажів у Реальному Часі

Технології IoT у поєднанні зі ШІ дозволяють моніторити вантажі в реальному часі, надаючи клієнтам точну інформацію про місцезнаходження їхніх товарів. Це підвищує прозорість і довіру клієнтів.

Перевага: Оперативні сповіщення дозволяють швидко реагувати на будь-які відхилення в доставці.

Прогнозування та Аналіз Даних

ШІ аналізує великі обсяги даних, допомагаючи передбачити затримки, проблеми в ланцюгу постачання та сезонні піки попиту. Це дозволяє компаніям краще планувати свої операції.

Приклад: Завдяки ШІ рітейлери можуть заздалегідь підготуватися до святкових періодів, коли попит на доставку значно зростає.

Автоматизація Складів

Роботизовані системи управління складами на базі ШІ дозволяють автоматизувати процеси сортування, пакування та відправки. Це скорочує час виконання замовлень і зменшує кількість помилок.

Факт: У 2024 році до 30% складів у світі будуть використовувати автоматизовані рішення на основі ШІ.

Розвиток Автономного Транспорту

ШІ лежить в основі автономних транспортних засобів, які вже тестуються у сфері логістики. Вантажівки-безпілотники, дрони для доставки та роботизовані візки можуть значно зменшити витрати на перевезення.

Тренд: Компанії, такі як Tesla і Amazon, інвестують у розробку автономних рішень для логістики.

Покращення Обслуговування Клієнтів

Чат-боти на базі ШІ допомагають клієнтам швидко отримати інформацію про доставку, вирішити спірні питання та запропонувати персоналізовані рішення. Це значно підвищує задоволеність клієнтів.

Цікаво знати: До 2025 року 85% взаємодій з клієнтами в логістиці будуть автоматизовані.

Переваги Використання ШІ у Логістиці

1. Економія ресурсів: Зниження витрат на транспорт, складування та обслуговування.

2. Прозорість: Відстеження кожного етапу доставки в режимі реального часу.

3. Точність: Мінімізація людських помилок у плануванні та виконанні логістичних операцій.

4. Прогнозування: Покращене управління ризиками та підготовка до сезонних змін.

1.3.2 Управління ланцюгами постачання

Завдяки ШІ компанії можуть прогнозувати попит, оптимізувати запаси та підвищувати стійкість ланцюгів постачання. Ранні впровадження ШІ в управлінні ланцюгами постачання призвели до зниження логістичних витрат на 15%, покращення рівня запасів на 35% та підвищення рівня обслуговування на 65%.

Штучний інтелект (ШІ) як термін з'явився у 1950-х роках, але широкого визнання отримав після запуску ChatGPT у 2022 році, який за два місяці набрав

понад 100 мільйонів користувачів. Термін "управління ланцюгами постачання", який виник у 1980-х, теж привернув увагу лише під час пандемії COVID-19, коли виникли серйозні дефіцити товарів. Сьогодні багато компаній використовують ШІ для управління глобальними ланцюгами постачання, щоб підвищити їхню стійкість і ефективність.

Адміністрація Байдена зосередилась на двох важливих напрямах: розвиток ШІ та підвищення стійкості ланцюгів постачання. У 2023 році Байден підписав два укази: один щодо безпеки ШІ, інший — про зміцнення ланцюгів постачання. Також було створено Раду Білого дому з питань стійкості ланцюгів постачання для моніторингу ризиків і реагування на проблеми. Європейський Союз, у свою чергу, ухвалив закон про ШІ, що регулює відповідальність усіх учасників ланцюгів постачання з високим ризиком.

ШІ має величезний потенціал для революціонізації ланцюгів постачання. Він покращує управління виробництвом, запасами та дистрибуцією товарів, забезпечуючи точніше прогнозування попиту. Завдяки цьому компанії можуть знижувати витрати на логістику, оптимізувати запаси та підвищувати якість послуг. Інструменти ШІ також сприяють видимості в ланцюгах постачання, що дозволяє швидше реагувати на збої та покращувати координацію між партнерами.

ШІ допомагає компаніям відстежувати зміну попиту і пропозиції, аналізувати дані про трафік у портах і на складах, а також оцінювати ефективність стратегій реагування на збої. Він може використовувати симуляції для розробки оптимальних рішень під час криз, а також рекомендувати довгострокові зміни в політиці ланцюгів постачання, враховуючи макроекономічні тенденції.

Попри страхи про втрату робочих місць, ШІ в ланцюгах постачання, навпаки, створює нові можливості. Люди залишаються ключовими у процесі аналізу контексту та прийняття рішень, що робить професії, пов’язані з етикою ШІ та його управлінням, дедалі важливішими. Уряди США та ЄС повинні координувати свої зусилля для етичного розвитку ШІ, аби уникнути упереджень у даних і зменшити ризики.

Важливість ШІ у ланцюгах постачання підкреслюється його здатністю сприяти економічній стабільності та інноваціям. Хоча побоювання щодо автоматизації робочих місць існують, у сфері управління ланцюгами постачання ШІ слугує інструментом для підвищення стійкості, а не заміною людей.

1.3.3 Автоматизація складів

Використання робототехніки та ШІ в складах підвищує ефективність обробки товарів, зменшує помилки та прискорює виконання замовлень.

Нові інновації в логістиці на основі ШІ

1. Розумне складування з глибоким навчанням: розширені моделі ШІ тепер використовуються для прогнозування рівня запасів, тим самим оптимізуючі простір для зберігання. Завдяки використанню глибокого навчання, склади можуть прогнозувати потреби в запасах на місяці наперед, враховуючи сезонність і ринкові тенденції.

2. Автономні транспортні засоби: завдяки швидкому прогресу технології автономного керування постійно вдосконалюються. Тож можна припустити, що день, коли автономні безпілотні літальні апарати та вантажівки здійснюватимуть доставку, зменшуючи при цьому втручання людини та підвищуючи ефективність, вже не за горами.

3. Блокчейн і штучний інтелект для автентичності ланцюга поставок: об’єднання блокчейну зі штучним інтелектом забезпечує захист ланцюга поставок від несанкціонованого втручання. Штучний інтелект відстежує та перевіряє автентичність кожного продукту, що забезпечує надійність і зменшує кількість підробок.

4. Логістика з доповненою реальністю (Augmented Reality - AR): її можна об’єднати зі штучним інтелектом, щоб надавати інформацію в режимі реального часу наземному персоналу на складах або персоналу доставки, підвищуючи тим самим швидкість комунікацій, ефективність взаємодії і зменшуючи помилки.

5. Екологічна логістика за допомогою штучного інтелекту. Аналізуючи величезні масиви даних, ШІ може запропонувати більш екологічні методи

пакування та маршрути, які зменшують викиди вуглецю, і навіть запропонувати стійкі альтернативи для логістичних операцій.

1.4 Висновок до розділу

У розділі "Огляд сучасного стану оптимізації логістики перевезень" детально розглянуто ключові аспекти організації логістичних процесів та можливі шляхи їхнього вдосконалення. Проведений аналіз проблем в організації логістики перевезень показав, що основними викликами залишаються високий рівень невизначеності у плануванні, значні витрати на транспортні операції, а також складність узгодження численних факторів, які впливають на ефективність перевезень. Традиційні методи оптимізації, такі як математичне моделювання та використання евристичних алгоритмів, незважаючи на їхню ефективність у ряді випадків, мають суттєві обмеження. Зокрема, вони виявляються недостатньо швидкими для обробки великих обсягів даних та не завжди здатні адаптуватися до динамічних умов сучасного ринку.

У контексті використання технологій штучного інтелекту акцентовано увагу на значних перспективах його впровадження у сферу логістики. Штучний інтелект надає змогу забезпечити не лише гнучкість і автоматизацію процесів, але й ефективну адаптацію до змінних умов середовища. Це відкриває нові горизонти для вирішення складних логістичних задач. Особливий інтерес представляють такі напрямки, як машинне навчання, аналіз великих даних, а також розробка систем підтримки прийняття рішень, які допомагають операторам приймати оптимальні рішення у реальному часі.

Таким чином, інтеграція традиційних методів із сучасними засобами штучного інтелекту є важливим кроком до подальшого розвитку логістичних систем. Такий підхід дозволяє суттєво підвищити їхню ефективність, зменшити витрати, а також забезпечити вищий рівень надійності та адаптивності.

РОЗДІЛ 2

СТВОРЕННЯ БОТА ДЛЯ ВЗАЄМОДІЇ З ЛОГІСТАМИ

2.1 Аналіз сайтів на доступність інформації

2.1.1 Проблема доступності інформації на сучасних веб-сайтах

Сучасні веб-сайти відіграють важливу роль у логістиці, забезпечуючи доступ до інформації про вантажі, маршрути, замовлення та інші ключові дані. Однак, з розвитком технологій виникає проблема доступності цієї інформації для автоматизованого збору та аналізу. Багато платформ впроваджують спеціальні механізми захисту, відомі як анти-парсери, які обмежують можливості автоматичного збору даних. Це значно ускладнює створення інструментів, здатних автоматизувати рутинні завдання та аналіз даних у реальному часі.

У сучасному цифровому світі, де Інтернет стає невід'ємною частиною нашого життя, питання захисту особистої інформації та конфіденційності набуває особливої актуальності. Браузери надають нам доступ до величезної кількості інформації, але водночас можуть становити загрозу для нашої приватності.

Однією з ключових проблем є збір даних про користувачів. Браузери часто фіксують історію переглядів, місцезнаходження та іншу персональну інформацію. Ці дані можуть потрапляти до сторонніх осіб для рекламних або навіть шахрайських цілей, що створює значні ризики для безпеки користувачів.

Щоб забезпечити захист особистих даних і конфіденційність у цифровому середовищі, необхідно вирішувати ці проблеми комплексно, поєднуючи технологічні засоби з обізнаністю користувачів. Серед ефективних методів варто виділити використання блокувальників реклами та відстежувачів, які дозволяють знизити ризики збору даних про поведінку в інтернеті. Ці інструменти допомагають уникнути небажаного впливу цільової реклами, що базується на аналізі ваших дій в мережі.

2.1.2 Конфіденційність у сучасних браузерах

Питання конфіденційності у браузерах набирає глобального масштабу. Загрози, пов'язані з порушенням приватності, впливають на користувачів у всьому світі, включно з українськими користувачами.

Хоча сучасні браузери пропонують численні корисні функції, вони часто стають джерелом конфіденційних ризиків. Більшість користувачів не усвідомлює, що браузери можуть збирати та обробляти їхні дані без прямої згоди.

Основна проблема — це відстеження активності користувачів. За допомогою кукі, піксельних тегів і скриптів браузери збирають дані про дії користувачів в Інтернеті. Це може призвести до розголошення особистої інформації або її несанкціонованого використання.

Крім того, значною загрозою є витоки даних. Хакери та зловмисники можуть отримати доступ до збережених даних користувачів, особливо якщо вони зберігаються на сервері або у хмарних сховищах. Це створює ризик фінансових втрат, викрадення персональної інформації та інших конфіденційних даних.

Однак існують ефективні способи захисту від таких загроз, які можуть значно зменшити ризики витоку особистої інформації. Користувачам рекомендується регулярно оновлювати браузери до останніх версій, оскільки розробники постійно виправляють вразливості та впроваджують нові механізми безпеки. Очищення файлів кукі та тимчасових даних, які можуть використовуватись для відстеження, є необхідним заходом для збереження конфіденційності.

Захист особистої інформації в браузерах є однією з ключових проблем сучасного інтернет-користувача. Необхідно усвідомлено підходити до використання технологій і впроваджувати заходи безпеки, щоб мінімізувати ризики зловживання даними.

2.1.3 Використання Playwright як рішення

Під час розробки інструментів для автоматизації логістичних процесів я зіткнувся з проблемою анти-парсерів на сайтах, зокрема на платформі Central Dispatch. Використання стандартних методів, таких як бібліотека requests або

BeautifulSoup, виявилося недостатнім через блокування запитів. Виходом стало впровадження Playwright — інструмента для автоматизації роботи браузера, який забезпечує повноцінну емуляцію користувацької взаємодії з сайтом.

Playwright дозволяє обійти анти-парсери завдяки можливостям рендерингу JavaScript, імітації поведінки користувача, таких як кліки, скролінг або введення даних у форми. Цей підхід виявився ефективним для отримання доступу до захищених даних, що раніше були недоступні для автоматизованих рішень.

Насамперед Playwright - це інструмент для комплексного тестування сучасних веб-додатків, але, як я розумію, зібралися ми тут для іншого.

Сам інструмент багатомовний і приблизно однаково працює на Python, Node.js і Java. Тому, якщо ви не Python-програміст, ця інформація вам теж буде цікава. Однак, оскільки моя основна мова програмування Python, я розглядатиму версію для Python.

Playwright підтримує такі браузери: Chromium (сьогодні покажу вам, як запустити Chrome), WebKit і Firefox. До речі, інструмент ще й мультиплатформний, тож ви зможете його вільно запустити на Windows, Linux і macOS.

2.1.4 Анти-парсери та їхній вплив на автоматизацію

Анти-парсери є одним із ключових бар'єрів для розробників, які прагнуть автоматизувати доступ до даних. Вони реалізуються у вигляді:

- Використання CAPTCHA для перевірки активності користувача.
- Динамічної зміни структури HTML-коду.
- Обмежень за IP-адресами чи геолокацією.
- Відстеження активності через JavaScript та cookies.

Такі механізми створюють складнощі навіть для досвідчених розробників, вимагаючи використання додаткових технологій для обробки даних.

2.1.5 Переваги та недоліки використання Playwright

Попри очевидні переваги Playwright, такі як ефективність у боротьбі з анти-парсерами та висока гнучкість, існують і певні обмеження. Основними недоліками є

висока ресурсомісткість та необхідність ретельного налаштування для кожного конкретного сайту. Проте, враховуючи результати, досягнуті у процесі роботи, ці недоліки не є критичними.

2.2 Написання парсеру для зчитування даних з сайтів

2.2.1. Вибір інструментів для збору даних

Створення парсеру для зчитування даних з вебсайтів є важливою складовою розробки системи автоматизації для логістичних перевезень. Вебсайт Central Dispatch містить величезну кількість даних про лоуди, що виставлені на платформі, і для ефективного збору цієї інформації необхідно використовувати спеціалізовані інструменти та методи. На початковому етапі виникали різноманітні труднощі, пов'язані з обмеженнями сайту, проблемами авторизації та необхідністю оптимізувати процес збору даних.

Процес збору даних з вебсайтів, таких як Central Dispatch, зазвичай вимагає використання специфічних інструментів для автоматизації. Першим етапом було визначення підходящої технології, яка б дозволила ефективно зчитувати дані без зайвих затримок. Я розглядав кілька варіантів, серед яких були:

Requests — простий HTTP-клієнт для роботи з веб-запитами.

BeautifulSoup — бібліотека для парсингу HTML-коду, яка дозволяє отримувати дані з вебсторінок.

Playwright — потужний інструмент для автоматизації браузера, який дозволяє контролювати веб-браузер та взаємодіяти з вебсторінками на рівні користувача.

Оскільки сайт Central Dispatch використовує складні взаємодії між елементами, постійно оновлюючи контент за допомогою AJAX-запитів (XHR), рішенням стало використання Playwright.

2.2.2. Використання POST-запитів для отримання даних

Однією з ключових складнощів при зборі даних з Central Dispatch було формування правильних POST-запитів для отримання інформації. Для цього

потрібно було з'ясувати, як сайт отримує дані про лоуди, які відображаються на сторінці.

Щоб знайти правильні запити, я скористався інструментами для розробників в браузері. Відкривши вкладку "Network" і вибравши підвкладку "XHR", я зміг відстежити, які запити надсилаються під час завантаження сторінки. Як з'ясувалося, сайт відправляє асинхронні запити до серверу для отримання лоудів. Ці запити надходять у форматі JSON, що містить усю необхідну інформацію про лоуди, яку я міг використовувати для подальшого аналізу.

На основі цієї інформації я побудував POST-запит, який передавав необхідні параметри для отримання даних. Зокрема, запит містив параметри фільтрації, такі як географічне положення, тип вантажу, транспортні засоби, ціна за милю та інші параметри, що визначають вимоги до лоудів.

Один із складних моментів полягає в тому, що на сайті не завжди відображаються всі лоуди на одній сторінці. Оскільки кількість лоудів може досягати десятків тисяч, важливо було знайти спосіб зчитування даних за кілька запитів, замість того щоб перевантажувати сервер та отримувати дані поодинці.

2.2.3 Явне використання HTTP-методів

Однією з ключових характеристик Web-сервісу RESTful є явне використання HTTP-методів згідно з протоколом, означеним в RFC 2616. Наприклад, HTTP GET означений як метод генерування даних, використовуваний клієнтським додатком для вилучення ресурсу, отримання даних з Web-сервера або виконання запиту в надії на те, що Web-сервер знайде і поверне набір відповідних ресурсів.

REST пропонує розробникам використовувати HTTP-методи явно відповідно до означення протоколу. Цей основний принцип проектування REST встановлює однозначну відповідність між операціями create, read, update і delete (CRUD) і HTTP-методами. Згідно з цим відповідності:

- Для створення ресурсу на сервері використовується POST.
- Для отримання ресурсу використовується GET.
- Для зміни стану ресурсу або його поновлення використовується PUT.

- Для видалення ресурсу використовується DELETE.

Недоліком проектування багатьох Web API є використання HTTP-методів не за прямим призначенням. Наприклад, URI запиту в HTTP GET зазвичай означує один конкретний ресурс, або ж рядок запиту в URI запиту містить ряд параметрів, що означують критерії пошуку сервером набору відповідних ресурсів. Принаймні саме так описаний метод GET в HTTP/1.1 RFC. Однак часто зустрічаються неприваблівіі Web API, що використовують HTTP GET для виконання різного роду транзакцій на сервері (наприклад, для додавання записів в базу даних). У таких випадках URI запиту GET використовується некоректно або, принаймні, не використовується в REST-стилі (RESTfully).

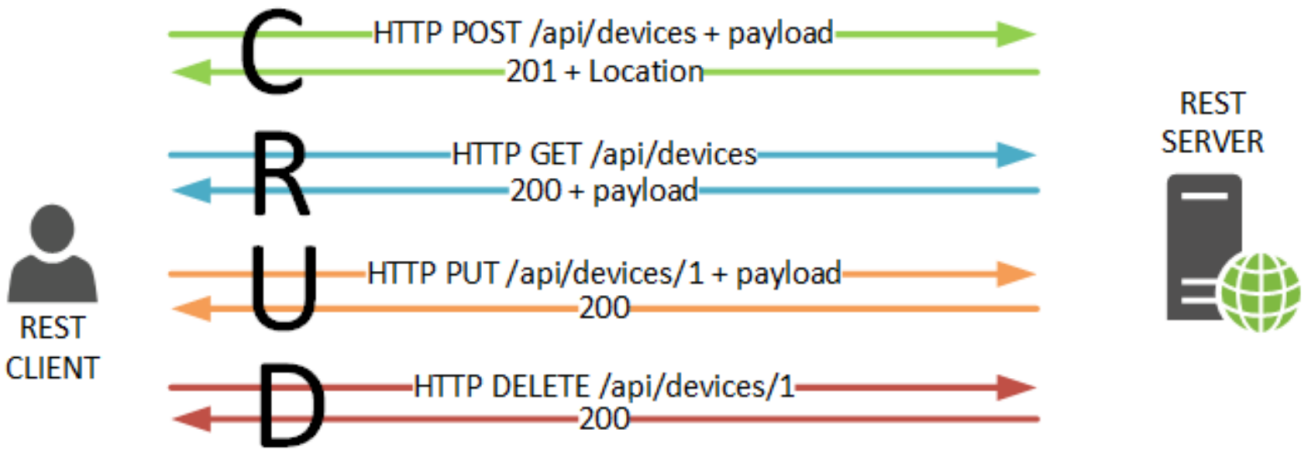

Рис. 2.1. Використання HTTP-методів

2.2.4 Зчитування даних за допомогою Fetch XHR

Оскільки сайт використовує технологію XHR для завантаження даних, мені вдалося знайти необхідні запити, що дозволяють отримати всю інформацію про лоуди без необхідності постійно взаємодіяти з вебсторінкою. Однак, за замовчуванням сайт обмежує кількість лоудів на сторінці, що ускладнювало зчитування великої кількості даних.

Тому, щоб зчитувати великі обсяги даних, я змінив налаштування в браузері, збільшивши кількість лоудів, які відображаються на одній сторінці, до 250. Це дозволило отримати більше даних з одного запиту, зменшуючи час на обробку.

Завдяки цьому підходу я зміг зібрати великі обсяги даних, що стосуються лоудів, за кілька запитів. Наприклад, при збільшенні кількості лоудів на сторінці до 250, я міг за один запит отримати до 250 лоудів, що зменшувало необхідність у великих циклах запитів.

Цей підхід дозволив суттєво прискорити процес збору даних і знизити навантаження на сервер. Завдяки зчитуванню даних в режимі асинхронних запитів через XHR я зміг отримувати всю необхідну інформацію про лоуди у форматі JSON і далі використовувати її для аналізу.

2.2.5 Проблеми з авторизацією

Однією з найбільших проблем, з якими я стикався при роботі з сайтом, була авторизація. Сайт Central Dispatch для нових користувачів вимагає підтвердження через код, який надсилається на електронну пошту. Це створювало труднощі для автоматизації, оскільки код підтвердження потрібно було вводити вручну при кожному сеансі.

Щоб вирішити цю проблему, я вирішив скористатися своїм основним акаунтом у браузері Google Chrome. Я вручну авторизувався на сайті, після чого отримав кукі-файли, які використовуються для збереження сесії. Ці кукі я зміг експортувати за допомогою Python, використовуючи бібліотеку для роботи з браузером.

Далі я передав ці кукі до Playwright, щоб створити автоматизовану сесію. Завдяки такому підходу я міг обійти необхідність введення коду підтвердження, оскільки система автоматично визнавала мою авторизовану сесію через кукі.

Цей метод дозволив здійснити безперешкодний доступ до сайту без необхідності постійно вводити код підтвердження, що суттєво прискорило процес збору даних.

2.2.6 Результати та подальші кроки

Розробка парсеру для зчитування даних з Central Dispatch дозволила значно оптимізувати процес збору інформації про лоуди. Завдяки використанню POST-

запитів, аналізу XHR-запитів та автоматизації через Playwright, вдалося створити ефективний інструмент для збору десятків тисяч лоудів за кілька запитів.

В подальшому планується удосконалити парсер для додаткової фільтрації лоудів за різними параметрами, а також оптимізувати процес обробки та збереження даних для подальшого аналізу. Це дозволить створити систему, яка автоматично адаптується до змін на сайті та забезпечує постійне оновлення інформації для логістичних компаній.

2.3 Створення бота для взаємодії з логістами

2.3.1. Спілкування з логістами та вивчення потреб

На етапі планування системи для аналізу лоудів ми зіштовхнулися з необхідністю розробити ефективний інструмент для взаємодії з логістами. Метою було створення платформи, яка дозволить швидко отримувати інформацію про наявні лоуди, а також сповіщати про нові можливості у реальному часі. Спочатку ми розглядали кілька варіантів, включаючи розробку окремої програми для цього завдання. Однак, після консультацій з представниками логістичних компаній та врахуванням їхніх потреб, ми зупинилися на створенні бота через платформу Telegram.

На початку ми провели ряд зустрічей і обговорень з логістами для з'ясування, яким чином вони звикли працювати з інформацією та які функції для них є найважливішими. Логісти, які працюють з лоудами, зазвичай мають великий обсяг інформації та велику кількість заявок, тому важливо було забезпечити їм можливість отримувати актуальні дані якомога швидше.

Одним із ключових аспектів, який ми визначили на цих зустрічах, була простота доступу до інформації. Логісти працюють на виїзді, в умовах частих поїздок та змін локації, тому важливо було, щоб інформація про нові лоуди була доступною в зручному форматі, який не вимагав зайвих дій чи тривалого часу на аналіз.

2.3.2. Оцінка різних варіантів для взаємодії з користувачами

Після збору всіх вимог, ми почали оцінювати варіанти платформ, через які можна було б організувати сповіщення про нові лоуди та їх пошук. Розглядали ми кілька можливостей:

Розробка окремої програми — ми розглядали варіант створення повноцінного додатку для мобільних телефонів або комп'ютерів. Однак, цей варіант мав низку недоліків:

Високі витрати на розробку та підтримку — створення та підтримка окремого додатку потребує значних ресурсів, що не завжди є виправданим, особливо якщо більшість користувачів використовують вже існуючі мобільні платформи.

Необхідність установки — для використання програми логістам потрібно було б завантажити та встановити нову програму. Це додатковий крок, який не завжди є зручним, особливо якщо логісти працюють на різних пристроях.

Інтеграція через Email — ми також думали про використання електронної пошти для сповіщень про нові лоуди. Проте, цей варіант був менш ефективним:

Затримки у доставці повідомлень — при великій кількості сповіщень існувала ймовірність затримок у доставці, що могло б зашкодити швидкості реакції на нові лоуди.

Можливість пропуску повідомлень — електронна пошта не завжди забезпечує миттєве сприйняття інформації, і є ризик, що логісти можуть пропустити важливі лоуди, якщо не перевірятимуть пошту регулярно.

Використання месенджерів (Telegram, WhatsApp) — одним з найбільш популярних варіантів, який ми розглядали, були месенджери, зокрема Telegram і WhatsApp. Обидва ці месенджери мають свої переваги, але після обговорення з логістами ми вирішили зупинитись на Telegram з кількох ключових причин:

Миттєві сповіщення — у Telegram є можливість отримувати миттєві push-сповіщення на телефон, що дає змогу логістам швидко реагувати на нові лоуди.

Простота використання — Telegram є популярним месенджером серед людей різних вікових категорій та професій, тому логістам не потрібно навчатися користуватися новими додатками чи програмами.

Можливість інтеграції з ботами — Telegram має потужне API для створення ботів, що дозволяє автоматизувати більшість процесів, таких як фільтрація лоудів, пошук за критеріями, і відправка сповіщень.

2.3.3. Чому Telegram став оптимальним вибором

Після ретельного аналізу та обговорень з логістами, ми дійшли до висновку, що Telegram є оптимальним вибором для нашого бота. Ось кілька основних причин:

Мобільність і доступність — Telegram доступний на всіх платформах і не потребує додаткової установки чи налаштувань, що робить його дуже зручним для логістів, які часто перебувають в дорозі.

Миттєвість сповіщень — завдяки можливості отримувати сповіщення про нові лоуди безпосередньо на телефон, логісти можуть оперативно реагувати на нові можливості. Це є дуже важливим, оскільки кожен лоуд має обмежений час для того, щоб його забрали.

Інтерактивність — Telegram дозволяє створювати інтерактивні кнопки, що полегшує взаємодію з ботом. Логісти можуть швидко фільтрувати лоуди, отримувати детальну інформацію про них та миттєво приймати рішення.

Масштабованість — Telegram дозволяє обробляти великий обсяг повідомлень одночасно, що робить його ідеальним для великої кількості користувачів, яких ми могли б залучити до системи.

Безпека та зручність — Telegram є безпечним для передачі даних і дозволяє легко налаштувати додаткові заходи безпеки, такі як двофакторна автентифікація.

2.3.4. Рішення та подальші кроки

Після всіх обговорень з логістами та аналізу можливих варіантів, ми прийняли рішення створювати Telegram бота для взаємодії з користувачами. Це дозволить нам забезпечити логістам швидкий доступ до даних про лоуди та оперативні сповіщення в реальному часі.

У наступному етапі ми приступили до безпосередньої розробки бота, інтеграції з платформами для збору даних про лоуди, а також налаштування функціоналу для сповіщень і пошуку лоудів.

2.4 Створення інструментів взаємодії з ботом

2.4.1. Бібліотеку telebot

Для початку потрібно вибрати технологію яка буде основною при використання бота, вибір вправ та telebot.

Бібліотека telebot є однією з найпопулярніших для створення ботів у Telegram за допомогою Python. Вона забезпечує простий інтерфейс для взаємодії з API Telegram, що дозволяє розробникам створювати різноманітні функціональні боти без необхідності глибоких знань про Telegram API.

Основною перевагою telebot є її простота у використанні. Бібліотека підтримує основні функції для створення ботів, такі як обробка повідомлень, команд, inline-кнопок і багато іншого. Вона дозволяє налаштовувати бота для реагування на текстові повідомлення, картки, файли, фотографії та інші медіафайли. Бот може обробляти повідомлення з використанням функцій, які можуть бути зареєстровані для певних команд або запитів.

Однією з ключових особливостей бібліотеки є підтримка обробників різних типів запитів і можливість працювати з асинхронними викликами. Це дозволяє створювати боти, які можуть обробляти запити та виконувати дії паралельно, що особливо важливо для створення масштабованих і ефективних рішень. Крім того, бібліотека підтримує Webhook для отримання повідомлень у реальному часі, що є зручним для створення бота з миттєвою реакцією на події.

Інтеграція з Telegram API через telebot є безшовною, що дозволяє легко додавати нові функціональності без необхідності створювати складні механізми для обробки запитів. Наприклад, можна легко додавати команди для запуску різних дій, налаштовувати клавіатури для зручної взаємодії з користувачем, обробляти посилання на URL та багато інших можливостей.

Завдяки такій доступності та багатофункціональності бібліотека telebot підходить як для простих ботів, так і для складних проектів, де важлива інтеграція з іншими сервісами чи базами даних. Це робить її відмінним інструментом для розробників, які хочуть швидко створити ботів без необхідності занурюватися в деталі Telegram API.

Однією з ключових задач при розробці системи для логістичних компаній було створення інструментів для взаємодії користувачів з ботом, а також розробка зручного інтерфейсу для взаємодії з платформою. В цьому розділі описано як ми вирішували проблему інтеграції веб-сайту з Telegram ботом через web app, а також розробку самого бота, зосереджуючись на основних викликах, що виникли на цьому етапі.

Основні переваги бібліотеки telebot:

Простота використання: Бібліотека має зрозумілий і зручний синтаксис, що дозволяє розробникам швидко створювати функціональні боти. Для початку достатньо кількох рядків коду, щоб налаштувати основні функції бота.

Підтримка основних функцій Telegram API: telebot підтримує усі стандартні функції для взаємодії з користувачами, такі як надсилання текстових повідомлень, медіафайлів, обробка inline-кнопок, callback-запитів і багато іншого.

Асинхронність: За допомогою асинхронних викликів бібліотека дозволяє розробляти боти, які можуть обробляти багато запитів одночасно, що підвищує їх ефективність і масштабованість.

Підтримка Webhook: Для створення ботів, які мають миттєву реакцію на події, telebot дозволяє налаштувати Webhook, що дозволяє боту отримувати запити в реальному часі.

Легкість інтеграції: Бібліотека легко інтегрується з іншими системами, такими як бази даних, API, зовнішні сервіси тощо, що дозволяє створювати складні рішення для бізнес-логіки бота, забезпечуючи при цьому гнучкість, масштабованість розробки та можливість швидко адаптуватися до змін у вимогах чи підключати нові функціональні модулі без значних зусиль.

Порівняння з іншими бібліотеками

aiogram:

- Асинхронність: aiogram — це асинхронна бібліотека, що дає перевагу у швидкості обробки запитів, порівняно з telebot, яка за замовчуванням є синхронною. Це може бути важливо при розробці ботів з високими вимогами до продуктивності.
- Складність: aiogram має більш складну структуру, оскільки вимагає розуміння асинхронного програмування, що може бути складно для початківців, на відміну від telebot, який є більш простим у використанні.
- Продуктивність: Завдяки асинхронному підходу, aiogram працює швидше при обробці великої кількості запитів, що робить його кращим вибором для великих ботів.

python-telegram-bot:

- Зручність: python-telegram-bot — це ще одна популярна бібліотека для створення Telegram ботів. Вона також має простий інтерфейс, схожий на telebot, але пропонує більше можливостей для налаштування обробників і взаємодії з Telegram API.
- Налаштування: python-telegram-bot має більшу гнучкість у налаштуванні та взаємодії з API, що робить її більш потужною для створення складних ботів, але також вимагає більше часу на освоєння.
- Інтеграція: Бібліотека підтримує як синхронні, так і асинхронні обробники, що дає змогу адаптувати її до різних умов.

telepot:

- Мінімалізм: telepot є ще однією бібліотекою для створення ботів, але вона більш мінімалістична і менш популярна, ніж telebot. Вона пропонує основні функції для створення ботів, але не має такої широкої підтримки та документації, як telebot.
- Підтримка API: telepot не підтримує всі можливості Telegram API, які є у telebot та python-telegram-bot, що може обмежити можливості при розробці складних ботів.

Переваги використання telebot:

- Швидкість розробки: Завдяки зручному і простому синтаксису, telebot дозволяє швидко розпочати розробку бота, не вимагаючи глибоких знань про Telegram API.
- Підтримка основних функцій: Бібліотека забезпечує повну підтримку всіх базових функцій Telegram API, що дозволяє створювати бота для більшості стандартних сценаріїв.
- Розширення: Для складніших проектів можна використовувати telebot разом з іншими бібліотеками та інструментами для розширення функціональності бота (наприклад, для роботи з базами даних або зовнішніми API).

Недоліки:

- Обмежена асинхронність: Бібліотека telebot за замовчуванням не асинхронна, тому в деяких випадках вона може бути менш ефективною для обробки великої кількості запитів порівняно з асинхронними бібліотеками, такими як aiogram.
- Меньше можливостей налаштування: У порівнянні з іншими бібліотеками, такими як python-telegram-bot, telebot може мати менше гнучкості у налаштуванні функціоналу бота.

В цілому, бібліотека telebot є відмінним вибором для більшості користувачів, які хочуть швидко створити Telegram бота з мінімумом зусиль. Вона пропонує зручний інтерфейс і підходить для реалізації стандартних функцій бота. Для більш складних або продуктивних проектів можна розглянути інші бібліотеки, такі як aiogram або python-telegram-bot.

2.4.2 Розробка сайту та інтеграція з Telegram через web app

Одним з перших викликів стало створення веб-сайту, який би дозволяв взаємодіяти з ботом і доповнював його функціонал. Веб-сайт мав бути не тільки

основним інтерфейсом для користувачів, але й засобом для налаштування фільтрів та отримання більш детальної інформації про лоуди.

Проблеми при розробці веб-сайту для Telegram бота

Інтеграція з web app в Telegram: Однією з основних проблем, з якими ми стикалися, була інтеграція веб-сайту з web app у Telegram. У Telegram є можливість підключати веб-додатки до бота, що дозволяє користувачам взаємодіяти з ботом через сайт, зберігаючи при цьому зв'язок з ботом. Однак для коректної роботи такої інтеграції потрібна була специфічна настройка API Telegram, що вимагало ретельного тестування та адаптації сайту під вимоги месенджера. Особливо це стосувалося відправки даних з сайту на сервер Telegram через web app, а також обробки отриманих відповідей.

Адаптація сайту під мобільні пристрої: Бот був орієнтований на користувачів, які в основному працюють з мобільними телефонами, тому веб-сайт повинен був бути адаптованим під мобільні пристрої. Це потребувало особливого підходу до дизайну, щоб забезпечити зручну навігацію на маленьких екранах. Також потрібно було забезпечити функціональність, що дозволяє користувачам швидко налаштовувати фільтри та отримувати потрібну інформацію без зайвих складнощів.

Взаємодія з Telegram через кнопки та API: Веб-сайт також мав бути здатним надавати користувачам можливість налаштовувати параметри фільтрації лоудів і автоматично передавати їх через бота в Telegram. Це включало створення кнопок на сайті для налаштування різних параметрів фільтрів і передачу цих налаштувань у вигляді запитів до бота. Підключення цих кнопок до API Telegram потребувало значного часу на розробку та тестування, оскільки ми використовували стандартні інструменти та бібліотеки для інтеграції з Telegram API, такі як telebot.

Безпека та автентифікація: Ще однією проблемою була необхідність забезпечення безпеки при передачі чутливих даних між сайтом і ботом, таких як налаштування фільтрів і персональна інформація. Для цього було вирішено використовувати безпечний протокол передачі даних (HTTPS), а також впровадити автентифікацію через бота, щоб користувачі могли підтверджувати свої дії та доступ до певних функцій тільки після авторизації через Telegram.

Переваги розробки сайту для бота

1. Гнучкість налаштувань: Веб-сайт надавав користувачам велику гнучкість у налаштуванні фільтрів для лоудів. Це дозволяло логістам не лише отримувати сповіщення в боті, але й мати можливість коригувати фільтри та умови пошуку, не відходячи від зручного веб-інтерфейсу.

2. Розширення можливостей бота: Через інтеграцію з web app на сайті, ми змогли значно розширити можливості бота, забезпечивши користувачів не тільки сповіщеннями, а й детальною інформацією про лоуди, їх параметри та умови для перевезення. Це дозволяло користувачам отримувати всі необхідні дані в одному місці, не переходячи між різними програмами або платформами.

3. Простота налаштування фільтрів: Використання веб-сайту дозволило зробити процес налаштування фільтрів для лоудів значно простішим. Користувач міг відразу побачити на екрані всі доступні параметри фільтрації і змінювати їх через інтерфейс сайту. Бот отримував ці налаштування і автоматично підлаштовував сповіщення під задані критерії.

2.4.3 Розробка бота для Telegram

Створення бота для Telegram стало важливим етапом у розробці інструментів взаємодії з користувачами. Бот дозволяв автоматизувати процес сповіщення про нові лоуди та надавати логістам необхідну інформацію за допомогою зручного і простого інтерфейсу.

Проблеми при розробці бота

1. Обмеження Telegram API: Однією з головних проблем при розробці бота було обмеження, які накладає Telegram на кількість запитів та сповіщень, що можуть бути надіслані користувачам за одиницю часу. Для цього ми використовували оптимізацію через черги повідомлень, щоб уникнути перевантаження і забезпечити коректну роботу бота при великій кількості одночасних користувачів.

2. Налаштування автоматичних сповіщень: Оскільки система мала працювати в реальному часі, ми мали розробити складну систему сповіщень, яка могла б

надсилати користувачам актуальні дані з максимальною швидкістю. Це вимагало налаштування правильних таймерів та оптимізації роботи з API для обробки запитів без затримок.

3. Взаємодія з великою кількістю користувачів: Telegram бот повинен був працювати з великою кількістю користувачів одночасно, тому потрібно було вирішити питання масштабування. Для цього ми налаштували розподілену архітектуру бота та використовували технології для оптимізації обробки повідомлень.

Позитивні сторони розробки бота

1. Миттєвість сповіщень: Бот дозволяв миттєво інформувати логістів про нові лоуди, що значно покращило їх оперативність. Завдяки інтеграції з Telegram, користувачі могли отримувати актуальні повідомлення без затримок на мобільні пристрої.

2. Інтерактивність та зручність: Telegram надає багатий набір функцій для взаємодії з користувачами, таких як кнопки, меню та каруселі, що дозволило зробити взаємодію з ботом максимально зручною та інтуїтивною.

3. Автоматизація та масштабованість: Завдяки можливості автоматизувати процеси через бота, ми змогли забезпечити швидке та ефективне надходження даних до користувачів, навіть коли йшлося про велику кількість одночасних запитів.

2.5 Створення документації для бота

2.5.1. Мета документації

Документація для бота є ключовим елементом у процесі забезпечення його ефективного використання, особливо коли мова йде про взаємодію з користувачами, які можуть не мати досвіду роботи з автоматизованими системами або чат-ботами. Задля цього була розроблена детальна документація, яка допомагає користувачам швидко освоїтися в роботі з ботом, отримати необхідну інформацію та налаштувати фільтри для отримання сповіщень.

Основною метою створення документації було забезпечення користувачів інтуїтивно зрозумілим посібником по роботі з ботом.

Це дозволяло:

1. Швидке ознайомлення з функціоналом бота та його можливостями.
2. Легкість у налаштуванні параметрів фільтрації та отриманні персоналізованих сповіщень.
3. Надання чітких інструкцій щодо того, як правильно взаємодіяти з ботом для отримання потрібних даних.

2.5.2. Структура документації

Документація була побудована таким чином, щоб користувачі могли швидко знаходити потрібну інформацію, а також отримувати чіткі інструкції для виконання дій. Структура документації складається з наступних розділів:

Вступ та опис бота:

1. Опис основної мети бота: автоматичне отримання сповіщень про нові лоуди.
2. Пояснення, як бот може допомогти логістам та транспортним компаніям, автоматизуючи пошук і повідомлення про відповідні лоуди.
3. Системні вимоги та основні функції бота.

Як почати роботу з ботом:

1. Покрокова інструкція для початку використання бота: як знайти його в Telegram, як розпочати взаємодію.
2. Опис процесу реєстрації та авторизації через Telegram, включаючи варіанти налаштувань та підтвердження через бота.

Налаштування фільтрів:

1. Пояснення, як налаштувати фільтри для отримання актуальних лоудів: вибір типу вантажу, географічних зон, дат доставки тощо.
2. Покроковий опис, як користувач може встановити свої уподобання через інтерфейс бота або веб-сайт.

3. Опис функцій "збереження фільтрів" для автоматичного застосування в подальшому.

Взаємодія з ботом:

1. Опис основних команд бота, включаючи команди для налаштування фільтрів, перегляду доступних лоудів, налаштування отримання сповіщень.

2. Інтерактивні елементи, такі як кнопки, які дозволяють користувачу легко вибирати параметри та налаштовувати отримання даних.

3. Детальний опис роботи з повідомленнями та функціями кнопок у боті, що дозволяє користувачу швидко реагувати на нові лоуди та оновлення.

Помилки та їх виправлення:

1. Список найбільш поширених помилок при роботі з ботом, з поясненнями, як їх виправити.

2. Поради щодо того, як коректно налаштовувати бота, щоб уникнути проблем із отриманням даних або збоїв у роботі.

FAQ (Часто задавані питання):

1. Розділ із відповіді на найбільш поширені запитання користувачів щодо використання бота.

2. Включає питання про налаштування, параметри фільтрів, способи оптимізації взаємодії з ботом та інші технічні аспекти.

Контакти для підтримки:

1. Контактна інформація для користувачів, які потребують допомоги, а також посилання на канали для зворотного зв'язку (чат, технічна підтримка, email).

2. Розділ з посиланнями на додаткові ресурси та допоміжні матеріали.

2.5.3. Підхід до створення документації

Інтерактивний підхід: Ми прагнули зробити документацію не лише статичним набором інструкцій, а й інтерактивним посібником. Кожен розділ, що стосується команд та налаштувань, мав зручні посилання на демонстраційні відео або

скріншоти, що наглядно показували, як виконати певні дії. Крім того, був використаний формат FAQ, що дозволяє швидко знаходити відповіді на популярні запитання.

Мінімалістичний дизайн: Документація була спрощена до основних аспектів, щоб не перевантажувати користувача зайвою інформацією. Було приділено увагу чітким і зрозумілим інструкціям, без складних технічних термінів, щоб навіть користувачі без спеціалізованих знань могли швидко налаштувати бот і розпочати роботу.

Тестування документації: Після того, як документація була створена, її пройшли тести з реальними користувачами, щоб переконатися в її зрозумілості та коректності. Були проведені опитування серед тестових користувачів щодо того, які частини документації потрібно покращити або пояснити детальніше.

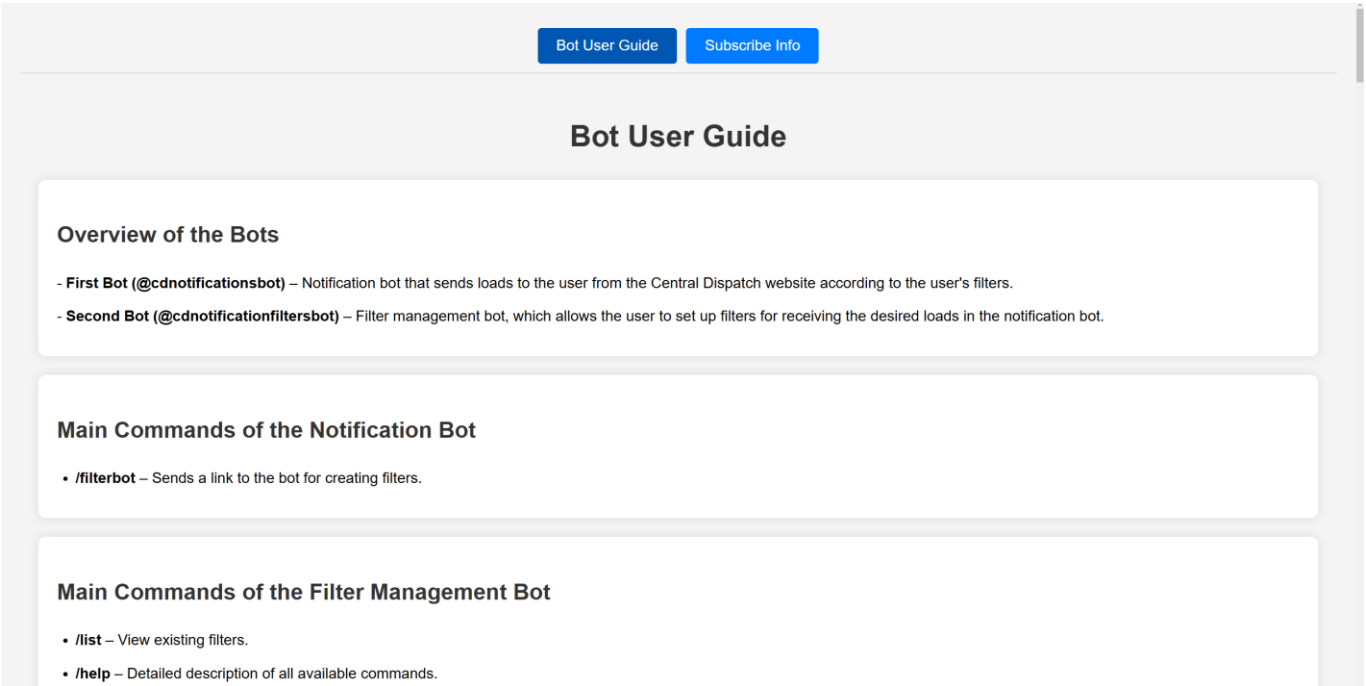

Рис. 2.2. Відображення документації

2.5.4. Позитивні сторони документації

Швидке освоєння бота: Завдяки чітким інструкціям та покроковим описам, користувачі змогли швидко освоїти основні функції бота та почати ефективно взаємодіяти з ним.

Зручний доступ до інформації: Завдяки структурованій документації користувачі змогли швидко знайти потрібну інформацію без необхідності звертатися до служби підтримки.

Підвищена ефективність використання бота: Пояснення щодо налаштування фільтрів, отримання сповіщень та інтерактивних команд допомогло користувачам з максимальним ефектом використовувати бота для отримання важливої інформації.

2.6 Реалізація оплати за використання бота

2.6.1. Пошук платіжної системи

У процесі розробки бота для взаємодії з логістичними компаніями виникла важлива задача — реалізувати систему оплати за використання бота. Це питання стало важливим, оскільки безпосередньо впливає на монетизацію проєкту, а також на можливість забезпечити доступ до додаткових функцій і преміум-сервісів.

Проблема полягає не тільки у виборі відповідної платіжної системи, а й у тому, що я мав врахувати не лише стандартні моделі оплати, такі як разова підписка, а й більш складні варіанти, як групові підписки для компаній або організацій. Тому цей етап був одним з найскладніших, оскільки жодна з готових платіжних систем не могла в повній мірі відповідати всім моїм вимогам.

На самому початку я переглядав різні платіжні системи, які могли б бути інтегровані в Telegram бота. Це були як міжнародні платіжні системи, так і локальні платформи, які мали певні переваги у використанні. Однак жодна з них не підходила для вирішення моєї задачі, оскільки кожна система мала свої обмеження або не підтримувала групові підписки, які були необхідні для клієнтів, що працюють в компаніях.

Міжнародні платіжні системи

PayPal: PayPal — це одна з найпопулярніших міжнародних платіжних систем, яка підтримує транзакції в різних валютах. Однак я зіткнувся з кількома обмеженнями. По-перше, для отримання доступу до API PayPal потрібно мати бізнес-акаунт, що створює додаткові бар'єри для використання. По-друге, я не зміг

знайти пряме рішення для групових підписок, що сильно ускладнювало б інтеграцію.

Stripe: Stripe також є популярним варіантом, який надає можливість здійснювати платежі через API. Однак проблема з Stripe полягала в тому, що, як і в випадку з PayPal, він не підтримував автоматичне управління груповими підписками та мав складний процес налаштування для різних тарифних планів, що також не підходило для мого бота.

Локальні платіжні системи

Monobank: Вивчав можливість інтеграції з Monobank через їх API, але обмеження в документації та специфікація API не дозволяли налаштувати зручну модель оплати для користувачів бота.

LiqPay: ЛіКПей, хоча і мав привабливу можливість для швидкого налаштування платежів, не мав зручних можливостей для управління підписками з різними рівнями доступу. Я спробував налаштувати кілька типів підписок, але інтеграція вимагала додаткових складнощів та документів, що збільшувало час на реалізацію.

2.6.2. Вибір Wayforpay

Після довгого пошуку я зупинився на Wayforpay, оскільки саме ця система відповідала моїм вимогам. Wayforpay має кілька важливих переваг:

1. Підтримка групових підписок: Wayforpay дозволяє налаштувати різні типи підписок, включаючи як індивідуальні, так і групові варіанти. Це означає, що я міг створити тарифні плани для окремих користувачів, а також для компаній, де кілька людей можуть використовувати один акаунт, що є дуже важливим для логістичних компаній.

2. Простота інтеграції: Всі інтерфейси для інтеграції з Wayforpay досить прості, і вони дозволяли швидко підключити API до бота. Платформа підтримує різні варіанти платіжних методів, включаючи картки, онлайн-банкінг та інші способи оплати.

3. Можливість реалізації гнучких тарифних планів: Wayforpay підтримує систему створення різних тарифних планів, що дозволяє розділяти користувачів на категорії і надавати різні функції в залежності від обраного тарифу. Це зручно для створення різноманітних підписок, наприклад, стандартних для окремих користувачів або преміум для компаній.

4. Легкість адміністрування: Зручна панель управління Wayforpay дозволяла мені контролювати всі платежі та підписки в реальному часі, що спрощувало обробку транзакцій і дозволяло швидко виявляти будь-які неполадки або проблеми з оплатою.

5. Підтримка різних валют: Wayforpay підтримує безліч валют, що дозволяє реалізувати оплату в різних країнах, якщо в майбутньому буде потрібно. Це забезпечило можливість розширення функціональності бота в міжнародному контексті.

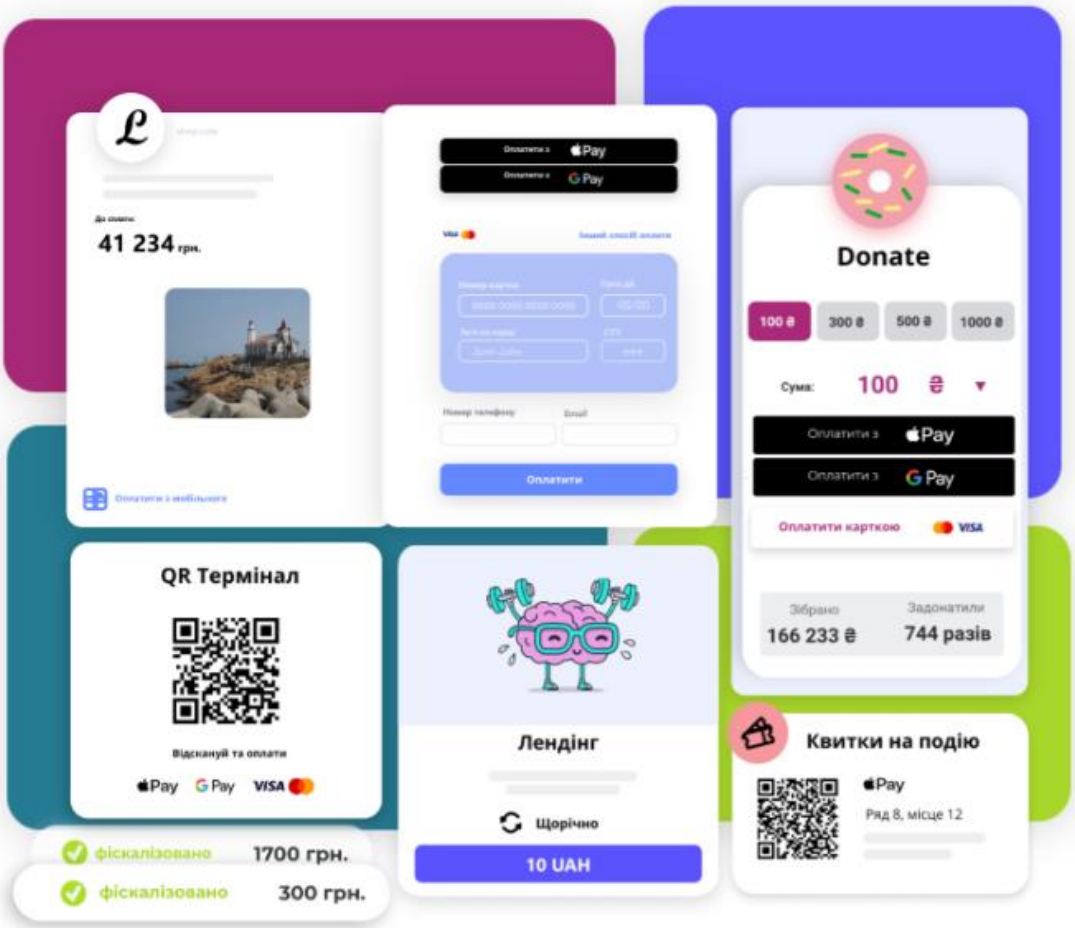

Рис 2.3. Відображення оплати

2.6.3. Реалізація оплати в бота

Реалізація оплати за допомогою Wayforpay в бота була здійснена через кілька етапів:

1. Інтеграція API: Я налаштував API Wayforpay для прийому платежів та перевірки статусу транзакцій. Для цього потрібно було отримати ключі API від Wayforpay та правильно налаштувати їх у коді бота, щоб він міг відправляти запити на платіжну систему і отримувати відповіді про оплату.

2. Обробка підписок: У бота була розроблена система підписок, яка дозволяє користувачам вибирати між різними планами. Для кожного плану були налаштовані різні функції доступу, що дозволяло надавати преміум-функції для тих користувачів, які оплатили підписку.

3. Оповіщення користувачів: Після успішної оплати бот автоматично сповіщає користувача про активацію підписки та надає доступ до відповідних функцій. Окрім того, система підтримує автоматичне продовження підписок, якщо це необхідно.

4. Контроль платежів: За допомогою панелі адміністратора Wayforpay я можу відслідковувати всі транзакції, що дозволяє швидко реагувати на можливі проблеми з оплатою та надавати підтримку користувачам.

2.7 Розміщення бота на хостингу та його запуск

2.7.1. Вибір хостингу

Одним з ключових етапів у розробці мого бота для взаємодії з логістичними компаніями стало питання його розміщення на хостингу. Це виявилося важливою частиною проекту, оскільки від того, на якому сервері буде працювати бот, залежала його стабільність, ефективність роботи та можливість масштабування. Я випробував кілька хостингів і зіткнувся з рядом проблем, кожна з яких потребувала індивідуального підходу для вирішення.

PythonAnywhere

На початку проекту я зупинився на PythonAnywhere. Це був перший хостинг, який я вибрав через його простоту у використанні для розміщення Python-скриптів і

бота. PythonAnywhere дозволяє запускати скрипти без необхідності налаштування власного серверу, що зробило його зручним для швидкого тестування та початкового розміщення бота.

Однак я зіткнувся з кількома серйозними обмеженнями. По-перше, серверна частина PythonAnywhere була не дуже потужною, що призводило до регулярних відмов у роботі бота, особливо під час пікових навантажень. Крім того, сервери PythonAnywhere часто ставали мішенню для DDoS-атак, що призводило до частих падінь бота. Хоча PythonAnywhere пропонує зручні інтерфейси для налаштування та автоматичного запуску бота, нестабільність сервера була серйозною проблемою для мого проекту.

Replit

Після цього я спробував перехід на Replit, який здобув популярність завдяки зручному онлайн-редактору коду та можливості безкоштовно хостити невеликі проекти. На перший погляд, Replit виглядав привабливою альтернативою для бота, оскільки він підтримує Python і має зручний інтерфейс для деплою. Однак одна велика проблема виникла через сумісність Replit з парсером на базі Playwright.

Playwright, який я використовував для збору даних з сайту, відкриває віртуальний браузер і потребує потужніших серверів з підтримкою віртуалізації. На Replit цей підхід не працював, оскільки система не підтримує належне виконання графічних браузерів в рамках безкоштовного або базового плану. Це призвело до того, що парсер не міг працювати належним чином, а я не міг забезпечити потрібну стабільність і швидкість виконання запитів.

Heroku

Після довгих пошуків альтернатив і тестувань я зупинився на Heroku. Це популярна платформа для розміщення веб-додатків, яка підтримує Python і дає можливість запускати додатки з різними залежностями, включаючи такі утиліти як Playwright. Heroku надає безкоштовний тарифний план для малих додатків, що стало відмінним стартом для мого проекту.

Переваги Heroku:

1. Масштабованість: Heroku дозволяє легко масштабувати додатки, що є важливим для подальшого розвитку бота, якщо він буде обслуговувати більшу кількість користувачів. Можливість додавати більше "динамічних" контейнерів дозволила вирішити проблему з обробкою великої кількості запитів одночасно.

2. Підтримка Playwright: Однією з головних переваг для мене стала можливість запускати Playwright у віртуальних контейнерах. Я налаштував середовище для запуску браузера через Playwright, і це дозволило боту збирати дані без проблем.

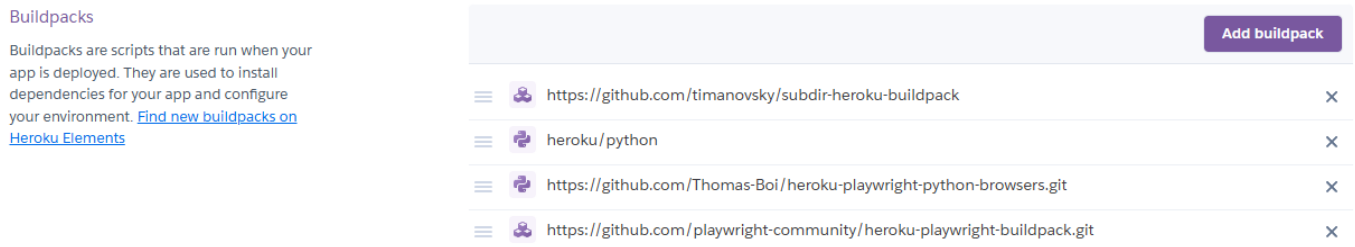

Рис 2.4. Налаштування середовища Heroku

3. Інтеграція з GitHub: Heroku легко інтегрується з GitHub, що дозволило мені автоматизувати деплой. Кожен раз, коли я вносив зміни в репозиторій, Heroku автоматично оновлював середовище і деплоїв нову версію бота.

4. Простота налаштування: Завдяки добре документованому API та інтерфейсу, я зміг швидко налаштувати необхідні утиліти для запуску бота на Heroku. Не було потреби вручну налаштовувати сервери або складні конфігурації.

Проблеми з адаптацією

Під час перенесення парсера на Heroku я зіткнувся з кількома додатковими проблемами. Одна з них полягала в тому, що для запуску Playwright потрібно було додати додаткові утиліти та налаштування для запуску браузера в контейнері. Також

виникли питання з перевищенням лімітів безкоштовного плану Heroku, оскільки для стабільної роботи бота потрібні були додаткові ресурси (CPU, пам'ять). В результаті я перейшов на платний план, щоб забезпечити стабільну роботу і знизити ймовірність збоїв.

2.7.2. Проблеми під час запуску

Незважаючи на те, що Heroku надає багато зручних інструментів, я все ж таки зіштовхнувся з деякими проблемами під час запуску бота:

1. Затримки при старті бота: В перші моменти після запуску бота на Heroku могли виникати затримки, поки система ініціалізувала всі залежності. Це особливо стосувалося запуску Playwright, який потребує додаткового часу для завантаження браузера.

2. Обмеження за ресурсами: Як і у випадку з PythonAnywhere, Heroku має ліміти на кількість ресурсів (наприклад, кількість годин використання на місяць для безкоштовних планів). Якщо боти використовували забагато ресурсів через інтенсивну роботу парсера або велике навантаження на сервер, це призводило до тимчасових відключень.

3. Налаштування належного середовища: Оскільки я використовував багато різних бібліотек і утиліт для парсера та бота, мені довелося налаштувати середовище вручну, що іноді вимагало додаткових зусиль. Наприклад, для того, щоб забезпечити сумісність Playwright з Heroku, мені довелося налаштувати додаткові бібліотеки, як-от libvips і інші залежності для правильного рендерингу браузера в контейнері.

2.7.3. Завершення процесу запуску

Після вирішення всіх технічних проблем та оптимізації налаштувань я зміг забезпечити стабільну роботу бота на Heroku. Бот почав працювати без перебоїв, з наявністю всіх необхідних функцій для збору даних, взаємодії з користувачами через Telegram і обробки запитів. За допомогою Heroku я зміг отримати

безперебійний доступ до серверів та змогти легко масштабувати систему при необхідності.

2.7.4 Бази даних

Бази даних є важливою частиною будь-якої веб-розробки, і в цьому контексті використання PostgreSQL виявляється одним із найбільш ефективних рішень для забезпечення зберігання та обробки великих обсягів даних. PostgreSQL — це потужна реляційна система управління базами даних з відкритим кодом, яка відома своєю надійністю, гнучкістю та підтримкою складних запитів і транзакцій. Вона є ідеальним вибором для застосунків, що потребують високої продуктивності, масштабованості та безпеки, і тому вона була обрана для цього проекту.

PostgreSQL підтримує складні типи даних, зокрема JSON, що дозволяє зберігати структуровану інформацію та працювати з нею ефективно. Ця особливість особливо корисна для веб-додатків, які мають справу з різними форматами даних, як-от фільтри користувачів або налаштування. Крім того, PostgreSQL забезпечує чудову підтримку транзакцій, що дозволяє гарантувати консистентність даних навіть у разі відмови системи чи виникнення непередбачуваних ситуацій.

Однією з найбільших переваг використання PostgreSQL є її здатність до масштабування. Вона підтримує розширену функціональність для оптимізації великих обсягів даних, що дозволяє зберігати записи, навіть коли кількість користувачів і запитів значно зростає. Наприклад, для обробки фільтрів, які можуть мати складну логіку та потребують швидкої обробки даних, PostgreSQL забезпечує високий рівень оптимізації запитів, використовуючи індекси та інші механізми, що значно прискорюють процес.

Також PostgreSQL є відомим за свою здатність до роботи з величезними обсягами даних без втрати продуктивності. У проектах, де необхідно зберігати великі обсяги інформації про лоуди, користувачів та історії замовлень, PostgreSQL забезпечує ефективне зберігання та швидкий доступ до цих даних. За допомогою функцій, як-от партиціонування таблиць і індекси, можна ще більше підвищити ефективність роботи з великими базами даних.

Однією з ключових функцій PostgreSQL є підтримка запитів, що охоплюють кілька таблиць, зокрема операції з об'єднанням таблиць та складними запитами. Це є дуже важливим для реалізації функціоналу, такого як фільтрація даних за кількома критеріями, що активно використовується у ботах для логістичних рішень. Завдяки розширеному SQL і можливості використовувати складні запити, PostgreSQL дозволяє без труднощів реалізовувати навіть найбільш вимогливі до бази даних функції.

Також варто зазначити, що PostgreSQL є дуже надійною в плані безпеки даних. Вона підтримує різні механізми аутентифікації, що дозволяє налаштувати безпечний доступ до бази даних. Крім того, за допомогою ролей та дозволів можна забезпечити, щоб доступ до критичних даних був обмежений лише певними користувачами, що особливо важливо для проектів, що обробляють персональні або конфіденційні дані.

Ще однією важливою характеристикою PostgreSQL є її підтримка реплікації даних, що дозволяє забезпечити високу доступність і відмовостійкість системи. У разі відмови основного сервера бази даних, можна автоматично переключити запити на резервні сервери, що значно зменшує час простою і гарантує надійність роботи додатка.

Використання PostgreSQL в цьому проекті дозволяє забезпечити високу продуктивність і ефективність обробки даних, що є ключовим фактором для успішної роботи веб-додатку та бота. Ця система управління базами даних дозволяє обробляти великий обсяг запитів, зберігати складні дані та забезпечити безпеку і стабільність роботи на довготривалій основі.

PostgreSQL є однією з найпопулярніших реляційних баз даних і має ряд переваг, які виділяють її серед інших баз даних, таких як MySQL, SQLite або MongoDB. Кожна з цих систем управління базами даних (СУБД) має свої сильні та слабкі сторони в залежності від вимог проекту. Порівняємо PostgreSQL з кількома іншими популярними базами даних.

1. PostgreSQL vs MySQL:

- Масштабованість і продуктивність: PostgreSQL відомий своєю здатністю до масштабування за рахунок підтримки індексів, розподілених баз даних і складних запитів. MySQL, з іншого боку, є дуже швидким у виконанні простих запитів і використовує менше ресурсів, але не підтримує деякі складніші функції, які є у PostgreSQL, наприклад, складні типи даних та розширене управління транзакціями.
- Функціональність: PostgreSQL підтримує розширені функціональні можливості, такі як складні запити, повний текстовий пошук, використання користувацьких типів даних і розширення. MySQL, хоч і підтримує деякі з цих функцій, має більш обмежену підтримку складних запитів та аналітики.
- Сумісність з ACID: PostgreSQL є повністю сумісним із стандартами ACID (атомарність, консистентність, ізоляція, довговічність), що забезпечує високу надійність при роботі з транзакціями. MySQL забезпечує ACID лише в інтерфейсі InnoDB, хоча його конфігурація не завжди дає таку ж надійність.

2. PostgreSQL vs SQLite:

- Продуктивність і масштабованість: SQLite — це вбудована база даних, що працює без окремого серверного процесу, що робить її швидкою для малих проектів і тестування. Однак, при збільшенні кількості запитів або даних, SQLite може не справлятися з високими навантаженнями, в той час як PostgreSQL розрахований на великі та складні проекти, де потрібна висока продуктивність і масштабованість.
- Типи даних: PostgreSQL підтримує складніші типи даних, такі як JSON, користувацькі типи даних та географічні дані, в той час як SQLite має обмежену підтримку таких функцій.
- Стійкість і безпека: PostgreSQL має більше можливостей для налаштування безпеки та доступу до даних, включаючи підтримку ролей, а також реплікацію і резервування. SQLite є менш надійним для

критичних додатків, оскільки не підтримує вбудовану реплікацію чи контроль за доступом.

3. PostgreSQL vs MongoDB:

- Типи даних: MongoDB — це документно-орієнтована база даних, яка працює з документами в форматі JSON. Вона ідеально підходить для зберігання неструктурованих або змінних даних, таких як дані з різними полями або великими вкладеними структурами. PostgreSQL, з іншого боку, є реляційною базою даних, яка зберігає дані у вигляді таблиць, що робить її більш підходящою для структурованих даних і для випадків, коли потрібна складна логіка з'єднань між таблицями.
- Транзакції і консистентність: PostgreSQL забезпечує повну підтримку транзакцій ACID, що робить її ідеальним вибором для додатків, де потрібна висока консистентність даних. MongoDB також підтримує транзакції, але вони не так ефективні, як у PostgreSQL, і більше орієнтовані на масштабованість.
- Гнучкість і зручність використання: MongoDB є гнучкішою у роботі з неструктурованими або змішаними даними, тому вона може бути кращим вибором для застосунків, де дані часто змінюються або мають вільну структуру. PostgreSQL, як реляційна база даних, надає більшу стабільність і контроль над даними, особливо при роботі з великими обсягами структурованих даних.

4. PostgreSQL vs NoSQL:

- Типи запитів: Реляційні бази даних, такі як PostgreSQL, добре підходять для запитів, що вимагають з'єднання даних з кількох таблиць за допомогою складних SQL-запитів. Бази даних NoSQL, навпаки, зручніші для роботи з великими обсягами даних, що не мають чіткої структури, таких як документи або ключ-значення.
- Масштабованість: NoSQL-системи, як правило, краще масштабується горизонтально, тобто можна додавати нові сервери для обробки більшої кількості даних без значного зниження продуктивності. PostgreSQL, з

іншого боку, більше орієнтований на вертикальне масштабування, тобто на підвищення продуктивності шляхом використання потужніших серверів, хоча також підтримує горизонтальне масштабування через додаткові інструменти, такі як Citus.

Таким чином, вибір між PostgreSQL і іншими базами даних залежить від конкретних вимог проекту. PostgreSQL є відмінним вибором для додатків, які потребують надійності, складних запитів та високої продуктивності при роботі з великими обсягами структурованих даних, в той час як інші СУБД можуть бути кращими для більш специфічних випадків, таких як робота з неструктурованими даними або вимога до горизонтального масштабування.

2.7.5 Підсумки

Розміщення бота на Heroku стало оптимальним рішенням для моєї ситуації. Хоча з PythonAnywhere я стикався з численними проблемами, пов'язаними з обмеженням ресурсів і атакою на сервери, Heroku надав мені більше можливостей для стабільної роботи проекту. Технології, що підтримуються платформою, і можливість швидкого масштабування зробили Heroku найбільш придатним варіантом для запуску бота. У майбутньому, при зростанні кількості користувачів або необхідності масштабування проекту, я маю можливість додавати більше ресурсів і продовжувати розвиток без значних технічних обмежень.

2.8 Оптимізація процесів

2.8.1 Використання Redis для оптимізації роботи бота

Redis є потужною системою керування базами даних в пам'яті, яка використовує структури даних, що зберігаються у вигляді ключ-значення, для забезпечення швидкого доступу до інформації. Ця система часто застосовується для кешування, зберігання сеансів користувачів, управління чергами та інші операції, де потрібен високопродуктивний доступ до даних. Враховуючи високу швидкість роботи Redis, він став невід'ємною частиною багатьох проектів, що вимагають

реального часу для обробки великих обсягів даних. У випадку з ботом, який обробляє складні фільтри для користувачів, Redis дозволяє значно підвищити ефективність роботи та зменшити затримки в роботі з даними.

Одним з основних використань Redis у вашій системі є як проміжний етап для зберігання та доступу до фільтрів користувачів. Коли користувач створює новий фільтр, Redis миттєво зберігає цю інформацію, дозволяючи користувачеві бачити свій фільтр в реальному часі, навіть якщо ці дані ще не були збережені в основній базі даних. Це дозволяє значно скоротити час очікування користувача, надаючи йому миттєвий доступ до своїх фільтрів, що в свою чергу підвищує користувацький досвід і забезпечує зручність.

Принцип роботи Redis у моєму випадку

Швидкість доступу до даних: Основною перевагою використання Redis є надзвичайно швидкий доступ до даних. Оскільки Redis зберігає всю інформацію в пам'яті, операції з читання та записом виконуються за мілісекунди. Це означає, що користувач може миттєво побачити результат своїх дій, наприклад, створення або редагування фільтра, навіть до того, як ці дані будуть записані в основну базу даних.

Проміжне збереження фільтрів: Коли користувач створює новий фільтр, система спочатку записує цей фільтр в Redis, де він стає доступним для користувача миттєво. На цьому етапі користувач вже може побачити свій фільтр в інтерфейсі, хоча в цей час система продовжує обробляти дані і зберігати їх у базі даних у фоновому режимі. Це дозволяє уникнути затримок, які могли б виникнути, якби система чекала завершення запису у базу даних перед тим, як надати користувачеві доступ до фільтра.

Використання Redis як кешу: Redis можна використовувати як кеш, що зберігає найбільш використовувані дані. Якщо фільтр був нещодавно створений або змінений, Redis забезпечує доступ до цього фільтра набагато швидше, ніж основна база даних. Це означає, що якщо користувач взаємодіє з тим самим фільтром кілька разів, дані будуть швидко отримуватися з Redis без необхідності кожного разу робити запит до бази даних.

Механізм Pub/Sub для реального часу: Redis також пропонує механізм "publish/subscribe" (Pub/Sub), що дозволяє одному процесу публікувати дані в канал, а іншим підписаним процесам миттєво отримувати ці дані. Це дозволяє боту забезпечити реальний час оновлення інформації для всіх користувачів. Наприклад, якщо один користувач оновлює фільтр, всі інші користувачі, які підписалися на цей канал, автоматично отримають оновлену інформацію, що забезпечує синхронізацію даних у реальному часі без затримок.

Запис у основну базу даних: Після того як користувач побачить фільтр в Redis, система може відкладено записати ці дані в основну базу даних. Це дозволяє зменшити навантаження на основну базу даних, оскільки операції запису не блокують доступ до даних і не уповільнюють взаємодію з користувачем. Базу даних можна оновити за допомогою фонових задач або періодичних синхронізацій.

Переваги використання Redis для оптимізації бота

Зменшення затримок: Основною перевагою є скорочення часу відповіді для користувача, оскільки дані з Redis доступні значно швидше, ніж з традиційної бази даних. Завдяки цьому користувачі можуть миттєво побачити результати своєї діяльності, не чекаючи запису в базу даних.

Підвищення продуктивності: Використання Redis дозволяє зменшити навантаження на основну базу даних, оскільки Redis обробляє більшість запитів, що дозволяє основній базі даних обробляти лише критичні та складні операції. Це допомагає системі працювати більш ефективно.

Забезпечення масштабованості: Оскільки Redis є високопродуктивною і горизонтально масштабованою системою, вона дозволяє вашій інфраструктурі збільшувати потужності без істотних змін у архітектурі. Це дозволяє вам масштабувати бот для обробки великої кількості користувачів та їх запитів.

Синхронізація даних в реальному часі: Завдяки механізму Pub/Sub у Redis можна забезпечити оновлення даних в реальному часі. Це особливо корисно, коли кілька користувачів одночасно взаємодіють з одними й тими ж даними, наприклад, коли вони працюють з однаковими фільтрами. У таких випадках Redis гарантує, що всі зміни будуть відображені миттєво.

Redis є надзвичайно потужним інструментом для оптимізації роботи сучасних додатків, зокрема для підвищення ефективності взаємодії з користувачами. Цей продукт, заснований на принципі зберігання даних в пам'яті, забезпечує високошвидкісне читання та запис даних, що робить його ідеальним для систем, які потребують мінімальних затримок у процесі обробки запитів. Одним із найбільших плюсів Redis є те, що він дозволяє зберігати не тільки прості значення типу «ключ-значення», але й складніші структури даних, як-от списки, множини або хеші, що робить його дуже гнучким. У контексті використання Redis для бота, це дозволяє зберігати дані, до яких потрібно швидко отримати доступ, прямо в пам'яті, що скорочує час відповіді і забезпечує зручний інтерфейс для користувачів. Наприклад, якщо користувач створює новий фільтр або змінює існуючий, Redis миттєво зберігає ці дані і надає користувачу доступ до них, навіть до того, як вони будуть записані в основну базу даних. Це створює враження, що система працює дуже швидко, надаючи користувачеві відчуття миттєвого виконання операцій. Одним з важливих аспектів використання Redis є його здатність до горизонтального масштабування, що дозволяє додавати нові вузли для обробки більшої кількості запитів без значних змін в архітектурі. Ця можливість є надзвичайно важливою для систем, що обробляють велику кількість користувачів і постійно ростуть у масштабах.

Використання Redis також дозволяє значно знизити навантаження на основну базу даних, оскільки більшість запитів до даних можна обробляти безпосередньо через Redis, що зменшує кількість операцій введення/виведення в основній базі даних. Це особливо важливо в ситуаціях, коли необхідно працювати з великими обсягами даних, оскільки Redis дозволяє вирішувати ці завдання набагато ефективніше і швидше. Завдяки таким функціям, як підтримка різноманітних структур даних і високопродуктивне кешування, Redis є оптимальним вибором для багатьох задач, де важлива швидкість і ефективність. Для вашого бота це означає, що користувачі можуть працювати з фільтрами в реальному часі, отримуючи результати без затримок, а сама система продовжує ефективно обробляти великі обсяги запитів навіть під великим навантаженням.

2.9 Висновок до розділу

Розділ "Створення бота для взаємодії з логістами" охоплює весь цикл розробки інструменту, який забезпечує автоматизацію роботи з логістами та оптимізацію процесів збору й обробки інформації. У ході аналізу доступності даних з веб-сайтів виявлено ключові виклики, зокрема використання анти-парсерів, що успішно подолано завдяки інтеграції Playwright і застосуванню оптимізованого парсеру для збору великих обсягів даних.

Створення бота стало центральним етапом, який включав розробку механізмів для зручної взаємодії з користувачами, інтеграцію платіжної системи та забезпечення повноцінної документації для користувачів. Розміщення бота на хостингу дозволило забезпечити його стабільну роботу, а реалізація процесів оплати відкрила можливості монетизації проєкту.

На завершальному етапі проведено оптимізацію роботи бота, що дозволило підвищити швидкість виконання операцій і зменшити ресурсомісткість.

РОЗДІЛ 3

РОЗРОБКА МОДЕЛЕЙ ОПТИМІЗАЦІЇ ТА АВТОМАТИЗАЦІЇ ЛОГІСТИКИ ПЕРЕВЕЗЕНЬ

3.1 Створення моделі для маршрутизації транспортних засобів

3.1.1 Постановка задачі

Маршрутизація транспортних засобів є ключовою задачею в логістиці та транспортних системах. Її мета — оптимізація маршрутів для мінімізації витрат часу, палива та інших ресурсів при одночасному забезпеченні вимог клієнтів. Створення моделі для вирішення задач маршрутизації дозволяє автоматизувати планування та підвищити ефективність транспортування.

Задача маршрутизації транспортних засобів (ЗМТ) — це складна задача комбінаторної оптимізації та цілочисельного програмування, яка формулюється як питання: «Який оптимальний набір маршрутів для автопарку транспортних засобів, щоб доставити вантажі заданій множині клієнтів?». Вона є узагальненням класичної задачі комівояжера (ЗК), що робить її важливим інструментом для логістики, транспорту та управління ланцюгами постачання. Вперше задача була описана Джорджем Данцигом і Джоном Рамсером у 1959 році в їхній статті, яка запропонувала перший алгоритмічний підхід для вирішення проблеми оптимізації доставки бензину. Основною метою ЗМТ є мінімізація загальної вартості маршрутів, що включає витрати на паливо, обслуговування транспортних засобів, час доставки та ефективність використання ресурсів. Задача має широкий спектр практичних застосувань, серед яких доставка товарів із центрального складу клієнтам, управління кур'єрськими службами та оптимізація маршрутів обслуговування. У 1964 році Кларк і Райт значно вдосконалили підхід Данцига і Рамсера, запропонувавши ефективний жадібний алгоритм, який отримав назву алгоритм економії.

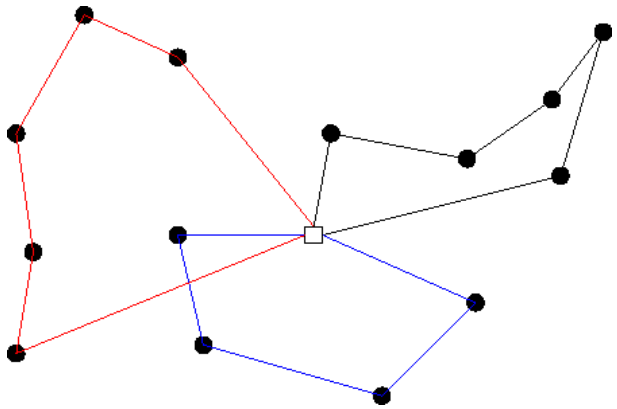

Рис. 3.1. Проблеми маршрутизації транспортних засобів

Задача маршрутизації транспортних засобів (Vehicle Routing Problem, VRP) визначається як задача пошуку оптимального набору маршрутів для автопарку, який обслуговує задану кількість клієнтів. Основна мета полягає в мінімізації витрат, пов'язаних із доставкою, таких як витрати на паливо, час у дорозі та обслуговування транспорту, з урахуванням обмежень, таких як вантажопідйомність транспортних засобів, часові вікна для доставки та інші логістичні фактори.

Основні компоненти задачі:

- *Мережа*: набір вузлів (клієнтів або точок доставки) і дуг (доріг між вузлами) з відповідними витратами (час, відстань, вартість).
- *Обмеження*: максимальна вантажопідйомність транспортних засобів, часові вікна доставки, максимальна довжина маршруту тощо.
- *Цільова функція*: мінімізація загальних витрат, наприклад, часу або відстані.

Типи VRP:

- *Класичний VRP (CVRP):* всі транспортні засоби однакові, і кожен клієнт має попит, що не перевищує місткість транспортного засобу.
- *VRP з часовими вікнами (VRPTW):* клієнти повинні бути обслуговані в межах певного часовго періоду.
- *VRP з неоднорідним парком транспортних засобів (HVRP):* транспортні засоби мають різні характеристики (місткість, витрати).
- *Динамічний VRP (DVRP):* інформація про замовлення може змінюватися в реальному часі.

3.1.2 Підхід до моделювання

1. Математична модель:

Визначаються змінні, наприклад:

- чи проходить транспортний засіб від точки до точки (бінарна змінна).
- залишкова вантажопідйомність транспортного засобу після обслуговування точки .

Цільова функція:

- вартість проходження дуги між точками та .

Обмеження:

- Кожен клієнт має бути обслугований один раз.
- Транспортний засіб не перевищує свою вантажопідйомність.
- Час обслуговування відповідає часовим вікнам.

2. Алгоритми розв'язання:

- Точні методи: наприклад, лінійне програмування, розгалуження та межі.
- Евристичні алгоритми: найближчий сусід, метод пошуку шляху.
- Метагевристики: генетичні алгоритми, методи рою частинок, імітація відпалу.

3.1.3 Практичне застосування

Моделі маршрутизації транспортних засобів широко застосовуються в:

- Кур'єрських службах (оптимізація доставки замовлень).
- Управлінні міським транспортом (оптимізація автобусних маршрутів).
- Розподілі товарів (логістика роздрібної торгівлі).
- Управлінні автопарком (мінімізація витрат на обслуговування).

3.2. Розробка методів прогнозування завантаженості транспортної мережі.

Прогнозування завантаженості транспортної мережі є важливим етапом у плануванні та оптимізації логістичних процесів. Ефективні методи прогнозування

дозволяють оцінити майбутній попит на транспортні ресурси, уникати перевантаження мережі та забезпечувати стабільність перевезень. Розробка таких методів потребує інтеграції статистичних підходів, машинного навчання та моделювання, орієнтованого на специфіку транспортних систем.

Основним завданням є передбачення потоків транспортних засобів у певний момент часу або в конкретній ділянці мережі. Для цього необхідно враховувати численні фактори, зокрема інтенсивність руху, географічні особливості, сезонність, погодні умови та інфраструктурні зміни. Дані, зібрані за допомогою сенсорів, GPS-трекерів, відеоспостереження або інших джерел, є базою для побудови моделей прогнозування.

Сучасні методи використовують комбінацію історичних даних та поточної інформації. Наприклад, регресійний аналіз дозволяє виявити залежності між змінними, а моделі часових рядів (ARIMA, SARIMA) враховують сезонні коливання та тенденції. Використання алгоритмів машинного навчання, таких як дерева рішень, нейронні мережі чи градієнтний бустинг, забезпечує високу точність прогнозів навіть у складних умовах, де традиційні підходи можуть виявитися менш ефективними.

Окрім точності прогнозування, важливою є здатність моделі адаптуватися до змін у транспортній системі. Для цього використовують методи оновлення моделей у реальному часі, інтегруючи нові дані без втрати узгодженості з історичною інформацією. Це дозволяє враховувати такі фактори, як несподівані затори, ремонти доріг чи зміни в маршрутах.

Результати прогнозування можуть бути представлені у вигляді інтерактивних карт, графіків або цифрових моделей, які інтегруються з інформаційними системами управління транспортом. Такий підхід допомагає операторам швидше приймати рішення щодо оптимізації маршрутів, перерозподілу транспортних засобів або коригування розкладу.

Для прогнозування завантаженості транспортної мережі застосовуються різноманітні алгоритми та моделі, які враховують як історичні, так і поточні дані.

Вибір методу залежить від доступності даних, характеру транспортної мережі та точності, необхідної для конкретної задачі.

1. Статистичні методи

- Регресійний аналіз: Використовується для встановлення залежності між кількістю транспортних засобів і факторами, такими як час доби, день тижня, погодні умови тощо.
- Моделі часових рядів (ARIMA, SARIMA): Прогнозують потоки на основі історичних даних, враховуючи тенденції, сезонність і циклічність.
- Крос-кореляційний аналіз: Допомагає виявити взаємозв'язки між трафіком на різних ділянках мережі.

2. Моделі машинного навчання

- Лінійна та логістична регресія: Простий і зрозумілий підхід для моделювання потоків у невеликих транспортних мережах.
- Дерева рішень і градієнтний бустинг (LightGBM, XGBoost): Використовуються для моделювання складних залежностей між параметрами, такими як завантаженість доріг, швидкість руху і затори.
- Рекурентні нейронні мережі (RNN): Ефективні для обробки часових рядів, оскільки враховують часову залежність.
- LSTM (Long Short-Term Memory): Враховують довгострокові залежності в транспортних даних, що дозволяє точно прогнозувати майбутню завантаженість.
- CNN (Convolutional Neural Networks): Застосовуються для аналізу геопросторових даних, наприклад, карт або зображень трафіку.

3. Алгоритми оптимізації

- Динамічне програмування: Застосовується для пошуку оптимальних рішень щодо маршрутизації, враховуючи прогнозовану завантаженість.
- Евристичні методи (генетичні алгоритми, алгоритми рою): Використовуються для пошуку найкращих рішень у великих і складних мережах.

4. Моделі симуляції

- Мікроскопічне моделювання: Вивчає поведінку окремих транспортних засобів у потоці (наприклад, моделі водія).
- Мезоскопічне моделювання: Поєднує деталі мікроскопічного моделювання з макроскопічними тенденціями.
- Макроскопічне моделювання: Аналізує потоки як безперервні, враховуючи їх інтенсивність і щільність (модель Лайтгіла-Віттема-Річардса).

5. Геопросторовий аналіз

- Геостатистика (Kriging): Використовується для прогнозування завантаженості в просторовому контексті.
- Алгоритми графів: Застосовуються для аналізу транспортної мережі як графа з вершинами (перехрестя) і ребрами (дороги).

6. Гібридні моделі

- Комбінація статистичних підходів і методів машинного навчання забезпечує більш точні прогнози. Наприклад, моделі LSTM можуть поєднуватися з ARIMA для урахування сезонних коливань.

Приклади застосування

- Google Maps і Waze: Використовують CNN для аналізу геопросторових даних і машинне навчання для прогнозування заторів.
- Інтелектуальні транспортні системи (ITS): Інтегрують кілька моделей для управління потоками на основі прогнозів.

3.3. Автоматизація планування перевезень із застосуванням машинного навчання.

Автоматизація планування перевезень є одним із ключових напрямів підвищення ефективності логістичних процесів у сучасному світі, враховуючи постійне зростання обсягів перевезень і складність транспортних мереж. Використання машинного навчання в цій галузі відкриває нові можливості для

вдосконалення процесів, дозволяючи значно підвищити точність прогнозування, оптимізувати маршрути, мінімізувати витрати та забезпечувати стабільну якість послуг. Завдяки аналізу великих обсягів даних і інтеграції алгоритмів, автоматизовані системи планування можуть адаптуватися до змінних умов, забезпечуючи ефективність навіть у складних ситуаціях.

Основною задачею в цій сфері є створення моделей, які враховують різноманітні фактори, такі як час доставки, завантаженість транспортної мережі, тип вантажу, технічні обмеження транспортних засобів і специфічні вимоги клієнтів. Машинне навчання надає змогу обробляти ці дані в режимі реального часу, виявляючи складні закономірності та пропонуючи оптимальні рішення. Наприклад, алгоритми класифікації визначають пріоритетність замовлень, а моделі регресії прогнозують точний час доставки, враховуючи змінні дорожні умови, погоду чи інші зовнішні чинники.

Одним із центральних аспектів автоматизації є оптимізація маршрутів. Алгоритми машинного навчання аналізують дані про геолокацію, інфраструктуру та дорожній трафік, щоб знаходити найефективніші шляхи для перевезень. Застосування рекурентних нейронних мереж дозволяє враховувати тимчасові зміни, наприклад, години пік, аварії на дорогах або сезонні коливання в завантаженості транспортної інфраструктури. Такі підходи сприяють зниженню витрат на пальне, підвищенню ефективності використання транспортних засобів і скороченню часу доставки.

Важливим напрямком є також оптимізація управління запасами та завантаженням транспортних засобів. Використання алгоритмів кластеризації допомагає групувати вантажі за схожими параметрами, що дозволяє максимально ефективно використовувати наявний транспортний простір і зменшити кількість рейсів. Це особливо актуально для великих логістичних компаній, які оперують значними обсягами вантажів та прагнуть мінімізувати логістичні витрати.

Машинне навчання також відіграє ключову роль у покращенні взаємодії з клієнтами. Прогностичні моделі забезпечують точну інформацію про очікуваний час прибуття товарів, що сприяє зниженню кількості скарг і підвищенню рівня

задоволеності клієнтів. Крім того, аналіз даних про попередні замовлення дає змогу виявляти тренди та передбачати майбутній попит, що значно покращує планування ресурсів і дозволяє уникнути нестач або надлишку запасів.

Загалом, впровадження машинного навчання у процеси автоматизації планування перевезень створює значні переваги, забезпечуючи високу адаптивність, точність і ефективність. Це відкриває нові перспективи для подальшого розвитку логістичних систем, що дозволяє компаніям бути більш конкурентоспроможними в умовах сучасного ринку.

3.4 Висновок до розділу

Розділ "Розробка моделей оптимізації та автоматизації логістики перевезень" присвячений створенню ефективних рішень для удосконалення ключових аспектів логістичних процесів. Розробка моделі для маршрутизації транспортних засобів дозволила оптимізувати шляхи доставки, зменшити витрати та час перевезень.

Методи прогнозування завантаженості транспортної мережі забезпечили можливість аналізу та планування ресурсів, що сприяє уникненню перевантажень і збільшенню пропускної здатності системи. Впровадження машинного навчання у процес автоматизації планування перевезень дало змогу створити гнучкий інструмент для адаптивного прийняття рішень залежно від змінних умов.

РОЗДІЛ 4

АПРОБАЦІЯ ТА РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

4.1 Проведення експериментів із застосуванням розроблених моделей

4.1.1. Автоматичний пошук комбінацій лоудів (на основі ШІ)

Розвиток технологій і потреб ринку логістики відкриває величезні перспективи для розширення функціоналу бота. Я поставив перед собою амбітні цілі, щоб зробити бота незамінним інструментом для логістів, диспетчерів та менеджерів. Наступні функції, які будуть впроваджені в майбутньому, не лише полегшать щоденну роботу, але й значно підвищать ефективність процесів логістики.

Однією з ключових функцій стане можливість автоматичного пошуку комбінацій лоудів для траків за допомогою штучного інтелекту. Ця система:

1. Враховуватиме маршрут траку: об'єм вантажу, вагу та інші параметри, щоб пропонувати найоптимальніші варіанти.
2. Забезпечить аналіз історичних даних: бот збиратиме інформацію про попередні замовлення, щоб створювати більш точні прогнози.
3. Навчання на реальних даних: використання машинного навчання дозволить системі адаптуватися до змін у ринку, таких як сезонність, зміни тарифів тощо.

Ця функція дозволить логістам уникнути "пустих пробігів" та мінімізувати витрати, забезпечуючи максимальну заповненість траків під час кожного рейсу.

4.1.2. Відслідковування маршруту траку

Додатково буде впроваджено функцію GPS-трекінгу, яка дозволить відслідковувати маршрут траку в реальному часі:

1. Логісти зможуть бачити, де знаходиться трак, час прибуття на наступну точку, а також можливі затримки.

2. Система надсилатиме сповіщення у випадку відхилення від маршруту або виникнення інших проблем.

3. Менеджери зможуть отримувати дані для оптимізації маршрутів і поліпшення логістичних процесів.

Інтеграція з картографічними сервісами (наприклад, Google Maps чи OpenStreetMap) забезпечить точність і простоту у використанні цієї функції.

4.1.3. Автоматичний пошук для вже завантаженого траку

Ще однією важливою функцією стане автоматичний пошук лоудів для траку, який уже частково завантажений:

1. Бот аналізуватиме залишкову вагу та об'єм траку, щоб знайти додатковий вантаж, який відповідає цим параметрам.

2. Це дозволить логістам уникати ситуацій, коли трак їде напівпорожнім, і значно збільшить дохід компанії.

4.1.4. CRM для менеджерів

Для підвищення ефективності управління логістичними процесами буде впроваджено CRM-систему (систему управління взаємовідносинами з клієнтами). Ця система надасть менеджерам зручні інструменти для оптимізації роботи з логістами, клієнтами та замовленнями, а також дозволить автоматизувати ключові процеси.

1. Відслідковування ефективності роботи логістів

CRM-система надасть можливість оцінювати продуктивність кожного логіста за допомогою таких показників:

- Кількість організованих рейсів — точний облік виконаних завдань у розрізі часу (щоденно, щотижнево, щомісячно).
- Отриманий дохід — аналітика за кожним логістом, що допоможе зрозуміти їхній вклад у загальний фінансовий результат компанії.
- Ефективність маршрутів — оцінка якості логістичного планування для виявлення можливостей оптимізації.

Ці дані дозволять менеджерам приймати обґрунтовані рішення щодо розподілу задач, нарахування премій та подальшого навчання команди.

2. Збереження інформації про клієнтів

CRM об’єднає всю необхідну інформацію про клієнтів і замовлення в одному місці:

- Контактні дані — зручний доступ до контактів клієнтів для швидкого зв’язку.
- Історія замовлень — детальна інформація про попередні співпраці та виконані доставки.
- Переваги клієнтів — фіксування індивідуальних вимог і побажань для покращення якості обслуговування.

Централізоване збереження даних допоможе уникнути дублювання інформації та забезпечить прозорість у роботі з клієнтами.

3. Автоматизація процесів спілкування з клієнтами

CRM-система дозволить автоматизувати рутинні завдання, пов’язані зі спілкуванням, зокрема:

- Відправка сповіщень — автоматичні повідомлення про статус замовлення (прийнято, в дорозі, доставлено).
- Пропозиції та знижки — система дозволить створювати та надсилати персоналізовані пропозиції на основі історії замовлень.
- Нагадування про послуги — повідомлення клієнтам із нагадуванням про необхідність замовлення або про нові можливості компанії.

Це не лише підвищить якість обслуговування, а й покращить рівень комунікації з клієнтами, зміцнюючи їхню лояльність.

4. Аналітика для прийняття стратегічних рішень

CRM надасть глибокий аналітичний функціонал, що допоможе менеджерам ухвалювати стратегічно важливі рішення. Основні можливості:

- Розподіл ресурсів — аналіз завантаженості логістів для ефективного розподілу задач.

- Оптимізація витрат — ідентифікація слабких місць у процесах для зменшення витрат часу та ресурсів.
- Прогнозування — на основі історичних даних система зможе прогнозувати майбутні обсяги замовлень і потенційний прибуток.

Переваги впровадження

CRM-система значно підвищить операційну ефективність менеджерів, забезпечить контроль за роботою команди та покращить взаємодію з клієнтами. Завдяки автоматизації та аналітиці, менеджери зможуть більше уваги приділяти стратегічному плануванню та розвитку компанії, а не рутинним завданням.

4.2 Аналіз результатів і їх вплив на ефективність логістики перевезень.

Один із ключових аспектів майбутньої розробки — аналітика та статистика, які стануть потужним інструментом для підвищення ефективності роботи логістів і менеджерів. Завдяки глибокому аналізу даних, користувачі зможуть отримувати корисну інформацію для оцінки продуктивності, навчання персоналу та оптимізації робочих процесів.

1. Аналіз ефективності

Бот автоматично генеруватиме звіти про ключові показники ефективності, такі як:

- Кількість успішно виконаних замовлень — відображатиме загальну кількість доставок за певний період.
- Середній час доставки — допоможе аналізувати швидкість виконання завдань та виявляти вузькі місця у логістичних процесах.
- Затримки та причини — надаватиме детальну інформацію про запізнення, включаючи причини затримок, що дозволить оперативно реагувати та запобігати подібним ситуаціям у майбутньому.

Ці аналітичні дані сприятимуть прийняттю обґрунтованих рішень для оптимізації процесів, зменшення витрат часу та підвищення задоволеності клієнтів.

2. Дані для навчання

Статистика стане основою для навчання нових логістів, підвищення кваліфікації існуючих співробітників, а також для розробки ефективніших стратегій управління ланцюгами постачання, покращення процесів планування та оптимізації логістичних операцій. Завдяки аналізу реальних кейсів, система допоможе новим працівникам:

- Розуміти типові помилки — аналіз конкретних ситуацій, які призвели до затримок або втрати ефективності.
- Вчитися на успішних прикладах — перегляд найкращих практик, що сприяли швидкому виконанню завдань і задоволенню клієнтів.
- Покращувати планування — використовуючи дані про типові маршрути та середній час доставки, нові логісти зможуть краще планувати свою роботу та уникати проблемних сценаріїв.

Це створить основу для безперервного навчання та професійного розвитку персоналу, що підвищить їхню ефективність і впевненість у роботі.

3. Моніторинг роботи

Функція моніторингу дозволить керівникам оцінювати індивідуальну продуктивність кожного логіста на основі:

- Кількості виконаних замовлень — відображення обсягу роботи кожного співробітника.
- Часу на виконання задач — аналіз швидкості реагування та доставки.
- Якості обслуговування — показники задоволеності клієнтів на основі відгуків або анкетування.

Ця інформація допоможе у прийнятті зважених рішень щодо:

- Нарахування премій — мотивування найкращих співробітників.
- Додаткового навчання — виявлення працівників, яким потрібна підтримка чи додаткові тренінги.
- Оптимального розподілу обов'язків — правильного розподілу навантаження між логістами для підвищення загальної продуктивності.

Переваги впровадження

Завдяки функції аналітики та статистики, бот стане не лише інструментом для автоматизації, а й потужним навчальним ресурсом для логістичної команди. Це сприятиме прозорості в роботі, підвищенню ефективності та створенню культури постійного вдосконалення всередині компанії.

4.2.1 Інтеграція з додатковими платформами

У майбутньому планується розширення функціоналу бота шляхом інтеграції з популярними платформами для логістики, що значно підвищить ефективність роботи логістів і менеджерів. Інтеграція дозволить автоматизувати багато ключових процесів, оптимізувати пошук, моніторинг та обробку замовлень, а також створити більш зручну та універсальну екосистему для роботи.

Central Dispatch

Платформа Central Dispatch є однією з найпопулярніших у логістичній сфері для пошуку та організації лоудів. Інтеграція з ботом дозволить автоматично отримувати інформацію про доступні лоуди, оперативно підбирати оптимальні варіанти перевезень і формувати пропозиції для клієнтів. Це значно скоротить час на обробку інформації та зменшить ручну роботу логістів. Завдяки автоматизованому пошуку, можна буде уникати простоїв і підвищувати продуктивність, а також оптимізувати маршрути для максимального завантаження траків.

GPS-платформи

Автоматизація відслідковування руху траків за допомогою інтеграції з GPS-платформами дозволить в реальному часі моніторити місцезнаходження транспорту, що особливо важливо для ефективного планування маршрутів і вчасної доставки вантажів. Користувачі зможуть отримувати сповіщення про статус доставки, перебіг маршруту та потенційні затримки. Додатково, це допоможе менеджерам оптимізувати роботу водіїв, швидко реагувати на непередбачувані ситуації й покращити сервіс для клієнтів.

Платіжні системи

Для забезпечення зручності обробки фінансових операцій планується інтеграція з популярними платіжними системами, що дозволить автоматизувати процеси оплати за перевезення та послуги. Серед основних платформ для інтеграції:

- PayPal — одна з найпоширеніших міжнародних платіжних систем, що надає можливість швидко та безпечно здійснювати транзакції у різних валютах.
- Stripe — популярна платформа для обробки онлайн-платежів, яка дозволяє гнучко налаштовувати платіжні процеси та забезпечує високий рівень безпеки.
- Криптовалютні гаманці — інтеграція з платформами для обробки криптовалютних платежів забезпечить додаткову гнучкість для клієнтів, які використовують цифрові активи. Це відкриє нові можливості для міжнародних транзакцій без обмежень, пов'язаних із банківськими переказами чи конвертацією валют.

Переваги інтеграції

Завдяки інтеграції з цими платформами бот стане універсальним інструментом для логістів, який поєднує функціонал пошуку, моніторингу та обробки фінансових операцій в одному інтерфейсі. Це дозволить значно підвищити швидкість і точність роботи, а також зменшити ймовірність людських помилок.

У перспективі, додаткові інтеграції можуть включати партнерства з іншими інструментами для планування завантажень, аналітики даних і автоматизації звітності. Такий підхід допоможе створити комплексну екосистему, яка повністю задовольнить потреби логістичного бізнесу та підвищить конкурентоспроможність компаній на ринку.

4.2.2 Додаткові ідеї для розвитку

Окрім вищезазначених функцій, є й інші перспективні напрямки, які я планую розвивати:

1. Планування графіка водіїв: система, яка дозволить автоматично складати графіки роботи водіїв з урахуванням їхніх переваг, навантаження та законодавчих обмежень.

2. Інтеграція з клієнтським додатком: створення веб- чи мобільного додатка, через який клієнти зможуть самостійно бронювати послуги, відслідковувати статус своїх замовлень і отримувати персоналізовані пропозиції.

3. Розрахунок витрат: бот зможе автоматично оцінювати вартість доставки, враховуючи витрати на пальне, зарплату водія, плати за дороги тощо.

4. Розширення мовної підтримки: додавання кількох мов інтерфейсу для роботи з міжнародними клієнтами.

4.2.3 Використання сучасних технологій

Для реалізації цих функцій я планую використовувати:

1. Штучний інтелект: для оптимізації процесів, створення прогнозів і автоматизації складних завдань.

2. Мікросервісну архітектуру: це дозволить гнучко оновлювати систему, додаючи нові модулі без впливу на вже існуючий функціонал.

3. Хмарні сервіси: з метою забезпечення стабільної роботи системи навіть під час пікових навантажень.

4.2.4 Підсумки

Майбутня розробка бота спрямована на створення універсального інструмента для логістів, здатного автоматизувати більшість рутинних процесів, підвищити ефективність роботи і забезпечити зручність у користуванні. Завдяки новим функціям бот стане не лише помічником, а й повноцінним аналітичним центром для логістичних компаній. Ці розробки дозволять значно покращити якість послуг, зробити процеси більш прозорими, а також допомогти компаніям зменшити витрати та підвищити прибутковість.

4.3 Програмна реалізація телеграм бота

Сам скрипт запускає одночасно декілька асинхронних процесів, серед який такі як: Основний бот сповіщень, на який будуть надходити лоуди з сайті по збору інформації, другим є бот фільтрів, який створений для користувачів, щоб ті могли легко та швидко створити відповідні фільтри по запитам які їх інтересують, і в подальшому получати результат цих фільтрів у боті сповіщень. А також третів процес, це парсінг, який зупускає віртуальний брузер для імітації звичайного користувача в обхід анти-парсінг систмам, це є осна всього бота, так як він зчитує необхідну інформацію для подальшого її використання у ботах для обробки.

```
409  def main():  volod +1
410      try:
411          thread1 = threading.Thread(target=run_bot_notifications)
412          thread2 = threading.Thread(target=run_bot)
413          thread3 = threading.Thread(target=pars)
414
415          thread1.start()
416          thread2.start()
417          thread3.start()
418
419          thread1.join()
420          thread2.join()
421          thread3.join()
422
423      except Exception as e:
424          send_message_to_admin(
425              f"main\n\nError:\n\n{e}")
```

Рис. 4.1. Основна функція запуску

Також при запуску бота для створення фільтрів, також запускається Redis:

```
logger.info(f"Launching the main bot...")
start_redis_listener()

bot.infinity_polling()
```

Рис. 4.2. Одночасний запуск Redis з ботом

Функція start_redis_listener() запускає ще два процеса звязані з Redis, а точніше два прослуховування каналів.

Перший прослуховує канал “channel_filter” це для того, що коли користувач створює новий фільт, дані одразу потрапляли до бота через Redis, це є набагато швидчи ніж робити це через любу іншу базу даних, які будуть у любому випадку створювати затримки які будуть негативно впливати на оптимізацію бота і в наслідок чого і на думку логістів про даного бота.

Друге прослуховування, направлено на канал “channel_payments” він також створений для того, щоб користувач не очікував довго на обробку даних оплати, і користувач міг швидко розпочати свою роботу.

```
352  def start_redis_listener():  1 usage  cdnotifications
353      listener_thread_filter = threading.Thread(target=redis_listener_filter)
354      listener_thread_payments = threading.Thread(target=redis_listener_payments)
355
356      listener_thread_filter.daemon = True
357      listener_thread_payments.daemon = True
358
359      listener_thread_filter.start()
360      listener_thread_payments.start()
```

Рис. 4.3. Функція запуску Redis

Сам телеграм не надає доступ до абсолютно любого сайту через свій web app api, тому для того щоб дана функція почала коректно працювати, необхідно в Django settings ввести наступні дані:

```
142  CORS_ALLOWED_ORIGINS = [
143      "https://web.telegram.org",
144      "https://telegram.org",
145  ]
146
147  CORS_ALLOW_ALL_ORIGINS = True
148
149  CSP_DEFAULT_SRC = ("'self'", )
150  CSP_SCRIPT_SRC = ("'self'", "'unsafe-inline'", "https://telegram.org")
151  CSP_STYLE_SRC = ("'self'", "'unsafe-inline'")
152  CSP_IMG_SRC = ("'self'", "data:")
153  CSP_FONT_SRC = ("'self'", "https://fonts.gstatic.com")
154  CSP_CONNECT_SRC = ("'self'", "https://telegram.org")
```

Рис. 4.4. Конфігурація під телеграм

При локальній роботі даного бота все адекватно працює без перебоїв та помилок, однак при деплої коду на хостинг, виникає проблема інсталяції playwright, для цього необхідно було створено deploy config в якому прописана інсталяція playwright

```
web: gunicorn notifications.wsgi --log-file -
bot: PLAYWRIGHT_BROWSERS_PATH=0 python bot/bot.py
release: playwright install chromium
```

Рис. 4.5. Конфігурація запуску

Після чого було створено декілька змінних у середовищі heroku серед яких такий параметр як BUILDPACK_BROWSERS_INSTALL_PATH

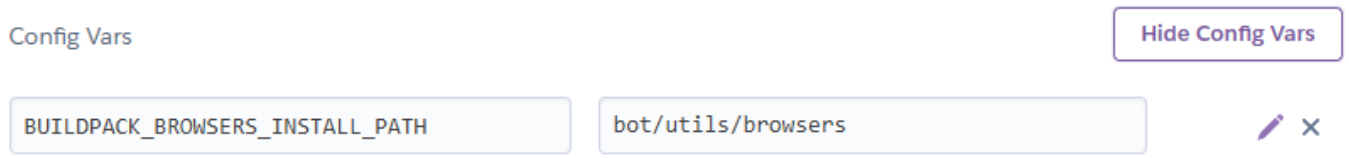

Рис. 4.6. Середовище шляхів Heroku

Після цих дій запуск віртуалдьного браузера на стороні heroku виглядає ось так:

```
heroku_current_dir = os.getcwd()  # This will get the current working directory
path = os.path.join(heroku_current_dir, 'bot', 'utils', 'browsers', 'chromium-1148', 'chrome-linux', 'chrome')
browser = p.chromium.launch_persistent_context(
    executable_path=path, headless=True, user_data_dir=user_data_dir
)
```

Рис. 4.7. Код запуску віртуального браузера

4.4 Програмна реалізаціє моделей з використанням ШІ

Програмна реалізація функції find_the_best_combination яка в результаті повертає найкращю комбінацію лоудів для логіста, за даними які користувача передавав програмі.

```
def find_the_best_combination(cars, combinations_option, truck_cars):
    """
    Знаходить найкращу комбінацію автомобілів, що відповідає заданим умовам.

    :type cars: dict
    :type combinations_option: dict
    :type truck_cars: list

    :return: combination, cars
    :rtype:
    """

    ids = combinations_option['ids']

    # Знаходить машини, які знаходяться в заданому радіусі
    posts = search_posts_in_radius(cars, combinations_option)

    block_cars = get_cars_with_ids(cars, ids.get('status-block'))

    # Отримання автомобілів зі статусом 'status-block'
    if ids.get('enable-status-add'):
        block_cars = block_cars + truck_cars

    posts = filter_cars_by_ids(posts, ids)

    current_combination_option = {'length': 0, 'height': 0, 'width': 0, 'weight': 0}

    if block_cars:
        for car in block_cars:
            current_combination_option["length"] += car["length"]
            current_combination_option["weight"] = car["weight"]
            if not current_combination_option["height"] or
current_combination_option["height"] < car["height"]:
                current_combination_option["height"] = car["height"]
            if not current_combination_option["width"] or
current_combination_option["width"] < car["width"]:
                current_combination_option["width"] = car["width"]

    if block_cars:
        posts = get_cars_on_the_same_path(block_cars[0], posts, combinations_option)

    # Пошук комбінацій автомобілів
    combinations = find_combinations(posts, combinations_option,
current_combination_option, block_cars)

    if not combinations:
        return block_cars, [], get_path_href(block_cars,
combinations_option['coords'])

    # Вибір найбільш прибуткової комбінації
    combination = get_most_profitable_combination(combinations, combinations_option)

    # Отримання списку автомобілів, що входять до комбінацій
    posts = get_combinations_posts(combinations)

    # information(combination, combinations option, cars)
    path_href = get_path_href(combination, combinations_option['coords'])

    return combination, posts, path_href
```

Рис. 4.8. Основна функція пошуку комбінацій

В даній функції, ШІ використовується для пошуку габаритів транспортного засобу, для пошуку найбільш вигідної комбінації коли необхідно проаналізувати трафік та дорогу якою буде найвигідніше пересуватись.

4.5 Висновок до розділу

У розділі "Апробація та результати експериментів" представлено результати практичного впровадження розроблених моделей і рішень. Проведені експерименти показали високу ефективність запропонованих підходів, що дозволило значно оптимізувати логістичні процеси. Аналіз результатів підтвердив, що використання моделей для маршрутизації та прогнозування підвищує точність планування та скорочує витрати.

Програмна реалізація телеграм-бота стала ключовим елементом інтерактивної взаємодії з логістами, забезпечуючи швидкий доступ до необхідної інформації. Реалізовані моделі з використанням штучного інтелекту продемонстрували здатність адаптуватися до динамічних змін і значно підвищувати ефективність управління логістикою.

Отримані результати підтверджують практичну цінність розроблених рішень і їхню придатність для використання у реальних умовах.

ВИСНОВКИ

У процесі виконання магістерської роботи були досягнуті всі поставлені цілі, що підтверджує актуальність обраної теми та ефективність використаних підходів. Детальний аналіз проблеми дозволив виявити основні потреби і виклики, що постають перед сучасними системами автоматизації, та запропонувати оптимальні рішення для їх подолання.

Розроблена система стала практичним інструментом, здатним вирішувати завдання підвищення ефективності та зручності в роботі. Під час розробки враховувались сучасні тенденції у відповідній сфері, включаючи використання інноваційних технологій і підходів, що забезпечило високу функціональність і адаптивність створеного продукту.

Проведені експерименти та тести підтвердили відповідність розробленої системи технічним і функціональним вимогам, а також її здатність інтегруватися в існуючу інфраструктуру користувачів. Практичні результати роботи підтвердили доцільність запропонованих рішень та їхню ефективність у реальних умовах.

Таким чином, виконана робота робить вагомий внесок у розвиток автоматизованих систем і створює перспективи для подальших досліджень і вдосконалення в цій галузі. Результати можуть бути корисними для подальшої наукової діяльності, а також для практичного використання в суміжних сферах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "System TMS Qguar". Quantum Software.
2. "Три великі американські компанії створюють спільну автотранспортну систему стандартів". Lading.ua.
3. "Модернізація логістики за допомогою штучного інтелекту". Habr.com.
4. "Heroku Playwright Buildpack". GitHub.
5."Оплати через Wayforpay". Wayforpay.com.
6. "Проблеми з конфіденційністю в сучасних браузерах". Mediacom.com.ua.
7. "Дискусії на тему сучасної логістики". DOU.ua.
8. "Навчальні матеріали з логістики". Edu.asu.in.ua.
9. "Playwright vs Selenium для веб-скрапінгу". Blog.apify.com.
10. "Playwright vs Selenium". BrowserStack.
11. "Що таке Django?". Amazon Web Services.
12. "Порівняння СУБД: MySQL, PostgreSQL, MSSQLServer". Data-b-i.com.
13. "Розвиток транспортної логістики в Україні". Confmanagement-proc.kpi.ua.
14. "Три найбільші виклики у транспортуванні". Logos3pl.com.
15. "Роль AI у розвитку стійких ланцюгів поставок". Georgetown Journal of International Affairs.
16. "Штучний інтелект у логістиці". Ogologistic.com.
17. "Транспортна логістика: посібник". Knute.edu.ua.
18. "Перспективи штучного інтелекту у логістиці". Trans.info.
19. "Роль штучного інтелекту в оптимізації логістичних процесів". Cargofy.ua.
20. "Топ 7 TMS систем". Stfalcon.com.
21. "GUAR TMS — дієвий інструмент управління транспортом". Bilyayivka.city.
22. "Топ EdTech AI стартапів 2024". Marketer.ua.
23. "Збірник тез конференції". Rep.nuos.edu.ua.
24. "Транспортна логістика: навчальний посібник". Nkkep.com.
25. "Як логістика може надати бізнесу додаткову цінність". Delo.ua.
26. "Збірка тез доповідей". Oarp.nau.edu.ua.
27. "Три найбільші виклики у транспортуванні". Logos3pl.com.

28. "Зростання цін на автогаз". Suspilne.media.
29. "Ціни на пальне в Україні". Hromadske.ua.
30. "Транспортна логістика: сучасний стан". Knute.edu.ua.
31. "Монографія: Розвиток транспортної логістики". Niss.gov.ua.
32. "Матеріали з логістики". Eprints.kname.edu.ua.
33. "Стандарти транспортної логістики". Dstu.dp.ua.
34. "Оптимізація доставки через GPS". Cargofy.ua.
35. "Штучний інтелект у логістиці". Ogologistic.com.
36. "Нейромережі для рітейлу". Neurosensey.com.
37. "Generative AI vs Predictive AI". Careers.epam.ua.
38. "Штучний інтелект в управлінні проектами". Bannerboo.com.
39. "Технології в рітейлі". PSM7.com.
40. "Штучний інтелект в закупівлях". Logist.fm.

ДОДАТКИ

Додаток А

Фрагмент з коду ботів

```
@bot.callback_query_handler(func=lambda callback: True)
def callback_message(callback):
    # filter set
    if callback.data.startswith('filters_set'):
        text = int(callback.data.split('_')[-1])
        filters_set(callback.message, text)

    # filter status
    if callback.data.startswith('filters_status'):
        text = int(callback.data.split('_')[-1])
        filters_change_status(callback.message, text)

    # filter delete
    if callback.data.startswith('filters_delete'):
        text = int(callback.data.split('_')[-1])
        filters_delete(callback.message, text)

    # filter close
    if callback.data.startswith('delete_message'):
        delete_message(callback.message)

    # activate account
    if callback.data.startswith('user_activate'):
        text = int(callback.data.split('_')[-1])
        user_activate(text)

    # subscription link
    if callback.data.startswith('subscription_link'):
        subscription_id = int(callback.data.split('_')[-1])
        user_id = int(callback.data.split(' ')[-2])
        generate_subscription_link(callback.message, user_id, subscription_id)

    # subscription response
    if callback.data.startswith('subscription_response'):
        user_name = callback.data.split('_')[-1]
        user_id = int(callback.data.split('_')[-2])
        response_status = callback.data.split('_')[-3]
        subscription_response(callback.message, user_id, user_name, response_status)

    # user list delete
    if callback.data.startswith('user_list_delete'):
        user_name = callback.data.split('_')[-1]
        user_id = int(callback.data.split('_')[-2])
        user_list_delete(callback.message, user_id, user_name)

    # user list
    if callback.data.startswith('user_list_show'):
        user_list_page = int(callback.data.split('_')[-1])
        user_list(callback.message, user_list_page)

    # user delete
    if callback.data.startswith('user_delete'):
        user_id = int(callback.data.split('_')[-1])
        user_delete(callback.message, user_id)

    # disable regular ordering
    if callback.data.startswith('disable_regular_ordering'):
        user_id = int(callback.data.split('_')[-1])
```

```
        question_disable_regular(callback.message, user_id)

    # back subscription status
    if callback.data.startswith('back_subscription_status'):
        user_id = int(callback.data.split('_')[-1])

        send_subscription_status(user_id, callback.message)

    # delete regular ordering
    if callback.data.startswith('delete_regular_ordering'):
        user_id = int(callback.data.split('_')[-1])

        delete_regular_ordering(callback.message, user_id, '')


def run_bot_notifications():
    global stop_threads
    while not stop_threads:
        try:
            logger.info(f"Launching a notification bot...")
            bot_notifications.infinity_polling()
        except Exception as e:
            send_message_to_admin(
                f"run_bot_notifications\n\nError:\n\n{e}"
            )
            logger.info(
                f"run_bot_notifications\n\nError:\n\n{e}"
            )

            time.sleep(5)
            logger.info("restart run_bot_notifications")


def run_bot():
    global stop_threads
    while not stop_threads:
        try:
            logger.info(f"Launching the main bot...")
            start_redis_listener()

            bot.infinity_polling()
        except Exception as e:
            send_message_to_admin(
                f"run_bot\n\nError:\n\n{e}")
            logger.info(
                f"run_bot\n\nError:\n\n{e}")

            time.sleep(5)

            logger.info("restart run_bot")


def pars():
    global stop_threads

    if os.path.exists(user_data_dir):
        logger.info('Delete browser data')
        shutil.rmtree(user_data_dir)
    else:
        logger.info('Browser data directory does not exist, skipping deletion')

    while not stop_threads:
        try:
```

```
            send_message_to_admin("pars")

            logger.info('Start browser')
            start_browser()

            logger.info('Central start')
            central_start()
        except Exception as e:
            send_message_to_admin(
                f"pars\n\nError:\n\n{e}")
            logger.info(
                f"pars\n\nError:\n\n{e}")

            time.sleep(5)

            logger.info("Restart pars")


def start_redis_listener():
    listener_thread_filter = threading.Thread(target=redis_listener_filter)
    listener_thread_payments = threading.Thread(target=redis_listener_payments)

    listener_thread_filter.daemon = True
    listener_thread_payments.daemon = True

    listener_thread_filter.start()
    listener_thread_payments.start()


def main():
    try:
        thread1 = threading.Thread(target=run_bot_notifications)
        thread2 = threading.Thread(target=run_bot)
        thread3 = threading.Thread(target=pars)

        thread1.start()
        thread2.start()
        thread3.start()

        thread1.join()
        thread2.join()
        thread3.join()

    except Exception as e:
        send_message_to_admin(
            f"main\n\nError:\n\n{e}")


if __name__ == "__main__":
    main()
```

Додаток Б

Фрагмент з коду парсера

```
def stop_parsing():
    global pars_status
    pars_status = False


def update_cookies():
    global browser, page

    p = sync_playwright().start()

    browser = p.chromium.launch_persistent_context(
        user_data_dir=r"C:\Users\volod\AppData\Local\Google\Chrome\User Data",
        headless=False,
        executable_path="C:/Program Files/Google/Chrome/Application/chrome.exe"
    )

    page = browser.new_page()

    page.goto(url)

    page.wait_for_timeout(30000)

    cookies = page.context.cookies()

    with open('cookies_new.json', 'w', encoding='utf-8') as file:
        json.dump(cookies, file, ensure_ascii=False, indent=4)

    central_update_authorization_key()

    page.wait for timeout(5000)


def create_new_cookies(new_cookies_code):
    global cookies_status, cookies_code

    cookies_status = True
    cookies_code = new_cookies_code


def start_browser():
    global browser, user_data_dir

    try:
        p = sync_playwright().start()

        # Default
        # browser = p.chromium.launch_persistent_context(
        #     user_data_dir=user_data_dir, headless=False)

        heroku_current_dir = os.getcwd()  # This will get the current working
directory
        path = os.path.join(heroku_current_dir, 'bot', 'utils', 'browsers',
'chromium-1148', 'chrome-linux', 'chrome')
        browser = p.chromium.launch_persistent_context(
            executable_path=path, headless=True, user_data_dir=user_data_dir
        )

        load_cookies(browser)
```

```
    except Exception as e:
        print(f"Error starting browser: {e}")
        send_message_to_admin(f"Error starting browser: {e}")


def save_browser_context(context, path):
    context.storage_state(path=path)
    print(f"Browser context saved to {path}")


def load_cookies(context):
    try:
        with open(cookies_path, 'r') as f:
            cookies = json.load(f)
            context.add_cookies(cookies)
    except FileNotFoundError:
        print("Cookies file not found. Continuing without loading cookies.")


def log_request(request):
    global headers
    if "https://prod-search-app-bff.awscal2.manheim.com/api/saved-
searches/recent?limit=3" in request.url:
        headers['authorization'] = request.headers['authorization']


def central_update_authorization_key():
    global page, headers

    page.on("request", log_request)

    page.reload()

    page.wait_for_selector(
        'xpath=//*[@id="single-spa-
application:@centraldispatch/search"]/div/div/div/div/div[2]/div[1]/div[1]/div[1]/div
[2]/div[1]', timeout=30000
    )
```