Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(навчально-науковий інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня магістра

Здобувача вищої освіти Каніболоцького Володимира Володимировича
(ПІБ)
академічної групи 123м-24 -1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)
за освітньо-професійною програмою «Комп'ютерна інженерія»
(офіційна назва)

на тему «Обґрунтування структури та параметрів комп'ютерної системи логістичної компанії з використанням чатботу замовлень та комунікації з клієнтом»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Цвіркун Л.І.			
розділів:				
синтез системи	проф. Цвіркун Л.І.			
розроблення програмного забезпечення	ас. Панферова Я.В.			
Рецензент				
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2025

ЗАТВЕРДЖЕНО:

завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)

_____ В.В. Гнатушенко
(підпис) (ініціали, прізвище)

«__ ___» _____ 2025 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня магістра
(бакалавра, магістра)

здобувача вищої освіти Каніболоцького В.В. **академічної групи** 123м-24-1
(прізвище та ініціали) (шифр)

спеціальності 123 Комп'ютерна інженерія

за освітньою-професійною програмою «Комп'ютерна інженерія»
(офіційна назва)

на тему «Обґрунтування структури та параметрів комп'ютерної системи логістичної компанії з використанням чатботу замовлень та комунікації з клієнтом»,
затверджену наказом ректора НТУ «Дніпровська політехніка» від 13 жовтня 2025 р. №1165-с

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	10.10.2025
Теоретичний	Обґрунтувати теоретичну базу для використання чатботу замовлень та комунікації з клієнтом логістичної компанії	24.10.2025
Синтез системи	Визначаються цілі системи, проводиться обґрунтування вибору бази даних, формуються вимоги до системи та обирається апаратне забезпечення	14.11.2025
Розроблення програмного забезпеченя	Розробка програмного забезпечення чатботу замовлень та комунікації з клієнтом на базі комп'ютерної системи логістичної компанії	28.11.2025
Експериментальний розділ	Проведення і обробка результатів експериментів з використанням чатботу замовлень та комунікації з клієнтом	05.12.2025

Завдання видано _____ проф. Л. І. Цвіркун
(підпис керівника) (ініціали, прізвище)

Дата видачі 05 вересня 2025 р.

Дата подання до екзаменаційної комісії 10.12. 2025 р.

Прийнято до виконання _____ Каніболоцький В.В.
(підпис здобувачавищої освіти) (ініціали, прізвище)

РЕФЕРАТ

Пояснювальна записка: 130 с., 73 рис., 9 табл., 1 дод., 29 джерел.

БАЗА ДАНИХ, КОМП'ЮТЕРНА СИСТЕМА, ЛОГІСТИЧНА КОМПАНІЯ, MONGODB, СЕРВЕР, ЧАТБОТ, TELEGRAM, PYTHON

Об'єкт дослідження: комп'ютерна система логістичної компанії з чатботом замовлень та комунікації з клієнтом.

Мета роботи: розробка чатботу замовлень та комунікації з клієнтом на базі комп'ютерної системи логістичної компанії та обґрунтування використаних програмних рішень, які забезпечують безперебійну експлуатацію системи.

Методи дослідження: застосовувались методи збору та аналізу даних про навантаження на сервер і БД в процесі автоматизації взаємодії з клієнтами, що включає базу даних MongoDB та чатбота в месенджері Telegram.

У пояснювальній записці представлено аналіз комп'ютерної системи логістичної компанії та можливість інтеграції чатботу для автоматизації взаємодії з клієнтами. Аналіз отриманих результатів дозволив визначити пріоритетні напрямки подальшого дослідження.

В теоретичному розділі проведений аналіз мов програмування з метою створення бота для роботи з клієнтами та описана методологія його розробки.

У розділі синтезу приведені основні цілі, сформульовані технічні вимоги до системи, визначено структурну схему комплексу технічних засобів мережі, а також обґрунтовано застосування апаратних засобів.

У розділі розроблення програмного забезпечення виконано опис алгоритму, згідно якого розроблена програмне забезпечення.

В експериментальному розділі проведено експерименти щодо впровадження чатботу замовлень та зв'язку з клієнтами і перевірено його функції на практиці.

Практична значимість отриманих результатів полягає в можливості використанням чатботу замовлень та комунікації з клієнтом для логістичної компанії.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

API (Application Programming Interface) – прикладний програмний інтерфейс;

Telegram API – програмний інтерфейс платформи Telegram;

AI (Artificial Intelligence) – штучний інтелект;

DB/БД (database) – база даних;

MongoDB – документо-орієнтована нереляційна база даних;

DHCP – протокол динамічної конфігурації вузлів;

ДБЖ – джерело безперебійного живлення;

Чатбот (chatbot) – програма, що імітує спілкування з людиною в текстовому або голосовому форматі;

КС – комп'ютерна система;

ОП – оперативна пам’ять;

ІТ – Information Technology, інформаційні технології;

ПЗ – програмне забезпечення;

FSM (Finite State Machine) – кінцевий автомат станів;

JSON (JavaScript Object Notation) – текстовий формат обміну даними;

ГМ – головне меню

REST-API – програмний інтерфейс веб-служб, що базується на архітектурному стилі REST (Representational State Transfer).

ВСТУП

Сучасні інформаційні технології стали невід'ємною складовою ефективного функціонування логістичних підприємств, забезпечуючи оптимізацію маршрутів, управління складськими запасами, координацію транспортних потоків та оперативну взаємодію з клієнтами. Особливої актуальності набуває автоматизація взаємодії з клієнтами, що забезпечує цілодобову доступність послуг, швидкість обробки запитів та зменшення операційних витрат.

Чатбот в месенджері Telegram як інструмент автоматизації клієнтського сервісу логістичного підприємства забезпечує миттєву взаємодію з клієнтами, автоматизацію процесів прийняття замовлень, відстеження вантажів, надання інформації про послуги та вирішення типових запитів без участі оператора.

Мета і завдання дослідження. Метою роботи є розробка чатботу замовлень та комунікації з клієнтом на базі комп'ютерної системи логістичної компанії та обґрунтування використаних програмних рішень, які забезпечують безперебійну експлуатацію системи, що передбачає проведення аналізу та дослідження можливостей програмних та технічних рішень у цьому напрямку.

Об'єкт дослідження – комп'ютерна система логістичної компанії з чатботом замовлень та комунікації з клієнтом.

Предмет дослідження – процес використання чатботу замовлень та комунікації з клієнтом.

Методи дослідження. Для досягнення поставленої мети використовувалися методи теорії масового обслуговування, експериментального тестування, аналізу літературних джерел та документації, метод порівняльного аналізу, теорія збору та аналізу даних, теорія математичної статистики.

Ідея роботи: впровадження автоматизованої системи взаємодії з клієнтами на базі чатботу в месенджері Telegram, що дозволить оптимізувати бізнес-процеси, знизити навантаження на персонал та покращити якість обслуговування клієнтів.

Наукові положення: встановлено, що для ефективної роботи чатботу замовлень та взаємодії з клієнтами, необхідно оптимізувати серверні ресурси,

підготувати відповідне ПЗ дослідження спроможності системи підтримувати додаткові навантаження на сервер. Розроблена структура комп'ютерної системи з інтегрованим чатботом забезпечує цілодобову доступність сервісу для клієнтів та співробітників логістичного підприємства з рівнем відмовостійкості 99.5%. Доведено, що впровадження NLP-алгоритмів для розпізнавання намірів користувачів підвищує точність автоматичних відповідей до 85-90%, що зменшує необхідність залучення операторів на 40%.

Наукові результати

Обґрунтовано доцільність застосування MongoDB як системи управління базами даних для чатботу логістичної компанії. Ця NoSQL-база забезпечує гнучке зберігання інформації про замовлення, маршрути доставки та клієнтські дані, а також надає швидкий доступ до документів та файлів у процесі обробки запитів користувачів. Впровадження інструментів моніторингу Prometheus та Grafana дозволяє в режимі реального часу відстежувати навантаження на систему, аналізувати продуктивність чатботу та оперативно реагувати на збільшення кількості звернень клієнтів у пікові періоди.

Обґрунтованість і достовірність наукових положень, висновків і рекомендацій забезпечена результатами аналітичних і експериментальних досліджень із залученням програмних засобів для моніторингу показників навантаження на MongoDB і серверну інфраструктуру логістичної компанії.

Практичне значення отриманих результатів полягає у розробці програмного забезпечення для чатботу замовлень та комунікації з клієнтами логістичної компанії. Розроблена система дозволяє автоматизувати повний цикл комунікації з клієнтами: від реєстрації замовлення до надання актуальної інформації про статус доставки у режимі реального часу.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Сучасний стан та тенденції цифровізації логістичної галузі України

Український логістичний сектор функціонує в умовах безпрецедентних викликів. Пандемія COVID-19, що розпочалася у 2020 році, стала першим серйозним випробуванням для галузі. Однак події 2022 року та подальша ескалація створили ще складніші умови для ведення бізнесу. Кожен наступний рік – 2023, 2024 та поточний 2025 – вимагав від логістичних підприємств надзвичайної адаптивності та стійкості. В таких умовах виживають лише найбільш ефективні та гнучкі підприємства. Експерти галузі відзначають відсутність активного входження іноземних операторів на український ринок у нинішніх реаліях, що змушує вітчизняні компанії зосереджуватися на підвищенні операційної ефективності та ретельному відборі бізнес-партнерів [20].

Незважаючи на комплекс викликів, спричинених війною, − руйнування транспортної інфраструктури, критичну нестачу кваліфікованого персоналу та систематичні перебої в електропостачанні, − вітчизняний транспортно-логістичний комплекс демонструє стійкість і зберігає тенденцію до розвитку. За станом на жовтень 2025 року в державі налічується 260,4 тис. активних господарюючих суб'єктів (підприємств та ФОП) транспортно-логістичної галузі, які здійснюють безперервну комерційну діяльність. Відповідно до результатів проведеного дослідження, третій квартал 2025 року характеризувався подвоєнням кількості зареєстрованих ФОП у сфері логістики – 8,3 тис. новостворених суб'єктів підприємництва проти 4,1 тис. попереднього періоду. Одночасно зафіксовано зростання реєстраційної активності компаній на рівні 11,7% у порівнянні з II кварталом. При цьому відбулося зменшення обсягів ліквідації бізнесу: свою діяльність завершили лише 34 юридичні особи та 4,3 тис. підприємців – фізичних осіб, що значно нижче показників першого кварталу поточного року (рисунок 1.1.) [16].

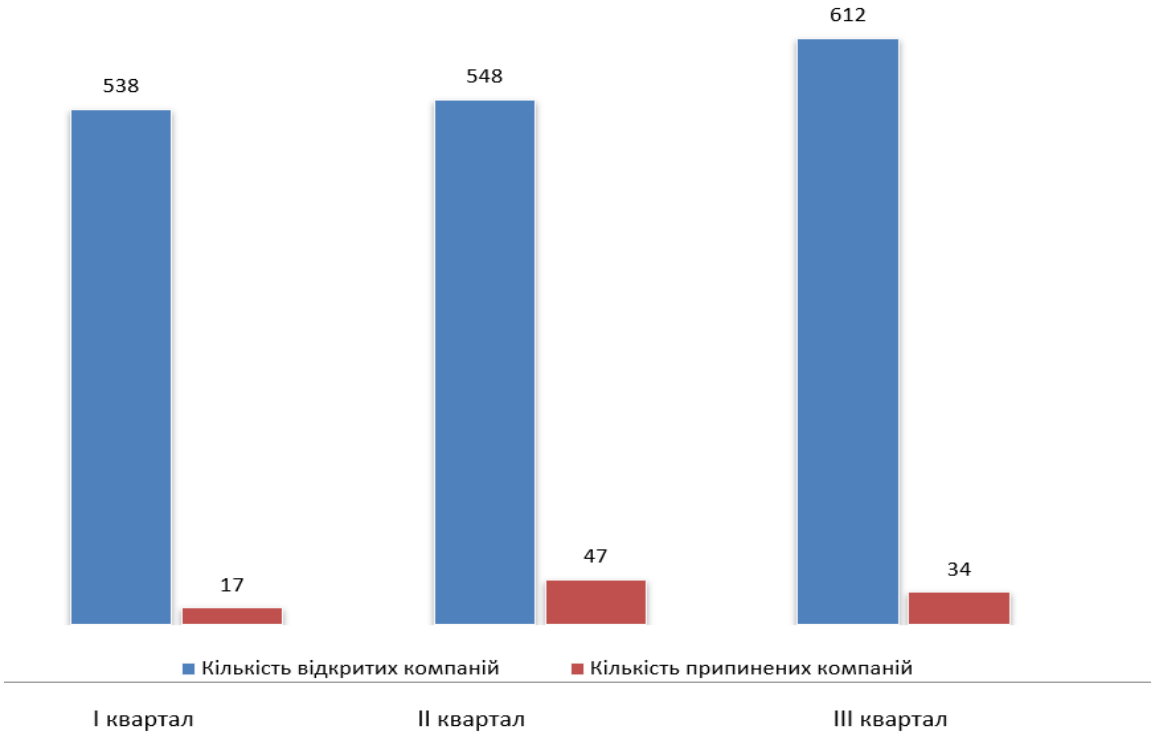

Рисунок 1.1 – Динаміка відкриття та закриття компаній у секторі «Транспорт і логістика України» у 2025 році [16]

П'ятнадцять найбільших операторів перевезень та логістичних послуг України продемонстрували сумарний дохід на рівні 233 млрд грн за підсумками 2024 року. Домінуюче становище займає «Укрзалізниця» із фінансовим показником 108,1 млрд грн, тоді як «Нова Пошта» та «Укрпошта» зафіксували доходи 48,5 млрд грн і 13,6 млрд грн відповідно [18].

Сучасні логістичні компанії послідовно інтегрують цифрові технології для оптимізації бізнес-операцій, зокрема програмне забезпечення класу WMS для управління складом, транспортні платформи TMS, засоби відстеження вантажів у режимі реального часу та інструментарій електронного документообігу, CRM-платформи для комунікації з клієнтами. Інформаційно-технологічна трансформація логістичних процесів охоплює різні аспекти діяльності підприємства: від координації логістичних ланцюгів до комунікації зі споживачами та раціоналізації операційної діяльності**.** Разом із тим, інтеграція IT-рішень формує нові виклики для логістичних підприємств. Компанії мають освоювати інноваційні інструменти, такі як штучний інтелект, BigData та інтернет речей, щоб підвищити

ефективність своєї роботи та зберегти конкурентоспроможність у динамічних умовах.

Відповідно до таблиці 1.1 чатботи у 2025 році використовують 55% логістичних компаній України, а оскільки кожен показник відображає частку компаній, що використовують конкретний канал незалежно від інших, сума перевищує 100% – і саме це свідчить про те, що сучасні компанії застосовують декілька каналів комунікації одночасно, що підтверджує актуальність розробки чатботу як складової комплексного підходу до цифровізації клієнтського сервісу.

Таблиця 1.1 – Динаміка впровадження каналів комунікації у логістичних компаній України за період 2000-2025рр.

Рік	Телефон,%	Emeil,%	Веб-сайт,%	Месенджери, %	Чатботи, %	Мобільні додатки,%	AI-системи,%
2000	95	5	0	0	0	0	0
2005	90	10	5	0	0	0	0
2010	80	25	20	5	0	0	0
2015	70	40	50	30	2	10	0
2020	60	45	70	60	15	35	5
2025	50	50	85	70	45	55	20

Одним із перспективних напрямів цифровізації логістичних процесів є автоматизація комунікації з клієнтами підприємства. Незважаючи на значні досягнення у впровадженні WMS та TMS, управління клієнтськими запитами та інформування споживачів про статус замовлень у логістичному центрі часто залишається недостатньо автоматизованим процесом, що створює додаткові витрати часу та ресурсів. У цьому контексті актуальним є дослідження можливостей впровадження спеціалізованого чатботу в месенджері Telegram для автоматизації обслуговування клієнтів та оптимізації процесу управління комунікаціями логістичного центру.

1.2 Характеристика об’єкту дослідження

Об’єктом дослідження є логістична компанія, яка обслуговує як приватних осіб, так і бізнес-клієнтів та спеціалізується на універсальних вантажних перевезеннях по території України, забезпечуючи транспортування широкого спектру товарів загального призначення. Компанія здійснює доставку побутових вантажів (меблі, техніка, електроніка), будівельних матеріалів (цегла, суміші, метал), комерційних товарів (одяг, текстиль, канцелярія), а також виробничого та офісного обладнання. Крім того, компанія надає послуги складського зберігання та обробки вантажів та консультаційні послуги з логістики.

Технічні можливості підприємства. Автопарк компанії налічує транспортні засоби різної тоннажності: легкі комерційні фургони (до 1.5 т), середньотоннажні вантажівки (3-5 т) та великотоннажні автомобілі (10-20 т).

Треба відмітити, що компанія не здійснює перевезення вантажів, які потребують температурного контролю, наливних речовин та небезпечних матеріалів через відсутність спеціалізованого транспорту.

Головний офіс підприємства розташований за адрессою: м. Дніпро вул. Набережна Заводська. Чисельність персоналу – 41 працівник. Період діяльності на ринку – з 2015 року.

Існуюча комп'ютерна мережа підприємства включає файл-сервер та інтернет-сервер, що забезпечують базові потреби в обробці та зберіганні інформації. Кожна робоча станція має доступ до серверів та підключення до мережі Інтернет. Основними видами трафіку в мережі є локальний обмін даними, робота з файл-сервером та доступ до інтернет-ресурсів.

Розробка комп'ютерної системи для логістичної компанії з використанням сучасних мережних технологій потребує попереднього аналізу структурних підрозділів, які планується об'єднати в мережу.

Логістична компанія має лінійно-функціональну організаційну структуру управління, яка характеризується чіткою ієрархією, розподілом сфер відповідальності та принципом єдиноначальності (рисунок 1.2).

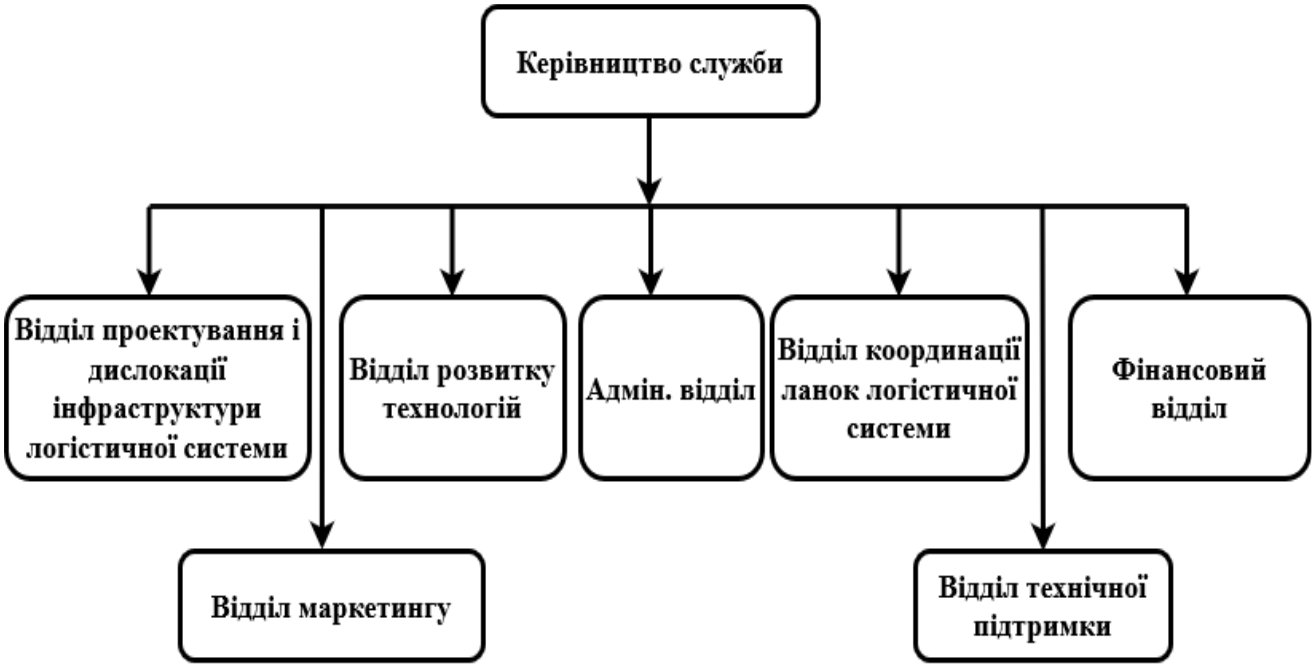

Рисунок 1.2 – Організаційна структура логістичної компанії

Діяльність структурних підрозділів є спеціалізованою і визначається їхніми основними функціональними характеристиками.

Організаційно-управлінська структура логістичного підприємства побудована за трьохрівневим принципом:

1) стратегічна ланка управління:

– генеральний директор – очолює підприємство та відповідає за загальне керівництво діяльністю, стратегічне планування та прийняття ключових рішень;

– заступник директора – координує роботу функціональних підрозділів та забезпечує виконання стратегічних завдань;

2) тактична ланка управління:

– начальник відділу логістики – відповідає за планування та координацію транспортних процесів;

– начальник комерційного відділу – забезпечує взаємодію з клієнтами, укладання договорів та розвиток клієнтської бази;

– головний бухгалтер – здійснює фінансовий облік та контроль;

– начальник відділу кадрів – управляє персоналом та кадровими процесами;

– IT-менеджер – відповідає за функціонування комп’ютерної системи та технічну підтримку;

3) операційна ланка управління:

– диспетчери – координують рух транспорту та контролюють виконання рейсів;

– водії – здійснюють безпосередньо транспортні операції;

– вантажники – забезпечують навантажувально-розвантажувальні роботи;

– менеджери з обслуговування клієнтів - приймають замовлення та консультують клієнтів;

– оператори складу - контролюють складські операції.

Основні бізнес-процеси підприємства:

– прийом та обробка замовлень від клієнтів;

– планування маршрутів та розподіл транспортних засобів;

– виконання транспортних операцій;

– контроль якості послуг та взаємодія з клієнтами;

– фінансове планування та облік.

Проблемні аспекти існуючої системи обслуговування клієнтів:

– обмежений час роботи операторів (робочі години);

– високе навантаження на персонал при обробці типових запитів;

– відсутність можливості миттєвого отримання інформації про статус замовлення;

– необхідність телефонного дзвінка для отримання базової інформації про послуги;

– складність відстеження вантажів для клієнтів.

Впровадження чатботу в месенджері Telegram в комп'ютерну систему для замовлень та комунікації з клієнтом дозволить забезпечити цілодобову доступність сервісу оформлення заявок, суттєво скоротити час обробки замовлень завдяки автоматизації збору даних, зменшити навантаження на диспетчерську

службу, мінімізувати ймовірність помилок при введенні інформації, підвищити рівень задоволеності клієнтів через зручний інтерфейс взаємодії, а також забезпечити прозорість логістичних процесів завдяки можливості відстеження статусу замовлення у режимі реального часу.

План приміщення головного офісу та офісного приміщення філії зображено на рисунках (рисунки 1.3 – 1.4).

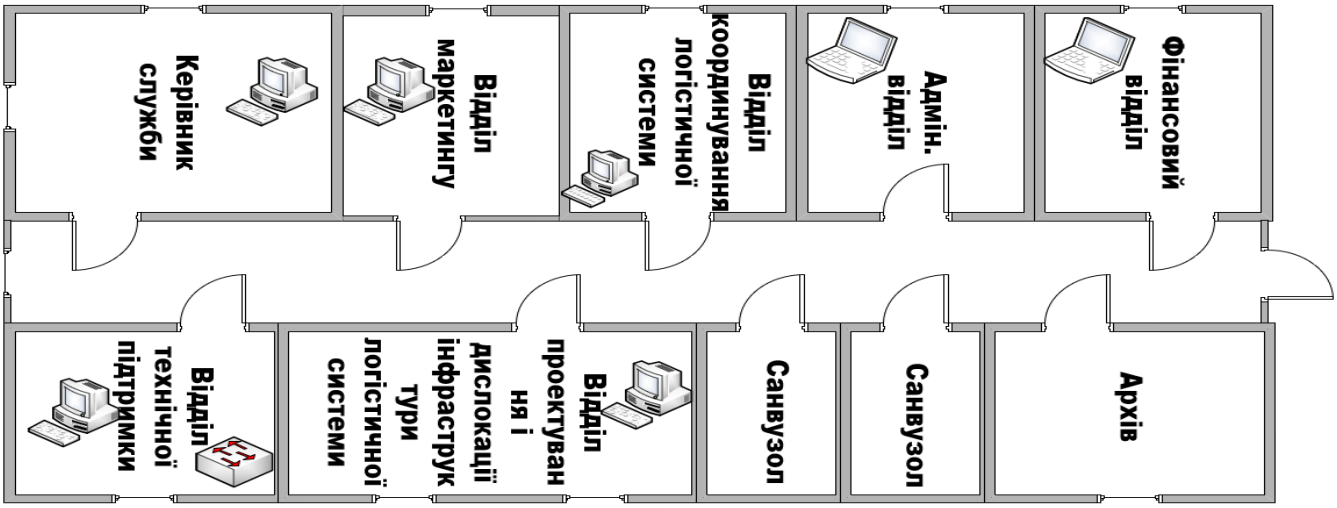

Рисунок 1.3 – План приміщення головного офісу

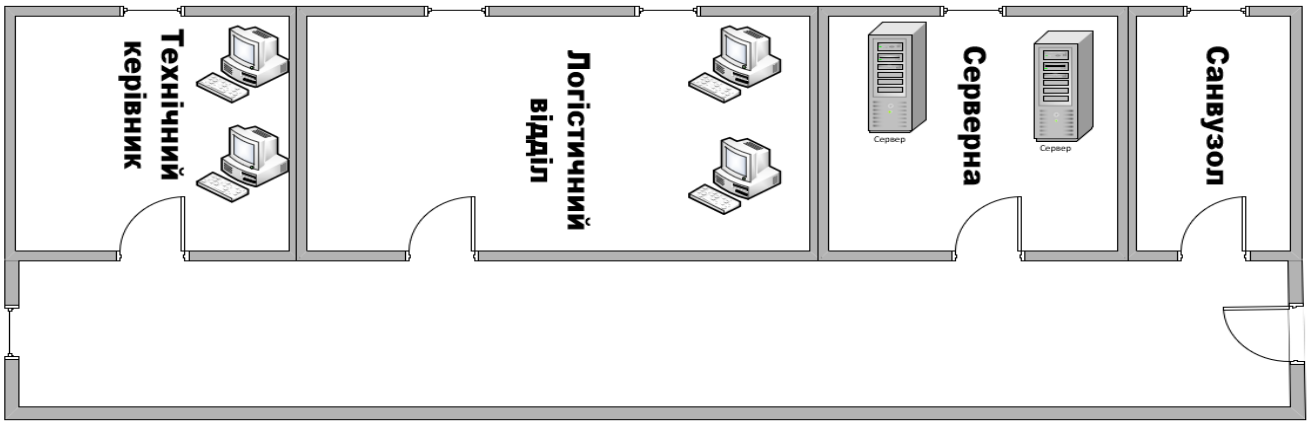

Рисунок 1.4 – План офісного приміщення філії

Перелік підсистем, їх призначення й основні характеристики:

– Application layer: загалом виконує завдання з управління мережевою політикою та її моніторінгу з інтеграцією з Telegram;

– Control layer: сервер, на якому розміщений Network Controller, і з якого здійснюється керування та моніторинг мережних пристроїв;

– Infrastructure layer: локальна комп'ютерної мережа (ЛКМ) та мережеві пристрої, відповідальні за пересилання трафіку.

Як уже згадано вище, згідно організаційної структури Infrastructure layer ЛКМ логістичного підприємства складається з восьми робочих груп (далі Підсистем):

– робоча група №1 – керівництво;

– робоча група №2 – адміністративний відділ;

– робоча група №3 – фінансовий відділ;

– робоча група №4 – відділ проектування і дислокації інфраструктури логістичної системи;

– робоча група №5 – відділ координації ланок логістичної системи;

– робоча група №6 – відділ маркетингу;

– робоча група №7 – відділ розвитку технологій;

– робоча група №8 – відділ технічної підтримки.

Робочі групи відокремлені в окремі 5 локальних мереж та підтримують кількість вузлів з майбутнім розширенням.

Комп'ютерна система логістичної компанії є об’єктом впровадження інтегрованого чатботу замовлень та комунікації з клієнтом в месенджері Telegram.

До складу приміщень компанії входять: головний офіс, що знаходиться в м. Дніпро, вул. Набережна Заводська (приміщення на першому поверсі в 260 м2) та філія за адресою: Проспект Б. Хмельницького (приміщення на першому поверсі в 75 м2).

В будівлі головного офісу розташовані підрозділи зі своїми структурними групами: керівництво, адмінвідділ, фінансовий відділ (в т.ч. відділ закупівель), відділ координації ланок логістичної системи, відділ проектування і дислокації інфраструктури логістичної системи, відділ маркетингу, технічний відділ. В приміщені будівлі філії розташований підрозділ маркетингу (рисунок 1.5).

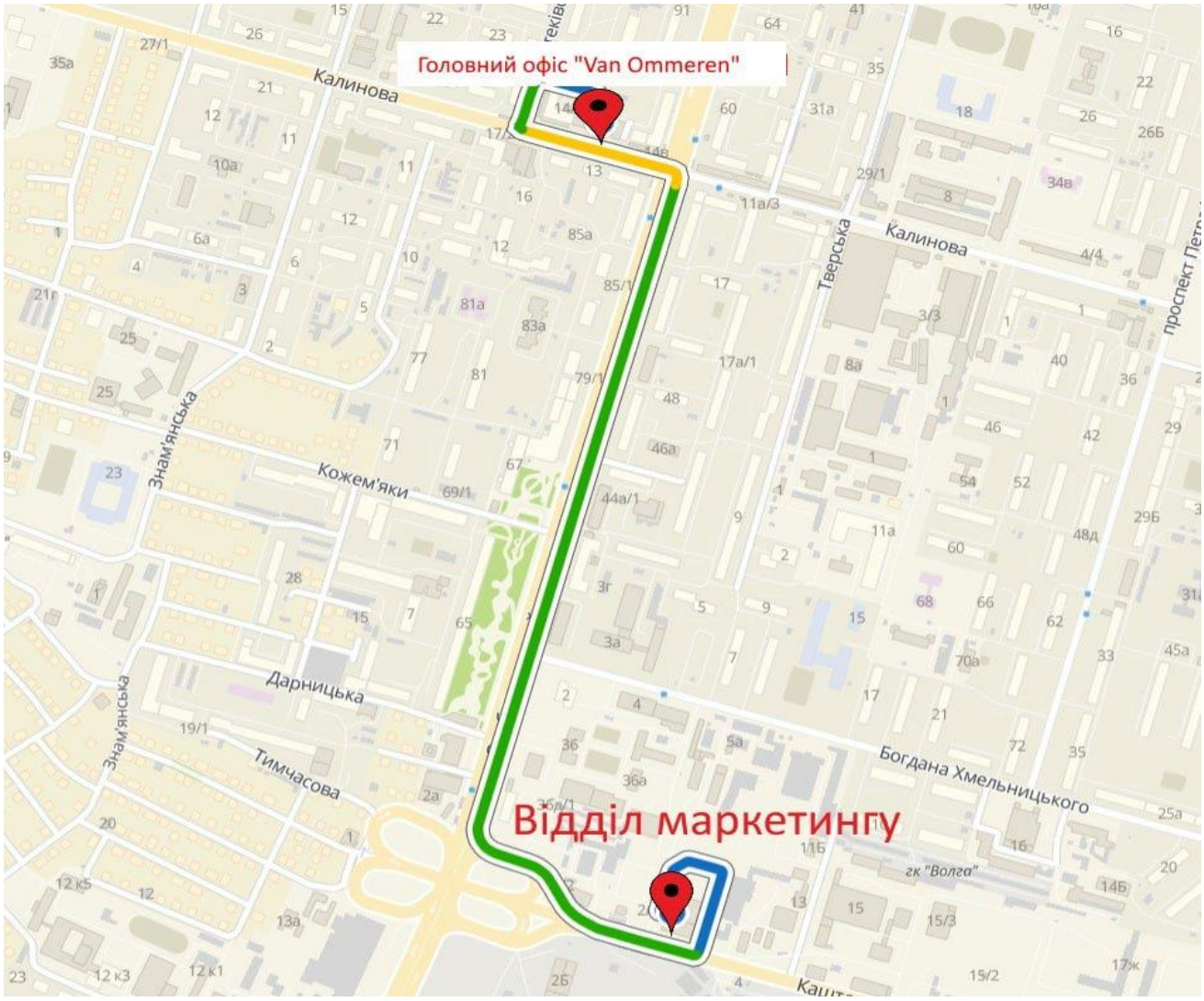

Рисунок 1.5 – Схема розташування об’єктів логістичної компанії

1.3 Проблеми сучасних комп’ютерних систем

Хоча інформаційні технології демонструють стрімкий розвиток, комп'ютерні системи стикаються з рядом перешкод, що впливають на ефективність їх роботи, надійність та безпеку. Вони особливо гостро проявляються у корпоративному середовищі, де від стабільного функціонування IT-інфраструктури залежить успішність бізнес-процесів.

Однією з найбільш критичних технічних проблем у сучасних комп’ютерних системах є масштабування та продуктивність. Зростання обсягів даних та кількості користувачів створює навантаження, з яким традиційні архітектури не завжди здатні впоратися ефективно, що призводить до уповільнення роботи систем, збільшення часу відгуку та зниження якості користувацького досвіду.

Коли в підприємстві використовуються різні програми та технології, які погано "спілкуються" між собою, виникають проблеми інтеграції різнорідних систем. Адже старі системи часто не сумісні з новими, що створює розрив в обміні даними та ускладнює автоматизацію робочих процесів.

Критично важливим аспектом залишається надійніть системи: програмні помилки, відмова обладнання, перебої в електропостачанні та мережеві збої можуть призвести до значних фінансових втрат та репутаційних ризиків.

Кібератаки –один з найактуальніших викликів безпеки у сфері IT-систем. Зростання кількості кібератак, включаючи DDoS-атаки, вірусні програми, фішинг та соціальну інженерію, вимагає постійного вдосконалення систем захисту. Особливу небезпеку становлять атаки на критично важливу інфраструктуру та персональні дані клієнтів.

Проблеми користувацького досвіду суттєво впливають на ефективність роботи з комп'ютерними системами. Складні та незрозумілі інтерфейси змушують користувачів витрачати багато часу на освоєння системи замість виконання основних робочих завдань, що також впливає на продуктивність праці та загального незадоволення співробітників від роботи з технологіями.

Недостатня адаптивність до потреб користувачів проявляється в тому, що багато систем не враховують специфіку роботи різних категорій користувачів та не забезпечують персоналізації робочого середовища.

У контексті логістичних компаній сучасні комп'ютерні системи стикаються з додатковими специфічними викликами:

– інтеграція з численними зовнішніми системами включає взаємодію з митними службами, банківскими установами, транспортними компаніями та клієнтськими системами, що створює складність в управлінні даними;

– реальний час обробки даних критично важливий для відстеження вантажів, управління маршрутами та координації логістичних операцій;

– мобільність та доступність систем для водіїв, експедиторів та співробітників вимагає розробки адаптивних рішень, які працюють в умовах нестабільного інтернет-з'єднання.

Для подолання зазначених викликів застосовуються та реалізуються різноманітні рішення, зокрема використання хмарних технологій, мікросервісної архітектури, штучного інтелекту у цифровізації процесів, а також впровадження інноваційних практик розробки ПЗ**.**

Окрема увага приділяється розробці користувацьких інтерфейсів, зокрема чатботів та мобільних додатків, які дозволяють спростити взаємодію користувачів з комп'ютерними системами та підвищити ефективність їх роботи.

1.4 Постановка завдання дослідження

Метою нашого дослідження є обґрунтуванні структури та параметрів комп’ютерної системи логістичної компанії та можливості впровадження та використанням чатботу замовлень та комунікації з клієнтом.

Завданням кваліфікаційної роботи є обґрунтування параметрів і структури комп'ютерної системи логістичної компанії та її спроможності щодо забезпечення можливості впровадження та використання чатботу в месенджері Telegram замовлень та комунікації з клієнтом, що передбачає здійснення аналітичного огляду та експериментального дослідження можливостей програмних та технічних рішень у цьому напрямку.

Для вирішення поставленої мети в роботі слід виконати наступні завдання:

– проаналізувати та сформулювати вимоги до комп’ютерної системи та чатботу замовлень та комунікації з клієнтом компанії;

– обґрунтувати параметри і структуру комп'ютерної системи та її спроможності в забезпеченні можливостей впровадження чатботу замовлень та комунікації з клієнтом;

– визначити мінімальні технічні вимоги до серверного обладнання для забезпечення стабільної роботи чатботу при обслуговуванні розрахункової кількості клієнтів;

– оцінити надійність та відмовостійкість серверної інфраструктури для забезпечення цілодобової роботи чат-бота взаємодії з клієнтами;

– провести порівняльний аналіз сучасних СУБД та обґрунтувати вибір MongoDB для зберігання даних клієнтів, замовлень та історії взаємодії через чатбот;

– обґрунтувати вибір месенджера Telegram як платформи для реалізації чат-бота та визначити оптимальний режим взаємодії з API;

– проаналізувати навантаження на серверну інфраструктуру при обробці клієнтських запитів та обміні даними через Telegram-бот;

– на основі проведених досліджень обґрунтувати технічну можливість впровадження розробленого чатботу в месенджері Telegram в існуючу IT-інфраструктуру логістичної компанії.

Розроблений чатбот має забезпечувати цілодобову доступність, високу швидкість обробки звернень, зручність використання для клієнтів різних категорій, відповідність вимогам захисту персональних даних та можливість інтеграції з розширеним функціоналом у майбутньому. Виконання поставлених завдань дозволить створити сучасне технологічне рішення, яке суттєво підвищить ефективність роботи з клієнтами та оптимізує бізнес-процеси логістичної компанії.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Чатботи в месенджері Telegram

Telegram є однією з найпопулярніших платформ миттєвого обміну повідомленнями, яка завдяки своїй відкритості та потужному API стала провідною платформою для розробки корпоративних ботів. Платформа надає розробникам широкі можливості для створення автоматизованих рішень, які ефективно інтегруються з бізнес-процесами компаній. Кількість щомісячних активних користувачів в Telegram (станом на серпень 2023 року) перевищила 800 млн.чол., в тому числі в країнах ЄС близько 41 млн. чол. [14].

Telegram-бот – це чатбот, що функціонує в екосистемі месенджера Telegram та взаємодіє з користувачами через Telegram Bot API.

Розрізняють базові чатботи, які працюють за принципом готових шаблонів відповідей, та прогресивні системи з елементами штучного інтелекту. Ці розумні системи еволюціонують у процесі використання, підвищуючи точність та релевантність своїх відповідей завдяки безперервному аналізу вхідних даних [19].

Чатбот являє собою програмне забезпечення, здатне відтворювати процес спілкування між людьми через текстові або голосові повідомлення. Іншими словами: це програмне забезпечення, що забезпечує автоматизацію різноманітних операцій і підвищує ефективність взаємодії користувачів з цифровими сервісами. Розроблені для месенджера Telegram, такі боти можуть опрацьовувати різні типи запитів: від надсилання актуальної інформації до здійснення пошукових запитів і навіть комерційних операцій. Ключовою особливістю боту є здатність генерувати автоматизовані відповіді на команди, введені користувачем. При цьому програмний модуль функціонує безпосередньо в середовищі Telegram, відтворюючи поведінкові патерни реального співрозмовника, що забезпечує інтуїтивно зрозумілий та ергономічний досвід використання [15].

Telegram Bot API забезпечує доступ до функціональності платформи через HTTP-інтерфейс, що дозволяє створювати боти практично на будь-якій мові програмування. Це робить платформу універсальним інструментом для

автоматизації взаємодії з клієнтами в різних галузях, включаючи логістику та транспорт. [15].

Розглянемо роботу ботів в Telegram. Отже, бот - це окрема програма, написана на різних мовах програмування на нашому окремому сервері. Алгоритм взаємодії користувача з ботом реалізується у такий спосіб: користувач надсилає команду боту, який транслює її на серверне обладнання, де відбувається обробка отриманого запиту програмним забезпеченням, після чого сервер повертає відповідь боту і виводить відповідь на екран додатка користувачеві. Управління запитами з чатботу відбувається через Framework поточного сайту, що дозволяє зберігати дані про поточного користувача та його дії.

Варто відмітити, що чатботи можуть обробляти запитання клієнтів за лічені секунди; а коли їм задають складне питання, вони можуть перенаправити сеанс чату до живого оператора. Компанії бачать у цьому велику перевагу: вони можуть задовольнити клієнтів, утримати їх на своєму сайті та заощадити гроші на персоналі підтримки, який їм доводиться наймати. Недивно, що світовий ринок чатботів, який у 2020 році оцінювався в 17 мільярдів доларів, за прогнозами, зросте до понад 102 мільярдів доларів до 2026 року [15].

Telegram чатбот забезпечує багатофункціональні можливості для результативного спілкування з користувачами та автоматизації бізнес-процесів компанії. Основний функціонал системи включає наступні компоненти [15]:

1) приймання та обробка повідомлень:

– отримання повідомлень від користувачів оперативно;
– підтримка різних типів контенту: текст, зображення, аудіо, відео;
– автоматичне розпізнавання та обробка команд користувача;

2) відправлення відповідей:

– надсилання текстових, медіа та комбінованих повідомлень;
– формування відповідей на запити користувачів;
– відправлення сповіщень про події, акції та оновлення;

3) інтерактивна взаємодія:

– використання клавіатур та інлайн-кнопок для навігації;

– швидкий доступ до основних функцій через меню команд;

– зручна навігація по розділах та функціях боту;

4) робота з базою даних:

– інтеграція з SQL базою даних для зберігання інформації;

– збереження даних користувачів та історії взаємодій;

– управління інформацією про товари, послуги та замовлення;

5) інтелектуальні можливості:

– застосування AI для обробки запитів;

– розпізнавання мови та генерація відповідей;

– система рекомендацій на основі історії взаємодій;

6) система сповіщень:

– регулярні оновлення про нові товари, послуги та акції;

– інформування про технічні роботи та зміни в системі;

7) безпека та захист даних:

– шифрування даних користувачів;

– забезпечення конфіденційності персональної інформації;

– захист цілісності даних та безпека транзакцій.

Прикладом однією з найуспішніших використань Telegram у вітчизняній логістиці – є Nova Pochta. Бот обслуговує понад 500 тисяч користувачів та обробляє більше 100 тисяч запитів щодня.

Функціональність Nova Pochta Bot:

– відстеження посилок за номером;

– розрахунок вартості доставки;

– пошук відділень та поштоматів;

– повідомлення про зміну статусу;

Показники ефективності:

– середній час відгуку: <2 секунд;

– точність автоматичних відповідей: 92%;

– зниження навантаження на call-center: 40%

– рівень задоволеності користувачів: 4.6/5.

2.2 Програмний інтерфейс Telegram Bot API

Telegram Bot API – це сукупність програмних інструментів, за допомогою яких розробники можуть створювати ботів для автоматичного обслуговування користувачів месенджера. API забезпечує широкий функціонал для обміну повідомленнями, обробки запитів та інтеграції з різними компонентами платформи Telegram.

Архітектура API базується на моделі клієнт-сервер з обміном даними через HTTP/HTTPS протоколи. Боти можуть взаємодіяти з користувачами, групами, каналами та іншими ботами, отримуючи вхідні повідомлення та надсилаючи відповіді. Використання HTTPS протоколу гарантує захищене передавання даних між ботом і серверною структурою Telegram.

Починаючи роботу з API, необхідно створити бота та отримати токен автентифікації. Токен є унікальним ідентифікатором, що використовується для авторизації всіх запитів до API. Важливим є збереження токена в захищеному середовищі в зв'язку з тим, що він надає повний доступ до функціоналу боту.

Для спрощення розробки існує ряд спеціалізованих бібліотек, що реалізують методи API. Найпоширенішими бібліотеками для Python є python-telegram-bot, aiogram та telepot. Ці інструменти абстрагують низькорівневі деталі взаємодії з API та надають функціональний інтерфейс для програмної реалізації**.**

Отримання оновлень від Telegram може здійснюватися двома методами: метод long polling і технологія webhooks**.** При використанні long polling підходу бот періодично надсилає запити до серверів Telegram для отримання нових повідомлень. Webhooks-метод передбачає реєстрацію URL-адреси, на яку Telegram автоматично надсилатиме інформацію про нові події в режимі online.

Опрацювання вхідних повідомлень включає аналіз їх змісту, ідентифікацію відправника та прийняття рішення щодо подальших дій. Бот може перевіряти тип повідомлення, його текстовий вміст, медіафайли та метадані для формування відповідної реакції на запит користувача.

Відправлення повідомлень здійснюється через виклик відповідних методів API із зазначенням необхідних параметрів: ідентифікатора чату, тексту повідомлення, опцій форматування та типу контенту. API підтримує відправлення текстових повідомлень, зображень, відео, аудіофайлів, документів та інших типів мультимедійного контенту.

Система обробки команд дозволяє ботам розпізнавати текстові команди від користувачів та реагувати на інтерактивні елементи, такі як інлайн-кнопки. Реалізація логіки обробки команд забезпечує гнучку взаємодію з користувачами та виконання різноманітних функцій відповідно до їхніх запитів.

2.3 Характеристика інструментів та технологій

2.3.1 Вибір мови програмування для побудови чатботу комп'ютерної системи логістичної компанії

Існує 6 мов програмування для чатботів. Проаналізуємо кожну з них:

- Python є пріоритетною мовою програмування в галузях обробки інформації, машинного навчання та створення чатботів. Python характеризується ергономічним синтаксисом, що забезпечує легке для читання та розуміння навіть для початківців-розробників. Оскільки Python не такий багатослівний, як багато інших мов, це відносно проста мова для створення прототипів чатботів, і вона не вимагає додаткового етапу компіляції, який потрібен деяким мовам програмування. Python забезпечує розробників розширеним арсеналом інструментів для реалізації алгоритмів machine learning та опрацювання текстової інформації. Серед найбільш функціональних рішень виділяється Natural Language Toolkit (NLTK) – комплексна бібліотека, яка користується визнанням у спільноті розробників як провідний інструмент для NLP-задач;
- Платформа Java характеризується високою ефективністю, стабільністю та потужними механізмами інтеграції з різнорідними технологічними екосистемами, що обумовлює її популярність при створенні корпоративних і високонавантажених Telegram-ботів. Мова демонструє переваги при обробці великих обсягів даних, забезпеченні швидкого відгуку на запити та горизонтального масштабування під зростаюче навантаження, проте має певні технологічні обмеження;
- Ruby – високорівнева, об'єктно-орієнтована мова програмування, що сприяє спрощенню процесу створення чатботів. Синтаксис Ruby як і Python зрозумілий для сприйняття та інтерпретації. Ruby також підтримує поширену техніку проектування алгоритмів, яка називається динамічним програмуванням, де ви можете змінювати код під час виконання, щоб відповідати мінливим потребам системи. Багато розробників захоплюються цією мовою через її чистий синтаксис та сторонні бібліотеки, які є зручними для користувача та добре задокументованими. Ruby має багатий арсенал бібліотек машинного навчання та

NLP, включаючи повноцінний фреймворк під назвою Stealth, який був розроблений з нуля для розробки чатботів;

– мову C++ також застосовують при розробці чатботів. Вона має найвищі показники швидкості роботи серед програм зі списку, тому вона часто використовується, коли продуктивність є пріоритетом. Однак це також низькорівнева мова програмування, тому це збільшення продуктивності має свій компроміс. C++ не є найлегшою мовою для вивчення, і в цій мові не так багато високорівневих бібліотек, спеціально призначених для створення чатботів, що означає, що вам доведеться створювати велику частину чатботу з нуля;

– PHP було створено для розробки веб-сайтів та веб-додатків і існує стільки ж, скільки існує веб-розробка. Одна з переваг PHP полягає в тому, що багато розробників знають її, і багато веб-сайтів уже використовують цю мову, що полегшує інтеграцію чатботу PHP в існуючі системи. Також існує фреймворк на PHP під назвою Botman, який надає всі інструменти, необхідні для створення чатботу, а також взаємодії з фреймворками веб-розробки PHP, такими як Laravel;

Clojure – мова програмування, яка є популярною для розробки чатботів. Вона працює на Java Virtual Machine (JVM) і може безперешкодно інтегруватися з існуючими системами, написаними на Java. Clojure – функціональна мова програмування, яка є діалектом Lisp. Вона використовує рекурсивні функції (які посилаються на себе для виконання) замість циклів для ітерації списків та масивів – функція, якій багато розробників віддають перевагу для обробки даних, що використовується в NLP.

Тепер ми можемо зробити висновок, що Python є оптимальним інструментом для створення чатботу в месенджері Telegram на базі комп'ютерної системи логістичної компанії завдяки поєднанню кількох ключових переваг: інтуїтивний синтаксис мови, розвинена екосистема спеціалізованих бібліотек, ефективна підтримка асинхронного програмування та активна спільнота розробників. Наявність готових програмних рішень і потужних фреймворків дозволяє оперативно розгортати функціональні боти, здатні опрацьовувати значні потоки клієнтських звернень та забезпечувати миттєву взаємодію.

2.3.2 Огляд програмних бібліотек для розробки чатботу

З метою обґрунтованого вибору інструментарію для розробки чатботу проведено порівняльний аналіз провідних програмних бібліотек: python-telegram-bot, aiogram та Telepot. Критеріями оцінювання виступали простота інтеграції, підтримка асинхронних операцій та повнота реалізації функціоналу Telegram API.

Python-telegram-bot являє собою одну з найбільш розповсюджених бібліотек для створення чатботу, що характеризується розгорнутою документацією і широкою підтримкою екосистеми розробників. Повну підтримку функціональних можливостей Telegram API забезпечує бібліотека, яка пропонує інтуїтивно зрозумілу архітектуру для розробки синхронних ботів. Основним обмеженням є недостатня підтримка асинхронних операцій, що призводить до зниження продуктивності системи при високих навантаженнях.

Aiogram – це бібліотека, архітектура якої орієнтована на асинхронну обробку запитів, що забезпечує суттєве підвищення продуктивності при одночасному обслуговуванні множинних користувацьких запитів. Дане рішення є оптимальним для високонавантажених проектів з великою кількістю одночасних користувачів. Певним недоліком можна вважати підвищену складність освоєння для розробників-початківців через необхідність володіння концепціями асинхронного програмування.

Telepot – мінімалістична бібліотека для невеликих проектів з обмеженими вимогами, проте характеризується невеликою популярністю та звуженим функціоналом порівняно з альтернативами .

На підставі проведеного порівняльного аналізу встановлено, що для реалізації чатботу зв'язку з клієнтами оптимальним технологічним рішенням є бібліотека aiogram. Її перевага полягає в нативній підтримці асинхронної обробки запитів, що забезпечує ефективне управління великою кількістю одночасних користувацьких звернень та гарантує високу продуктивність системи в умовах інтенсивного навантаження.

2.4 Створення моделі комп'ютерної системи логістичної компанії

Розробка моделі комп'ютерної системи логістичної компанії з інтегрованим чатботом потребує системного підходу до архітектурного проектування. Модель повинна враховувати існуючу IT-інфраструктуру, бізнес-вимоги та технічні обмеження для забезпечення ефективної інтеграції нових компонентів.

Принципи проектування моделі. Модульність системи є ключовим принципом, який дозволяє розділити комплексну систему на компактні функціональні модулі. Telegram-бот проєктується як окремий модуль, який може працювати незалежно від інших компонентів, але при цьому ефективно з ними взаємодіяти.

Основні модулі майбутньої системи:

– модуль обробки повідомлень Telegram;
– модуль інтеграції з існуючими базами даних;
– модуль бізнес-логіки для обробки запитів;
– модуль звітності та аналітики;
– модуль адміністрування та налаштувань.

Простота інтеграції забезпечується через використання стандартних протоколів обміну даними. Бот буде підключатися до існуючих систем через API, що не потребує кардинальних змін у поточній архітектурі.

Масштабованість проектується з урахуванням майбутнього зростання кількості користувачів. Система повинна легко розширюватися при збільшенні навантаження без значних змін в архітектурі.

Архітектура інтеграції. Трирівнева модель обрана як основа для проектування системи. Ця модель є простою для розуміння та ефективною для реалізації:

рівень інтерфейсу користувача включає Telegram-інтерфейс, через який клієнти взаємодіють з системою. На цьому рівні обробляються команди, відображаються меню та формуються відповіді у зрозумілому для користувача форматі;

рівень бізнес-логіки містить всі правила обробки запитів клієнтів. Тут вирішується, як саме обробляти кожне звернення, до якої системи звертатися за інформацією та у якому форматі надавати відповідь;

рівень даних відповідає за зберігання інформації та інтеграцію з наявними базами даних компанії. Цей рівень забезпечує отримання актуальної інформації про замовлення, клієнтів та статуси доставки.

Точки інтеграції з існуючими системами. Підключення до CRM-системи дозволить боту отримувати інформацію про клієнтів та їх попередні звернення. Це забезпечить персоналізований підхід до обслуговування та швидше вирішення питань.

Інтеграція з TMS надасть доступ до актуальної інформації про статус доставки, місцезнаходження вантажів та очікувані терміни доставки. Клієнти зможуть отримувати цю інформацію миттєво через бот.

З'єднання з біллінговою системою дозволить автоматично розраховувати вартість послуг та надавати клієнтам точну інформацію про тарифи.

Інтеграція з базою знань забезпечить бота готовими відповідями на найпоширеніші питання клієнтів. Це значно прискорить обслуговування та зменшить навантаження на операторів.

Алгоритм роботи системи. Початкова обробка запиту відбувається коли клієнт надсилає повідомлення боту. Система визначає тип запиту (відстеження, скарга, загальна інформація) та направляє його до відповідного модуля обробки.

Пошук інформації здійснюється через запити до відповідних баз даних компанії. Система автоматично знаходить потрібну інформацію та форматує її для відображення у Telegram.

Формування відповіді включає перетворення технічної інформації у зрозумілий для клієнта формат. Відповідь може включати текст, кнопки для подальших дій або файли з документами.

Логування взаємодії забезпечує збереження історії всіх звернень для подальшого аналізу та покращення якості обслуговування.

Забезпечення надійності. Резервування критичних компонентів гарантує безперебійну роботу системи навіть при відмові окремих елементів. Бот матиме резервні канали з'єднання з основними системами компанії.

Система моніторингу буде відстежувати роботу всіх компонентів та автоматично повідомляти адміністраторів про можливі проблеми.

Процедури відновлення розроблені для швидкого усунення технічних збоїв та мінімізації впливу на обслуговування клієнтів.

Створена модель забезпечує гнучку та надійну основу для впровадження чатботу в існуючу IT-інфраструктуру компанії з мінімальними ризиками та максимальною ефективністю.

3 СИНТЕЗ КОМП'ЮТЕРНОЇ СИСТЕМИ ЛОГІСТИЧНОЇ КОМПАНІЇ

3.1 Цілі впровадження системи

В комп'ютерну систему логістичної компанії, призначеної для координації взаємодії між структурними підрозділами організації, ми впровадимо чатбот замовлень та комунікації з клієнтом.

Компанія надає транспортно-логістичні послуги, адаптовані під індивідуальні потреби кожного замовника.

Основні напрямки діяльності:

– транспортно-логістичні послуги – доставка товарів автомобільним транспортом у межах міста та Україні. Функції: планування оптимальних маршрутів, підбір відповідного типу транспорту, координація доставки та документальний супровід;

– експедиторські послуги – повний комплекс послуг з організації перевезення, включаючи оформлення товарно-транспортних документів та страхування вантажів;

Система спрямована на автоматизацію процесів обслуговування клієнтів за допомогою чатботу в месенджері Telegram. Чатбот виступає інструментом оптимізації комунікації, забезпечуючи ефективну обробку замовлень на перевезення, запитів щодо статусу доставки та оперативне вирішення логістичних питань.

Цілі системи для клієнтів:

– отримання інформації про компанію – бот надає відомості про спектр логістичних послуг, тарифи та умови співпраці;

– оформлення замовлення на перевезення – можливість надіслати через бота текстові повідомлення, фотографії вантажу, документи, голосові повідомлення з деталями замовлення (тип вантажу, габарити, маршрут, терміни доставки);

– відстеження статусу вантажу – отримання актуальної інформації про місцезнаходження та стан вантажу на всіх етапах транспортування;

– отримання консультації – можливість залишити контактний номер телефону для дзвінка від менеджера з питань розрахунку вартості перевезення, вибору оптимального маршруту або вирішення проблемних ситуацій.

Цілі системи для адміністрації компанії:

– централізований доступ до запитів клієнтів з можливістю оперативного реагування через текстові повідомлення, надсилання фотографій, документів, відео або голосових повідомлень;

– ведення бази даних замовників та історії їх замовлень;

– адміністрування доступу користувачів, включаючи блокування та розблокування при необхідності;

– масова розсилка інформаційних повідомлень клієнтам про зміни тарифів, нові послуги або важливі сповіщення;

– контроль швидкості обробки запитів та зняття обмежень на відправлення повідомлень для пріоритетних клієнтів.

Запити від клієнтів:

– отримання консультацій щодо умов перевезення та тарифів;

– надання інформації про вантаж для розрахунку вартості доставки (вага, габарити, маршрут, терміни);

– узгодження деталей замовлення після розрахунку вартості перевезення;

– відстеження статусу доставки та місцезнаходження вантажу;

– надання підтримки при виникненні проблем під час транспортування (затримки, пошкодження, зміна маршруту).

Формати відповідей клієнту:

– текстові повідомлення;

– відправлення фото (документів, накладних, фото вантажу);

– надсилання файлів (рахунки, договори, акти виконаних робіт);

– голосові повідомлення для оперативної комунікації;

– можливість здійснення телефонного дзвінка на запит клієнта;

– можливість особистої зустрічі в офісі компанії для обговорення великих або регулярних замовлень.

Етапи взаємодії з замовником:

а) прийняття замовлення:

– збір інформації про характеристики вантажу (вага, габарити, тип);

– визначення маршруту та термінів доставки;

– розрахунок вартості перевезення з урахуванням усіх логістичних операцій;

б) узгодження умов:

– формування комерційної пропозиції з деталізацією маршруту, вартості та термінів;

– обговорення особливих вимог до транспортування (страхування, супровід);

– підписання договору та підтвердження замовлення;

в) виконання та контроль:

– надання клієнту трек-номера для відстеження вантажу;

– регулярне інформування про статус перевезення на ключових етапах маршруту;

– оформлення та передача супровідної документації;

– підтвердження доставки з наданням відповідних документів.

Система забезпечує безперервний цикл логістичного обслуговування компанії від прийняття замовлення до підтвердження доставки, гарантуючи прозорість та ефективність усіх процесів.

3.2 Обґрунтування вибору бази даних для системи

При створенні автоматизованої системи керування замовлень та комунікації з клієнтом логістичної компанії на основі чатботу в месенджері Telegram, одним з ключових питань є визначення оптимального рішення для організації зберігання даних.

З огляду на специфіку поставленого завдання, виникає необхідність у виборі системи управління базами даних, що здатна гарантувати стабільність роботи, ефективність обробки запитів та можливість масштабування. Ми розглянемо бази та порівняємо їх характеристики для визначення найбільш оптимального варіанту.

Системи керування базами даних поділяються на два основних класи: реляційні (RDBMS) та нереляційні (NoSQL), кожен з яких має свої особливості застосування при розробці сучасних програмних рішень, включаючи Telegram-ботів. У межах даного розділу розглянуто теоретичні основи реляційних систем на прикладі MySQL та PostgreSQL, а також нереляційних рішень, представлених системами Redis та MongoDB, що дозволить обґрунтувати вибір оптимальної технології для конкретних завдань.

3.2.1 Реляційні бази даних (RDBMS)

RDBMS (Relational Database Management Systems - системи керування реляційними базами даних) належать до найбільш розповсюджених технологій організації структурованої інформації з наявністю логічних взаємозв'язків. При розробці Telegram-ботів реляційні бази даних застосовуються для систематизованого збереження відомостей про користувачів, архіву запитів, текстових повідомлень та інших категорій даних, що вимагають впорядкованого розміщення з функціями пошуку та відбору.

Характерні риси реляційних баз даних:

– інформація організується у табличному форматі, де кожна таблиця містить рядки (записи) і колонки (атрибути), що забезпечує логічну структуризацію даних.

– система підтримує створення взаємозв'язків між окремими таблицями за допомогою механізму зовнішніх ключів, що гарантує консистентність та цілісність інформації.

– взаємодія з базою даних здійснюється через стандартизовану мову SQL, що забезпечує виконання повного спектру операцій: вибірку, додавання, модифікацію та вилучення записів.

– підтримка транзакційної моделі ACID гарантує збереження консистентності даних навіть у випадку системних збоїв або аварійних ситуацій.

Основні переваги застосування реляційних баз даних у розробці Telegram-ботів:

– програмні боти переважно оперують структурованою інформацією (профілі користувачів, журнали звернень, користувацькі параметри, текстові відгуки), для якої реляційна модель забезпечує оптимальну організацію та маніпулювання.

– використання SQL надає широкі можливості для фільтрації, упорядкування, модифікації та вилучення записів, а також формування комплексних запитів із залученням декількох таблиць через операції об'єднання (JOIN) .

– Telegram-бот здатен оперувати різноманітними категоріями інформації: архівом діалогів, індивідуальними параметрами користувачів, даними про товари та замовлення. Реляційна модель дозволяє організувати збереження без надлишковості через систему міжтабличних зв'язків.

– реляційні бази даних демонструють ефективність при роботі з великими масивами структурованої інформації, забезпечуючи масштабованість функціоналу боту та підтримку сучасних механізмів захисту даних (системи аутентифікації, криптографічний захист та інші методи).

3.2.1.1 База даних MySQL

MySQL являє собою одну з найбільш розповсюджених систем управління базами даних реляційного типу з відкритим вихідним кодом, що знайшла широке застосування у сфері веб-розробки, зокрема при створенні чатботів, веб-сайтів, електронних торгових майданчиків та інших інформаційних систем.

Архітектура MySQL базується на моделі "клієнт-сервер", де серверна частина відповідає за збереження та управління даними, а клієнтські додатки здійснюють взаємодію через SQL-запити. Система забезпечує повний спектр операцій для маніпулювання реляційними даними, відомих як CRUD-операції (створення, читання, оновлення, видалення), і демонструє високу продуктивність при обробці запитів завдяки вбудованим механізмам оптимізації, що є критично важливим при опрацюванні значних масивів даних.

Головною особливістю MySQL є реалізація транзакційної моделі, яка гарантує збереження цілісності інформації навіть у випадку системних збоїв або критичних помилок. Використання підсистеми зберігання InnoDB забезпечує паралельну обробку множинних запитів та повну відповідність принципам ACID (атомарність, консистентність, ізольованість, довговічність). Система володіє розвиненим механізмом індексування, що суттєво підвищує швидкодію виконання комплексних запитів та підтримуються різні варіанти об'єднання таблиць через операції JOIN, що спрощує роботу з реляційно пов'язаними даними.

3.2.1.2 СУБД PostgreSQL

PostgreSQL являє собою об'єктно-реляційну систему управління базами даних, яка характеризується високим рівнем надійності, можливостями горизонтального та вертикального масштабування, а також підтримкою комплексних аналітичних запитів. Функціональний арсенал системи охоплює реалізацію транзакційної моделі ACID, механізми повнотекстового пошуку, розвинену систему індексування, роботу з великими бінарними об'єктами (BLOB), а також інструментарій для визначення користувацьких типів даних та програмованих функцій. Будучи open-source платформою, PostgreSQL передбачає можливість розширення базової функціональності через систему плагінів та додаткових модулів. Висока відмовостійкість та ефективність при обробці значних масивів інформації зробили PostgreSQL провідним рішенням для розробки корпоративних систем та високонавантажених веб-сервісів.

Конкурентною перевагою PostgreSQL є імплементація розширених механізмів обробки даних, зокрема нативна підтримка JSON-структур, просторових даних (через модуль PostGIS) та операцій з масивами. Система демонструє ефективну паралельну обробку запитів та підтримує різні схеми реплікації, що забезпечує високу доступність та масштабованість у розподілених інфраструктурах. Завдяки зазначеним технічним характеристикам PostgreSQL оптимально підходить для впровадження в системах з підвищеними вимогами до надійності та швидкодії опрацювання значних інформаційних потоків, включаючи фінансові сервіси, чатботи та складні веб-додатки.

3.2.2 Нереляційні бази даних (NoSQL)

NoSQL (нереляційні СУБД) становить клас технологій, призначених для збереження та обробки великих масивів неструктурованої або слабоструктурованої інформації. Ці системи демонструють ефективність у середовищах з динамічно змінюваними даними без фіксованої схеми, що є характерним для чатботів, зокрема платформи Telegram. При розробці Telegram-ботів NoSQL-рішення застосовуються для збереження користувацьких профілів, історії запитів, текстових повідомлень, конфігураційних параметрів та супутніх метаданих.

Архітектура NoSQL-систем не передбачає обов'язкової попередньої декларації схеми даних, дозволяючи зберігати інформацію в різноманітних форматах (JSON, BSON, XML), що забезпечує гнучкість при роботі з неструктурованими або частково структурованими даними. У контексті Telegram-ботів така властивість є особливо цінною для збереження повідомлень з гетерогенним контентом (текстові дані, зображення, відеоматеріали), де кожен об'єкт може мати унікальну структуру.

Важливою характеристикою нереляційних БД виступає забезпечення горизонтального масштабування через розподілення навантаження між множинними серверами або вузлами кластера, що забезпечує підвищення загальної продуктивності системи. Telegram-боти з інтенсивним потоком повідомлень, особливо в масштабних каналах, можуть ефективно використовувати дану архітектурну особливість для забезпечення стабільної роботи при зростанні навантаження.

Більшість NoSQL-рішень оптимізовані для високошвидкісних операцій вибірки та внесення даних, які забезпечують мінімальну затримку при обробці запитів у режимі реального часу. Така характеристика є критично важливою для інтерактивних Telegram-ботів, де система зв'язку вимагає миттєвої реакції на користувацькі запити та оперативної обробки вхідних повідомлень.

3.2.2.1 База даних Redis

Redis являє собою систему збереження даних типу "ключ-значення" з резидентним розміщенням інформації в оперативній пам'яті, що забезпечує надвисоку швидкість виконання операцій отримання та збереження даних. Саме ця архітектурна особливість робить Redis оптимальним рішенням для високонавантажених систем, що потребують мінімальної латентності, зокрема для реалізації механізмів кешування, керування користувацькими сесіями, організації черг повідомлень та роботи з тимчасовими даними. Система підтримує розгалужений набір структур даних, серед яких рядки, списки, множини та хеш-таблиці, яка, при проектуванні різноманітних алгоритмів обробки інформації, забезпечує гнучкість. Важливою перевагою Redis є реалізація атомарних операцій, зокрема при роботі з чергами та іншими складними структурами, що дозволяє створювати надійні розподілені системи.

Архітектура Redis передбачає вбудовані механізми реплікації для резервного копіювання інформації, що посилює рівень доступності та відмовостійкості системи. Незважаючи на пріоритетність роботи в оперативній пам'яті, Redis реалізує механізми персистентності з метою збереження інформації на постійних носіях, а це забезпечує можливості відновлення інформації після перезапуску системи. Підтримка моделі "публікація-підписка" (pub/sub) дозволяє організувати ефективний обмін повідомленнями між розподіленими компонентами масштабованих систем. Завдяки поєднанню високої продуктивності та функціональної гнучкості, Redis знайшов широке застосування в системах реального часу, включаючи Telegram-ботів, де критично важливим є мінімальний час відгуку на користувацькі запити.

3.2.2.2 База даних MongoDB

MongoDB представляє собою документо-орієнтовану СУБД**,** що використовує формат BSON (Binary JSON) для збереження інформації. Даний формат забезпечує ефективне збереження комплексних та неструктурованих даних, що робить MongoDB оптимальним рішенням для систем з гетерогенними даними без фіксованої схеми. Система характеризується гнучкою моделлю даних, що дозволяє інкапсулювати різнотипну інформацію в межах одного документа та підтримувати систему індексування для оптимізації операцій пошуку та сортування. Архітектура MongoDB базується на принципі пріоритету продуктивності над строгою транзакційністю, що забезпечує високу швидкодію при інтенсивних навантаженнях.

Ключовою конкурентною перевагою MongoDB є реалізація горизонтального масштабування через механізм шардингу, який забезпечує автоматичний розподіл даних між множинними серверами з централізованим керуванням. Така архітектура гарантує високу доступність системи та ефективну обробку масивних обсягів інформації без деградації продуктивності. Система також володіє розвиненими механізмами реплікації, що забезпечують відмовостійкість та безперервну доступність даних в разі виходу зі строю окремих серверних вузлів. Завдяки зазначеним технічним характеристикам MongoDB стала провідним рішенням для розробки високонавантажених динамічних веб-додатків, включаючи чатботи та системи обробки великих масивів даних.

При виборі рішення для управління даними чатботу замовлень та комунікації з клієнтом MongoDB демонструє оптимальне поєднання технічних характеристик, що відповідають специфічним вимогам даного типу додатків.

Документо-орієнтована архітектура MongoDB з використанням JSON-подібного формату забезпечує необхідну гнучкість для збереження гетерогенної інформації про користувачів та їхні запити без потреби в попередньому визначенні жорсткої схеми даних. Така властивість є критично важливою для Telegram-ботів, які обробляють різноманітний контент, включаючи текстові повідомлення,

мультимедійні файли та документи різних форматів, дозволяючи динамічно адаптувати структуру збереження даних згідно поточних потреб.

Здатність MongoDB до горизонтального масштабування через додавання нових серверних вузлів забезпечує ефективну обробку зростаючого потоку користувацьких запитів без втрати продуктивності. Така архітектурна особливість робить дану СУБД оптимальним рішенням для проектів з високими вимогами до масштабованості та безперервної доступності сервісу.

Використання формату BSON для організації сховища документів забезпечує значну швидкість виконання операцій отримання та збереження даних, яка дозволяє обробляти значні обсяги інформації в режимі реального часу. Дана характеристика є визначальною для чатботів, які повинні забезпечувати одночасне обслуговування великої кількості користувачів з мінімальною затримкою відповіді.

Наявність розвинених програмних бібліотек для популярних мов програмування, зокрема Python, та підтримка асинхронної обробки запитів, забезпечує ефективну роботу системи при високих навантаженнях. Інтуїтивно зрозумілий програмний інтерфейс MongoDB спрощує інтеграцію з Telegram API та іншими компонентами екосистеми бота.

Отже, MongoDB являє собою оптимальне технологічне рішення для реалізації чатботу замовлень та зв'язку з клієнтом логістичної компанії, поєднуючи необхідну функціональну гнучкість, масштабованість, продуктивність та простоту інтеграції в єдиній системі управління даними.

3.3 Формулювання технічних вимог до комп'ютерної системи логістичної компанії

3.3.1 Вимоги до реалізації системи

Серверна інфраструктура системи базується на використанні DNS-сервера, що дозволяє оптимізувати фінансові витрати на обслуговування.

Інтеграція з месенджером Telegram реалізується через відповідний API-інтерфейс, який забезпечує двосторонній обмін повідомленнями між ботом та користувачами.

Програмна логіка бота побудована на базі фреймворку Aiogram, що забезпечує ефективну обробку команд, текстових повідомлень та мультимедійних файлів.

Для збереження та управління даними обрано документо-орієнтовану СУБД MongoDB, яка оптимально підходить для обробки слабоструктурованих даних.

Модель даних БД включає такі атрибути**:**

– унікальний ідентифікатор користувача в Telegram;

– найменування користувача;

– поточний стан облікового запису (активний/деактивований);

– тип користувача (автоматизований бот/реальний клієнт);

– архів транспортних замовлень;

– контактні дані для зв'язку.

Адміністративний функціонал системи передбачає:

– масову розсилку інформаційних повідомлень активним користувачам із виключенням деактивованих облікових записів;

– можливість обмеження доступу для користувачів, що порушують регламент роботи системи або поширюють спам;

– регулювання інтервалів між повідомленнями під час активних переговорів щодо умов транспортування для підвищення оперативності комунікації.

Програмний сервер забезпечує координацію всіх компонентів системи, опрацювання клієнтських запитів та роботу з базою даних.

Технічні характеристики та експлуатаційні параметри:

– кросплатформність: підтримка операційних середовищ Windows та Linux;

– продуктивність: здатність опрацьовувати до 100 звернень за секунду при паралельній роботі 100 користувачів. При піковому навантаженні (200 паралельних з'єднань) допустимий відсоток збоїв не перевищує 0.1%;

– швидкодія: затримка автоматизованих відповідей не перевищує 1000 мілісекунд;

– оперативність обслуговування: звернення щодо стандартних питань опрацьовуються протягом 30 хвилин. Для складних логістичних розрахунків або нестандартних маршрутів термін опрацювання становить від 2 до 4 годин;

– графік функціонування: повноцінна підтримка клієнтів забезпечується в робочі години логістичного центру (08:00 – 18:00);

– обсяги інформації: система розрахована на збереження до 5 МБ даних на кожного користувача, враховуючи транспортну документацію, зображення вантажів та супровідні матеріали;

– безперервність роботи: наявність джерела безперебійного живлення гарантує автономну роботу системи протягом 12 годин при відключенні основного електропостачання.

3.3.2 Вимоги до функцій, виконуваних системою

Серверна інфраструктура комп'ютерної системи повинна включати розгорнутий DNS-сервіс компанії та встановлений чатбот замовлень та зв'язку з клієнтом.

Чатбот замовлень та зв'язку з клієнтом логістичної компанії в мережі Telegram буде надавати її клієнтам такі функції:

– інформації про транспортне підприємство – надання базових даних про діапазон транспортних послуг, географію перевезень та тарифи;

– відправлення матеріалів (зображення, відеофайли, документація, текст, голосові записи) менеджменту логістичної організації з метою розрахунку вартості перевезення, надання інформації про характеристики вантажу (вага, габарити, тип вантажу, маршрут доставки) або для отримання консультації та вирішення питань щодо поточних чи виконаних перевезень;

– надання клієнтом свого контактного номера телефону шляхом натискання кнопки швидкого доступу або введення цифр вручну, після чого бот фіксує заявку у системі та формує завдання для диспетчера здійснити дзвінок замовнику з метою уточнення умов та вартості перевезення, розрахунку складних маршрутів або вирішення оперативних питань з доставкою вантажів.

Окрім цього чатбот дозволяє обробляти мультимедійні та текстові формати інформації в рамках комунікації між замовниками послуг і представниками підприємства: фотографічні матеріали, відеоконтент, аудіоповідомлення, текстові дані, файлову документацію.

Панель управління для адміністраторів включає такі можливості:

– база даних усіх клієнтів системи;
– керування доступом користувачів (блокування/відновлення);
– функція групового інформування користувачів;
– видалення тимчасових обмежень користувача;
– перегляд журналу транзакцій клієнта.

3.3.3 Вимоги до видів забезпечення замовлень та комунікації з клієнтом

3.3.3.1 Вимоги до інформаційного забезпечення

Чатбот замовлень та комунікації з клієнтом повинен мати зручну для клієнтів форму, щоб забезпечити легке і зрозуміле користування.

Інформаційна модель документо-орієнтованої бази даних MongoDB повинна містити наступні атрибути:

– ідентифікатор запису в колекції;
– ідентифікатор користувача в системі Telegram;
– найменування облікового запису користувача;
– статус активності користувача (активний/деактивований);
– результат верифікації на автоматизовані запити;
– історія транспортних замовлень користувача;
– контактні реквізити для оперативного зв'язку.

Архітектура інформаційного обміну між модулями системи здійснюється через наступні механізми:

– клієнтська взаємодія з системою здійснюється через програмний інтерфейс Telegram API, що забезпечує двосторонній обмін повідомленнями між користувачем та ботом;

– обробка клієнтських запитів та формування відповідей реалізується через програмний фреймворк Aiogram, який забезпечує асинхронну обробку вхідних повідомлень;

операції з постійним сховищем даних MongoDB виконуються через асинхронний драйвер Compas, що забезпечує високу швидкодію при роботі з документо-орієнтованими структурами та підтримує неблокуючі операції введення-виведення.

3.3.3.2 Вимоги до лінгвістичного забезпечення

З метою якісної взаємодії між користувачами та системою логістичної компанії необхідно визначити стандарти лінгвістичного забезпечення, що регламентують мовні аспекти інтерфейсу та принципи формування текстових повідомлень.

Математичне моделювання роботи серверної інфраструктури з інтегрованим Telegram-ботом реалізовано на базі теорії масового обслуговування, що дозволяє оптимізувати процеси обробки клієнтських запитів та прогнозувати навантаження на систему.

Основні вимоги до мовного забезпечення системи:

– мовна локалізація – інтерфейс чатботу повинен функціонувати українською мовою як основною мовою спілкування, забезпечуючи зрозумілість всіх команд, меню та системних повідомлень для цільової аудиторії;

– візуальна комунікація – використання емодзі-символів для покращення сприйняття інформації та інтуїтивної навігації: транспортні іконки для позначення типів перевезень, статусні індикатори для відображення етапів доставки, візуальні маркери для категоризації послуг;

– стандартизація термінології – використання уніфікованого логістичного глосарію для забезпечення однозначності інтерпретації транспортних термінів, умов доставки та технічних характеристик перевезень;

– адаптивність повідомлень – формування текстових відповідей з урахуванням контексту запиту, використання професійної логістичної термінології для корпоративних клієнтів та спрощених формулювань для приватних замовників;

– структурованість інформації – логічна організація текстових блоків з використанням форматування для виділення ключової інформації: тарифів, термінів доставки, контактних даних.

3.3.3.3 Вимоги до технічного забезпечення

Для забезпечення стабільного функціонування комп'ютерної системи логістичної компанії з інтегрованим Telegram-ботом, побудованим на базі фреймворку Aiogram з використанням документо-орієнтованої СУБД MongoDB та асинхронної архітектури обробки запитів, серверне обладнання повинно відповідати наступним технічним специфікаціям.

Центральний процесор (CPU):

– багатопотокова архітектура процесора для одночасної обробки численних клієнтських звернень;

– високочастотний процесор (від 2.4 ГГц) для ефективного виконання асинхронних транзакцій та мінімізації затримок.

Оперативна пам'ять (RAM):

– мінімальний обсяг 8 ГБ для забезпечення буферизації даних та кешування запитів;

– рекомендований обсяг 16 ГБ для оптимальної продуктивності при піковому навантаженні.

Підсистема зберігання даних:

– твердотільні накопичувачі (SSD) для високошвидкісного доступу до бази даних та системних файлів;

– дисковий простір не менше 50 ГБ для розміщення операційної системи, бази даних клієнтів та архіву транспортної документації.

Мережева інфраструктура:

– стабільне інтернет-з'єднання з мінімальною пропускною здатністю 50 Мбіт/с та низькою латентністю;

– для високонавантажених періодів рекомендована швидкість каналу від 100 Мбіт/с для забезпечення безперебійного обміну даними.

Система резервного електроживлення:

– система UPS (Uninterruptible Power Supply) з автономністю роботи не менше 12 годин для забезпечення безперервності логістичних операцій при відключенні основного електропостачання або надзвичайних ситуаціях.

3.3.3.4 Вимоги до організаційного забезпечення

Для забезпечення ефективного функціонування чатботу зв'язку з клієнтами компанії необхідно створити організаційну структуру підтримки з наступними елементами:

1) адміністративний персонал:

призначити команду адміністраторів боту (2-3 особи) для забезпечення безперервної підтримки користувачів. Адміністратори працюють за графіком змін та виконують наступні функції:

– оперативне опрацювання запитів клієнтів та партнерів;
– модерація повідомлень та управління чорним списком користувачів;
– моніторинг технічної стабільності роботи боту;
– моніторинг та обробка статистики звернень.

2) режим роботи:

встановити робочий графік підтримки з 8:00 до 17:00 у робочі дні (можливе розширення робочого часу з урахуванням специфіки логістичної галузі). Для критичних запитів передбачити можливість позаштатної підтримки або автоматичних відповідей у неробочий час.

3) стандарти обслуговування:

– максимальний час першої відповіді: 15-30 хвилин у робочий час;
– повне вирішення стандартного запиту: до 2 годин;
– ескалація складних питань профільним спеціалістам (диспетчерам, менеджерам з доставки).

4) навчання персоналу:

провести інструктаж адміністраторів щодо функціоналу боту, типових сценаріїв обслуговування клієнтів логістичної компанії та протоколів комунікації.

3.3.3.5 Вимоги до програмного забезпечення

Для забезпечення роботи розробленої системи необхідна наявність середовища розробки, що підтримує виконання програм мовою Python, зокрема Visual Studio Code або його функціональних аналогів.

На комп'ютері має бути встановлено месенджер Telegram для здійснення перевірки працездатності чатбота та проведення тестових операцій у процесі його налагодження.

Робоче місце оператора логістичної компанії повинно бути оснащене пакетом офісних програм MS Office або його вільним аналогом (LibreOffice) для опрацювання документації, що надходить через систему чатбота.

Для забезпечення коректної взаємодії програми з базою даних необхідна наявність встановленого програмного забезпечення MongoDB, а також відповідних інструментів для обробки файлів різних форматів, включаючи зображення транспортної документації та супровідних матеріалів.

3.3.4 Вимоги до захисту інформації

Безпека даних системи чатботу в месенджері Telegram реалізується через систему адміністративних та технологічних заходів**:**

1) ізоляція серверної інфраструктури. Розгортання боту на локальному сервері без прямого підключення до глобальної мережі суттєво знижує ймовірність несанкціонованого доступу до конфіденційної інформації клієнтів. Така архітектура створює замкнене середовище зберігання даних з обмеженими каналами зовнішньої взаємодії;

2) протидія автоматизованим атакам. Система повинна включати механізми обмеження частоти надсилання повідомлень від окремих користувачів. Встановлення часових інтервалів між запитами запобігає перевантаженню серверних ресурсів внаслідок масових автоматизованих звернень. Додатково необхідна реалізація верифікації користувачів для відсіювання автоматичних програм (ботів) на етапі ініціалізації діалогу;

3) криптографічний захист каналів зв'язку. Програмний інтерфейс Telegram забезпечує використання безпечного протоколу HTTPS для взаємодії з серверами платформи. Це гарантує конфіденційність передачі інформації між компонентами системи;

4) розмежування прав доступу до бази даних. СУБД MongoDB підтримує рольову модель керування доступом, що дозволяє чітко визначити повноваження кожного користувача системи. Додатково застосовується шифрування даних як на рівні файлового сховища, так і в процесі передачі по мережі, що створює багаторівневий захист інформаційних активів компанії.

3.3.5 Вимоги до ергономіки системи

Користувацький інтерфейс чатботу має бути спроєктований з урахуванням принципів зручності та доступності для різних категорій користувачів – від приватних клієнтів до корпоративних замовників логістичних послуг:

– простота навігації. Структура меню боту повинна забезпечувати швидкий доступ до ключових функцій: відстеження вантажів, оформлення замовлення, розрахунок вартості доставки, зв'язок з оператором. Кількість кроків для виконання типової операції має бути мінімальною – не більше 3-4 переходів;

– зрозумілість інтерфейсу. Команди та кнопки боту формулюються чіткою діловою мовою без використання технічного жаргону. Кожна дія супроводжується короткими підказками, які пояснюють наступний крок. Особлива увага приділяється користувачам, які вперше взаємодіють із ботом – передбачено режим ознайомлення з основними можливостями;

– візуальне оформлення. Для підвищення залученості користувачів та покращення сприйняття інформації доцільно використовувати графічні елементи: тематічні стікери або емодзі для позначення статусів доставки (наприклад, "вантаж прийнято", "в дорозі", "доставлено"), іконки для різних типів послуг. Це робить комунікацію більш живою та допомагає швидше орієнтуватися в інформації;

– адаптивність. Інтерфейс має коректно відображатися на різних пристроях (смартфони, планшети) та враховувати можливість роботи в умовах обмеженого інтернет-з'єднання, що важливо для мобільних користувачів логістичної сфери.

3.3.6 Показники призначення

Для стабільного функціонування системи використанням чатботу замовлень та комунікації з клієнтом, реалізованої в месенджері Telegram, необхідно створити умови, що відповідають вимогам технологічного процесу логістичного підприємства. Першочергової уваги потребує серверне обладнання, оскільки воно зберігає облікову інформацію операторів компанії, дані про замовлення клієнтів, а також програмну частину чатбота. Система повинна забезпечувати доступ до зазначеної інформації як в межах корпоративної мережі, так і з віддалених робочих місць для авторизованих користувачів. Обов'язковою умовою є захист даних від можливого пошкодження або втрати внаслідок технічних збоїв, аварійних ситуацій або несанкціонованого доступу.

Для підтримки працездатності системи необхідна наявність системного адміністратора з вищою освітою за спеціальністю «Комп'ютерна інженерія» (ступінь бакалавра або вище). Його функціональні обов'язки включають забезпечення безперебійної роботи впровадженої системи автоматизації. Має бути розроблений регламент планових перевірок та профілактичних заходів щодо підтримки працездатності серверного обладнання і програмного забезпечення чатбота. Додатково передбачається виконання позапланових робіт з налагодження та усунення несправностей за зверненнями користувачів системи.

Штат компанії повинен включати оператора, відповідального за опрацювання вхідних запитів, що надходять від клієнтів через інтерфейс чатбота, а також формування відповідей на їх звернення.

3.4 Розробка схеми функціональної структури

Функціональна архітектура системи складається з чотирьох взаємопов'язаних компонентів, кожен з яких виконує специфічні завдання в процесі обробки клієнтських запитів.

Клієнтський застосунок Telegram. Програмне середовище (мобільна або десктопна версія месенджера), через яке здійснюється безпосередня комунікація з клієнтами логістичної компанії. Застосунок отримує від серверної частини боту структуровані повідомлення, меню та запити даних, а передає назад текстові введення користувача, файлові вкладення та команди керування.

Програмний інтерфейс Telegram API. Посередницький шар, що забезпечує технічну взаємодію між ботом та серверами месенджера. На вхід API надходять різноформатні дані від користувачів: текстові повідомлення, мультимедійні файли (зображення, документи, аудіозаписи), а також сигнали від інтерактивних елементів інтерфейсу. Вихідним результатом є передача цих даних боту та доставка відповідей користувачам у зворотному напрямку.

Чатбот замовлень та комунікації з клієнтами. Центральний програмний модуль, розроблений мовою Python, який реалізує бізнес-логіку системи. Бот аналізує вхідні команди та запити клієнтів, формує відповідні реакції (інформаційні повідомлення, запити додаткових даних, переадресація до операторів) та керує потоком діалогу відповідно до сценаріїв обслуговування.

Система управління базою даних. Репозиторій для збереження структурованої інформації про клієнтів, історію їхніх звернень, статуси замовлень та параметри доставки. База приймає запити на читання, запис або модифікацію даних від програмної логіки боту, а у відповідь надає запитувану інформацію або підтвердження виконання операцій.

Всі компоненти функціонують синхронно, забезпечуючи безперервний цикл обробки клієнтських звернень від моменту надходження запиту до формування остаточної відповіді.

3.5 Короткий опис мережевої інфраструктури об'єкту дослідження

Проектування комп'ютерної мережі логістичного підприємства базувалося на збалансованому підході, що враховує три ключові критерії: економічну ефективність, технічну придатність та відповідність функціональним завданням організації. Вибір мережевого обладнання здійснювався з урахуванням технічних параметрів, які повністю забезпечують потреби інформаційної системи підприємства, при цьому зберігаючи фінансову раціональність у межах бюджетних можливостей підприємства.

Перелік підсистем, їх призначення й основні характеристики:

– Application layer: загалом виконує завдання з управління мережевою політикою та її моніторінгу з інтеграцією з Telegram;

– Control layer: сервер, на якому розміщений Network Controller, і з якого здійснюється керування та моніторинг мережних пристроїв;

– Infrastructure layer: локальна комп'ютерної мережа (ЛКМ) та мережеві пристрої, відповідальні за пересилання трафіку.

Робочі групи відокремлені в окремі 5 локальних мереж та підтримують кількість вузлів з майбутнім розширенням відповідно до таблиці 3.2.

Таблиця 3.2 – Робочі групи та кількість вузлів в сегменті мережі

№	Умовне позначення	Робоча група	Кількість ПК	Принтери
LAN1	DIRECTION	РГ №1+2	5	1
LAN2	DLS	РГ №3+4+5	12	1
LAN3	DEVNET	РГ №7	3	1
LAN4	MARKETING	РГ №6	3	1
LAN5	ITSUPPORT	РГ №8	2	1
Загалом			25	5

В мережі DLS налаштовані віртуальні локальні мережі для відповідної робочої групи.

ЛКМ має виділені сервери: файл-сервер, web-сервер та сервер, на якому розміщений Network Controller, і з якого здійснюється керування та моніторинг мережі.

FTP-сервер знаходиться у відділі керівництва. Інформація про мережеве обладнання та всі налаштування зберігаються на цьому сервері.

Web-сервер знаходиться у відділі технічної підтримки, а сервер з Network Controller знаходиться у відділі розвитку технологій.

Під час реалізації мережевої інфраструктури впроваджено комплекс мережевих протоколів та сервісів (рисунок 3.1).

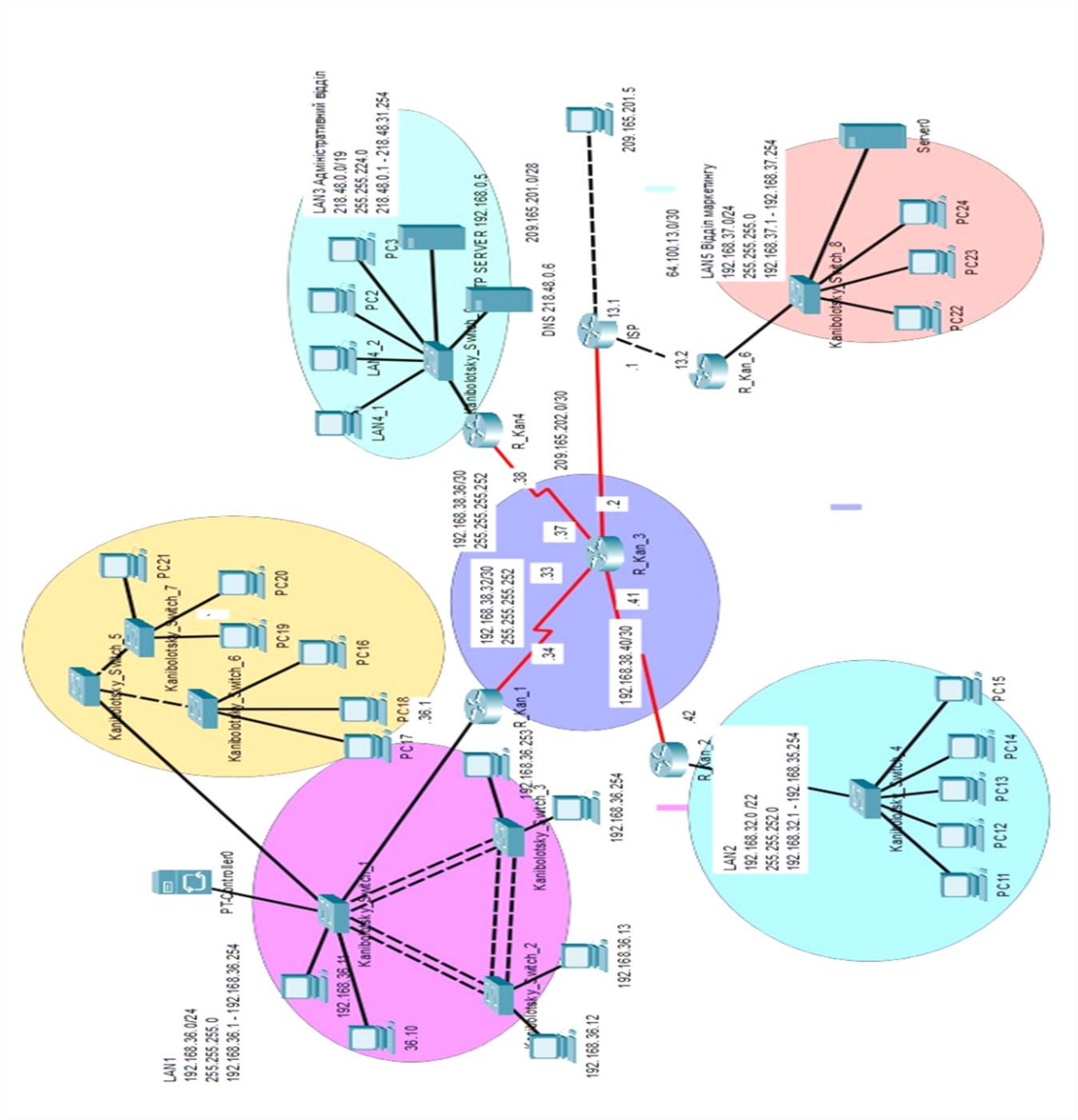

Рисунок 3.1 – Топологія мережі з використання SDN логістичної компанії

3.6 Вибір та обґрунтування застосування апаратних засобів

В логістичному підприємстві вже налаштована та функціонує комп’ютерна система із певними апаратними засобами.

В комп’ютерній мережі логістичної компані використовуються наступні пристрої відповідно до таблиці 3.3.

Таблиця 3.3 – Пристрої, використані при побудові корпоративної мережі логістичної компанії

Позиція	Тип пристрію	Кількість	Загальнакількість
1	Комп’ютер (PC)	Основна мережа – 27 Віддалена мережа – 3	30
2	Комутатор	Основна мережа – 6 Віддалена мережа – 1	7
3	Маршрутизатор	Основна мережа – 4 Віддалена мережа – 1	5
4	Сервер	Основна мережа – 2 Основна мережа – 1	3

Замовником було прийнято рішення використовувати обладнання підприємства DELL та Cisco. Розробка технічних вимог до комп’ютерної або кіберфізичної системи. Повні характеристики мережевого обладнання приведені нижче (рисунки 3.2 – 3.5) відповідно до таблиць 3.4 – 3.7.

Рисунок 3.2 – Комп'ютер Dell Vostro 3681 v15

Таблиця 3.4 – Технічні характеристики комп'ютера Dell Vostro 3681 v15

Технічні характеристики комп'ютера	Intel Core i5-10400 (2.9 - 4.3 ГГц)/RAM 16 ГБ/HDD 1 ТБ + SSD 480 ГБ/Intel UHD Graphics 630/без ОД/LAN/Wi- Fi/Bluetooth/кардридер Windows 10 Pro/клавіатура + миша
Процесор	Шестиядерный Intel Core i5-10400 (2.9 - 4.3 ГГц)
Чипсет плати	Intel B460
Обсяг оперативної пам'яті	16 ГБ
Порти На передній панелі:	2 x USB 3.2 Gen 1 Type-A 2 x USB 2.0 Комбінований аудіороз'єм для навушників/мікрофона Кардридер
На задній панелі:	2 x USB 3.2 Gen 1 Type-A2 x USB 2.0x HDMI 1 x VGA 1 x LAN (RJ45) 1 Гбіт/сек
Потужність БЖ	260 Вт
Відеокарта	Intel UHD Graphics 630
Охолодження	Охолодження ЦП: BOX
Додаткові можливості	Модуль бездротового зв'язку Wi-Fi 802.11ac
Обсяг SSD	480 ГБ
Тип відеокарти	Інтегрована
Обсяг HDD	1 ТБ
Частота ядра	2,9

Рисунок 3.3 – Сервер Dell PowerEdge R440 A13 (per440ceeM02)

Таблиця 3.5 – Технічні характеристики Сервера Dell PowerEdge R440 A13 (per440ceeM02)

Форм-фактор	1U Rackmount
Кількість ядер	8
Тип процесорів	Intel Xeon
Процесор	Восьмиядерний Intel Xeon Silver 4208 (2.1-3.2ГГц)
Обсяг оперативної пам'яті	16Gb
Частота	2.1 ГГц
Швидкість DIMM:	До 2666 млн транзакцій у секунду
Тип пам'яті:	RDIMM LRDIMM
Слоти для модулів пам'яті:	16 слотів для модулів DDR4 DIMM Підтримуються тільки реєстрові модулі DDR4 DIMM з ECC
Кількість ЦП в комплекті	1
Контролери SAS/SATA	PERC H730P
Рівні RAID	0/1/5/6/10/50/60
Кількість БЖ в комплекті	1
Порти на передній панелі:	1 x виділений USB-порт iDRAC Direct1 x порт USB 2.0 1 x відеопорт
Порти на задній панелі	1 x виділений мережевий порт iDRAC1 x послідовний порт 2 x порти USB 3.0 1 x відеопор
Слоти розширення:	1 x PCIe 3.0 x16
Додаткові характеристики	
Передні відсіки дисководів:	До 10 дисків SAS/SATA 2.5" (жорстких дисків або твердотілих накопичувачів) і до 4 твердотілінакопичувачі NVMe макс. місткістю 48 Тбайт
	або:До 4 жорстких дисків SAS/SATA 3.5" макс.місткістю 56 Тбайт Кількість LAN (RJ-45)

Маршрутизатор Cisco 4321 Integrated Services призначений для надання розширених послуг у середовищі філій малого підприємства. За замовчуванням він забезпечує 50 Мбіт/с. Завдяки продуктивності за принципом «оплата за зростанням», можна збільшити пропускну здатність до100 Мбіт/с, придбавши ліцензії FL-4320-PERF-K9 [12].

Рисунок 3.4 – Маршрутизатор Cisco ISR4321

Таблиця 3.6 – Технічні характеристики Cisco ISR4321/K9

Виробник	Cisco Systems, Inc
Модель	CISCO ISR4321
Form Factor	External - modular - 2U
Розміри (WxDxH)	43.8 cm x 30.5 cm x 8.9 cm
Вага	8.2 kg
DRAM	4 Гб (installed) / 8 GB (max)
Flash	4 Гб (installed) / 8 GB (max)
Загальна кількість WAN та LAN портів	2
Інтегровані порти WAN	1 GE / SFP 1 GE
Aggregate Througput	50 Мб до 100 Мб
Протоколи маршрутизації	OSPF, IS-IS, BGP, EIGRP, DVMRP,PIM-SM, IGMPv3, GRE, PIM-SSM, static IPv4 routing, static IPv6 routing

Рисунок 3.5 – Комутатор Cisco WS-C2960X-24PS-L

Таблиця 3.7 – Технічні характеристики комутатора

Cisco WS-C2960X-24PS-L

Manufacturer:	Cisco
roduct ID:	WS-C2960X-24PS-L
Product Description:	Cisco Catalyst 2960-X 24 GigE PoE 370W, 4 x 1G SFP, LAN Base
Product Type:	Ethernet Switch
Interfaces/Ports	
Total Number of Network Ports:	24
Number of PoE (RJ-45) Ports:	24
Port/Expansion Slot Details:	24 x Gigabit Ethernet Network
	4 x Gigabit Ethernet Expansion Slot
Media & Performance	
Media Type Supported:	Twisted Pair
Twisted Pair Cable Standard:	Category 5
Ethernet Technology:	Gigabit Ethernet
Network Technology:	10/100/1000Base-T
I/O Expansions	
Number of Total Expansion Slots:	4
Expansion Slot Type:	SFP
Shared SFP Slot:	No
Number of SFP Slots:	4
Network & Communication	
Layer Supported:	2
Management & Protocols	
Manageable:	Yes
Management:	Web-based Management;VLAN; QoS; Syslog; Telnet; RMON I, II; SNMP v1, v2c, v3;CLI; DHCP
Memory	
Standard Memory:	512 MB
Memory Technology:	DRAM
Flash Memory:	128

Нами досліджено технічну базу та структуру мережі IT-системи логістичної компанії. За результатами розроблено архітектуру чатботу автоматизації клієнтського сервісу.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Функціональне призначення програмного забезпечення

Програмне забезпечення чатботу "Cargo-Bot" для комп'ютерної системи логістичної компанії розробляється з метою створення програмних засобів автоматизації комунікації з клієнтами, обробки замовлень на вантажні перевезення, інтеграції з картографічними сервісами для розрахунку відстаней та маршрутів, надання консультаційних послуг через інтелектуального AI-асистента, а також формування звітної документації (ТТН) та збереження даних у хмарній базі MongoDB.

Ключовою перевагою застосування чатботу є цілодобова доступність сервісу для клієнтів, що дозволяє формувати замовлення незалежно від графіку роботи диспетчерської служби.

4.2 Обґрунтування технічних характеристик програмного забезпечення

4.2.1 Постановка завдання на розробку програми

Розробити програму для чатботу, що функціонуватиме на базі платформи Telegram та забезпечуватиме наступні функції: реєстрацію та обробку замовлень на вантажні перевезення, автоматичний розрахунок вартості доставки з урахуванням реальних відстаней між містами через OpenRouteService API, надання AI-консультацій через Google Gemini API, формування електронних ТТН-накладних, сповіщення клієнтів про зміну статусу замовлень через email та Telegram, авторизацію операторів з підтримкою декількох методів верифікації, збереження та управління даними у базі MongoDB.

4.2.2 Алгоритм функціонування програми

Оскільки серед вимог до комп'ютерної системи логістичної компанії є безперервне функціонування та обробка замовлень у режимі реального часу, програма побудована на основі асинхронної архітектури, що забезпечує її циклічну роботу та одночасну обробку запитів від багатьох користувачів.

4.2.3 Опис організації вхідних та вихідних даних

Отримувані від користувачів дані (замовлення, контактна інформація, параметри вантажу) зберігаються у відповідних колекціях MongoDB. Для зберігання файлів ТТН-накладних використовується технологія GridFS.

Для отримання даних з бази використовується пошук за значеннями окремих полів серед документів відповідних колекцій за запитом програми. Основними колекціями бази даних є: orders (замовлення), clients (клієнти), operators (оператори), operators_list (список операторів для авторизації), reviews (відгуки та оцінки), chat_messages (історія чатів).

4.2.4 Обґрунтування вибору програмних засобів

В процесі розробки програмного забезпечення для чатботу логістичної компанії було обрано мову програмування Python, яка належним чином підходить для розробки чатботів на базі Telegram Bot API та має достатньо великий набір бібліотек для роботи з зовнішніми сервісами.

Середовищем розробки було обрано Visual Studio Code від компанії Microsoft завдяки його зручності та безкоштовній моделі розповсюдження.

Для збереження даних було використано MongoDB – документно-орієнтовану базу даних NoSQL, що зберігає дані у форматі JSON. Для візуалізації даних використано MongoDB Compass.

При розробці програми було використано наступний набір бібліотек:

1. Aiogram (версія 3.13) – асинхронна бібліотека для створення Telegram-ботів на Python із підтримкою зручного керування подіями, станами та діалогами.
2. Pymongo – бібліотека для взаємодії Python-програми з базою даних MongoDB.
3. GridFS – інструмент для зберігання файлів у MongoDB шляхом розбивання їх на частини (chunks).
4. Requests – бібліотека для виконання HTTP-запитів до зовнішніх API (OpenRouteService, геокодування).
5. Python-docx – бібліотека для генерації документів Microsoft Word (ТТН-накладні).
6. OpenAI – бібліотека для інтеграції з API штучного інтелекту GPT для реалізації AI-консультанта.
7. Aiosmtplib – асинхронна бібліотека для відправки email-повідомлень клієнтам.
8. Prometheus-client – бібліотека для збору та експорту метрик системи моніторингу.

4.3 Опис розробленої програми

4.3.1 Загальні відомості про структуру програми

Структурно програма чатботу "Cargo-Bot" складається з декількох модулів, кожен з яких виконує свої задачі, такі як запуск бота, робота з обробниками повідомлень та станами, формування клавіатур, генерація документів, відправка email-повідомлень, збір метрик тощо:

- **main.py** – основний файл програми, що відповідає за ініціалізацію бота, підключення роутерів, запуск сервера метрик Prometheus та неперервну роботу через механізм polling;
- **config.py** – конфігураційний файл, що містить токени API (Telegram, OpenRouteService, Gemini), параметри підключення до MongoDB, налаштування тарифів, типів вантажу, системний промпт для AI-консультанта та інші константи;
- **app/handlers.py** – основний модуль, що містить класи станів FSM (Finite State Machine), допоміжні функції для розрахунку відстаней, вартості, часу доставки, а також усі обробники (handlers) повідомлень та callback-запитів для клієнтів та операторів;
- **app/keyboards.py** – модуль, що містить функції для генерації Reply та Inline клавіатур для навігації користувачів по меню бота;
- **app/ttn_generator.py** – модуль для генерації товарно-транспортних накладних (ТТН) у форматі .docx на основі шаблону або програмно;
- **app/email_sender.py** – модуль для асинхронної відправки email-повідомлень (підтвердження замовлень, ТТН-накладні, зміна статусу, коди верифікації);
- **app/metrics.py** – модуль для збору та експорту метрик роботи бота у форматі Prometheus (кількість повідомлень, замовлень, помилок API, час обробки);
- **template_ttn.docx** – шаблон товарно-транспортної накладної з плейсхолдерами для автоматичного заповнення даними замовлення;
- **monitoring/** – директорія з конфігураційними файлами для системи моніторингу (Prometheus, Grafana, docker-compose).

Файлова структура чатботу (див рисунок 4.1) організована за модульним принципом: основна логіка розміщена в директорії **app/** (обробники повідомлень handlers.py, клавіатури keyboards.py, генератор ТТН ttn_generator.py, відправка email та метрики), а конфігурація, точка входу main.py та шаблон накладної template_ttn.docx — у кореневій директорії. Окрема директорія **monitoring/** містить конфігураційні файли для системи моніторингу (docker-compose.yml, prometheus.yml), що забезпечує відстеження роботи бота в production-середовищі.

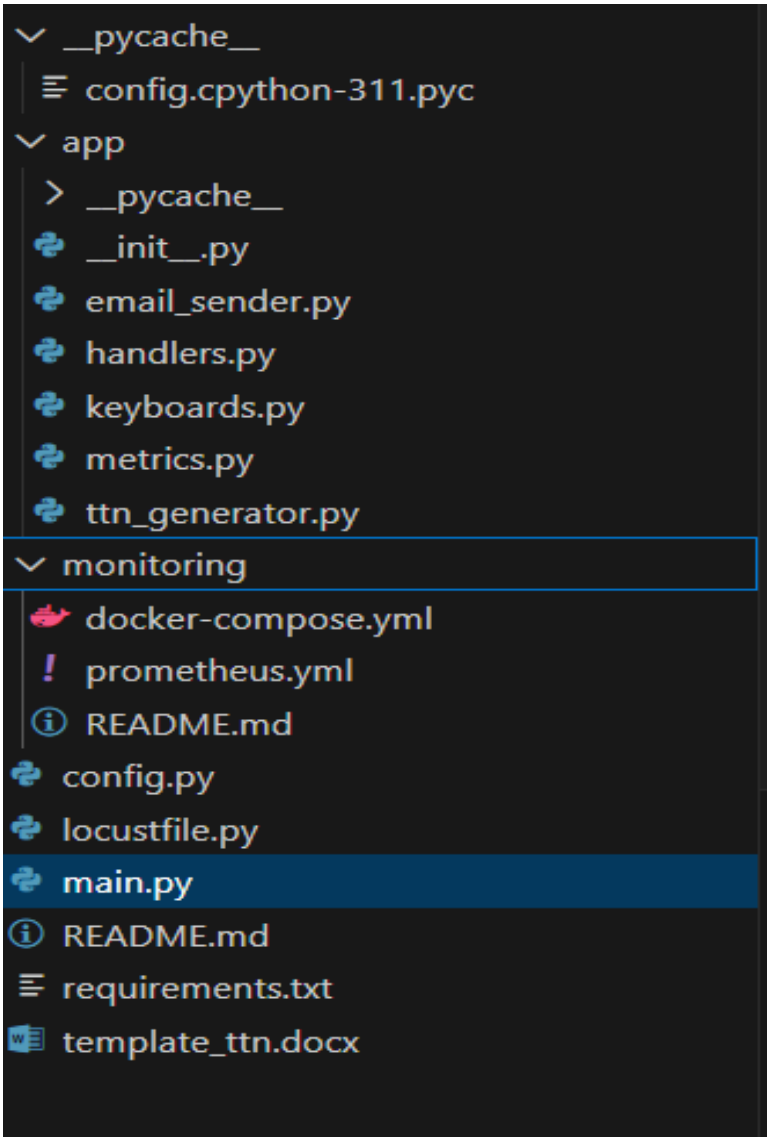

Рис. 4.1 – Файлова структура чатботу

4.3.2 Опис алгоритму роботи програми чатботу

На схемах, що зображено на рисунках 4.2 – 4.10, наведено алгоритм роботи програми, який починає свою роботу після введення користувачем команди /start. Програма складається з декількох гілок, кожна з яких відповідає за свій сегмент роботи програми та функції.

Нижче наведено перелік умовних позначень на цих схемах:

1) (К) – перехід до меню клієнта, в якому відбувається створення замовлень, відстеження, консультації та інші операції;

2) (К1) – (К8) – переходи до підменю клієнта: К1 – нове замовлення, К2 – мої замовлення, К3 – відстежити, К4 – тарифи, К5 – AI консультант, К6 – зв'язок з оператором, К7 – відгуки, К8 – контакти;

3) (О1) – перехід до гілки нових замовлень для оператора;

4) (О2) – перехід до гілки перегляду всіх замовлень оператором;

5) (О3) – перехід до гілки статистики для оператора;

6) (ГМ) – повернення до головного меню з вибором ролі.

Основна гілка програми (рисунок 4.2) описує початковий етап взаємодії користувача з ботом. Після введення команди /start або повернення з інших гілок через з'єднувач ГМ відбувається очищення поточного стану користувача та виведення привітального повідомлення з кнопками головного меню. Користувач обирає свою роль: клієнт або оператор. При виборі ролі клієнта відбувається перехід до відповідної гілки через з'єднувач К. При виборі ролі оператора система пропонує обрати метод авторизації: за Telegram ID, за особистим токеном або за кодом з електронної пошти. Залежно від обраного методу виконується відповідна процедура перевірки. У разі невдалої авторизації користувач може повторити спробу або повернутися до головного меню. Після успішної авторизації оператор отримує доступ до панелі керування з можливістю обрання типу взаємодії: перегляд нових замовлень (перехід до НЗ), перегляд усіх замовлень (перехід до ВЗ), перегляд статистики (перехід до СТ) або повернення до головного меню (ГМ).

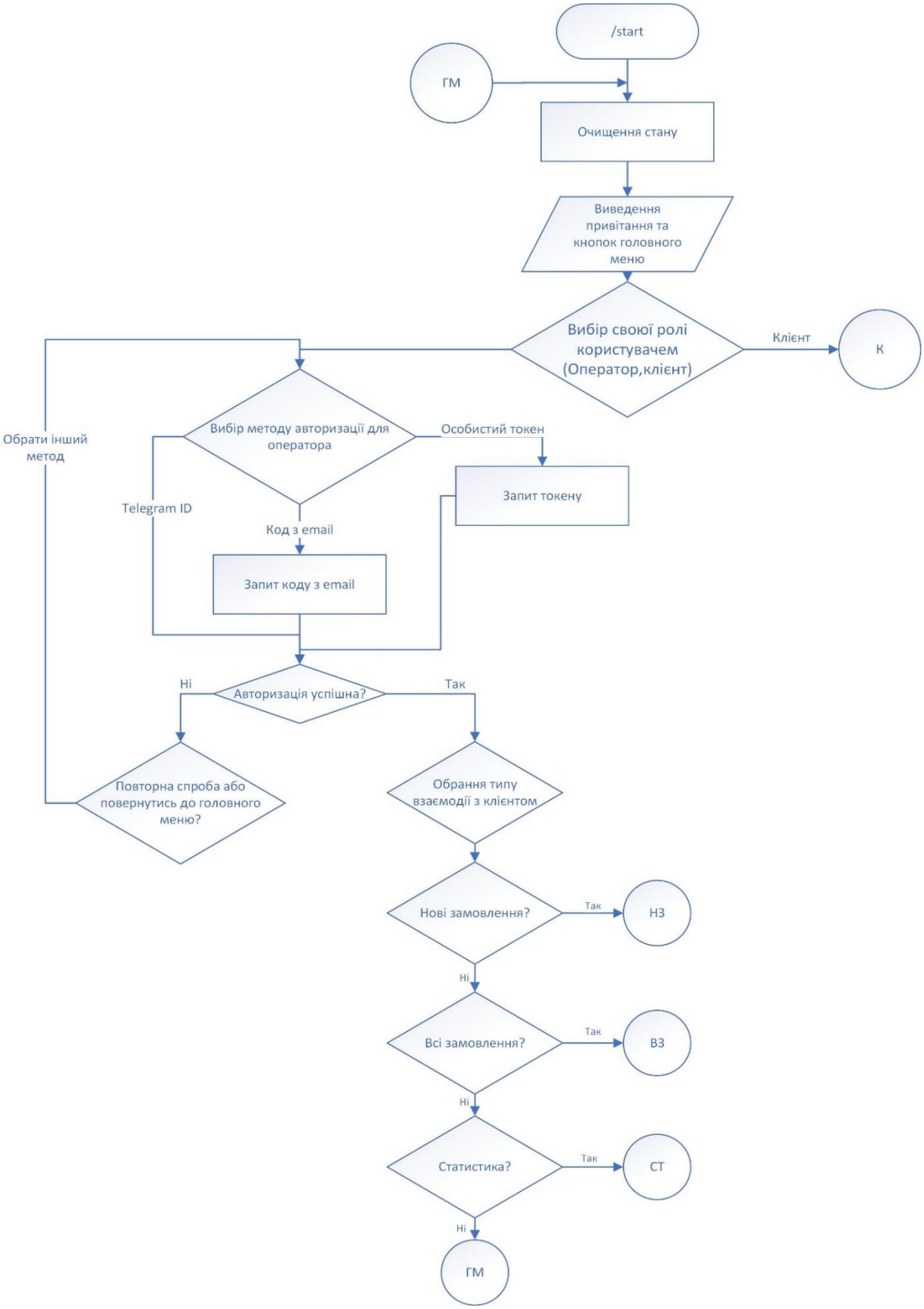

Рисунок 4.2 – Основна гілка програми, що бере початок з команди /start

Гілка меню клієнта (рисунок 4.3) описує взаємодію користувача з системою після вибору ролі клієнта. Вхід до цієї гілки здійснюється через з'єднувач К з основної гілки програми. Після переходу клієнту виводиться головне меню з переліком доступних функцій. Далі алгоритм послідовно перевіряє вибір дії клієнтом через серію умовних блоків. При виборі опції «Нове замовлення» відбувається перехід до гілки оформлення замовлення через з'єднувач К1. Якщо клієнт обирає «Мої замовлення», здійснюється перехід до гілки перегляду власних замовлень через з'єднувач К2. Опція «Відстежити» забезпечує перехід до функції відстеження статусу замовлення через з'єднувач К3. При виборі «Тарифи» клієнт переходить до перегляду цінової політики компанії через з'єднувач К4. Опція «AI Консультант» забезпечує доступ до інтелектуального помічника на базі штучного інтелекту через з'єднувач К5. Вибір «Зв'язок з оператором» дозволяє клієнту встановити комунікацію з живим оператором через з'єднувач К6. Опція «Відгуки» надає можливість залишити або переглянути відгуки про послуги через з'єднувач К7. При виборі «Контакти» клієнт отримує контактну інформацію компанії через з'єднувач К8. Якщо жодна з опцій не обрана, алгоритм повертається до головного меню через з'єднувач ГМ.

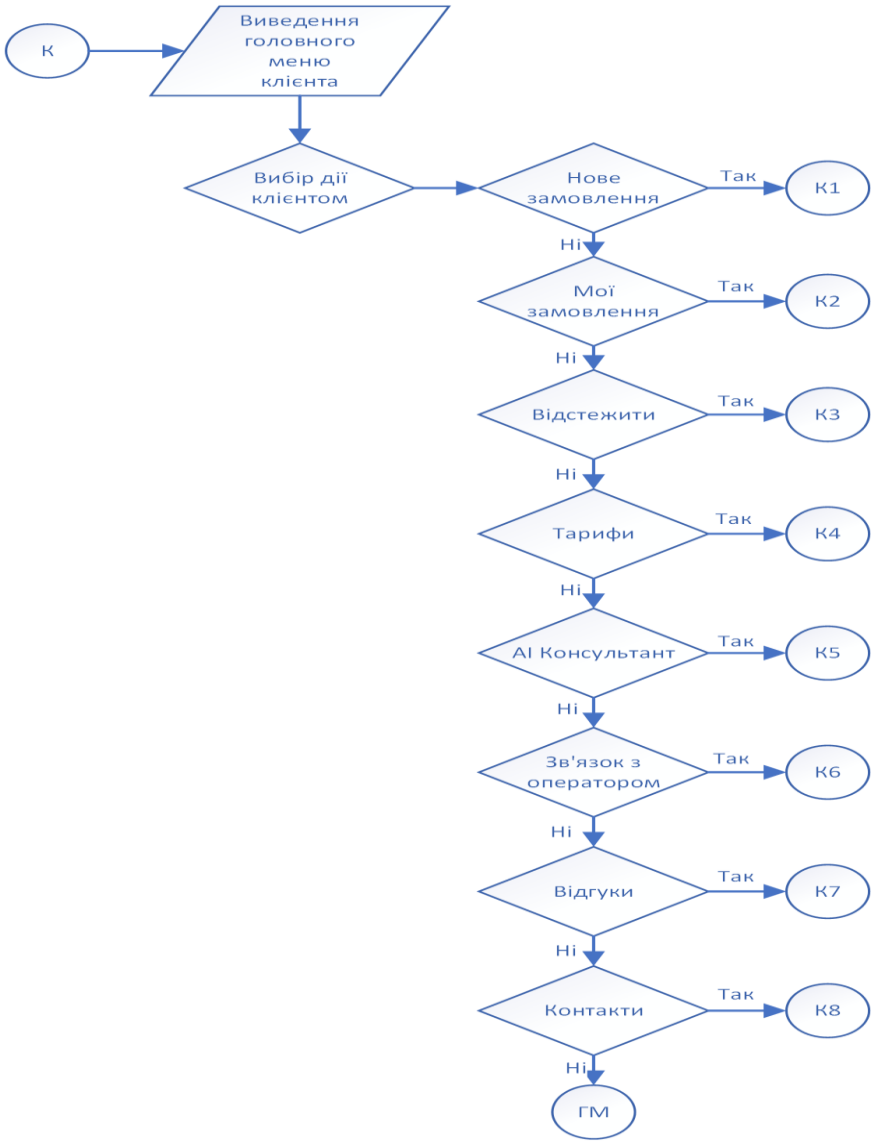

Рисунок 4.3– Гілка меню клієнта з восьми пунктів

Гілка створення замовлення описує десятикроковий процес оформлення нового замовлення клієнтом. Вхід до цієї гілки здійснюється через з'єднувач К1 з меню клієнта. На першому кроці клієнт вводить своє ім'я для накладної. На другому та третьому кроках вводяться адреси відправлення та доставки відповідно. Після введення адрес система виконує розрахунок відстані за допомогою функції get_distance та генерує посилання на карту OpenStreetMap для візуалізації маршруту. На четвертому кроці клієнту виводиться маршрут з можливістю підтвердження або зміни адрес. При підтвердженні адрес клієнт переходить до п'ятого кроку – вибору типу вантажу. На шостому кроці обирається категорія ваги, на сьомому – вводиться опис вантажу. Восьмий крок передбачає опціональне

завантаження фото вантажу. На дев'ятому та десятому кроках клієнт вводить контактний номер телефону та email (опціонально). Після введення всіх даних виконується розрахунок вартості доставки та виводиться підсумок замовлення для перевірки. При підтвердженні замовлення воно зберігається в базі даних за допомогою функції save_order_to_db, операторам надсилається сповіщення про нове замовлення, а клієнт отримує повідомлення про успішне створення замовлення.

Гілка AI консультанта (рисунок 4.4) описує процес взаємодії клієнта з інтелектуальним помічником на базі штучного інтелекту. Вхід до цієї гілки здійснюється через з'єднувач K5 з меню клієнта після натискання кнопки «AI Консультант». На початку роботи користувачу виводиться привітальне повідомлення AI-консультанта з інструкціями щодо використання. Далі система переходить у режим очікування запиту від користувача. Алгоритм перевіряє, чи натиснуто кнопку «Завершити»: якщо так, здійснюється перехід до головного меню через з'єднувач ГМ. Якщо користувач вводить текстовий запит, виконується формування запиту з системним промптом та контекстом розмови для забезпечення релевантних відповідей у сфері логістичних послуг. Сформований запит відправляється до Google Gemini API для обробки нейронною мережею. Після отримання результату алгоритм перевіряє успішність відповіді: у разі виникнення помилки з'єднання або обробки користувачу виводиться відповідне повідомлення про помилку; при успішному отриманні відповіді вона виводиться користувачу. Після виведення відповіді або повідомлення про помилку алгоритм повертається до етапу очікування наступного запиту, що забезпечує безперервний діалог з AI-консультантом (рисунок 4.4).

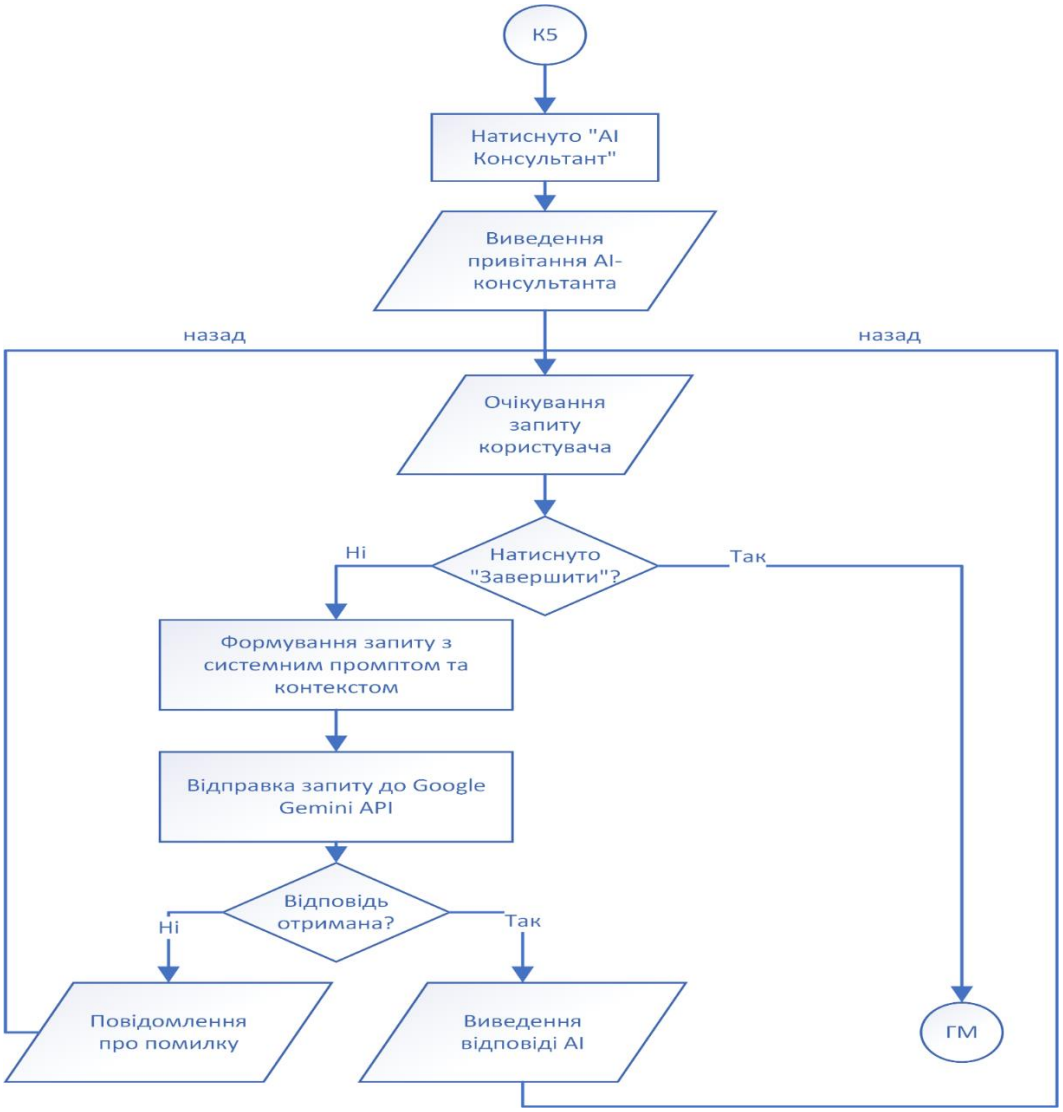

Рисунок 4.4 – Гілка AI-консультанта на базі Google Gemini

Гілка панелі оператора (рисунок 4.5) описує процес обробки нових замовлень авторизованим оператором системи. Вхід до цієї гілки здійснюється через з'єднувач НЗ з основної гілки програми після успішної авторизації оператора. На початку виконується запит до бази даних для отримання замовлень зі статусом [НОВЕ]. Далі алгоритм перевіряє наявність нових замовлень: якщо такі відсутні, виводиться відповідне повідомлення «Немає нових замовлень» і здійснюється повернення до головного меню через з'єднувач ГМ. За наявності нових замовлень оператору виводиться їх список, після чого він обирає конкретне замовлення для детального перегляду. Після виведення деталей замовлення оператор має можливість обрати одну з дій. При виборі опції «Прийняти» відбувається зміна статусу замовлення на [ОК], надсилається сповіщення клієнту про прийняття замовлення, після чого

здійснюється перехід до головного меню. При виборі опції «Призначити водія» оператор послідовно обирає водія та транспортний засіб, дані зберігаються в системі. Опція «Генерувати ТТН» ініціює виклик функції generate_ttn для формування товарно-транспортної накладної, після чого сформований файл відправляється користувачу. При виборі кнопки «Назад» оператор повертається до списку замовлень (рисунок 4.5).

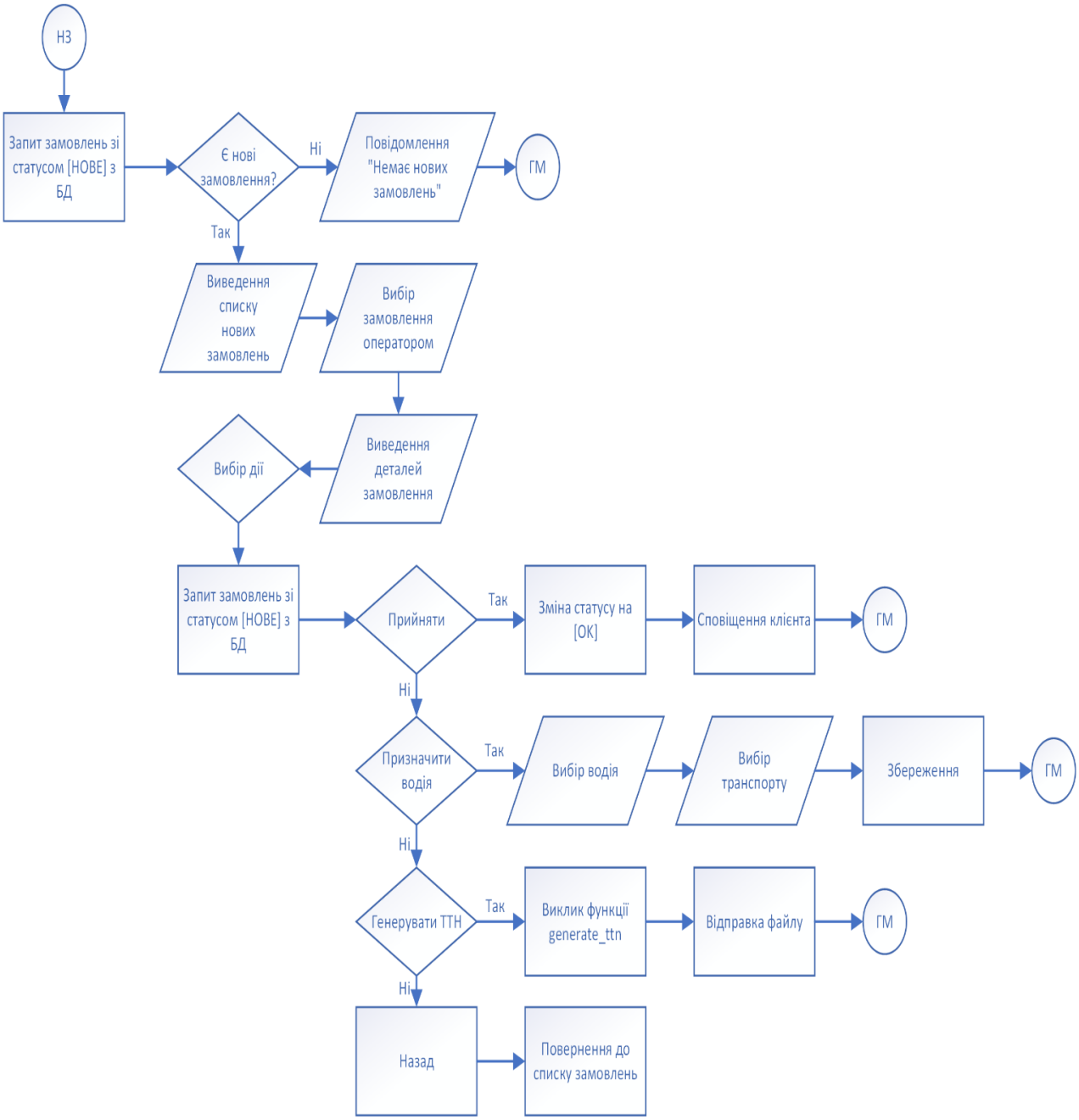

Рисунок 4.5 – Гілка панелі оператора

Гілка перегляду всіх замовлень (з'єднувач ВЗ) описує процес відображення повного списку замовлень для оператора. Вхід до цієї гілки здійснюється через з'єднувач ВЗ з панелі оператора. На початку виконується запит усіх замовлень з бази даних. Далі оператору виводиться список замовлень з можливістю фільтрації за різними критеріями. Після перегляду здійснюється повернення до головного меню через з'єднувач ГМ.

Гілка перегляду статистики (з'єднувач СТ) описує процес відображення статистичних даних для оператора. Вхід до цієї гілки здійснюється через з'єднувач СТ з панелі оператора. На початку виконується запит статистичних даних з бази даних, що включає інформацію про кількість замовлень, виконані доставки, фінансові показники тощо. Далі оператору виводиться зведена статистика у зручному форматі. Після перегляду здійснюється повернення до головного меню через з'єднувач ГМ (рисунок 4.5.1)

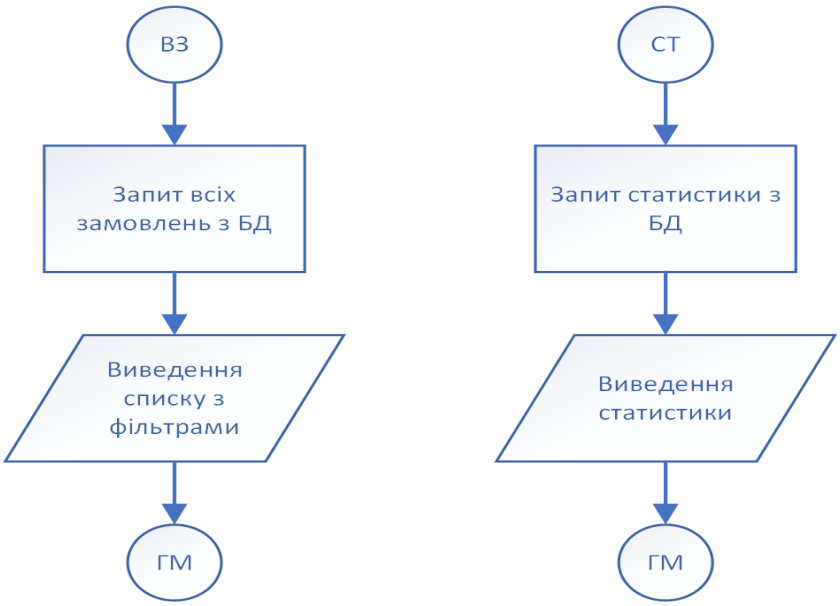

Рисунок 4.5.1 – Гілки панелі всі замовлення і гілка статистики

Представлений алгоритм роботи функції save_order_to_db відповідає за збереження нового замовлення до бази даних MongoDB. Функція розпочинає свою роботу з отримання даних замовлення, сформованих на етапі оформлення клієнтом. Далі виконується генерація унікального ідентифікатора замовлення order_id за

допомогою алгоритму UUID, що гарантує унікальність кожного запису. На наступному етапі формується документ замовлення order_doc, який містить усі необхідні поля для збереження. Після цього встановлюється підключення до бази даних MongoDB. Сформований документ вставляється до колекції orders за допомогою методу insert_one. Алгоритм перевіряє успішність операції вставки: у разі успішного збереження функція повертає згенерований order_id для подальшого використання в системі; у разі виникнення помилки виконується логування помилки для подальшого аналізу, після чого функція завершує свою роботу (рисунок4.6).

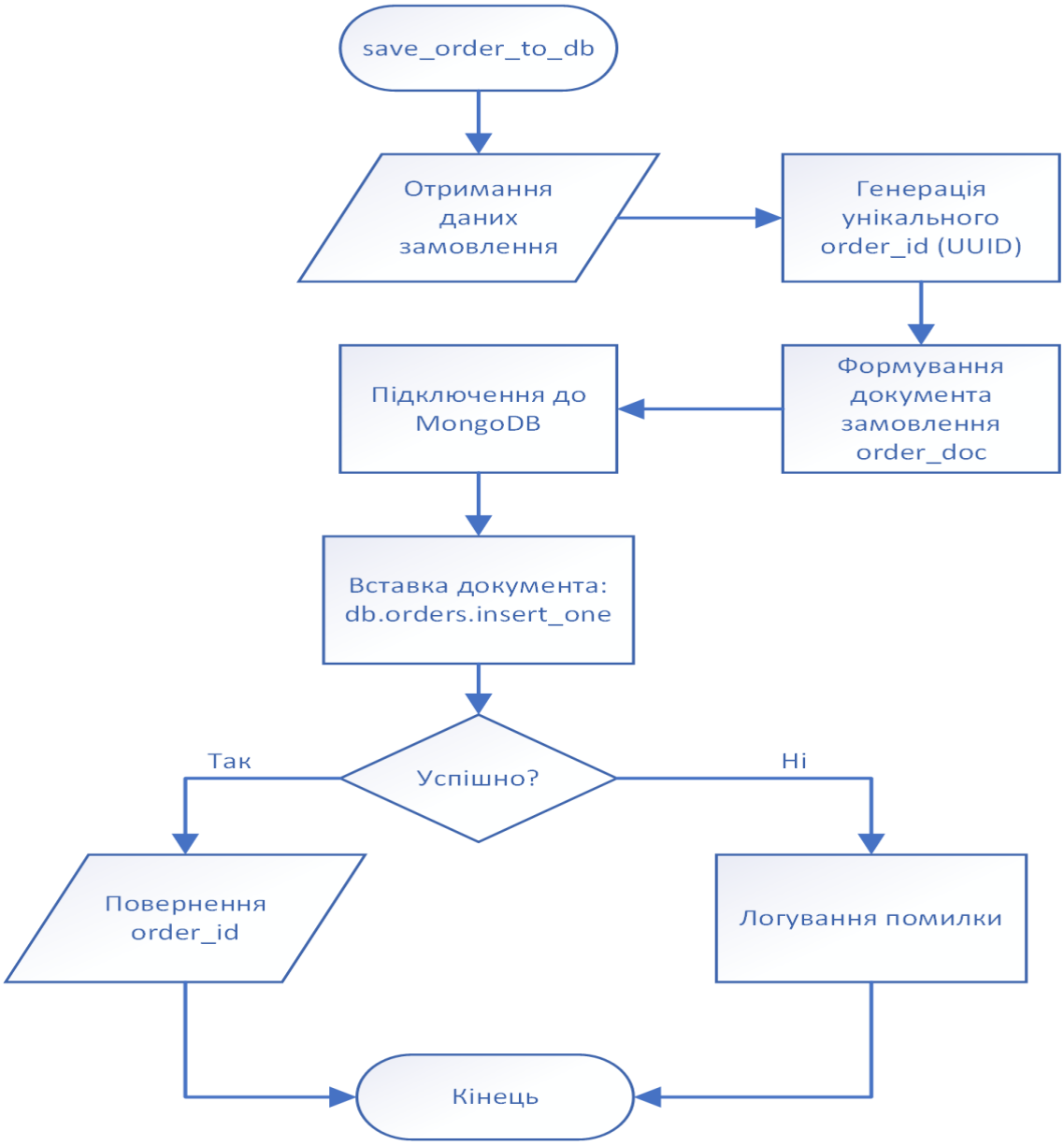

Рисунок 4.6 – Функція збереження замовлення в базу даних

Алгоритм роботи функції авторизації оператора забезпечує перевірку прав доступу до адміністративної панелі системи. Функція розпочинає роботу з виведення доступних методів авторизації. Першим перевіряється метод авторизації за Telegram ID: якщо користувач обирає цей метод, виконується пошук відповідного ідентифікатора в базі даних; у разі успішного знаходження оператор отримує доступ до панелі оператора, в іншому випадку виводиться повідомлення про помилку. Якщо обрано авторизацію за токеном, система запитує персональний токен у користувача, після чого виконує пошук у базі даних; при збігу токена надається доступ до панелі, при невідповідності – виводиться повідомлення про помилку. Найскладнішим є метод авторизації через електронну пошту: спочатку запитується email користувача, виконується пошук у базі даних; якщо оператора з таким email знайдено, генерується 6-значний код підтвердження, який відправляється на вказану адресу за допомогою функції send_email та зберігається в кеші системи. Далі користувач вводить отриманий код, і система перевіряє його відповідність: при вірному коді надається доступ до панелі оператора, при невірному — виводиться повідомлення про помилку з можливістю повернення до вибору методу авторизації (рисунок 4.7).

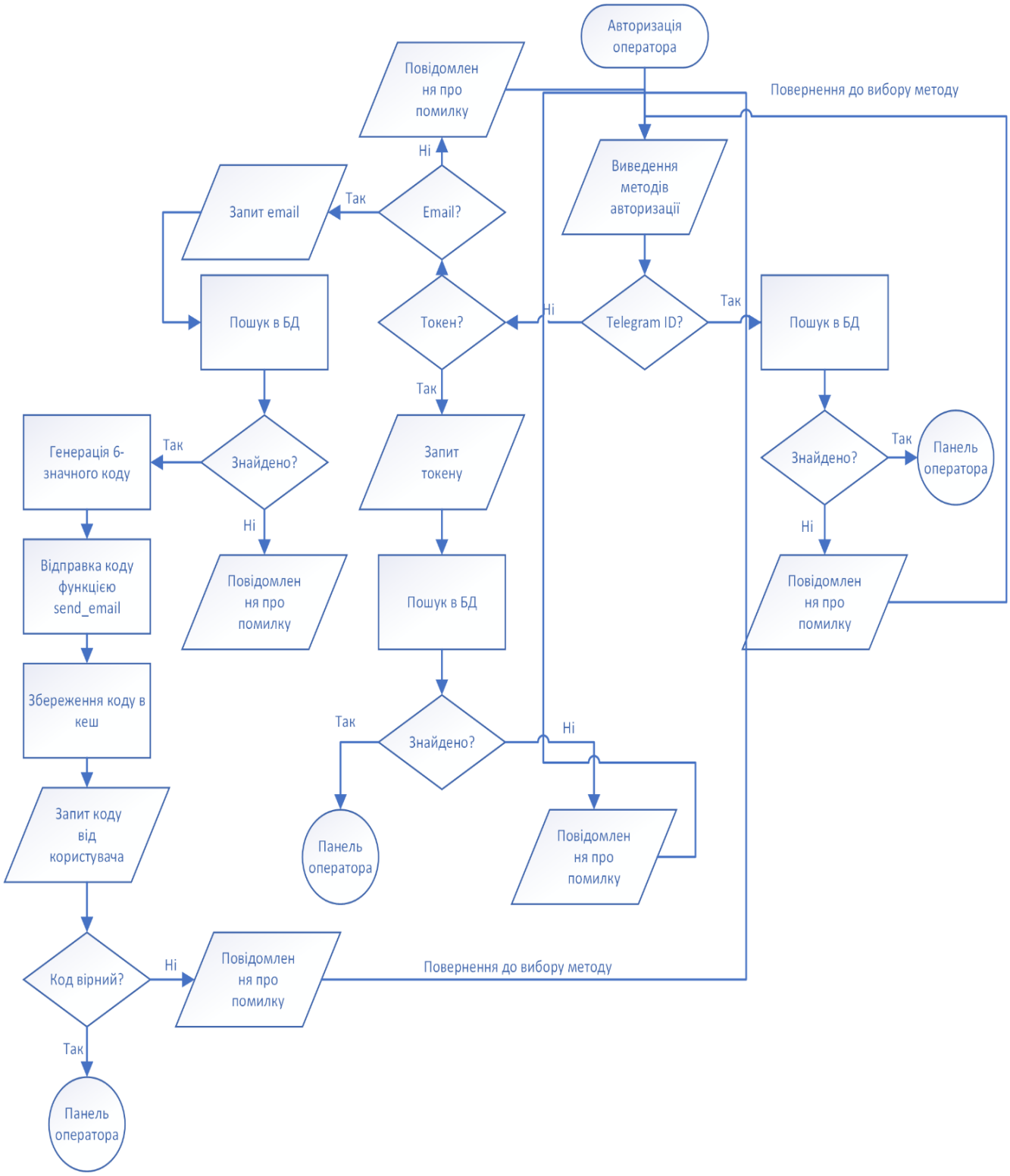

Рисунок 4.7 – Функція авторизації оператора (*authorize_operator)*

Алгоритм роботи функції get_distance відповідає за розрахунок відстані між двома географічними точками маршруту доставки. Функція розпочинає роботу з

отримання вхідних параметрів – адрес відправлення та доставки. На першому етапі виконується геокодування адреси відправлення через Nominatim API для отримання географічних координат. Алгоритм перевіряє успішність отримання координат: у разі успіху координати зберігаються для подальшого використання; якщо API не повернув результат, використовуються локальні координати міста з константи LOCAL_COORDS як резервний варіант. Аналогічна процедура виконується для адреси доставки: геокодування через Nominatim API з перевіркою результату та можливим використанням локальних координат. Після отримання координат обох точок виконується запит до OpenRouteService API для розрахунку реальної відстані маршруту з урахуванням дорожньої мережі. Якщо відповідь від API отримана успішно, виконується парсинг результату; у разі недоступності сервісу застосовується альтернативний розрахунок відстані за формулою Haversine, яка обчислює відстань по прямій між двома точками на сфері. На завершальному етапі виконується конвертація отриманого значення з метрів у кілометри, після чого функція повертає розраховану відстань distance_km (рисунок 4.8).

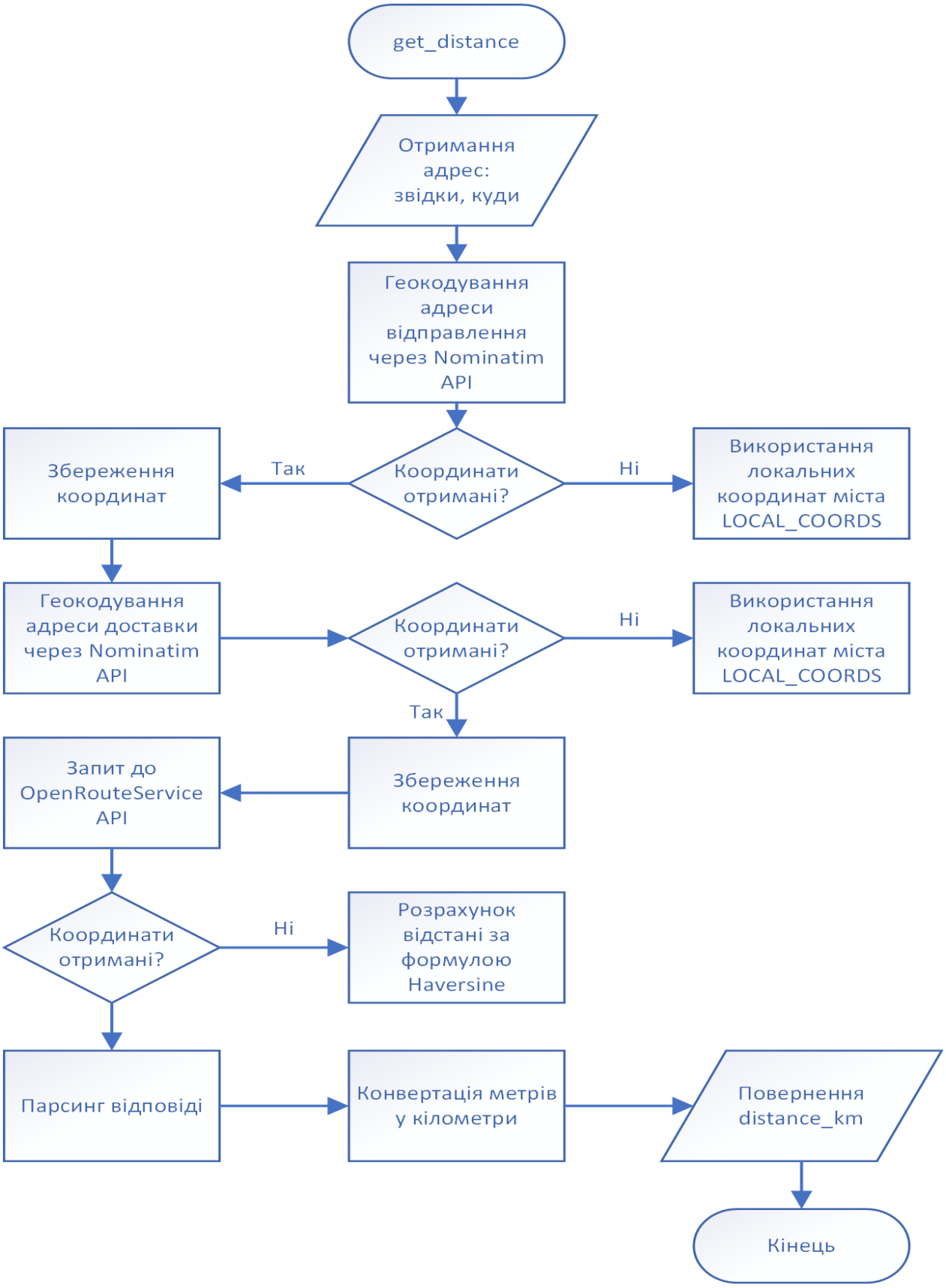

Рисунок 4.8 – Функція розрахунку відстані між містами (*get_distance)*

Наступним нам представлено алгоритм роботи функції generate_ttn, яка відповідає за автоматичну генерацію товарно-транспортної накладної (ТТН) на основі даних замовлення. Функція розпочинає роботу з отримання даних замовлення, що включають ідентифікатор замовлення order_id, інформацію про маршрут, характеристики вантажу та контактні дані. Далі виконується завантаження попередньо підготовленого шаблону документа template_ttn.docx, який містить структуру накладної з плейсхолдерами для динамічних даних. На наступному етапі формується словник для заміни плейсхолдерів, що включає такі поля: ORDER_ID (ідентифікатор замовлення), DATE (дата створення), SENDER (дані відправника), RECEIVER (дані отримувача), CARGO (опис вантажу), WEIGHT (вага) та COST (вартість доставки). Після формування словника виконується заміна всіх плейсхолдерів у документі на реальні дані замовлення. Далі генерується унікальне ім'я файлу за шаблоном TTN_{order_id}_{date}.docx для забезпечення унікальності кожного документа. Сформований документ зберігається у тимчасову папку системи, після чого функція повертає шлях до згенерованого файлу для подальшої відправки оператору або клієнту (рисунок 4.9).

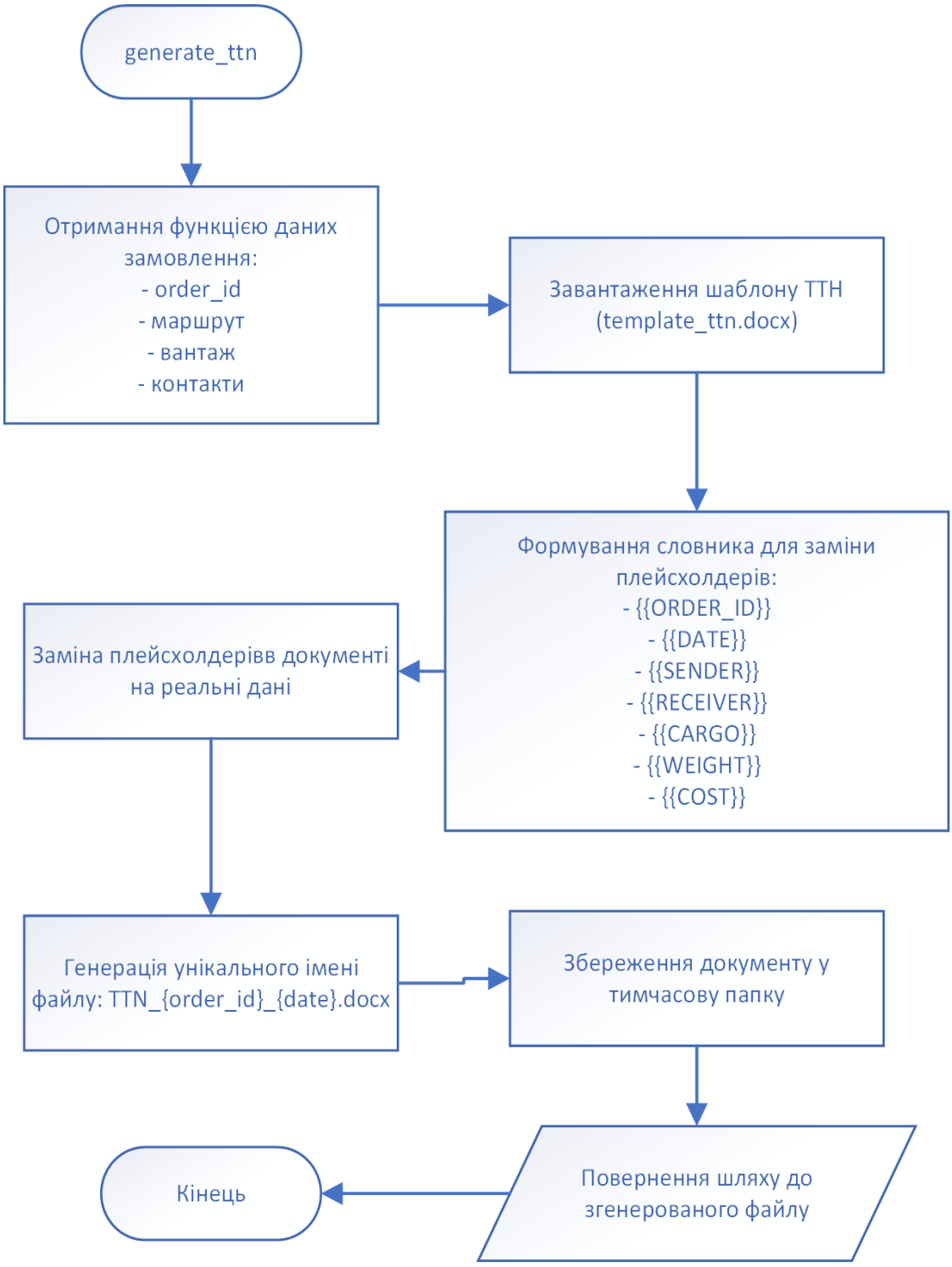

Рисунок 4.9 – Функція генерації ТТН-накладної (*generate_ttn)*

І нарешті представлено алгоритм роботи функції send_email (див. рисунок 4.10), яка забезпечує відправку електронних листів через SMTP-протокол. Функція розпочинає роботу з отримання вхідних параметрів: адреси отримувача (to), теми листа (subject), тексту повідомлення (body) та опціонального вкладення (attachment). На наступному етапі формується об'єкт повідомлення типу MIMEMultipart, який дозволяє створювати складені електронні листи. Алгоритм перевіряє наявність вкладення: якщо файл для прикріплення відсутній, процес переходить безпосередньо до підключення до SMTP сервера; якщо вкладення присутнє, воно додається до повідомлення, після чого також виконується підключення до сервера. Після спроби підключення до SMTP сервера перевіряється успішність з'єднання: у разі невдачі виконується логування помилки, функція повертає значення False та завершує роботу. При успішному підключенні виконується авторизація на сервері за допомогою методу login з використанням облікових даних. Далі здійснюється відправка сформованого повідомлення, після чого з'єднання з сервером закривається. Функція повертає значення True, що свідчить про успішну відправку листа (рисунок 4.10) .

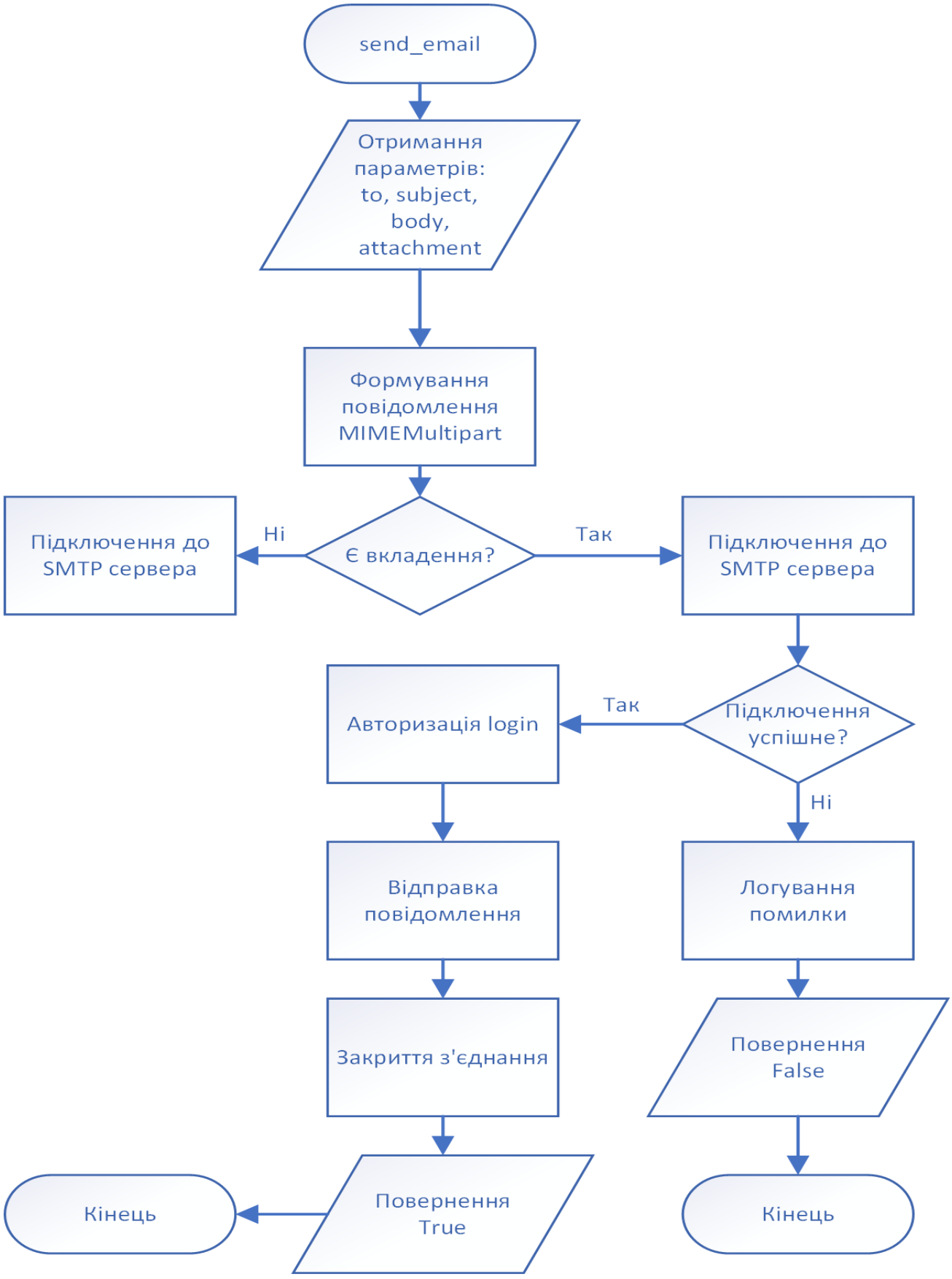

Рисунок 4.10 – Функція відправки email-повідомлень (*send_email)*

4.3.3 Особливості програмної реалізації

При розробці програмного забезпечення важлива увага приділяється забезпеченню стабільності функціонування платформи під час інтенсивного навантаження. Асинхронний підхід на основі asyncio дозволяє ефективно обробляти одночасні запити від множини користувачів без блокування основного циклу виконання.

Для підвищення надійності системи реалізовано механізм обробки помилок із логуванням усіх критичних подій через модуль logging. Всі запити до зовнішніх API (OpenRouteService, Google Gemini) обгорнуті в блоки try-except з встановленими таймаутами (10 секунд) для запобігання зависань.

Реалізовано багаторівневий підхід до отримання координат міст: локальна база координат понад 50 міст України забезпечує миттєвий доступ без запитів до API; для інших міст виконується запит до ORS Geocoding API з кешуванням результатів.

Безпека даних забезпечується шляхом зберігання конфіденційної інформації (токен бота, API-ключі, пароль оператора) у окремому конфігураційному файлі config.py, який виключається з системи контролю версій.

4.3.4 Використовувані технічні засоби

У межах даного дослідження для розгортання та апробації функціоналу програмного забезпечення чатбота і системи управління бази даних застосовується серверне обладнання на основі портативного комп'ютера з попередньо інстальованою та конфігурованою СУБД MongoDB.

Технічні характерстики програмного забезпечення:

1) ноутбук – операційна система Windows 10 Pro, версія 22H2;
2) центральний процесор Intel Core i5-8300H;
3) 16,0 Гб оперативна пам’ять;
4) 500 МБ – вільне місце на диску для встановлення та стабільної працездатності системи;
5) мережеве з’єднання: стабільний доступ до інтернету.

Для використання функцій програми з боку клієнта необхідний пристрій з встановленим додатком Telegram (Android, iOS або Desktop версії) та можливістю інтернет-з'єднання.

4.3.5 Виклик і завантаження

Для ініціалізації роботи чатботу на сервері використовується файл **main.py**, який активує основний функціонал програми командою **python main.py**. Одночасно на сервері має бути запущено службу бази даних MongoDB, що необхідна для зберігання інформації.

З боку користувача робота починається після відкриття бота в Telegram та введення команди **/start**. Доступ до чатботу здійснюється через обліковий запис @KanibolotskiyVV_bot **.**

4.3.6 Вхідні та вихідні дані програми чатботу

4.3.6.1 Вхідні дані

Для початку роботи з ботом користувачем використовується команда /start. Під час створення замовлення програма отримує з діалогу текстові повідомлення від користувача з даними: ім'я для накладної, адреси відправлення та доставки, тип вантажу, категорія ваги, опис, контактний телефон, email (опціонально), а також фотографії вантажу (опціонально).

Програма автоматично отримує Telegram ID користувача для ідентифікації та зв'язку замовлень з клієнтами. Для авторизації операторів використовуються токени або email з кодом верифікації.

4.3.6.2 Вихідні дані

Взаємодія з програмою чатботу та навігація за його гілками здійснюється шляхом натискання користувачем кнопок клавіатури (Reply Keyboard або Inline Keyboard), що розміщуються ботом під рядком для введення повідомлень або безпосередньо у повідомленнях.

Програма чатботу надає текстову інформацію користувачам у процесі роботи: підтвердження замовлень з розрахованою вартістю та посиланням на карту маршруту, сповіщення про зміну статусу, відповіді AI-консультанта, інформацію про тарифи та контакти компанії.

Програма генерує та надсилає ТТН-накладні у форматі .docx на email клієнтів. Також система надсилає email-повідомлення про підтвердження замовлення, зміну статусу та інші події.

4.4 Функціональний огляд програмного забезпечення з прикладами роботи

Програмний код, створеного чатботу замовлень і комунікації з клієнтом логістичної компанії, наведено у додатку А, має формат діалогу з користувачем.

4.4.1 Функції чатботу для клієнта

Потрапляючи на початок діалогу з чат-ботом, користувач має можливість обрати одну з двох ролей: клієнт або оператор (рисунок 4.11).

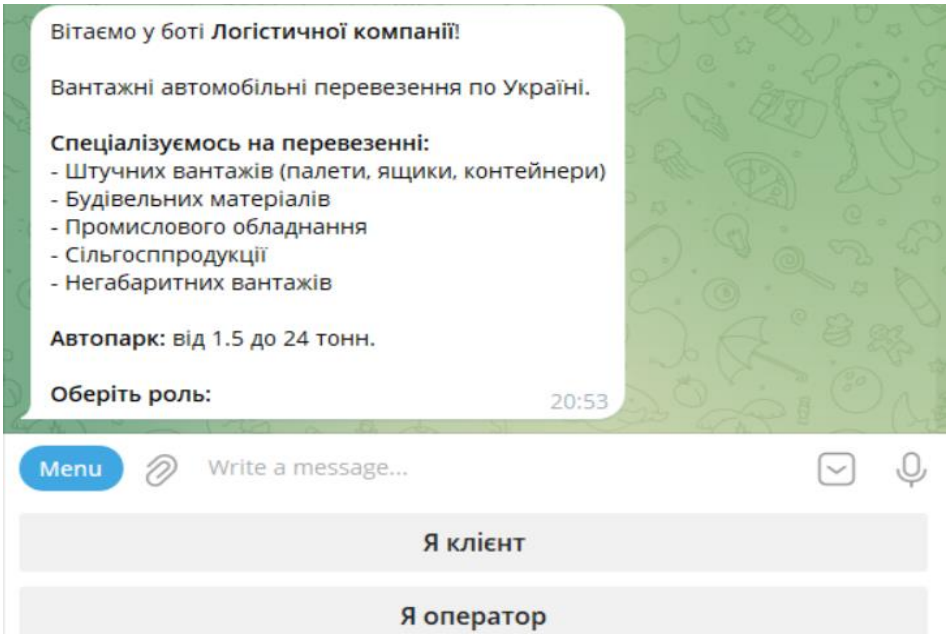

Рисунок 4.11 – Вибір ролі користувачем

Наступним є процес оформлення замовлення на перевезення вантажу. Користувач вводить адресу відправлення (крок 2) та адресу доставки (крок 3). Система автоматично розраховує відстань маршруту через інтеграцію з OpenRouteService API та надає посилання для перегляду маршруту на карті. На кроці 4 користувач перевіряє введені дані та може їх змінити за потреби (рисунок 4.12).

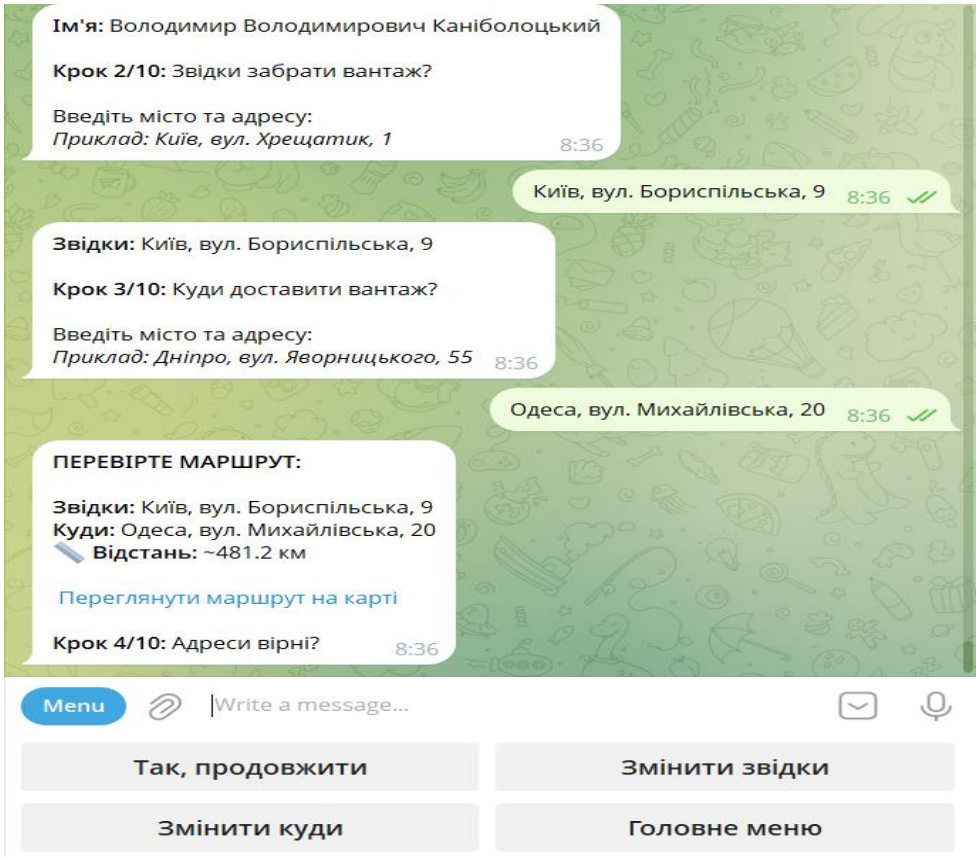

Рисунок 4.12 – Процес оформлення замовлення: введення адрес та перевірка маршруту

Крок вибору типу вантажу. Система пропонує п'ять категорій: штучний вантаж, будівельні матеріали, сільгосппродукція, промислове обладнання та негабаритний вантаж. Кожна категорія має власний тарифний коефіцієнт, який впливає на розрахунок вартості перевезення (рисунок 4.13).

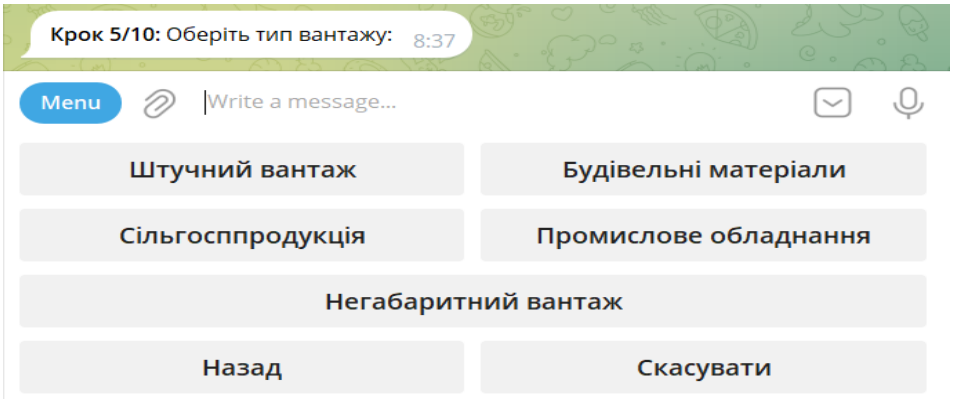

Рисунок 4.13 – Вибір типу вантажу

Після вибору типу вантажу клієнту пропонується обрати вагову категорію (рисунок 4.14). Система надає шість діапазонів ваги: до 1.5 тонн, 1.5-3 тонни, 3-7 тонн, 7-12 тонн, 12-20 тонн та 20-24 тонн. Обрана категорія використовується для розрахунку вартості перевезення та підбору відповідного транспортного засобу оператором (рисунок 4.14.).

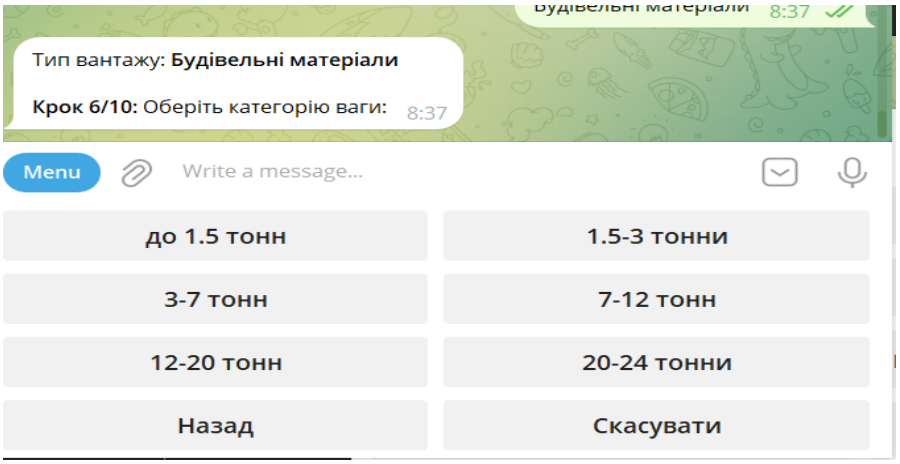

Рисунок 4.14 – Вибір вагової категорії вантажу

Далі клієнт вводить текстовий опис вантажу із зазначенням його характеристик. Наступним кроком є можливість додати фотографію вантажу для більш точної оцінки або пропустити цей крок. Після завантаження фото система підтверджує його збереження та пропонує ввести контактний телефон для зв'язку з клієнтом (рисунок 4.15).

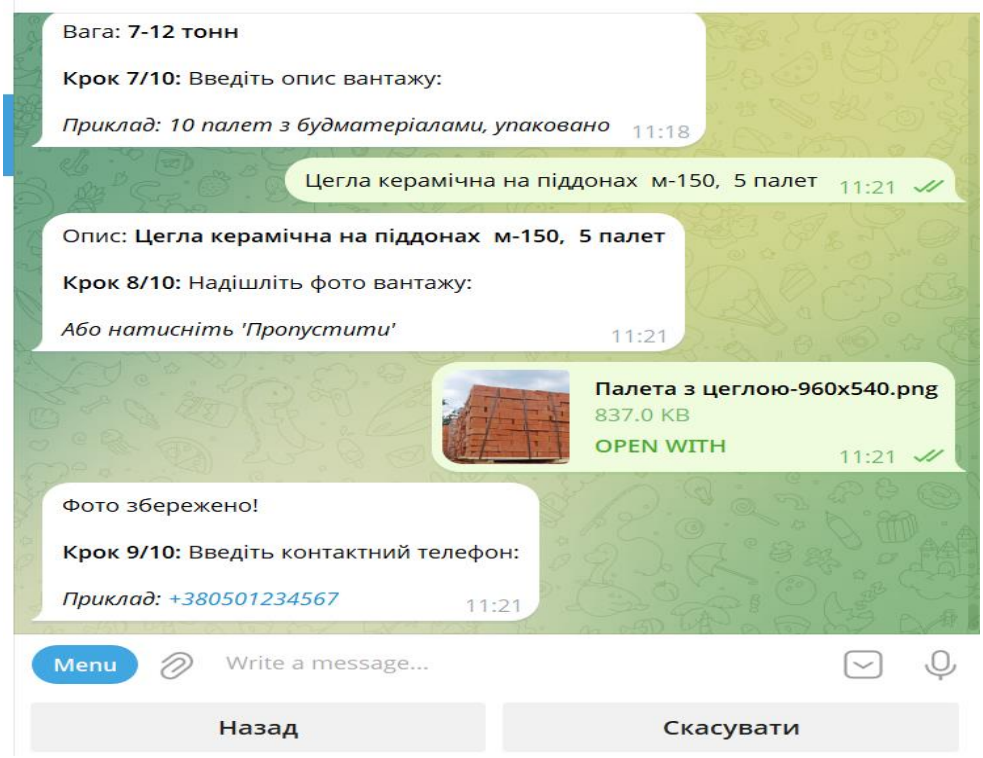

Рисунок 4.15 – Введення опису вантажу, додавання фото та контактного телефону

На завершальному етапі клієнт вводить контактний телефон для зв'язку. Останнім кроком є введення електронної пошти для отримання ТТН-накладної або можливість пропустити цей крок. Після введення всіх даних система переходить до перевірки замовлення (рисунок 4.16).

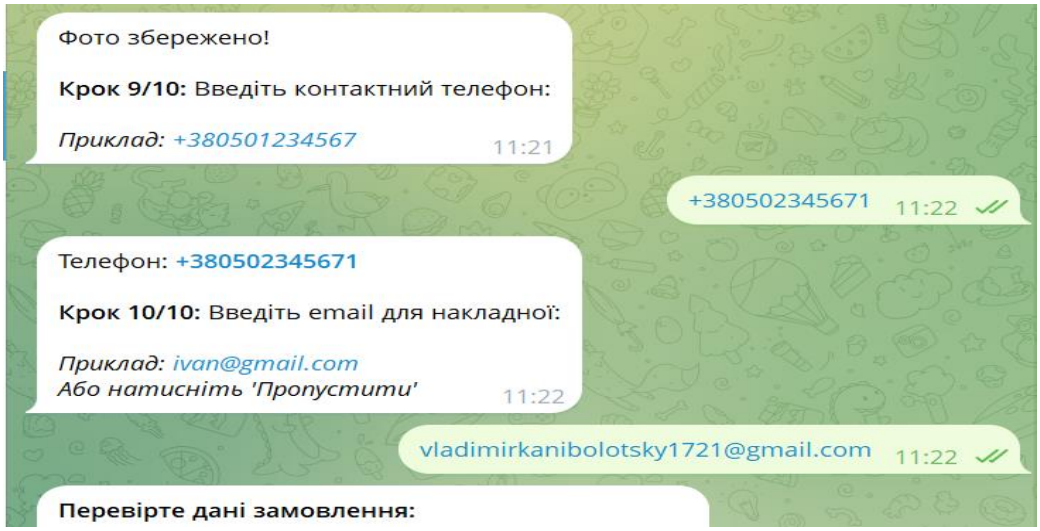

Рисунок 4.16 – Введення контактних даних клієнта

Наступним кроком система формує підсумок замовлення для перевірки. Клієнту відображається повна інформація: ПІБ відправника, адреси маршруту з посиланням на карту, характеристики вантажу, контактні дані та детальний розрахунок вартості. Система автоматично обчислює вартість на основі відстані, ваги та типу вантажу, а також розраховує орієнтовний час доставки. У розділі оплати вказано суму передоплати, реквізити та посилання для онлайн-оплати. Клієнт може підтвердити замовлення або скасувати його (рисунок 4.17).

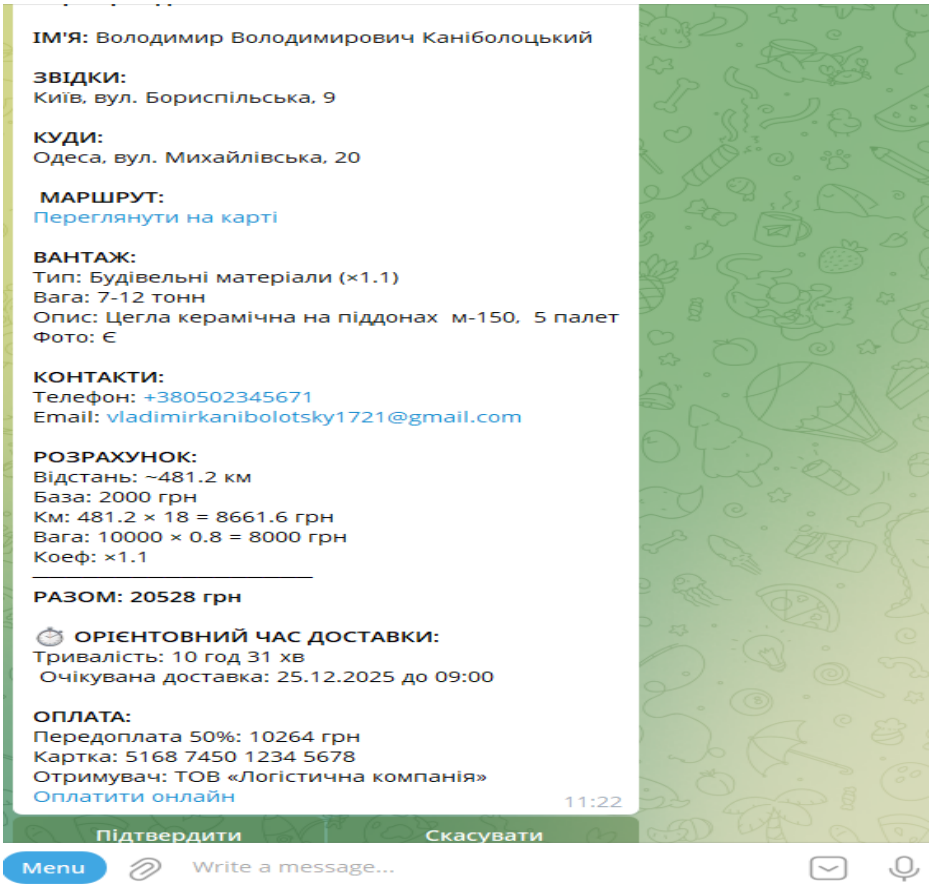

Рисунок 4.17 – Підсумок замовлення з розрахунком вартості

Після підтвердження замовлення всі дані зберігаються у колекції orders бази даних MongoDB у форматі JSON-документа. Документ містить унікальний ідентифікатор замовлення, Telegram ID клієнта, контактні дані, адреси відправлення та доставки, параметри вантажу, розраховану вартість, поточний статус та історію його змін. Фотографія вантажу зберігається у закодованому форматі Base64 (рисунок 4.18) .

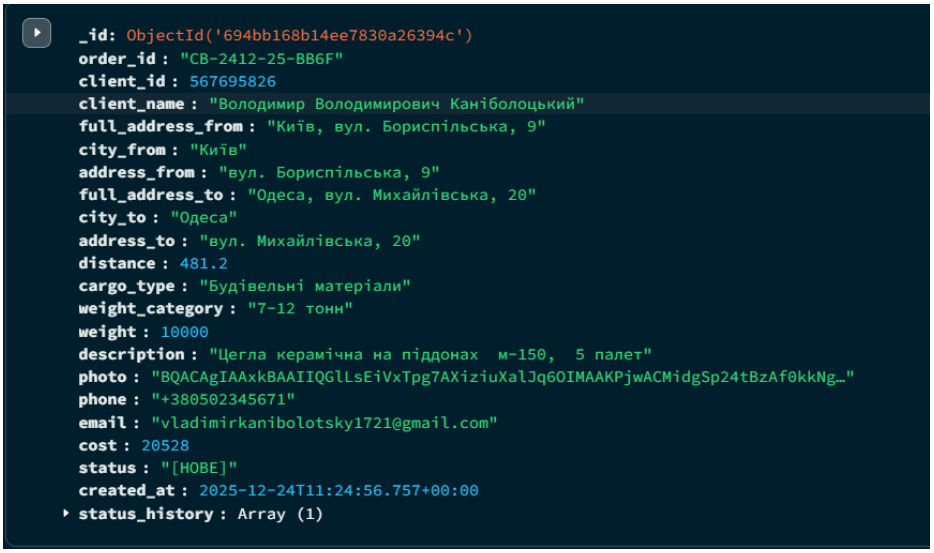

Рисунок 4.18 – Структура документа замовлення у базі даних MongoDB

Система автоматично зберігає інформацію про клієнтів у колекції users. Кожен документ містить Telegram ID користувача, username, ім'я, дату реєстрації та час останньої активності. Ці дані використовуються для ідентифікації клієнтів та надсилання сповіщень про статус замовлень (рисунок 4.19).

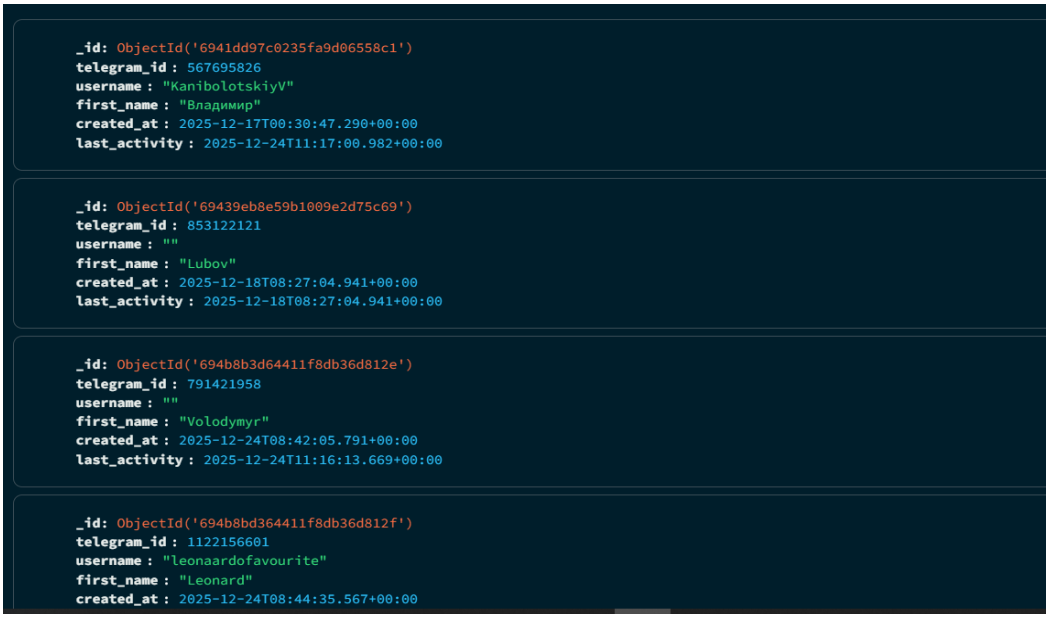

Рисунок 4.19 – Колекція клієнтів у базі даних MongoDB

У колекції drivers зберігається інформація про водіїв компанії. Кожен документ містить унікальний ідентифікатор водія, ПІБ, контактний телефон, номер

посвідчення водія, стаж роботи, категорію прав та статус активності. Ці дані використовуються оператором при призначенні водія на замовлення (рисунок 4.20).

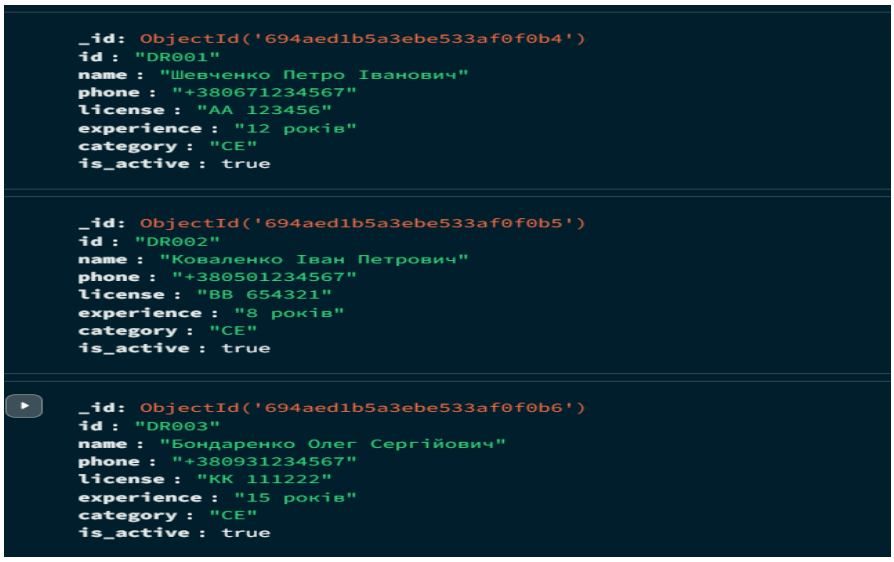

Рисунок 4.20 – Колекція водіїв у базі даних MongoDB

У колекції vehicles зберігається інформація про автопарк компанії. Кожен документ містить унікальний ідентифікатор, державний номер, марку та модель автомобіля, тип (малотоннажний, середньотоннажний), вантажопідйомність, об'єм кузова та статус активності. Ці дані використовуються оператором при призначенні транспорту на замовлення відповідно до ваги та типу вантажу (рисунок 4.21).

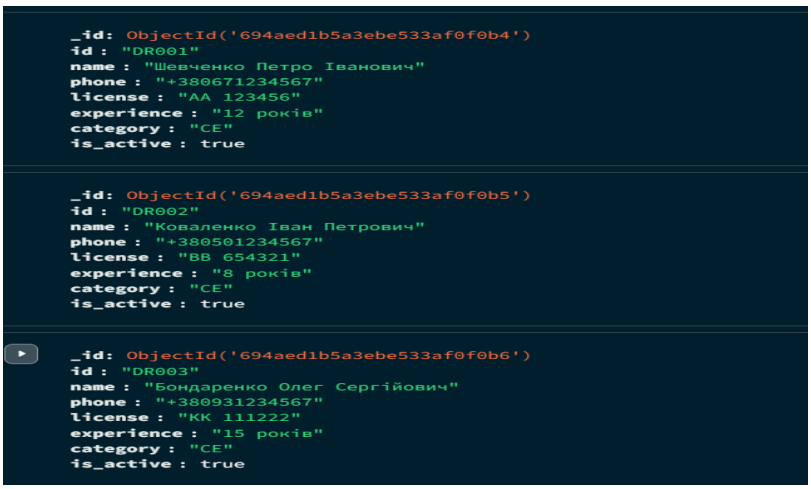

Рисунок 4.21 – Колекція транспортних засобів у базі даних MongoDB

Після підтвердження замовлення клієнт отримує повідомлення про успішне створення заявки. Система відображає унікальний номер замовлення, маршрут, відстань та вартість. Також надаються реквізити для оплати передоплати з посиланням на сервіс Приват24 для онлайн-оплати. Клієнт отримує інформацію про те, що реквізити та накладна будуть продубльовані на вказану електронну пошту (рисунок 4.22).

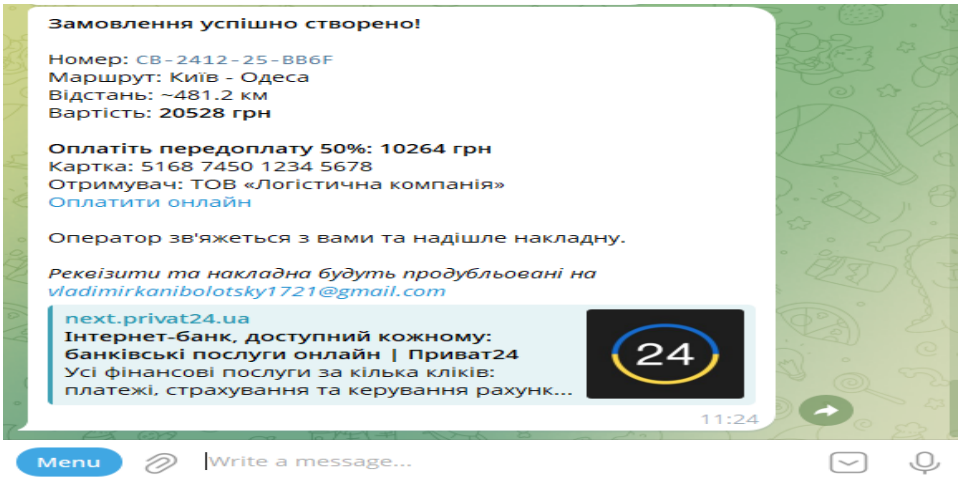

Рисунок 4.22 – Підтвердження створення замовлення

Паралельно з повідомленням у Telegram клієнт отримує лист на електронну пошту з детальною інформацією про замовлення. Лист містить номер замовлення, статус, маршрут, характеристики вантажу, орієнтовний час доставки, вартість та реквізити для оплати передоплати з посиланням на онлайн-оплату через Приват24 (рисунок 4.23).

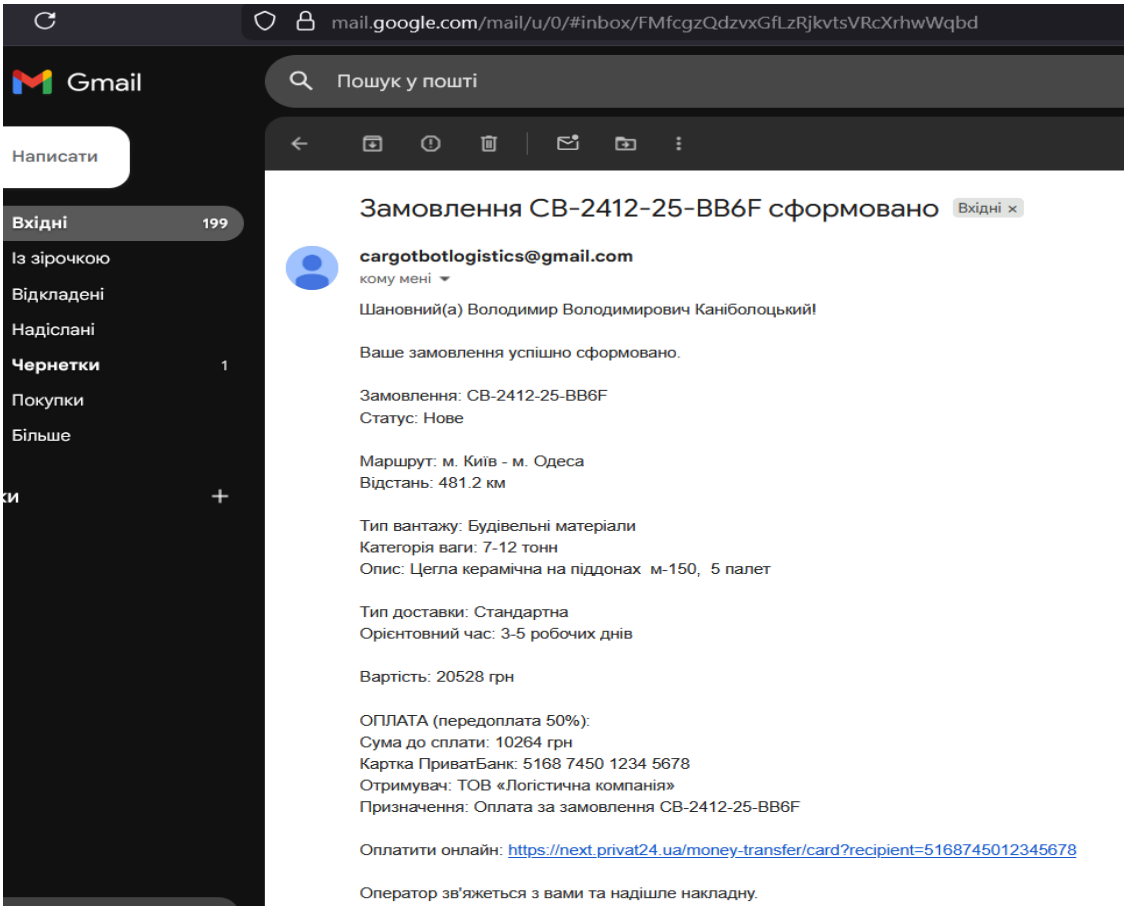

Рисунок 4.23 – Email-повідомлення клієнту про створення замовлення

У колекції operators зберігається інформація про персонал компанії з доступом до адміністративної панелі. Кожен документ містить Telegram ID, токен для авторизації, електронну пошту, ПІБ, роль (admin або operator) та статус активності. Система підтримує розмежування прав доступу між адміністраторами та операторами (рисунок 4.24).

Рисунок 4.24 – Колекція операторів у базі даних MongoDB

Всі дані системи зберігаються у базі даних MongoDB (рисунок 4.25). База даних cargo_bot містить наступні колекції: clients – інформація про клієнтів, drivers – дані водіїв, vehicles – транспортні засоби, operators – персонал з доступом до адміністративної панелі, orders – замовлення на перевезення, verification_codes – коди підтвердження для авторизації. Для зберігання файлів ТТН-накладних використовуються колекції GridFS: fs.files містить метадані файлів, fs.chunks – бінарні дані файлів, розбиті на частини. Для демонстрації структури бази даних використано інтерфейс MongoDB Compass.

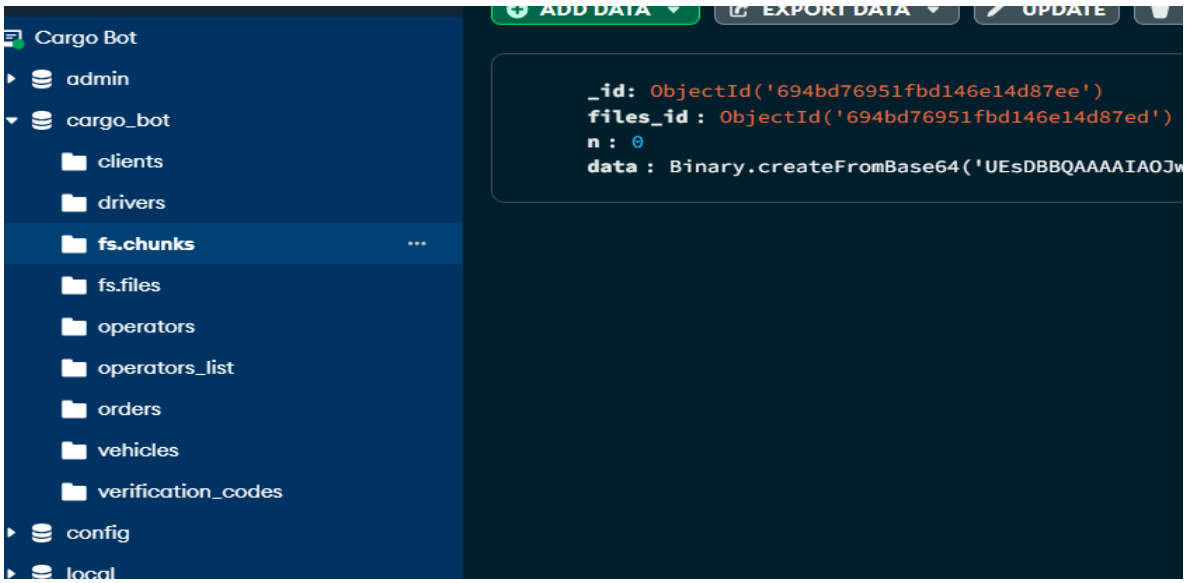

Рисунок 4.25 – Структура бази даних у MongoDB Compass

4.4.2 Функції AI-консультанта

Система містить модуль інтелектуального консультанта на базі OpenAI API. При виборі функції «AI Консультант» клієнт отримує можливість задавати питання щодо вартості доставки, типів вантажів, тарифів та умов перевезення. Консультант надає відповіді у режимі реального часу, що дозволяє клієнтам швидко отримати необхідну інформацію без очікування на відповідь оператора (рисунок 4.26).

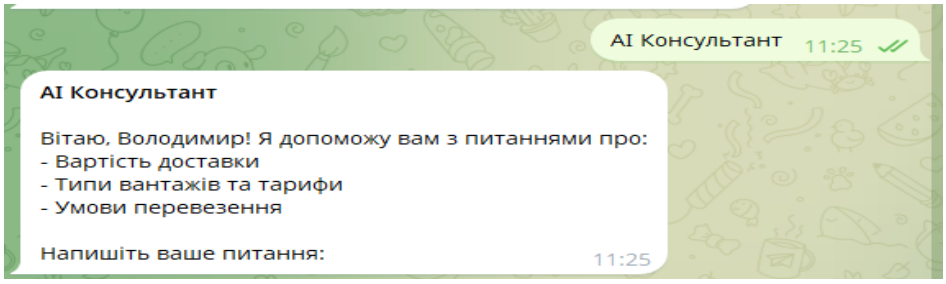

Рисунок 4.26 – Інтерфейс AI-консультанта

AI-консультант здатний виконувати попередній розрахунок вартості перевезення на основі заданих тарифів компанії. У відповідь на запит клієнта система надає детальну калькуляцію з урахуванням базової ставки, вартості за кілометр, вартості за вагу та коефіцієнта типу вантажу. Це дозволяє клієнту отримати орієнтовну вартість до оформлення замовлення (рисунок 4.27).

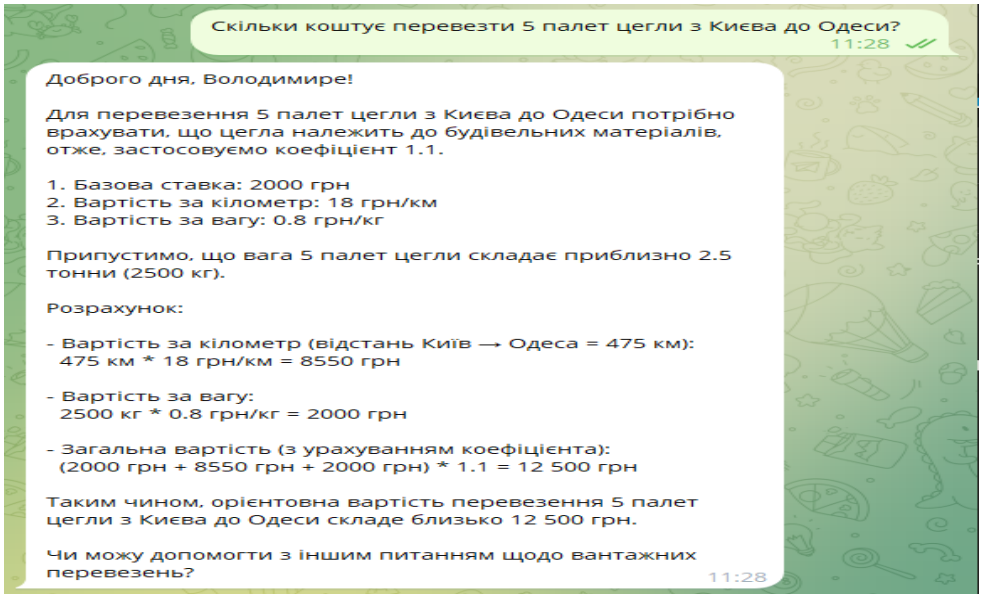

Рисунок 4.27 – Приклад роботи AI-консультанта з розрахунком вартості

AI-консультант також інформує клієнтів про обмеження послуг компанії. У випадку запиту на перевезення вантажів, які не входять до переліку послуг (наприклад, харчові продукти, що потребують рефрижераторів), система коректно відмовляє та рекомендує звернутися до спеціалізованих компаній (рисунок 4.28).

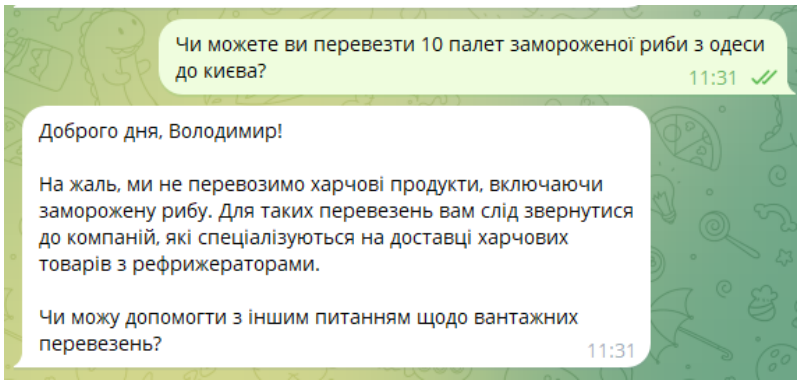

Рисунок 4.28 – Приклад відмови AI-консультанта у нетиповому запиті

Клієнт має можливість переглянути історію своїх замовлень через функцію «Мої замовлення». Система відображає список замовлень із зазначенням номера, маршруту, вартості та поточного статусу. Для отримання детальної інформації клієнт може ввести номер конкретного замовлення (рисунок 4.29).

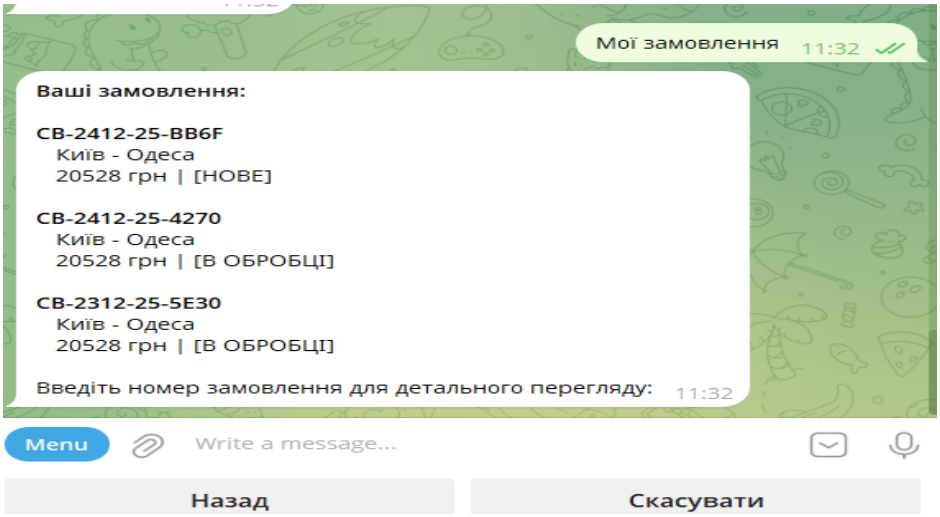

Рисунок 4.29 – Перегляд списку замовлень клієнта

Клієнт може ознайомитися з тарифами компанії через відповідний пункт меню. Система відображає базову вартість перевезення, ставки за кілометр та вагу, коефіцієнти для різних типів вантажу, перелік послуг, що входять у вартість, приклади розрахунку та умови оплати. Це дозволяє клієнту самостійно оцінити орієнтовну вартість перевезення (рисунок 4.30).

Рис. 4.30 – Інформація про тарифи компанії

При введенні номера замовлення клієнт отримує детальну інформацію про нього. Відображається поточний статус, маршрут з посиланням на карту, характеристики вантажу та вартість. Клієнт має можливість переглянути історію статусів, написати повідомлення оператору, скасувати замовлення або відкрити маршрут на карті (рисунок 4.31).

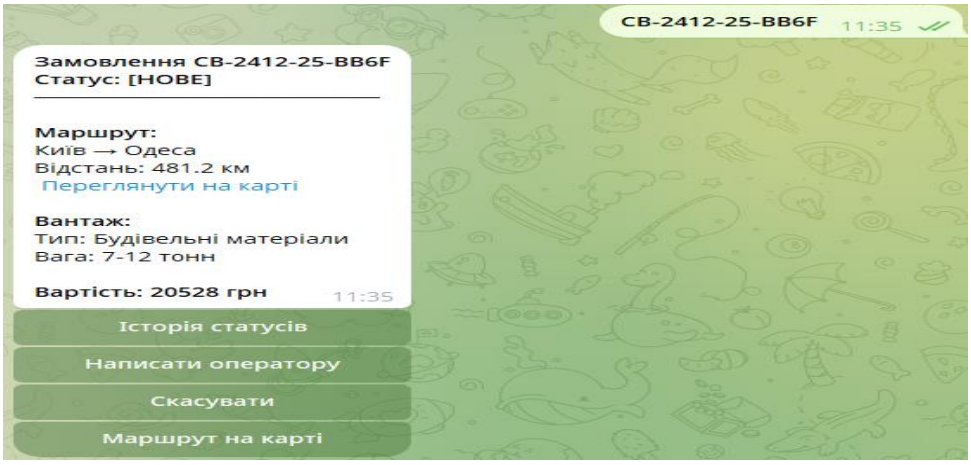

Рисунок 4.31 – Детальний перегляд замовлення клієнтом

При натисканні кнопки «Маршрут на карті» клієнт отримує інформацію про адреси відправлення та доставки з розрахованою відстанню. Посилання «Відкрити на карті» дозволяє переглянути маршрут у картографічному сервісі (рисунок 4.32).

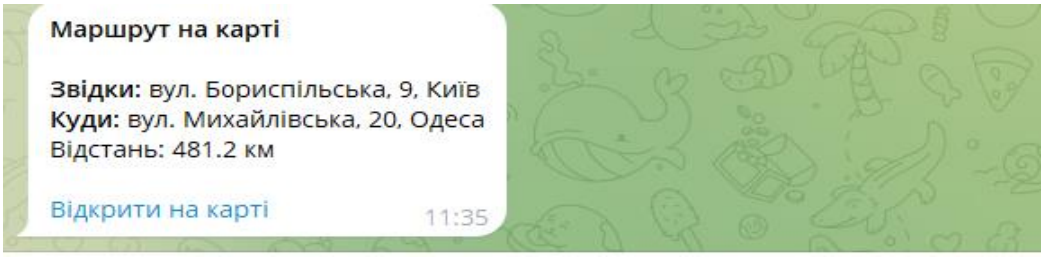

Рисунок 4.32 – Перегляд маршруту замовлення

При переході за посиланням відкривається картографічний сервіс OpenStreetMap з побудованим маршрутом. Клієнт може переглянути детальний маршрут з покроковими інструкціями, відстанню та орієнтовним часом у дорозі. Маршрут розраховується через інтеграцію з OpenRouteService API (рисунок 4.33).

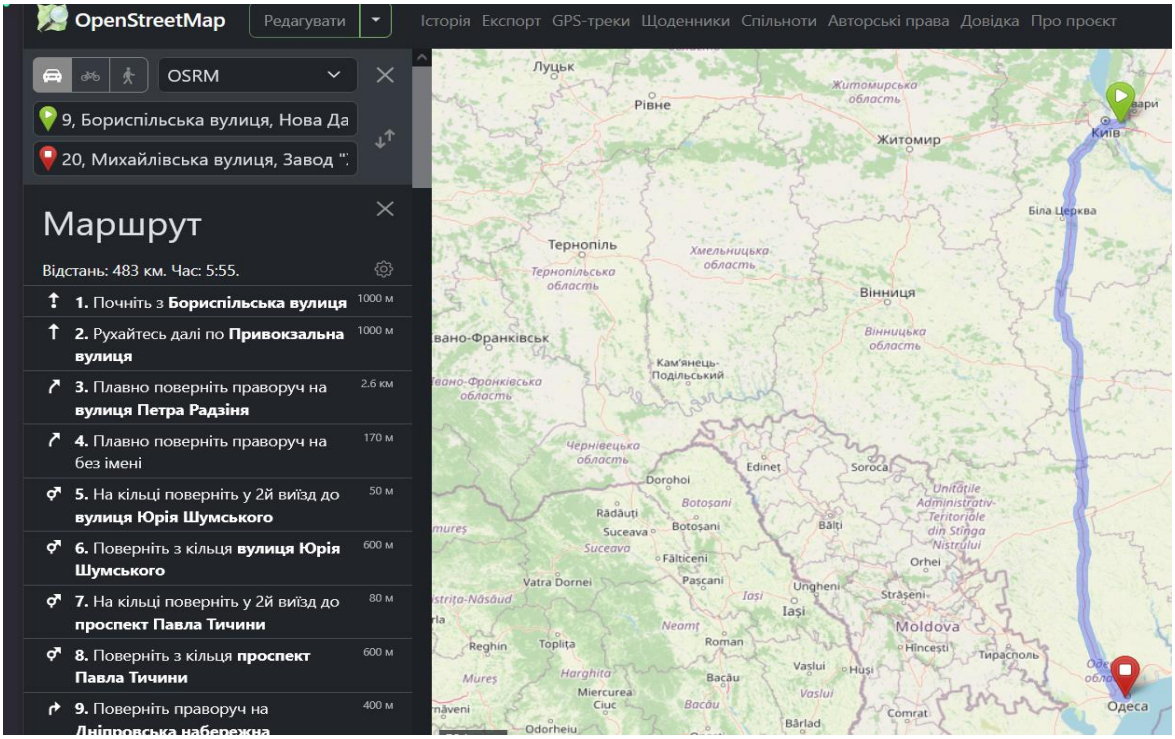

Рисунок 4.33 – Візуалізація маршруту в OpenStreetMap

Через пункт меню «Контакти» клієнт може отримати повну контактну інформацію компанії. Відображаються номери гарячої лінії, офісу та диспетчерської служби з графіком роботи, електронні адреси для різних типів звернень, інформація про компанію та адреси офісів (рисунок 4.34).

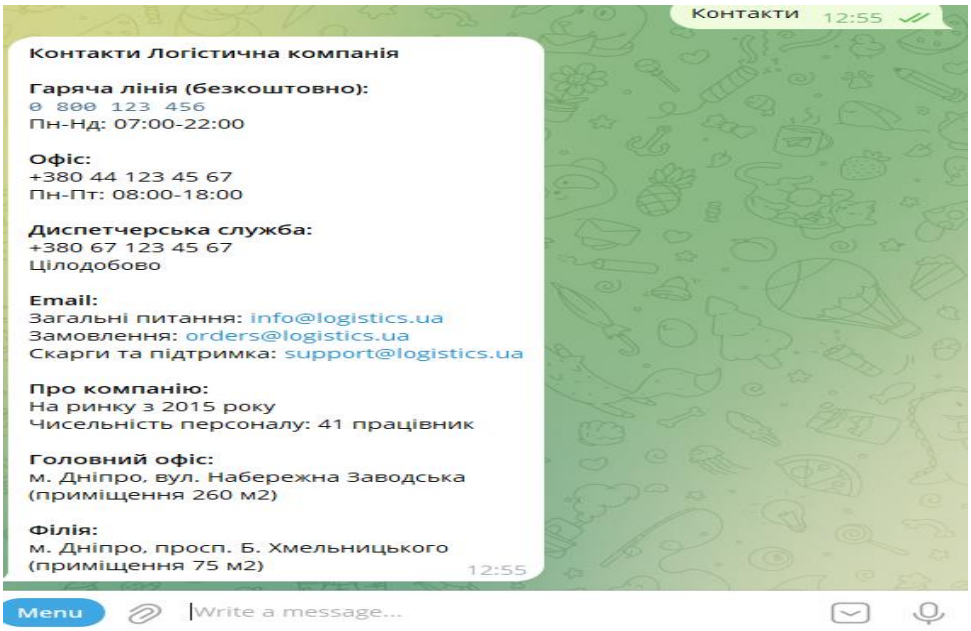

Рисунок 4.35 – Контактна інформація компанії

4.4.3 Функції панелі оператора

Тепер розглянемо гілку для оператора. Доступ до адміністративних функцій обмежується за допомогою білого списку операторів, що зберігається у колекції operators бази даних (рисунок 4.24 з колекцією operators). При виборі «Я оператор» система пропонує обрати метод авторизації. Після успішної перевірки за Telegram ID оператор отримує доступ до панелі керування з можливістю перегляду нових та всіх замовлень, а також статистики (рисунок 4.36).

При спробі авторизації з Telegram-акаунту, ID якого відсутній у базі даних операторів, система відмовляє у доступі. Користувачу відображається його Telegram ID та повідомлення про те, що цей ідентифікатор не знаходиться в списку операторів. Система пропонує спробувати інший метод авторизації: за токеном або за Email (рисунок 4.36).

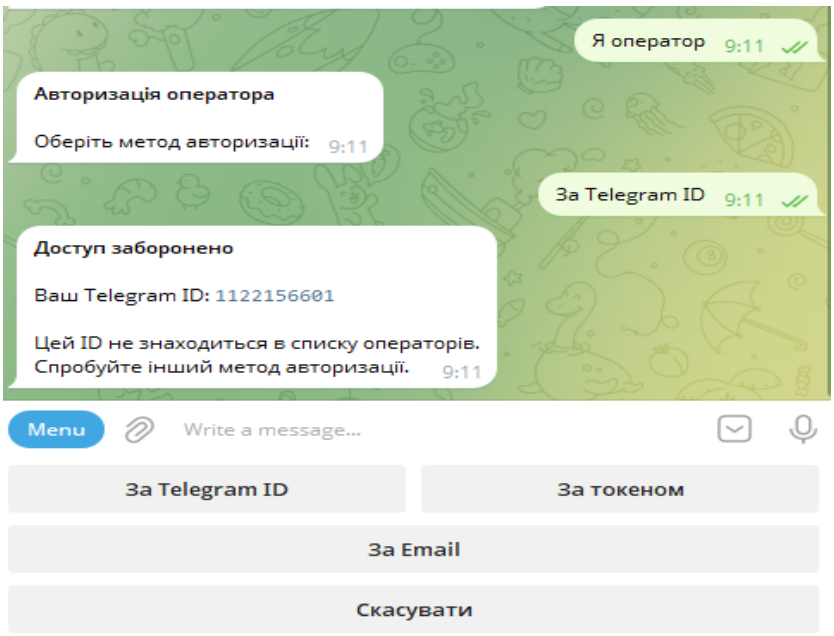

Рисунок 4.36 – Відмова в авторизації для незареєстрованого користувача

При авторизації з Telegram-акаунту, ID якого присутній у базі даних, система надає доступ до панелі оператора (рисунок 4.37). Оператору відображається привітання з його ім'ям та метод авторизації. Панель оператора містить функції перегляду нових та всіх замовлень, статистики та можливість виходу з панелі.

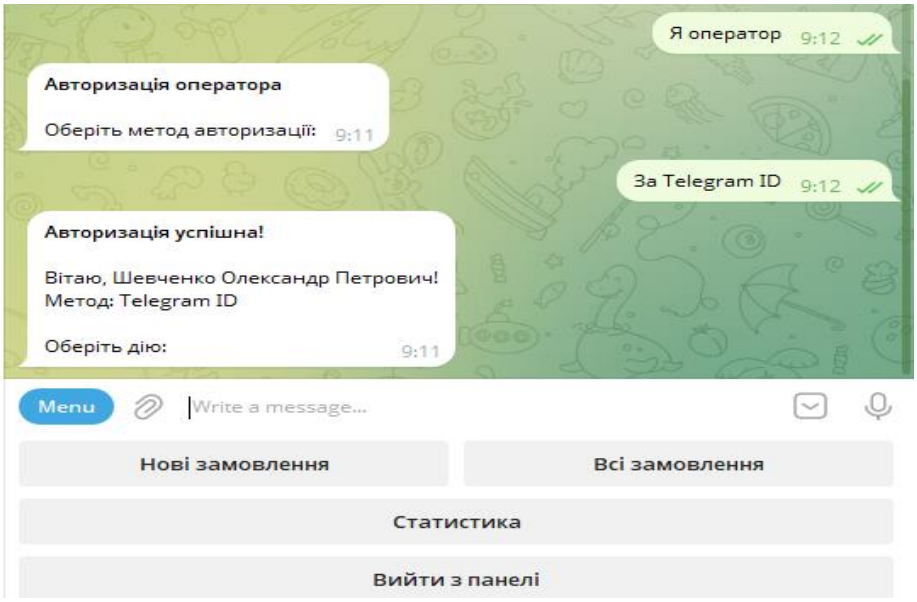

Рисунок 4.37 – Успішна авторизація оператора

Переглянемо нові замовлення і бачимо список нових замовлень. При виборі функції «Нові замовлення» оператор отримує список замовлень зі статусом «Нове». Відображається кількість нових замовлень, їх номери, маршрути, вартість та дата створення. Для детального перегляду оператор вводить номер конкретного замовлення (риснок 4.38).

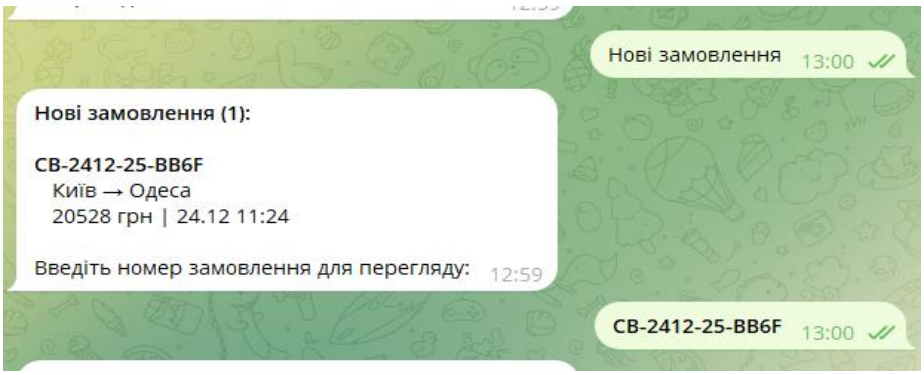

Рисунок 4.38 – Перегляд нових замовлень оператором

При натисканні кнопки «Взяти в роботу» оператор переходить до процесу призначення виконавців. На першому кроці обирається водій зі списку доступних. Після вибору водія система пропонує обрати транспортний засіб із зазначенням марки та вантажопідйомності (рисунок 4.39).

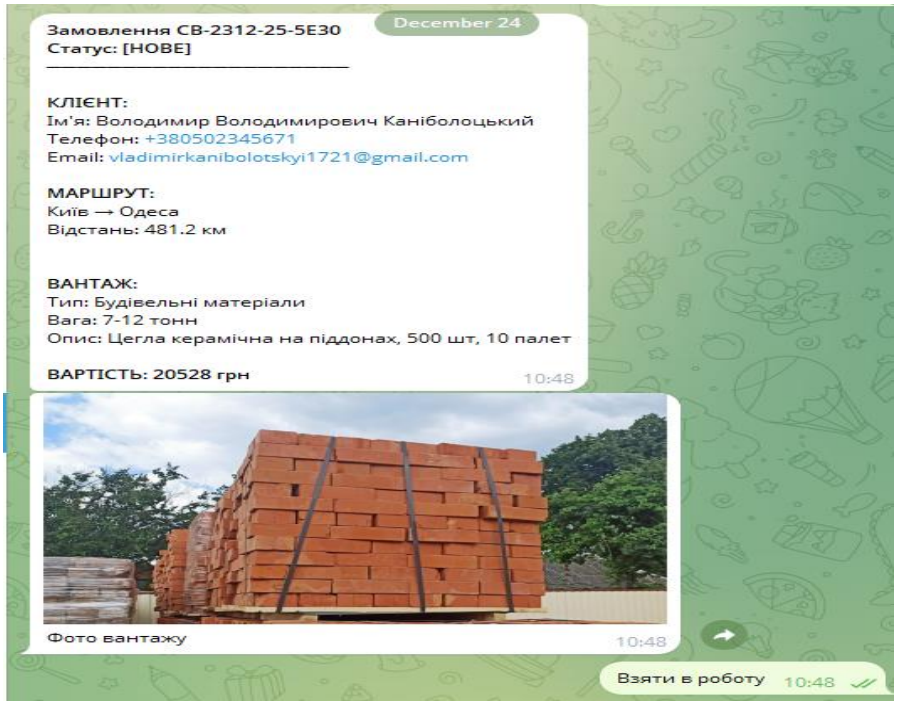

Рисунок 4.39 – Перегляд замовлення з фотографією вантажу

Після вибору водія оператор переходить до вибору транспортного засобу. Система відображає список доступних автомобілів із зазначенням марки, моделі та вантажопідйомності, що дозволяє підібрати оптимальний транспорт відповідно до характеристик вантажу (рисунок 4.40).

Рисунок 4.40 – Вибір транспортного засобу для замовлення

Після вибору водія та транспорту система підтверджує взяття замовлення в роботу. Оператору відображається повна інформація про призначених виконавців: ПІБ водія, його телефон, марка та державний номер автомобіля. Клієнт автоматично отримує сповіщення, а оператору пропонується сформувати ТТН-накладну (рисунок 4.41).

Рисунок 4.41 – Завершення процесу призначення виконавців

Наступний крок: клієнт автоматично отримує сповіщення у Telegram. Повідомлення містить номер замовлення, новий статус «В обробці», інформацію про призначеного водія та транспортний засіб. Клієнту повідомляється про очікування ТТН-накладної (рисунок 4.42).

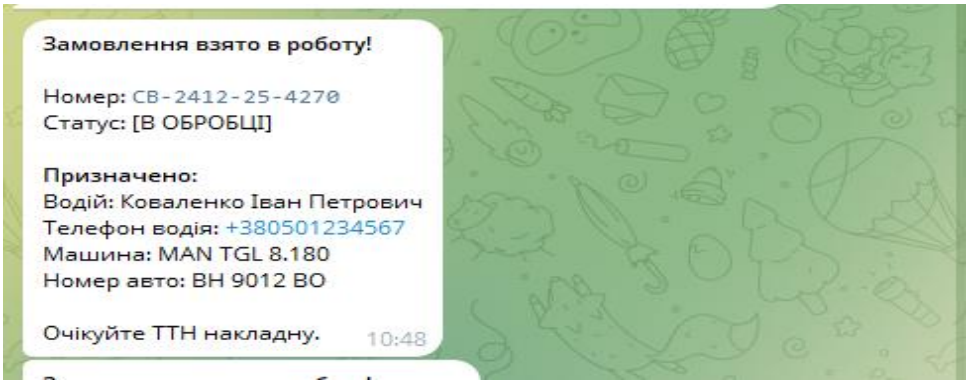

Рисунок 4.42 – Сповіщення для клієнта про прийняття замовлення в роботу

Після формування ТТН оператором клієнт автоматично отримує накладну у Telegram (рисунок 4.43). Повідомлення містить файл ТТН у форматі .docx, номер замовлення, маршрут та інформацію про призначеного водія і транспорт. Головне меню бота дозволяє клієнту створити нове замовлення, переглянути існуючі, відстежити статус, звернутися до AI-консультанта або зв'язатися з оператором.

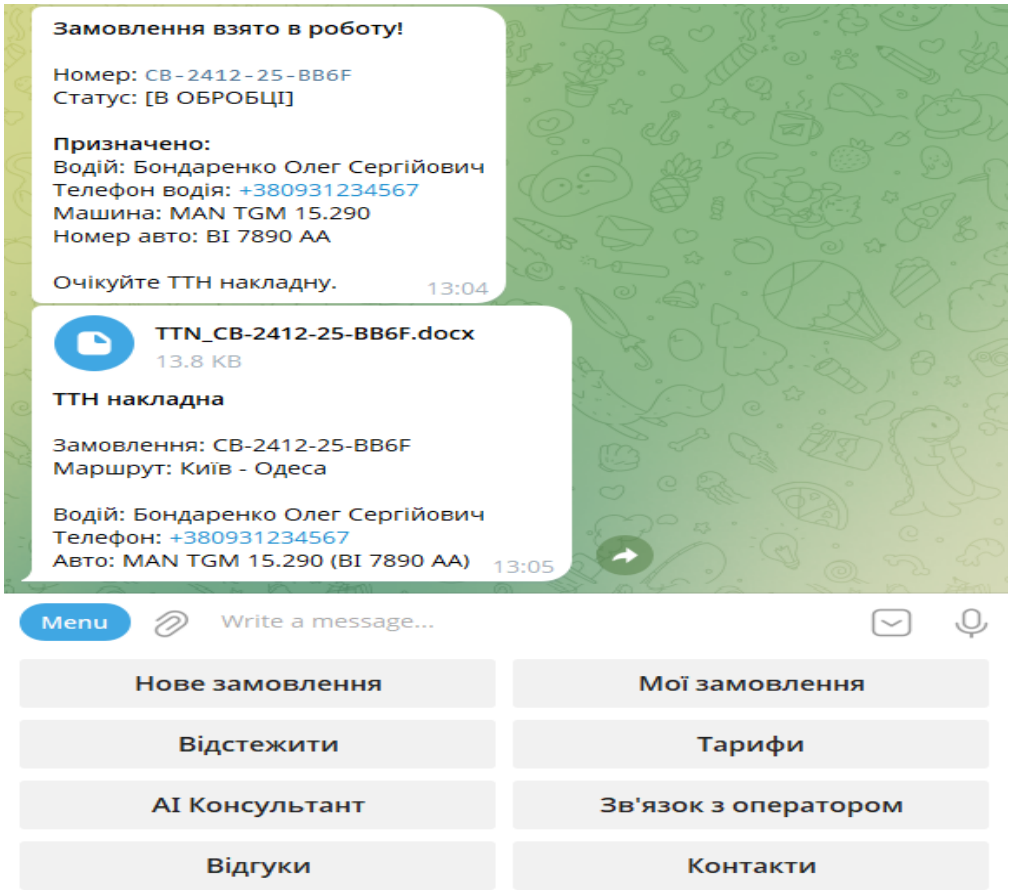

Рисунок 4.43 – Отримання ТТН-накладної клієнтом

Система сповіщень реалізована через два канали зв'язку: Telegram та електронну пошту. Для відправки email-повідомлень використовується асинхронна бібліотека aiosmtplib, яка забезпечує неблокуючу відправку листів через SMTP-протокол сервера Gmail. Клієнт отримує сповіщення про створення замовлення, зміну статусу та ТТН-накладну як у месенджері, так і на електронну пошту, що гарантує своєчасне інформування незалежно від обраного каналу комунікації.

Паралельно зі сповіщенням у Telegram клієнт отримує лист на електронну пошту про зміну статусу замовлення (рисунок 4.44). Повідомлення містить номер

замовлення, маршрут та новий статус. Таким чином, клієнт завжди поінформований про стан свого замовлення через два канали зв'язку.

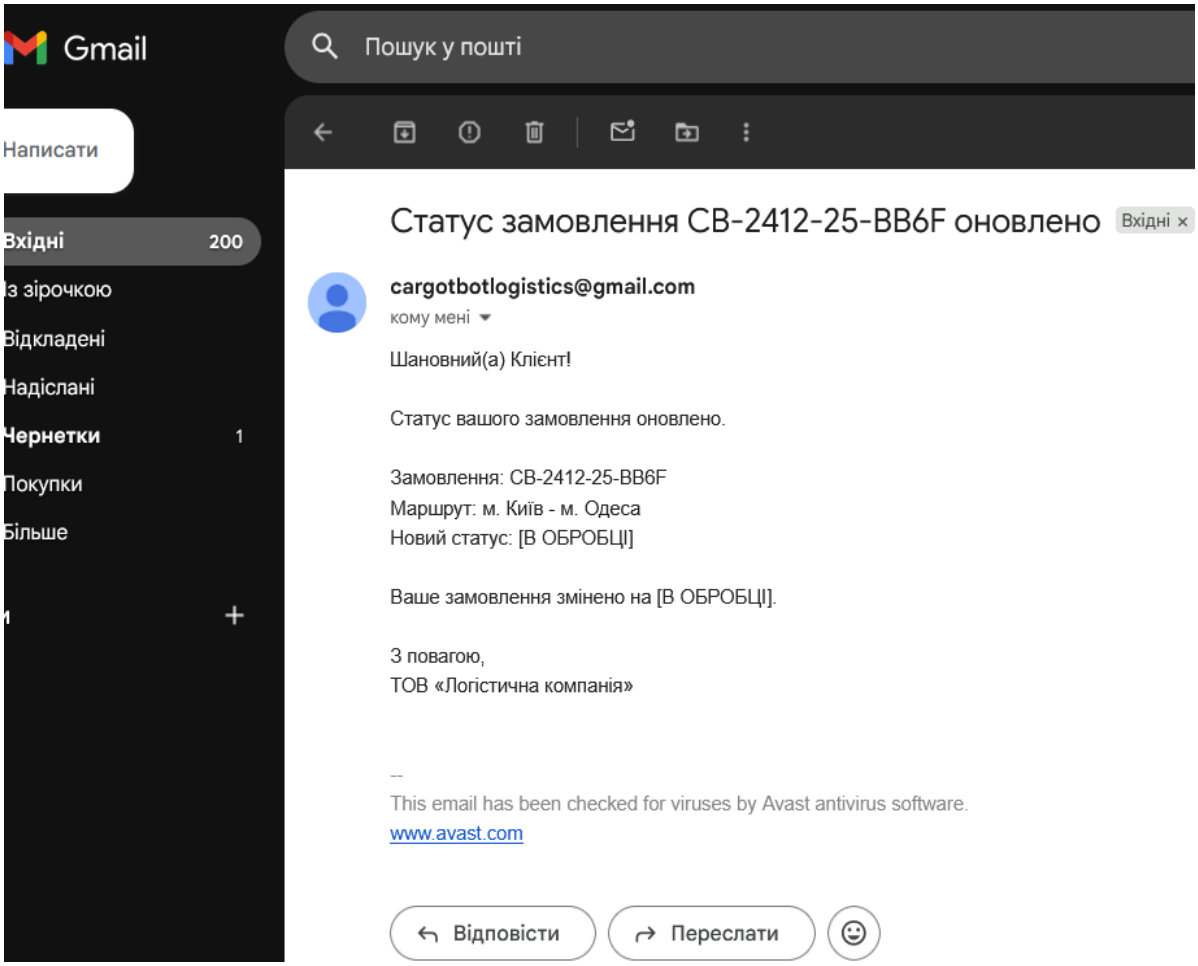

Рисунок 4.44 – Email-сповіщення клієнта про зміну статусу замовлення

Клієнт також отримує ТТН-накладну на електронну пошту (рисунок 4.45). Лист містить інформацію про прийняття замовлення в обробку, маршрут, вартість та вкладений файл накладної у форматі .docx. Це забезпечує клієнту зручний доступ до документів незалежно від наявності Telegram.

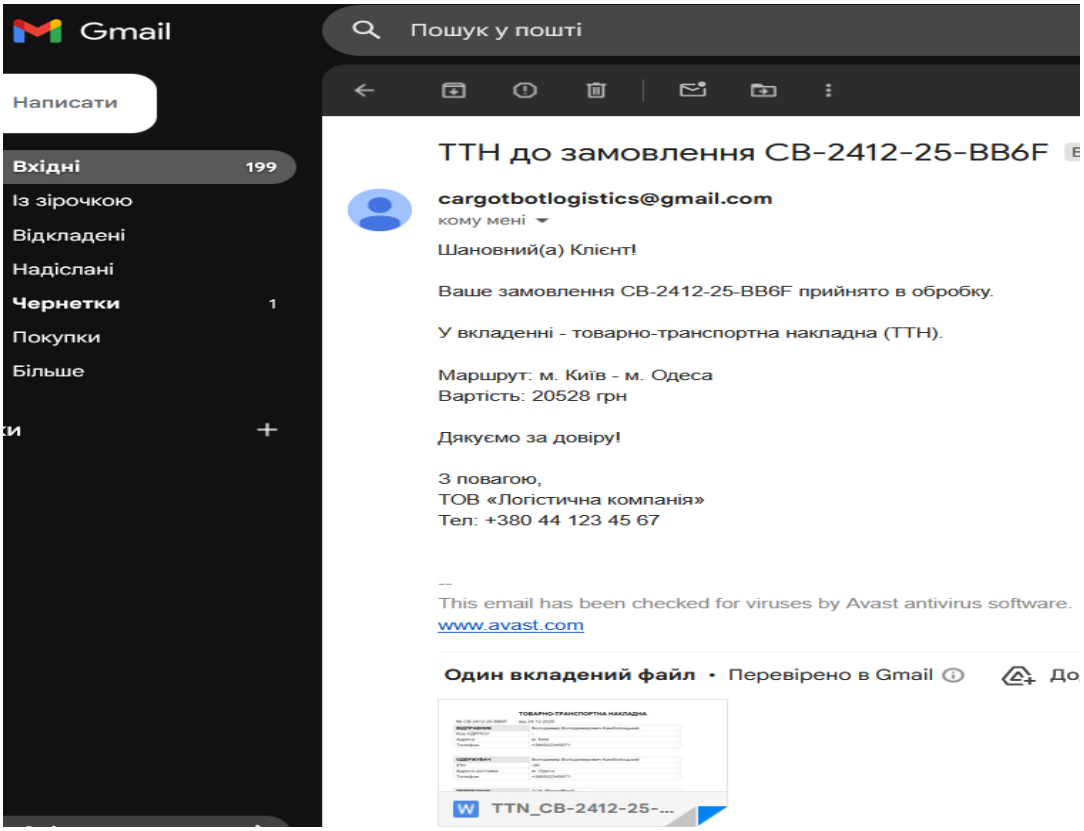

Рисунок 4.45– Email-повідомлення з ТТН-накладною

На рисунку 4.46 представлено структуру збереження товарно-транспортної накладної (ТТН) у базі даних MongoDB з використанням технології GridFS. Колекція fs.files містить метадані файлу: назву документа (filename), номер замовлення (order_id), ім'я клієнта (client_name), дату створення (created_at), тип файлу (contentType) та розмір (length). Бінарні дані файлу зберігаються в колекції fs.chunks. Такий підхід дозволяє зберігати документи будь-якого розміру безпосередньо в базі даних, забезпечуючи цілісність даних та зв'язок між замовленням і його ТТН накладною (рисунок 4.46).

Рисунок 4.46 – Збереження ТТН накладної в MongoDB GridFS

Таким чином було розроблено та обґрунтовано функціональні та технічні характеристики програмного забезпечення чатботу для логістичної компанії.

В ході розробки було реалізовано:

– систему оформлення замовлень з 10-кроковим процесом створення заявки на перевезення вантажу;

– панель оператора з функціями управління замовленнями, призначення водіїв та транспортних засобів;

– автоматичну генерацію товарно-транспортних накладних (ТТН) зі збереженням у MongoDB GridFS;

– AI-консультант на базі OpenAI GPT для автоматизованої підтримки клієнтів;

– систему сповіщень через Telegram та Email;

– інтеграцію з OpenRouteService API для розрахунку маршрутів та вартості доставки.

Спираючись на отримані в ході дослідження дані, було написано програмний код чатботу мовою Python з використанням фреймворку aiogram 3.13 та бази даних MongoDB. Архітектура системи забезпечує взаємодію між клієнтами, операторами та базою даних у реальному часі.

Всі функції програмного забезпечення було експериментально протестовано та підтверджено їх працездатність, що дозволяє впровадити розроблену систему для автоматизації зв’язку з клієнтами логістичної компанії.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Мета і завдання експериментального дослідження

Мета експерименту – дослідити продуктивність та навантаження на серверну інфраструктуру логістичної компанії під час взаємодії чатботу з клієнтами та обробки їх замовлень.

Завданням експерименту є дослідження навантаження на сервер від роботи чатботу логістичної компанії cargo_bot, а саме аналіз продуктивності бази даних MongoDB та швидкодії операцій при одночасній роботі багатьох користувачів із системою.

Основні цілі експерименту:

– визначення максимальної пропускної здатності системи;

– аналіз часу відповіді на різні типи запитів до бази даних;

–перевірка стабільності роботи системи при тривалому навантаженні;

– моніторинг використання ресурсів сервера та бази даних.

5.2 Методика проведення експериментального дослідження

Для оцінки продуктивності розробленої системи було обрано метод автоматизованого навантажувального тестування. На відміну від ручного тестування, яке передбачає залучення реальних користувачів для взаємодії з системою, автоматизоване тестування дозволяє:

– точно контролювати кількість одночасних користувачів та інтенсивність навантаження;

–відтворювати ідентичні сценарії тестування для порівняння результатів;

– проводити тривалі тести без втоми та людського фактору;

– збирати детальну статистику по кожній операції в реальному часі;

– симулювати пікові навантаження, які складно відтворити вручну.

Автоматизоване навантажувальне тестування є стандартом у сучасній розробці програмного забезпечення, оскільки дозволяє виявити проблеми продуктивності ще до впровадження системи у виробниче середовище.

Серед популярних інструментів для автоматизованого навантажувального тестування можна виділити:

– Apache JMeter – потужний інструмент з графічним інтерфейсом, написаний мовою Java, підтримує широкий спектр протоколів, але має високий поріг входження та споживає багато ресурсів;

– Gatling – інструмент на базі Scala з акцентом на високу продуктивність, використовує DSL для написання сценаріїв, що може бути складним для початківців;

– k6 – сучасний інструмент від Grafana Labs, написаний мовою Go, використовує JavaScript для сценаріїв, орієнтований на DevOps та CI/CD;

– Locust – інструмент з відкритим кодом, написаний мовою Python, дозволяє описувати поведінку користувачів звичайним Python-кодом, має зручний веб-інтерфейс для моніторингу.

Для проведення експерименту було обрано інструмент Locust з наступних причин:

– написаний мовою Python, що відповідає технологічному стеку проекту та спрощує інтеграцію;

–простий та зрозумілий синтаксис для опису сценаріїв тестування;

– вбудований веб-інтерфейс для візуалізації результатів у реальному часі;

– низьке споживання ресурсів порівняно з JMeter;

– можливість легко масштабувати тестування на декілька машин;

–активна спільнота та якісна документація.

Для моніторингу стану бази даних MongoDB під час тестування було розгорнуто стек моніторингу, що включає:

– MongoDB Exporter – утиліта для збору розширеної інформації про функціонування MongoDB та експорту цих даних у форматі, сумісному з Prometheus. Exporter збирає метрики про з'єднання, операції, використання пам'яті, розмір баз даних та інші показники;

– Prometheus – система моніторингу та збору метрик з відкритим кодом, що використовується для отримання даних від MongoDB Exporter та їх зберігання у

вигляді часових рядів. Prometheus періодично опитує exporter та накопичує історію метрик;

– Grafana – інструмент для візуалізації даних метрик у вигляді інтерактивних графіків та dashboard-ів, що інтегрується з Prometheus та дозволяє створювати наочні звіти про стан системи в реальному часі.

MongoDB Exporter надає широкий набір метрик для моніторингу. У рамках експерименту було обрано наступні ключові метрики:

– mongodb_up – статус доступності бази даних (1 – працює, 0 – недоступна);

– mongodb_connections – інформація про з'єднання з базою даних (активні, доступні, поточні);

– mongodb_dbstats_dataSize – розмір даних у кожній базі даних у байтах;

– mongodb_memory – використання пам'яті (резидентна та віртуальна).

Ці метрики дозволяють оцінити стабільність роботи бази даних, ефективність використання ресурсів та виявити потенційні проблеми під час навантаження.

Тестовий сценарій (locustfile.py) було розроблено для симуляції типових операцій користувачів Telegram-бота:

– створення замовлень на перевезення вантажу (create_order);

– отримання списку замовлень клієнта (get_my_orders);

– відстеження статусу замовлення (track_order);

– перегляд нових замовлень оператором (operator_new_orders);

– оновлення статусу замовлення (update_status);

– реєстрація нових клієнтів (register_client);

– отримання статистики замовлень (get_statistics);

– отримання списку водіїв (get_drivers);

– отримання списку транспортних засобів (get_vehicles).

Кожен віртуальний користувач виконував ці операції у випадковому порядку з інтервалом 1-3 секунди між запитами, що імітує реальну поведінку користувачів системи.

5.3 Вимоги до проведення експерименту

Логістична компанія обробляє в середньому 50-100 замовлень на добу. Враховуючи пікові навантаження в робочі години, очікувана кількість одночасних користувачів системи може досягати 10-20 осіб (клієнти, оператори, водії).

Для проведення експерименту використовувався локальний сервер з наступними характеристиками:

– операційна система: Windows 10/11;

– процесор: Intel Core i5-8300H (4 ядра, 2.3 ГГц);

– оперативна пам'ять: 16 ГБ;

– база даних: MongoDB 8.0;

– мова програмування: Python 3.13.

Параметри навантажувального тестування:

– кількість віртуальних користувачів: 15;

– швидкість створення користувачів: 5 користувачів/секунду;

– тривалість тесту: 2 години 40 хвилин;

– інтервал між запитами: 1-3 секунди.

Кількість віртуальних користувачів (15) було обрано з урахуванням очікуваного пікового навантаження на систему, що дозволяє оцінити поведінку системи в умовах максимального навантаження.

5.4 Результати експерименту

В ході експерименту було проведено тривале навантажувальне тестування системи за допомогою інструменту Locust. Тест тривав 2 години 40 хвилин з 15 віртуальними користувачами, які одночасно працювали із системою.

Представлено графік кількості запитів на секунду (RPS) протягом всього періоду тестування (рисунок 5.1).

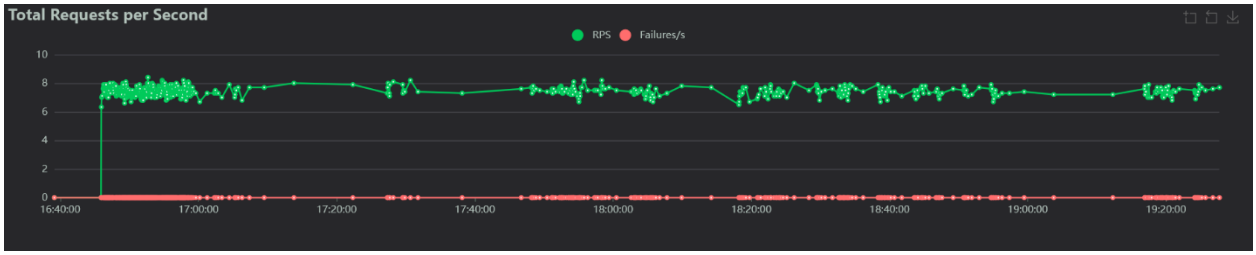

Рисунок 5.1 – Графік кількості запитів на секунду (RPS)

Зелена лінія на графіку відображає кількість успішно оброблених запитів, яка стабільно тримається на рівні 7–8 запитів на секунду протягом всього тестування. Червона лінія (Failures/s) знаходиться на нульовому рівні, що підтверджує відсутність помилок при обробці запитів. Рівномірність графіку свідчить про стабільну роботу системи без деградації продуктивності з часом.

Наступним спостерігаємо графік часу відповіді системи в мілісекундах (рисунок 5.2).

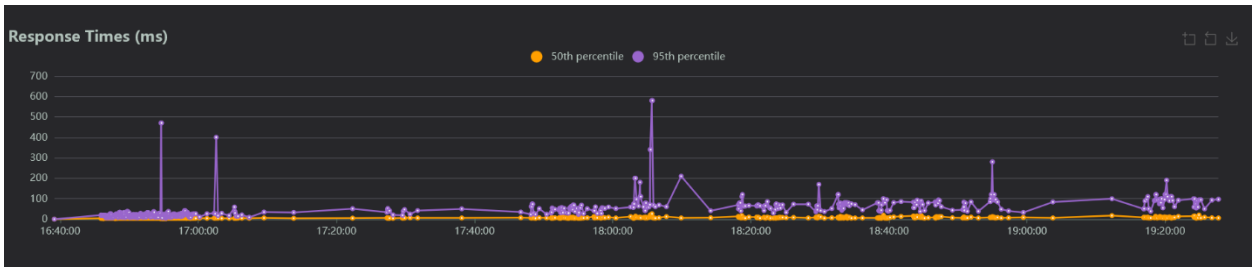

Рисунок 5.2 – Графік часу відповіді системи

Жовта лінія показує медіану (50-й перцентиль), яка стабільно знаходиться на рівні 10-20 мс. Фіолетова лінія відображає 95-й перцентиль, який переважно тримається в межах 50–100 мс з окремими піками до 400–600 мс. Ці піки пояснюються періодичними операціями збору статистики (get_statistics), які є

найбільш ресурсомісткими. Загалом час відповіді значно нижчий за допустимий поріг 100 мс для інтерактивних систем.

Представлено графік кількості активних віртуальних користувачів протягом тестування (рисунок 5.3).

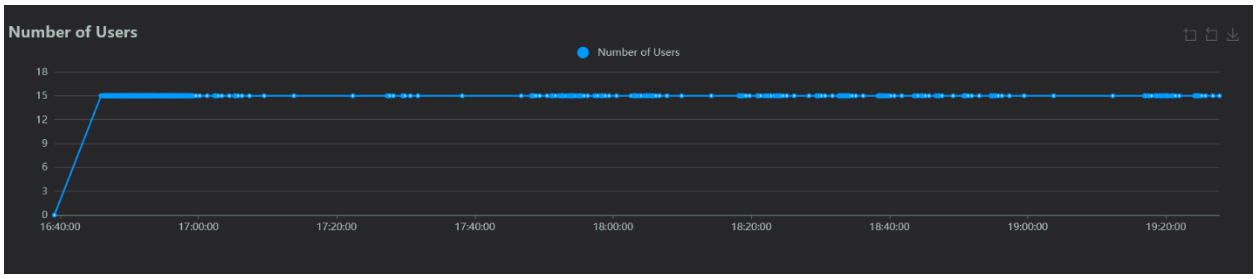

Рисунок 5.3 – Графік кількості віртуальних користувачів

На початку тесту спостерігається швидке зростання від 0 до 15 користувачів зі швидкістю 5 користувачів на секунду. Після досягнення цільового значення кількість користувачів залишається стабільною на рівні 15 протягом всього тестування, що забезпечує постійне навантаження на систему.

Детальна статистика виконання операцій за результатами тестування відповідно до таблиці 5.1.

Таблиця 5.1– Статистика виконання операцій за результатами тестування

Операція	**Запитів**	**Медіана (мс)**	**Середній (мс)**	**95% (мс)**	**Макс. (мс)**
create_order	19 184	3	4,79	6	1 042
get_my_orders	11 271	16	18,22	35	964
track_order	11 465	8	11,2	25	716
operator_new_orders	7 554	28	29,75	55	1 269
update_status	7 354	6	7,53	13	575
get_statistics	3 854	69	70,8	110	1 266
register_client	3 849	3	4,45	7	487
get_drivers	3 680	3	4,81	7	831
get_vehicles	3 803	3	4,26	7	469
Всього	**72 014**	**6**	**14,3**	**51**	**1 269**

Для моніторингу стану бази даних MongoDB було використано Grafana з підключенням до Prometheus та MongoDB Exporter. На рисунках 5.4-5.9 представлено dashboard з основними метриками роботи бази даних.

Показано статус роботи MongoDB протягом тестування (рисунок 5.4).

Рисунок 5.4 – Статус MongoDB в Grafana

Графік демонструє стабільне значення метрики mongodb_up, яке дорівнює 1 протягом всього періоду тестування, що підтверджує безперебійну роботу бази даних без жодних збоїв чи перезавантажень.

Представлено графік з'єднань з базою даних MongoDB (рисунок 5.5).

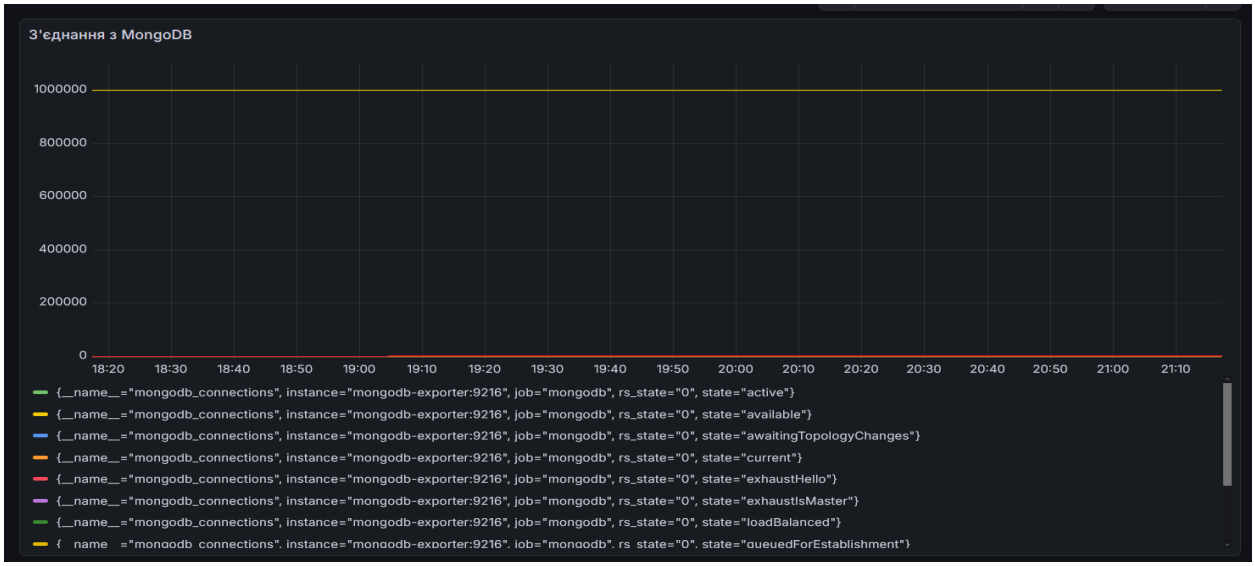

Рисунок 5.5 – З'єднання з MongoDB

Графік показує різні типи з'єднань з базою даних. Жовта лінія на рівні близько 1 000 000 відображає кількість доступних з'єднань, що свідчить про значний запас для масштабування системи. Активні з'єднання знаходяться на значно нижчому рівні.

Відображено динаміку зміни розміру бази даних cargo_bot під час тестування (рисунок 5.6).

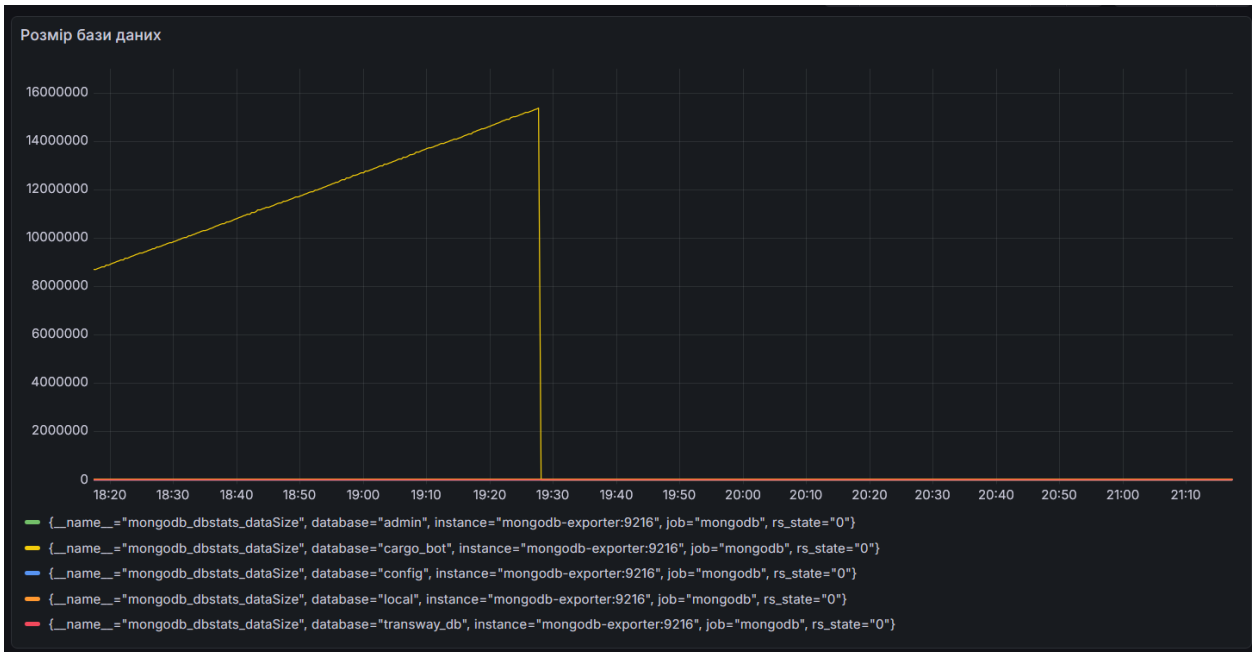

Рисунок 5.6 – Розмір бази даних cargo_bot

Жовта лінія на графіку показує зростання розміру бази даних cargo_bot з приблизно 8 МБ на початку тестування до близько 15 МБ в пік навантаження. Це зростання пояснюється активним створенням нових замовлень (понад 19 000 записів) під час тестування. Різке падіння на графіку після 19:30 пов'язане із зупинкою системи моніторингу після завершення експерименту.

Нижче представлено використання пам'яті базою даних MongoDB (рисунок 5.7).

Рисунок 5.7 – Використання пам'яті MongoDB

Графік демонструє два типи пам'яті: віртуальну (синя лінія на рівні близько 4 500 МБ) та резидентну (зелена лінія на нижньому рівні). Стабільність показників свідчить про ефективне управління пам'яттю без аномальних стрибків чи витоків.

Детальний графік використання резидентної пам'яті MongoDB (рисунок 5.8).

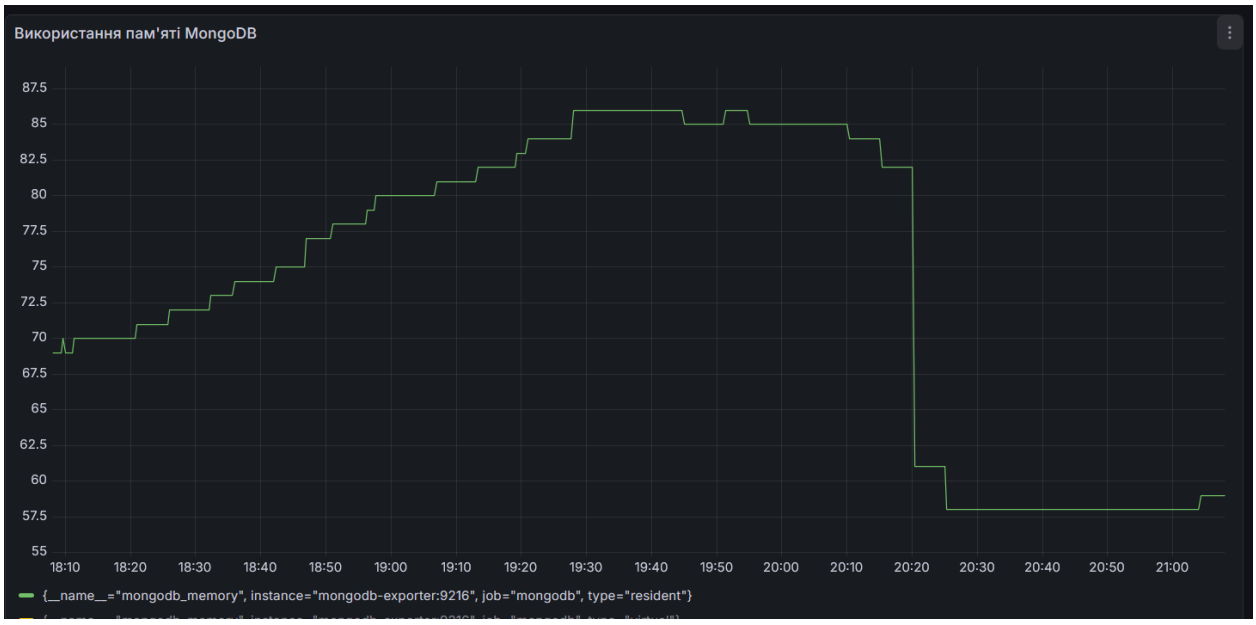

Рисунок 5.8 – Використання резидентної пам'яті MongoDB

Графік показує поступове зростання резидентної пам'яті з приблизно 70 МБ на початку тестування до 85-87 МБ в період активного навантаження. Після

завершення тестування спостерігається зниження до 58-60 МБ, що свідчить про коректне звільнення ресурсів базою даних.

Наступний – детальний графік активних з'єднань з MongoDB (рисунок 5.9).

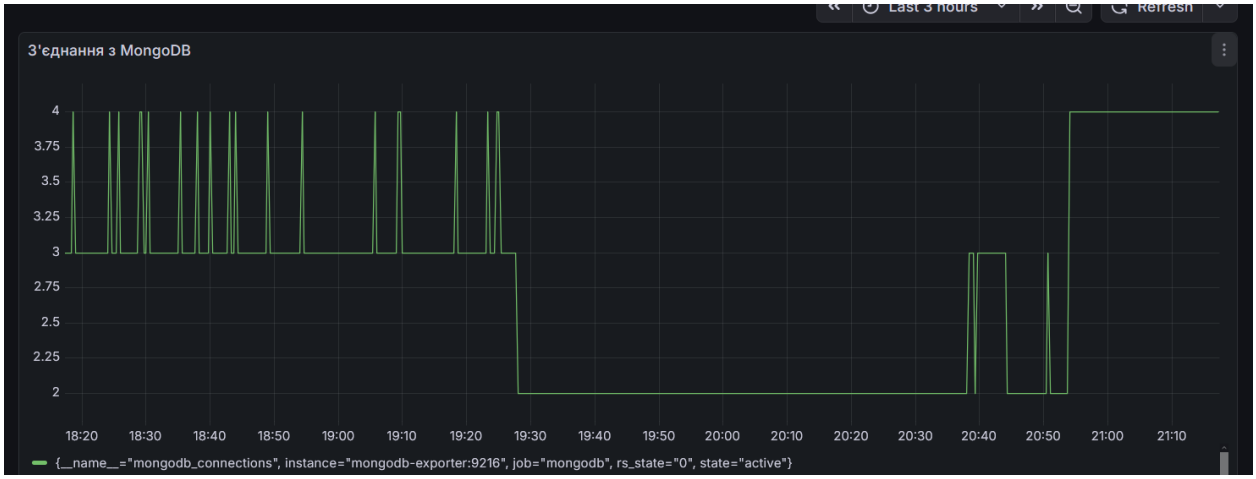

Рисунок 5.9 – Активні з'єднання з MongoDB

Графік демонструє кількість активних з'єднань під час тестування. Спостерігається коливання в межах 2-4 з'єднань під час активного навантаження, що є оптимальним показником для пулу з'єднань Python-застосунку.

5.5 Аналіз та оцінка отриманих результатів

Аналіз результатів тестування дозволяє зробити наступні висновки щодо продуктивності окремих операцій:

1. Створення замовлень (create_order) — найчастіша операція (19 184 запитів, 26,6% від загальної кількості). Медіана часу виконання становить лише 3 мс, що є відмінним показником для операції запису в базу даних. Це підтверджує ефективність індексування колекції orders.

2. Отримання замовлень (get_my_orders) — 11 271 запит із медіаною 16 мс. Дещо більший час виконання пояснюється необхідністю фільтрації та сортування результатів, проте залишається в межах прийнятних значень.

3. Відстеження замовлення (track_order) — 11 465 запитів із медіаною 8 мс. Швидкий пошук за ідентифікатором замовлення забезпечує оперативне інформування клієнтів про статус доставки.

4. Операції оператора (operator_new_orders) — 7 554 запитів із медіаною 28 мс. Більший час виконання зумовлений складнішими запитами з фільтрацією за статусом та сортуванням.

5. Оновлення статусу (update_status) — 7 354 запитів із медіаною 6 мс. Швидке оновлення документів підтверджує ефективність операцій модифікації в MongoDB.

6. Статистика (get_statistics) — найповільніша операція (медіана 69 мс), оскільки включає агрегаційні запити для підрахунку замовлень за різними критеріями. Проте цей час є прийнятним для адміністративних функцій.

7. Допоміжні операції (register_client, get_drivers, get_vehicles) — виконуються з медіаною 3 мс, що свідчить про ефективну роботу з невеликими колекціями.

Аналіз даних моніторингу Grafana:

1. Розмір бази даних cargo_bot зріс з 8 МБ до 15 МБ під час тестування, що свідчить про активне створення нових записів (понад 19 000 замовлень).

2. Використання пам'яті залишалось стабільним: віртуальна пам'ять близько 4 500 МБ, резидентна 70-87 МБ. Відсутність аномальних стрибків підтверджує ефективне управління пам'яттю MongoDB.

3. Кількість активних з'єднань коливалась в межах 2-4, що значно менше доступного ліміту (близько 1 000 000), залишаючи запас для масштабування.

4. Статус MongoDB стабільно дорівнював 1 протягом всього тестування, що підтверджує безперебійну роботу бази даних.

5.6 Висновки до експериментального розділу

За результатами проведеного навантажувального тестування можна зробити наступні висновки:

1. Система демонструє стабільну роботу під тривалим навантаженням. Протягом 2 годин 40 хвилин безперервного тестування з 15 одночасними користувачами не було зафіксовано жодного збою чи помилки.

2. Висока продуктивність бази даних. Середній час відповіді на запити становить 14,3 мс, а медіана — лише 6 мс, що значно нижче допустимого порогу в 100 мс для інтерактивних систем.

3. Нульовий відсоток помилок. З 72 014 запитів не було зафіксовано жодної помилки, що свідчить про високу надійність розробленої системи.

4. Ефективна робота MongoDB. База даних успішно обробляла одночасні операції читання та запису, підтверджуючи правильність вибору MongoDB для проекту та ефективність розробленої схеми даних.

5. Стабільне використання ресурсів. Моніторинг через Grafana показав відсутність аномалій у використанні пам'яті та з'єднань протягом всього тесту.

6. Запас для масштабування. Система використовувала лише незначну частину доступних ресурсів, що дозволяє збільшити кількість одночасних користувачів у декілька разів без погіршення продуктивності.

Таким чином, розроблена система Telegram-бота cargo_bot для логістичної компанії здатна забезпечити стабільну та швидку роботу в умовах реального навантаження і може бути рекомендована до впровадження у виробниче середовище.

ВИСНОВКИ

Кваліфікаційна робота є завершеною науковою роботою, в якій вирішена науково-практична задача обґрунтування структури та параметрів комп'ютерної системи для логістичної компанії з використанням Telegram-бота. В ході виконання роботи було проведено комплексний аналіз вимог до системи, розроблено архітектуру програмного забезпечення та експериментально підтверджено його працездатність.

Основні висновки і результати роботи полягають у наступному:

1. Проведено аналіз сучасних технологій розробки Telegram-ботів та обрано оптимальний стек технологій: мова програмування Python 3.13, фреймворк aiogram 3.x для асинхронної обробки повідомлень, база даних MongoDB 8.0 для гнучкого зберігання даних.

2. Розроблено архітектуру системи, що включає модулі обробки замовлень, управління водіями та транспортними засобами, систему сповіщень клієнтів, модуль генерації документів та AI-консультант на базі OpenAI GPT.

3. Реалізовано чатбот в месенджері Telegram з повним функціоналом для трьох категорій користувачів: клієнтів (створення та відстеження замовлень), операторів (обробка замовлень, призначення водіїв) та адміністраторів (управління системою, перегляд статистики).

4. Впроваджено систему моніторингу на базі Prometheus та Grafana для відстеження стану бази даних MongoDB в реальному часі, що дозволяє оперативно виявляти та усувати потенційні проблеми.

5. Проведено навантажувальне тестування системи за допомогою інструменту Locust. За 2 години 40 хвилин тестування система обробила 72 014 запитів без жодної помилки при середньому часі відповіді 14,3 мс.

6. Експериментально підтверджено стабільність роботи системи під навантаженням: база даних MongoDB демонструвала стабільні показники використання пам'яті та з'єднань протягом всього тестування.

7. Доведено придатність розробленої системи до впровадження у виробниче середовище логістичної компанії з можливістю обслуговування до 100 замовлень на добу та підтримкою одночасної роботи до 15-20 користувачів.

Таким чином, поставлені в роботі завдання виконано в повному обсязі. Розроблена система чатботу cargo_bot відповідає сучасним вимогам до програмного забезпечення для логістичних компаній та може бути рекомендована до практичного впровадження.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДСТУ 3008-2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. -К.: Держстандарт, 2015. – 26 с.

2. PyTelegramBotAPI – [Електронний ресурс]. https://pytba.readthedocs.io/en/latest/ (дата звернення: 17.09.2025).

3. MySQL Documentation – [Електронний https://dev.mysql.com/doc/ (дата звернення: 17.09.2025).

4. Цвіркун Л.І. Атестація здобувачів вищої освіти. Методичні рекомендації до виконання кваліфікаційної роботи магістра здобувачами галузі знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія / Л.І. Цвіркун, В.В. Гнатушенко, С.М. Ткаченко; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». –Дніпро: НТУ«ДП», 2024. – 54 с. (дата звернення: 10.11.2025).

5. Цвіркун Л.І. Глобальні комп'ютерні мережі. Програмування мовою PHP: навч.посібник / Л.І. Цвіркун, Р.В. Липовий, під заг. ред. Л.І. Цвіркуна. – Д.: Національний гірничий університет, 2013. – 239 с.

6. Цвіркун Л.І. Практична підготовка. Методичні рекомендації до виконання виробничої та передатестаційної практик магістрами галузі знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія / Л.І. Цвіркун, Д.О. Бешта, С.М. Ткаченко, В.В. Гнатушенко ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро: НТУ «ДП», 2022. – 21 с.

7. Цвіркун Л.І. Використання мессенджерів як системи оповіщення користувачів локальних систем домашньої автоматизації. / Л.І. Цвіркун, Л.В. Бешта, Ю.А. Миронов // Системні технології. Зб. наук. пр. НМЕтАУ. − 2021. − № 4. − с. 95−101

8. Цвіркун, Л.І. Розробка програмного забезпечення комп'ютерних систем. Програмування: навч. посібник / Л.І. Цвіркун, А.А. Євстігнєєва, Я.В.

Панферова, під заг. ред. Л.І. Цвіркуна. – 3-є вид., випр. – Д.: Національний гірничий університет, 2016. – 223 с. – ISBN 978-966-350-595-4.

9. Цвіркун Л.І. Розробка програмного забезпечення комп'ютерних систем. Програмування: навч. посіб. [Електронний ресурс] / Л.І. Цвіркун, А.А. Євстігнєєва, Я.В. Панферова ; під заг. ред. проф. Л.І. Цвіркуна ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – 1 електрон. опт. диск (CD-ROM) ; 12 см. – Систем. вимоги (мінімальні): Процесор 32-розрядний (x86) 233 МГц ; 512 МБ RAM ; 128 МБ Video ; від 4-х до 48-х CD-ROM ; Windows 7. – Назва з контейнера. – Дніпро: НТУ «ДП», 2019. – ISBN 978-966-350-638–8.

10. Панасюк, О.В., Іваненко, Р.С. (2021). Особливості створення чатботів для автоматизації бізнес-процесів. Вісник НТУУ "КПІ", 47(2), 76–83. (дата звернення: 11.10.2025).

11. Комп'ютерні мережі. Книга 2 : [навч. посіб.] / А. Г. Микитишин, М. М. Митник, П. Д. Стухляк, В. В. Пасічник. – Львів : «Магнолія 2006», 2019.– 328 с.

12. Компанієць В.В. Світові тренди сучасного транспортно-логістичного сервісу // Вісник економіки транспорту і промисловості No 70- 71, 2020 – с. 22-32. Режим доступу http://btie.kart.edu.ua/issue/download/13496/7147 (дата звернення 10.09.2025).

13. (https://old.libkor.com.ua/php/lib_theme_files/Bot.pdf) (дата звернення 10.11.2025).

14. https://sendpulse.kz/knowledge-base/chatbot/telegram/create-telegram-chatbo (дата звернення 15.11.25).

15. (https://delo.ua/news/logistika-ukrayini-vidnovlyujetsya-u-iii-kvartali-kilkist-novix-biznesiv-zrosla-vdvici-454279/ (дата звернення 25.11. 2025).

16. https://hub.kyivstar.ua/articles/galuzevi-trendi-stan-logistichnoyi-galuzi-v-ukrayini-trendi-ta-osoblivosti) (дата звернення 05.10. 2025).

17. https://blog.youcontrol.market/loghistika-pid-chas-viini-sotni-miliardiv-ghrivien-dokhodu/ (дата звернення 05.10. 2025).

18. https://www.oracle.com/ua/chatbots/what-is-a-chatbot/ (дата звернення 23.11. 2025).

19. https://logist.fm/publications/logistika-v-epohu-zmin (дата звернення 27.10. 2025).

20. Логування в Python. URL: https://docs.python.org/uk/3/howto/logging.html (дата звернення 04.12. 2025).

21. Створення асинхронних програм за допомогою Python (asyncio). URL: https://realpython.com/async-io-python/ (дата звернення 03.12. 2025).

22. Створення спеціалізованого чат-боту. URL: https://dou.ua/forums/topic/46902/ (дата звернення 03.11. 2025).

23. Python з MongoDB. URL: https://www.mongodb.com/resources/languages/python (дата звернення 30.11. 2025).

24. Паралельне виконання в Python. URL: https://docs.python.org/uk/3.11/library/concurrency.html (дата звернення 07.11. 2025).

25. Як створити Telegram-бота на python. URL: https://www.youtube.com/watch?v=oCD9-2IQtko (дата звернення 03.11. 2025).

26. Теорія систем масового обслуговування: навч. посібник / А. Л. Литвинов; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 141 с.

27. Словник термінів ІТ і комп'ютерної інженерії / В.В. Гнатушенко, Г.М. Коротенко, В.І. Олевський [та ін.]; за ред. В.В. Гнатушенка, Г.М. Коротенка, Л.І. Цвіркуна. - Дніпро : НТУ «ДП», 2025. − 709 с.

28. Tsvirkun, L., Myronov, Y. Challenges and Specificities of Adopting Continuous Integration within Scalable Cloud Environments. International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2023.

29. Миронов Ю., Цвіркун Л. (2024). Аналіз методів структурної оптимізації процесів неперервної інтеграції. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 133–139. https://doi.org/10.32782/IT/2024-3-14.

ДОДАТОК А
ТЕКСТ ПРОГРАМИ ЧАТБОТУ ЗАМОВЛЕНЬ ТА КОМУНІКАЦІЇ З КЛІЄНТАМИ ЛОГІСТИЧНОЇ КОМПАНІЇ

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ЧАТБОТУ ЗАМОВЛЕНЬ ТА КОМУНІКАЦІЇ З КЛІЄНТОМ ЛОГІСТИЧНОЇ КОМПАНІЇ

Текст програми

804.02070743.25008-01 12 01

Листів 14

2025

АНОТАЦІЯ

Дана програма містить програмний код чатботу cargo_bot, що є складовою комп'ютерної системи логістичної компанії для замовлень та комунікації з клієнтом.

Програма призначена для ведення діалогу з клієнтами та операторами, обробки замовлень на перевезення вантажів, розрахунку вартості доставки, генерації товарно-транспортних накладних (ТТН), інтеграції з AI-консультантом та взаємодії з базою даних MongoDB.

Програма написана мовою Python з використанням фреймворку aiogram 3.13 для роботи з Telegram Bot API, бібліотеки pymongo для взаємодії з MongoDB, бібліотеки python-docx для генерації документів, а також інтеграції з OpenAI GPT та Google Gemini для AI-консультацій.

Система розроблена з використанням сучасних інструментів програмування у середовищі Visual Studio Code. Розробка спрямована на підвищення ефективності бізнес-процесів логістичної компанії за рахунок автоматизації ключових операцій і покращення якості обслуговування клієнтів.

ЗМІСТ

С.

1 МОДУЛЬ MAIN

```
# Головний модуль запуску бота

import asyncio
import logging

from aiogram import Bot, Dispatcher
from aiogram.fsm.storage.memory import MemoryStorage
from aiogram.client.default import DefaultBotProperties
from aiogram.enums import ParseMode

from config import BOT_TOKEN
from app.handlers import router, set_bot
from app.metrics import start_metrics_server

# Налаштування логування
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'
)
logger = logging.getLogger(__name__)

# Створення бота та диспетчера
bot = Bot(
    token=BOT_TOKEN,
    default=DefaultBotProperties(parse_mode=ParseMode.HTML)
)
dp = Dispatcher(storage=MemoryStorage())
```

```
# Реєстрація роутера
dp.include_router(router)

# Встановлюємо бота для handlers
set_bot(bot)


async def main():
    """Головна функція запуску бота"""
    logger.info("=" * 50)
    logger.info("CARGO-BOT запускається...")
    logger.info("=" * 50)

    # Запускаємо сервер метрик для Prometheus
    start_metrics_server(port=8000)
    logger.info("Prometheus metrics: http://localhost:8000/metrics")

    # Видаляємо webhook та запускаємо polling
    await bot.delete_webhook(drop_pending_updates=True)

    logger.info("Бот успішно запущено!")
    logger.info("Для зупинки натисніть Ctrl+C")

    await dp.start_polling(bot)


if __name__ == "__main__":
    try:
        asyncio.run(main())
```

```
    except KeyboardInterrupt:
        logger.info("Бот зупинено користувачем")
    except Exception as e:
        logger.error(f"Критична помилка: {e}")
```

2 МОДУЛЬ CONFIG

2.1 API ключі та налаштування

```
# Конфігураційний файл Cargo-Bot
# Telegram-бот для логістичної компанії TransWay

# API ключі зовнішніх сервісів

# Токен Telegram бота (BotFather API)
BOT_TOKEN = "BOT_TOKEN "

# API ключ сервісу маршрутизації OpenRouteService
ORS_API_KEY = "ORS_API_KEY"

# API ключ Google Gemini для AI-консультанта
GEMINI_API_KEY = "GEMINI_API_KEY "

# API ключ OpenAI для AI-консультанта (альтернатива Gemini)
OPENAI_API_KEY = "OPENAI_API_KEY"  #
# Вибір AI провайдера: "gemini", "openai", або "auto" (спробує обидва)
AI_PROVIDER = "auto"

# Налаштування підключення до MongoDB
MONGO_URI = "mongodb://localhost:27017"
DB_NAME = "cargo_bot"
```

3 МОДУЛЬ HANDLERS (ключові фрагменти)

3.1 Імпорт бібліотек

```
# Модуль обробників повідомлень бота

import logging
import random
import string
import re
import os
import asyncio
import time
import uuid
from datetime import datetime, timedelta

from aiogram import Router, F, Bot
from aiogram.types import Message, FSInputFile, CallbackQuery
from aiogram.filters import Command, StateFilter
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import State, StatesGroup

from pymongo import MongoClient
import requests
import google.generativeai as genai
import openai

from config import (
    MONGO_URI, DB_NAME, ORS_API_KEY, GEMINI_API_KEY,
    TARIFFS, CARGO_TYPES, WEIGHT_CATEGORIES, DELIVERY_CONDITIONS,
    CITIES, LOCAL_COORDS, GEMINI_SYSTEM_PROMPT, EMAIL_ENABLED,
```

```
    INITIAL_OPERATORS, ADMIN_ID,
    AUTO_SIMULATION, SIMULATION_CONFIG,
    DELIVERY_TIME_CONFIG,
    PAYMENT_CARD, PAYMENT_CARD_DISPLAY, PAYMENT_PREPAYMENT,
PAYMENT_RECEIVER,
    OPENAI_API_KEY, AI_PROVIDER
)
from app.keyboards import (
    get_main_menu, get_operator_menu, get_cities_keyboard,
    get_cargo_types_keyboard, get_weight_keyboard, get_delivery_keyboard,
    get_confirm_keyboard, get_back_keyboard, get_ai_keyboard,
    get_status_keyboard, get_order_actions_keyboard, remove_keyboard,
    get_photo_keyboard, get_email_keyboard, get_order_details_keyboard,
    get_auth_method_keyboard, get_cancel_keyboard,
    get_confirm_cancel_keyboard,
    get_confirm_keyboard_inline, get_confirm_cancel_inline,
    get_status_inline, get_order_actions_inline, get_client_order_inline,
    get_role_keyboard,
    get_order_operator_actions, get_drivers_keyboard, get_vehicles_keyboard,
    get_ttn_actions_keyboard, get_confirm_ttn_keyboard,
    get_rating_keyboard, get_skip_comment_keyboard,
    get_client_order_details_keyboard, get_chat_keyboard,
    get_confirm_route_keyboard, get_operator_order_inline
)
from app.ttn_generator import generate_ttn, get_ttn_file, get_ttn_by_order
from app.email_sender import send_order_email, send_ttn_email, send_status_update_email,
send_cancellation_email, send_verification_code
from app.metrics import (
    track_message as metric_message,
    track_order as metric_order,
```

```
    track_error as metric_error,
    track_email as metric_email,
    track_api_request as metric_api,
    track_api_time as metric_api_time,
    track_ttn_generated as metric_ttn,
    track_ai_request as metric_ai,
    track_route as metric_route,
    track_order_cost as metric_cost,
    track_order_distance as metric_distance,
    track_auth as metric_auth,
    MESSAGE_PROCESSING_TIME,
    Timer
)

# Налаштування логування
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Підключення до MongoDB
client = MongoClient(MONGO_URI)
db = client[DB_NAME]
```

3.2 Підключення до MongoDB

```
client = MongoClient(MONGO_URI)
db = client[DB_NAME]

# Ініціалізація операторів в базі даних
def init_operators():
```

```
@router.message(Command("start"))
async def cmd_start(message: Message, state: FSMContext):
    """Обробник команди /start — вибір ролі"""
    await state.clear()

    # Метрика: команда /start
    metric_message("start")

    user_id = message.from_user.id
    username = message.from_user.username or ""
    first_name = message.from_user.first_name or "Користувач"

    # Зберігаємо/оновлюємо користувача
    client_doc = db.clients.find_one({"telegram_id": user_id})

    if not client_doc:
        db.clients.insert_one({
            "telegram_id": user_id,
            "username": username,
            "first_name": first_name,
            "created_at": datetime.now(),
            "last_activity": datetime.now()
        })
        logger.info(f"Новий користувач: {first_name} (ID: {user_id})")
    else:
        db.clients.update_one(
            {"telegram_id": user_id},
            {"$set": {"last_activity": datetime.now()}}
```

4 МОДУЛЬ KEYBOARDS

```
# Модуль клавіатур бота


from aiogram.types import (
    ReplyKeyboardMarkup, KeyboardButton, ReplyKeyboardRemove,
    InlineKeyboardMarkup, InlineKeyboardButton
)
from config import CITIES, CARGO_TYPES, WEIGHT_CATEGORIES,
DELIVERY_CONDITIONS


# Клавіатура вибору ролі
def get_role_keyboard():
    """Клавіатура вибору ролі при старті"""
    keyboard = ReplyKeyboardMarkup(
        keyboard=[
            [KeyboardButton(text="Я клієнт")],
            [KeyboardButton(text="Я оператор")]
```

4.1 Клавіатури типу reply

```
        ],
        resize_keyboard=True
    )
    return keyboard


# Головне меню клієнта
def get_main_menu():
    """Головне меню для клієнтів"""
    keyboard = ReplyKeyboardMarkup(
        keyboard=[
            [KeyboardButton(text="Нове замовлення"), KeyboardButton(text="Мої замовлення")],
```

```
            [KeyboardButton(text="Відстежити"), KeyboardButton(text="Тарифи")],
            [KeyboardButton(text="AI Консультант"), KeyboardButton(text="Зв'язок з оператором")],
            [KeyboardButton(text="Відгуки"), KeyboardButton(text="Контакти")]
        ],
        resize_keyboard=True
    )
    return keyboard

# Меню оператора
def get_operator_menu():
    """Меню для операторів"""
    keyboard = ReplyKeyboardMarkup(
        keyboard=[
            [KeyboardButton(text="Нові замовлення"), KeyboardButton(text="Всі замовлення")],
            [KeyboardButton(text="Статистика")],
            [KeyboardButton(text="Вийти з панелі")]
        ],
        resize_keyboard=True
    )
    return keyboard

def get_order_operator_actions(order_status="[НОВЕ]"):
    """Клавіатура дій з замовленням для оператора"""
    buttons = []

    if order_status == "[НОВЕ]":
        buttons.append([KeyboardButton(text="Взяти в роботу")])
    elif order_status == "[В ОБРОБЦІ]":
        buttons.append([KeyboardButton(text="Сформувати ТТН")])
```

4.2 Клавіатури типу inline

```
    return keyboard


def get_ttn_actions_keyboard():
    """Клавіатура після формування ТТН"""
    keyboard = ReplyKeyboardMarkup(
        keyboard=[
            [KeyboardButton(text="Надіслати ТТН клієнту")],
            [KeyboardButton(text="Змінити статус на [ДОРОГА]")],
            [KeyboardButton(text="Назад до замовлення")]
        ],
        resize_keyboard=True
    )
    return keyboard

# Клавіатура вибору методу авторизації
def get_auth_method_keyboard():
    """Клавіатура вибору методу авторизації оператора"""
    keyboard = ReplyKeyboardMarkup(
        keyboard=[
            [KeyboardButton(text="За Telegram ID"), KeyboardButton(text="За токеном")],
            [KeyboardButton(text="За Email")],
            [KeyboardButton(text="Скасувати")]
        ],
        resize_keyboard=True
    )
    return keyboard
```