Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система керування вантажними перевезеннями.
Back-end адмінської частини
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-6

Мотречко В. В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Саманцов О.О.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Мотречку Володимиру Володимировичу
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система керування вантажними перевезеннями. Back-end адмінської частини.

Затверджена наказом по університету від 20.05. 2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 02.06.2024

3. Вихідні дані до роботи Розробити Back-end частину програмної системи для контролю керування вантажними перевезеннями

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, впровадження, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	*виконано*
2	Створення специфікації ПЗ	15.04.2024	*виконано*
3	Проектування ПЗ	25.04.2024	*виконано*
4	Розробка ПЗ	19.05.2024	*виконано*
5	Тестування ПЗ	23.05.2024	*виконано*
6	Оформлення пояснювальної записки	24.05.2024	*виконано*
7	Підготовка презентації та доповіді	25.05.2024	*виконано*
8	Попередній захист	26.05.2024	*виконано*
9	Нормоконтроль, рецензування	02.06.2024	*виконано*
10	Здача роботи у електронний архів	03.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	07.06.2024	*виконано*

Дата видачі завдання 08 травня 2024р.

Студент _____ Мотречко В.В.
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Саманцов О.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра: 89 с., 58 рис., 10 джерела інформації.

БАЗА ДАНИХ, ЛОГІСТИКА, ІНФОРМАЦІЙНА СИСТЕМА, МУВІНГ, ПРОЄКТУВАННЯ, KOTLIN, DOMAIN DRIVEN DESIGN

Мета розробки – розробка програмної системи для управління вантажними перевезеннями, реалізація необхідної функціональності для контролю якості вантажних перевезень, вирішення конфліктних ситуацій, що виникають під час перевезення, та аналіз статистики щодо перевезень. Як метод вирішення цієї задачі було обрано середовище для розробки IntelliJ Idea, мови програмування – Kotlin на базі використання фреймворку Spring Boot. Для організації взаємодії між сервісами використовується система віддаленого виклику процедур з відкритим кодом gRPC та REST API.

У результаті розробки була створена back-end частина програмної системи для управління вантажними перевезеннями, яка забезпечує можливість контролю за перевезенням вантажу, вирішення конфліктних ситуацій між замовником та водієм, а також аналіз статистики перевезень.

DATABASE, DESIGN, DOMAIN DRIVEN DESIGN, INFORMATION SYSTEM, KOTLIN, LOGISTICS, MOVING

The purpose of the pre-certification practice is to develop a software system for managing freight transportation, implementing the necessary functionality to control the quality of freight transportation, resolving conflict situations that arise during transportation, and analyzing transportation statistics. As a method of solving this problem, we chose the IntelliJ Idea development environment, Kotlin programming language based on the Spring Boot framework. The open-source remote procedure call system gRPC and the REST API are used to organize interaction between services.

As a result of the development, the back-end part of the software system for cargo transportation management was created, which provides the ability to control cargo transportation, resolve conflict situations between the customer and the driver, and analyze transportation statistics.

Я, Мотречко Володимир Володимирович, студент гр. ПЗПІ-20-6, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система керування вантажними перевезеннями. Back end адмінської частини.», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений із діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Індустрія переїздів відіграє важливу роль у логістичному та транспортному секторі, сприяючи ефективним переїздам для бізнесу та особистим переїздам. Однак, незважаючи на свою важливість, галузь часто стикається з такими проблемами, як неефективність, відсутність кастомізації та недостатня інтеграція технологій, особливо на ринках, що розвиваються, таких як Україна.

Ця кваліфікаційна робота має на меті вирішити ці проблеми шляхом проектування та розробки інтегрованого додатку, адаптованого для українського ринку перевезень.

Основними цілями цієї роботи є:

– розробити гнучку та ефективну систему, яка спрощує адміністрування та моніторинг за системою;

– інтегрувати передові технології для покращення можливостей кастомізації та планування;

– покращити загальний користувацький досвід завдяки оновленням даних в режимі реального часу, та сучасним технологіям.

Завдання, окреслені для цього проекту, включають:

– розробку загальної архітектури системи, яка підтримує взаємодію в режимі реального часу та управління даними;

– розробку алгоритмів, які оптимізують моніторинг, звітність та швидке реагування на несподівані події у системі;

Панель адміністратора є критично важливим компонентом запропонованого додатку для надання послуг переїзду, розробленого спеціально для адміністраторів та допоміжного персоналу для ефективного нагляду та управління системою. Панель адміністратора слугує центром управління додатком мувінгової служби, надаючи адміністраторам інструменти, необхідні для комплексного управління користувачами, водіями. Вона включає в себе наступні функції:

– реєстрація та керування обліковими записами користувачів;

– перевірка та керування профілями та розкладами водіїв;

– моніторинг в режимі реального часу активних завдань на перевезення та їх статусів;

– обробка та вирішення системних помилок і скарг користувачів;

– детальна аналітика та звітність для прийняття рішень на основі даних.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної галузі

1.1.1 Вступ до галузі переїздів

Галузь переїздів охоплює широкий спектр послуг, які мають на меті допомогу як окремим особам, так і бізнесам у перенесенні їхнього майна з одного місця на інше. Діяльність у цій галузі варіюється від резиденційних переїздів до переїздів офісів і спеціалізованих послуг з переміщення великих або крихких предметів. Ця індустрія відчуває вплив різних глобальних факторів, включаючи урбанізацію, економічні зміни та зміни в поведінці споживачів, які часто впливають на частоту та відстань переїздів.

1.1.2 Сучасні тренди ринку

Інтеграція цифрових технологій: З ростом цифровізації, послуги переїзду все більше включають цифрові рішення для оптимізації операцій, від автоматизованих систем бронювання до реалізації можливості відстеження в реальному часі. Такі інновації спрощують логістику і покращують загальну ефективність.

Фокус на досвід споживачів: Сучасні споживачі очікують персоналізованого обслуговування і повної прозорості процесів переїзду. Це спонукало компанії звертати більшу увагу на обслуговування клієнтів і підтримку, з використанням технологій для підвищення взаємодії та задоволення клієнтів.

Екологічні турботи: Зростає тенденція до екологічності в індустрії, з компаніями, які приймають екологічно чисті практики, такі як використання рециклованих упаковочних матеріалів та оптимізація маршрутів для зниження викидів вуглецю.

1.1.3 Технологічний ландшафт

Технології істотно трансформують традиційні послуги переїзду, роблячи їх більш ефективними і зручними для клієнтів. Найважливіші технологічні інновації включають телематику та системи управління автопарком, мобільні застосунки та хмарні обчислення.

Телематика та системи управління автопарком: Ці системи забезпечують дані в реальному часі про місцезнаходження транспортних засобів, оптимізацію маршрутів і управління паливом, що є вирішальним для зниження оперативних витрат і підвищення надійності послуг.

Мобільні застосунки: Мобільні застосунки надають користувачам можливості замовлення, оплати та відстеження, все зі смартфона, підвищуючи доступність та зручність послуг переїзду.

Хмарні обчислення: Використання хмарних платформ дозволяє компаніям ефективно масштабувати свою діяльність та безпечно зберігати великі обсяги даних, включаючи інвентар та інформацію про клієнтів.

1.1.4 Аналіз конкурентного середовища

Індустрія перевезень в Україні є динамічною та конкурентною, з різними гравцями, починаючи від великих національних компаній, таких як "Нова Пошта", до спеціалізованих місцевих фірм, таких як "Мураха" в Києві. Ці компанії встановили значну присутність і встановили високі стандарти в галузі. Однак навіть у цьому конкурентному середовищі існують стратегічні шляхи, за допомогою яких новий учасник може не тільки конкурувати, але й потенційно досягти успіху.

1.1.5 Огляд основних конкурентів

Першим з розглянутих основних конкурентів є Нова Пошта. До сильних сторін можна віднести: розгалужена мережа, передова технологічна інтеграція, різноманітні логістичні послуги. Пропонуючи широкий спектр послуг, "Нова Пошта" може не забезпечити той самий рівень персоналізованої уваги та індивідуального обслуговування, який можуть запропонувати менші спеціалізовані компанії. Саме це і може бути слабкою стороною конкурета.

Наступним розглянутим конкурентом на ринку є Мураха. Серед сильних сторін цієї компанії: місцевий досвід у Києві, комплексні рішення для переїзду, конкурентні ціни на місцевому ринку. Слабкою стороною можна вважати обмежене географічне покриття за межами Києва, що може бути недоліком для клієнтів, які прагнуть вийти на національний рівень.

1.1.6 Можливості для отримання конкурентних переваг

Щоб зайняти конкурентну нішу на українському ринку переїздів, нова компанія може застосувати такі стратегічні підходи:

а) спеціалізація та персоналізація:

1) унікальна торгова пропозиція (УТП): зосередження на високо персоналізованих послугах, пристосованих до конкретних потреб клієнтів, таких як елітні послуги з переїзду або спеціалізовані послуги з перевезення чутливого обладнання. Це може відрізнити послугу від ширших, але менш спеціалізованих пропозицій "Нової пошти".

2) залучення клієнтів: впровадження надійної системи зворотного зв'язку з клієнтами та постійного вдосконалення. Залучення клієнтів після надання послуг може допомогти вдосконалити пропозиції та підвищити рівень задоволеності – сфера, в якій великі компанії можуть не досягти успіху.

б) технологічна перевага:

1) інноваційні технічні рішення: розробка та інтеграція найсучасніших технологій, що виходять за рамки базових систем відстеження та бронювання.

2) аналітика даних: Використовувати передову аналітику для прогнозування тенденцій переїзду клієнтів і персоналізації маркетингових стратегій, потенційно пропонуючи послуги заздалегідь на основі прогнозованої поведінки.

в) орієнтація на навколишнє середовище: інвестиція в екологічні практики, такі як багаторазове використання пакувальних матеріалів, електричні транспортні засоби та програми компенсації викидів вуглецю. Це не тільки допомагає навколишньому середовищу, але й приваблює екологічно свідомих споживачів, виділяючи компанію серед конкурентів, таких як "Мураха".

г) стратегічні партнерства: укладення альянсів з компаніями, що займаються нерухомістю, фірмами, що займаються облаштуванням житла, та дизайнерами інтер'єрів, щоб пропонувати комплексні послуги. Такий інтегрований підхід до надання послуг може зробити процес переїзду більш плавним для клієнтів і забезпечити комплексне рішення, яке конкуренти можуть не запропонувати.

1.1.7 Потенційні виклики та стратегії їх подолання

Конкурувати з такими відомими брендами, як "Нова Пошта", може бути непросто. Щоб вирішити цю проблему, можна запустити на початку агресивні маркетингові кампанії та партнерство з відомими брендами можуть допомогти підвищити впізнаваність.

Вихід на ринок з усталеними гравцями вимагає значних інвестицій. Пропонування вступних цін або пакетних послуг може залучити перших клієнтів і згенерувати рекомендації з вуст в уста.

Зосередившись на персоналізованих послугах, технологічних інноваціях, екологічній стійкості та стратегічному партнерстві, нова мувінгова компанія може не тільки конкурувати, але й встановити нові стандарти в українській мувінговій індустрії. Ці стратегії слід виділити як ключові диференціатори в тезі, що демонструють потенційні конкурентні переваги над усталеними гравцями, такими як "Нова Пошта" та "Мураха".

1.1.8 Регуляторне середовище

Індустрія переїздів підлягає різноманітним регуляціям, які варіюються в залежності від країни та регіону. Це може включати транспортні правила, трудове законодавство і стандарти щодо обробки та транспортування товарів. Дотримання цих правил є критично важливим для уникнення правових проблем і забезпечення безпеки працівників і клієнтів.

1.2 Виявлення та вирішення проблем

1.2.1 Визначення ключових проблем

Проаналізована галузь перевезень в Україні стикається з кількома значними проблемами, які можна класифікувати як операційна неефективність, недостатня технологічна інтеграція та вплив на навколишнє середовище. Кожна з цих категорій охоплює конкретні проблеми, які в сукупності погіршують якість послуг і перешкоджають зростанню галузі.

Проблема операційної неефективності включає в себе:

– Фрагментований ринок: велика кількість дрібних операторів часто призводить до непослідовної якості послуг. Без єдиних стандартів споживачі стикаються з труднощами у виборі надійних постачальників.

– Недостатнє планування маршруту: більшість мувінгових компаній не використовують передові технології оптимізації маршрутів, що призводить до збільшення часу в дорозі та зростання операційних витрат.

– Ручні операції: Покладання на ручні процеси бронювання, планування та управління переїздами збільшує ймовірність людських помилок і знижує загальну ефективність.

До технологічних недоліків можна віднести:

– Низький рівень впровадження технологій: Існує значне відставання у впровадженні сучасних технологій, які полегшують онлайн-бронювання, відстеження в режимі реального часу та аналітику даних.

– Проблеми з управлінням даними: Невеликим мувінговим компаніям часто бракує інфраструктури для ефективного управління даними, що призводить до низького рівня обслуговування клієнтів і нездатності приймати обґрунтовані рішення.

– Відсутність цифрових платіжних рішень: Обмежене використання цифрових платіжних платформ може створювати незручності для клієнтів і обмежувати сферу діяльності мувінгових компаній.

Серед екологічних проблем можна відмітити:

– Використання застарілих транспортних засобів: Багато компаній використовують застарілі транспортні засоби, які не є економічними та мають високий рівень викидів, що негативно впливає на екологічну стійкість.

– Утилізація відходів: Процес переїзду часто призводить до утворення значної кількості відходів, включаючи пакувальні матеріали, такі як картон і пластик, які не переробляються або утилізуються відповідально.

1.2.2 Стратегії вирішення проблем

Для вирішення цих виявлених проблем необхідний багатогранний підхід, який охоплює технологічні інновації, вдосконалення нормативно-правової бази та екологічний менеджмент.

Для підвищення операційної ефективності слід сфокусуватися на галузевій стандартизації, удосконаленні оптимізації маршрутів та автоматизації послуг. Розробка та впровадження галузевих стандартів може допомогти уніфікувати

якість послуг, що полегшить клієнтам довіру та вибір мувінгових послуг. Впровадження інструментів планування маршрутів на основі GPS і штучного інтелекту може значно скоротити час у дорозі та споживання пального. Впровадження автоматизованих систем для бронювання, управління взаємовідносинами з клієнтами та операціями може впорядкувати процеси та зменшити кількість помилок, що допускаються вручну.

Для вирішення проблем також варто використати інтеграцію технологій, яка передбачає впровадження нових технологій, якісну аналітику даних та інтеграцію цифрових платежів. Заохочення використання надійних IT-рішень для управління бронюваннями, відстеження товарів у реальному часі та збору відгуків клієнтів підвищить прозорість та ефективність. Використання інструментів аналізу даних для аналізу поведінки клієнтів, оптимізації операцій і прогнозування ринкових тенденцій може принести значну користь мувінговим компаніям. Впровадження безпечних методів цифрових платежів може забезпечити зручність для клієнтів і розширити операційні можливості сервісів.

Для підвищення екологічної стійкості можна приділити увагу дослідженню політики щодо екологічно чистих транспортних засобів та ініціативі з переробки відходів. Сприяння використанню нових, більш економних транспортних засобів або транспортних засобів на альтернативних видах палива може зменшити вплив перевезень на навколишнє середовище. Створення програм з переробки або повторного використання пакувальних матеріалів, в свою чергу, може мінімізувати відходи та сприяти підвищенню екологічної відповідальності.

Вирішуючи ці проблеми за допомогою окреслених стратегій, галузь перевезень в Україні може підвищити свою операційну ефективність, ефективно інтегрувати сучасні технології та зменшити свій вплив на навколишнє середовище. Такі вдосконалення не лише принесуть користь постачальникам послуг, зменшивши витрати та залучивши більше клієнтів, але й підвищать рівень задоволеності клієнтів та сприятимуть досягненню ширших суспільних цілей, таких як екологічна стійкість. Такий комплексний підхід гарантує, що рішення будуть стійкими та ефективними, прокладаючи шлях до більш надійної та готової до майбутнього індустрії перевезень.

1.3 Постановка проблеми

Спираючись на ідентифікацію та попередні стратегії вирішення проблеми, описані в попередніх розділах, необхідно сформулювати комплексну постановку проблеми, яка охоплює фундаментальні виклики, що стоять перед галуззю перевезень в Україні. Таке формулювання проблеми слугує наріжним каменем для подальшої розробки та впровадження індивідуального технологічного рішення, спрямованого на трансформацію галузі.

Основна проблема, з якою стикається галузь перевезень в Україні, полягає в неефективності та застарілих операційних методах, які заважають сектору ефективно реагувати на вимоги сучасного швидкозмінного ринку. Відсутність стандартизованих процесів, недостатнє використання технологій та екологічна недбалість разом погіршують якість послуг та операційну ефективність, перешкоджаючи конкурентоспроможності та сталому розвитку галузі.

Операційна неефективність провокує наступні наслідки:

– нестабільна якість послуг через фрагментованість ринку з безліччю дрібних операторів;

– неоптимальне планування маршрутів, що призводить до збільшення часу доставки та споживання пального;

– переважно ручні процеси бронювання та планування, схильні до помилок та неефективності.

Технологічні недоліки стають причиною:

– обмеженої технологічної інтеграції, що обмежує можливості відстеження, взаємодії з клієнтами та оперативного управління;

– недосконалості систем управління даними, що призводить до неефективного прийняття рішень та управління взаємовідносинами з клієнтами;

– мінімального впровадження цифрових платіжних систем, що обмежує зручність для клієнтів та операційну гнучкість для постачальників.

Якщо виділити вплив на навколишнє середовище, то продовження використання застарілих, неефективних транспортних засобів призводить до збільшення викидів та погіршення стану довкілля. І не варто забувати про недостатню переробку та утилізацію матеріалів, що використовуються в процесі перевезень, таких як картон та пластик.

1.4 Постановка цілей для розвитку

Основною метою для вирішення цих проблем є розробка комплексної цифрової платформи, спеціально розробленої для української індустрії перевезень. Ця платформа має на меті інтегрувати передові технологічні рішення, які оптимізують операції, підвищують рівень залучення клієнтів та сприяють екологічній стійкості.

Цілі цифрової платформи:

а) операційна досконалість:

1) розробити уніфіковану платформу, яка стандартизує пропозиції послуг у всій галузі, підвищуючи надійність і довіру;

2) впровадити вдосконалений алгоритм оптимізації маршрутів, який скорочує час у дорозі та зменшує використання пального;

3) автоматизувати обслуговування клієнтів та операційні процеси, щоб зменшити кількість помилок, що допускаються вручну, та операційні витрати.

б) технологічна інтеграція:

1) інтегрувати технологію відстеження в режимі реального часу, щоб забезпечити прозорість і підвищити рівень задоволеності клієнтів;

2) розгорнути надійну систему управління даними, яка дозволить ефективно приймати рішення на основі аналітики;

в) екологічна відповідальність:

1) сприяти використанню екологічно чистих транспортних засобів у мережі платформи;

2) налагодити партнерство зі службами переробки для ефективного управління відходами, що утворюються від пакувальних матеріалів.

Ця постановка проблеми та поставлені цілі мають на меті спрямувати розробку рішення, яке не лише усуває безпосередні операційні та технологічні недоліки, але й узгоджується з ширшими екологічними цілями.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Під час виконання кваліфікаційної роботи треба спроектувати систему для адмін панелі системи для перевезення вантажів. Дана система має надавати користувачам наступні функції:

1. Управління інформаційною панеллю

Вимога: Система повинна надавати інформаційну панель, яка пропонує огляд найважливіших даних, включаючи звернення, активність користувачів, замовлення та статистику.

Функціональність: Інформаційна панель повинна дозволяти адміністратору швидко переглядати та переходити до детальних розділів, таких як інформація про клієнта, списки роботодавців(водіїв) та деталі замовлень.

2. Управління користувачами

Створення/видалення користувачів:

Вимога: Можливість створювати та видаляти облікові записи користувачів як для клієнтів, так і для співробітників, що включає в себе збір основної інформації про користувача під час створення та забезпечення цілісності даних при видаленні користувачів.

3. Аутентифікація співробітників

Вимога: Безпечна функція входу для всіх співробітників у систему, що забезпечує безпеку даних та контроль доступу. Використання токенів авторизації, для поглибленого захисту.

4. Управління підтримкою

Обробка звернень в технічну підтримку:

Вимога: Функціонал для перегляду, створення, оновлення та видалення тікетів підтримки. Це включає в себе написання та оновлення відповідей на тікети, а також управління як головною підтримкою, так і звичайними ролями підтримки. Також ця вимога включає в себе розробку та підтримку бота, для месенджера telegram, який надає можливість перегляду створених користувачем звернень, створити нове звернення або почати чат з підтримкою, по існуючому зверненню.

5. Пошук і фільтрація:

Вимога: Можливість пошуку звернень і користувачів за різними критеріями та полями для ефективного управління запитами на підтримку.

5. Управління інформацією про клієнтів та роботодавців

Перегляд деталей:

Вимога: Можливість перегляду детальних списків та інформації про клієнтів і роботодавців, яка включає контактні дані, історію обслуговування та поточний статус.

6. Змінювати інформацію:

Вимога: Адміністратори повинні мати можливість оновлювати інформацію про клієнтів і роботодавців(водіїв) за необхідності, щоб відобразити зміни в послугах або контактних даних.

7. Управління замовленнями

Контроль замовлень:

Вимога: Система повинна дозволяти переглядати та шукати всі замовлення, з можливістю перегляду детальної інформації про замовлення, включаючи пакети послуг, дати та інформацію про клієнта.

8. Статистична звітність

Перегляд статистики стосовно замовлень, та звернень в технічну підтримку.

Вимога: Можливість переглядати вичерпну статистику, пов'язану із замовленнями та зверненнями для аналізу показників ефективності компанії.

9. Додаткові функції

Стабільність, низький час відгуку на запит

Вимога**:** Система повинна мати високу стабільність, відмовостійкість, низький час відгуку на запит користувача. Система повинна забезпечувати безперебійну роботу протягом тривалого часу. Це передбачає мінімізацію збоїв та переривань у роботі системи. Профілювання та оптимізація коду для зменшення часу виконання запитів та підвищення продуктивності системи. Використання механізмів кешування (Redis) для зберігання часто запитуваних даних, що дозволяє зменшити час доступу до них. Оптимізація запитів до бази даних, використання індексів та налаштування параметрів бази даних для швидкої обробки запитів.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Детальний огляд адміністративних можливостей

У контексті нашої комплексної системи програмного забезпечення для надання послуг переїзду, роль адміністратора є ключовою. Як показано на наданій UML-діаграмі, адміністратор відіграє багатогранну роль, яка охоплює широкий спектр функціональних можливостей, призначених для забезпечення безперебійної та ефективної роботи платформи. У цьому розділі детально розглядається кожна функція, яку має право виконувати адміністратор, підкреслюючи вирішальну роль цієї ролі у підтримці операційної цілісності та ефективності системи.

На рисунку 4.1 зображено use case діаграму для адміністратора системи, нижче розглянуто її більш детально.

Функції керування користувачами

Створення та видалення користувачів

– Створення користувачів: Адміністратор має можливість додавати нових користувачів до системи. Це передбачає введення в систему важливої інформації, такої як імена, адреси, ролі та інші відповідні дані. Ця функція має вирішальне значення, оскільки вона гарантує, що лише авторизовані та легітимні користувачі зможуть отримати доступ до системи та взаємодіяти з нею.

– Видалення користувачів: Адміністратор також має право видаляти користувачів з системи. Ця функція особливо важлива для підтримки безпеки системи, оскільки вона дозволяє адміністратору оперативно видаляти користувачів, які більше не потребують доступу до системи або чий доступ може становити ризик для її цілісності.

– Можливості управління підтримкою.

Робота з персоналом служби підтримки

Створення та видалення керівника служби підтримки та допоміжного персоналу: Адміністратор може призначати або звільняти головного спеціаліста з підтримки та інший допоміжний персонал. Це важлива адміністративна функція, оскільки вона безпосередньо впливає на якість обслуговування клієнтів і пропонованої підтримки. Вона дозволяє адміністратору гарантувати, що команда

підтримки укомплектована компетентним і ефективним персоналом, який може ефективно розглядати і вирішувати проблеми користувачів.

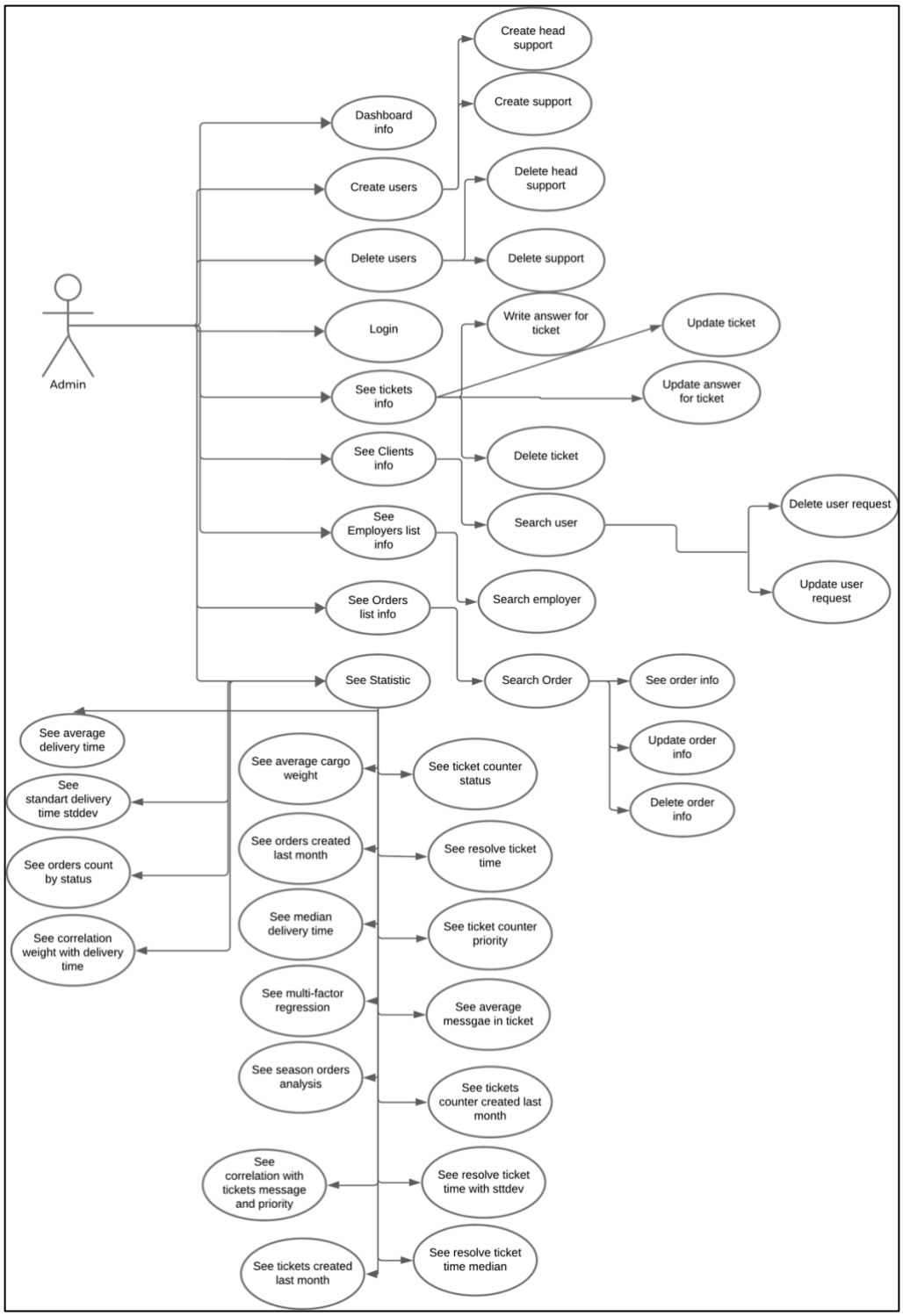


Рисунок 4.1 – Use Case діаграма для адміністратора

Операції з управління тікетами

Керування відповідями на тікети: Здатність адміністратора писати, оновлювати та видаляти відповіді на тікети відіграє вирішальну роль у відносинах з клієнтами. Це включає підготовку початкових відповідей на запити та оновлення цих відповідей у міру розвитку ситуації або отримання додаткової інформації. Функція видалення відповідей має вирішальне значення для управління інформацією, що надається користувачам, гарантуючи, що вона залишається точною і актуальною.

Моніторинг та звітність

– Детальний перегляд інформації: Адміністратори мають доступ до детального перегляду інформації про клієнтів, списків роботодавців та деталей замовлень. Ця функція необхідна для моніторингу поточної діяльності на платформі та забезпечення безперебійної та ефективної роботи.

– Можливості пошуку: Можливість пошуку конкретних користувачів, роботодавців або замовлень розширює можливості адміністратора щодо ефективного управління платформою. Це дозволяє швидко знаходити інформацію, яка необхідна для вирішення конкретних запитів або проблем, що можуть виникнути.

Статистичні дані та показники ефективності:

– Доступ до статистики та даних компанії: Перегляд статистичних даних та інформації про компанії, пов'язані з платформою, дозволяє адміністраторам приймати обґрунтовані рішення на основі реальних даних.

– Аналіз продуктивності: Функції для перегляду середніх показників, успішних і неуспішних пакетів є життєво важливими для контролю якості та покращення сервісу. Ці інструменти дають адміністратору чітке уявлення про те, наскільки добре працюють різні аспекти послуги, виділяючи області, де необхідні поліпшення.

Користувачі системи, з іншими ролями, мають схожий функціонал, в залежності від того, що саме дозволяє робити їм їхня роль.

Також, як ще одна додаткова функція, був розроблений telegram бот, для спілкування користувача з технічною підтримкою. На рисунку 3.2 зображено Use Case діаграму для взаємодії користувача.

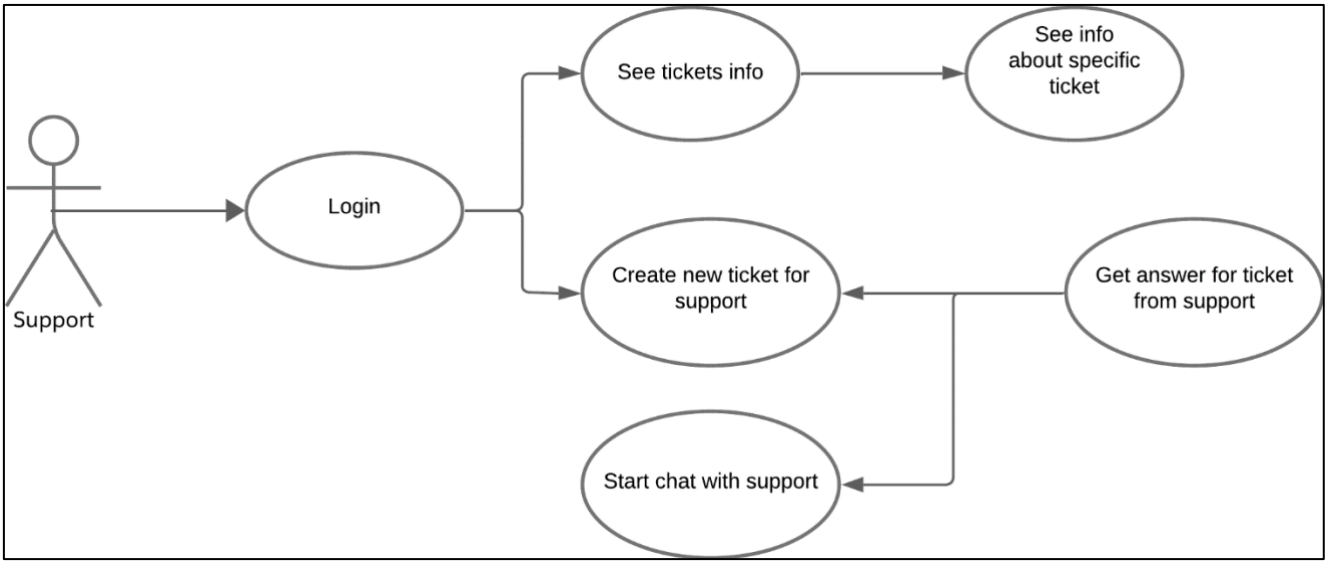


Рисунок 3.2 – Use Case діаграма для користувача

– Вхід: Користувач входить до системи через Telegram-бота, надаючи необхідні аутентифікаційні дані.

– Перегляд інформації про тікети: Після входу користувач може переглядати загальну інформацію про всі свої тікети.

– Перегляд інформації про конкретний тікет**:** Користувач може вибрати конкретний тікет для перегляду більш детальної інформації про нього.

– Створення нового тікета: Користувач може створити новий тікет для звернення до технічної підтримки.

– Отримання відповіді на тікет від підтримки: Після створення тікета користувач може отримувати відповіді від технічної підтримки.

– Почати чат з підтримкою: Користувач може ініціювати чат з технічною підтримкою для вирішення питань в режимі реального часу.

Ця діаграма ілюструє, як користувач може взаємодіяти з системою через Telegram-бота для отримання технічної підтримки. Бот забезпечує зручний та швидкий доступ до інформації про тікети і дозволяє ефективно спілкуватися з технічною підтримкою.

3.2 Проєктування архітектури ПЗ

При розробці програмної системи для служби переїздів був прийнятий складний і сучасний архітектурний підхід, зокрема, з акцентом на доменно-

орієнтованому проектуванні (DDD) в поєднанні з гексагональною архітектурою. У цьому розділі детально розглядається, як ці принципи проектування інтегровані в загальну архітектуру системи, забезпечуючи гнучкість, масштабованість і ремонтопридатність. Крім того, буде пояснено діаграму розгортання, щоб проілюструвати, як різні компоненти системи взаємодіють один з одним і з зовнішніми об'єктами.

Доменно-орієнтоване проектування [2] – це методологія, яка зосереджується на комплексних потребах, пов'язуючи реалізацію з еволюціонуючою моделлю основних бізнес-концепцій (див. рис. 4.2).

DDD включає наступні стратегічні та тактичні аспекти проектування:

– повсюдна мова: створення спільної мови між розробниками та експертами предметної області для забезпечення узгодженості у всьому додатку;

– обмежені контексти: основна предметна область поділяється на кілька обмежених контекстів для мінімізації залежностей і уточнення різних областей функціональності, гарантуючи, що модель залишається логічно послідовною в межах кожного контексту;

– сутності та об'єкти цінності: розрізнення між сутностями, які мають чітку ідентичність, та об'єктами-значеннями, які описують характеристики;

– агрегати: кластер об'єктів предметної області, які можна розглядати як єдине ціле при зміні даних, що забезпечує цілісність і узгодженість.

Використання DDD у цьому проекті забезпечує структурований підхід до розробки програмної системи, який відображає тонкощі та глибоке розуміння предметної області, сприяючи кращій комунікації між членами команди та більш згуртованим і цілеспрямованим модулям системи.

Гексагональна архітектура

Гексагональна архітектура [3] (див. рис 4.3), також відома як архітектура портів та адаптерів, використовується для подальшого вдосконалення дизайну системи, зосереджуючи увагу на основній логіці, відокремлюючи її від зовнішніх проблем.

Цей стиль архітектури дозволяє:

– Кілька інтерфейсів для програми: Ядро програми оточене шестикутником, до якого можна приєднати різні бічні адаптери. Це робить додаток дуже гнучким з точки зору середовища та вихідного або вхідного характеру потоку даних.

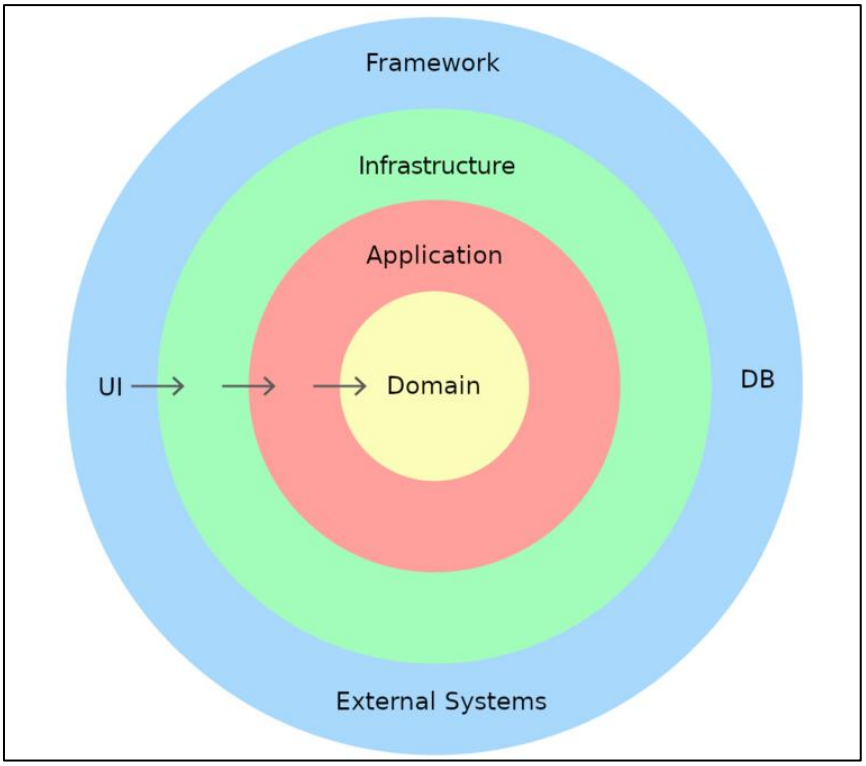


Рисунок 4.2 – Принципи доменно-орієнтованого проектування (DDD)

Нижче наведено приклад(рис 4.3), як повинна виглядати гексагональна архітектура.

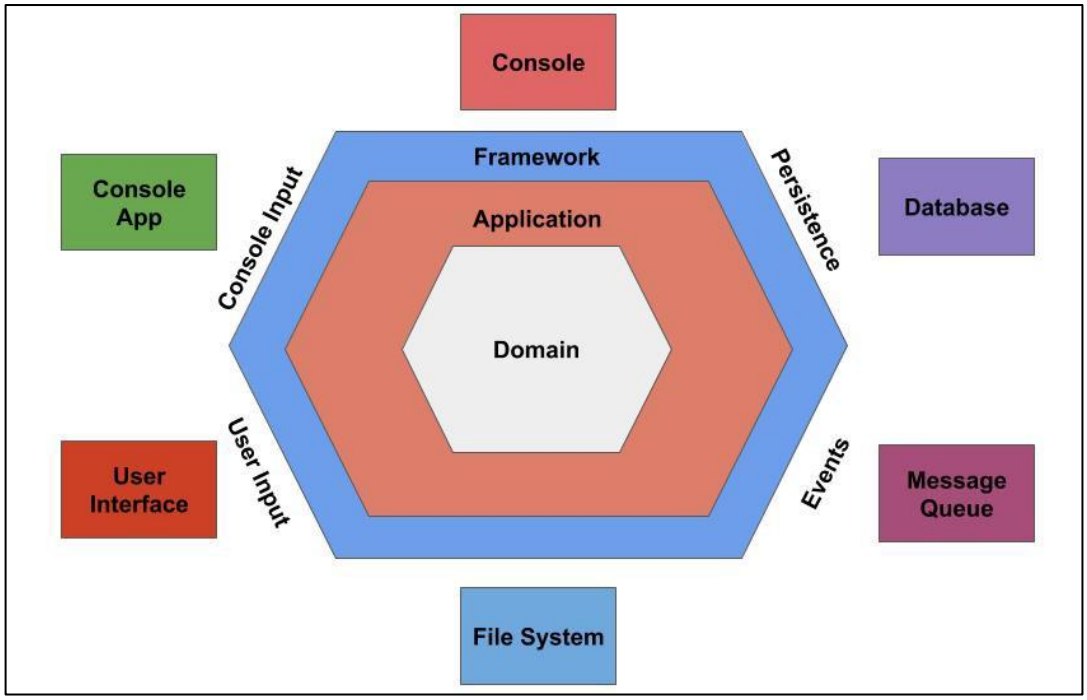


Рисунок 4.3 – Принцип гексагональної архітектури

– Незалежність від зовнішніх агентств: Завдяки визначенню чітких портів, основна функціональність системи залишається незалежною від зовнішніх технологій і механізмів доставки. Це полегшує тестування та обслуговування.

– Адаптери: На зовнішній стороні шестикутника адаптери забезпечують зв'язок між зовнішнім світом і портами, визначеними на ядрі, які керують входами і виходами даних додатку.

В кінці у нас виходить наведена на рис. 4.4 структура проекту, для дотримання чистої архітектури.

Наведена на рис. 4.5 діаграма розгортання пояснює, як різні компоненти системи конфігуруються та взаємодіють за допомогою різних протоколів.

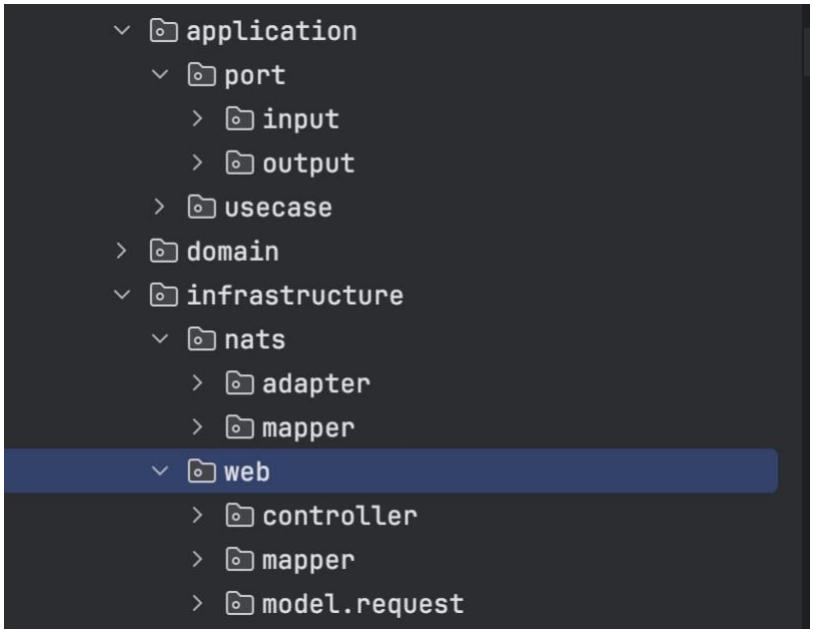


Рисунок 4.4 – Організація проекту в IDE

Telegram API:

Цей компонент відповідає за взаємодію з Telegram сервером через HTTP. Він забезпечує обробку повідомлень та передачу їх до сервісу підтримки бота.

Support Bot SVC:

Сервіс підтримки бота реалізований за допомогою Kotlin Spring Application, який працює в середовищі Tomcat Servlet Engine. Він взаємодіє з Telegram API через HTTP і з Redis базою даних, що розміщена в контейнері Docker, для кешування та швидкого доступу до даних.

Docker:

Docker використовується для розгортання Redis бази даних, що забезпечує швидкий і надійний доступ до даних, необхідних для роботи сервісу підтримки бота.

User-svc:

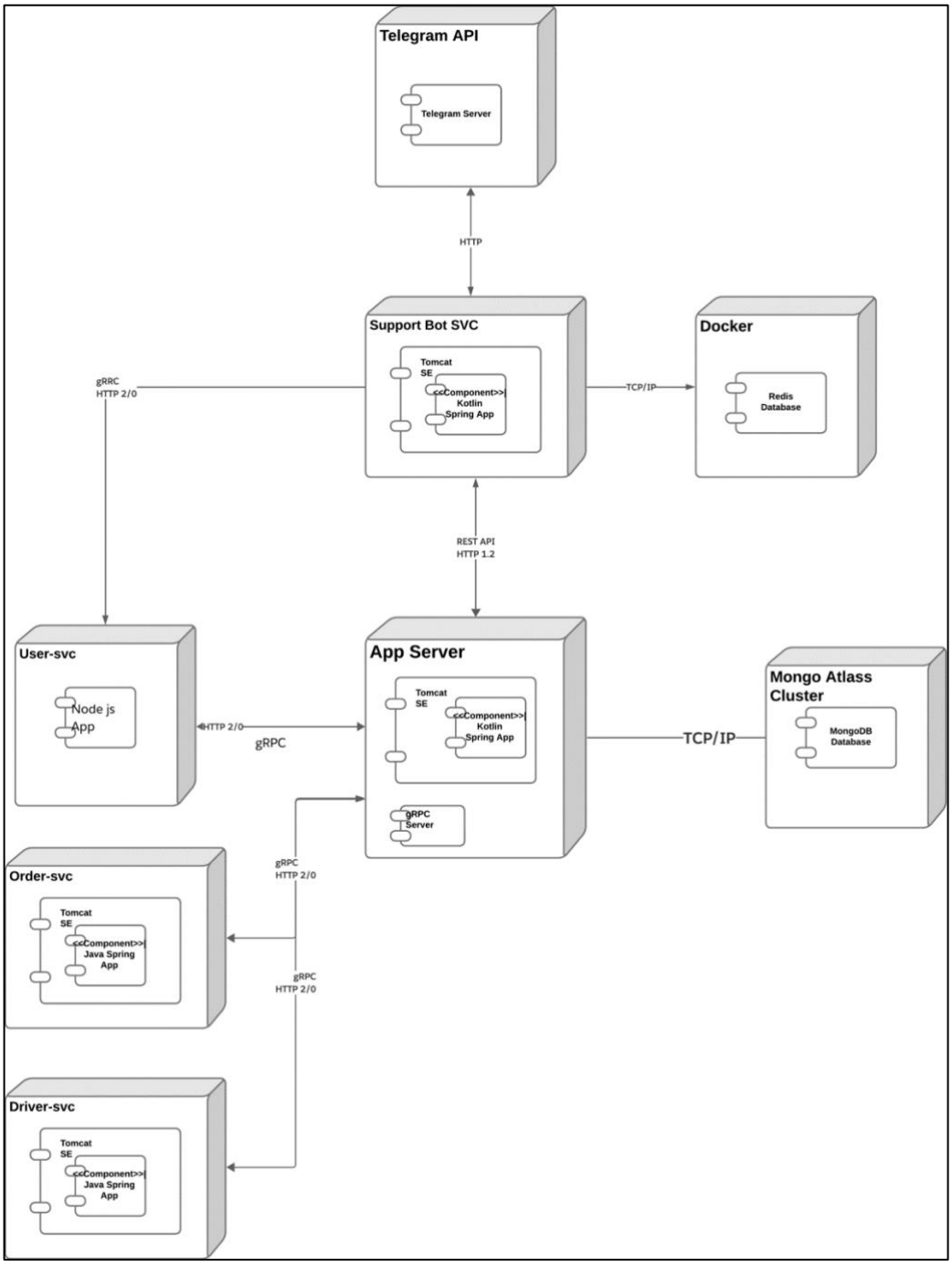


Рисунок 4.5 – Діаграма розгортання застосунку

Цей сервіс реалізований на Node.js і взаємодіє з основним сервером додатків через gRPC та HTTP/2.0, забезпечуючи зв'язок у реальному часі та швидкий обмін повідомленнями.

Order-svc:

Сервіс замовлень працює на Java Spring Application, який також використовує Tomcat Servlet Engine. Він взаємодіє з основним сервером додатків через gRPC та HTTP/2.0 для забезпечення надійного і швидкого обміну даними.

Driver-svc:

Сервіс водіїв, аналогічно до сервісу замовлень, реалізований на Java Spring Application з використанням Tomcat Servlet Engine і використовує gRPC та HTTP/2.0 для взаємодії з основним сервером додатків.

Admin Api Server:

Сервер додатків є центральним компонентом архітектури. Він реалізований на Kotlin Spring Application, що працює в середовищі Tomcat Servlet Engine. Сервер взаємодіє з усіма сервісами через gRPC та HTTP/2.0, а також з MongoDB Atlas Cluster через TCP/IP для зберігання і управління даними.

MongoDB Atlas Cluster:

MongoDB використовується як основна база даних для зберігання та управління даними. Її безсхемна структура дозволяє гнучко працювати з різними типами даних і структурами, що необхідні для операційної діяльності.

Ця архітектура забезпечує високий рівень масштабованості, надійності та ефективності обробки даних у реальному часі. Вона є критично важливою для безперебійної роботи сервісу підтримки бота та забезпечення високої якості взаємодії з користувачами через Telegram. Завдяки продуманій інтеграції компонентів, використанню сучасних технологій та протоколів зв'язку, система може легко адаптуватися до зростаючих потреб користувачів, забезпечуючи стабільність та швидкість обробки великих обсягів даних. Ця архітектура демонструє передовий підхід до розробки програмного забезпечення, що дозволяє досягти максимальної продуктивності та надійності в роботі з реальними користувачами.

3.3 Проєктування структури зберігання даних

Проектування структури зберігання даних для додатку мувінгових послуг є важливим аспектом архітектури системи, що забезпечує ефективне управління даними та їхню цілісність. Дизайн схеми, як показано на діаграмі "сутність-зв'язок"

(ERD) [1] (див. рис. 4.6), полегшує реалізацію бізнес-логіки та ефективно підтримує функціональність додатку.

Огляд ER – діаграми

Представлена ERD описує структурований підхід до роботи з даними додатку. Ось ключові сутності та їх взаємозв'язки:

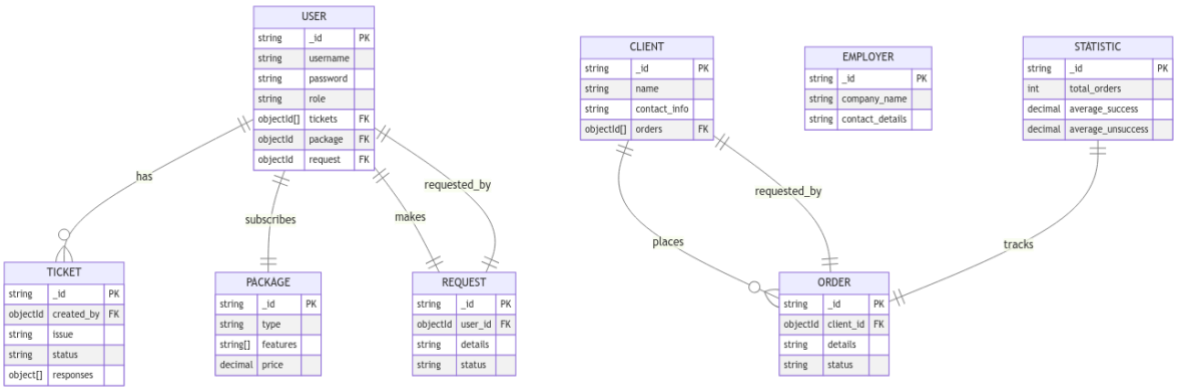


Рисунок 4.6 – ER-діагарама

КОРИСТУВАЧ

Атрибути: ідентифікатор (PK), ім'я користувача, пароль, роль, квитки (FK), пакет (FK), запит (FK)

Опис: Представляє користувачів системи, зберігає облікові дані для входу, відмінності ролей та посилання на пов'язані з ними квитки, пакети та запити.

КВИТОК

Атрибути: id (PK), created_by (FK), issue, status, responses

Опис: Керує запитами на підтримку або обслуговування, створеними користувачами, відстежуючи їхній прогрес і відповіді, пов'язані з кожним тікетом.

ПАКЕТ

Атрибути: id (PK), type, features[], price

Опис: Визначає різні пакети послуг, доступні користувачам, деталізуючи тип, включені функції та ціну кожного пакета.

ЗАПИТ

Атрибути: id (PK), user_id (FK), details, status

Опис: Фіксує конкретні запити користувачів, пов'язані з послугами, деталізуючи їх характер і поточний статус обробки.

КЛІЄНТ

Атрибути: id (PK), name, contact_info, orders (FK)

Опис: Представляє клієнтів, які розміщують замовлення, містить основну контактну інформацію та посилання на їхні відповідні замовлення.

РОБОТОДАВЕЦЬ

Атрибути: id (PK), назва_компанії, контактні_дані

Опис: Зберігає інформацію про роботодавців або суб'єктів господарювання, які користуються послугою, включаючи реквізити компанії та контактну інформацію.

ЗАМОВЛЕННЯ

Атрибути: id (PK), client_id (FK), details, status

Опис: Керує замовленнями, розміщеними клієнтами, записуючи конкретні деталі та статус кожного замовлення.

СТАТИСТИКА

Атрибути: id (PK), total_orders, average_success, average_unsuccess

Опис: Агрегує статистичні дані, пов'язані з виконанням замовлень, такі як загальна кількість замовлень та відсоток успішності.

Описи відношень

Користувачі мають Тікети, Пакети та Заявки: Вказує на право власності та підзвітність, гарантуючи, що всі дії, пов'язані з обслуговуванням, відстежуються по кожному користувачеві.

Клієнти розміщують замовлення: Відображає бізнес-транзакції, в яких клієнти замовляють послуги переїзду.

Роботодавці отримують замовлення: Показує, які роботодавці беруть участь у виконанні конкретних замовлень.

Статистика відстежує замовлення: Цей зв'язок агрегує та аналізує дані про замовлення для отримання значущої бізнес-інформації.

Безпека: Конфіденційна інформація, зокрема паролі та контактні дані, повинні бути зашифровані або хешовані для захисту даних користувачів.

3.4 Створення UI / UX або іншого дизайну системи

При розробці програмного забезпечення для мувінгових послуг інтеграція добре продуманого користувацького інтерфейсу (UI) та користувацького досвіду (UX) має вирішальне значення. Щоб розробники та кінцеві користувачі могли безперешкодно взаємодіяти з функціоналом програми, використовується Swagger

– широко використовуваний інструмент для проектування, побудови та документування RESTful API. У цьому розділі розглядається, як Swagger використовується для покращення UI/UX дизайну системи шляхом надання інтерактивного та зручного інтерфейсу для управління операціями API.

Специфікація OpenAPI забезпечує стандартний, мовно-діагностичний інтерфейс для RESTful API, який дозволяє як людям, так і комп'ютерам виявляти і розуміти можливості сервісу без доступу до вихідного коду, додаткової документації або перевірки мережевого трафіку. При правильному визначенні через OpenAPI, RESTful API описується у форматі, який є читабельним та інтерактивним.

Інструменти Swagger повністю сумісні зі специфікацією OpenAPI і надають потужний набір для розробки та інтеграції API.

Swagger – це програмний фреймворк з відкритим вихідним кодом, що підтримується великою екосистемою інструментів, які допомагають розробникам проектувати, створювати, документувати та використовувати RESTful веб-сервіси. Він пропонує веб-інтерфейс, який дозволяє як розробникам, так і кінцевим користувачам візуалізувати та взаємодіяти з ресурсами API, не маючи жодної логіки реалізації. На рисунку 4.7 зображено приклад використання інструменту Swagger для документації та тестування API, зокрема для аутентифікації користувача через відповідний endpoint.

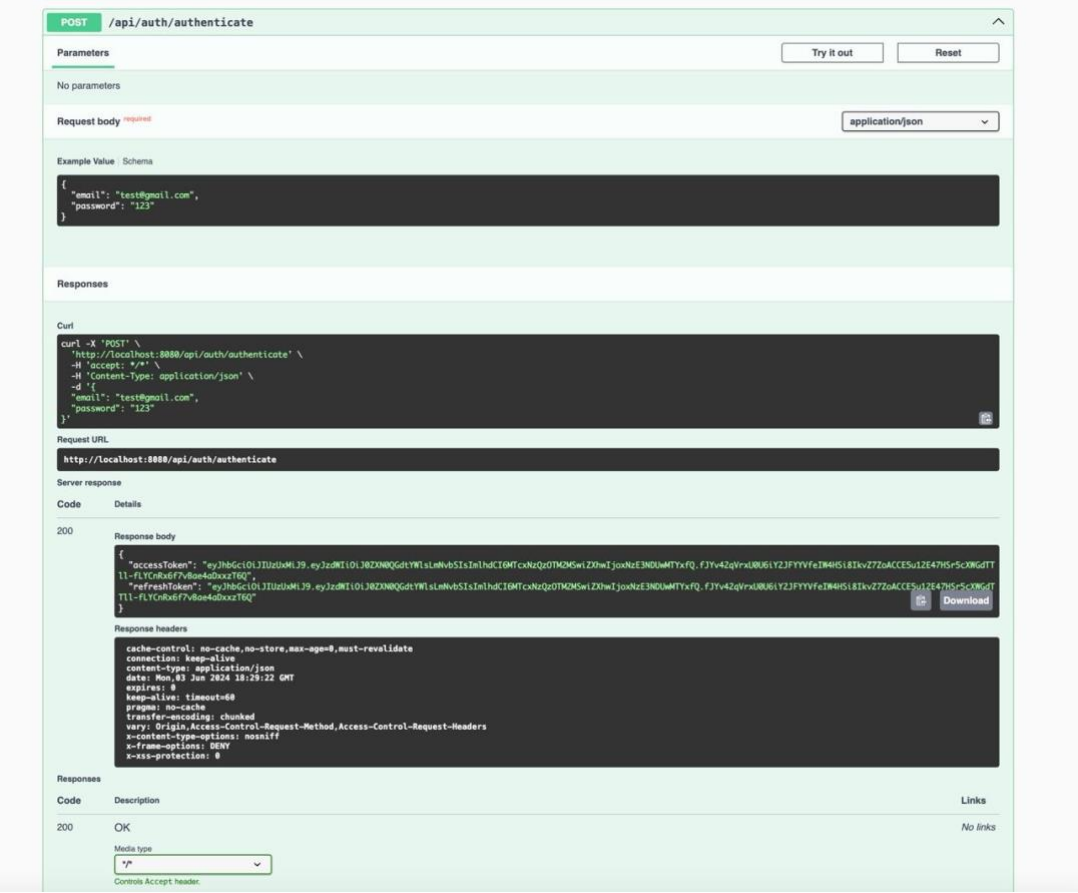


Рисунок 4.7 – Приклад використання інструменту Swagger

На рисунку Цей скріншот демонструє, як розробники можуть скористатися перевагами інтерактивного інтерфейсу Swagger для ефективного тестування і налагодження своїх API.

Swagger надає зручний і інтуїтивно зрозумілий інтерфейс для введення необхідних даних аутентифікації. Розробники можуть легко заповнити відповідні поля для введення email та паролю, а також відправити запит на сервер. Це значно спрощує процес взаємодії з API, оскільки не потрібно вручну формувати запити або використовувати окремі інструменти для тестування.

У відповідь на запит, Swagger відображає детальну інформацію про результати, включаючи коди статусу, заголовки відповіді та тіло відповіді. Це дозволяє розробникам миттєво бачити, чи був запит успішним, і якщо ні, які помилки виникли. Зокрема, на зображенні показано успішний результат аутентифікації з відповідним JSON-токеном доступу, що підтверджує правильність введених даних та коректну роботу API.

Крім того, Swagger генерує приклади curl-запитів, які можуть бути використані розробниками для подальшого автоматизованого тестування або

інтеграції у скрипти розгортання. Це робить процес розробки більш гнучким і зручним, дозволяючи зосередитися на логіці бізнес-процесів, а не на технічних деталях взаємодії з API.

Інструменти Swagger повністю сумісні зі специфікацією OpenAPI, що дозволяє забезпечити стандартизований підхід до документування та тестування RESTful API. Це є важливою перевагою при розробці масштабованих систем, де важливо підтримувати єдині стандарти якості та забезпечувати надійну взаємодію між різними компонентами.

Переваги використання Swagger:

– Інтерактивна документація: Swagger надає динамічно згенеровану інтерактивну документацію API, з якою користувачі можуть взаємодіяти безпосередньо через інтерфейс Swagger. Це дозволяє проводити тестування кінцевих точок API в режимі реального часу.

– Генерація клієнтського SDK: Swagger може автоматично генерувати клієнтські бібліотеки на декількох мовах, що прискорює розробку і тестування інтерфейсу.

– Узгодженість дизайну: Використання Swagger забезпечує послідовне застосування принципів RESTful на всіх кінцевих точках, сприяючи стандартизації та зменшенню помилок під час розробки.

– Безпека та авторизація: Swagger підтримує включення визначень безпеки в документацію, яка може визначати вимоги до ключів API, конфігурації OAuth та інші протоколи безпеки, необхідні для безпечної взаємодії з API.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Прийнятті рішення, під час розробки статистики

Під час розробки системи для мувінгових послуг особливу увагу було приділено збору та обробці статистичних даних, які дозволяють оцінювати продуктивність та ефективність роботи системи. Нижче наведено декілька найцікавіших та найважливіших статистик, реалізованих у нашій системі, разом із відповідними формулами та поясненнями.

4.1.1 Статистика кількості тикетів за статусом

Функція getTicketCountByStatus підраховує кількість тикетів у системі, згрупованих за їх статусом. Це дозволяє отримати уявлення про поточний стан тикетів, які знаходяться в обробці, завершені або вимагають додаткової уваги. Ця статистика є критично важливою для управління робочим процесом та пріоритизації завдань.

```
fun getTicketCountByStatus(): Map<SupportTicketStatus, Long> {
    return ticketRepository.findAll().groupingBy { it.status
}.eachCount().mapValues { it.value.toLong() }}
```

Ця функція допомагає відслідковувати кількість тикетів у різних станах, що дозволяє ефективніше управляти робочим процесом та пріоритизувати завдання.

4.1.2 Середній час вирішення тикетів

Функція getAverageResolutionTime обчислює середній час, необхідний для вирішення тикетів, що мають статус "RESOLVED". Цей показник допомагає оцінити, наскільки ефективно команда підтримки справляється з запитами користувачів.

Проведемо розрахунки за формулою 1.1:

$$\text{Average Resolution Time} = \frac{\sum \text{Resolution Time}}{\text{Number of Resolved Tickets}} \qquad (1.1)$$

```
fun getAverageResolutionTime(): Double {
    val resolvedTickets =
ticketRepository.findByStatus(SupportTicketStatus.RESOLVED)
    val totalResolutionTime = resolvedTickets.sumOf {
        Duration.between(it.createdAt, it.updatedAt).toMillis()
    }
    return if (resolvedTickets.isNotEmpty()) {
        totalResolutionTime.toDouble() / resolvedTickets.size
    } else {
        0.0
    }
}
```

Ця функція дозволяє відслідковувати ефективність команди підтримки, обчислюючи середній час вирішення тикетів.

4.1.3 Середній та стандартне відхилення часу вирішення тикетів

Функція getResolutionTimeWithStdDev обчислює середній час вирішення тикетів та стандартне відхилення цього показника. Середній час вирішення показує загальну тенденцію, а стандартне відхилення дозволяє оцінити варіативність цього показника.

```
fun getResolutionTimeWithStdDev(): Pair<Double, Double> {
    val                          resolvedTickets                          =
ticketRepository.findByStatus(SupportTicketStatus.RESOLVED)
    val resolutionTimes = resolvedTickets.map {
        Duration.between(it.createdAt, it.updatedAt).toMillis().toDouble()
    }

    val mean = if (resolutionTimes.isNotEmpty()) resolutionTimes.sum() /
resolutionTimes.size else 0.0
    val variance = if (resolutionTimes.isNotEmpty()) resolutionTimes.map {
(it - mean) * (it - mean) }
        .sum() / resolutionTimes.size else 0.0
    val stdDev = sqrt(variance)

    return Pair(mean, stdDev)
}
```

Ця функція дозволяє отримати уявлення про загальну тенденцію та варіативність часу вирішення тикетів.

4.1.4 Кореляція між пріоритетом тикету та кількістю повідомлень у ньому

Функція getCorrelationPriorityMessageCount обчислює кореляцію між пріоритетом тикету та кількістю повідомлень у ньому. Ця статистика допомагає зрозуміти, чи існує зв'язок між пріоритетом тикету та тим, скільки взаємодій відбувається між користувачами та підтримкою.

```
fun getCorrelationPriorityMessageCount(): Double {
    val tickets = ticketRepository.findAll()
    val priorities = tickets.map { it.priority.ordinal.toDouble() }
    val messageCounts = tickets.map { it.conversation.size.toDouble() }

    val meanPriority = priorities.average()
    val meanMessageCount = messageCounts.average()

    val covariance = priorities.zip(messageCounts).sumOf { (p, m) -> (p -
meanPriority) * (m - meanMessageCount) }
    val variancePriority = priorities.sumOf { (it - meanPriority) * (it -
meanPriority) }
    val varianceMessageCount = messageCounts.sumOf { (it - meanMessageCount)
* (it - meanMessageCount) }

    return if (variancePriority != 0.0 && varianceMessageCount != 0.0) {
        covariance / sqrt(variancePriority * varianceMessageCount)
    } else {
        0.0
    }
}
```

Таким чином, прийняті програмні рішення щодо збору та обробки статистичних даних дозволяють детально аналізувати ефективність роботи системи, виявляти проблемні зони та приймати обґрунтовані рішення для покращення роботи служби підтримки.

4.2 Прийняті рішення під час розробки мікро – сервіса для телеграм бота

4.2.1 Використання gRPC, Redis, RestTemplate, Telegram API, Spring

При розробці телеграм-бота для системи підтримки клієнтів було прийнято низку архітектурних рішень для забезпечення масштабованості, надійності та ефективності обробки запитів користувачів. Нижче наведено опис основних технологій та підходів, що були використані, а також їх роль у реалізації проекту.

4.2.2 gRPC

gRPC був обраний для міжсервісної комунікації завдяки його ефективності та підтримці численних мов програмування. Цей протокол на основі HTTP/2 забезпечує швидку та безпечну передачу даних між мікросервісами, зменшуючи затримки та підвищуючи пропускну здатність. Використання gRPC дозволяє досягти високої продуктивності при обробці великої кількості запитів.

Приклад реалізації gRPC клієнта може виглядати наступним чином:

```
// Приклад коду gRPC клієнта для взаємодії між сервісами
val channel = ManagedChannelBuilder.forAddress("localhost",
50051).usePlaintext().build()
val stub = MyServiceGrpc.newBlockingStub(channel)
// Виклик методу на сервері
val response =
stub.myMethod(MyRequest.newBuilder().setField("value").build())
```

Цей приклад демонструє, як gRPC клієнт може використовуватися для взаємодії між сервісами, забезпечуючи ефективне та надійне спілкування.

4.2.3 Redis

Redis використовується для кешування та швидкого доступу до часто запитуваних даних. Це дозволяє зменшити навантаження на базу даних та прискорити обробку запитів. Кешування у Redis особливо ефективне для зберігання сесій користувачів, тимчасових даних та результатів частих запитів.

Приклад використання Redis у Spring:

```
@Service
class CacheService(private val redisTemplate: RedisTemplate<String, Any>) {

    fun saveToCache(key: String, value: Any) {
        redisTemplate.opsForValue().set(key, value)
    }

    fun getFromCache(key: String): Any? {
        return redisTemplate.opsForValue().get(key)
    }
}
```

Приклад використання Redis у Spring демонструє, як можна використовувати Redis для кешування даних, що дозволяє зменшити навантаження на базу даних та прискорити обробку запитів.

4.2.4 RestTemplate

RestTemplate є частиною Spring Framework і використовується для виклику RESTful веб-сервісів. У нашому проекті RestTemplate застосовується для взаємодії з зовнішніми API, такими як Telegram API. Це дозволяє легко інтегрувати зовнішні сервіси та обробляти HTTP запити та відповіді.

Приклад використання RestTemplate:

```
@Service
class TelegramService(private val restTemplate: RestTemplate) {

    fun sendMessage(chatId: String, text: String) {
        val url = "https://api.telegram.org/bot<token>/sendMessage"
        val request = mapOf("chat_id" to chatId, "text" to text)
        restTemplate.postForObject(url, request, String::class.java)
    }
}
```

Приклад використання RestTemplate демонструє, як можна інтегрувати зовнішні сервіси, такі як Telegram API, для обробки HTTP запитів та відповідей.

4.2.5 Telegram API

Telegram API використовується для інтеграції з Telegram платформою, що дозволяє нашому боту взаємодіяти з користувачами. Використання Telegram API забезпечує обробку повідомлень, відправку відповідей та виконання інших дій, необхідних для роботи бота.

Приклад обробки повідомлень за допомогою Telegram API:

```
@Controller
@RequestMapping("/webhook")
class TelegramWebhookController(private val telegramService:
TelegramService) {

    @PostMapping
    fun onUpdate(@RequestBody update: Update) {
        val message = update.message?.text
        if (message != null) {
```

```
            telegramService.sendMessage(update.message.chat.id.toString(),
"Received: $message")
        }
    }
}
```

Приклад обробки повідомлень за допомогою Telegram API демонструє, як можна обробляти вхідні повідомлення та відправляти відповіді користувачам, використовуючи інтеграцію з Telegram платформою.

4.2.6 Spring

Spring Framework є основою нашого проекту, забезпечуючи інверсію керування (IoC), залежність ін’єкції (DI) та інші функціональні можливості, що спрощують розробку та підтримку застосунку. Використання Spring дозволяє легко керувати конфігураціями, обробляти веб-запити, взаємодіяти з базами даних та іншими сервісами.

Приклад налаштування основного класу Spring Boot застосунку:

```
@SpringBootApplication
class SupportBotApplication
fun main(args: Array<String>) {
    runApplication<SupportBotApplication>(*args)
}
```

Прийняті архітектурні рішення дозволяють створити масштабовану, надійну та ефективну систему для обробки запитів користувачів через Telegram бота. Використання таких технологій, як gRPC, Redis, RestTemplate, Telegram API та Spring, забезпечує високу продуктивність та швидкість реагування на запити користувачів, що є критично важливим для успіху будь-якої сучасної сервісної платформи.

4.3 Прийняті рішення під час розробки статистики замовлень

При розробці системи для мувінгових послуг значну увагу було приділено збору та аналізу статистичних даних про замовлення. Це дозволяє не лише оцінювати ефективність роботи системи, а й виявляти тенденції та кореляції, що можуть впливати на бізнес-процеси. Нижче наведено декілька найцікавіших та

найважливіших статистик, реалізованих у нашій системі, разом із відповідними формулами та поясненнями.

4.3.1 Середній час доставки

Функція getAverageDeliveryTime обчислює середній час доставки для завершених замовлень. Цей показник допомагає оцінити загальну ефективність логістики та виявити можливості для оптимізації процесів доставки.

```
fun getAverageDeliveryTime(): Double {
    val orders = fetchCompletedOrders()
    val totalDuration = orders.sumOf { Duration.between(it.departureTime,
it.destinationTime).toMillis() }
    return totalDuration.toDouble() / orders.size
}
```

Ця функція дозволяє відслідковувати середній час доставки завершених замовлень для оцінки ефективності логістики.

4.3.2 Кореляція між вагою вантажу та часом доставки

Функція getCorrelationBetweenWeightAndDeliveryTime обчислює кореляцію між вагою вантажу та часом доставки для завершених замовлень. Ця статистика допомагає зрозуміти, чи впливає вага вантажу на час доставки. Висока кореляція може свідчити про те, що важчі вантажі доставляються довше, що може вимагати додаткових ресурсів або оптимізації процесів.

Кореляційний коефіцієнт r може приймати значення від -1 до 1. Значення, близьке до 1, вказує на сильну позитивну кореляцію, близьке до -1 - на сильну негативну кореляцію, а значення, близьке до 0, свідчить про відсутність кореляції.

```
fun getCorrelationBetweenWeightAndDeliveryTime(): Double {
    val orders = fetchCompletedOrders()
    val n = orders.size
    val sumWeight = orders.sumOf { it.cargoWeight }
    val sumTime = orders.sumOf { Duration.between(it.departureTime,
it.destinationTime).toMillis() }
    val sumWeightTime =
        orders.sumOf { it.cargoWeight * Duration.between(it.departureTime,
it.destinationTime).toMillis() }
    val sumWeightSquare = orders.sumOf { it.cargoWeight.pow(2) }
```

```
    val sumTimeSquare =
        orders.sumOf          {          Duration.between(it.departureTime,
it.destinationTime).toMillis().toDouble().pow(2) }

    val numerator = n * sumWeightTime - sumWeight * sumTime
    val denominator =
        sqrt((n * sumWeightSquare - sumWeight.pow(2)) * (n * sumTimeSquare
- sumTime.toDouble().pow(2)))

    return if (denominator == 0.0) 0.0 else numerator / denominator
}
```

Кореляційний коефіцієнт r може приймати значення від -1 до 1. Значення, близьке до 1, вказує на сильну позитивну кореляцію, близьке до -1 на сильну негативну кореляцію, а значення, близьке до 0, свідчить про відсутність кореляції.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення є ключовим етапом у процесі розробки, що дозволяє забезпечити високу якість продукту та відповідність вимогам користувачів. У цьому розділі ми розглянемо підходи, що були використані для тестування телеграм-бота для системи підтримки клієнтів, включаючи автоматизоване тестування, ручне тестування, тестування продуктивності та безпеки.

5.1 Підходи до тестування

Під час тестування програмного забезпечення ми використовували різні підходи, зокрема:

1. Автоматизоване тестування
2. Ручне тестування

Кожен із цих підходів забезпечує комплексне покриття тестами, що дозволяє виявити більшість помилок на різних етапах розробки. Однак у даному проекті ми зосередилися переважно на автоматизованому тестуванні.

5.2 Автоматизоване тестування

Автоматизоване тестування є основним інструментом для забезпечення стабільності та якості програмного забезпечення. У нашому проекті ми використовували такі види автоматизованого тестування:

5.2.1 Юніт-тестування (Unit Testing)

У нашому проекті ми використовували юніт-тестування як основний метод автоматизованого тестування. Юніт-тести дозволяють перевіряти окремі модулі або компоненти системи. Для цього ми використовували фреймворк JUnit, який забезпечує зручний спосіб написання та виконання тестів. Наразі покриття тестами становить 40%, що дозволяє виявляти більшість критичних помилок та забезпечує базовий рівень якості програмного забезпечення.

Ось приклад юніт-тестів для класу TicketService:

```
    @Test
    fun `should create ticket successfully`() {
        val ticket = Ticket(id = ObjectId.get().toString(), title = "Test
Ticket")
        `when`(ticketRepository.save(ticket)).thenReturn(ticket)

        val createdTicket = ticketService.createTicket(ticket)

        assertNotNull(createdTicket)
        assertEquals(ticket.id, createdTicket.id)
        verify(ticketRepository, times(1)).save(ticket)
    }

    @Test
    fun `should throw exception when ticket not found`() {
        val id = ObjectId.get().toString()

`when`(ticketRepository.findById(ObjectId(id))).thenReturn(Optional.empty()
)

        val exception = assertThrows(TicketNotFoundException::class.java) {
            ticketService.getTicketById(id)
        }

        assertEquals("Ticket not found with id: $id", exception.message)
        verify(ticketRepository, times(1)).findById(ObjectId(id))
    }
```

Ці юніт-тести перевіряють ключові функції класу `TicketService`: створення тикету та отримання тикету за його ідентифікатором. Перший тест перевіряє, чи створюється тикет успішно і чи повертається коректний результат, тоді як другий тест перевіряє, чи генерується виняток, коли тикет не знайдено.

5.2.2 Покриття тестами та його значення

Покриття тестами становить 40%, що є достатнім для виявлення більшості критичних помилок у системі. Високе покриття тестами є важливим показником якості програмного забезпечення, оскільки дозволяє забезпечити надійність і стабільність роботи системи. Однак, варто зазначити, що навіть високе покриття тестами не гарантує відсутність усіх помилок, тому важливо також використовувати інші методи тестування, такі як інтеграційне та функціональне тестування.

5.2.3 Переваги автоматизованого тестування

Автоматизоване тестування має численні переваги, включаючи:

– Швидкість та ефективність: Автоматизовані тести можуть бути виконані швидше, ніж ручні тести, що дозволяє зекономити час та ресурси.

– Повторюваність: Тести можуть бути виконані стільки разів, скільки потрібно, без змін у результатах, що дозволяє забезпечити стабільність системи.

– Раннє виявлення помилок: Автоматизовані тести можуть бути виконані на ранніх етапах розробки, що дозволяє виявити помилки до того, як вони вплинуть на інші частини системи.

– Підтримка безперервної інтеграції (CI): Автоматизоване тестування є невід'ємною частиною процесу безперервної інтеграції, що дозволяє виявляти та виправляти помилки на кожному етапі розробки.

ВИСНОВКИ

Протягом розробки кваліфікаційної роботи було створено комплексне програмне забезпечення для адаптованого сервісу мувінгу, спрямоване на ринок України. На початковому етапі проекту було здійснено глибокий аналіз предметної області, який дозволив ідентифікувати ключові виклики і потреби сучасного сервісу перевезень. Визначення цих потреб дало можливість точно окреслити основні функції та вимоги до програмного забезпечення, які були реалізовані на наступних етапах розробки.

За допомогою сучасних технологій та підходів, таких як Kotlin, Spring Boot, а також використанням архітектурних рішень на кшталт DDD і гексагональної архітектури, було створено масштабоване та ефективне рішення. Система інтегрована з засобами Swagger та OpenAPI для забезпечення прозорості, гнучкості інтерфейсів і легкості у використанні API.

Завдяки грамотній реалізації проекту, програмний комплекс демонструє високу продуктивність та надійність, відповідає сучасним вимогам ринку вантажних перевезень і забезпечує ефективний контроль за всіма аспектами перевезення вантажів. Система не тільки спрощує процеси управління та виконання замовлень, але й пропонує розширені можливості для моніторингу і аналізу робочих процесів, що є ключовим аспектом для підвищення ефективності та задоволення клієнтів.

Загалом, розроблене програмне забезпечення демонструє здатність адаптуватися до змінних умов ринку і потреб користувачів, надаючи стабільну та надійну платформу для управління перевезеннями і логістикою в Україні.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Thomas M. Connolly, Carolyn E. Begg. Database Systems: A Practical Approach to Design, Implementation and Management: 5th Edition. USA: Pearson, 2009. 1400 p.

2. Martin, Robert C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2017.

3. Fowler, Martin. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002.

4. Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1994.

5. Sommerville, Ian. Software Engineering. 10th Edition. Pearson, 2015. 816 p.

6. Bloch, Joshua. Effective Java. 3rd Edition. Addison-Wesley Professional, 2018. 416 p.

7. Pressman, Roger S. Software Engineering: A Practitioner's Approach. 8th Edition. McGraw-Hill Education, 2014. 944 p.

8. Larman, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3rd Edition. Prentice Hall, 2004. 736 p.

9. Stevens, W. P., Myers, G. J., and Constantine, L. L. Structured Design. Classic edition. Pearson, 1997. 480 p.

10. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. 3rd Edition. MIT Press, 2009. 1312 p.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

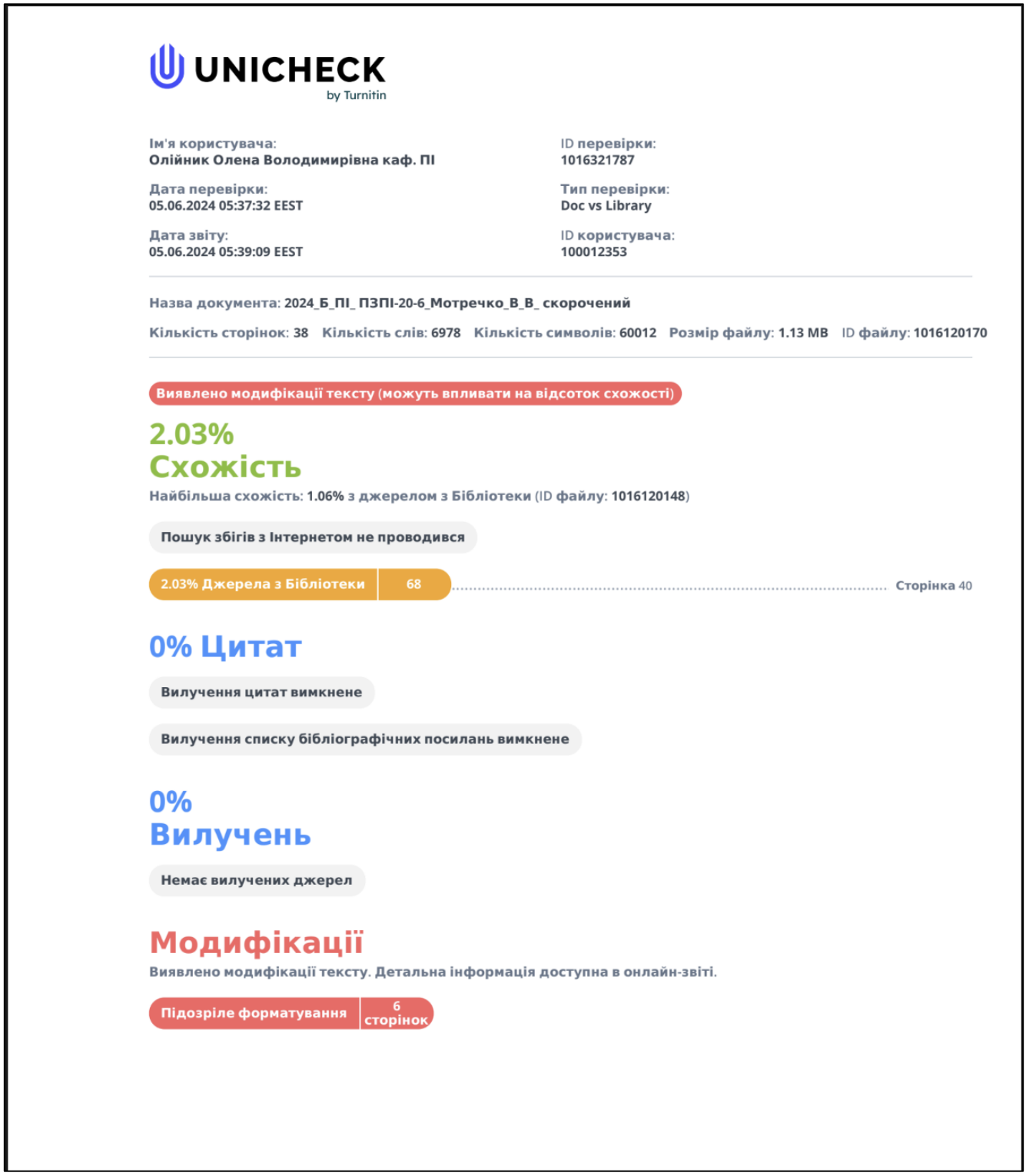


Рисунок А.1 – Результаті перевірки на унікальність тексту в базі ХНУРЕ

ДОДАТОК Б

Слайди презентації

Рисунок Б.1 – Слайд 1

Рисунок Б.2 – Слайд 2

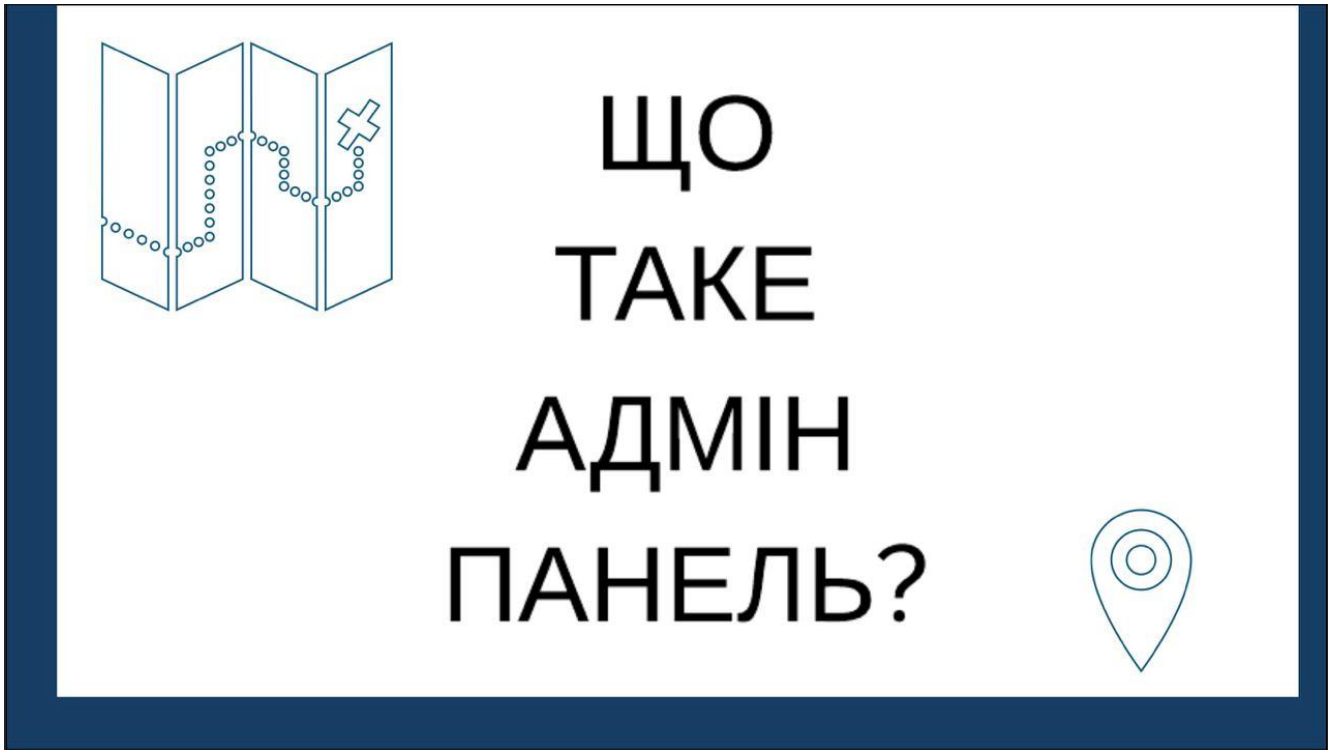


Рисунок Б.3 – Слайд 3

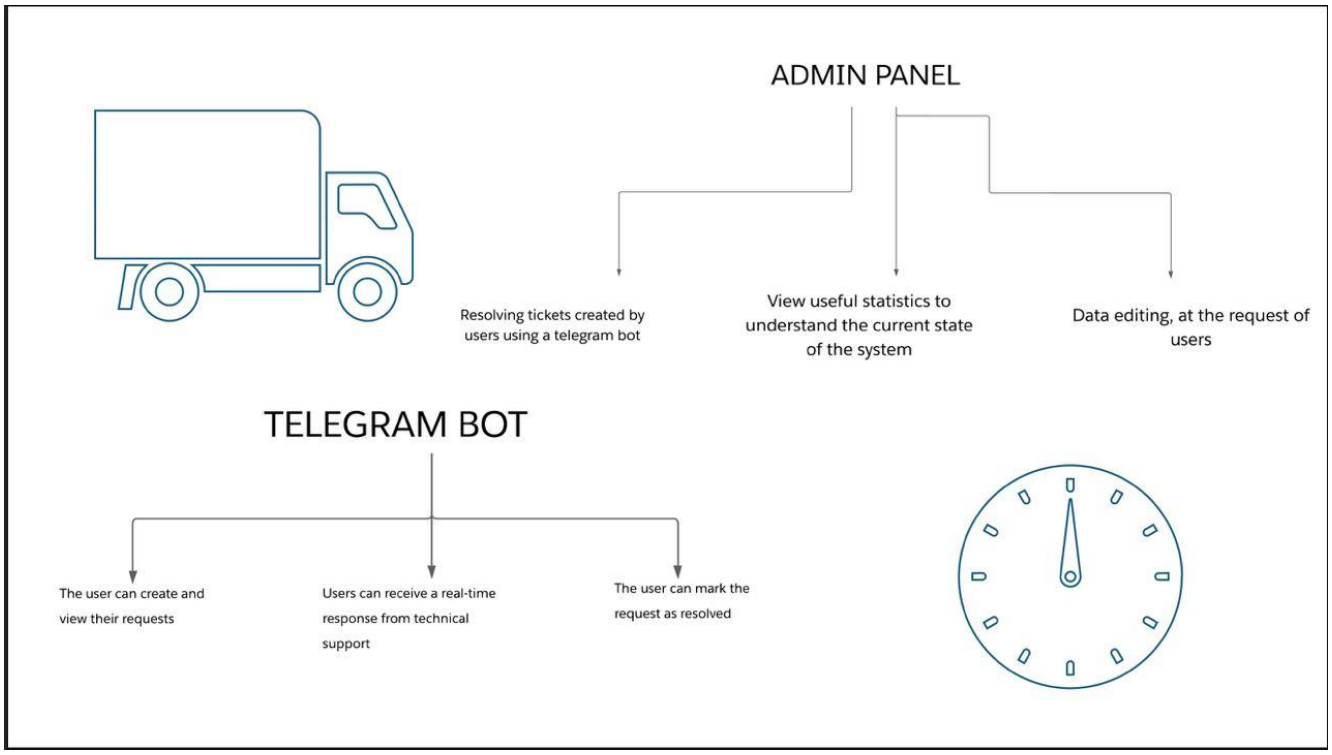


Рисунок Б.4 – Слайд 4

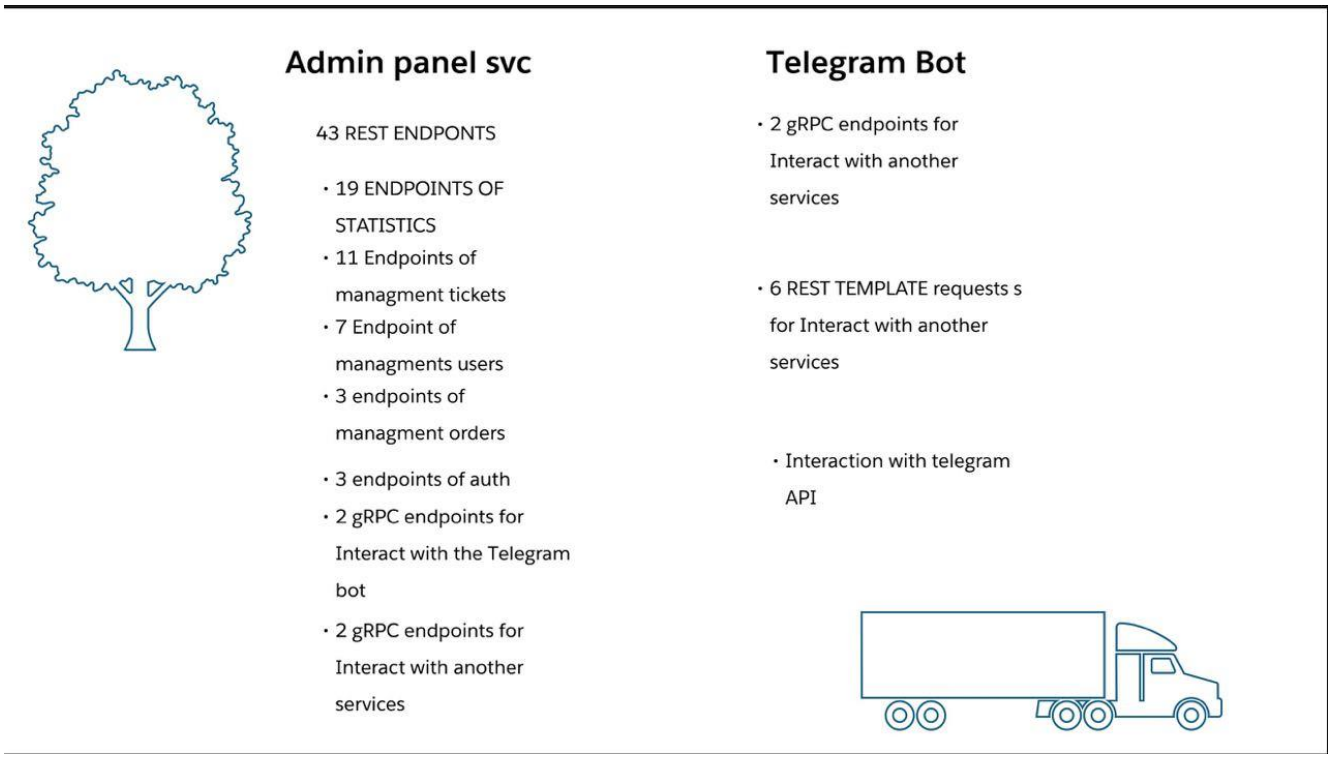


Рисунок Б.5 – Слайд 5

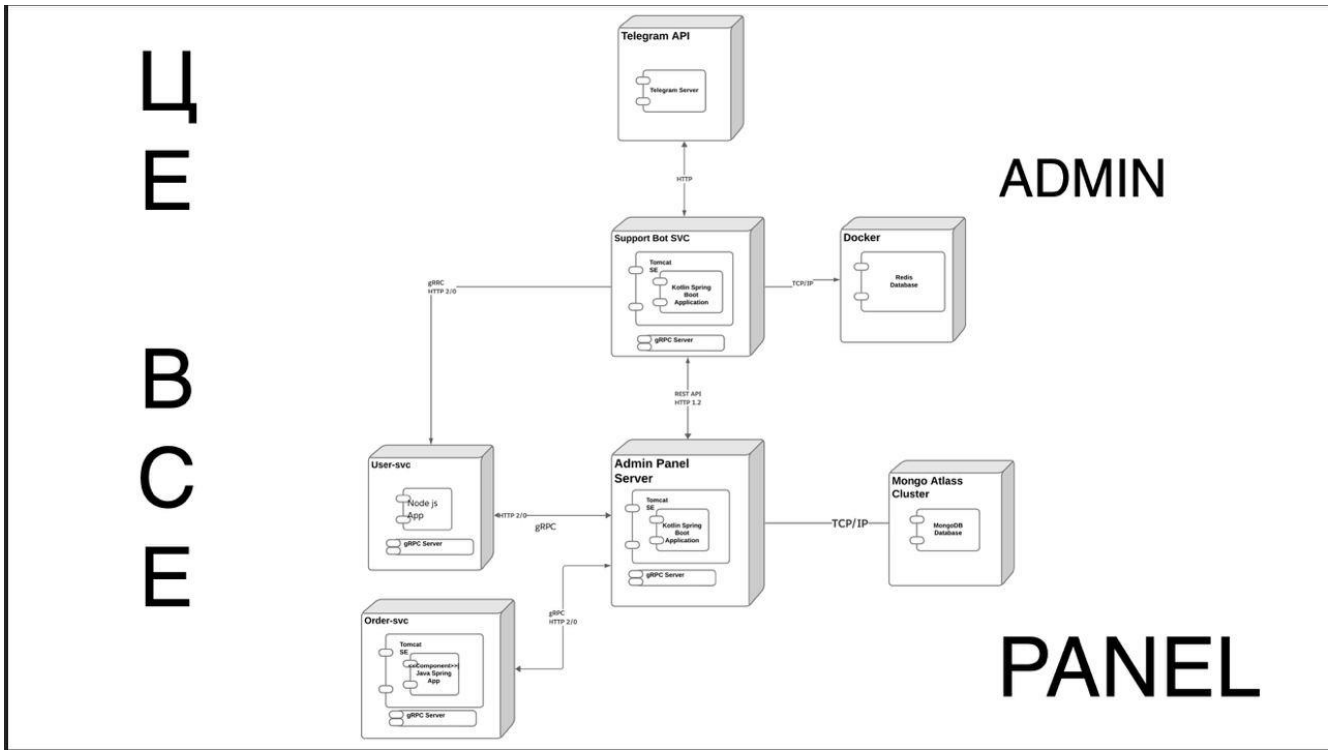


Рисунок Б.6 – Слайд 6

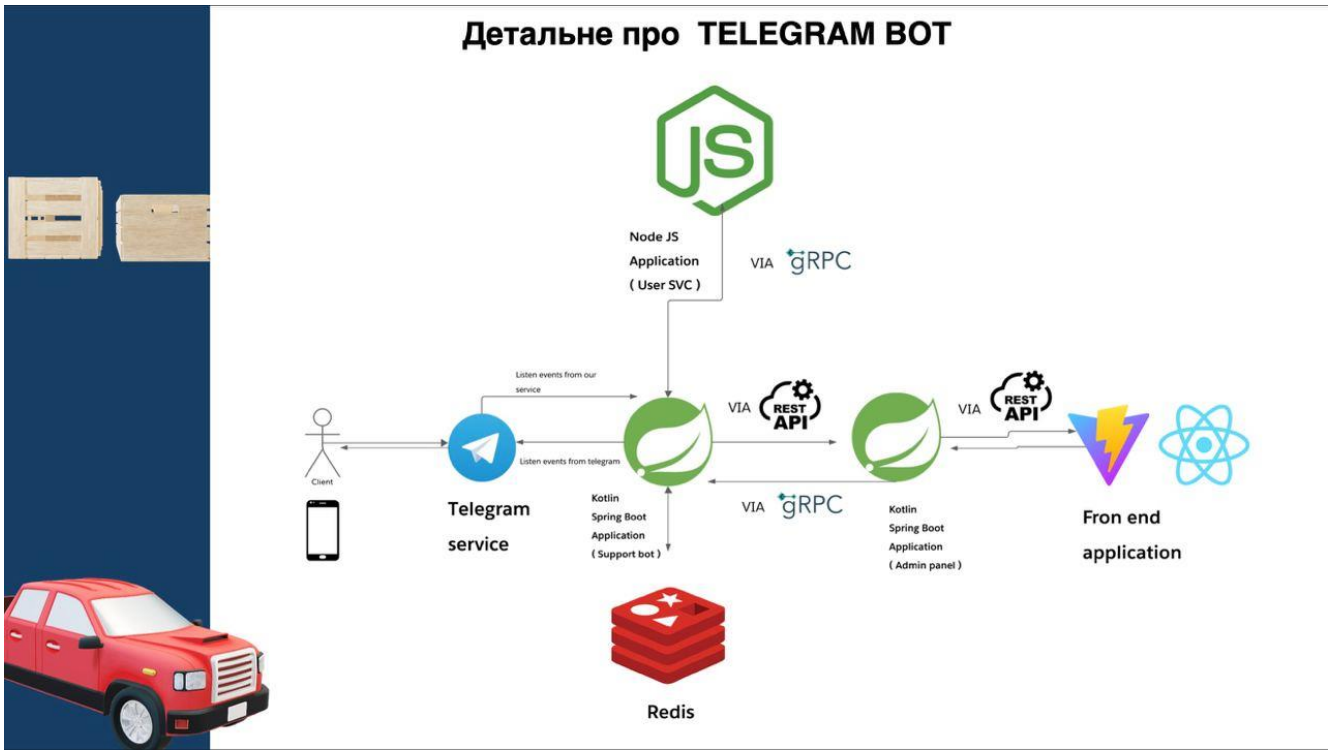


Рисунок Б.7 – Слайд 7

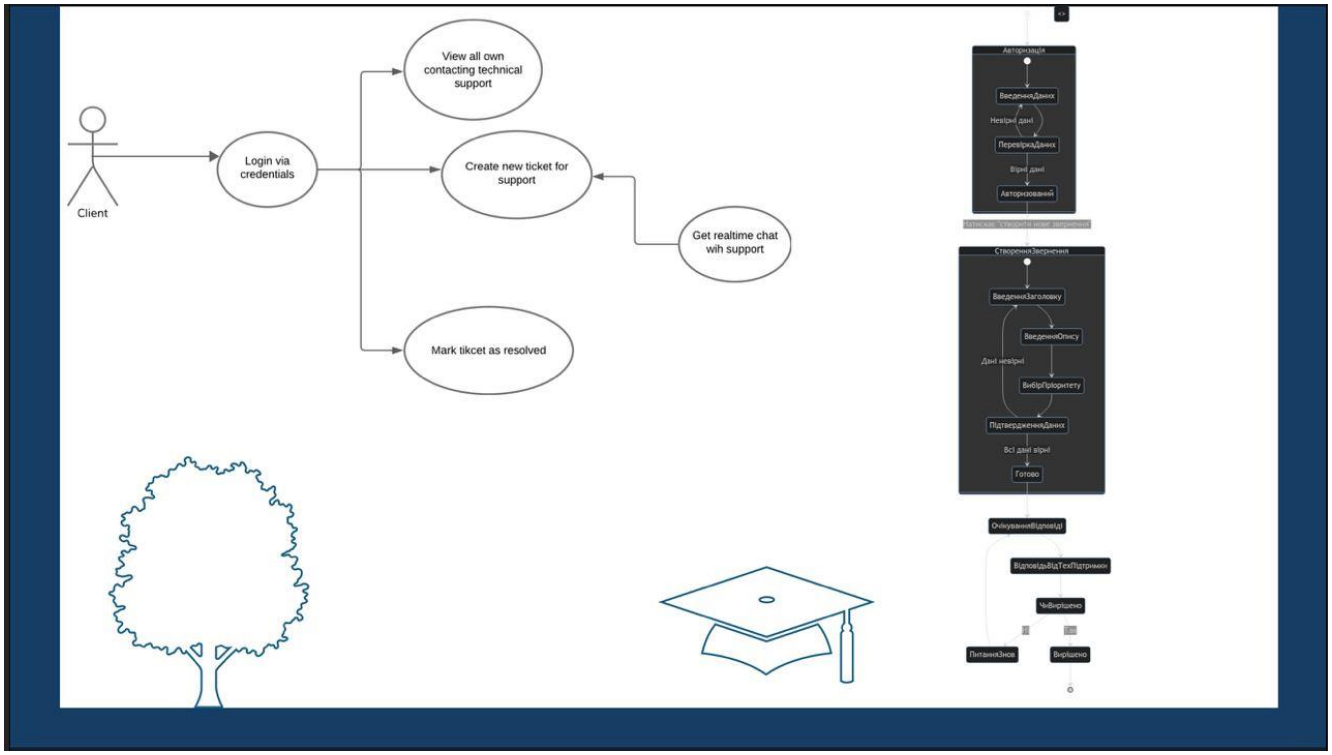


Рисунок Б.8 – Слайд 8

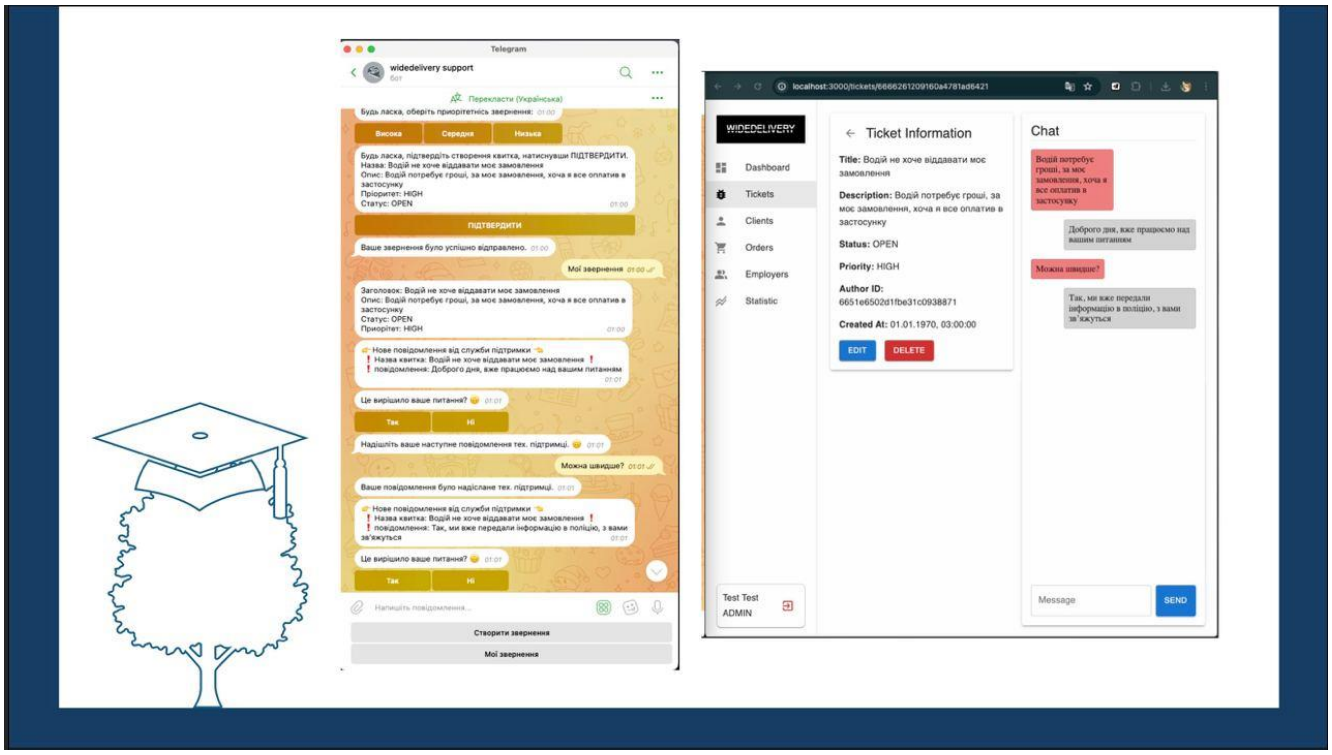


Рисунок Б.9 – Слайд 9

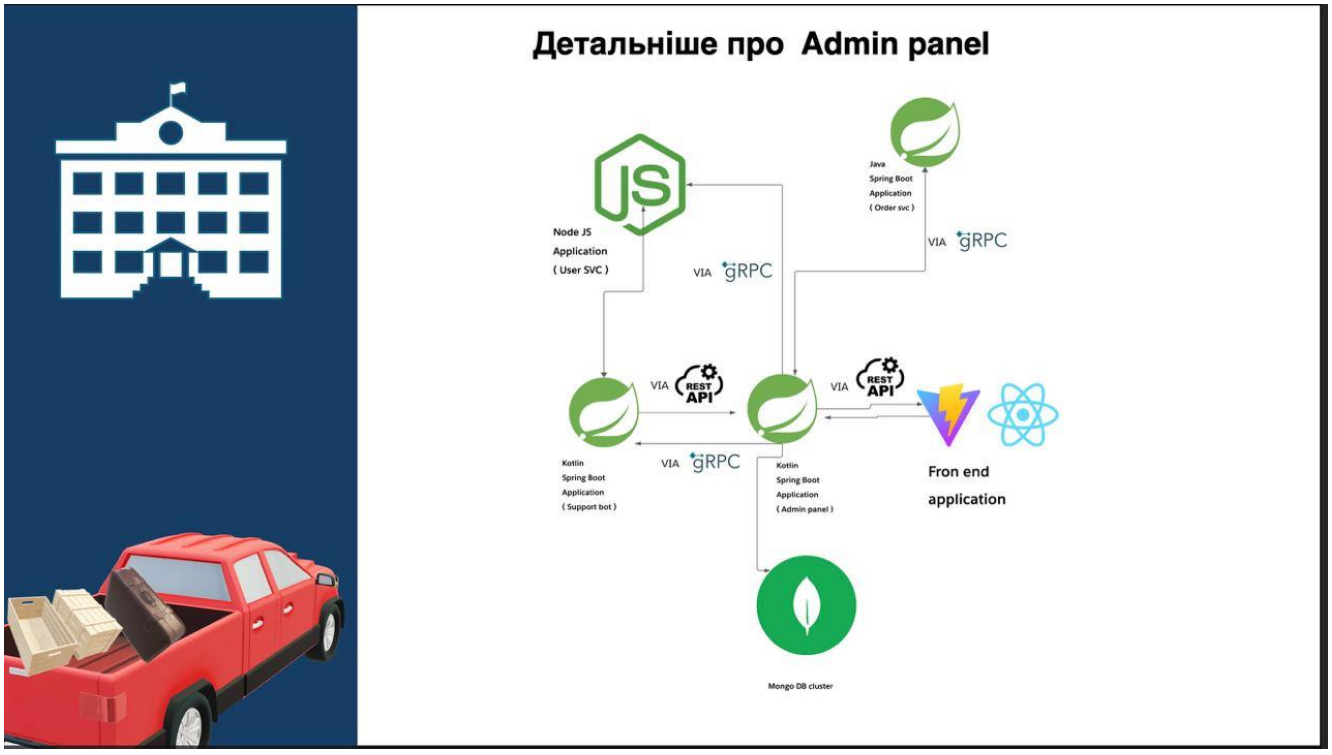


Рисунок Б.10 – Слайд 10

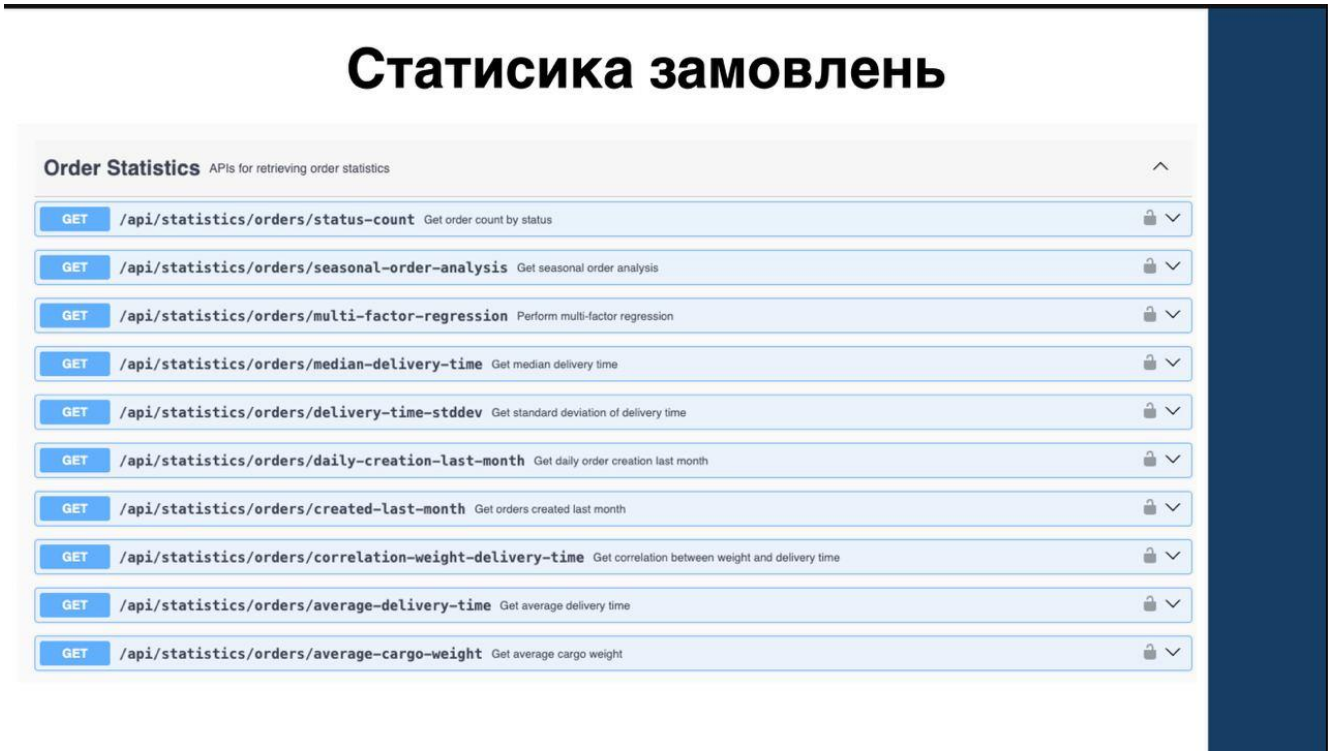


Рисунок Б.11 – Слайд 11

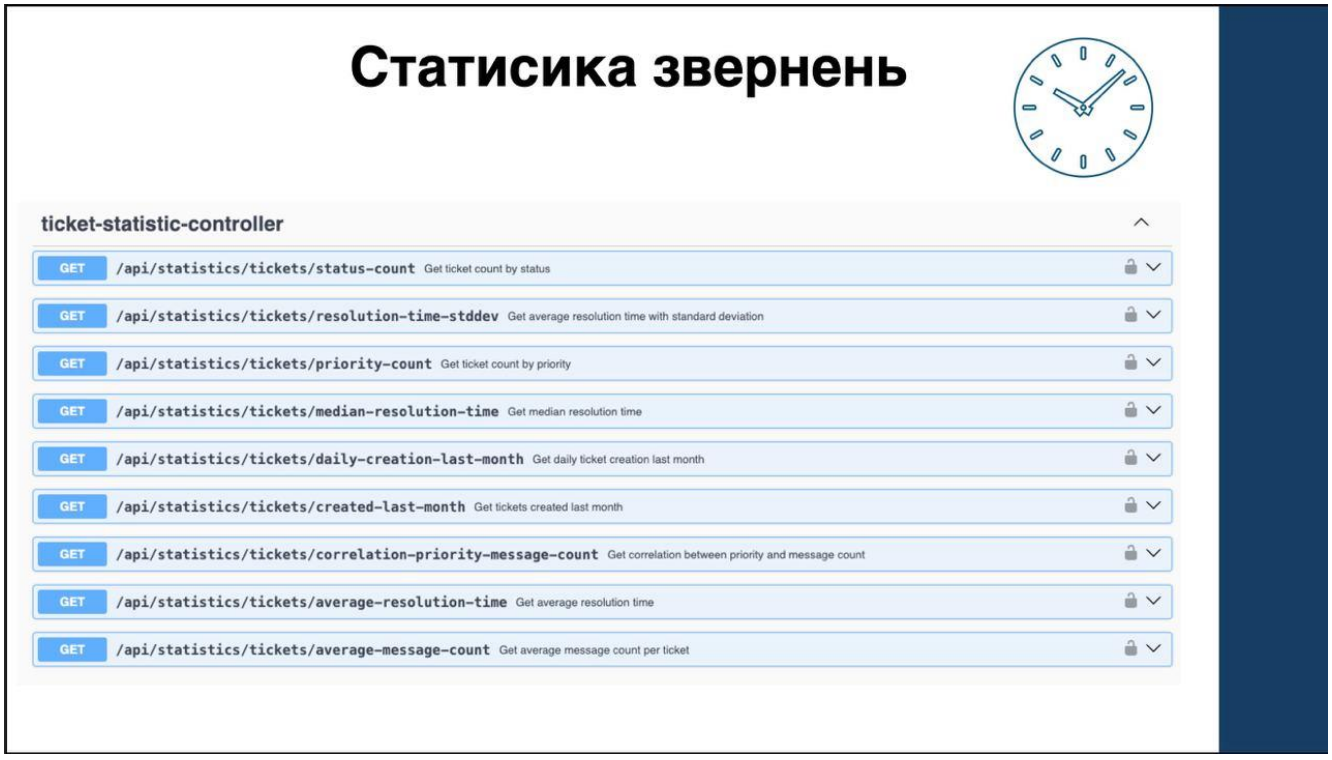


Рисунок Б.12 – Слайд 12

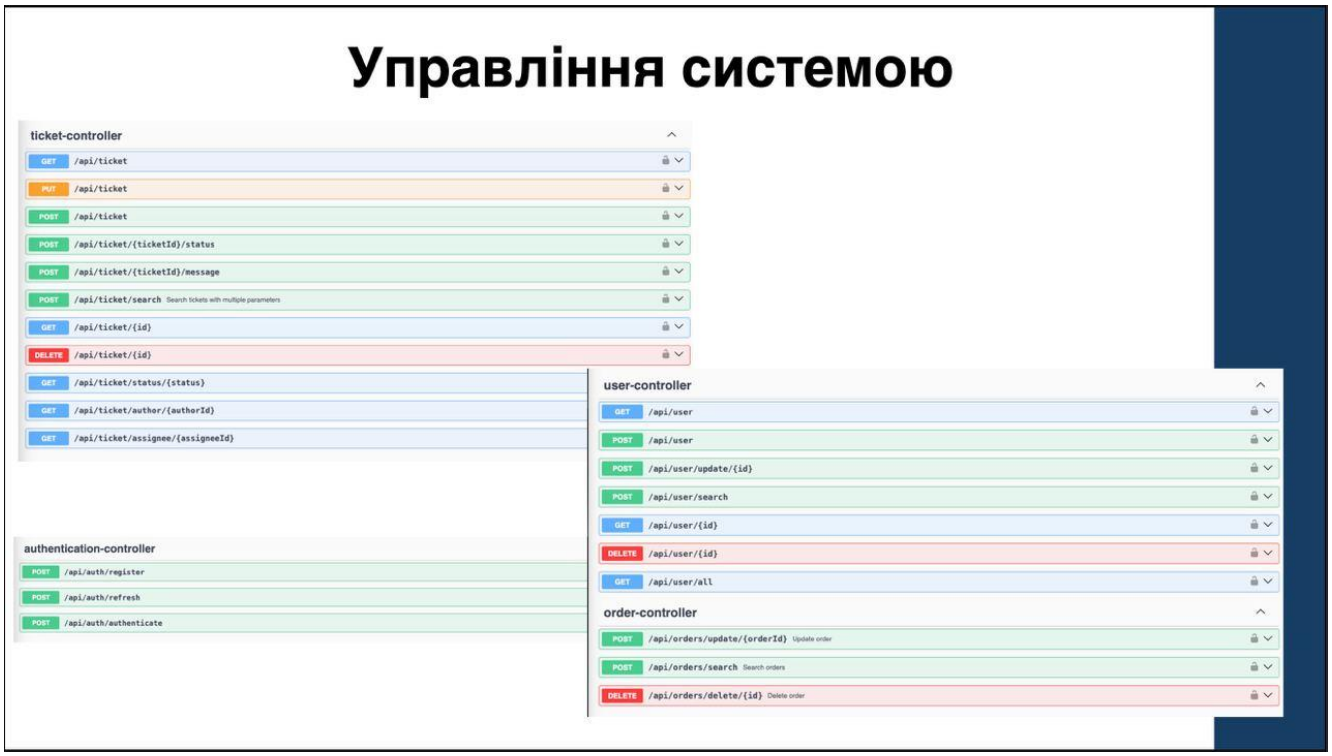


Рисунок Б.13 – Слайд 13

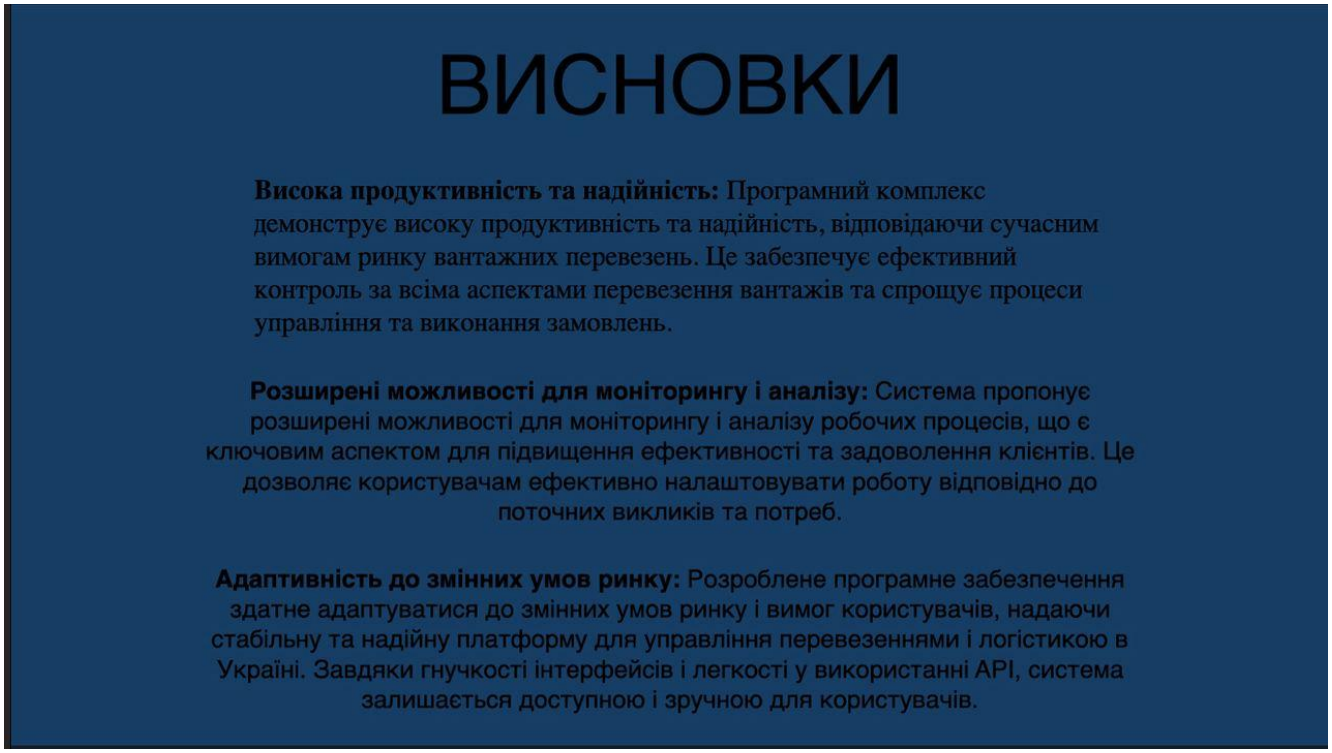
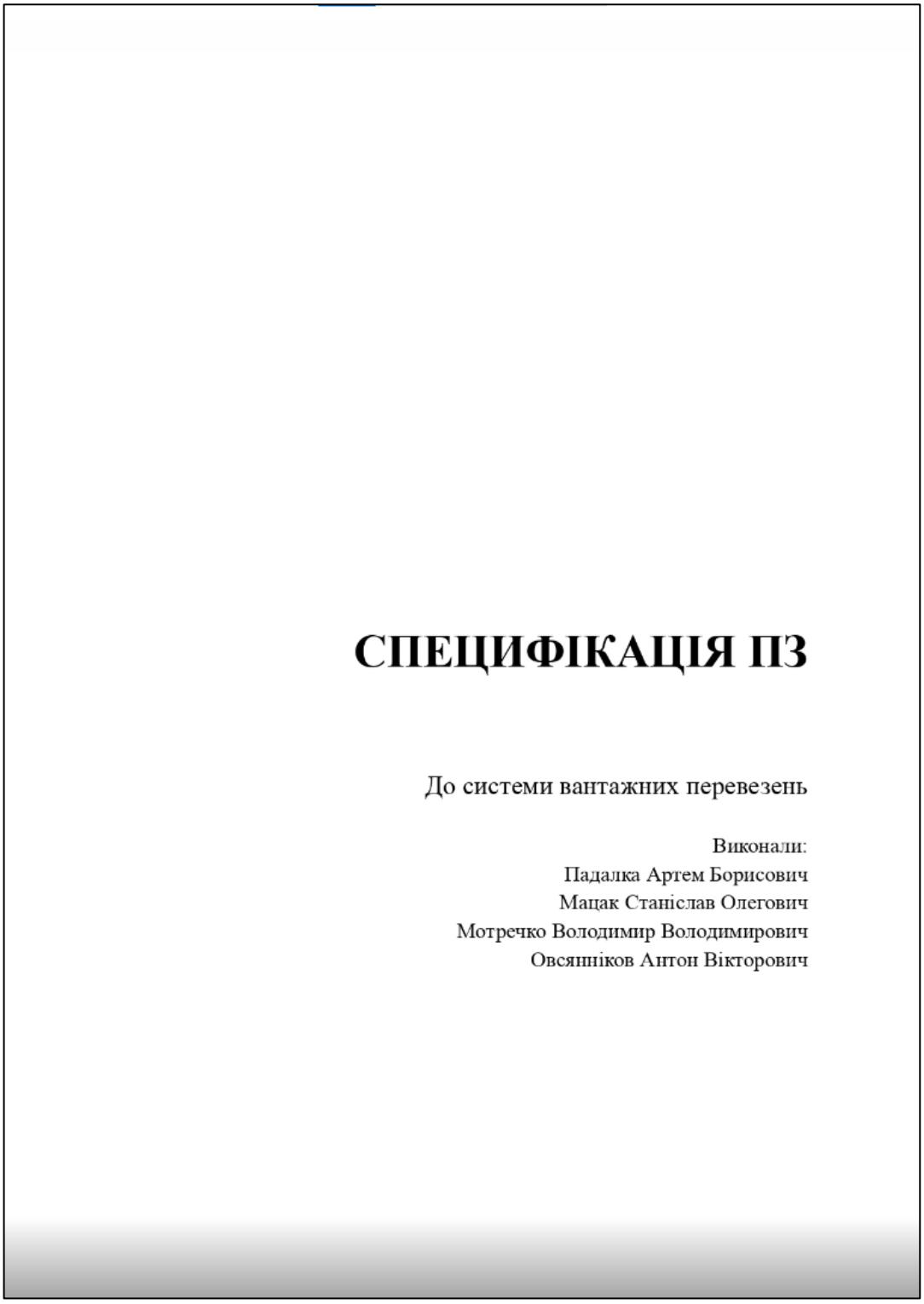


Рисунок Б.14 – Слайд 14

ДОДАТОК В

SRS-документ



СПЕЦИФІКАЦІЯ ПЗ

До системи вантажних перевезень

Виконали:
Падалка Артем Борисович
Мацак Станіслав Олегович
Мотречко Володимир Володимирович
Овсяніков Антон Вікторович

Рисунок В.1 – Сторінка 1 SRS-документу

Специфікація ПЗ

1. Вступ

1.1 Огляд продукту

Система вантажних перевезень, що ми розробляємо, є інноваційним рішенням для організації доставки вантажів в межах міста або між містами. Концепція системи базується на моделі Uber (офлайн-послуги, доступні відразу після оплати з мобільного додатку), але з деякими відмінностями, пов'язаними з особливостями вантажних перевезень.

Система повинна надати клієнтам зручний та ефективний спосіб замовлення вантажного перевезення "не відриваючись від екрану мобільного пристрою", де клієнт мусить вказати габарити, тип вантажу та час і місце доставки. Водії, які можуть бути як представниками компаній, що займаються вантажними перевезеннями, так і звичайними користувачами, мають можливість приймати замовлення та виконувати доставку.

Продукт спрямований на модернізацію традиційних підходів до транспортної логістики, забезпечуючи зручні, швидкі та вартісно ефективні рішення для мувінгових компаній та кінцевих користувачів.

Система дозволяє користувачам замовляти перевезення вантажів через мобільний застосунок, пропонуючи опції для негайних та запланованих доставок. Водії можуть реєструватися у системі, вказуючи свої маршрути та доступність, що дозволяє системі автоматично спарювати замовлення з відповідними водіями, оптимізуючи навантаження та мінімізуючи порожні поїздки.

Ключовим нововведенням є впровадження моделі спільної участі, де користувачі можуть ділитися ресурсом своїх транспортних засобів (зокрема, вантажним простором автомобілів), для збільшення загальної ефективності і зниження витрат. Система також надає докладну інформацію про статус доставки, включаючи трекінг у реальному часі,

Рисунок В.2 – Сторінка 2 SRS-документу

оцінки та відгуки, що сприяє підвищенню довіри та задоволеності клієнтів.

Це програмне забезпечення має на меті не тільки підвищити ефективність логістичних операцій, але й зробити процес перевезення більш прозорим та доступним для всіх учасників ринку.

Система передбачає два варіанти доставки: “онлайн” (тобто, максимально швидко) або з запланованим часом.

Для водіїв передбачено можливість вказувати радіус пошуку, якщо вони готові приймати замовлення прямо зараз, або вказувати маршрут, за яким вони будуть їхати в майбутньому. Система автоматично підбирає замовлення, що проходять в межах радіусу або вздовж маршруту, та повідомляє клієнта про можливі варіанти доставки. Після прийняття замовлення водієм, клієнт оплачує вартість доставки та очікує, що в зазначений час водій прибуде та забере (чи допоможе завантажити) необхідний товар.

1.2 Мета

Основна мета створення програмного забезпечення - забезпечити зручну та ефективну організацію доставки вантажів для клієнтів у будь-якій точці країни; пов’язати логістично складні регіони та надати клієнтам широкий вибір способів задоволення їх потреб у перевезенні - з варіацією цін, термінів, допомоги в завантаженні/розвантаженні тощо.

Окрім того, ми хочемо максимально зменшити кількість “пустих” поїздок для вантажних автомобілів малого та середнього бізнесу - тобто максимізувати їх корисний ресурс. Для цього система буде забезпечувати можливість підбору вантажів, що мають схожий маршрут доставки, та можливість об'єднання їх в одне замовлення, а також надавати водіям можливість планувати свої маршрути з урахуванням можливих замовлень на перевезення вантажів.

Ми вбачаємо наше призначення у створенні зручного, надійного та ефективного ринку вантажних перевезень, що задовольняє потреби всіх

Рисунок В.3 – Сторінка 3 SRS-документу

учасників процесу, від користувачів до водіїв та компаній, що займаються перевезеннями.

1.3 Межі

Межі системи:
- мобільний додаток для клієнтів;
- мобільний додаток для водіїв;
- веб-інтерфейс для адміністрування системи;
- серверна частина для мобільних додатків (мікросервіси);
- серверна частина для веб-інтерфейсу для адміністрування системи (мікросервіси);
- база даних, яка зберігає інформацію про клієнтів, водіїв, вантажі, замовлення та доставки.

Межі функціоналу:
- можливість замовлення вантажних перевезень клієнтами через мобільний додаток;
- можливість для водіїв приймати замовлення та виконувати доставку;
- можливість для адміністратора системи керувати роботою системи, відстеження замовлень та доставки, аналіз статистики та звітності;
- можливість для адміністратора системи коригувати замовлення та вести діалоги з клієнтами та водіями, для вирішення проблем/питань.

Межі взаємодії з іншими системами:
- інтеграція з картографічними сервісами для відображення місцезнаходження водіїв та маршрутів доставки (Google Maps);
- інтеграція з платіжними системами для оплати вартості доставки через мобільний додаток (Google Pay);
- інтеграція з системами збору та аналізу метрик та іншої статистики щодо роботи системи (Elasticsearch, Kibana, Logstash).

Рисунок В.4 – Сторінка 4 SRS-документу

1.4 Посилання

Посилання на документи, що стосуються вимог до системи вантажних перевезень:

1. ДСТУ 2609-94 Вантажні автомобільні перевезення. Терміни та визначення
2. ДСТУ ISO 9001:2015 Системи управління якістю. Вимоги (ISO 9001:2015, IDT)
3. ДСТУ 3649:2010 КОЛІСНІ ТРАНСПОРТНІ ЗАСОБИ. Вимоги щодо безпечності технічного стану та методи контролювання

Посилання на документи, що стосуються вимог до програмного забезпечення:

1. ДСТУ ISO/IEC/IEEE 12207:2018 Інженерія систем і програмних засобів. Процеси життєвого циклу програмних засобів (ISO/IEC/IEEE 12207:2017, IDT)
2. ДСТУ ISO/IEC 25010:2016 Інженерія систем і програмних засобів. Вимоги до якості систем і програмних засобів та її оцінювання (SQuaRE). Моделі якості системи та програмних засобів (ISO/IEC 25010:2011, IDT)

1.5 Означення та абревіатури

АПІ (API) - Application Programming Interface, програмний інтерфейс застосунку.

GPS (GPS) - Global Positioning System, глобальна система позиціонування.

Рисунок В.5 – Сторінка 5 SRS-документу

REST (Representational State Transfer) - архітектурний стиль веб-сервісів, що використовується для взаємодії між клієнтськими та серверними додатками.

СКБД - Система керування базами даних.

WebSocket, WS - протокол, що призначений для обміну інформацією між браузером та вебсервером в режимі реального часу.

UI (User Interface) - графічний інтерфейс користувача, що використовується для взаємодії з програмним забезпеченням.

UX (User Experience) - загальний досвід користувача при взаємодії з програмним забезпеченням.

HTTPS (Hypertext Transfer Protocol Secure) - протокол передачі гіпертексту, що використовується для безпечного з'єднання між клієнтом та сервером (з'єднання мобільного застосунку із сервером встановлюється за допомогою HTTP/S).

OAuth (Open Authorization) - протокол авторизації, що використовується для надання обмеженого доступу до ресурсів користувача (авторизація через Google Sign-In працює з використанням протоколу OAuth 2.0).

Користувач - особа, що використовує мобільний застосунок (клієнт-замовник або водій).

Клієнт - особа, яка замовляє послуги з перевезення вантажів.

Водій - особа, яка виконує перевезення вантажів.

Логбук водія - електронне візуальне представлення останніх подій, що були зроблені водієм.

Замовлення - запит клієнта на перевезення вантажу.

Запланована поїздка - інформація, надана водієм, щодо його майбутньої поїздки між вказаними містами чи місцями, впродовж якої він може виконати замовлення.

Маршрут - шлях, за яким здійснюється перевезення вантажу.

Рисунок В.6 – Сторінка 6 SRS-документу

Радіус пошуку - відстань, в межах якої водій готовий прийняти замовлення.

Доставка - процес перевезення вантажу від місця відправлення до місця призначення.

Вантаж - предмети, що перевозяться.

Габарити - розміри вантажу.

Тип вантажу - характеристика вантажу, що визначає особливості його перевезення (харчові продукти, меблі, будівельні матеріали тощо).

Час доставки - час, протягом якого має бути здійснено доставку вантажу.

Оцінка - оцінка клієнтом якості наданих послуг з перевезення вантажу.

Відгук - коментар клієнта про надані послуги з перевезення вантажу.

Адміністратор - особа, що здійснює управління системою.

Модератор - особа, що представляє службу підтримки, комунікує з клієнтами, водіями та компаніями і має можливість редагувати замовлення, скасовувати оплату тощо.

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Ми бачимо Wide Delivery як екосистему, яка об'єднає клієнтів, водіїв та компанії, що займаються перевезеннями, надаючи їм інструменти для швидкої, надійної та ефективної доставки вантажів.

Наші плани розвитку сягають далеко за межі простого створення зручного додатку. Ми прагнемо стати лідером на ринку, задаючи нові стандарти якості, зручності та прозорості. Для досягнення цієї мети ми зосередимося на трьох ключових напрямках.

По-перше, ми будемо поступово розширювати географію обслуговування Wide Delivery. Наша мета - охопити всю територію України, зв'язавши навіть найвіддаленіші населені пункти. Ми прагнемо

Рисунок В.7 – Сторінка 7 SRS-документу

забезпечити доступність наших послуг для всіх, хто потребує швидкої та надійної доставки вантажів, незалежно від їх місця розташування.

По-друге, ми постійно працюємо над удосконаленням функціоналу мобільного додатку. Ми плануємо впровадити нові зручні функції, які зроблять процес замовлення та відстеження доставки ще більш інтуїтивним та ефективним. Наша команда аналізує відгуки користувачів, вивчає потреби ринку та слідкує за останніми технологічними трендами, щоб запропонувати нашим клієнтам найкращі рішення.

По-третє, ми активно працюємо над інтеграцією Wide Delivery з іншими системами. Ми розуміємо, що багато компаній вже використовують власне програмне забезпечення для управління логістикою, тому ми надамо їм можливість інтегрувати свої системи з Wide Delivery через API. Це дозволить автоматизувати процес отримання замовлень, управління доставкою та обміну даними, спрощуючи роботу та підвищуючи ефективність бізнес-процесів.

2.2 Функції програмного забезпечення

Програмне забезпечення для управління вантажними перевезеннями включає в себе широкий спектр функцій, які забезпечують ефективність та зручність використання для клієнтів, водіїв та адміністраторів системи. Нижче наведено список кожної з основних функцій:

- реєстрація та авторизація користувачів (клієнтів та водіїв);
- авторизація користувачів (адміністратор, головний помічник, помічник);
- введення інформації про вантаж та його характеристики;
- введення інформації про місце розташування вантажу та місце призначення;
- введення інформації про час доставки;
- обчислення вартості доставки;
- формування замовлення на доставку;
- пошук водіїв, які підходять під замовлення;
- приймання/відхилення замовлень водіями;

Рисунок В.8 – Сторінка 8 SRS-документу

- надсилання електронних листів з інформуванням про зміну статусу замовлення;
- відстеження статусу замовлення;
- спільна робота водіїв та клієнтів з доставкою;
- ведення статистики та звітності;
- адміністрування системи з веб-інтерфейсу (підтримка онлайн-чатів, редагування інформації про замовлення та скарги, тощо);
- інтеграція з картографічними сервісами (Google Maps/Google Streets);
- інтеграція з платіжними системами (Google Pay);
- забезпечення безпеки та конфіденційності даних користувачів;
- надання технічної підтримки користувачам.

2.3 Характеристики користувачі

Система передбачає наявність таких основних категорій користувачів: клієнт, водій, адміністратор, модератор.

Клієнти - фізичні та юридичні особи, які потребують послуг з доставки вантажів.

Водії - фізичні особи або транспортні компанії, що надають послуги з доставки вантажів.

Адміністратори - особи, що відповідають за управління системою, переглядом звітності, менеджмент персоналу

Модератори - особи, що мають доступ до інформації щодо клієнтів, водіїв, замовлення та можуть підтримувати комунікацію і вирішувати проблеми із взаємодією інших користувачів із системою.

Вимоги клієнтів:

Рисунок В.9 – Сторінка 9 SRS-документу

- просте та інтуїтивно зрозуміле користування додатком;
- можливість швидкого та зручного замовлення вантажоперевезень;
- можливість замовити допомогу в завантаженні вантажу чи його розвантаженні;
- можливість відстежувати статус замовлення в реальному часі;
- надійність та пунктуальність виконання замовлення;
- гарантії безпеки та повернення коштів у разі проблем із доставкою замовлень;
- зручні опції оплати;
- зручні варіанти авторизації (включно з Google Sign-In);
- можливість залишити відгук про виконання замовлення.
- можливість звернутись до технічної підтримки, якщо є така потреба

Вимоги водіїв:

- просте та інтуїтивно зрозуміле користування додатком;
- можливість швидкого та зручного прийняття замовлень;
- можливість планування маршрутів та вантажоперевезень, в тому числі запланованих на значний час наперед;
- можливість відстежувати статус виконання замовлення;
- надійну та своєчасну оплату за виконані перевезення;
- можливість залишити відгук про виконання замовлення.
- можливість звернутися до служби підтримки в текстовому форматі.

Вимоги адміністраторів:

- доступ до інформації про клієнтів та водіїв та можливість її редагування;
- можливість створювати, редагувати та видаляти модераторів та інших адміністраторів системи;
- доступ до інформації про замовлення та можливість їх редагування, скасування та повернення коштів;

Рисунок В.10 – Сторінка 10 SRS-документу

- можливість вести чат з клієнтами та водіями для надання допомоги та розв'язання проблем;
- можливість керувати скаргами та поверненнями коштів;
- доступ до статистики та звітів про роботу системи;
- можливість керувати доступом користувачів до системи та налаштуваннями безпеки.

Модератор має аналогічні до адміністратора вимоги щодо системи, однак він не має доступу до редагування інформації щодо персоналу, інших менеджерів та адміністраторів.

2.4 Загальні обмеження

Обмеження функціоналу

- система не передбачає перевезення небезпечних вантажів, таких як вибухонебезпечні, отруйні, радіоактивні та інші речовини, що підпадають під дію чинного законодавства щодо перевезення небезпечних вантажів;
- система не передбачає перевезення вантажів, що порушують чинне законодавство або права інтелектуальної власності;
- система не передбачає перевезення живих істот, за винятком тварин, які перевозяться разом з власниками в спеціальних контейнерах для перевезення тварин;
- система не передбачає перевезення вантажів, які перевищують максимально допустимі габарити та вагу, встановлені для перевезення на конкретному типі транспортного засобу;
- максимальні допустимі габарити вантажу визначаються параметрами транспортних засобів, що використовуються водіями-перевізниками. Користувач повинен вказати точні габарити свого вантажу під час створення замовлення, щоб система могла коректно підібрати відповідний транспортний засіб. Водій повинен

Рисунок В.11 – Сторінка 11 SRS-документу

вказати точні габарити свого транспортного засобу, для коректного пошуку замовлення під його автомобіль;

- система надає можливість користувачеві вибрати один з двох варіантів доставки: якомога швидша або запланована на певну дату в майбутньому. При виборі швидкої доставки, користувач розуміє, що вартість може бути вищою, ніж при замовленні на майбутнє. У разі вибору запланованої доставки, користувач зобов'язаний вказати точну дату і час, коли вантаж буде готовий до відправлення.

Обмеження географії:

- система не розрахована на міжнародні перевезення, географія місця призначення замовлення обмежується лише територією держави, звідки замовлення створено (початково - в межах України);
- система може не працювати в деяких районах, які є небезпечними для перевезення вантажів (регіон ведення бойових дій, тимчасово окуповані території тощо). Система повинна сповіщувати про неможливість перевезення вантажу ще на етапі створення замовлення.

Обмеження, що стосуються водіїв-перевізників:

- водії-перевізники повинні використовувати транспортні засоби, що відповідають вимогам системи щодо габаритів і типу вантажу. Система не допускає використання транспортних засобів, що не відповідають цим вимогам, для виконання замовлень на перевезення вантажів;
- водії-перевізники можуть вказувати радіус пошуку, якщо вони готові прийняти замовлення прямо зараз, або вказувати маршрут, за яким вони будуть їхати в майбутньому. Система підбирає замовлення, що проходять в межах радіусу чи вздовж маршруту (з можливими відхиленнями) і сповіщає водія-перевізника про можливі варіанти.

Рисунок В.12 – Сторінка 12 SRS-документу

Водій зобов'язаний вчасно повідомляти систему про зміни в своєму розкладі або маршруті;

- водії-перевізники можуть надавати послуги з допомоги в завантаженні та розвантаженні вантажу, але це не є обов'язковою умовою. Якщо водій-перевізник не може надати таку послугу, він повинен чітко вказати це в своєму профілі. Ця послуга обов’язково сплачується додатково до тарифу перевезення, про що клієнта буде повідомлено відразу при створенні замовлення.

Обмеження відповідальності:

- Система не несе відповідальності за вантаж під час перевезення. Відповідальність за вантаж покладається на відправника та водія-перевізника. У разі пошкодження або втрати вантажу, сторони зобов'язані самостійно вирішувати питання щодо компенсації збитків. Команда модераторів системи має обов’язок допомагати сторонам під час вирішення таких питань, однак не несе відповідальності за будь-які пошкодження;
- сторони зобов'язані дотримуватися вимог щодо безпеки перевезень, встановлених чинним законодавством. У разі порушення цих вимог, сторони несуть відповідальність відповідно до закону;
- система не гарантує надання послуг з перевезення вантажів у всіх випадках. У разі відсутності водіїв-перевізників, що готові виконати замовлення або за наявності інших обставин, що перешкоджають наданню послуг, система повідомляє користувача про неможливість виконання замовлення.

2.5 Припущення і залежності

Для того, щоб Wide Delivery успішно функціонував та розвивався, ми враховуємо низку важливих факторів. По-перше, ми виходимо з припущення, що наші користувачі мають доступ до сучасних мобільних

Рисунок В.13 – Сторінка 13 SRS-документу

пристроїв з операційною системою Android та стабільним інтернет-з'єднанням, що дозволить їм повноцінно використовувати всі можливості мобільного додатку. Ми також очікуємо, що водії, які приєднаються до платформи, мають необхідний досвід та кваліфікацію для керування вантажними автомобілями, а також відповідально ставляться до правил дорожнього руху. Крім того, ми розраховуємо на те, що всі учасники платформи - клієнти, водії та транспортні компанії - будуть діяти відповідно до чинного законодавства України, яке регулює сферу вантажних перевезень. Ми також спираємося на стабільну та безперебійну роботу сторонніх сервісів, таких як Google Maps, Google Pay та інші платформи, інтегровані в Wide Delivery.

Не менш важливим є розуміння залежностей, які можуть впливати на роботу платформи. Очевидно, що стабільне інтернет-з'єднання є критично важливим як для клієнтів, так і для водіїв. Будь-які проблеми з інтернетом можуть призвести до збоїв у роботі Wide Delivery та ускладнити процес замовлення та відстеження доставки. Точність та доступність геолокаційних даних, що надходять з мобільних пристроїв, також є ключовим фактором для коректної роботи платформи. Неточності або відсутність геолокації можуть спричинити помилки у визначенні місця розташування клієнтів та водіїв, а також ускладнити побудову оптимальних маршрутів доставки.

Безпека та надійність інтегрованих платіжних систем також є важливою складовою успіху Wide Delivery. Від них залежить можливість клієнтів зручно та безпечно оплачувати замовлення.

Не менш важливою є довіра та репутація платформи. Ми докладаємо максимум зусиль, щоб створити Wide Delivery як надійний та безпечний сервіс, який гарантує якісне виконання замовлень, захищає інтереси всіх учасників та формує позитивний досвід взаємодії.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до зовнішніх інтерфейсів

3.1.1 Інтерфейс користувача

Основна взаємодія користувача з системою Wide Delivery відбувається через мобільний додаток, який розроблено з метою забезпечення максимально зручного та інтуїтивно зрозумілого доступу до

Рисунок В.14 – Сторінка 14 SRS-документу

всіх функцій сервісу. Додаток, призначений як для клієнтів, так і для водіїв, має дозволяти виконувати всі необхідні дії безпосередньо в ньому, без необхідності звертатися до сторонніх сервісів.

Користувачі можуть авторизуватися в системі за допомогою облікового запису Google або ж створити обліковий запис, використовуючи свою електронну пошту. Клієнти легко створюють замовлення на перевезення, вказуючи всі необхідні параметри вантажу: його габарити, тип, адресу завантаження та доставки, бажаний час прибуття, а також мають можливість зазначити необхідність допомоги з завантаженням та розвантаженням. Водії, в свою чергу, мають доступ до детальної інформації про доступні замовлення, включаючи маршрут, тип вантажу, вартість доставки та інші важливі деталі.

Навігація в додатку має бути простою та інтуїтивно зрозумілою, дозволяючи швидко переміщуватися між різними екранами та розділами. Система забезпечує можливість відстежувати поточний статус замовлення. Користувачі також мають можливість редагувати свої профілі, змінюючи особисту інформацію та налаштування. У системі є можливість звертатися до служби підтримки у зручному месенджері Telegram, де їм зможуть надати допомогу Support'и.

Wide Delivery має включати систему оплати з чітко визначеними тарифами, що гарантує прозорість та передбачуваність вартості послуг.

Адміністратори системи Wide Delivery взаємодіють з нею через спеціалізований веб-інтерфейс, який забезпечує зручний та інтуїтивно зрозумілий доступ до всіх функцій управління сервісом. Авторизація здійснюється через ввід пошти та паролю, після чого адміністратори можуть керувати базою користувачів, включаючи створення, редагування та видалення облікових записів. Вони мають доступ до детальної інформації про всі замовлення, можуть переглядати та редагувати деталі замовлень, відстежувати статус доставки в реальному часі, а також аналізувати різні аспекти замовлень для підвищення ефективності операцій.

Крім того, веб-інтерфейс адміністратора надає можливості для аналізу статистики, що допомагає в ухваленні обґрунтованих рішень. Вони також відповідають за підтримку безпеки та конфіденційності даних користувачів. Для надання підтримки користувачам, адміністратори можуть використовувати вбудований чат або електронну пошту, що

Рисунок В.15 – Сторінка 15 SRS-документу

дозволяє швидко реагувати на запити та вирішувати проблеми, забезпечуючи високий рівень обслуговування.

3.1.2 Апаратні інтерфейси

Пристрої, що використовують мобільний додаток, повинні мати не менш ніж 2ГБ ОЗУ, і мати частоту процесора не нижче 1 ГГц. Мобільні пристрої повинні мати 350 Мб вільної пам'яті у постійному сховищі для встановлення додатку та його нормального функціонування.

3.1.3 Програмний інтерфейс

Мобільний додаток створюється першочергово для Android-пристроїв, отже Android версії 7.0+ є вимогою щодо ОС мобільного пристрою.

Будуть використовуватися наступні API:

- Google OAuth 2.0 API;
- Google Maps API;
- Google Pay API.

Браузери, що запускатимуть веб-застосунок адміністраторської частини мають підтримувати стандарт ECMAScript 2015 (ES6). Отже, мінімальними необхідними версіями найпопулярніших у світі браузерів є:

- Chrome 51 + або вище;
- Edge 15 або вище;
- Safari 10 або вище;
- Mozilla Firefox 54 або вище;
- Opera 38 або вище;
- Internet Explorer 11 (частково).

Веб-застосунок, призначений для адміністраторів та модераторів, може бути запущений з Windows 7 або вище, Linux, MacOS 10.5 або вище.

Рисунок В.16 – Сторінка 16 SRS-документу

3.1.4 Комунікаційний протокол

Wide Delivery використовує різноманітні комунікаційні протоколи для забезпечення ефективної та надійної взаємодії між різними компонентами системи.

Для спілкування мобільного додатку з сервером буде застосовано HTTP/HTTPS протокол. Це забезпечить безпечну передачу даних та захистить конфіденційну інформацію користувачів. Окрім того, буде впроваджено технологію WebSocket для підтримки швидкої передачі постійних пакетів даних із сервера на клієнт. WebSocket дозволить реалізувати функції реального часу, такі як відображення поточного місцезнаходження вантажу на карті та оновлення статусу замовлення.

Внутрішня комунікація між мікросервісами та основним API на стороні сервера буде здійснюватися за допомогою системи gRPC. gRPC - це сучасний високопродуктивний фреймворк для віддаленого виклику процедур (RPC), який забезпечує швидку та ефективну взаємодію між сервісами. Використання gRPC дозволить оптимізувати роботу серверної частини Wide Delivery та забезпечити її масштабованість.

REST буде використовуватися для надання зовнішнього API для інтеграції з іншими системами, такими як платіжні шлюзи та картографічні сервіси. REST API забезпечить стандартизований інтерфейс для взаємодії з Wide Delivery та дозволить стороннім розробникам легко інтегрувати свої сервіси з нашою платформою.

3.1.5 Обмеження пам'яті

Система Wide Delivery повинно бути оптимізованою для роботи з великими обсягами даних, зберігаючи при цьому швидку та стабільну роботу системи. Для цього необхідно застосовувати алгоритми стиснення даних, кешування, а також вміло використовувати пам'ять пристрою.

Рисунок В.17 – Сторінка 17 SRS-документу

Основна інформація зберігатиметься в реляційних та NoSQL базах даних, тож додаток повинен мати мінімальну кількість інформації, що зберігається безпосередньо на мобільному девайсі.

3.1.6 Операції

Основна мета: Опис операцій, які здійснюються в рамках системи, дозволяє забезпечити зручність і ефективність процесів взаємодії користувачів з додатком.

Операції системи включають:

1. **Створення замовлення:** Користувачі можуть створювати замовлення через мобільний додаток, вказуючи деталі вантажу (габарити, тип, час доставки, місце відправлення та прибуття). Система автоматично підбирає водія на основі заданих критеріїв.
2. **Оновлення статусу замовлення:** Водії можуть оновлювати статус замовлення у режимі реального часу, що дає можливість користувачам відстежувати процес доставки.
3. **Підтвердження доставки:** Після завершення доставки водій вводить статус як завершений, а клієнт отримує повідомлення про успішну доставку з можливістю залишити відгук.
4. **Скасування замовлення:** Користувач має можливість скасувати замовлення до моменту його прийняття водієм, з інформуванням водія та поверненням коштів.
5. **Оплата за допомогою інтегрованих платіжних систем:** Користувачі можуть оплачувати послуги через зручні та безпечні платіжні системи, інтегровані в мобільний додаток.

Рисунок В.18 – Сторінка 18 SRS-документу

Зауваження: Всі операції повинні супроводжуватися детальними повідомленнями про статус операції для забезпечення прозорості та контролю за процесом доставки вантажів.

3.1.7 Функції продукту

3.1.7.1 Реєстрація, авторизація (мобільний застосунок)

Вступ

Користувач повинен мати можливість авторизуватися через Google або зареєструвати обліковий запис через пошту для подальшої авторизації в додатку під своїм обліковим записом.

Вхідні дані

Користувач завантажив застосунок, вводить електронну пошту та пароль або авторизується через обліковий запис Google та натискає кнопку підтвердження.

Обробка

За наявності акаунту, перевіряється правильність введеного паролю, інакше створюється новий акаунт. Для Google Sign In зберігати пароль в БД не потрібно.

Результати

У разі правильних надісланих даних користувач отримує токен доступу до системи та токен відновлення токену доступу. Застосунок перенаправляє його на головну сторінку мобільного додатку.

Обробка помилок

Користувач не зможе зареєструватися/увійти у разі будь-якої помилки, неправильного паролю, неіснуючого акаунту, або якщо намагається пройти аутентифікацію з Google, при створеному за допомогою пошти та паролю акаунті чи навпаки. Back-end повинен повертати HTTP-статус 500 для внутрішніх помилок сервера, 403 для неправильно введених пошти чи пароля та некоректного OAuth-токену.

3.1.7.2 Створення замовлення (мобільний застосунок)

Вступ

Користувач повинен мати можливість створити замовлення після успішної аутентифікації.

Вхідні дані

Рисунок В.19 – Сторінка 19 SRS-документу

Користувач вводить інформацію про вантаж, його тип, габарити, місце призначення та час відправлення, зазначає необхідність у допомозі з завантаження чи розвантаження.

Обробка

Замовлення перевіряється на валідність, оцінюється можливість його виконання, визначається його ціна та розпочинається підбір водіїв, що можуть виконати це замовлення.

Результати

У разі коректного створення замовлення воно валідується та переходить на етап пошуку водія, що зможе виконати замовлення. Також користувач отримує інформацію про орієнтовну вартість замовлення із пропозицією щодо сплати за допомогою Google Pay.

Обробка помилок

У разі помилки створення замовлення користувач отримує інформацію про причину помилки та контактами служби підтримки, для уточнення деталей. Back-end клієнтської частини повинен повертати повідомлення, чому саме замовлення не може бути прийнято до обробки.

3.1.7.3 Оплата замовлення

Вступ

Клієнт повинен оплатити замовлення для того, щоб водій розпочав його виконання.

Вхідні дані

Замовлення успішно прийнято до обробки та клієнтський back-end визначнив вартість замовлення.

Обробка

Сторонній сервіс проводить оплату та передає результат оплати серверу. Сервер перевіряє, чи справді сторонній сервіс провів оплату, і чи запит не був виконаний зловмисником.

Результати

У разі успішної оплати розпочинається процес підбору водія.

Обробка помилок

У разі проведення оплати та списання коштів, але відсутності зміни статусу замовлення, користувач має змогу звернутись до служби підтримки. У разі помилкових відповідей від постачальника платіжних послуг, користувачу буде запропоновано ще раз оплатити замовлення.

Рисунок В.20 – Сторінка 20 SRS-документу

3.1.7.4 Перегляд статусу замовлення в режимі реального часу (мобільний застосунок)

Вступ

Клієнт повинен бути проінформованим про зміну статусу замовлення та його поточне місцерозташування.

Вхідні дані

Підтверджене замовлення, що розпочало своє виконання. Водій підтвердив відправку до місця призначення. Водій передає своє місцезнаходження за допомогою увімкненого на мобільному пристрої GPS.

Обробка

Перевірка того, що GPS збігається з визначеним маршрутом перевезення та збереження поточного місцезнаходження, оцінка очікуваного часу прибуття. Надсилання сповіщення клієнтові щодо статусу замовлення за допомогою електронної пошти.

Результати

Клієнт дізнається про статус замовлення та поточне місцезнаходження вантажу.

Обробка помилок

В разі помилки отримання місцезнаходження впродовж виначеного часу модератор служби підтримки зв'язується із водієм. Також надсилається електронний лист про помилку, що сталась під час виконання замовлення.

3.1.7.5 Підбір замовлення для водія (мобільний застосунок)

Вступ

Система підбирає замовлення для водія та пропонує прийняти його.

Вхідні дані

Водій вказує свої налаштування - радіус пошуку, або запланований маршрут, можливість бути вантажником, а також поточний GPS (надає дозвіл додатку на передачу).

Обробка

Система підбирає замовлення, що відповідають вказаним у налаштуваннях критеріях, та пропонує перелік підібраних замовлень.

Результати

Рисунок В.21 – Сторінка 21 SRS-документу

Водій бачить список замовлень та приймає їх, або відхиляє. В разі відхилення, замовлення більше не з'являється у списку та передається наступному водієві.

Обробка помилок

В разі помилок з отриманням замовлень водієві буде запропоновано звернутись до служби підтримки застосунку.

3.1.7.6 Звернення до служби підтримки (мобільний застосунок, система модерації)

Вступ

Користувач має можливість звернутись до служби підтримки, що працює за допомогою чат-боту Telegram.

Вхідні дані

Користувач мобільного застосунку (водій чи клієнт) потребує допомоги служби підтримки та натискає кнопку в мобільному застосунку, що розпочинає діалог в чат-боті Telegram.

Обробка

Користувач інтерфейсу адміністратора відповідає на скаргу в інтерактивному вигляді у виді чату.

Результати

Після завершення спілкування по скарзі, користувач інтерфейсу адміністратора змінює статус скарги до відповідного рішення, прийнятого обома сторонами. Клієнт також має можливість завершити розгляд питання, якщо його було вирішено.

3.1.7.7 Перегляд статистики на головній сторінці (система модерації)

Вступ

Користувач інтерфейсу адміністратора після логіну заходить на головну сторінку.

Вхідні дані

Користувач інтерфейсу адміністратора повинен бути зареєстрованим.

Обробка

Користувач інтерфейсу адміністратора перевіряється на аунтифікацію та після цього йому надається доступ до головної сторінки.

Результати

Після завершення логіну, користувач інтерфейсу адміністтратора може передивитися статистику по кількості скарг за останній місяць за

Рисунок В.22 – Сторінка 22 SRS-документу

кожен день, кількість замовлень за останній місяць за кожен день, та кількість скарг за статусами, кількість замовлень за статусами.

3.1.7.8 Вхід користувача до інтерфейсу адміністратора (система модерації)

Вступ

Користувач інтерфейсу адміністратора повинен увійти у свій обліковий запис.

Вхідні дані

Користувач інтерфейсу адміністратора повинен перейти на сторінку логіну та ввести пошту та пароль.

Обробка

Користувач інтерфейсу адміністратора перевіряється на аунтифікацію та після цього йому надається доступ до головної сторінки.

Результати

Після завершення логіну, користувач інтерфейсу адміністтратор має доступ до сторінок відповідно до своєї ролі у системі.

3.1.7.9 Перегляд статистики на сторінці статистики (система модерації)

Вступ

Користувач інтерфейсу адміністратора переглядає статистику.

Вхідні дані

Користувач інтерфейсу адміністратора повинен бути аутентифікований у системі та перейти на сторінку статистики.

Обробка

Користувач інтерфейсу адміністратора перевіряється на аунтифікацію та після цього йому надається доступ до сторінки статистики

Результати

Після завершення перевірки аутентифікації, користувач інтерфейсу адміністратор має доступ до сторінки статистики де може побачити статистику по скаргам та замовленням.

3.1.7.10 Зміна налаштувань замовлення (система модерації)

Вступ

Користувач інтерфейсу адміністратора за проханням клієнта змінює налаштування замовлення клієнта.

Рисунок В.23 – Сторінка 23 SRS-документу

Вхідні дані

Користувач інтерфейсу адміністратора отримує прохання від клієнта та переходить на сторінку замовлення де натискає на потрібне замовлення.

Обробка

Користувач інтерфейсу адміністратора домовляється з користувачем та змінює відповідно до потреби користувача замовлення.

Результати

Після завершення, користувач інтерфейсу адміністратора бачить успішно змінене замовлення та повідомляє про це користувачу.

3.1.7.11 Видалення замовлення (система модерації)

Вступ

Користувач інтерфейсу адміністратора за певних технічних обставин видаляє замовлення.

Вхідні дані

Користувач інтерфейсу адміністратора отримує прохання від клієнта, або із-за технічних причин повинен перейти на сторінку замовлення де натискає у потрібному замовленні кнопку видалили.

Обробка

Відправляється запит на сервер, після цього потрібне замовлення знаходиться і видаляється з бази даних та повертає результат видалення.

Результат

Після завершення, користувач інтерфейсу адміністратора бачить успішне видалення замовлення та повідомляє про це користувачу.

3.1.7.12 Видалення скарг (система модерації)

Вступ

Користувач інтерфейсу адміністратора за певних технічних обставин видаляє скаргу.

Вхідні дані

Користувач інтерфейсу адміністратора отримує прохання від клієнта, або із-за технічних причин повинен перейти на сторінку скарг де натискає у потрібній скарзі кнопку видалили.

Обробка

Відправляється запит на сервер, після цього потрібна скарга знаходиться і видаляється з бази даних та повертає результат видалення.

Результат

Рисунок В.24 – Сторінка 24 SRS-документу

Після завершення, користувач інтерфейсу адміністратора бачить успішне видалення скарги та повідомляє про це користувачу.

3.1.7.13 Переглядь інформації про користувача інтерфейсу адміністратора (система модерації)

Вступ

Користувач інтерфейсу адміністратора хоче подивитися інформацію про себе.

Вхідні дані

Користувач інтерфейсу адміністратора хоче подивитися інформацію про себе.

Обробка

Відправляється запит на сервер, після цього інформація за своїм профілем надсилається у відповідь.

Результат

Після завершення, користувач інтерфейсу адміністратора бачить інформацію про себе.

3.1.8 Припущення й залежності

При розробці системи вантажних перевезень ми робимо наступні припущення:

- користувачі мають доступ до мобільного інтернету та сумісних пристроїв на Android/IOS для взаємодії з нашим додатком;
- водії мають належні права та документи для здійснення вантажних перевезень, а також транспортні засоби, що відповідають вимогам безпеки та технічного стану;
- компанії, що займаються вантажними перевезеннями, мають необхідні ліцензії та дозволи на надання послуг з перевезення вантажів;
- водії зобов'язані дотримуватися правил дорожнього руху та інших нормативних вимог під час виконання замовлень на перевезення вантажів;

Рисунок В.25 – Сторінка 25 SRS-документу

- геолокаційні дані, що використовуються в додатку, надають точну та актуальну інформацію про місцезнаходження користувачів та водіїв, в тому числі під час виконання замовлення.
- система оплати в додатку функціонує належним чином, забезпечуючи безпечну та зручну оплату послуг з перевезення вантажів;

Залежності системи:

- для належної роботи системи необхідна взаємодія з зовнішніми сервісами та інтерфейсами, зокрема картографічними сервісами (Google Maps API, Google Street API), геолокаційними сервісами, системами оплати (Google Pay) тощо;
- система залежить від стабільної роботи мережі Інтернет, як для користувачів, так і для водіїв. Коректність відображення актуального місцезнаходження вантажу залежить від якості підключення до мобільної мережі;
- для забезпечення безпеки та якості послуг система залежить від дотримання користувачами та водіями правил дорожнього руху та інших нормативних вимог;
- система залежить від рівня довіри між користувачами та водіями, а також від якості наданих послуг з перевезення вантажів;
- система залежить від актуальності та повноти інформації про вантажі, маршрути та інші параметри, що вводяться користувачами та водіями;
- система залежить від ефективної взаємодії між користувачами, водіями, адміністраторами системи та іншими учасниками процесу перевезення вантажів.

Рисунок В.26 – Сторінка 26 SRS-документу

3.2 ФУНКЦІОНАЛЬНІ ВИМОГИ

3.2.1 FR-1 Реєстрація, авторизація

3.2.1.1 Вступ

Користувач повинен мати право авторизуватися через Google або зареєструвати обліковий запис через пошту для подальшої авторизації в додатку під своїм обліковим записом.

3.2.1.2 Вхідні дані

Користувач вводить свої дані у форму авторизації/реєстрації та натискає кнопку підтвердження. Або авторизується через обліковий запис Google

3.2.1.3 Обробка

Користувач вводить свої дані у форму авторизації/реєстрації та натискає кнопку підтвердження.

3.2.1.4 Результати

У разі правильних надісланих даних користувача перенаправляє на головну сторінку мобільного додатку, в іншому випадку отримує повідомлення про помилку.

3.2.1.5 Обробка помилок

Користувач не зможе зареєструватися/увійти у разі будь-якої помилки.

3.2.2 FR-2 Створення замовлення

3.2.2.1 Вступ

Основний процес створення замовлення зі сторони клієнта.

3.2.2.2 Вхідні дані

Користувач вводить інформацію про вантаж, його тип, габарити, місце призначення та час відправлення, зазначає необхідність у допомозі з завантаження чи розвантаження.

3.2.2.3 Обробка

Рисунок В.27 – Сторінка 27 SRS-документу

Замовлення перевіряється на валідність, оцінюється можливість його виконання, визначається його ціна та розпочинається підбір водіїв, що можуть виконати це замовлення.

3.2.2.4 Результати

У разі коректного створення замовлення воно валідується та переходить на етап пошуку водія, що зможе виконати замовлення. Також користувач отримує інформацію про орієнтовну вартість замовлення.

3.2.2.5 Обробка помилок

У разі помилки створення замовлення користувач отримує інформацію про причину помилки та контактами служби підтримки, для уточнення деталей.

3.2.3 FR-3 Оплата замовлення

3.2.3.1 Вступ

Клієнт повинен оплатити замовлення для того, щоб водій розпочав його виконання.

3.2.3.2 Вхідні дані

Водій підтвердив замовлення, створене клієнтом. Клієнт отримав сповіщення про це та кінцеву варітсть перевезення.

3.2.3.3 Обробка

Сторонній сервіс проводить оплату та передає результат оплати серверу. Сервер перевіряє, чи справді сторонній сервіс провів оплату, чи запит виконаний зловмисником.

3.2.3.4 Результати

У разі успішної оплати водій сповіщується про це та може виконувати замовлення.

3.2.3.5 Обробка помилок

Рисунок В.28 – Сторінка 28 SRS-документу

Користувач має змогу провести оплату кілька разів. Після чого замовлення буде відхилено або створено квиток у службу підтримки.

3.2.4 FR-4 Перегляд статусу замовлення в режимі реального часу

3.2.4.1 Вступ

Клієнт має можливість переглянути поточне місцезнаходження.

3.2.4.2 Вхідні дані

Підтверджене замовлення, що розпочало своє виконання. Водій підтвердив відправку до місця призначення. Водій передає своє місцезнаходження за допомогою увімкненого на мобільному пристрої GPS.

3.2.4.3 Обробка

Перевірка того, що GPS збігається з визначеним маршрутом перевезення та збереження поточного місцезнаходження, оцінка очікуваного часу прибуття.

3.2.4.4 Результати

Клієнт переглядає місцезнаходження вантажу та очікуваний час прибуття.

3.2.4.5 Обробка помилок

В разі помилки отримання місцезнаходження впродовж виначеного часу модератор служби підтримки зв'язується із водієм.

3.2.5 FR-5 Підбір замовлення для водія

3.2.5.1 Вступ

Система підбирає замовлення для водія та пропонує прийняти його.

3.2.5.2 Вхідні дані

Рисунок В.29 – Сторінка 29 SRS-документу

Водій вказує свої налаштування - радіус пошуку, або запланований маршрут, можливість бути вантажником, а також поточний GPS (надає дозвіл додатку на передачу).

3.2.5.3 Обробка

Система підбирає замовлення, що відповідають вказаним у налаштуваннях критеріях, та пропонує перелік підібраних замовлень.

3.2.5.4 Результати

Водій бачить список замовлень та приймає їх, або відхиляє. В разі відхилення, замовлення більше не з'являється у списку та передається наступному водієві.

3.2.5.5 Обробка помилок

В разі помилок з отриманням замовлень водієві буде запропоновано звернутись до служби підтримки застосунку.

3.2.6 FR-6 Звернення до служби підтримки

3.2.6.1 Вступ

Клієнт або водій має можливість звернутись до служби підтримки.

3.2.6.2 Вхідні дані

Текст звернення до служби підтримки.

3.2.6.3 Обробка

Система отримує звернення, котре було зареєстроване через телеграм - бота, ті віддає його команді технічної підтримки.

3.2.6.4 Результати

Модератор або клієнт закриває звернення тоді, коли клієнт чи водій задоволені отриманою відповіддю та допомогою.

3.2.6.5 Обробка помилок

В разі виникнення помилок, модератор сповіщує команду розробки про проблему під час звернення.

Рисунок В.30 – Сторінка 30 SRS-документу

3.2.7 FR-7 Перегляд статистики системи

3.2.7.1 Вступ

Адміністратор системи може переглядати статистику стосовно звернень та замовлень.

3.2.7.2 Вхідні дані

Відкриття сторінки статистики, з відповідним доступом

3.2.7.3 Обробка

Система перевіряє, чи має користувач достатньо прав для перегляду статистики, і якщо користувач має відповідні права, то віддає статистичні дані.

3.2.7.4 Результати

Відкриття сторінки з статистикою.

3.2.7.5 Обробка помилок

У разі виникнення помилок, користувачу буде відображене відповідне вікно, з тим причиною, чому саме він бачить цю помилку

3.2.8 FR-8 Модерація системи

3.2.8.1 Вступ

Адміністратор системи або модератор має право, у разі отримання запиту від користувача внести зміни у замовлення, виправити помилку в акаунті користувача, водія.

3.2.8.2 Вхідні дані

Рисунок В.31 – Сторінка 31 SRS-документу

Відкриття сторінки з користувачами або замовленнями

3.2.83 Обробка

Система перевіряє, чи має користувач достатньо прав для перегляду відповідної сторінки, та після процесу аутентифікації пропускає користувача далі.

3.2.8.4 Результати

Відкриття сторінки з користувачами системи або замовленнями

3.2.8.5 Обробка помилок

У разі виникнення помилок, користувачу буде відображене відповідне вікно, з тим причиною, чому саме він бачить цю помилку

3.2.8 FR-8 Детальний пошук для адміністраторів системи

3.2.8.1 Вступ

Адміністратор системи або модератор має право, у разі отримання запиту скористатися детальним пошуком у системі, з усіма можливими параметрами, для того, щоб знайти відповідне замовлення, або користувача/водія.

3.2.8.2 Вхідні дані

Відкриття сторінки з користувачами або замовленнями

3.2.83 Обробка

Рисунок В.32 – Сторінка 32 SRS-документу

Система перевіряє, чи має користувач достатньо прав для перегляду відповідної сторінки, та після процесу аутентифікації пропускає користувача далі.

3.2.7.4 Результати

Відкриття сторінки з користувачами системи або замовленнями

3.2.7.5 Обробка помилок

У разі виникнення помилок, користувачу буде відображене відповідне вікно, з тим причиною, чому саме він бачить цю помилку

3.3.1 Надійність

Надійність програмного забезпечення є критичною вимогою, оскільки система займається координацією та управлінням логістичними процесами, що впливають на щоденні операції користувачів і бізнесів. Для забезпечення високого рівня надійності програмного продукту, наступні критерії повинні бути виконані:

1. Відновлення після збоїв: Система повинна мати можливість автоматично відновлювати роботу після непередбачених збоїв. Це включає реалізацію стратегій на кшталт автоматичного перезапуску сервісів та використання транзакційної пам'яті для забезпечення цілісності даних.
2. Резервне копіювання даних: Періодичне резервне копіювання даних має бути автоматизовано. Система має забезпечувати легке відновлення даних з резервних копій у випадку їх втрати або пошкодження. Дозволяється налаштувати автоматичне резервне копіювання обраної СКБД.
3. Висока доступність: Система повинна бути розроблена з використанням архітектурних патернів, що підтримують високу доступність, включаючи класичний поділ на кластери, балансування навантаження та відмовостійкість.

Рисунок В.33 – Сторінка 33 SRS-документу

4. Швидке відновлення: Мінімізація часу відновлення системи після збою є обов'язковою. Система повинна мати здатність швидко відновлювати операції без значної втрати даних.
5. Моніторинг стану системи: Необхідно імплементувати інструменти для постійного моніторингу стану всіх компонентів системи, щоб забезпечити раннє виявлення потенційних проблем та уникнення непередбачених збоїв. Зокрема, за допомогою збору логів та метрик з Logstash до Elasticsearch.
6. Уніфікація середовища: для уникнення додаткових проблем під час розгортання компонентів системи необхідно створити проект, кожен елемент якого (сервіс) працює в Docker-контейнері, що працює однаково в оточенні розробника та production-оточенні за умови наявності однакових змінних середовища.

Ці вимоги до надійності забезпечують, що система може функціонувати ефективно і безперебійно, максимізуючи задоволеність користувачів і надійність бізнес-процесів.

3.3.2 Доступність

Система Wide Delivery має бути доступною для користувачів протягом більшої частини доби, враховуючи особливості ринку вантажних перевезень. Доступ до функціоналу платформи, такого як створення замовлень, пошук водіїв, відстеження доставки та спілкування з службою підтримки, буде забезпечено в проміжку часу з 6:30 до 24:00. Цей часовий діапазон обрано з урахуванням типових годин роботи більшості транспортних компаній та індивідуальних перевізників, а також потреб клієнтів, які, як правило, здійснюють замовлення на перевезення вантажів в денний та вечірній час.

Для забезпечення безпеки та конфіденційності даних, доступ до функціоналу Wide Delivery буде надано виключно авторизованим користувачам. Кожен користувач, незалежно від того, чи це клієнт, водій або представник транспортної компанії, повинен буде пройти процедуру реєстрації та створити особистий обліковий запис. Для входу в систему буде використана безпечна система авторизації, яка захистить облікові записи користувачів від несанкціонованого доступу.

3.3.4 Супроводжуваність

Рисунок В.34 – Сторінка 34 SRS-документу

Основна мета: Забезпечити легкість та ефективність обслуговування, оновлення, налаштування та розширення програмного продукту протягом усього життєвого циклу.

Опис атрибутів супроводжуваності:

1. **Модульність:** Система розроблена з використанням чітко визначених модулів, що дозволяє легко замінювати або оновлювати окремі компоненти без впливу на решту системи.
2. **Читабельність коду:** Програмний код має бути написаний з дотриманням стандартів кодування та коментування, що забезпечує його легкість для розуміння та подальшого супроводження розробниками.
3. **Документація:** Повна технічна документація системи має бути надана разом з продуктом. Документація повинна включати керівництва для розробників та користувачів, опис архітектури системи, а також процедури оновлення та вирішення проблем.
4. **Логування та моніторинг:** Система повинна включати розширені можливості логування та моніторингу, що дозволяє легко відслідковувати роботу програми та швидко виявляти та усувати помилки.
5. **Інтернаціоналізація:** Програмний продукт має підтримувати локалізацію для різних мов та регіональних налаштувань, що дозволяє адаптувати продукт для різних міжнародних ринків.
6. **Підтримка версій:** Всі оновлення програмного забезпечення мають супроводжуватись чітким веденням версій, що дозволяє користувачам легко переходити між версіями та відновлювати попередні конфігурації при необхідності.

Зауваження: Висока супроводжуваність забезпечує нижчі загальні витрати на володіння продуктом та збільшує задоволеність користувачів, що є критично важливим для довгострокового успіху програмного продукту.

Рисунок В.35 – Сторінка 35 SRS-документу

3.3.5 Переносимість

Проект має бути налаштований для роботи у контейнеризованому середовищі за допомогою Docker-compose, що дозволяє забезпечити легке та швидке розгортання на різних платформах: Ubuntu, Windows, і MacOS. Розгортання повинно відбуватися за допомогою однієї команди (`make start`), що спрощує процес ініціації та управління.

Конфігурація повинна включати всі необхідні сервіси та залежності, які описуються в docker-compose.yml файлі для кожного з репозиторіїв. Усі репозиторії (окремі сервіси) мають бути поєднаними в один проект. Додатково в кожному з репозиторіїв потрібно створити Makefile, що містить команду make start, яка автоматизує процес розгортання контейнерів та залежностей.

Таким чином гарантується ідентичність середовища та легкий запуск з будь-якої платформи розробки чи розгортання.

3.3.6 Продуктивність

Система Wide Delivery повинна демонструвати високу продуктивність та забезпечувати швидку реакцію на дії користувачів, навіть за умов значного навантаження. Мобільний додаток повинен бути оптимізований для роботи на пристроях з різними характеристиками та забезпечувати плавну роботу інтерфейсу без помітних затримок. Зокрема, головний екран додатку, що містить інтерактивну карту та форми для введення даних, повинен завантажуватись менше ніж за 2 секунди. Це забезпечить позитивний користувацький досвід та дозволить клієнтам швидко розпочати процес замовлення перевезення.

Серверна частина Wide Delivery повинна бути спроможна обробляти запити від великої кількості користувачів одночасно, забезпечуючи стабільну роботу платформи навіть у періоди пікового навантаження. Система має бути спроектована з урахуванням можливості масштабування, що дозволить збільшувати її потужність паралельно зі зростанням кількості користувачів та замовлень.

3.4 Вимоги бази даних

Необхідно обрати NoSQL базу даних, що відповідає PA/EC рішенню за теоремою PACELC. Висока доступність та консистентність даних є

Рисунок В.36 – Сторінка 36 SRS-документу

важливими вимогами до системи. База даних має легко масштабуватись та легко піддаватись партиціонуванню. Чудовим кандидатом для такої БД є MongoDB.

Також систему необхідно забезпечити сховищем кешованих даних для швидкого доступу до потрібної інформації. Зокрема, такою інформацією є сесії користувачів, що виконали аутентифікацію та використовують мобільний застосунок, а також дані для роботи чат-боту служби підтримки в Telegram.

Логи та метрики, зокрема щодо запитів користувачів, повинні зберігатись у пошуковому сервері Elasticsearch задля можливості швидкого доступу до напівструктурованої інформації.

3.5 Інші вимоги

Кожна зміна документу має бути обговорена командою розробників і власником проекту. Після внесення цих змін - підписана кожним із них.

Рисунок В.37 – Сторінка 37 SRS-документу

ДОДАТОК Г

Приклад програмного коду (UserService)

```
@Service
internal class UserService(
    private val userRepository: UserRepository,
    private val customMongoSupportUserRepository:
CustomMongoSupportUserRepository
) {
    fun getUsers(page: Int, size: Int): Page<SupportUser> =
        userRepository.findAll(PageRequest.of(page, size))

    fun getUserById(id: String) = userRepository.findById(ObjectId(id))
        .orElseThrow { IllegalArgumentException("User with id $id not
found") }

    fun createUser(supportUser: SupportUser) =
userRepository.save(supportUser)

    fun updateUser(userId: String, supportUser: UpdateSupportUserRequest):
SupportUser {
        val user = userRepository.findById(ObjectId(userId))
            .orElseThrow {
                IllegalArgumentException("User with id $userId not found")
            }

        val updatedUser = user.copy(
            email = supportUser.email ?: user.email,
            firstName = supportUser.firstName ?: user.firstName,
            lastName = supportUser.lastName ?: user.lastName,
            role = supportUser.role ?: user.role,
            userPassword = supportUser.password ?: user.userPassword
        )

        return userRepository.save(updatedUser)
    }

    fun deleteUser(id: String) = userRepository.deleteById(ObjectId(id))

    fun getUserByEmail(email: String): SupportUser =
userRepository.findByEmail(email)
        ?: throw IllegalArgumentException("User with email $email not
found")

    fun search(request: SearchEmployeeRequest): Page<SupportUser> {
        return customMongoSupportUserRepository.searchUsers(
            request.email,
            request.firstName,
            request.lastName,
```

```
            request.role,
            PageRequest.of(request.page!!, request.size!!)
        )
    }
}
```