Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Проектування і програмна реалізація мобільного додатку для підприємства з надання логістичних послуг»

Виконав: студент групи ІПЗ19-1
Спеціальність 121 Інженерія програмного забезпечення
Зелінський Володимир Олегович
(прізвище та ініціали)
Керівник к. т. н., доцент Чупілко Т.А.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Дніпропетровський державний університет внутрішніх справ
(місце роботи)
доцент кафедри інформаційних технологій
(посада)
к. т. н., доцент Косиченко О.О.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2023

АНОТАЦІЯ

Зелінський В.О. Проектування і програмна реалізація мобільного додатку для підприємства з надання логістичних послуг.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2023.

Обсяг роботи складає 66 сторінок, 16 рисунків, 17 використаних джерел літератури, 2 додатки.

Мета проекту: спроектувати та розробити програмний засіб, що вирішить проблему надання логістичних послуг для підприємства.

Проаналізовано наявні програми які вирішують описану проблему, описано їх переваги та недоліки та сформульовано вимоги й задачі, яким повинна відповідати система.

Розробку системи розбито на підмодулі. Змодельовано та реалізовано архітектуру взаємодії всіх підсистем в рамках цілісної системи, що дозволяє легко розширювати та масштабувати продукт. Описано можливі варіанти взаємодії користувача з програмною системою.

У першому розділі наведено теоретичні відомості про використані інструменти для вирішення задачі, проведено аналіз та порівняльну характеристику технологій для розробки проектованого ПЗ, зроблено вибір інструментів для вирішення поставленого завдання на основі проведеного аналізу.

У другому розділі проведено системний аналіз, проведено опис предметної області, проведено огляд та аналіз існуючих аналогів, технічне та системне забезпечення розробки.

У третьому розділі проведено практичну реалізацію проектованої системи, надано інформаційне, технічне та системне забезпечення розробки.

Ключові слова: Інформаційна система, Java, MYSQL, мобільний додаток, база даних, веб-додаток, клієнт-серверна архітектура, зручний інтерфейс.

ANNOTATION

Zelenskyi V.O. Design and software implementation of a mobile application for enterprises providing logistics services.

Qualification work for obtaining a bachelor's degree in specialty 121 "Software Engineering". - University of Customs and Finance, Dnipro, 2023.

The volume of the thesis is 66 pages, 16 drawings, 17 sources used, 2 additions.

The purpose of the graduation project is to design and develop software that will solve the problem of providing logistics services for the enterprise.

Programs that solve the described problem have been analyzed, their advantages and disadvantages have been described, and requirements and tasks that the system should meet have been formulated.

The functionality of the system is divided into submodules. The architecture of interaction of all subsystems within the integrated system has been modeled and implemented, which allows easy expansion and scaling of the product. Possible options for user interaction with the software system are described.

The first chapter provides theoretical information about the tools used to solve the problem, an analysis and comparative characteristics of technologies for developing the designed software are carried out, and tools for solving the task are selected based on the analysis.

The second chapter provides a system analysis, describes the subject area, provides an overview and analysis of analytical analogues, technical and system software development.

The third chapter provides practical implementation of the designed system, provides information, technical and system software development.

Keywords: Information system, Java, MYSQL, mobile application, database, web application, client-server architecture, convenient interface.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЕН – експрес накладна

РМО – робоче місце оператора

ПЗ – програмне забезпечення

МД – мобільний додаток

UI – User Interface

SPA – Single-Page Applications

UML – Unified Modeling Language

HTTP – Hyper Text Transfer Protocol

ВСТУП

Актуальність роботи з проектування та програмної реалізації мобільного додатку для підприємства з надання логістичних послуг є вкрай важливою у сучасному світі. З постійним зростанням глобалізації та електронної комерції, попит на ефективні логістичні рішення зростає. Підприємства з надання логістичних послуг шукають інноваційні та зручні способи керування своїми операціями, відстеження вантажів та забезпечення високої якості обслуговування.

Мобільні додатки стають невід'ємною частиною логістичних підприємств, оскільки вони забезпечують мобільність, швидкий доступ до інформації та зручний спосіб комунікації з клієнтами та співробітниками. Завдяки такому додатку, підприємство може оптимізувати свої процеси, зменшити час доставки, покращити службу підтримки та підвищити задоволення клієнтів.

Незважаючи на існування аналогічних додатків для підприємств з надання логістичних послуг, цей проект має свою унікальну новизну. Тут поєднано новітні технології, а також високорівневі мови програмування такі як Java, Spring Framework і бази даних, з інноваційним підходом до проектування і реалізації мобільного додатку. У проекті передбачено ефективну архітектуру та розширену функціональність, що відрізняє цей проект від інших рішень такого плану у цій сфері.

Практична цінність цього проекту полягає в тому, що мобільний додаток, спроектований та розроблений для підприємства з надання логістичних послуг, допомагає автоматизувати та оптимізувати ряд процесів. Він спрощує роботу користувача з логістичними задачами, забезпечує швидкий доступ до інформації про статус доставки, дозволяє здійснювати сканування посилок та багато іншого. Цей додаток надає зручні та ефективні інструменти для управління логістичними процесами та покращення якості обслуговування.

Таким чином, робота з проектування та програмної реалізації мобільного додатку для підприємства з надання логістичних послуг є дуже актуальною і має великий потенціал для покращення ефективності та конкурентоспроможності логістичних підприємств. Цей проект відповідає не лише за сучасні потреби логістичної індустрії, але й пропонує інноваційні рішення для вирішення складних задач.

Метою роботи є програмна реалізація програмного забезпечення мобільного додатку для підприємства з надання логістичних послуг є створення інноваційного та ефективного інструменту, який сприятиме автоматизації та оптимізації логістичних процесів підприємства. Для більш детального розуміння нижче наведено список основних завдань цієї роботи:

1. Забезпечення зручного та швидкого доступу до інформації: Мобільний додаток має надавати користувачам можливість отримувати оперативну інформацію про стан вантажів, транспорту, розклади доставки та інші логістичні дані. Це допоможе підприємству зберігати інформацію в централізованому вигляді та забезпечить швидкий обмін даними між різними структурними підрозділами.

2. Покращення ефективності операцій: Додаток повинен допомогти управляти різними логістичними операціями, такими як складування, транспортування, вантажоперевезення та вивантаження. Шляхом автоматизації цих процесів та забезпечення точності даних, додаток дозволить підприємству підвищити продуктивність, знизити час доставки та збільшити рівень обслуговування.

3. Покращення комунікації та співпраці: Мобільний додаток має забезпечувати зручну комунікацію між співробітниками підприємства та клієнтами. Це дозволить зменшити час на вирішення проблем та взаємодію замовників, сприятиме покращенню якості обслуговування та підвищенню клієнтської задоволеності.

4. Забезпечення безпеки та захисту даних: Однією з важливих мет роботи є забезпечення безпеки та захисту конфіденційності даних. Додаток повинен

мати механізми аутентифікації та авторизації користувачів, захищені протоколи передачі даних та захист від несанкціонованого доступу. Крім того, має бути забезпечена можливість резервного копіювання та відновлення даних для запобігання втраті інформації.

5. Підтримка інтеграції з іншими системами: Додаток повинен бути здатним інтегруватися з існуючими логістичними системами, такими як системи управління складом, системи відстеження вантажів та системи обліку транспорту. Це забезпечить плавний обмін даними та підвищить ефективність всієї логістичної інфраструктури.

6. Постійне вдосконалення та підтримка: Розроблений мобільний додаток повинен бути готовим до масштабування та змін. Метою є забезпечення постійного вдосконалення функціональності, врахування змін в вимогах користувачів та технологій, а також підтримка та вирішення можливих проблем та помилок.

Таким чином, мета роботи полягає у створенні мобільного додатку, який сприятиме оптимізації логістичних процесів підприємства, поліпшенні комунікації та співпраці, забезпеченні безпеки даних та підтримці інтеграції з іншими системами.

Переддипломна практика на робочому місці показала існуючий брак досконалості схожих систем на ринку та в цілому серед вже розроблених програм такого плану. Тому була поставлена мета спроектувати і розробити більш якісний додаток хоч і вужчої спеціалізації, оскільки головним користувачем має бути представник підприємства з надання логістичних послуг.

Предметом дослідження є програмна реалізація програмного забезпечення мобільного додатку для підприємства з надання логістичних послуг є сам мобільний додаток і його функціональні можливості, структура та архітектура, процес розробки та впровадження.

Об'єктом дослідження є підприємство з надання логістичних послуг, його потреби, вимоги та процеси, які можуть бути оптимізовані та поліпшені за допомогою мобільного додатку. До об'єкта дослідження також відносяться

користувачі додатку - логістичні менеджери, водії, клієнти та інші, які взаємодіють з додатком та мають вплив на його функціональність та ефективність.

Основні аспекти дослідження включають вивчення потреб та вимог підприємства, аналіз існуючих рішень та систем логістики, розробку архітектури додатку, вибір технологій та інструментів розробки, проектування і реалізацію функцій додатку, тестування, впровадження та підтримку.

Місце роботи серед аналогів – спроектований і розроблений у цій роботі додаток має бути кращим, за існуючі аналоги через свою простоту логічної структури, а також більш зручніший та легший до сприйняття інтерфейс користувача.

Постановка завдання проектування і програмної реалізації мобільного додатку для підприємства з надання логістичних послуг включає наступні складові:

1. Опис потреби: Визначення основних потреб і вимог підприємства з надання логістичних послуг, які можуть бути задоволені за допомогою мобільного додатку. Це можуть бути функції, які полегшують відстеження вантажу, керування транспортом, оптимізація маршрутів, сповіщення про статуси доставки тощо.

2. Аналіз конкурентів: Вивчення і аналіз існуючих аналогічних систем та додатків у галузі логістики для виявлення їх переваг і недоліків. Це дозволить визначити конкурентні переваги та унікальні функціональні можливості, які потрібно реалізувати у розроблюваному додатку.

3. Визначення функціональності: Опис детальної функціональності, яку повинен мати мобільний додаток, включаючи основні функції, можливості, взаємодію з базою даних, роботу з сервером, обмін даними з іншими системами тощо. Це можуть бути такі функції, як авторизація користувачів, відстеження вантажу, створення заявок на доставку, генерація звітів тощо.

4. Проектування архітектури: Визначення структури та архітектури мобільного додатку, включаючи вибір підходящих технологій та фреймворків, проектування бази даних, створення діаграм класів та взаємодій.

5. Розробка і реалізація: Написання програмного коду, реалізація функціональності, тестування та налагодження мобільного додатку. В цьому етапі використовуються мова програмування Java, веб-фреймворк Spring, база даних для зберігання і обробки інформації, а також інші технології, необхідні для реалізації задуманої функціональності.

6. Тестування і валідація: Проведення ряду тестів, які перевіряють працездатність та стабільність мобільного додатку. Валідація забезпечує відповідність розробленого додатку вимогам та очікуванням користувачів.

7. Впровадження та підтримка: Установка та використання мобільного додатку на реальних пристроях користувачів, надання підтримки та вирішення проблем, що виникають у процесі експлуатації.

Таким чином, завдання проектування і програмної реалізації мобільного додатку для підприємства з надання логістичних послуг включає всі етапи розробки від аналізу потреби до впровадження та підтримки. Основною метою є створення функціонального та ефективного додатку, який сприятиме оптимізації логістичних процесів, полегшенню роботи підприємства та підвищенню задоволення користувачів.

Проектування та програмна реалізація мобільного додатку для підприємства з надання логістичних послуг має великий потенціал для покращення продуктивності, зниження витрат та підвищення клієнтської задоволеності. Крім того, впровадження такого додатку може допомогти підприємству бути конкурентоспроможними на ринку логістичних послуг та зайняти провідні позиції у галузі.

Пояснювальна записка містить: анотацію, перелік умовних позначень, вступ, 3 основні розділи, висновок, список використаних джерел.

РОЗДІЛ 1
АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ МОБІЛЬНОГО ДОДАТКУ

1.1 Аналіз існуючих систем для реалізації проекту

Логістичні послуги є необхідною складовою багатьох підприємств, що займаються постачанням товарів та послуг. У зв'язку зі зростанням конкуренції і швидкими змінами в бізнес-середовищі, ефективне управління логістичними процесами стає все більш важливим для забезпечення успіху підприємства. Одним з ефективних інструментів управління логістикою є використання мобільних додатків, які спрощують доступ до інформації та оптимізують роботу співробітників.

Мобільний додаток для підприємства з надання логістичних послуг є програмним забезпеченням, розробленим з метою поліпшення ефективності та точності логістичних процесів, забезпечення зручного та швидкого доступу до інформації та оптимізації роботи співробітників. Основні завдання, які вирішуються за допомогою такого додатку, включають автоматизацію та оптимізацію логістичних процесів, покращення ефективності роботи системи, забезпечення швидкості доступу до інформації та впровадження інноваційних функцій[1].

Існуючі системи:

- Delivery App
- Logistics Management System (LMS)
- Fleet Management App
- Inventory Management App
- CargoTrack

1.2 Delivery App

Ця система має велику популярність серед підприємств з надання логістичних послуг. Вона надає зручний інтерфейс для замовлення та відстеження доставок, має вбудовані функції оплати і оцінювання. Проте, вона не завжди підходить для більших підприємств зі складними логістичними вимогами.

Delivery App - це зазвичай інтерактивний мобільний додаток, який забезпечує замовлення та доставку товарів або послуг. Додаток може мати два інтерфейси: для користувачів, які роблять замовлення, і для кур'єрів або постачальників послуг, які виконують доставку.

Основні функції та можливості, які можуть бути присутніми в Delivery App, включають:

1. Реєстрація користувачів: Користувачі можуть створювати облікові записи, заповнюючи необхідну інформацію, таку як ім'я, контактні дані та адреса доставки.
2. Вибір товарів або послуг: Користувачі мають можливість переглядати каталог товарів або послуг, вибирати потрібні позиції і додавати їх до свого замовлення.
3. Оформлення замовлення: Користувачі можуть здійснювати оформлення замовлення, обираючи спосіб оплати та вказуючи деталі доставки, такі як адреса та бажаний час доставки.
4. Відстеження замовлення: Користувачам надається можливість відстежувати статус свого замовлення в реальному часі, включаючи підтвердження замовлення, підготовку товарів або послуг до доставки та очікуваний час прибуття.

1. Комунікація з кур'єром: у додатку можуть бути функції спілкування або контакту з кур'єром, наприклад, через чат або телефон, щоб користувачі могли отримувати актуальну інформацію про своє замовлення або вирішувати питання, пов'язані з доставкою.

6. Оцінка та відгуки: Після завершення доставки користувачам може бути надана можливість оцінити якість обслуговування та залишити відгук про додаток або про конкретний замовлення. Це допомагає покращувати якість обслуговування та надає іншим користувачам інформацію при прийнятті рішення щодо замовлення.
7. Управління кур'єрами та доставками: для ефективного управління доставками додаток може мати панель адміністратора, де адміністратори можуть керувати кур'єрами, розподіляти замовлення, відстежувати роботу кур'єрів та оптимізувати процес доставки.
8. Платіжна система: Додаток може включати платіжну систему, яка дозволяє користувачам здійснювати оплату за замовлення безпосередньо через додаток. Це може включати в себе використання кредитних карт, електронних платіжних систем або платіжних гаманців.
9. Розширені можливості: Залежно від конкретної системи Delivery App, можуть бути доступні додаткові функціональні можливості, такі як використання геолокації для точного визначення місця доставки, програма лояльності для залучення і утримання клієнтів, інтеграція з соціальними мережами для спільного використання замовлень та багато іншого.

Варто відзначити, що реалізація Delivery App може варіюватись в залежності від конкретних вимог та потреб.

1.3 Logistics Management System (LMS)

Logistics Management System (LMS) - це програмна система, призначена для управління логістичними процесами в підприємстві з надання логістичних послуг. LMS допомагає автоматизувати та оптимізувати різні етапи логістичного ланцюга, включаючи замовлення, складські операції, вантажні перевезення, відстеження вантажів, облік і фінансові операції.

Основні функції та можливості, які можуть бути присутні в Logistics Management System, включають:

1. Замовлення та управління замовленнями: Система дозволяє здійснювати прийом замовлень від клієнтів та ефективно керувати ними. Це включає реєстрацію замовлення, обробку деталей, розподіл завдань між відповідальними працівниками та контроль виконання замовлень.
2. Управління складськими операціями: Система дозволяє вести облік товарів на складі, контролювати рух товарів, здійснювати інвентаризацію, виконувати вимоги до зберігання товарів та оптимізувати процеси зберігання та комплектації замовлень.
3. Управління транспортними перевезеннями: Система надає засоби для планування та координації вантажних перевезень. Вона дозволяє відстежувати рух транспортних засобів, оптимізувати маршрутизацію, контролювати терміни доставки та забезпечувати ефективну комунікацію з водіями.
4. Відстеження вантажів: Система надає можливість відстежувати місцезнаходження вантажів протягом всього логістичного ланцюга. Це дозволяє клієнтам та підприємству отримувати актуальну інформацію про стан вантажів, їх рух і очікуваний час доставки. Інформація може бути представлена у реальному часі через спеціальні онлайн-інтерфейси або мобільні додатки.
5. Облік та фінансові операції: Logistics Management System забезпечує можливість обліку та контролю фінансових операцій, пов'язаних з логістичними послугами. Це може включати виставлення рахунків, оплату, управління розрахунками з клієнтами та постачальниками послуг.
6. Звітність та аналітика: Система надає засоби для створення звітів, аналізу даних та оцінки продуктивності логістичних процесів. Це дозволяє підприємству отримувати усвідомлені рішення щодо оптимізації процесів, виявлення проблемних ситуацій та вдосконалення ефективності.
7. Інтеграція з іншими системами: Logistics Management System може бути інтегрована з іншими бізнес-системами, такими як система управління відносинами з клієнтами (CRM), система управління складом (WMS) або

система управління виробництвом (ERP). Це сприяє покращенню обміну даними та синхронізації інформації між різними департаментами підприємства.

Враховуйте, що Logistics Management System може мати різні функціональні можливості, залежно від конкретного розробника або постачальника системи. При виборі LMS важливо враховувати потреби вашого підприємства та забезпечити відповідність системи вимогам вашого бізнесу [2].

1.4 Fleet Management App

Fleet Management App (FMA), або додаток для управління автопарком, є комплексною програмною системою, створеною для оптимізації та автоматизації операцій з управління автопарком у підприємствах, що мають власний автопарк. Ось детальніше про основні функціональні можливості Fleet Management App:

1. Відстеження та моніторинг транспорту: FMA використовує технологію GPS для відстеження та моніторингу автотранспорту в режимі реального часу. Вона надає детальну інформацію про місцезнаходження автомобілів, швидкість руху та історію маршрутів. Ця функція дозволяє керівникам автопарку мати повну видимість над автопарком, забезпечуючи ефективне використання ресурсів та покращення продуктивності.

2. Обслуговування та ремонт: FMA допомагає у керуванні графіками обслуговування та ремонту транспорту. Вона надає нагадування про рутинні технічні обслуговування, такі як заміна масла, перевороти шин та технічні перевірки, на основі пробігу або часових інтервалів. Додаток також дозволяє фіксувати та відстежувати історію ремонтів, що сприяє превентивному обслуговуванню та зменшує ризик непередбачених поломок.

3. Управління паливом: FMA допомагає контролювати споживання палива та оптимізувати його використання. Вона відстежує паливні транзакції, пробіг та паливну ефективність для кожного автомобіля в автопарку. Аналізуючи ці дані,

керівники автопарку можуть виявити споживання палива, оптимізувати маршрути та впроваджувати економічні заходи для зниження витрат на паливо. Додаток також може надавати інформацію про ціни на паливо в різних заправних станціях, що дозволяє вибрати найвигідніші варіанти заправки.

4. Моніторинг водіїв та безпека: FMA включає функцію моніторингу водіїв, що дозволяє відстежувати їх поведінку на дорозі, таку як швидкість, гальмування та прискорення. Це допомагає виявляти небезпечні стиль водіння та сприяє безпеці на дорозі. Крім того, додаток може надавати інформацію про технічний стан автомобілів, такий як рівень палива, тиск у шинах, стан акумулятора тощо, що допомагає уникнути аварійних ситуацій через несправності автомобілів.

5. Аналітика та звітність: FMA забезпечує різноманітні аналітичні звіти та статистику щодо використання автопарку. Вона може надавати інформацію про пробіг, час руху, споживання палива, вартість обслуговування, ефективність водіїв та інші показники. Ці дані допомагають здійснювати стратегічне планування, приймати рішення щодо покращення ефективності автопарку та зменшення витрат.

Fleet Management App є потужним інструментом для підприємств, що мають власний автопарк. Вона спрощує та автоматизує багато аспектів управління автопарком, забезпечуючи зниження витрат, покращення продуктивності та безпеки на дорозі. Цей додаток допомагає зберегти час, гроші та ресурси, які раніше були витрачені на ручне ведення та координацію автопарку. Додаток також сприяє зменшенню непередбачених затримок та поломок, оскільки він допомагає виявляти проблеми з транспортними засобами наперед, дозволяючи проводити регулярне обслуговування та ремонт.

Загалом, Fleet Management App є важливим інструментом для підприємств з власним автопарком, оскільки вона допомагає автоматизувати та оптимізувати процеси управління автопарком, забезпечуючи зниження витрат, підвищення безпеки та ефективності. Цей додаток використовується в різних галузях, включаючи логістику, доставку, будівництво, громадський транспорт та багато

інших, допомагаючи оптимізувати управління автопарком для досягнення успіху[3].

1.5 Inventory Management App

Inventory Management App (додаток для управління запасами) - це програмне забезпечення, створене для ефективного контролю, організації та оптимізації процесів управління запасами в підприємствах. Основна мета цього додатку - забезпечити належний рівень запасів, зменшити втрати і покращити продуктивність підприємства. Ось детальніше про функціональні можливості Inventory Management App:

1. Система відстеження запасів: Додаток дозволяє точно відстежувати кількість та місцезнаходження всіх товарів в підприємстві. Він зберігає інформацію про товари, включаючи назву, опис, кількість, одиницю виміру, ціну та дату отримання/передачі. Це дозволяє підприємству мати повну видимість щодо наявних запасів, уникати нестачі або переплати, а також вчасно поповнювати запаси при необхідності.

2. Замовлення та постачання: Додаток допомагає автоматизувати процеси замовлення та постачання товарів. Він може використовувати дані про кількість запасів та рівень попиту, щоб генерувати автоматичні замовлення або нагадування про необхідність поповнення запасів. Це дозволяє забезпечити наявність товарів у необхідний час і уникнути затримок у постачанні.

3. Оптимізація запасів: Додаток аналізує дані про споживання, попит та прогнозує майбутні потреби у товарах. Він допомагає підприємству оптимізувати розмір запасів, уникати зайвих запасів або нестачі. Це дозволяє зменшити витрати на утримання запасів та підвищити ефективність управління запасами.

4. Інвентаризація та перевірка: Додаток дозволяє проводити інвентаризацію запасів швидко і точно. Він забезпечує можливість сканування штрих-кодів або використання RFID технологій для швидкого і точного

визначення кількості товарів. Це допомагає уникнути помилок, що виникають під час ручного ведення інвентаризації та дозволяє знизити час, необхідний для перевірки запасів.

5. Аналітика та звітність: Додаток надає аналітичні звіти та статистику про стан запасів, витрати на запаси, оборотність запасів та інші ключові показники. Це допомагає керівникам приймати обґрунтовані рішення щодо управління запасами, виявляти можливості для ефективності та зниження витрат.

6. Інтеграція з іншими системами: Додаток може інтегруватися з іншими системами, такими як системи управління продажами або системи управління виробництвом. Це дозволяє автоматично синхронізувати дані про продажі, виробництво та споживання, що сприяє більш точному та ефективному управлінню запасами.

Inventory Management App є незамінним інструментом для підприємств, які мають склади та великі обсяги запасів. Вона допомагає уникнути непотрібних витрат, покращити контроль над запасами та забезпечити наявність товарів в потрібний час. Додаток може використовуватися в різних галузях, включаючи роздрібну торгівлю, виробництво, логістику та інші сектори[4].

1.6 CargoTrack

CargoTrack - це інноваційна система, спеціально розроблена для управління та моніторингу логістичних процесів у підприємстві з надання логістичних послуг. Основні функціональні можливості системи включають:

1. Замовлення та розподіл: CargoTrack дозволяє замовляти послуги логістичних перевезень та ефективно розподіляти їх між різними водіями та транспортними засобами. Вона надає зручний інтерфейс для введення замовлень, вибору оптимальних маршрутів та розподілу навантажень.

2. Моніторинг та відстеження: Система пропонує функціонал для постійного моніторингу руху транспортних засобів, стану доставок та виконання

замовлень. Це дозволяє підприємству та його клієнтам отримувати реальний час інформацію про місцезнаходження вантажу та очікуваний час доставки.[9]

3. Електронна документація: CargoTrack забезпечує можливість генерації та обміну електронною документацією, такою як рахунки, товарні накладні та інші необхідні документи. Це спрощує процеси обліку, фінансового звітності та взаємодії з клієнтами.

4. Звіти та аналітика: Система забезпечує широкі можливості створення звітів та аналізу даних щодо логістичних процесів. Вона дозволяє генерувати звіти про виконані замовлення, витрати на транспорт, продуктивність перевезень та інші ключові показники. Це дозволяє підприємству аналізувати ефективність своїх операцій, виявляти потенційні проблеми та приймати обґрунтовані рішення для поліпшення логістичних процесів.

5. Інтеграція з іншими системами: CargoTrack може бути інтегрована з іншими існуючими системами у підприємстві, такими як системи управління складами, системи обліку, системи електронної комерції тощо. Це забезпечує безперебійний обмін даними та оптимізує робочі процеси.

У порівнянні з іншими схожими системами, CargoTrack відрізняється своїм інтуїтивно зрозумілим та відносно простим інтерфейсом користувача, який спрощує навчання та використання системи. Крім того, система пропонує широкий спектр функціональних можливостей, які відповідають потребам підприємства з надання логістичних послуг. Вона підтримує масштабування і розширення, що дозволяє адаптуватись до зростаючих потреб бізнесу. Нижче (рисунок 1.1) буде зображена порівняльна таблиця схожих систем, де зображені певні переваги окремих систем в порівнянні з іншими. Для порівняння були обрані такі аспекти: Функціональність, технології що були використані, мобільна підтримка, інтеграція, зручність інтерфейсу. Ці параметри є базовими, для визначення переваг чи недоліків для додатку.

Аспекти	**Logistic Management System**	**CargoTrack**	**Fleet Management App**	**Inventory Management App**
Функціональність	Відстеження товарів, управління запасами, планування маршрутів, обробка замовлень	Відстеження вантажів, управління доставкою, графік руху, звітність	Управління флотом автомобілів, слідкування за технічним станом, планування обслуговування	Управління запасами, відстеження товарів, оптимізація замовлень, звітність
Технології	Java, Spring Framework, Hibernate	PHP, Laravel	Java, Spring Boot, MySQL	Python, Django, PostgreSQL
Мобільна підтримка	Нативний мобільний додаток для Android та iOS	Веб-додаток, доступний на мобільних пристроях через браузер	Нативний мобільний додаток для Android та iOS	Веб-додаток, доступний на мобільних пристроях через браузер
Інтеграція	Інтеграція зі сторонніми системами, API для обміну даними	Інтеграція з логістичними партнерами, API для обміну даними	Інтеграція з системами GPS, API для обміну даними	Інтеграція з POS-системами, API для обміну даними
Зручний інтерфейс	Простий і інтуїтивний інтерфейс, настроювання робочого простору	Сучасний і зрозумілий інтерфейс, персоналізація вигляду	Зручний і практичний інтерфейс, налаштування за вимогами користувача	Простий і легкий у використанні інтерфейс, можливість настроювання

Рисунок 1.1 – Порівняльна таблиця схожих систем

Виходячи із наведених даних можна зрозуміти що кожна із схожих систем є повноцінною, корисною програмою зі своїми перевагами та недоліками, кожна має персонально розроблений інтерфейс користувача, і всі вони мають підтримку API що дуже гарно при користуванні.

1.7 Висновки до першого розділу

У данному розділі було проведено аналіз засобів реалізації для проектування та розробки мобільного додатку з надання логістичних послуг. У процесі дослідження були вивчені схожі системи, такі як Delivery App, Logistics Management System (LMS), Fleet Management App, Inventory Management App і CargoTrack.

Виявлено, що всі згадані системи володіють певними перевагами і можуть бути використані в розробці мобільного додатку для логістичних послуг. Вони надають різноманітний функціонал, який включає авторизацію користувача, керування транспортними засобами, інвентаризацію, відстеження поставок та багато іншого.

Однак, після детального аналізу, було визначено, що жодна з цих систем не відповідає повністю всім вимогам та особливостям проекту, що розглядається. Тому, для досягнення поставлених цілей, було прийнято рішення розробити власний мобільний додаток, використовуючи мову програмування Java, веб-фреймворк Spring і базу даних, описану раніше.

Це рішення обумовлено необхідністю розробки додатку, який буде повністю відповідати специфіці логістичного підприємства та надавати унікальний функціонал, що підтримується вищезгаданими системами.

Отже, на основі проведеного аналізу, можна зробити висновок, що розробка власного мобільного додатку на основі Java, Spring framework і вище описаної бази даних є оптимальним рішенням для задачі надання логістичних послуг. Це дозволить забезпечити потрібну функціональність, гнучкість та ефективність додатку, враховуючи специфікацію завдання. Застосування інструментів, які були успішно використані у схожих системах, дозволить забезпечити зручний інтерфейс, надійну авторизацію користувачів, ефективне сканування та обробку даних, а також моніторинг та аналіз результатів.

Аналіз схожих систем, таких як Delivery App, Logistics Management System (LMS), Fleet Management App, Inventory Management App і CargoTrack, надав

багато цінних висновків та рекомендацій для вдосконалення функціоналу та унікальності розроблюваного додатку. Застосування успішних практик та ідей з цих систем дозволить створити ефективний, зручний у використанні та конкурентоспроможний мобільний додаток для підприємства з логістичними послугами.

Основні завдання проектування і програмної реалізації мобільного додатку включають розробку функціоналу, встановлення системи авторизації користувача, реалізацію меню сканування з різними опціями, розробку інтерфейсу користувача та інтеграцію з сервером та базою даних. Дотримання цих завдань сприятиме створенню потужного та функціонального додатку, що забезпечить ефективну роботу підприємства з надання логістичних послуг.

Крім того, проведений аналіз схожих систем дав змогу оцінити їх переваги та недоліки, що були враховані при проектуванні та розробці власного мобільного додатку. Це дозволило покращити функціонал та унікальність системи, щоб вона відповідала конкретним потребам підприємства з надання логістичних послуг.

Таким чином, розробка і програмна реалізація мобільного додатку для підприємства з надання логістичних послуг на базі Java, Spring framework та вище описаної бази даних є актуальним та ефективним рішенням. Цей додаток дозволить підприємству забезпечити ефективну логістичну систему, підвищити продуктивність та забезпечити задоволення клієнтів.

Продовження роботи над проектуванням та програмною реалізацією мобільного додатку дозволить реалізувати задуману систему, здійснити її тестування та впровадження в роботу підприємства. Цей додаток має потенціал покращити логістичні процеси, забезпечити ефективне керування ресурсами та зробити підприємство більш конкурентоспроможним на ринку логістичних послуг.

РОЗДІЛ 2
ВІДОМОСТІ ПРО ВИКОРИСТАНІ ІНСТРУМЕНТИ

2.1 Програмне забезпечення. Опис вхідної / вихідної інформації

Програмне забезпечення мобільного додатку для логістичного підприємства вирішує основні задачі проблематика яких пов'язана з наданням послуг і обробкою даних потрібних працівникам. Наприклад:

- Оптимізація логістичних процесів
- Зручний доступ до робочої системи працівникам
- Покращення ефективності роботи системи
- Забезпечення швидкості доступу до інформації
- Впровадження інноваційних функцій до системи

Мобільний додаток має працювати безперебійно, мати низькі вимоги до апарату користувача для швидкості його роботи щоб у працівників не з'являлося труднощів у користуванні, а також повинен містити логічний та легкий інтерфейс користувача [5].

Додаток дозволить отримувати працівникам підприємства актуальну інформацію про логістичні операції, у режимі реального часу дозволить проводити маніпуляції над ЕН, що дозволить значно краще розподіляти робочий ресурс а також вплине на загальну продуктивність. Для вирішення подібного плану задач користувач повинен мати вхідну інформацію [6].

Повний функціонал системи вимагає на вхід такі дані: логін та пароль співробітника для пропуску в систему, відділення, на якому в робочий час зафіксовано працівника, оскільки без фіксації часу не буде доступу до програми, і перелік завдань які можна виконати використовуючи сканер [7].

На вихід користувач отримує екран звітності, де він може бачити в режимі реального часу що пішло не так при виконанні будь-якої операції сканування. Після цього користувач зможе прийняти дії для вирішення цих проблем.

2.2 Опис підсистем програмного комплексу

Перелік необхідних підсистем:

- Система управління базами даних;
- Система користувачів;
- Система зчитування вхідної інформації
- Система виведення результатів обробки інформації
- Система експорту вихідних даних програми

2.2.1 Система управління базами даних

Система управління базою даних (СУБД) отримує інструкцію від адміністратора бази даних і, відповідно, інструктує систему про необхідність внесення необхідних змін.

У запрограмованій системі мобільного додатку для підприємства з надання логістичних послуг СУБД використовується для завантаження, вилучення або зміни існуючих даних з системи.База даних вирішує задачу зберігання даних проекту у кабінеті користувача та зберігання особистих даних користувача таких як: логін, пароль, ім'я. СУБД завжди забезпечує незалежність даних. Будь-які зміни в механізмі зберігання виконуються без зміни всієї програми.[9]

2.2.2 Система користувачів

Розроблена система користувачів включає в себе автентифікацію та авторизацію.

Під час реєстрації створюється особистий обліковий запис у системі, з метою отримання доступу до повного функціоналу. У системі мобільного додатку зареєстрованим користувачам надається можливість користування повним функціоналом програми і відображення всіх модулів системи.

Автентифікація це основа безпеки системи, яка полягає в перевірці достовірності даних про користувача сервером. Процес перевіряє користувача, а саме: правильність введення логіну та пароля [8].

Під час процесу авторизації відбувається перевірка прав користувача і визначається можливість доступу в систему. Для реалізації такої системи можна створити окремий клас, в додатку А наведено приклад класу, що можна використати для авторизації у розроблюваному програмному забезпеченні.

В цьому прикладі було створено клас `UserSystem`, який управляє списком користувачів. Він містить методи для додавання, видалення, пошуку користувачів за іменем, перевірки наявності вже зайнятого імені користувача та аутентифікації користувача за іменем і паролем. Клас `User` представляє користувача зі змінними для імені та пароля.

У головному класі `Main` створюємо екземпляр `UserSystem`, додаємо кілька користувачів та демонструємо виклик методів для перевірки наявності користувача за іменем та аутентифікації користувач.

2.2.3 Система зчитування вхідних даних користувача

Система додавання вхідних даних користувачем надає можливість обрати необхідну функцію МД яка потрібна йому саме в той чи інший етап роботи, будь то робоче місце з обробки вантажу, де працівник може прийняти відправлення або видати замовлення клієнту, або це буде функція сканування відправлень чи вивантаження авто, де потрібно у відомість сканування зчитати сканером кожен штрих-код посилки яка приїхала, чи буде їхати з поточного підрозділу. Таким чином існує декілька написаних функцій, одна з яких буде використовувати камеру пристрою користувача для зчитування штрих-кодів, інша буде конвертувати отриману інформацію, та розшифровувати штрих-коди для передачі основної інформації далі та використовувати у роботі. Також вхідними даними можуть бути інформація, яка має бути необхідною для сканування. Наприклад, якщо користувачеві треба створити групування відправлень, для

цього додаток перш ніж створити палету запросить певні вхідні дані, а саме: бейдж скануючого, напрямок, куди вантаж буде прямувати у поточну хвилю, а також штрих-код самої тари, в яку буде відбуватися сканування. [10]

2.2.4 Система виведення результатів обробки інформації

Система має виводити користувачеві звіт програми в результаті виконаної одної з його дій, тобто якщо це було вивантаження, має бути вивід на екран залишку посилок, які мали приїхати цим рейсом, чи все було вивантажено чи щось було не помічене в авто, якщо це групування вантажу до відправки то при завантаженні працівник має змогу бачити що саме було відсканована до поточної відомості сканування, і що саме має поїхати з підрозділу і у який час, адже система налаштована таким чином, що посилки які теоретично вже не встигнуть запакувати до поточного рейсу автоматично переходять до наступного. Тобто виконуючи одну з дій сканування користувач має змогу відслідковувати та контролювати послідовність дій. [10]

2.2.5 Система експорту вихідних даних програми

Система надає можливість експорту вихідних даних до бази даних і серверів із попереднім збереженням даних. Також на сервері міститься інформація про користувача, його логін та пароль, для входу до МД. Більше того, якийсь час після виконання будь-якої з операцій сканування остаточні звіти будуть зберігатися на сервері, і користувач матиме змогу перейти до них та переглянути за потреби. Дані будуть зберігатися до автоматичного стирання з серверу що відбуватиметься наступного дня. [11]

2.3 Використовувані мови програмування

У програмному забезпеченні мобільного додатку для підприємства з надання логістичних послуг використовується новітня мова програмування Java, а також фреймворк Spring. Також для розширення функціоналу ПЗ були використані бази даних та робота з сервером.

2.3.1 Java

Java - це потужна і популярна мова програмування, яка була розроблена компанією Sun Microsystems (згодом придбаною Oracle Corporation) у 1995 році. Java стала однією з найбільш використовуваних мов програмування завдяки своїй надійності, переносимості та широкому спектру застосувань.

Ось деякі з основних переваг мови програмування Java:

1. Переносимість: Java базується на концепції "Write Once, Run Anywhere" (WORA), що означає, що програми, написані на Java, можуть працювати на різних платформах, таких як Windows, macOS і Linux, без необхідності виконувати зміни у вихідному коді.

2. Об'єктно-орієнтоване програмування: Java підтримує концепцію об'єктно-орієнтованого програмування (ООП), що дозволяє створювати модульні, розширювані і повторно використовувані кодові блоки. Це сприяє покращенню якості програми, спрощенню розробки і підтримці коду.

3. Багатопоточність: Java надає потужні засоби для роботи з багатопоточними програмами. Розробники можуть легко створювати та керувати багатопоточними додатками для ефективної використання обчислювальних ресурсів та покращення відгуку програми.

4. Безпека: Java має вбудовану систему безпеки, яка дозволяє запускати програми в обмеженому середовищі (Java Virtual Machine - JVM) з контролем доступу до ресурсів комп'ютера. Це дозволяє запобігати вразливостям та атакам з боку зловмисників.

5. Велика бібліотека: Java має велику стандартну бібліотеку, яка містить багато готових класів і методів для різних завдань, таких як робота з файлами, мережевими операціями, роботою з базами даних, графікою та багато іншого. Це спрощує розробку додатків, оскільки багато рутинних завдань вже реалізовано і можна легко використовувати готові компоненти.

6. Підтримка розробки: Java має потужні інтегровані середовища розробки (Integrated Development Environments - IDEs), такі як Eclipse, IntelliJ IDEA і NetBeans, які надають зручні інструменти для написання, налагодження і тестування коду. Це сприяє підвищенню продуктивності розробників і полегшує роботу з проектами.

7. Велике співтовариство розробників: Java має велике співтовариство розробників, яке надає підтримку, документацію, навчальні матеріали та відповіді на запитання. Це означає, що завжди є ресурси та підтримка для вирішення проблем та розширення знань.

8. Широке застосування: Java використовується у багатьох галузях, включаючи веб-розробку, мобільні додатки, вбудовані системи, фінансові послуги, наукові дослідження і багато інших. Це дозволяє розробникам мати різноманітні можливості кар'єрного зростання і працювати в різних сферах.

Загалом, Java є потужною, переносною та надійною мовою програмування, яка має велику спільноту розробників та широкі можливості застосування. Вона підходить для розробки різноманітних додатків і є одним із лідерів у сфері програмування.

2.3.2 Frameworks

Spring Framework є одним з найпопулярніших і розширених фреймворків для розробки Java-додатків. Він надає комплексні інструменти та інфраструктуру для розробки різноманітних додатків, включаючи веб-додатки, мікросервіси, додатки для операційних систем та інше. Ось деякі ключові компоненти та можливості Spring Framework:

1. Inversion of Control (IoC): Spring використовує підхід IoC, що дозволяє керувати створенням та керуванням об'єктами в додатку. Замість того, щоб розробник самостійно створював об'єкти, Spring бере на себе відповідність за створення та керування об'єктами через контейнер IoC. Це спрощує розробку та покращує модульність та тестируваність додатку.

2. Dependency Injection (DI): Spring надає механізми для ін'єкції залежностей, які дозволяють зменшити пряму залежність між компонентами додатку. Завдяки DI реалізується слабке зв'язування, що полегшує зміни в структурі додатку і поліпшує тестування.

3. Spring MVC: Spring надає підтримку для розробки веб-додатків за допомогою патерну Model-View-Controller (MVC). Spring MVC дозволяє легко створювати контролери, представлення та модель даних для обробки HTTP-запитів і створення відповідей для клієнтів. Це дозволяє ефективно створювати веб-додатки з доброю організацією коду і розділенням обов'язків.

4. Spring Data: Spring Data дозволяє спростити доступ до різних джерел даних, таких як реляційні бази даних, NoSQL-сховища, кеші та інші. Він надає спільний ін терфейс для взаємодії з даними, який дозволяє розробникам зосередитись на бізнес-логіці додатку, не затримуючись на деталях доступу до даних.

5. Spring Security: Spring Security забезпечує механізми аутентифікації, авторизації та керування безпекою для веб-додатків. Він надає розширюваність і конфігурованість для встановлення прав доступу, обробки авторизаційних токенів, захисту від атак та багато іншого.

Це лише деякі з компонентів та можливостей Spring Framework. Все це робить Spring потужним фреймворком для розробки різноманітних додатків на Java з підтримкою багатьох аспектів розробки, таких як веб-розробка, доступ до даних, безпека та розподілені системи.[12]

2.4 Опис структури БД для мобільного додатку

Для мобільного додатку з надання логістичних послуг, яке включає авторизацію користувача та меню сканування з 11 пунктами, можна розглянути наступну структуру бази даних:

1. Таблиця "Користувачі":
   - user_id (первинний ключ)
   - username
   - password
   - role (роль користувача: адміністратор, оператор, клієнт тощо)
2. Таблиця "Сканування":
   - scan_id (первинний ключ)
   - user_id (зовнішній ключ, пов'язаний з полем user_id в таблиці "Користувачі")
   - scan_type (тип сканування: Досканувати, транспортування, міське авто, міжміське авто тощо)
   - scan_datetime (дата та час сканування)
3. Таблиця "Машини клієнтів":
   - car_id (первинний ключ)
   - user_id (зовнішній ключ, пов'язаний з полем user_id в таблиці "Користувачі")
   - unloading_status (статус вивантаження машини клієнта)
4. Таблиця "Зони сканування":
   - zone_id (первинний ключ)
   - zone_name (назва зони сканування)
5. Таблиця "Комірки":
   - cell_id (первинний ключ)
   - cell_name (назва комірки)
   - cell_status (статус комірки: вільна, зайнята)
6. Таблиця "Інші операції":

- operation_id (первинний ключ)

- user_id (зовнішній ключ, пов'язаний з полем user_id в таблиці "Користувачі")

- operation_name (назва операції)

Це лише загальна структура бази даних для мобільного додатку, і вона може бути розширена або змінена залежно від конкретних вимог та функціональності додатку[13].

2.5 Висновки до другого розділу

У розділі "Теоретичні відомості про використані інструменти для вирішення задачі" було розглянуто ключові інструменти, які використовуються при проектуванні та розробці мобільного додатку для надання логістичних послуг. Ці інструменти включають мову програмування Java, Spring Framework та базу даних.

Мова програмування Java була обрана з багатьох причин. Вона є потужною, об'єктно-орієнтованою мовою програмування, яка надає широкі можливості для розробки мобільних додатків. Використання Java дозволяє створювати надійні та масштабовані додатки, що є важливими для логістичних підприємств, які операційно активні та мають велику кількість транзакцій та даних.

Spring Framework є одним з найпопулярніших фреймворків для розробки Java додатків. Він надає якісні інструменти для створення високопродуктивних, модульних та легко розширюваних додатків. Завдяки Spring Framework ми можемо ефективно керувати залежностями, реалізувати архітектурні шаблони та забезпечити безпеку та масштабованість нашого додатку.

Описана база даних включає таблиці, що відображають основні сутності і функціональність системи. Ця база даних дозволяє зберігати та керувати інформацією про користувачів, транспортні засоби, сканування, вивантаження, палети, передачу зміни та переміщення в комірку. Використання бази даних

дозволяє забезпечити стійкість та цілісність даних, швидкий доступ до необхідної інформації та зручне управління даними у всіх аспектах розробки мобільного додатку для надання логістичних послуг. Крім того, база даних може бути легко розширена або модифікована для задоволення майбутніх потреб підприємства.

У цьому розділі було розглянуто теоретичні аспекти інструментів, які використовуються в проектуванні та розробці мобільного додатку для надання логістичних послуг. Мова програмування Java надає потужність та гнучкість для створення високопродуктивних та надійних додатків. Spring Framework допомагає управляти залежностями, підтримувати модульність та забезпечувати безпеку та масштабованість додатку. База даних забезпечує ефективне зберігання та управління даними, що використовуються в системі [14].

В результаті проведеного аналізу було виявлено, що використання мови програмування Java є відповідним вибором для розробки мобільного додатку, оскільки вона є потужною, широко використовуваною та має велику спільноту розробників. Використання Spring framework дозволить забезпечити ефективну роботу з різними компонентами додатку, такими як контролери, сервіси та репозиторії.

Також було виявлено, що база даних, описана у попередніх розділах, відповідає потребам збереження та керування даними, необхідними для функціонування мобільного додатку. Використання схожих систем, таких як Delivery App, Logistics Management System (LMS), Fleet Management App, Inventory Management App і CargoTrack, надає цінні ідеї та рекомендації для покращення функціоналу та унікальності розроблюваного додатку.

Висновок розділу "Теоретичні відомості про використані інструменти для вирішення задачі" демонструє, що обрані інструменти, такі як мова програмування Java, Spring Framework та база даних, є ефективними та потужними для розробки мобільного додатку для надання логістичних послуг. Їх використання дозволяє забезпечити функціональність, масштабованість, безпеку та ефективну роботу системи.

РОЗДІЛ 3
ЗАСОБИ РОЗРОБКИ

3.1 Опис програмної реалізації

Мобільний додаток має допомагати користувачу із вирішенням базових задач швидко і безперебійно. Для цього потрібно три основні модулі, з яких і буде складатися додаток, кожен з яких буде розбиватися на функціональні підблоки, загальна кількість яких тринадцять. Головним модулем додатку буде модуль сканування вантажу.

На рисунку 3.1 наведена схема структури додатку, на якій розташовані всі програмні модулі.

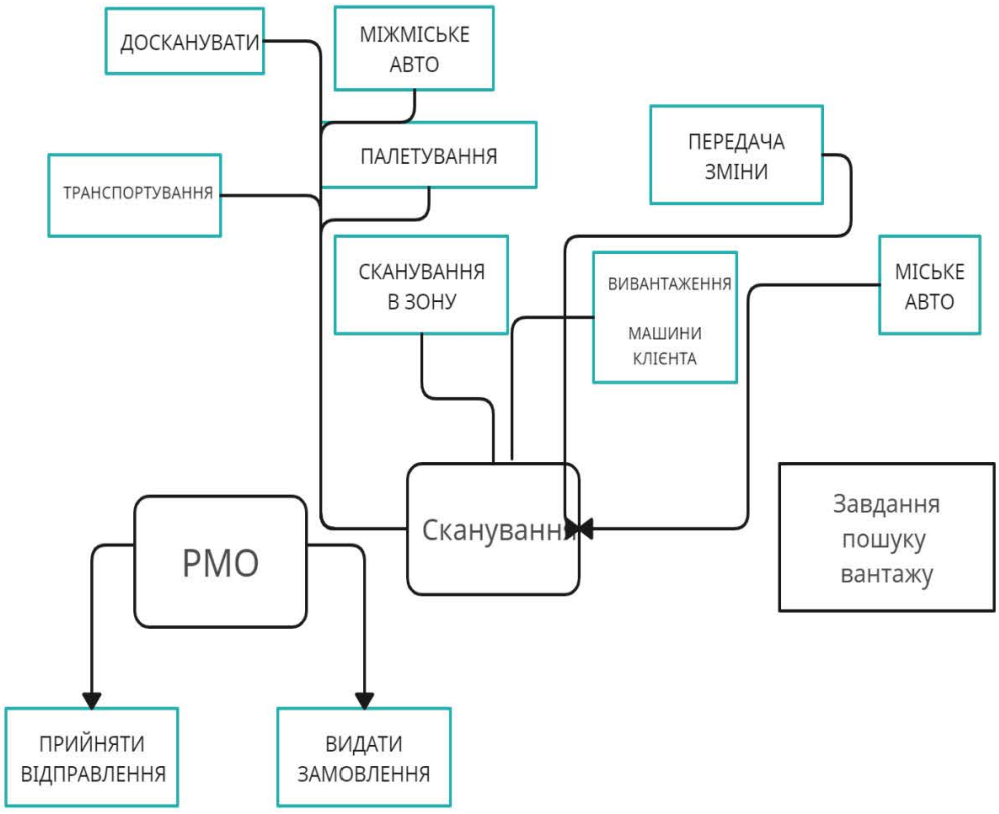

Рисунок 3.1 – Схема структури додатку

Для кожного з необхідних модулів та задач системи було проаналізовано наявні варіанти вирішення. Серед них було обрано ті, що будуть задовольняти наступні умови:

- Безперешкодна інтеграція підсистем між собою
- Швидкодія
- Можливість оптимізації під конкретну область.

Наприклад для створення модуля додавання вхідних даних користувачем та підсистеми, що відповідає за зчитування вхідної інформації було вирішено використати зчитувальний елемент самого пристрою користувача, тобто для сканування використовується за наявності основний сканер, якщо додаток було запущено на сканері, або якщо це смартфон, додаток використає в такому разі його камеру.

Виходячи з кінцевої цілі, користувач обирає потрібну для себе дію додатку.

3.2 Вимоги до програмного і апаратного забезпечення

Програмне забезпечення мобільного додатку має займати не великий об'єм пам'яті для того, щоб мати змогу запуску на будь-якому мобільному пристрої користувача. Мінімальні вимоги для працездатності додатку:

- Мобільний телефон із системою Android 5.0 та більш пізні версії
- Наявність камери, та дозволу на її використання в налаштуваннях
- Доступ до місцезнаходження
- Доступ до сховища пам'яті пристрою
- Доступ до мережі інтернет
- Отримання даних з мережі інтернет
- Play install referrer API

Виконуючи ці умови користувач зможе повною мірою використовувати весь функціонал додатку на своєму пристрої.

3.3 Опис функціональності системи

Мобільний додаток містить у собі одного користувача, який перш ніж буде мати доступ до програми, має пройти авторизацію.

У Уніфікованій мові моделювання (UML) діаграма прецедентів може узагальнити деталі користувачів вашої системи (також відомих як актори) та їх взаємодії з системою. Щоб побудувати її потрібно використовувати набір спеціалізованих символівта роз'ємів. Коректно побудована діаграма допоможе представити:

- Сценарії, в яких система чи додаток взаємодіють з людьми, організаціями чи зовнішніми системами;
- Цілі, які система чи додаток допомагає досягти цим організаціям (відомим як актори);
- Обсяг системи.

Діаграма прецедентів не вкладається в багато деталей - наприклад, не змоделюєпорядок виконання кроків. Натомість діаграма прецедентів належного використання зображує огляд взаємозв'язку між випадками використання, акторами та системами на високому рівні.

UML - це інструментарій моделювання, який можна використовувати для побудови діаграм. Корпуси для використання представлені з міткою овальної форми.Фігурні палички представляють дійових осіб у процесі, а участь актора в системі моделюється лінією між актором та випадком використання. Щоб зобразити межу системи, потрібно намалювати поле навколо самого випадку використання.

Загальні компоненти діаграми прецедентів:

Актори: Користувачі, які взаємодіють із системою. Актором може бути людина, організація або зовнішня система, яка взаємодіє з програмою чи системою. Вони повинні бути зовнішніми об'єктами, які виробляють або споживають дані.

Система: конкретна послідовність дій та взаємодій між суб'єктами та системою. Система може також називатися сценарієм.

Цілі: кінцевий результат більшості випадків використання. Успішна діаграма повинна описувати діяльність та варіанти, які використовуються для досягнення мети. Кейси використання: овали в горизонтальній формі, які представляють різні види використання, які може мати користувач.

Асоціації: лінія між учасниками та випадки використання. У складних діаграмах важливо знати, з якими акторами пов'язані дані випадки використання.

Системні граничні поля: Коробка, яка встановлює область системи для використання справ. Усі випадки використання поза межами поля розглядаються за межами цієї системи.

Нижче, на рисунку 3.2 зображено діаграму прецедентів для розробленого мобільного додатку.

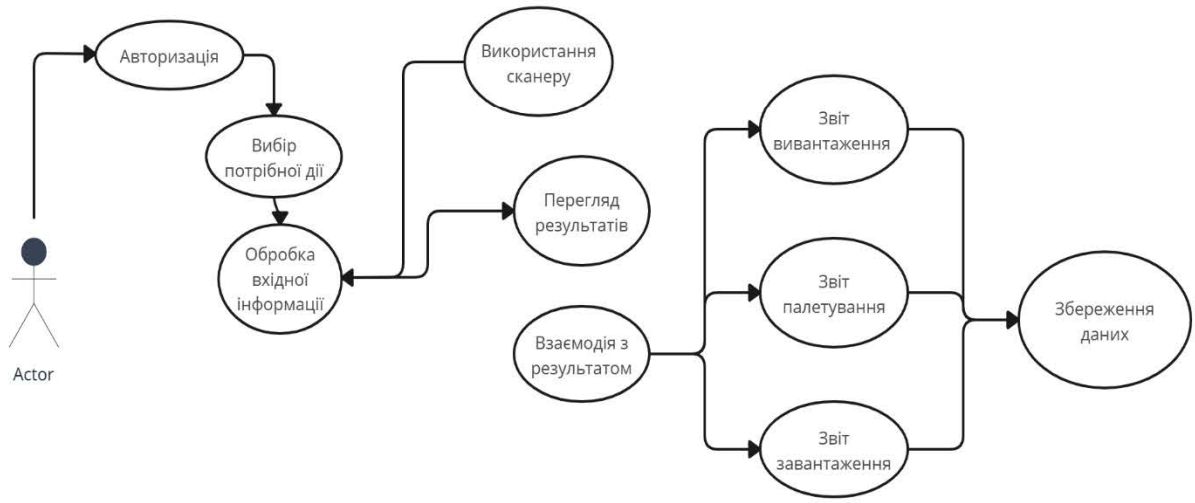

Рисунок 3.2 – Діаграма прецедентів

Першим кроком, що бачить користувач, коли заходить до додатку це вікно авторизації. Тут йому необхідно вписати у поля логін та пароль персональні дані для входу до системи. Далі, якщо працівник зафіксований на робочому місці, тобто його робочий день почато, програма пропустить його до основного меню, де він має зробити вибір необхідної дії, яка йому необхідна в певний момент від

додатку. На екрані він бачить три варіанти роботи системи. Перший – РМО (Робоче місце оператора), тут працівник має змогу використовувати додаток як звичайне робоче місце, тут можна видати посилку або зробити відправлення. Другий – сканування, це усі дії, задля виконання яких працівнику може знадобитися сканер, тут маємо 11 пунктів, а саме:

- Досканувати
- Транспортування
- Міське авто
- Міжміське авто
- Вивантаження машини клієнта
- Сканування в зону
- Палетування
- Передача зміни
- Переміщення в комірку
- Інші операції
- Вихід

І третій варіант – Завдання пошуку вантажу. Тут працівник одразу в режимі реального часу може спостерігати всі помилки свого підрозділу з логістичних послуг. Тобто, якщо якась накладна (посилка) поїхала не своєчасно, або приїхала до підрозділу пізніше плану, або під час однієї з операцій сканування якась із накладних не відсканувалась, працівник одразу побачить це у звіті пошуку вантажу.

3.4 Архітектура додатку

Спілкування користувача з системою відбувається за моделлю клієнт — сервер. Тобто сервер не може надсилати ніякі дані клієнту без попереднього запиту. Сервер не зберігає ніяких проміжних даних про клієнта, при кожному запиті клієнт має передавати повну інформацію про себе. Це дозволяє системі

буди незалежною від кількості серверів, придатною до швидкого масштабування та гнучкою.

Сервер відповідає за бізнес-логіку додатку. Він не зберігає даних, для цього система під'єднана до SQL бази даних. Алгоритми обробки даних також зберігаються на сервері, ще одним компонентом сервера є система автентифікації.

Додаток має архітектуру моноліту. Це означає що всі модулі додатку тісно пов'язані між собою. Через це програма містить менше коду, але це унеможливлює розділення її на окремі компоненти. Цей підхід використовують в роботі у невеликих командах.

Рисунок 3.3 Клієнт – серверна архітектура

MVT (Model View Template) - це модель дизайну програмного забезпечення. Це колекція трьох важливих компонентів Перегляд, Модель та Шаблон.

Модель допомагає обробляти базу даних. Це рівень доступу до даних, який обробляє дані.

Шаблон - це презентаційний шар, який повністю обробляє частину інтерфейсу користувача. Перегляд використовується для виконання бізнес-логіки та взаємодії з моделлю для перенесення даних та візуалізації шаблону.

Немає окремого контролера і повний додаток заснований на перегляді, моделі та шаблоні. Ось чому його називають додатком MVT.

На рисунку деталізовано серверну архітектуру MVT фрейморку Spring на основі якого побудовано систему мобільного додатку для логістичного підприємства. Рисунок показує MVT на основі керуючого потоку. Тут користувач запитує ресурс на Spring, що в свою чергу працює як контролер і перевіряє наявний ресурс у URL. Якщо URL-карти відображаються, вигляд, який взаємодіє з моделлю та шаблоном, відображає шаблон. Spring відповідає користувачеві і надсилає шаблон як відповідь.[7]

3.5 Опис роботи мобільного додатку

1. Авторизація користувача. Вхід в систему відбувається для користувача за допомогою введення логіну та паролю. Нижче на рисунку 3.4 представлено вигляд вікна входу. Передбачається, що користувач мобільного додатку для підприємства з надання логістичних послуг уже зареєстрований до системи, має власний логін та пароль. Для логіну в таких операціях зазвичай використовують персональну інформацію користувача, таку, яка буде завжди у доступі, і неможливо забути, а пароль він здатен обрати сам, сервіс зі зміни паролю не пов'язано з додатком. Додаток лише має звірити вхідну інформацію із вже існуючим списком.

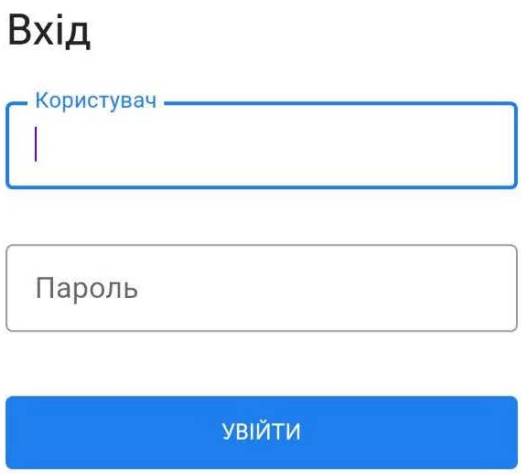

Рисунок 3.4 – Вікно авторизації у додатку

2. Після введення логіну та пароля, якщо все вірно, користувач потрапляє до головного меню додатку, який має простий інтерфейс, і представляє користувачу на вибір одну з трьох основних задач цього додатку. Нижче на рисунку 3.5 показано меню додатку, де користувач має змогу обрати потрібну для нього дію

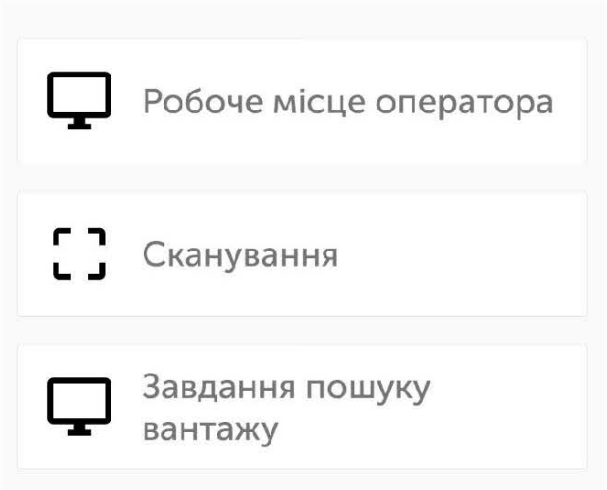

Рисунок 3.5 – Головне меню додатку

3. Якщо користувач обирає перший варіант, тобто перехід до РМО (Робоче місце оператора) він потрапляє до іншого меню, що буде зображено нижче, на рисунку 3.6, там знову користувач має буде обрати відповідно критичну

для себе дію додатку в необхідний момент, будь то відправлення вантажу чи видача клієнту.

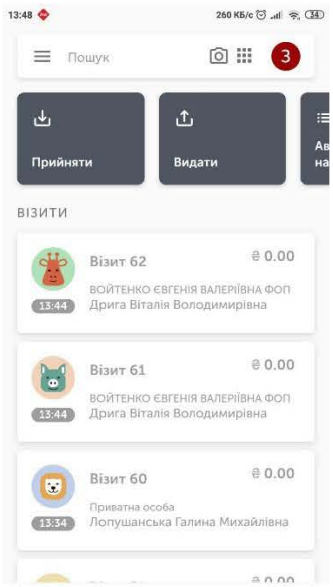

Рисунок 3.6 – Вигляд РМО (Робоче місце оператора)

4. Натиснувши прийняти, додаток запропонує у вигляді логічної послідовності від користувача дані для обробки, які потім будуть занесені до ЕН (експрес накладної) і згодом після пакування (за необхідності) на відправлення буде приклеєно маркування, що буде містити вхідні дані користувача. Нижче, на рисунку 3.7 представлено вигляд вхідних даних користувача, та послідовну структуру, як додаток пропонує введення вхідних даних користувачеві. Все йде послідовно, спершу – відправник, інформація про того, хто виконує відправлення. Відправникові не обов'язково вносити в це поле свої дані, все буде залежати від політики компанії з надання логістичних послуг. Далі оформлення інформації про відправлення. Тут треба зазначити що існує автоматичне розподілення типів відправки, а саме: документи, до 1кг і посилки, від 2 і вище. За цим параметром також буде залежати від політики логістичної компанії. Далі –

інформація про отримувача, і його дані, номер телефона, ПІБ, місто, та номер відділення, адреси, тощо.

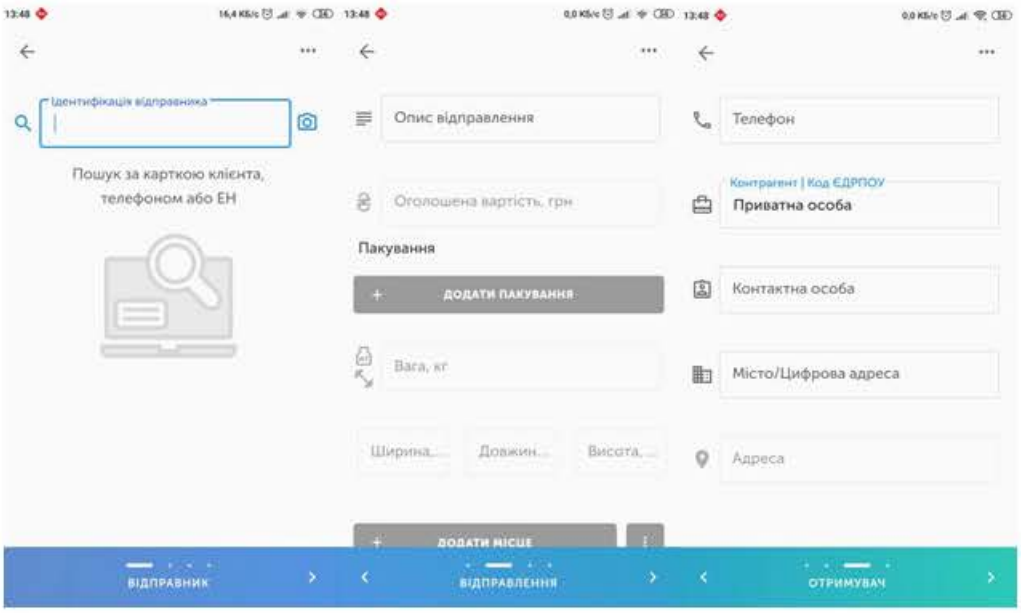

Рисунок 3.7 – Вхідні дані для відправки

На рисунку вище зображені необхідні вхідні дані від користувача для додатку аби прийняти відправлення, далі інформація заноситься до бази даних, і генерується персональний ключ, у якому буде міститися ця інформація, уже цей ключ і буде нанесений на маркування для цього відправлення і буде використовуватися у всьому логістичному ланцюжку для відстеження відправлення.

5. Якщо в РМО обрати інший пункт, видача відправлення, додаток запропонує один із варіантів прийняття вхідних даних, або можна ввести номер телефона отримувача, або персональний ключ посилки, тобто ЕН, або також можна ввести прізвище клієнта. На рисунку нижче, 3.8 зображено реалізацію видачі замовлень. Далі при введені даних клієнта на екрані з'явиться перелік доступних посилок для видачі клієнту, а також місце полички, де вони знаходяться, із вказаним номером комірки, в яку ті були вивантажені, або переміщені, або якщо з ними відбувалась іще будь-яка операція сканування вантажу.

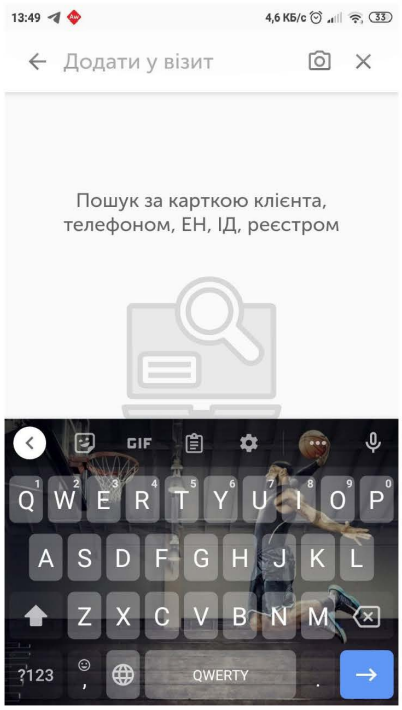

Рисунок 3.8 – Видача замовлення в РМО

Також вже видані замовлення зберігаються у файлі поточна зміна, і доступні до відображення користувачеві, у вигляді списку. Користувач матиме змогу скористуватися цими даними за потреби протягом деякого часу після проведення візиту. Нижче на рисунку 3.9 буде зображений список вже виданих замовлень та їх відображення у додатку, у розділі РМО(Робоче місце оператора).

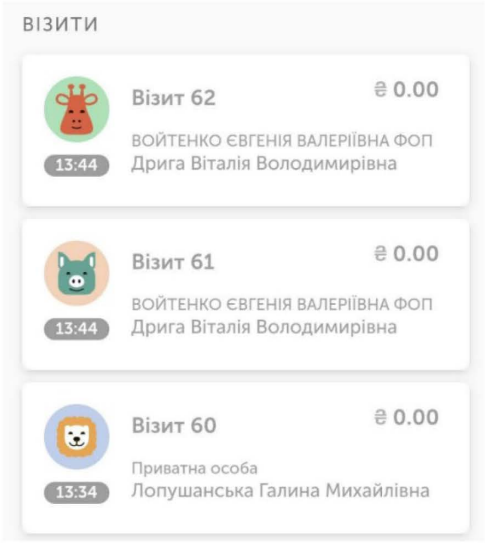

Рисунок 3.9 – Список вже виданих замовлень

6. Обравши другий пункт головного меню додатку, а саме Сканування, користувач також перейде до іншого меню, де буде відображено ще 11 пунктів завдань, які пропонує застосунок. Нижче на рисунку 3.10 зображено меню сканування. Кожен з обраного пункту несе свою дію в мобільному додатку. Наприклад функція транспортування дозволяє створити користувачеві відомість сканування для дії вивантаження/завантаження. Для створення нової відомості сканування користувачеві необхідні будуть вхідні дані, а саме: бейдж скануючого, штрих-код тари, і напрямок (якщо завантаження). У результаті цього все, що буде відсканировано у відомість, буде записано вихідною інформацією до бази даних, а пізніше потрапить на сервер для зберігання даних щодо поточного рейсу. А функція досканувати створена для швидкого переходу режиму сканування з палетування, для вже створених палет, тобто для групування тари, якщо тару вже створено, то вже не потрібно при наступному скануванні знов створювати палетування, можна зайти у режим досканувати, і обрати вже існуючу палету.

Рисунок 3.10 – Меню сканування мобільного додатку

Для реалізації сканеру будемо використовувати бібліотеку ZXing (Zebra Crossing), далі нам треба буде імпортувати необхідні класи:

```
import com.google.zxing.BarcodeFormat;
import com.google.zxing.DecodeHintType;
import com.google.zxing.MultiFormatReader;
import com.google.zxing.Result;
import com.google.zxing.client.j2se.BufferedImageLuminanceSource;
import com.google.zxing.common.HybridBinarizer;
```

Наступним кроком для реалізації буде функція, для зчитування штрих-кодів з використанням оригінальної камери використовуваного пристрою, на який встановлено додаток.

Далі потрібно буде відправити зчитані дані до серверу. Для цього можна використовувати клієнтський API, такий як java.net.HttpUrlConnection, для відправки POST-запиту на сервер з даними штрих-коду. Для цього також потрібна функція. Приклад описаних вище функцій міститься у додатку Б.

Тепер, викликаючи функцію scanBarcode() можна звертатися до алгоритму для зчитування штрих-кодів з камери, або алгоритм використає апаратний сканер, якщо такий є. А також викликаючи sendBarcodeDataToServer() можна відправляти дані, отримані зі сканеру на сервер. Нижче на рисунку 3.11 показано приклад роботи сканеру який працює використовуючи камеру девайсу, на який встановлено мобільний додаток.

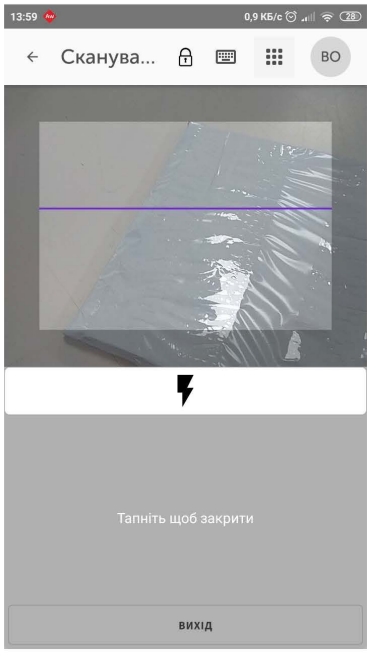

Рисунок 3.11 – Приклад роботи камери в режимі сканеру

Також існують обмеження сканування. Сканер здатен розпізнати різницю, між скануванням персонального ключа ЕН і штрих-кодом палет. Тобто, створюючи групування тари, якщо працівник помилково відсканує, або під сканер випадково чи у його поле зору потрапить інший штрих-код з’явиться повідомлення на екрані. Нижче, на рисунку 3.12 зображено повідомлення про помилкове сканування, або сканування у інший розділ додатку. Крім того, передбачено перевірку на створення відомості сканування у вже раніше створену, тобто якщо користувач створив групування відправлень до тари, і пізніше намагається створити знову вже створену відомість додаток попередить що така вже існує. Додатком передбачені певні форс-мажорні ситуації, що можуть трапитись на підприємстві з надання логістичних послуг.

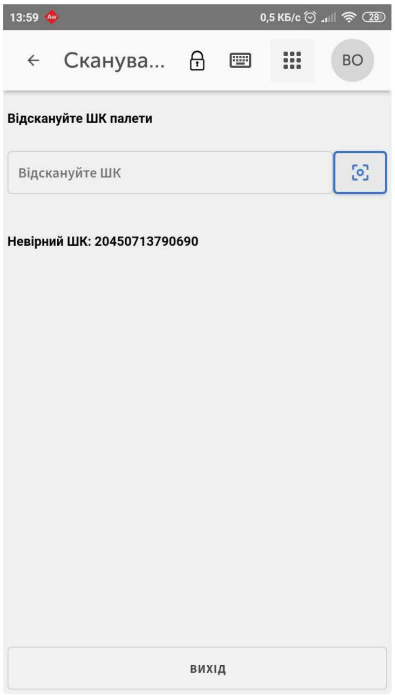

Рисунок 3.12 – Невірний штрих-код

Крім того, з метою протидії переповнення масивів інформації від зайвої, по типу сканування одного штрих-коду десятками разів, існує ще одне обмеження сканування у будь-яку поточну відомість. Для привернення уваги користувача при спробі повторного сканування хоча б одного, на екрані з’явиться повідомлення з написом: Увага! Присутнє сканування в поточну відомість, перевір відскановані раніше. Якщо все ж таки користувач впевнений, що штрих-код не було відскановано жодного разу, але сканер все одно видає таку помилку, можна примусово змусити зчитати саме цей штрих-код підтвердивши його бейджем скануючого. Нижче на рисунку 3.13 буде наведено приклад такого повідомлення. Проігнорувати дане повідомлення користувачеві не вдасться оскільки допоки він намагатиметься виконати сканування вже відсканованої накладної це повідомлення буде з’являтись на екрані і не давати змогу повторного сканування.

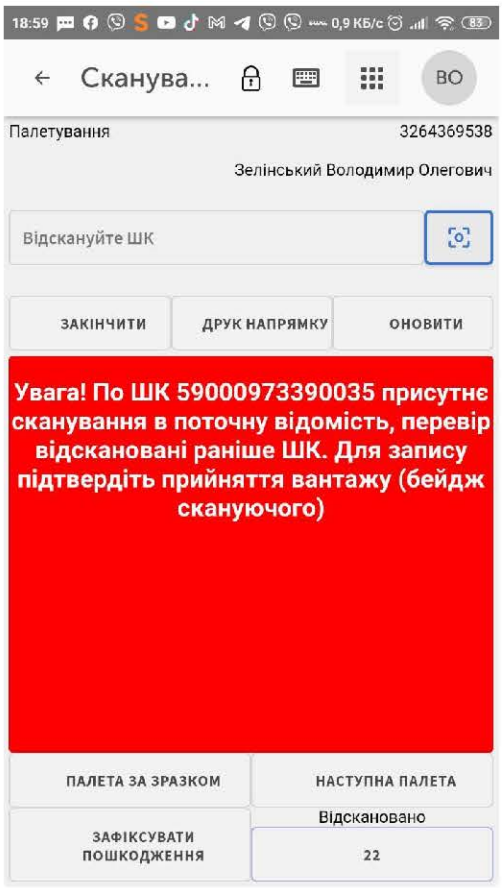

Рисунок 3.13 – Помилка подвійного сканування

Окрім того, на рисунку, зображеному вище, користувач може бачити додаткові функції, під час сканування, це і перелік відсканованих раніше, у вигляді списку, також можна зафіксувати пошкодження вантажу, створити палету за зразком – у такому випадку додаток скопіює дані, що необхідні для створення групування посилок, і створить ідентичну, наступну, якщо у попередній тарі вже немає місця, ще у режимі поточного сканування одразу можна перейти до іншої палети одразу, не виходячи зі сканеру. Також виконуючи операції сканування додаток дає змогу зафіксувати пошкодження вантажу. Для цього потрібно відсканувати накладну, далі обрати меню зафіксувати пошкодження, і з наявного переліку обрати потрібне пошкодження, далі знову відсканувати накладну вантажу і натиснути кнопку зафіксувати.

3.6 Тестування додатку

Забезпечення якості це діяльність по забезпеченню того, щоб розробник програмної системи надавав замовникам найкращий продукт або послугу. Quality Assurance (QA) фокусується на поліпшенні процесів доставки якісних продуктів. Організація повинна забезпечити ефективність і результативність процесів у відповідності зі стандартами якості, встановленими для програмних продуктів.

Для забезпечення стовідсоткової якості продукту необхідно проводити функціональне тестування за участі розробників та мануальне тестування за участі групи потенційних користувачів продукту. На рисунку 3.14 зображено життевий цикл розробки продукту з фазами тестування.

Рисунок 3.14 – Етапи тестування при розробці продукту

Функціональне тестування - це тип тестування програмного забезпечення, при якому система тестується на відповідність функціональним вимогам/специфікаціям. Функції перевіряються шляхом подачі на них вхідних даних і перевірки вихідних даних. Функціональне тестування гарантує, що вимоги належним чином задоволені додатком. Цей тип тестування пов'язаний не з тим, як відбувається обробка, а з результатами обробки. Він імітує фактичне використання системи, але не робить ніяких припущень про структуру системи. Під час функціонального тестування використовується метод Box Testing, що включає в себе White Box testing та Black Box testing рисунок 3.15.

White box testing - це тестування внутрішньої структури, дизайну і кодування програмного рішення. У цьому типі тестування код видно тестеру. Основна увага приділяється перевірці потоку вхідних і вихідних даних через додаток, поліпшенню дизайну та зручності використання, посилення безпеки. White Box testing зазвичай виконується розробниками, і включає тестування програмного коду для наступного:

- Внутрішні проблеми в безпеці;
- Зламаний або погано структурованіий код;
- Потік вхідних даних через код;
- Перевірка результату;
- Функціональність умовних циклів;

Тестування кожного оператора, об'єкта і функції на індивідуальній основі. Тестування може проводитися на системному, інтеграційному і модульному рівнях розробки програмного забезпечення. Однією з основних цілей тестування whitebox є перевірка робочого процесу для додатка. Він включає в себе тестування ряду зумовлених вхідних даних по відношенню до очікуваних або бажаних вихідних даних, так що коли конкретний ввід не призводить до очікуваного вихідному сигналу, ви зіткнулися з помилкою.

Black Box testing визначається як методика тестування, при якій функціональність тестованої програми тестується без урахування внутрішньої структури коду, деталей реалізації і знання внутрішніх шляхів програмного забезпечення. Цей тип тестування повністю заснований на вимогах і специфікаціях програмного забезпечення. У BlackBox Testing розробник фокусується на входах і виходах програмної системи, не турбуючись про внутрішні процеси цієї програми.

Black-Box може бути будь-якою програмною системою, яку потрібно протестувати. Наприклад, операційна система, така як Windows, веб-сайт, такий як Google, база даних, така як Oracle. У Black Box Testing є можливість

протестувати ці додатки, просто зосередившись на входах і виходах, не знаючи їх внутрішньої реалізації коду.

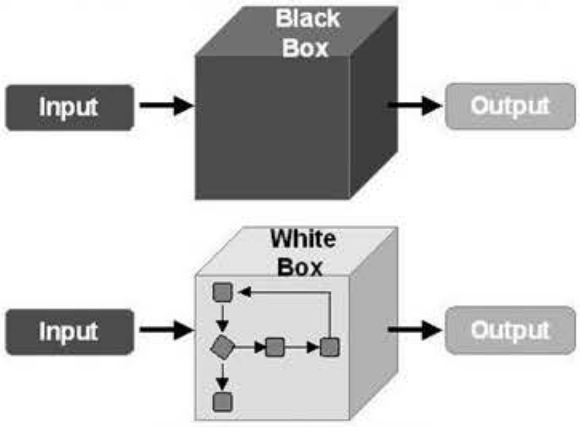

Рисунок 3.15 – Різниця між Black Box Testing та White Box testing

Функціональне тестування зазвичай виконується на рівнях системного тестування та приймального тестування.

Як правило, функціональне тестування включає в себе наступні етапи:

- Визначення функцій, які має виконувати програмне забезпечення;
- Створення вхідних даних на основі характеристик функції;
- Визначення висновків на основі характеристик функції;
- Виконання контрольного тесту;
- Порівняння фактичних та очікуваних результатів.

Функціональне тестування є більш ефективним, коли умови тестування створюються безпосередньо з вимог користувача або бізнесу. Коли умови тестування створюються з системної документації (системні вимоги/проектна документація), дефекти в цій документації не будуть виявлені в ході тестування, і це може стати причиною гніву кінцевих користувачів, коли вони будуть використовувати програмне забезпечення.

Користувацьке тестування, також відоме як бета-тестування або тестування кінцевим користувачем, визначається як тестування програмного

забезпечення користувачем або клієнтом, щоб визначити, чи може воно бути прийнято чи ні.

Це остаточне тестування, яке виконує після завершення функціонального тестування.

Основна мета цього тестування - перевірка відповідності програмного забезпечення вимогам бізнесу. Ця перевірка виконується кінцевими користувачами, знайомими з вимогами бізнесу.

Оскільки приймальний тест користувача є останнім тестом, який проводиться перед запуском програмного забезпечення, очевидно, що це останній шанс для замовника протестувати програмне забезпечення і виміряти, чи підходить він для цієї мети.

Кількість етапів, що виконуються у виконанні тесту на прийняття користувача, може варіюватися залежно від того, наскільки детально кожна команда хоче визначити кожен етап у процесі. Ці кроки зазвичай включають:

- Етап планування, де окреслені вимоги бізнесу та стратегія для UAT
- Ідентифікація та створення тестових сценаріїв. Ці тестові сценарії повинні охоплювати якомога більше функціональних випадків, з якими стикаються кінцеві користувачі.
- Вибираються групи для тестування. Розробники можуть вирішити, щоб декілька кінцевих користувачів тестували програмне забезпечення або відкривали тест ще багатьом, пропонуючи безкоштовну пробну версію в Інтернеті.
- Етап тестування та документації, коли кінцеві користувачі починають тестувати програмне забезпечення та реєструються будь-які потенційні помилки та інші проблеми.
- Фаза виправлення, коли команда розробників вносить корективи в код, вирішує будь-які помилки або вносить запропоновані зміни.

Після цього програмне забезпечення має бути готовим до випуску у своє виробниче середовище.

У процесі розробки системи мобільного додатку для підприємсва з надання логістичних послуг було проведено тестування.

Деякі випадки тест-кейсів для фукціонального тестування:

- Перевірка коректності завантаження/вивантаження поточного авто;
- Коректність функції групування відправлень а саме палетування;
- Перевірка наявності у користувача правильного логіну та пароля для входу до системи.

Для проведення користувацького тестування було розроблено тест-кейси та залучено п'ять потенційних користувачів програми для перевірки її якості, та для того, щоб переконатися, що програма задовольняє їх потреби.

Приклади тест-кейсів для користувацького тестування:

- Вибір камери;
- Перевірка роботи функцій відкриття сканеру на різних пристроях;
- Перевірка сканування різного типу штрих-кодів.

Тестування як функціональне так і користувацьке було проведено успішно. Знайдені помилки було налагоджено. Проведено тестування налагоджень, що показало що система працює вдало. Це свідчить про те, що мобільний додаток для підприємства з надання логістичних послуг розроблений і створений правильно і готовий до використання. Також після тестування виявлені помилки були успішно виправлені що дало змогу не допустити їх впровадження у подальшу логіку структури додатку і до самого робочого процесу користування додатком. Після проходження додатком повного переліку тестування було опрацьовано недоліки. Особливу увагу було приділено роботі сканеру, який використовує камеру смартфона. Цей функціонал дозволяє зчитувати штрих-коди та перетворювати їх на буквенний або цифровий вигляд. Приклад роботи сканера та результати його роботи були продемонстровані, що дало можливість оцінити ефективність і точність роботи даного функціоналу.

3.7 Висновки до третього розділу

У даному розділі було детально розглянуто засоби розробки, використані при проектуванні та програмній реалізації мобільного додатку для підприємства з надання логістичних послуг. Проект був реалізований на мові програмування Java з використанням Spring framework і бази даних, яка була розроблена з урахуванням потреб додатку.

Опис програмної реалізації дав можливість зрозуміти, як було створено додаток та як він функціонує. Схема структури додатку допомогла уявити загальну організацію компонентів та модулів, а опис функціональності системи розкривав основні можливості та завдання, що вирішуються додатком.

Діаграма прецедентів дозволила проаналізувати взаємодію користувачів з системою та визначити основні сценарії використання. Архітектура додатку надала уявлення про логічну та фізичну організацію компонентів, зв'язки між ними та використання відповідних шаблонів проектування.

Опис роботи та рисунки додатку надали можливість ознайомитися з виглядом та функціональними можливостями додатку. Рисунки головного меню та інших меню демонстрували структуру та організацію інтерфейсу, а приклади дії програми надали уявлення про взаємодію з користувачем та результати роботи додатку.

Особливу увагу було приділено роботі сканеру, який використовує камеру смартфона. Цей функціонал дозволяє зчитувати штрих-коди та перетворювати їх на буквенний або цифровий вигляд. Приклад роботи сканер сканера та результати його роботи були продемонстровані, що дало можливість оцінити ефективність і точність роботи даного функціоналу.

Останнім кроком було проведення тестування додатку. Це включало в себе виконання різних тестових сценаріїв з метою перевірки правильності роботи функцій, виявлення та виправлення можливих помилок. Тестування додатку дозволило переконатися в його стабільності та надійності.

Загальною метою цього розділу було описати та продемонструвати всі аспекти програмної реалізації мобільного додатку для підприємства з надання логістичних послуг. Він включав опис технологій, використаних для розробки, структуру додатку, функціональність, архітектуру, роботу та результати сканера, а також результати тестування. Всі ці аспекти спільно працюють для створення функціонального, зручного та надійного мобільного додатку, який забезпечує оптимальне управління та надання логістичних послуг підприємства.

У розділі було розглянуто функціонал додатку при діях користувача. Було наведено зображення екрану ключових кроків роботи додатку а також описані основні дії для роботи із застосунком.

Таким чином після встановлення додатку користувач має увімкнути його, авторизуватися, обрати з меню необхідний розділ додатку, після чого обрати вже потрібну дію додатку, для того аби використати весь функціонал застосунку. Отже з наведеного вище можна сказати що на виході отримуємо програму з простою логікою, для використання різних задач, від створення посилки до переміщення її між підрозділами чи на своєму підрозділі у комірки на полицях для зручності та зменшення витрати часу для пошуку посилки клієнту. Засновуючись на проведеному аналізі та дослідженні, можна зробити висновок, що розроблене програмне забезпечення відповідає поставленим цілям і завданням. Він забезпечує зручний і ефективний інтерфейс для користувачів, широкі можливості управління логістичними процесами, а також надійну та стабільну роботу.

В процесі розробки були використані сучасні інструменти та технології, такі як мова програмування Java, Spring framework та база даних. Це дозволило створити додаток, який має гнучку архітектуру та може легко розширюватися та адаптуватися до змінних потреб підприємства.

Один з ключових функціональних елементів додатку - сканер, забезпечує швидке та точне зчитування штрих-кодів з використанням камери смартфона. Це дозволяє зручно та ефективно виконувати операції, пов'язані з ідентифікацією товарів та контролем їх руху в системі.

ВИСНОВКИ

Під час виконання дипломної роботи була проведена робота з вивченням технологій розробки мобільного додатку з використанням технологій Java, MySQL, API, JavaScript. Робота на тему "Проектування і програмна реалізація мобільного додатку для підприємства з надання логістичних послуг" є результатом дослідження та розробки програмного засобу, спрямованого на вирішення проблеми надання логістичних послуг для підприємства.

У ході дослідження були проаналізовані існуючі програмні рішення, що вирішують описану проблему, визначені їх переваги та недоліки. На основі цього аналізу були сформульовані вимоги та задачі, яким повинна відповідати розроблювана система.

У роботі було розроблено програмне забезпечення мобільного додатку, яке надає функціонал зі зручним інтерфейсом для відстеження товарів, управління запасами, планування маршрутів та обробки замовлень. Додаток був розбитий на підмодулі, а архітектура взаємодії всіх підсистем була змодельована та реалізована, забезпечуючи легку розширюваність та масштабованість продукту.

Під час розробки використовувалися мова програмування Java, фреймворк Spring та Hibernate для реалізації бізнес-логіки та доступу до бази даних. Такий вибір технологій забезпечує переносність, безпеку та надійність розроблюваного додатку.

Процес впровадження розробленого додатку може виявитися складним і вимагати налаштування та інтеграції з існуючими системами підприємства. Однак, завдяки зручному інтерфейсу та інтуїтивній природі додатку, складність використання для користувачів має бути зменшена.

Загалом, дипломна робота успішно виконана, пропонуючи реалістичну та ефективну програмну реалізацію мобільного додатку для підприємства з надання логістичних послуг. Розроблений додаток може стати цінним ресурсом для підприємства, допомагаючи покращити процеси управління та забезпечити конкурентні переваги у сфері логістики.

В процесі розробки були використані сучасні інструменти та технології, такі як мова програмування Java, Spring framework та база даних. Це дозволило створити додаток, який має гнучку архітектуру та може легко розширюватися та адаптуватися до змінних потреб підприємства.

Один з ключових функціональних елементів додатку - сканер, забезпечує швидке та точне зчитування штрих-кодів з використанням камери смартфона. Це дозволяє зручно та ефективно виконувати операції, пов'язані з ідентифікацією товарів та контролем їх руху в системі.

Тестування додатку показало його стабільність та надійність, що є критичними аспектами для успішної роботи логістичного підприємства. Багатофункціональний інтерфейс додатку забезпечує зручне та інтуїтивно зрозуміле використання для користувачів.

Загальний висновок полягає в тому, що розроблене програмне забезпечення мобільного додатку є потужним інструментом для підтримки та оптимізації логістичних процесів в підприємстві. Він дозволяє знизити час, затрати та помилки, пов'язані з ручним управлінням та обробкою даних. При правильному впровадженні та використанні додатку, підприємство зможе покращити ефективність та конкурентоспроможність своїх послуг, забезпечити точність та швидкість обробки інформації, покращити взаємодію з клієнтами та знизити загальні витрати. Програмне забезпечення мобільного додатку забезпечує зручний доступ до необхідної інформації, спрощує процеси замовлення та відстеження доставки, дозволяє вести ефективний контроль над списком транспортних засобів та оптимізувати складські процеси.

Засновуючись на аналізі засобів розробки, функціональності та результатів тестування, можна стверджувати, що програмне забезпечення мобільного додатку є ефективним та надійним інструментом для підтримки логістичних послуг у підприємстві. Він допомагає автоматизувати процеси, полегшує роботу співробітників та підвищує загальну продуктивність підприємства.

Проектування та програмна реалізація мобільного додатку на основі Java з використанням Spring framework та вище описаної бази даних дозволяють

створити масштабовану та гнучку систему, яка відповідає сучасним вимогам логістичної галузі. Застосування подібних систем, як Delivery App, Logistics Management System (LMS), Fleet Management App, Inventory Management App та CargoTrack, є доказом успішного впровадження подібних рішень у сфері логістики.

Загалом, програмне забезпечення мобільного додатку для підприємства з надання логістичних послуг, розроблений на базі Java з використанням Spring framework та супроводжується вище описаною базою даних, є ефективним та перспективним рішенням для автоматизації та оптимізації логістичних процесів. Він дозволяє забезпечити ефективне управління.

Запропоновані результати дослідження можуть бути використані для подальшого розвитку додатку, враховуючи специфічні потреби та вимоги підприємства з надання логістичних послуг. Додаткові можливості для розширення можуть включати інтеграцію з системами електронної комерції, аналітичними інструментами для оптимізації маршрутів та планування запасів, а також підтримку розширених функцій звітності та аналітики.

У підсумку, дипломна робота пропонує практичні рішення та впровадження мобільного додатку для підприємства з надання логістичних послуг. Розроблений додаток демонструє потенціал для покращення ефективності логістичних операцій, забезпечення точного відстеження товарів, оптимізації маршрутів та керування запасами. Його інтуїтивний інтерфейс та функціонал спрощують роботу користувачів та сприяють підвищенню задоволеності клієнтів. Зазначене програмне забезпечення мобільного додатку може стати цінним активом для логістичного підприємства, допомагаючи забезпечити ефективне та конкурентоспроможне надання логістичних послуг. Як результат вдалось реалізувати додаток що відповідає спроектованим вимогам до програмного і апаратного забезпечення. Додаток займає мінімальний обсяг пам’яті а для повноти використання необхідно дати доступ до деяких дозволів користувачем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Martin Christopher «Logistics and Supply Chain Management» 423 стор.,ISBN: 1292083794
2. Logistic Management System Інтернет – ресурс, режим доступу [https://optimoroute.com/logistics-management-system]
3. Fleet Management System Інтернет – ресурс, режим доступу [https://www.chevinfleet.com/learning-zone/what-is-fleet-management-software/]
4. W. Matthew «Definitive Guide to Inventory Management» 274 стор., ISBN: 9781628256642, 2021
5. А. Муррей. «Android Programming: The Big Nerd Ranch Guide» - книга з програмування на платформі Android. (3th edition) 624 стор., ISBN 978-0134706054 2017
6. Б. Нітте, Р. Мурфі. «Head First Java»(3th edition) 720 стор., ISBN: 9781491910771 2022
7. Robert C. Martin: Clean Code: A Handbook of Agile Software Craftsmanship 464 стор., ISBN: 9780132350884
8. О. Васильєв «Програмування мовою Java» 696 стор., ISBN : 9789661058797 2020
9. А.Ю. Берко «Система баз даних» 423 стор., 2020
10. Сервер, і як з ним працювати Інтернет – ресурс, режим доступу [https://kokoc.com/terminy/chto-takoe-server/]
11. А. Больє «Learning SQL» 402 стор., ISBN: 978-617-7987-01-6, 2020
12. Spring Framework Інтернет - ресурс, режим доступу [https://spring.io/projects/spring-framework]
13. K. Walls «Spring in action» (6th edition) 489 стор., 2022
14. Sam Newman «Building microservices» 566 стор., 2020

15. Сканер, як використовувати Інтернет – ресурс, режим доступу [https://systemgroup.com.ua/uk/o-kompanii/news/skanuvaty-palcem-naylegshyy-skaner-kilce-zebra-rs-5100-i-z-chym-yogo-nosyty]

ДОДАТОК А

Окремий клас для реалізації авторизації користувача у системі мобільного додатку для підприємства з надання логістичних послуг:

```
import java.util.ArrayList;
import java.util.List;
public class UserSystem {
    private List<User> users;
    public UserSystem() {
        users = new ArrayList<>();
    }
    public void addUser(User user) {
        users.add(user);
    }
    public void removeUser(User user) {
        users.remove(user);
    }
    public User findUserByUsername(String username) {
        for (User user : users) {
            if (user.getUsername().equals(username)) {
                return user;
            }
        }
        return null;
    }
    public boolean isUsernameTaken(String username) {
        for (User user : users) {
            if (user.getUsername().equals(username)) {
                return true;
            }
```

```
        }
        return false;
    }

    public boolean authenticateUser(String username, String password) {
        User user = findUserByUsername(username);
        if (user != null) {
            return user.getPassword().equals(password);
        }
        return false;
    }
}

public class User {
    private String username;
    private String password;

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        UserSystem userSystem = new UserSystem();

        // Додавання користувачів
        User user1 = new User("user1", "password1");
        User user2 = new User("user2", "password2");
        userSystem.addUser(user1);
        userSystem.addUser(user2);

        // Перевірка наявності користувача за іменем
        boolean isUsernameTaken = userSystem.isUsernameTaken("user1");
        System.out.println("Is username taken? " + isUsernameTaken);

        // Аутентифікація користувача
        boolean isAuthenticated = userSystem.authenticateUser("user1",
"password1");
        System.out.println("Is user authenticated? " + isAuthenticated);
    }
}
```

ДОДАТОК Б

Функція зчитування штрих-кодів використовуючи апаратну камеру пристрою користувача:

```
public void scanBarcode() {
    try {
        MultiFormatReader reader = new MultiFormatReader();
        // Налаштування читача штрих-кодів
        reader.setHints(getDecodeHints());
        // Отримання зображення з камери
        BufferedImage image = getCameraImage();
        // Конвертація зображення у формат, зрозумілий для ZXing
        LuminanceSource source = new BufferedImageLuminanceSource(image);
        BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(source));
        // Розпізнавання штрих-коду
        Result result = reader.decode(bitmap);
        // Отримання даних з розпізнаного штрих-коду
        String barcodeData = result.getText();
```

Функція відправки POST-запиту на сервер з даними штрих-коду:

```
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
public void sendBarcodeDataToServer(String barcodeData) {
    try {
        // URL сервера
        URL url = new URL("http://logistserver-url.com/barcode");
        // Встановлення з'єднання
        HttpURLConnection.connection=(HttpURLConnection)
url.openConnection();
```

```
            connection.setRequestMethod("POST");
            connection.setDoOutput(true);
            // Встановлення параметрів запиту
            String postData = "barcode=" + barcodeData;
            byte[] postDataBytes = postData.getBytes("UTF-8");
            // Відправка даних на сервер
            OutputStream outputStream = connection.getOutputStream();
            outputStream.write(postDataBytes);
            outputStream.flush();
            outputStream.close();
            // Отримання відповіді від сервера
            int responseCode = connection.getResponseCode();
            // Обробка відповіді сервера, якщо необхідно
        } catch (Exception e) {
            e.printStackTrace();
        }
}
```