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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ РОЗПІЗНАВАННЯ ТОВАРІВ ЗА ШТРИХ-КОДАМИ ТА УПРАВЛІННЯ СКЛАДСЬКИМИ ДОКУМЕНТАМИ

Виконав студент 2 курсу, групи 2КІ-23м
Спеціальності 123 — Комп'ютерна інженерія
Палагнюк В.І.
Керівник к.т.н., доц. каф. ОТ
Кожем'яко А.В.
« » _____ 2024 р.
Опонент к.т.н., доц. каф. ПЗ
Майданюк В.П.
« » _____ 2024 р.

Допущено до захисту
д.т.н., проф. Азаров О.Д.

« » _____ 2024 р.

ВНТУ 2024

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Галузь знань — інформаційні технології

Освітній рівень — магістр

Спеціальність — 123 Комп'ютерна інженерія

Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

д.т.н., проф. Азаров О.Д.

« » _____ 2024 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту **Палагнюку Віктору Ігоровичу**

1 Тема роботи «Інформаційна система для розпізнавання товарів за штрих-кодами та управління складськими документами» керівник роботи Кожем'яко Андрій Вікторович к.т.н., доцент, затверджено наказом вищого навчального закладу від 17.09.2024 року № 310.

2 Строк подання студентом роботи грудень 2024 р.

3 Вихідні дані до роботи: інформація про товар, отримана зі сканованого штрих-коду, операційна система — Android, пристрій — смартфон.

4 Зміст розрахунково пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз процесу розпізнавання товарів за штрих-кодами, дослідження методів управління складськими документами та аналогів, мобільний застосунок для розпізнавання товарів за штрих-кодами та обігу складських документів, економічна частина.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): технічне завдання, функціональна схема, структурна схема, зовнішній вигляд аналогів, зовнішній вигляд розробленої системи.

6 Консультанти розділів роботи приведені в таблиці 1

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Кожем'яко Андрій Вікторович к.т.н., доцент		
5	Небава М.І., к.е.н., професор каф. ЕПВМ		

7 Дата видачі завдання 18.09.2024р.

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Примітка
1	Постановка задачі	08.09.2024р.	Виконано
2	Огляд існуючих рішень	15.09.2024р.	Виконано
3	Теоретичні дослідження, огляд технологій для реалізації	25.10.2024р.	Виконано
4	Розробка і тестування інформаційної системи	24.11.2024р.	Виконано
5	Розрахунок економічної частини	12.12.2024р.	Виконано

Студент Палагнюк Віктор Ігорович

Керівник к.т.н., доц. Кожем'яко Андрій Вікторович

АНОТАЦІЯ

УДК 004.056.53

Палагнюк В. І. Інформаційна система для розпізнавання товарів за штрих-кодами та управління складськими документами. Магістерська кваліфікаційна робота зі спеціальності 123 «Комп'ютерна інженерія». Вінниця: ВНТУ, 2024. 121 с. На укр. мові. Бібліогр.: 50 назв, 14 табл., 26 рис.

У кваліфікаційній роботі було розроблено програмний додаток "Інформаційна система для ідентифікації товарів за штрих-кодами та обігу складських документів", що призначений для автоматизованої обробки відсканованих штрих-кодів, їх ідентифікації та перевірки інформації з бази даних. Додаток розроблений на мовах програмування Java та Kotlin в середовищі розробки Android Studio і використовується для створення документів та їх автоматизованої відправки на сервер.

Ключові слова: документообіг, мобільна розробка, Android, Java, Kotlin, ООП, Android Studio, Retrofit, Model-View-Presenter.

ABSTRACT

УДК 004.056.53

Palagniuk Viktor. Information System for Bar Codes Identification of Goods and Circulation of Warehouse. Master’s thesis in the specialty 123 «Computer Engineering». Vinnytsia: VNTU, 2024. In Ukrainian language. Bibliography: 50 titles, fig: 26, tabl. 14.

In the qualification work, a software application "Information System for Product Identification by Barcodes and Warehouse Document Management" was developed. It is designed for automated processing of scanned barcodes, their identification, and verification of information from a database. The application was developed using Java and Kotlin programming languages in the Android Studio development environment and is used for creating documents and their automated sending to a server.

Keywords: document management, mobile development, Android, Java, Kotlin, OOP, Android Studio, Retrofit, Model-View-Presenter.

ЗМІСТ

СПИСОК ПОЗНАЧЕНЬ І ПРИЙНЯТИХ СКОРОЧЕНЬ

ПЗ	Програмне забезпечення
ООП	Об’єктно орієнтоване програмування
OS	Операційна система
SDK	Набір для розробки ПЗ
API	Інтерфейс програмування додатків
REST	Передача репрезентативного стану
MVP	Model - View - Presenter
IDE	Інтегроване середовище розробки
UI	Інтерфейс користувача
UX	Досвід користувача
ТЗД	Термінал збору даних

ВСТУП

Одним із ключових елементів ефективного управління бізнесом є інвентаризація товарів на складах. Високий оборот продукції та велика кількість одиниць товару на складах вимагають ретельного контролю. Помилки в процесах прийому, відвантаження чи переміщення товарів можуть призвести до утворення надлишків або дефіциту, що негативно впливає на фінансові результати компанії. Тому правильне планування документообігу та інвентаризаційних процедур є основою для успішного управління запасами, виконання законодавчих вимог і забезпечення економічної стабільності підприємства. Інтеграція сучасних технологій у цей процес, таких як автоматизовані системи управління складом та застосування штрих-кодування, є важливим напрямом для вдосконалення процесу інвентаризації. Вони сприяють підвищенню точності обліку, прискоренню виконання операцій і мінімізації людських помилок. Враховуючи критичну важливість інвентаризації для загального контролю за бізнес-процесами, дослідження і вдосконалення методів управління запасами є необхідною умовою для підтримки конкурентоспроможності підприємства та забезпечення його сталого розвитку.

Об’єкт дослідження — мобільні пристрої та їх сучасні технології, а також процеси ідентифікації товарів.

Предмет дослідження — методи та інструменти розробки мобільних додатків за допомогою мов програмування та розмітки, таких як Kotlin та XML.

Основною метою дослідження є впровадження автоматизованих систем для процесів ідентифікації та обліку товарів на складах, визначення точного залишку товарів, виявлення можливих недостач або надлишків, запобігання помилкам при сортуванні, моніторинг умов зберігання та стану продукції, а також виявлення випадків крадіжок.

Для досягнення цієї мети необхідно вирішити кілька ключових **завдань**:

— розробити структуру додатку, яка буде інтуїтивно зрозумілою та зручною для користувачів;

— визначити найбільш підходящі мови програмування та середовища для розробки мобільного додатку;

— створити базу даних для зберігання інформації про товари та документи;

— реалізувати функціональність для сканування штрих-кодів, що спростить обробку товарів;

— впровадити систему управління документами;

— провести тестування програмного забезпечення для забезпечення його надійності та ефективності.

Методи дослідження: аналітичний метод для вивчення теоретичної бази дослідження, експериментальний метод, метод обробки даних, метод моделювання, розробка ПЗ, аналітичний метод дослідження.

Наукова новизна полягає у вдосконаленні методів автоматизації ідентифікації та обліку товарів за рахунок введення до уніфікації системи для роботи з різними обліковими платформами без необхідності додаткового налаштування, також можливості віддалено керування системою складського обігу.

Практична цінність роботи полягає в тому, що розроблена система ідентифікації товарів за допомогою штрих-кодів може бути успішно інтегрована в роботу магазинів та складів. Це дозволить автоматизувати процеси обліку та ідентифікації товарів, підвищити ефективність роботи співробітників, зменшити кількість помилок та значно прискорити обробку документів.

Апробація результатів роботи здійснена в електронному збірнику міжнародної науково-практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ та управління», що відбулась 20-21 листопада 2024 року. Робота опублікована за напрямком «Цифрові технології в науці, освіті та промисловості».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Аналіз процесів інвентаризації

Штрихове кодування є одним із найбільш популярних і зручних методів автоматичної ідентифікації, який широко застосовується в різних сферах економічної діяльності. Цей метод ґрунтується на використанні оптичного зчитування даних, закодованих у вигляді чорно-білих смуг. Система кодування включає комбінацію широких і вузьких штрихів та проміжків між ними, де кожна одиниця або нуль у послідовності відображається у вигляді певного елемента коду. Широкі елементи відповідають одиницям, а вузькі — нулям. Такий підхід дозволяє не лише швидко ідентифікувати товар, але й значно спрощує його облік та контроль у складських системах. За допомогою спеціальних сканерів, штрихові коди можуть миттєво розпізнаватися і оброблятися, що мінімізує час, необхідний на інвентаризацію або пошук товару в базі даних. Це автоматично скорочує можливість людських помилок та збільшує точність операцій обліку. Окрім того, система штрихового кодування дозволяє зберігати важливу інформацію про продукт, наприклад, його серійний номер, дату випуску або іншу унікальну ідентифікаційну інформацію, що полегшує управління логістичними процесами та інтеграцію з іншими інформаційними системами.

Штриховий код — це сукупність паралельних ліній та проміжків між ними, розташованих у певному порядку і з певними розмірами, що регламентуються відповідними стандартами та правилами[1].

Перші згадки про технологію штрихового кодування датуються ще 1930-ми роками, коли у Гарвардській школі бізнесу вперше було захищено дисертацію з цієї теми, яка охоплювала принципи та можливості застосування кодування в торгівлі та логістиці. Після завершення Другої світової війни почали активно розробляти патенти на використання штрих-кодів, а перші спроби практичного застосування відбулися в 1960-х роках, коли їх почали використовувати на залізниці США для ідентифікації вагонів. Штрихове кодування отримало новий

поштовх до розвитку з появою Універсального товарного коду (UPC), затвердженого в 1973 році в США, який став першим стандартом для використання в торговельній сфері та промисловості. У Європі аналогічний стандарт, відомий як Європейська система кодування (EAN), з'явився у 1977 році, закріпивши єдині принципи штрихового кодування для європейських країн. В Україні система штрихового кодування була офіційно запроваджена рішеннями Кабінету Міністрів України №180 та №326 у 1993 році, що дозволило інтегрувати національні стандарти в міжнародну систему ідентифікації, сприяючи обміну даними на міжнародному рівні та підвищуючи ефективність управління товарними потоками. Рішення щодо створення стандартів та впровадження в практику штрихового товарного кодування в Україні прийнято постановами Кабінету міністрів України № 180 від 11 березня 1993 року та № 326 від 4 травня 1993 року. 30 жовтня 1994 року Європейська Асоціація прийняла Україну в її члени, присвоївши їй товарну нумерацію «ЕАМ — Україна», а в грудні 1994 року Кабінет міністрів України прийняв постанову «Про Асоціацію товарної нумерації України «ЕАМ — Україна».

На сьогоднішній день штрихове кодування використовується для широкого спектру завдань, включаючи ідентифікацію товарів, управління складськими процесами, контролю документів та відстеження логістичних процесів. В Україні чинні стандарти регулюють застосування штрихових кодів для маркування товарів, а також стандарти на використання електронних документів для обліку та обігу продукції, що включають:

— ДСТУ 3144-95. Штрихове кодування. Терміни та визначення;

— ДСТУ 3145-95. Штрихове кодування. Загальні вимоги;

— ДСТУ 3146-95. Штрихове кодування. Маркування об'єктів ідентифікації, штрихові кодові позначення ЕАК;

— ДСТУ 3147-95. Штрихове кодування. Маркування об'єктів ідентифікації. Форма та розміщення штрихових позначок ЕАМ на тарі та пакуванні товарної продукції;

— ДСТУ 3148-95. Штрихове кодування. Система електронного обліку документів на постачання продукції;

— КНД 50-051-95. Штрихове кодування. Вибір і застосування — штрихових кодів.

Встановлені вимоги цих стандартів є обов'язковими для застосування у всіх типах нормативної документації, а також у довідкових, навчальних та методичних матеріалах. Також, вони обов’язкові для використання всіма підприємствами, установами та організаціями, які діють на території України, незалежно від їх форми власності.

Сучасні штрихові коди класифікуються за кількома критеріями, що визначають їхню форму, структуру та можливості застосування, кожен із яких розширює їхні функції у різних сферах.

Залежно від структури штрихові коди поділяють на[2]:

— цифрові — штрихові коди, які використовуються для кодування тільки числових даних, прості у використанні, особливо підходять для товарів, де немає потреби в більш деталізованій інформації, окрім чисел, таких як серійні номери або партії;

— буквено-цифрові — штрихові коди, які поєднують у собі як літери, так і цифри, що дозволяє створювати складніші й більш інформативні коди, що зручно, наприклад, для ідентифікації складських або інвентарних об'єктів, де в одному коді може бути зашифровано і код виробника, і тип продукції;

— дискретні — штрихові коди, що складаються з окремих символів, кожен з яких має визначене значення, і часто застосовуються для об’єктів з чіткою потребою у розділенні інформації, що полегшує її обробку;

— безперервні — штрихові коди, які складаються з ліній змінної ширини, що дає змогу кодувати більше даних на обмеженому просторі, роблячи їх особливо зручними для компактних етикеток або малогабаритних об’єктів;

— двонапрямні — штрихові коди, які можуть бути скановані з будь-якого боку, що підвищує швидкість обробки інформації, особливо в автоматизованих системах зчитування, таких як касові термінали, що знижує ймовірність

помилки, які критично важливо передбачити для швидкої ідентифікації товарів у роздрібній торгівлі;

— контроле-придатні — штрихові коди, що мають вбудовані контрольні механізми, такі як контрольні суми, що дозволяють перевіряти правильність зчитування та забезпечують більшу точність, запобігають помилковому розпізнаванню, що є важливим, наприклад, у медицині чи фармацевтиці, де неправильно зчитаний код може призвести до серйозних наслідків;

— з фіксованою довжиною коду — штрихові коди, які мають точно визначену кількість символів, що стандартизує їх використання;

— зі змінною довжиною коду — штрихові коди, які можуть адаптуватися до різних обсягів інформації, що дозволяє ефективніше використовувати простір на етикетці та зберігати більше даних;

— з різною інформативною щільністю — штрихові коди, в яких інформація може бути кодована менш щільно або більш щільно в залежності від вимог до точності та обсягу, при цьому коди з високою щільністю дозволяють зберігати більше інформації на меншій площі, що підходить для дрібних товарів чи невеликих упаковок, тоді як коди з меншою щільністю зручні для великих об'єктів, де інформативність не є критичною.

Такі характеристики сучасних штрих-кодів дозволяють не тільки прискорити ідентифікацію товарів і управління ними, але й адаптувати систему кодування до потреб конкретного підприємства, підвищуючи гнучкість логістичних і складських процесів.

Для детального огляду дослідження необхідно розглянути основні терміни й визначення штрихового кодування.

Символіка штрихового коду — це спеціально визначений набір знаків, що відповідають певним правилам побудови та структурній організації, які забезпечують унікальне кодування інформації. Вона складається з послідовності чорних і білих смуг різної товщини або з модулів іншої форми, кожен з яких відповідає конкретному значенню в системі кодування. Символіка визначає, які комбінації штрихів і проміжків можуть використовуватися для передачі даних,

яким чином формується початок та закінчення коду, і встановлює правила перевірки точності його зчитування. Структурованість символіки дає змогу уникати помилок під час ідентифікації товарів, а також стандартизувати передачу інформації між різними системами та пристроями зчитування.

Знак штрихового коду — це елемент конкретної системи символіки, що несе закодовану інформацію через унікальну комбінацію штрихів та проміжків, які розташовані за суворо визначеними правилами. Цей знак містить дані, які можуть бути зчитані за допомогою спеціальних пристроїв, таких як сканери, що використовують оптичне розпізнавання. Кожен знак у штриховому коді має свою послідовність і ширину штрихів, а також точні інтервали між ними, що дозволяє ідентифікувати товари, контролювати логістичні процеси, та передавати інформацію безпосередньо у системи обліку. Такий підхід забезпечує високу точність зчитування даних, дозволяючи легко інтегрувати штрихове кодування в управлінські та облікові процеси підприємств, незалежно від їх масштабів і специфіки діяльності.

Структура штрихового коду — це складна організація окремих елементів, з яких формуються знаки, а також певні правила їхнього розташування та взаємозв’язків між ними. Структура включає не лише самі штрихи й проміжки, але і логічний порядок їхнього чергування, що дозволяє однозначно ідентифікувати інформацію, закладену в коді. До структури належать такі аспекти, як початкові та кінцеві символи, які вказують на межі коду, контрольні символи для перевірки цілісності даних, а також ширина і висота окремих штрихів та інтервалів між ними. Правила побудови структури коду забезпечують узгодженість зчитування на всіх рівнях — від ручного введення до автоматизованих систем контролю, дозволяючи правильно розпізнавати дані незалежно від технічних особливостей пристроїв. Завдяки чіткій структурі штрихові коди можуть ефективно інтегруватися в логістичні та облікові системи, забезпечуючи надійне ідентифікування товарів у різних галузях промисловості та торгівлі.

Штрихова позначка — це набір даних, представлених у формі штрихового коду разом з додатковими елементами, створений за визначеними правилами для автоматизованої ідентифікації облікових одиниць.

Елемент штрихового коду — це окремий проміжок чи штрих в знаках штрихового коду.

Штрих штрихового коду — це складовий елемент коду, який являє собою ділянку поверхні носія, обмежену паралельними лініями і має колір із нижчим коефіцієнтом відбиття порівняно з рештою поверхні. Він відіграє ключову роль у формуванні коду, забезпечуючи контрастність, необхідну для коректного зчитування інформації.

Проміжок штрихового коду — це складовий елемент коду, що представляє собою ділянку поверхні, розташовану між двома сусідніми штрихами, який створює контрастність між штрихами, забезпечуючи правильне розпізнавання коду при скануванні і відіграє важливу роль у структурі коду, допомагаючи зберігати необхідний ритм і точність при зчитуванні інформації.

Роздільний проміжок штрихового коду — це відстань між останнім штрихом одного знака і першим штрихом наступного знака в дискретному штриховому коді. Цей проміжок слугує для розмежування окремих знаків, полегшуючи їх розпізнавання і точність зчитування. Він є важливим елементом у побудові дискретного коду, оскільки дозволяє чітко розділяти послідовності символів, що допомагає зчитувачеві правильно інтерпретувати кожен символ окремо.

Інформаційний знак штрихового коду — це елемент коду певної символіки, який представляє відповідний символ комп’ютерного алфавіту та виконує функцію передачі закодованої інформації, перетворюючи послідовність штрихів і проміжків у цифровий або буквений символ, що згодом розпізнається та обробляється системою. Такий знак є основною одиницею коду, за допомогою якої забезпечується точність та швидкість автоматичної ідентифікації об’єктів.

Додатковий знак штрихового коду — це елемент коду, який виконує допоміжні функції в структурі штрихового коду, зокрема обмеження або

розділення знаків у позначці. У різних системах штрихового кодування розрізняють такі додаткові знаки, як «Старт» та «Стоп» для позначення початку і завершення коду, контрольний знак для перевірки коректності зчитування, знаки обмеження ліворуч і праворуч, а також спеціальні символи, що підвищують надійність кодування, такі як візуальний знак, знак стабілізації, «штрих-носій», модуля та інші. Ці додаткові знаки допомагають забезпечити точність зчитування інформації та відповідність коду заданим стандартам.

В Україні рекомендується використовувати такі найперспективніші і найпоширеніші штрихові коди[3]: Код EAN-13 (EAN-8), Код ITF, Код 39, Код 128.

Для ефективного розв'язання завдань автоматичної ідентифікації доцільно використовувати штрихові коди з урахуванням їхніх функціональних можливостей, що визначаються специфічними характеристиками та параметрами. Зокрема, такі характеристики, як тип кодування, щільність інформації, напрямок зчитування та розмір комірок, можуть значно вплинути на швидкість та точність обробки даних у різних умовах. Детальні порівняльні характеристики штрихових кодів, які дозволяють обирати оптимальні рішення для конкретних завдань, наведені в таблиці 1.1, де коди порівнюються за критеріями інформаційної ємності, зчитуваності та інших ключових параметрів.

Таблиця 1.1 — Порівняльна характеристика штрих-кодів.

Назва характеристики	Вид штрихового коду			
	Код ITF	Код EAN/UPC	Код 39	Код 128
Вид закодованої інформації	Цифрова	Цифрова	Абетково-цифрова	Абетково-цифрова
Інформаційна щільність	2,7 мм на цифру за мінімальної ширини елемента 0,3 мм	2,1 мм на цифру за ширини модуля 0,3 мм	4,8 мм на знак за ширини модуля 0,3 мм	3,3 мм на знак або 1,7мм на цифру за ширини модуля 0,3 мм
Вимоги до точності друку	Низькі	Високі	Низькі	Середні

Продовження таблиці 1.1

Контролепри-датність знаку	Так	Ні	Так	Так
Контроле-придатність позначки	Так	Так	Так	Так
Тип кодування	Дво-ширинне	Мультиширинне	Фіксована ширина	Мультиширинне
Напрямок зчитування	Однобічний	Однобічний	Багатонаправ-лений	Багатонаправ-лений
Розмір комірок	Середній	Малий	Великий	Середній
Можливість двовимірного зчитування	Ні	Ні	Ні	Так
Мінімальний розмір коду	Помірний	Компактний	Великий	Компакнтий
Оптимальне застовування	Логістика та маркування коробок	Роздрібна торгівля	Промислові товари, складські операції	Багатофункціона льне кодування

Штриховий код, що застосовується для ідентифікації товарів, наноситься на упаковку або етикетки, прикріплені до неї, за допомогою друкарських методів або наклеювання спеціальних етикеток і ярликів. Він може розташовуватись як на транспортній, так і на споживчій упаковці, забезпечуючи можливість зчитування коду на різних етапах логістики та продажу.

Для правильного розміщення штрихового коду на упаковці існують суворі правила, спрямовані на забезпечення його легкого зчитування і надійної ідентифікації товару. Зокрема, кожна одиниця товару повинна мати лише один унікальний код EAN. Цей код зазвичай розташовують на зворотній стороні

упаковки у правому нижньому куті, де він менш схильний до пошкоджень та перекриття іншими елементами. У випадках, коли на зворотній стороні немає можливості розмістити код, допускається розміщення на лицьовій стороні упаковки, проте з дотриманням інших рекомендацій.

Для підвищення точності зчитування важливо забезпечити, щоб штриховий код був орієнтований вертикально, мав насичений темний колір на світлому фоні та залишався вільним від інших графічних елементів, які можуть перекривати або спотворювати код. Важливо, щоб штриховий код не порушував цілісності інших елементів упаковки й не накладався на текстові або зображувальні блоки.

Окрім того, існують визначені стандарти щодо розмірів штрихового коду, щоб забезпечити зчитуваність сканерами в різних умовах освітлення та кутів зчитування. Ці вимоги обов'язкові для дотримання, оскільки від них залежить точність автоматичного розпізнавання товару. Дотримання всіх зазначених стандартів допомагає оптимізувати процес ідентифікації товарів, забезпечуючи зручність і швидкість обробки товарної інформації на всіх етапах логістичного ланцюга.

1.2 Огляд існуючих рішень

ТЗД — це спеціалізований пристрій, створений для оперативного збору та обробки інформації про товари. Він широко використовується для проведення інвентаризації на складах, обліку товарів під час їх надходження та витрат, а також для виконання інших операцій у сфері складського господарства. Крім того, ТЗД активно застосовуються в організації виїзної торгівлі та у сферах послуг, що передбачають виїзд до клієнта, таких як логістика, кур'єрська доставка та управління складськими запасами.

Ці пристрої зазвичай обладнані високоякісними дисплеями для зручного відображення інформації та різними засобами введення даних, такими як функціональна клавіатура або сенсорний екран. Вони можуть включати в себе вбудовані сканери для зчитування штрих-кодів або RFID-міток, що значно

спрощує процес ідентифікації товарів. Додатково, деякі моделі ТЗД можуть бути оснащені зчитувачами магнітних карток, що надає можливість обробки платежів на місці.

Завдяки наявності бездротового зв'язку, термінали можуть швидко передавати зібрані дані на центральні системи управління, що дозволяє здійснювати моніторинг товарних запасів в реальному часі. У деяких моделях передбачена можливість розширення пам'яті, що дозволяє зберігати більший обсяг інформації та адаптувати пристрій під конкретні потреби бізнесу. Таким чином, ТЗД є незамінними помічниками у сучасній логістиці та управлінні товарними запасами, сприяючи підвищенню ефективності бізнес-процесів.

Термінал збору даних (ТЗД) є важливим інструментом для збору інформації про товари шляхом зчитування штрих-кодів або RFID-міток. Цей процес включає в себе не лише зчитування, але й зберігання, накопичення та обробку зібраних даних, що потім передаються до інформаційної системи або бази даних. Завдяки своїй функціональності, ТЗД забезпечують високу ефективність і точність обліку товарів у різних сферах діяльності.

Багато моделей терміналів обладнані клавіатурами, що дає можливість операторам вводити додаткові дані, які не можуть бути зчитані автоматично. Це може включати інформацію про стан товару, коментарі або специфікації, що дозволяє отримати більш повну картину про управління запасами. Після завершення збору даних оператор може відправити їх на персональний комп’ютер або сервер для подальшої обробки та аналізу отриманої інформації.

Для підключення до комп’ютера ТЗД використовують різні методи, що залежать від конкретної моделі пристрою. Це можуть бути спеціальні підставки, що підключаються до послідовного порту комп'ютера і виконують функції зарядного пристрою, інфрачервоні порти для бездротової передачі даних, Bluetooth-з’єднання для легкого обміну інформацією на короткій відстані, а також дротові з’єднання через USB або Ethernet. Деякі моделі також підтримують Wi-Fi з’єднання, що дозволяє терміналу підключатися до місцевих мереж і швидко передавати дані в реальному часі[5].

Термінал збору даних CARIBE PL-40L 2D(див. рис.1.1) є мобільним комп’ютером, що відповідає промисловим стандартам і поєднує у своїй конструкції характеристики, притаманні сучасним смартфонам, а також новітнім терміналам збору даних[6]. Цей пристрій обладнано потужною батареєю ємністю 4000 мАг, що забезпечує тривалу автономність у роботі. Важливою перевагою є його стійкість до впливу екстремально низьких температур, що робить його ідеальним для використання в умовах, де температура може опускатися до -20 °C.

Термінал CARIBE PL-40L 2D оснащений модулем для швидкого лазерного сканування, здатним обробляти всі типи одновимірних штрих-кодів (1D) з вражаючою швидкістю до 100 сканувань на секунду. Це дозволяє забезпечити високу продуктивність у процесах збору даних. Окрім того, існує можливість вибору модифікації цього терміналу з інтегрованим сканером, що підтримує розпізнавання як одномірних, так і двовимірних штрих-кодів (1D і 2D), що розширює його функціональність та сферу застосування. Ці характеристики роблять CARIBE PL-40L 2D незамінним інструментом у багатьох галузях, де необхідна швидка і точна обробка інформації.

Рисунок 1.1 — Термінал збору даних CARIBE PL-40L 2D

Термінал збору даних CARIBE PL-40L 2D (рис.1.2)[7] — це сучасний мобільний пристрій, який відповідає вимогам промислового рівня і поєднує в собі функціонал смартфона та новітніх технологій збору даних. Пристрій

оснащений потужною батареєю на 4000 мАг, що забезпечує тривалу роботу без підзарядки. Він розроблений для роботи в екстремальних умовах і витримує температуру до -20 °C, що дозволяє використовувати його навіть у холодних середовищах.

Цей термінал має модуль високошвидкісного лазерного сканування, який дозволяє зчитувати всі види одновимірних штрих-кодів (1D) зі швидкістю до 100 сканувань на секунду, що сприяє підвищенню продуктивності. Для додаткової універсальності можлива модифікація терміналу з підтримкою двовимірних кодів (2D), що дозволяє зчитувати як 1D, так і 2D штрих-коди. Такий функціонал робить CARIBE PL-40L 2D ефективним рішенням для завдань, де важливі висока швидкість і точність збору даних, наприклад, на складах, у логістиці та роздрібній торгівлі.

Рисунок 1.2 — Термінал збору даних Caribe PL-50L 2D

Термінал збору даних Zebra TC25(див. рис.1.3)[8] — це надійний та багатофункціональний пристрій, створений для зручної роботи в будь-яких умовах. Завдяки підтримці технології WWAN він забезпечує швидке підключення в будь-якій точці світу, що робить його ідеальним рішенням для професійного використання. Zebra TC25 відмінно працює навіть у важких умовах, включаючи дощ, високі та низькі температури, а також витримує падіння.

TC25 обладнаний вбудованою камерою та модулем сканування, що дозволяє миттєво зчитувати 1D і 2D штрих-коди з високою швидкістю, без

необхідності точного прицілювання чи затримок. Він працює на базі операційної системи Android 7.1.2 (Nougat), що забезпечує надійність і продуктивність. Підтримка 4G і VoLTE дозволяє швидко передавати дані та здійснювати чіткі голосові виклики. Завдяки технології Bluetooth 4.2 пристрій може взаємодіяти з іншими девайсами, зокрема принтерами, для спрощення робочих процесів.

Особливість Zebra DataWedge дозволяє автоматично інтегрувати штрих-коди у вже наявні програми без додаткового програмування, що значно прискорює роботу. Міцність корпусу забезпечує захист від води і пилу (IP65), а робочий температурний діапазон від -10 °C до 50 °C робить його ефективним у різних кліматичних умовах. Пристрій здатний витримати падіння з висоти 1,2 метра, а батарея ємністю 3000 мАг гарантує роботу протягом цілого дня. Додатковий PowerPack може бути встановлений для користувачів, які потребують тривалішої автономності. Серед інших переваг — можливість миттєвого голосового зв'язку, а також захищене текстове повідомлення для надійної комунікації між співробітниками, що додає пристрою зручності та безпеки у використанні.

Рисунок 1.3 — Термінал збору даних Zebra TC25

Таблиця 1.2 — Порівняльна характеристика ТЗД

Назва	CARIBE PL-40L 1D	CARIBE PL-50L 2D	Zebra TC25
ОС	Android 7.0	Android 7.0	Android 7.1.2
Скановані типи штрих-кодів	1D	1D або 2D	1D або 2D

Продовження таблиці 1.2

Тип сканера	Лазерний	Лазерний	Фото
Підтримувані штрих-коди	EAN, UPC, Code39, Code128, Code32, Code93, Codabar,	EAN, UPC, Code39, Code128, PDF417, MicroPDF417, RSS, QR код, мікро QR код, MaxiCode	EAN, UPC, Code39, Code128, Aztec Code, Data Matrix, MaxiCode, Micro QR Code, QR Code
Клас захисту	IP65	IP66	IP65
Процесор	ARM Cortex-A53 CPU 1.3 ГГц	ARM CORTEX-A53, Quad-Core 1.5 ГГц	ARM Cortex A53 1.4 ГГц
Безпроводні інтерфейси передачі даних	Bluetooth, GPS, Wi-Fi	Bluetooth, GPS, GSM, Wi-Fi	Bluetooth, GPS, Wi-Fi
Швидкість передачі даних	До 15 Мбіт/с	До 20 Мбіт/с	До 10 Мбіт/с
Можливість передачі інформації на великі відстані	Присутня	Присутня	Присутня
Стійкість до падінь	до 1,5 м	до 1,5 м	до 1,2 м
Акумулятор	Літій-іонний	Літій-іонний	Літій-іонний
Ємність акумулятора	4000 мА·год	4800 мА·год	3000 мА·год
Дисплей	4.0”	4.7”	4.3”
Оперативна пам’ять (RAM)	1 ГБ	2 ГБ	2 ГБ
Вбудована пам’ять (ROM)	8 ГБ	16 ГБ	16 ГБ
Діапазон робочих температур	Від -10°C до 50°C	Від -20°C до 55°C	Від -10°C до 50°C

Продовження таблиці 1.2

Додаткові функції	NFC, зчитувач карт	NFC, PTT (push-to-talk)	VoLTE, DataWedge

На основі порівняльної таблиці(див. табл.1.2), можна зробити висновки щодо призначення та особливостей використання кожної з трьох моделей терміналів збору даних, які були розглянуті. Кожен із них має свої унікальні характеристики, що роблять їх придатними для різних умов та завдань:

CARIBE PL-40L 1D — це пристрій, який працює на операційній системі Android 7.0 і оснащений лазерним сканером для зчитування 1D штрих-кодів. Він підтримує основні типи штрих-кодів, зокрема EAN, UPC, Code39, Code128 та інші, що робить його зручним для базових операцій у роздрібній торгівлі, логістиці та на складах. Завдяки класу захисту IP65 цей термінал може витримувати певні рівні пилу та вологи, що підходить для помірно складних умов експлуатації. Додатково, завдяки підтримці бездротових інтерфейсів передачі даних, таких як Bluetooth, GPS та Wi-Fi, пристрій може передавати інформацію на значні відстані, що забезпечує гнучкість у використанні. Модель має акумулятор ємністю 4000 мА·год, що дозволяє їй працювати тривалий час без необхідності підзарядки.

CARIBE PL-50L 2D є покращеною версією порівняно з PL-40L. Так само, як і попередня модель, він працює на операційній системі Android 7.0, але має лазерний сканер, здатний зчитувати як 1D, так і 2D штрих-коди, включаючи QR-коди, PDF417, MaxiCode, та інші. Ця універсальність дозволяє пристрою працювати в умовах, де потрібно зчитувати різноманітні штрих-коди, що особливо корисно у сфері логістики, складування та роздрібної торгівлі. Завдяки вищому класу захисту IP66, пристрій більш стійкий до впливу води та пилу, що робить його придатним для більш суворих умов експлуатації. Модель підтримує додатковий набір бездротових інтерфейсів передачі даних, таких як Bluetooth, GPS, GSM і Wi-Fi, що дозволяє користувачам бути завжди на зв’язку. Ємність акумулятора на 4800 мА·год забезпечує тривалу автономну роботу навіть у високих навантаженнях.

Zebra TC25 є найбільш технологічно розвиненим пристроєм серед трьох моделей. Він працює на операційній системі Android 7.1.2, що забезпечує підвищену стабільність роботи та сумісність з новішими додатками. Замість лазерного сканера цей пристрій оснащений фотосканером, що дозволяє зчитувати як 1D, так і 2D штрих-коди без необхідності точного прицілювання. Це підвищує ефективність роботи в умовах, коли необхідно зчитувати штрих-коди з високою швидкістю та точністю. TC25 має клас захисту IP65, що дозволяє йому витримувати вплив пилу та бризок води, а також витримує падіння з висоти до 1,2 метра, що робить його підходящим для роботи в інтенсивних умовах. Пристрій підтримує бездротові інтерфейси передачі даних, включаючи Bluetooth, GPS і Wi-Fi, що дозволяє передавати дані швидко та на значні відстані. Ємність акумулятора становить 3000 мА·год, а для тих, хто потребує більше енергії, доступний додатковий PowerPack. Також пристрій підтримує функцію VoLTE для якісного голосового зв’язку і функцію DataWedge, що дозволяє легко інтегрувати зчитування штрих-кодів у існуючі додатки без додаткового програмування.

Таким чином, кожна з моделей має свої переваги, що робить їх придатними для різних сфер використання. CARIBE PL-40L 1D підходить для базових завдань у відносно помірних умовах, CARIBE PL-50L 2D забезпечує більшу універсальність завдяки підтримці 2D штрих-кодів та підвищеному класу захисту, а Zebra TC25 є найбільш просунутим пристроєм, що поєднує в собі стабільну роботу, високу точність сканування та розширені комунікаційні можливості для складних умов експлуатації.

1.3 Опис використаних технологій

Android Studio[9] — інтегроване середовище розробки (IDE) для платформи Android, представлене 16 травня 2013 року на конференції Google I/O. Середовище побудоване на базі вихідного коду продукту IntelliJ IDEA Community Edition, що розвивається компанією JetBrains. Android Studio розвивається в рамках відкритої моделі розробки та поширюється під ліцензією Apache 2.0.

Особливості Android Studio[10]:

— організація проєкту;

— редактор коду;

— емулятор Android;

— візуальні редактори;

— керування залежностями;

— інструменти для налагодження та оптимізації.

Android Studio пропонує зручну структуру проєкту, де всі файли, ресурси та налаштування мають свою визначену позицію. Це сприяє упорядкуванню проєкту та спрощує процес навігації між файлами і компонентами застосунку.

Вбудований редактор коду в Android Studio надає розробникам широкий набір функцій, таких як автозаповнення, підказки, автоматичне форматування, перевірка синтаксису та коректності відступів. Він підтримує декілька мов програмування, включаючи Java, Kotlin і C++, що дозволяє використовувати їх у різних проєктах.

Завдяки вбудованому емулятору Android, розробники можуть запускати та тестувати застосунки на віртуальних пристроях без необхідності фізичного апарату. Це особливо зручно для перевірки сумісності застосунку з різними версіями Android і типами пристроїв.

Android Studio надає кілька інструментів для візуальної розробки інтерфейсу, наприклад, Layout Editor для створення і редагування XML-макетів, та Drawable Editor для редагування графічних елементів, таких як іконки та фонові зображення.

Android Studio інтегрує Gradle — систему управління залежностями, яка дозволяє легко додавати бібліотеки та налаштовувати залежності проєкту. Це допомагає розширювати можливості застосунків та спрощує їх підтримку.

Android Studio забезпечує розробників розширеними засобами для профілювання та налагодження. Ці інструменти дозволяють аналізувати продуктивність застосунку, знаходити помилки, відстежувати використання ресурсів і оптимізувати роботу, підвищуючи якість та ефективність застосунку.

Android[11] — це операційна система, а також потужна платформа, призначена для використання на мобільних телефонах, планшетах і різних портативних пристроях, розроблена компанією Google на основі ядра Linux. Ця система отримала потужну підтримку від консорціуму Open Handset Alliance (OHA), що об’єднує кілька десятків провідних компаній у галузі мобільних технологій, включаючи виробників пристроїв, операторів зв’язку та розробників програмного забезпечення. Така широка підтримка сприяла швидкому розвитку і популяризації платформи Android.

Одним із ключових елементів операційної системи Android є Dalvik — віртуальна машина Java, яка забезпечує можливість ефективного виконання додатків і програмного забезпечення на мобільних пристроях з обмеженими ресурсами. Ця віртуальна машина оптимізована для роботи на пристроях з невеликою кількістю оперативної пам'яті та слабким процесором, що дозволяє Android працювати швидко та стабільно навіть на недорогих смартфонах.

Основна частина додатків для Android створена за допомогою мови програмування Java, що робить процес розробки зрозумілим і доступним для багатьох програмістів. Однак з часом платформа також почала підтримувати інші мови, такі як Kotlin та C++, розширюючи можливості для розробників.

Важливою особливістю Android є його відкритий код, що дозволяє будь-кому вносити зміни та створювати власні модифікації. Це сприяє постійному вдосконаленню платформи, появі нових функцій і розширень, а також дозволяє виробникам пристроїв адаптувати операційну систему під свої конкретні потреби. Відкритість коду також стимулює розвиток незалежних спільнот, які активно займаються розробкою нових інструментів, додатків та прошивок для Android-пристроїв, що робить цю платформу однією з найбільш гнучких і персоналізованих на ринку.

Software Development Kit (SDK)[12] — це комплексний набір інструментів і ресурсів, призначених для розробників програмного забезпечення, який дозволяє створювати додатки для конкретної технології, платформи або операційної системи. Він надає програмістам усе необхідне для повноцінного

процесу розробки: від компіляторів і бібліотек до інструментів тестування й налагодження. Основною метою створення SDK є полегшення та прискорення розробки, зокрема за рахунок автоматизації рутинних завдань і стандартизації робочих процесів.

SDK включає цілий спектр інструментів, серед яких компілятори, дебагери, бібліотеки, емулятори для тестування додатків на різних пристроях, а також інструменти для створення графічного інтерфейсу користувача. Наприклад, компілятор перетворює код у виконуваний файл, а дебагер допомагає розробнику знаходити і виправляти помилки. Емулятор, своєю чергою, дозволяє перевірити функціональність додатку на віртуальному пристрої, що особливо корисно на ранніх етапах розробки.

Особливу роль у складі SDK відіграє супровідна документація, яка є важливим джерелом інформації для розробників. Ця документація містить докладні описи всіх доступних інструментів і бібліотек, включених у SDK, та їхнього застосування на практиці. Документація часто надає приклади коду, пояснення параметрів, методів та інших компонентів, щоб полегшити розробникам розуміння структури та принципів роботи SDK. Крім того, вона містить рекомендації щодо найкращих практик та оптимального використання ресурсів.

Документація SDK не тільки пояснює, як користуватися інструментами, але й дає змогу розробникам швидше освоїтися в новому середовищі та почати створювати програми відповідно до стандартів обраної платформи. Завдяки такій підтримці, розробники можуть уникати багатьох типових помилок, підвищувати ефективність коду та скорочувати час розробки. Також, оскільки документація оновлюється разом із оновленнями SDK, вона дозволяє програмістам залишатися в курсі нових можливостей та інструментів, які з'являються в системі, що робить їх роботу більш продуктивною і якісною.

Android SDK[12] — це комплексний набір інструментів, необхідних для створення програмного забезпечення на платформі Android. До цього набору входять різноманітні інструменти, такі як налагоджувач, що допомагає знаходити

й виправляти помилки, бібліотеки, які забезпечують доступ до стандартних функцій Android, емулятор, що дозволяє запускати й тестувати програми на віртуальних пристроях, а також супровідна документація. Крім того, Android SDK містить приклади коду і навчальні матеріали, які дозволяють розробникам швидко освоїти основи роботи з Android та отримати доступ до додаткових функцій платформи.

Цей набір інструментів підтримує кілька популярних операційних систем, що робить його доступним для широкого кола розробників. Зокрема, Android SDK можна використовувати на комп'ютерах під керуванням сучасних дистрибутивів Linux, на Mac з версією операційної системи Mac OS X 10.5.8 або новішою, а також на Windows, починаючи з версії Windows 7. Завдяки цьому розробники можуть вибирати середовище, яке їм найбільше підходить, і з легкістю створювати додатки для Android незалежно від платформи, на якій вони працюють.

Ключовою особливістю Android SDK є можливість створення додатків, які повністю інтегровані з екосистемою Android. Бібліотеки, що входять до складу SDK, надають доступ до основних API Android, таких як робота з інтерфейсом користувача, доступ до баз даних, інтеграція з іншими додатками, використання сенсорів та багато іншого. Завдяки цьому розробники можуть створювати додатки, які відповідають усім стандартам якості та функціональності, встановленим для платформи Android.

Емулятор Android, який є частиною SDK, дозволяє тестувати додатки на віртуальних пристроях з різними конфігураціями, починаючи від різних версій Android до різних характеристик пристроїв, таких як розмір екрану та продуктивність. Це дає можливість перевірити сумісність додатка з різноманітними пристроями, не потребуючи фізичного доступу до кожного з них.

Android Virtual Device (AVD)[14] — це конфігурація, яка дозволяє імітувати характеристики різних пристроїв Android, таких як телефони, планшети, пристрої Wear OS, Android TV або пристрої ОС Automotive, у віртуальному емуляторі Android. Використовуючи емулятор, можна запускати і тестувати

програми без використання реального Android пристрою. Емулятор Android надає зручне середовище для розробки та налагодження програм, де можна відтворювати різні конфігурації пристроїв і перевіряти, як додатки працюють на різних пристроях з різними версіями Android. За допомогою AVD можна налаштовувати різні параметри, такі як розмір екрану, роздільна здатність, версія Android, обсяг оперативної пам'яті та багато іншого.

SDK для розробки програм для Android містить не тільки емулятор, але й ряд інших інструментів, призначених для налагодження та налаштування додатків. Наприклад, одним з важливих інструментів є Android Debug Bridge (ADB), який працює з командного рядка і надає можливість створювати, видаляти і налаштовувати віртуальні пристрої. ADB, крім того, надає можливість створювати і оновлювати проекти для платформи Android, а також оновлювати Android SDK, додаткові компоненти та документацію. З використанням командного рядка та інших інструментів SDK, які входять до складу Android Studio, розробники можуть зручно налагоджувати та налаштовувати свої додатки для оптимальної роботи на платформі Android.

Java[15] — це об'єктно-орієнтована мова програмування, яка була представлена компанією "Sun Microsystems" у 1995 році як основний елемент платформи Java. Починаючи з 2009 року, подальший розвиток мови та платформи Java веде компанія "Oracle", яка придбала "Sun Microsystems" у тому ж році. Важливим принципом Java є її платформна незалежність, досягнута завдяки використанню байт-коду. В офіційній реалізації Java, програми компілюються у спеціальний проміжний код, відомий як байт-код, який сам по собі не є безпосередньо виконуваним кодом.

Замість цього, байт-код інтерпретується віртуальною машиною Java (JVM) див. Додаток А, яка адаптує код для конкретної платформи, на якій програма виконується. Завдяки такому підходу Java дозволяє розробникам писати програми, що можуть працювати на будь-якій операційній системі, яка підтримує JVM, не вимагаючи перекомпіляції коду для кожної платформи. Це робить Java універсальною мовою, яка широко використовується для розробки

різноманітного програмного забезпечення — від веб-додатків до мобільних застосунків і великих корпоративних систем.

Java SE 11[16] — це одна з ключових версій платформи Java, випущена компанією Oracle у вересні 2018 року, яка належить до Standard Edition (Java SE) і включає набір інструментів, бібліотек і специфікацій для створення та виконання Java — додатків. Ця версія стала особливо значущою, оскільки була першою довгостроковою версією (LTS) після Java SE 8, забезпечуючи стабільність та довготривалу підтримку, що є важливим для великих корпоративних систем і проєктів, які потребують надійного середовища.

Java SE 11 принесла ряд нових можливостей і удосконалень, що поліпшують зручність, продуктивність і гнучкість платформи. Основні нововведення включають[17]:

— модульна система Java (Java Platform Module System, або JPMS);

— видалення застарілих API та модулів;

— впровадження стандартного HTTP-клієнта;

— розширення та оптимізація String API.

Модульна система Java (JPMS) вперше представлена у Java 9, вона продовжує розвиватися і в Java SE 11, дозволяючи розробникам гнучкіше організовувати програмні компоненти, керувати залежностями та зменшувати розмір програми, видаляючи невикористовувані модулі.

З Java SE 11 було виключено деякі старі API, зокрема Java EE та CORBA, що сприяє оптимізації і спрощенню платформи. Це дозволяє зробити JVM більш легкою та продуктивною, а також забезпечує сучаснішу базу для розробки.

Новий HTTP — клієнт, що повністю підтримує HTTP/2 та асинхронні операції, значно полегшує роботу з мережевими запитами. Завдяки йому розробники можуть ефективніше взаємодіяти з веб-сервісами та реалізовувати асинхронні мережеві операції.

У Java SE 11 було додано нові методи для роботи з рядками, такі як isBlank(), strip(), repeat() та інші. Це дозволяє легше і швидше обробляти текстові дані, а також сприяє зменшенню коду і покращенню його читабельності.

Java SE 11 також відзначається підвищеною продуктивністю та безпекою, що є важливими факторами для сучасних програмних проєктів. У довгостроковій перспективі ця версія стала основою для багатьох корпоративних систем, оскільки Oracle забезпечує її оновленнями безпеки та виправленнями помилок на протязі тривалого часу. Завдяки новим функціям і довготривалій підтримці, Java SE 11 залишається популярною версією для розробників, які прагнуть до стабільної та функціональної платформи для своїх додатків.

Kotlin[18] — це статично типізована мова програмування, розроблена компанією JetBrains для роботи на JVM. Основною метою розробників було створити мову, яка була б компактнішою, безпечнішою та зручнішою, ніж Java. У результаті цієї оптимізації вдалося досягти швидшої компіляції та поліпшеної підтримки в середовищі розробки (IDE). Завдяки своїй лаконічності та розширеним можливостям, Kotlin швидко здобув популярність серед розробників Android-додатків, особливо завдяки кільком важливим перевагам[19]:

— інтероперабельність з Java;
— безпека та обробка нульових значень;
— компактний та читабельний синтаксис;
— підтримка функціонального програмування.

Однією з ключових переваг Kotlin є його здатність безперешкодно взаємодіяти з Java-кодом. Завдяки цьому розробники можуть використовувати існуючі бібліотеки та компоненти, написані на Java, у своїх Kotlin-проектах, і навпаки — додавати Kotlin-код у Java-проекти. Така взаємодія значно полегшує перехід на Kotlin для компаній, які вже мають великий код на Java, дозволяючи поступово впроваджувати Kotlin у проекти без необхідності переписувати всю кодову базу.

Kotlin відомий своїм прагненням до безпечного коду завдяки вбудованим інструментам, що допомагають уникати типових помилок. Особливо корисною є система роботи з "nullable types", де розробник може чітко визначити, чи допускає змінна значення `null`. Це значно знижує ризик виникнення помилки

NullPointerException, яка є однією з найбільш поширених у Java. Такий підхід забезпечує більш надійне виконання програм і допомагає уникати несподіваних збоїв.

Синтаксис Kotlin відрізняється лаконічністю та зрозумілістю, що дозволяє скоротити кількість коду без втрати його змістовності. Завдяки відсутності зайвих шаблонів і конструкцій, розробники можуть писати код, який легше підтримувати та читати. Наприклад, операції з властивостями об’єктів у Kotlin можуть бути записані в одну лінію, що зменшує громіздкість коду, характерну для Java.

Kotlin включає функціональні можливості, такі як лямбда-вирази, вищі функції, колекції з операціями `map`, `filter`, `reduce` тощо. Це дозволяє писати більш лаконічний і зрозумілий код, використовуючи функціональні підходи, які спрощують роботу з великими обсягами даних та роблять процес обробки даних ефективнішим. Завдяки підтримці функціонального програмування Kotlin надає розробникам інструменти для створення більш гнучких і надійних додатків, використовуючи новітні підходи до написання коду.

Порівнюючи використання мов програмування Kotlin та Java у розробці мобільних додатків для платформи Android, можна детально розглянути їх основні переваги та недоліки, а також визначити, яка з них є більш оптимальною для конкретних потреб проекту. Kotlin, що був офіційно оголошений мовою для Android-розробки в 2017 році, набув великої популярності серед розробників завдяки своїй зручності та потужності. Однією з ключових переваг Kotlin є його безшовна інтероперабельність з Java, що дозволяє плавно переходити на Kotlin без необхідності переписувати великий обсяг існуючого коду. Завдяки компактному та лаконічному синтаксису Kotlin дозволяє значно зменшити кількість рядків коду, що робить розробку більш ефективною та швидкою. Окрім того, Kotlin підтримує функціональне програмування, надаючи можливість створювати чистіший та більш читабельний код. Мова також має сучасні можливості, такі як вбудовані обробники null-значень, лямбда-функції та багато інших інструментів для полегшення роботи розробників.

З іншого боку, Java залишається однією з найпоширеніших і найстабільніших мов програмування для Android. Вона має багатий набір інструментів і бібліотек, що дозволяють ефективно реалізувати складні функціональні можливості в мобільних додатках. Більше того, Java має величезну спільноту розробників, що забезпечує постійну підтримку та вдосконалення мовних інструментів. Це робить Java стабільним і надійним вибором для великих корпоративних додатків, де важливі такі фактори, як перевірені технології та підтримка впродовж багатьох років.

Таким чином, хоча обидві мови мають свої сильні сторони, вибір між ними залежить від конкретних вимог проекту. Kotlin ідеально підходить для швидкої розробки та оптимізації коду, тоді як Java залишається надійним вибором для великих, довготривалих проектів. У даній дипломній роботі обидві мови використовуються як основні інструменти для розробки Android-додатків, адже вони забезпечують максимальну зручність і швидкість у процесі реалізації мобільного додатку з урахуванням потреб користувача та бізнес-логіки.

ООП[20] є однією з основних парадигм програмування, що розглядає програму як сукупність «об’єктів», котрі взаємодіють між собою для досягнення певних цілей. У рамках ООП програми структуровані у вигляді об'єктів, кожен з яких відображає реальні чи абстрактні сутності і володіє властивостями (атрибутами) та методами (функціями), що визначають його поведінку та взаємодію з іншими об'єктами.

MVP[21] — це шаблон проектування, який розділяє логіку відображення та обробки подій на три основні компоненти(див. рис.1.4): модель (Model), представлення (View) та презентер (Presenter). Завдяки цьому підходу додаток стає більш організованим, що сприяє його легшій розширюваності та спрощує процес тестування. Використання цього шаблону дозволяє чітко визначити відповідальність кожного компонента, що знижує складність підтримки та оновлення коду.

Рисунок 1.4 — Візуалізація архітектурних компонентів MVP

Компоненти архітектурного шаблону MVP (Model-View-Presenter) виконують різні, але взаємопов'язані ролі, що забезпечують ефективну організацію коду та полегшують підтримку та розширення додатку. Кожен з цих компонентів має чітко визначену відповідальність, що сприяє створенню масштабованих та тестованих рішень у розробці програмного забезпечення:

— модель (model);
— вид (view);
— презентер (presenter).

Модель (Model) є основним елементом, що відповідає за зберігання, обробку даних і бізнес-логіку додатку. Модель може включати різні класи, які здійснюють доступ до баз даних, мережевих ресурсів або інші джерела даних, обробляють інформацію, виконують необхідні обчислення, маніпулюють даними, а також забезпечують виконання операцій збереження або отримання інформації. Важливо, що модель не має жодного знання про те, як саме ці дані будуть представлені користувачеві, а її основною метою є надання правильних результатів для подальшої обробки.

Вид (View) відповідає за відображення інтерфейсу користувача (UI) та забезпечення зручної взаємодії між додатком і користувачем. Вид отримує інформацію від презентера і відображає її на екрані, а також відправляє користувацькі дії (натискання кнопок, введення даних) до презентера для подальшої обробки. Це може бути реалізовано через активності, фрагменти або інші компоненти UI, що відображають дані користувача. Вид не містить логіки обробки даних, а лише виступає як інтерфейс для взаємодії з користувачем.

Презентер (Presenter) функціонує як посередник між моделлю та видом. Він отримує вхідні дані від виду, виконує необхідні операції та обробку з

використанням моделі та потім передає результати назад до виду для відображення. Презентер відповідає за управління логікою додатку, такою як обробка користувацьких подій, виклик методів моделі для отримання або обробки даних, а також за оновлення вигляду відповідно до отриманих результатів. Це дозволяє підтримувати розділення обов’язків між компонентами і забезпечує можливість тестування бізнес-логіки без прив'язки до конкретного інтерфейсу користувача.

REST — це архітектурний стиль, який забезпечує доступ до інформаційних ресурсів через веб. Він був вперше описаний Роєм Філдінгом у 2000 році, одним з авторів протоколу HTTP, і з того часу здобув широку популярність завдяки своїй простоті та ефективності. REST ґрунтується на принципах роботи Всесвітньої павутини і активно використовує можливості протоколу HTTP, що робить його потужним інструментом для побудови розподілених веб-систем.

Передача даних у REST здійснюється через кілька стандартних форматів, зокрема HTML, XML або JSON, з останнім із них, JSON, став найбільш популярним завдяки своїй легкості та зручності в роботі з JavaScript. Один з ключових принципів REST — це підтримка кешування, що дозволяє зберігати копії даних на стороні клієнта чи посередника для прискореного доступу без необхідності повторного звертання до сервера. Це забезпечує значну економію ресурсів і знижує навантаження на сервер.

REST також характеризується безсесійністю, тобто сервер не зберігає стан клієнта між запитами. Кожен запит до сервера є самодостатнім, оскільки він містить усю необхідну інформацію для обробки запиту. Це забезпечує простоту у використанні та масштабуванні, оскільки кожен запит може бути оброблений незалежно від попередніх, що робить систему більш стійкою до збоїв.

Однією з головних переваг REST є його масштабованість і здатність до еволюції з часом. Завдяки своїй архітектурній простоті, REST дозволяє створювати гнучкі та розширювані веб-сервіси, які можуть без проблем працювати на різних платформах і пристроях, від мобільних телефонів до серверних систем. Це дозволяє розробникам створювати надійні й масштабовані

рішення, які легко адаптуються до нових вимог і змін у технологічному середовищі.

REST став стандартом для розробки API[22] (інтерфейсів програмування додатків), що дозволяє забезпечити ефективну взаємодію між різними програмними компонентами та сервісами через мережу. Завдяки своїй легкості у впровадженні та підтримці, REST став основним підходом для створення веб-сервісів і API, що забезпечують інтеграцію між різними системами і платформами, включаючи мобільні додатки, веб-сайти та інші онлайн-сервіси.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ІДЕНТИФІКАЦІЇ ТОВАРІВ ЗА ШТРИХ-КОДАМИ ТА ОБІГУ СКЛАДСЬКИХ ДОКУМЕНТІВ

2.1 Загальні положення про розроблювану ІС та її архітектура

Інформаційна система для ідентифікації товарів за штрих-кодами й автоматизації обігу складських документів призначена для ефективного управління товарними операціями. Основним завданням є створення зручного інтерфейсу користувача, який дозволить швидко переглядати результати після введення даних, виконуючи всі необхідні операції з товарними позиціями.

Застосунок розробляється для:

— автоматизації ідентифікації та обліку товарів;

— оптимізації управління запасами у магазинах і на складах;

— прискорення обробки складських документів;

— мінімізації помилок, які виникають через людський фактор.

Головні функціональні можливості системи охоплюють:

– перевірку коректності штрих-кодів, як відсканованих, так і введених вручну за допомогою клавіатури;

— створення нових документів і здійснення операцій з ними;

— зберігання документів локально на пристрої;

— передачу документів на віддалений сервер для подальшої обробки та архівування.

Вихідні дані: інформація про товар, отримана зі сканованого штрих-коду.

Вхідні дані: штриховий код товару.

Основна вимога до застосунку: інформаційна система має бути реалізована у вигляді мобільної інформаційної системи, що працює на операційній системі Android. Додаток повинен забезпечувати можливість пошуку товарів у базі даних через сканування їхніх штрих-кодів, що дозволить користувачу швидко і зручно отримувати інформацію про продукт.

Вимоги до процесу сканування штрих-кодів:

— перевірка на наявність товарів в базі даних;

— додавання товарів в документ;

— можливість змінювати кількість продуктів в документі.

Клієнтський додаток для ідентифікації товарів за штриховими кодами та управління складськими документами має багаторівневу архітектуру, що включає кілька функціональних підсистем. Кожна з цих підсистем виконує певні завдання, що разом забезпечують цілісне та ефективне функціонування системи. Загальна структура архітектури(див. рис.2.1) складається з таких основних компонентів:

— підсистема обробки документів;

— підсистема сканування штрих-кодів;

— підсистема візуалізації;

— локальна база даних;

— серверна частина.

Підсистема обробки документів — цей компонент відповідає за створення, редагування, збереження та обробку всіх складських документів. ПОД забезпечує автоматизоване формування документів, а також управління їхньою структурою та вмістом, що дозволяє знизити ризики людських помилок і підвищити швидкість виконання операцій.

Підсистема сканування штрих-кодів — забезпечує функцію сканування штрих-кодів за допомогою мобільної камери пристрою або спеціального сканера. Ця підсистема інтегрує дані з бази товарів для ідентифікації продукції, дозволяючи користувачу швидко додавати товар до документа, перевіряти його наявність, а також вести облік.

Підсистема візуалізації — відповідає за зручний інтерфейс, що відображає інформацію для користувача та забезпечує інтерактивну взаємодію з додатком. Інтуїтивно зрозумілий дизайн ПВ дозволяє користувачам легко орієнтуватися в функціоналі додатка, що особливо важливо для працівників складу або торгових точок.

Локальна база даних — на клієнтському пристрої зберігається основна інформація про товари та документи. Завдяки локальному зберіганню даних додаток може працювати автономно, без постійного підключення до інтернету, що є важливою вимогою для мобільного рішення.

Серверна частина — забезпечує синхронізацію даних між клієнтським додатком і зовнішніми системами, такими як облікові або управлінські системи компанії. Серверна частина дозволяє підтримувати актуальність інформації про товари і документи в режимі реального часу, а також забезпечує централізоване управління даними.

Така багаторівнева архітектура забезпечує стабільну та ефективну роботу програми, дозволяючи легко здійснювати ідентифікацію товарів, облік складських операцій та безперебійну взаємодію між різними системами. Завдяки цьому користувачі можуть працювати з додатком швидко, зручно та надійно, отримуючи всі необхідні дані для виконання складських і торговельних операцій.

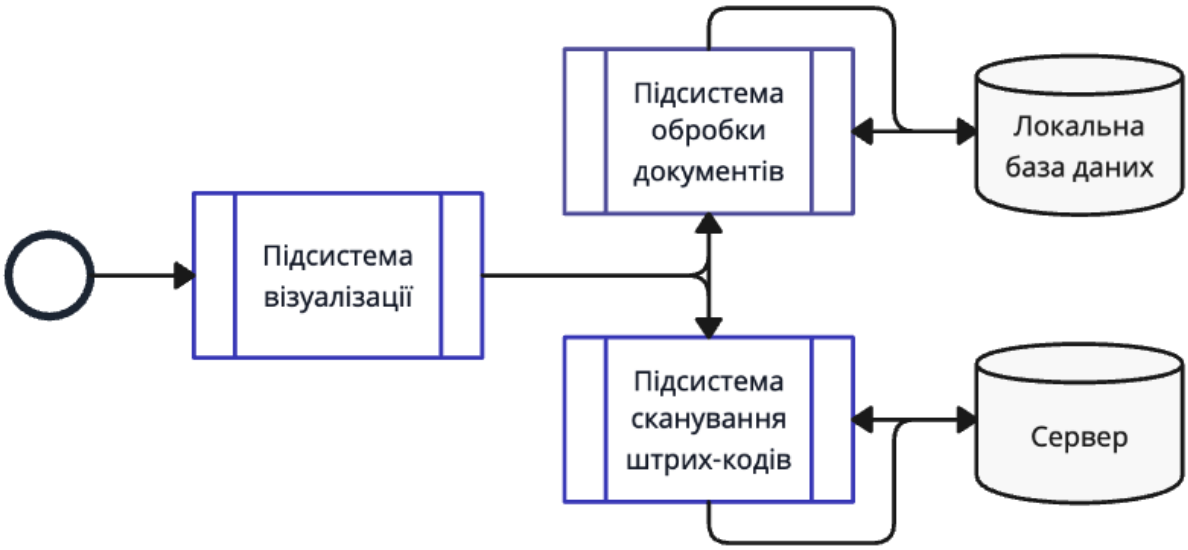

Рисунок 2.1 — Архітектура інформаційної системи

Архітектура клієнтського додатку для ідентифікації товарів за штрих-кодами є багатокомпонентною системою, де кожен модуль відповідає за специфічний функціонал і тісно взаємодіє з іншими частинами програми. Основні елементи архітектури включають три важливі підсистеми: модуль

сканування штрих-кодів, модуль управління документами та модуль візуалізації інтерфейсу користувача. Дані про товари зберігаються в локальній базі, а для забезпечення синхронізації та обміну інформацією з іншими системами використовується віддалений сервер.

Робота системи починається з активації підсистеми візуалізації, яка відповідає за побудову графічного інтерфейсу та ініціалізацію необхідних даних. Це дозволяє користувачеві зручно взаємодіяти з додатком, швидко отримуючи доступ до потрібних функцій і можливостей. Інтерфейс створений для максимального спрощення навігації та дозволяє оперативно працювати з інформацією про товари.

Для виконання задач, додаток активно взаємодіє з двома іншими підсистемами: підсистемою сканування штрих-кодів та підсистемою керування документами. Підсистема сканування може функціонувати як в режимі автоматичного зчитування штрих-кодів за допомогою камери або спеціального сканера, так і в ручному режимі, де користувач може вводити штрих-коди вручну. Після сканування підсистема виконує перевірку наявності товару в локальній базі даних, забезпечуючи перевірку коректності введених даних і їх актуальності.

Підсистема обробки документів виконує обробку інформації про товари, що додаються до документів. Вона дозволяє створювати нові документи, редагувати дані, видаляти непотрібні позиції та додавати нові товари до документації. Після того, як документи підготовлені, вони можуть бути збережені на пристрої або надіслані на віддалений сервер для подальшої обробки чи архівування. Це забезпечує цілісність та актуальність інформації для зовнішніх систем.

Локальна база даних зберігає всю необхідну інформацію про товари та взаємодіє з підсистемою сканування для забезпечення швидкого пошуку та доступу до даних. Після створення документів дані синхронізуються з віддаленим сервером, що дозволяє централізовано контролювати товарообіг, спрощувати процес інвентаризації та забезпечувати надійність зберігання документів.

2.2 Проектування бази даних інформаційної системи

Однією з ключових вимог до додатку є забезпечення можливості кешування відсканованої інформації на пристрої користувача. Це дозволить додатку працювати швидко та автономно, навіть без доступу до інтернету. Для реалізації даної функції, локальна база даних повинна зберігати всю необхідну інформацію про товари, а також структурувати дані про документи та їхні елементи.

Базу даних слід організувати таким чином, щоб вона забезпечувала ефективне зберігання і доступ до інформації для різних функціональних модулів системи. Основними таблицями є:

Таблиця «Товари» (Products) зберігає всі основні дані, що характеризують товари, які можуть бути відскановані або знайдені у системі(див. табл.2.1). Збереження даних таблиці дозволяє швидко отримувати інформацію про товар для обробки та відображення. Основні поля включають:

— назва товару;
— штриховий код;
— вартість;
— кількість.

Таблиця 2.1 — Поля таблиці Products

Тип	**Назва елементу**	**Назва**
String	ID_PRODUCT	Унікальний ідентитфікатор товару
String	PRODUCT_NAME	Назва товару
String	BAR_CODE	Штриховий код товару
String	COST	Вартість товару
String	QUANTITY	Кількість на складі

Таблиця «Документи» (Documents) містить інформацію про документи, що створюються в процесі обліку та управління товарообігом(див. табл.2.2), що дозволяє контролювати облік та статус документів, які генеруються у системі. Основні поля включають:

— номер документа;

— остання дата редагування;

— тип документа;

— чи змінювався за допомогою мобільного додатку.

Таблиця 2.2 — Поля таблиці Documents

Тип	**Назва елементу**	**Назва**
String	ID_DOCUMENT	Унікальний ідентифікатор документу
String	NUMBER	Номер документа
String	MODIFICATION_DATE	Остання дата редагування
String	TYPE_DOCUMENT	Тип документа
String	IS_CHANGED_IN_APP	Чи був змінений у мобільному додатку

Таблиця «Елементи документів» (DocumentItems) слугує для зберігання інформації про кожен елемент у конкретному документі(див. табл.2.3),, що забезпечує можливість деталізації вмісту кожного документа. Основні поля включають:

— назва товару;

— ID документу;

— штриховий код;

— залишок товару на складі;

— кількість товару на складі;

— вартість.

Таблиця 2.3 — Поля таблиці DocumentItems

Тип	**Назва елементу**	**Назва**
String	ITEM_ID	ID документу
String	PRODUCT_NAME	Назва товару
String	ID_DOCUMENT	ID документу
String	BAR_CODE	Штриховий код товару
String	BALANCE	Залишок товару на складі
String	QUANTITY	Кількість на складі
String	COST	Вартість товару

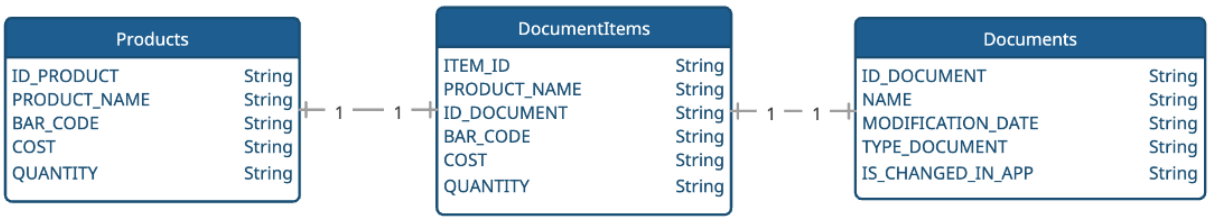

Рисунок 2.2 — Діаграма спроектованої локальної бази даних

У розробленій базі даних між таблицями встановлені зв’язки типу один-до-одного(див. рис.2.2), що дозволяє зручно поєднувати та структурувати дані. Зокрема, таблиця Products зберігає інформацію про товари та пов’язана з таблицею DocumentItems через поле BAR_CODE, яке використовується для унікальної ідентифікації товару. Завдяки цьому, кожен товар, що включений у певний документ, може бути швидко знайдений і отриманий з таблиці товарів.

Крім цього, таблиця Documents, яка містить дані про всі створені документи, має зв’язок із таблицею DocumentItems за допомогою поля ID_DOCUMENT. Це поле виконує роль посилання на документ, до якого належать окремі позиції товарів, що зберігаються у таблиці DocumentItems. Таким чином, кожен запис в DocumentItems прив’язаний до конкретного документа, забезпечуючи повну деталізацію складу кожного документа.

Для організації структури бази даних використовуються первинні ключі, що забезпечують унікальність кожного запису у таблицях та дозволяють швидкий доступ до даних. Зокрема, первинний ключ у таблиці Products дозволяє знайти конкретний товар за його унікальним штрих-кодом, а в таблиці Documents — за унікальним ідентифікатором документа. Це забезпечує логічну цілісність бази даних та дозволяє уникнути дублювання даних, що є важливим для оптимізації роботи додатку.

Усі елементи у таблицях, такі як BAR_CODE, ID_DOCUMENT та інші, представлені у форматі String. Використання строкових типів даних спрощує обробку інформації, а також полегшує інтеграцію бази даних з інтерфейсом користувача, де більшість даних може бути відображена у текстовому вигляді. Це

рішення забезпечує гнучкість при обробці та представленні інформації, дозволяючи легко змінювати та оновлювати дані в базі.

Таким чином, побудована структура бази даних є надійною та добре масштабованою, що дозволяє ефективно зберігати інформацію про товари, документи та позиції в документах. Це забезпечує високий рівень продуктивності системи, а також дозволяє користувачу швидко отримувати потрібну інформацію без зайвих затримок.

2.3 Обґрунтування вибору технології, мови програмування та інструментальних засобів реалізації для розробки системи.

Для реалізації програмного коду інформаційної системи, яка буде використовуватися для ідентифікації товарів за штрих-кодами та обробки складських документів, планується застосування мови програмування Java та Kotlin, середовища розробки Android Studio, а також XML-файлів для побудови інтерфейсу користувача. Такий вибір технологій дозволяє забезпечити високу продуктивність, гнучкість та зручність у розробці системи.

Основними мовами для розробки додатків під Android є Java і Kotlin. Java, яка є однією з найпоширеніших мов програмування, характеризується стабільною базою та великою кількістю бібліотек, що допомагають реалізовувати складні функціональні можливості. Її багаторічний досвід у галузі Android-розробки сприяє швидкому розв'язанню технічних задач завдяки великій кількості документації та підтримки з боку спільноти. Kotlin, у свою чергу, є сучасною мовою програмування, що поєднує переваги Java та підтримує безпечніші й продуктивніші підходи до написання коду. Вона спрощує синтаксис, зменшуючи кількість коду, та забезпечує додатковий захист від багатьох типових помилок, як-от NullPointerException. Завдяки використанню обох мов Java та Kotlin розробка стає не тільки ефективнішою, а й надає більше можливостей для оптимізації та розширення функціоналу системи в майбутньому.

Android Studio — це офіційна платформа для розробки Android-додатків, яка пропонує повний набір інструментів для створення, налагодження та

тестування мобільних додатків. Вона забезпечує інтегрований редактор коду з підсвіткою синтаксису, автоматичними підказками, перевіркою помилок у реальному часі та зручним інтерфейсом для управління залежностями. Крім цього, Android Studio включає емулятор Android для тестування додатків на різних моделях пристроїв і екранів, що дає змогу оптимізувати додаток для широкого спектра користувачів. Інтегровані інструменти, такі як інспектор макетів та аналізатор продуктивності, допомагають розробникам виявляти і виправляти проблеми з інтерфейсом та продуктивністю. Використання Android Studio суттєво прискорює процес розробки завдяки зручності і широким можливостям для автоматизації.

Для опису інтерфейсу користувача використовується XML-розмітка, яка є стандартним підходом для Android-додатків. XML забезпечує чітке розмежування між бізнес-логікою додатка та його візуальною складовою, що робить код більш структурованим і легким для підтримки. Завдяки використанню XML можна легко налаштовувати розташування елементів інтерфейсу, їх стиль, а також взаємодію з користувачем. Крім того, XML дає змогу створювати різні макети для різних розмірів екранів та орієнтацій, що є особливо важливим для адаптації додатка до різних пристроїв. Використання XML дозволяє швидко оновлювати інтерфейс без потреби змінювати код, що робить розробку більш гнучкою та масштабованою.

Таким чином, вибір технологій — Java та Kotlin для програмування, Android Studio для розробки, налагодження та тестування, а також XML для створення інтерфейсу — є оптимальним і обґрунтованим для реалізації інформаційної системи, орієнтованої на ідентифікацію товарів за штрих-кодами та управління складськими документами. Цей підхід дозволяє створити потужне та надійне рішення, яке буде легко підтримувати, розширювати та адаптувати під нові вимоги платформи Android.

2.4 Алгоритми шифрування штрих-кодів

У сучасних інформаційних системах використовуються різні види алгоритмів шифрування для захисту даних, які вбудовані в штрих-коди. Серед них найбільш популярними є симетричні алгоритми, такі як AES, що забезпечують швидке та ефективне шифрування за допомогою одного ключа; хешування, наприклад, алгоритми SM3 або SHA-256, що створюють контрольні суми для перевірки цілісності даних; та механізми HMAC, які використовують комбінацію хеш-функцій і секретних ключів для підтвердження автентичності. Комплексне застосування цих технологій дозволяє гарантувати конфіденційність, захист від підробки та цілісність даних у системах обліку товарів.

Основні види алгоритмів шифрування:

— симетричне;

— асиметричне;

— хеш-алгоритми;

— динамічне шифрування.

Симетричне шифрування — це метод шифрування даних, при якому один і той самий секретний ключ використовується як для шифрування, так і для розшифрування інформації. Підхід базується на тому, що відправник і отримувач повинні мати доступ до одного і того ж ключа, який залишається конфіденційним для сторонніх осіб.

Одним із прикладів симетричного шифрування штрих-кодів є шифрування SM4. SM4 — це блочний шифр, який використовує 128-бітний ключ та працює в 32 раунди.

$$rki = K_i \oplus FK_i, \tag{2.1}$$

де K — 128-бітний секретний ключ для шифрування;

rki — раундові ключі, що представляють 32 раунди шифрування, згенеровані з основного ключа K;

FK_i — фіксовані параметри раунду.

Формула 2.1 відображає процес розширення секретного ключа K за допомогою операції виключної диз'юнкції (операція, що набуває значення «істина» тоді й лише тоді, коли значення «істина» має суто один з її операндів).

У SM4 визначені чотири ключових параметри:

— FK_0 = 0xA3B1BAC6;

— FK_1 = 0x56AA3350;

— FK_2 = 0x677D9197;

— FK_3 = 0xB27022DC.

Константи FK додають ентропію до початкового ключа, що ускладнює атаки на алгоритм. Вони забезпечують унікальність шифрування — специфічні зміни для кожного раунду, гарантуючи, що навіть при однаковому початковому ключі результат шифрування буде відрізнятися.

Алгоритм SM4 базується на 32 раундах шифрування, кожен із яких включає застосування раундової функції F. Вхідні дані розбиваються на чотири 32-бітні слова X_0, X_1, X_2, X_3 які обробляються раунд за раундом для отримання зашифрованого тексту.

Функція F виконує основну обчислювальну роботу, перетворюючи вхідні слова та раундовий ключ у вихідне слово. Формула функції для кожного раунду (2.2):

$$X_{i+4} = F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rki), \quad (2.2)$$

де X_i, X_{i+1}, X_{i+2}, X_{i+3} — чотири послідовні 32-бітні слова;

rki — раундовий ключ для поточного раунду;

F(Xi, Xi+1, Xi+2, Xi+3, rki) — складається з двох основних операцій: нелінійного перетворення та лінійного перетворення.

Операція нелінійного перетворення включає застосування S-блоків, які забезпечують заплутування. Формула (2.3):

$$T(B) = S(B_0) \parallel S(B_1) \parallel S(B_2) \parallel S(B_3), \quad (2.3)$$

де B — 32-бітне слово;

B_i, B_1, B_2, B_3 — байти B;

S(·) — функція заміни, яка використовує S-блок (таблицю нелінійних замін).

Після нелінійного перетворення застосовується лінійне перетворення для подальшого розсіювання. Формула (2.4):

$$L(B) = B \oplus (B \lll 2) \oplus (B \lll 10) \oplus (B \lll 18) \oplus (B \lll 24), \quad (2.4)$$

де B — результат нелінійного перетворення;

⋘ — циклічний зсув вліво на n позицій.

Комбінуючи нелінійне та лінійне перетворення, отримуємо повну функцію T (2.5):

$$T'(B) = L(T(B)), \quad (2.5)$$

Для кожного раунду і виконується операція шифрування (2.6):

$$X_{i+4} = X_i \oplus T'(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rki), \quad (2.6)$$

де X_i, X_{i+1}, X_{i+2}, X_{i+3} — чотири послідовні 32-бітні слова, що використовуються для обчислення нового слова X_{i+4};

T'(·) — функція, що відповідає за заплутування і розсіювання даних;

rki — раундовий ключ, додається через операцію виключної диз'юнкції, що залежить від раунду.

Після 32 раундів шифрування результатом є (2.7):

$$Y = X_{35}, X_{34}, X_{33}, X_{32}, \quad (2.7)$$

Дешифрування виконується аналогічно, але раундові ключі застосовуються у зворотному порядку. Такий підхід забезпечує надійну заплутаність та розсіювання даних, що робить алгоритм SM4 стійким до багатьох криптографічних атак.

Основні принципи симетричного шифрування:

— один секретний ключ для всіх операцій з даними, без якого неможливо розшифрувати зашифровану інформацію.

— швидкість і ефективність: Симетричні алгоритми зазвичай швидші за асиметричні, оскільки вони виконують обчислення меншої складності.

— безпека ключа: Головна складність симетричного шифрування полягає у безпечній передачі ключа між сторонами, щоб уникнути його перехоплення зловмисниками.

Асиметричне шифрування — це криптографічний метод, що використовує дві різні, але пов'язані між собою пари ключів:

— публічний ключ (public key);

— приватний ключ (private key).

Публічний ключ використовується для шифрування даних і є відкритим, його можна передавати будь-кому, а приватний ключ використовується для розшифрування даних і зберігається в таємниці

Одним із прикладів асиметричного шифрування є криптографія на еліптичних кривих, що базується на математичних властивостях еліптичних кривих над кінцевими полями. Вона забезпечує високий рівень безпеки при використанні ключів меншої довжини, що робить ECC ефективною для мобільних додатків і систем із обмеженими ресурсами.

Еліптична крива задається рівнянням (2.8):

$$y^2 = x^3 + ax + b \pmod p, \tag{2.8}$$

де p — просте число (порядок поля);

a, b ∈ F_p — коефіцієнти, які визначають криву;

$4a^3 + 27b^2 \not\equiv 0 \pmod p$ — умова, що гарантує відсутність особливих точок(точок із вертикальними або незрозумілими тангенсами), має гладку структуру, походить із властивостей еліптичних кривих і визначає, чи є задана еліптична крива "невиродженою".

Набір точок на кривій E, разом із точкою на нескінченності (O), утворює абелеву групу з операцією додавання. Для двох точок $P(x_1, y_1)$ і $Q(x_2, y_2)$ сума $R(x_3, y_3) = P + Q$ визначається ра формулами 2.9, 2.10, 2.11:

Якщо $P \neq Q$:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod p \tag{2.9}$$

$$x_3 = \lambda^2 - x_1 - x_2 \pmod p$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod p$$

де λ — є коефіцієнтом нахилу прямої, яка проходить через дві точки $P_1(x_1, y_1)$ і $P_2(x_2, y_2)$ або дотикається до кривої в точці P_1 (у випадку, коли $P_1 = P_2$)

x, y — координати точок.

Якщо P = Q (подвоєння точки):

$$\lambda = \frac{3x_1^2 + a}{2y_1} \pmod p \tag{2.10}$$

$$x_3 = \lambda^2 - 2x_1 \pmod p$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod p$$

Якщо P + Q = O (точка на нескінченності):

$$R = O, \tag{2.11}$$

Наступними кроками буде генерація закритого та відкритого ключів

$$d \in [1;\ n - 1], \tag{2.12}$$

де d — закритий ключ;

n — порядок, найменше додатнє число, для якого R = O, при O — точка на нескінченності.

$$Q = d \cdot R, \tag{2.13}$$

де Q — закритий ключ;

R — генераторна, базова точка на кривій, яка належить F_p.

d — закритий ключ, скаляр.

Множення точки R на скаляр d (Q = d · R) виконується за допомогою алгоритму подвоєння і додавання. Для d, представленого у двійковій формі, використовуються такі кроки:

— початковою точкою є Q = O;

— для кожного біта d_1 (від старшого до молодшого) подвоїти поточну точку Q та якщо $d_1 = 1$ — додати R до Q.

Наступним кроком в алгоритмі є генерація підпису ECC. Для цифрового підпису використовується алгоритм ECDSA (Elliptic Curve Digital Signature Algorithm).

Необхідно обрати число k випадковим чином, яке буде унікальним для кожного повідомлення, щоб уникнути компроментації закритого ключа d.

$$k \in [1, n-1], \tag{2.14}$$

де k — випадкове число;

n — порядок базової точки R.

Обчислення точки G виконується за формулою 2.15. Операція включає додавання R до себе k разів.

$$G = k \cdot R, \tag{2.15}$$

де k — випадкове число;

G — точка на кривій;

R — генераторна, базова точка на кривій, яка належить F_p.

Наступним кроком є визначення компонента r за формулою 2.16:

$$r = x_G \pmod n, \tag{2.16}$$

де r — компонент, що є першою частиною цифрового підпису;

x_G — координати точки на кривій;

n — порядок базової точки R.

Наступним кроком є визначення компонента s за формулою 2.17:

$$s = k^{-1} \cdot (H(m) + d \cdot r) \pmod{n}, \quad (2.17)$$

де s — компонент, що є другою частиною цифрового підпису;

k^{-1} — мультиплікативна обернена до k за модулем n, що задовольняє $k \cdot k^{-1} \equiv 1 \pmod{n}$;

H(m) — криптографічна хеш функція, результат якої має фіксовану довжинуб що є меншою за початкову інформацію m;

m — інформація, що має бути зашифрованою (в даному випадку — штрих-код);

r — перша частина цифрового підпису;

d — закритий ключ.

Отже результуючим підписом є пара (r, s).

Для перевірки підпису необхідно обичслити обернену мультиплікативну до s за модулем n (2.18):

$$s^{-1} \pmod{n}, \quad (2.18)$$

де s — компонент, що є другою частиною цифрового підпису;

n — порядок базаової точки R.

$$u_1 = H(m) \cdot s^{-1} \pmod{n} \quad (2.19)$$

$$u_2 = r \cdot s^{-1} \pmod{n},$$

де u_1, u_2 — точки, використовуються для обрахунку лінійної комбінації еліптичної кривої точок;

s^{-1} — обернений модуль n до s;

H(m) — криптографічна хеш функція, результат якої має фіксовану довжинуб що є меншою за початкову інформацію m;

m — інформація, що має бути зашифрованою (в даному випадку — штрих-код);

r — перша частина цифрового підпису;

n — порядок базаової точки R;

d — закритий ключ.

$$P = u_1 \cdot R + u_2 \cdot Q, \tag{2.20}$$

де P — точка для перевірки;

Q — відкритий ключ;

R — генераторна, базова точка на кривій, яка належить F_p.

Якщо координата x_P точки $P=(x_P,y_P)$ задовільняє умову 2.21, то підпис (r. s) є дійсним. Інакше підпис вважається недійсним

$$x_P \pmod n = r \tag{2.21}$$

Переваги асиметричного алгоритму ECC:

— криптографічна стійкість забезпечується складністю проблеми дискретного логарифма над еліптичною кривою;

— при відомій парі (r, s) і відкритого ключа Q неможливо відновити закритий ключ d;

— повторне використання k для різних $H(m_1)$ та $H(m_2)$ може призвести до компроментції закритого ключа d.

Отже, симетричне шифрування забезпечує високу швидкість обробки даних та ефективність у середовищах, де передача ключів між сторонами є безпечною. SM4 ідеально підходить для швидкого шифрування великих обсягів інформації в межах локальних систем. Асиметричне шифрування на основі еліптичних кривих (ECC) забезпечує більш високий рівень безпеки при використанні менших ключів, що є важливим для мобільних додатків із обмеженими ресурсами. ECC особливо корисне для передачі даних у відкритих або ненадійних мережах, оскільки дозволяє уникнути прямого обміну секретним ключем.

Для збільшення рівня надійності симетричного шифрування SM4 необхідно подвоїти кількість раундів оригінального шифрування, що дзволить

отримати більш розсіяний результат, який важче дешифрувати. Детальний аналіз переваги модифікування оригінального алгоритму приведений в таблиці 2.4.

Таблиця 2.4 – Порівняння ефективності шифрування в 32 та 64 раунди.

Параметр	**32 раунди**	**64 раунди**	**Різниця**
Швидкість шифрування	Базова швидкість	Вдвічі повільніше за базову швидкість	Зменшення швидкості на 50% (з 1млс до 2 млс)
Швидкість дешифрування	Базова швидкість	Вдвічі повільніше за базову швидкість	Зменшення швидкості на 50% (з 1млс до 2 млс)
Надійність	84%	99%	Підвищення найдісності на 10-15% завдяки більшій складності
Розсіювання даних	Високий рівень	Дуже високий рівень	Покращення розсіювання на 20-30%
Вирати пам’яті	Помірний рівень	Незначно зростає	Зростання на 5-10% завдяки більшій складності
Стійкість до атак	Висока	Дуже висока	Стікійсть до лінійного диференціального аналізу покращується експоненційно

У диференційному криптоаналізі ймовірність успішної атаки залежить від кількості знайдених диференціальних характеристик $P_{диф}$ (2.22).

$$P_{\text{диф}} = (p_i)^n, \tag{2.22}$$

де $P_{диф}$ — кількість диференціальних характеристик;

p_n — ймовірність, що диференціальний зв’язок залишиться незмінним після раунду;

i — номер раунду;

n — кількість раундів шифру;

$$P_{\text{диф } 32} = (2^{-2})^{32} \approx 5.42 \cdot 10^{-20} \quad (2.23)$$
$$P_{\text{диф } 64} = (2^{-2})^{32} \approx 3.16 \cdot 10^{-39}$$

Визначення стікості алгоритмів (2.24):

$$S_{32} = 1 - P_{\text{диф } 32} \approx 75\% \quad (2.23)$$
$$S_{64} = 1 - P_{\text{диф } 64} \approx 99.9\%$$

Таким чином стійкість алгоритму до несанкціонованих атак зростає на приблизно 25% після модифікації (збільшення кількості раундів шифрування до 64).

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ІДЕНТИФІКАЦІЇ ТОВАРІВ ЗА ШТРИХ-КОДАМИ ТА ОБІГУ СКЛАДСЬКИХ ДОКУМЕНТІВ

3.1 Опис класів та функцій реалізованого програмного засобу

Програмне забезпечення складається з різних типів класів, кожен із яких виконує специфічні завдання, забезпечуючи коректне функціонування всієї системи. Класи у додатку охоплюють важливі аспекти роботи: вони відповідають за відображення графічного інтерфейсу, управління поведінкою елементів під час взаємодії з користувачем, обробку та передачу даних між сервером і клієнтською частиною, а також за збереження інформації в локальній базі даних для забезпечення офлайн-доступу. Завдяки чіткій структурі класів забезпечується висока гнучкість додатку та легкість його підтримки. Докладний лістинг програмного коду розробленого додатка, що ілюструє його структуру та взаємодію компонентів, можна знайти в додатках Г, Ж та Е.

Під час реалізації програми було створено класи activity, presenter тощо.

MainActivity — це клас, що визначає головний екран додатку, з якого користувач починає взаємодію. Цей клас виконує роль активності та реалізує інтерфейс MainShape.View, забезпечуючи ключові функції для роботи з графічним інтерфейсом і обробки подій користувача:

— onCreate — виконує ініціалізацію активності, тобто встановлює макет інтерфейсу, налаштовує необхідні компоненти, а також викликає метод checkPermission з презентера для перевірки дозволів;

— onStart — конфігурує параметри вікна для взаємодії з користувачем, забезпечуючи оптимальні налаштування введення;

— onClick — метод, що обробляє натискання на елементи управління; відповідно до ідентифікатора елемента, виконує різні дії, включаючи запуск інших активностей або виконання певних операцій.

SettingsActivity — активність, що відповідає за конфігурацію параметрів додатку, успадковує AppCompatActivity і включає в себе кілька змінних і методів для забезпечення налаштувань користувача:

— onCreate, який ініціалізує активність, встановлюючи відповідний вміст інтерфейсу та ініціалізуючи необхідні змінні, реалізує взаємодію з UserManager для завантаження раніше збережених налаштувань;

— onKeyDown, який обробляє натискання кнопки "Назад" — якщо користувач натискає кнопку "Back", активність буде закрита;

— initViews, який відповідає за ініціалізацію елементів інтерфейсу користувача і налаштування відповідних обробників подій для взаємодії з ними;

— updateStores, що оновлює список магазинів у спінері, використовуючи наданий список даних;

— storeRequestFailed, який відображає повідомлення користувачу в разі невдачі при отриманні даних про магазини;

— setSelectedStore, який зберігає вибраний користувачем магазин зі спінера для подальшого використання;

— onClickSave, що обробляє натискання кнопки "Зберегти", після чого зберігаються налаштування користувача і закривається активність.

DocsListActivity — це активність, що відповідає за відображення списку документів у додатку. Вона успадковує клас AppCompatActivity та реалізує інтерфейс DocsShape.DocsView. Даний клас містить кілька змінних і методів, що виконують різноманітні функції для ефективної роботи з документами та інтерфейсом користувача:

— initViews, який ініціалізує елементи інтерфейсу користувача, налаштовує їхню взаємодію та обробку подій;

— onCreate, що ініціалізує активність, налаштовує вміст екрану та ініціалізує змінні, також взаємодіє з UserManager для отримання коду магазину та викликається DocsPresenter для завантаження та оновлення списку документів;

— requestDocTypes, який запитує доступні типи документів через DocsPresenter, забезпечуючи можливість користувачу вибирати тип документа;

— updateAdapter, який оновлює адаптер mAdapter за допомогою курсора, який передається в параметрі, для правильного відображення даних в списку;

— setDocTypes, що зберігає список типів документів, що надходить від DocsPresenter, для подальшого використання в додатку;

— openAddDocumentDialog, який відкриває діалогове вікно для додавання нового документа, де користувач може ввести необхідні дані;

— showEmptyDocsMessage, який відображає повідомлення користувачу, коли список документів порожній, щоб інформувати його про відсутність даних;

— updateActionStatus, який оновлює статус кнопок або інших елементів керування в інтерфейсі в залежності від вибору елементів у списку документів;

— onAddDocClick, який обробляє натискання кнопки для додавання нового документа та викликає метод openAddDocumentDialog;

— onOptionsItemSelected, що обробляє вибір пунктів меню, включаючи перевірку натискання кнопки "Back" для виходу з активності;

— ViewHolder, що є вкладеним класом адаптера DocItemAdapter, який використовується для представлення елементів списку документів у RecyclerView, відповідає за налаштування та збереження посилань на елементи розмітки, що використовуються для відображення кожного документа;

— onCreateViewHolder — створює новий об'єкт ViewHolder для кожного елемента списку документів;

— onBindViewHolder — заповнює дані елемента списку, такі як номер документа, дата та коментар, використовуючи курсор, який містить відповідну інформацію;

— getItemCount, який повертає загальну кількість елементів у списку, що забезпечує правильне відображення всіх доступних документів.

GoodCardActivity — активність, яка забезпечує перегляд деталей товару. Спадкує AppCompatActivity та містить змінні й методи для керування

інтерфейсом та функціоналом, пов'язаним із пошуком та відображенням товару. Функції:

— initViews, який налаштовує елементи інтерфейсу користувача, що необхідні для відображення інформації про товар;

— onCreate, який ініціалізує активність, встановлює макет екрану, налаштовує змінні й отримує код магазину з UserManager, що дозволяє завантажити потрібні дані для товару;

— clickGoBack, який обробляє натискання кнопки "Назад" для повернення до попереднього екрану, закриваючи поточну активність;

— onKeyDown, який обробляє натискання клавіш пристрою; якщо натиснута клавіша "Enter", виконує запит через goodQuery для отримання даних про товар на основі введеного штрих-коду;

— goodQuery, який виконує запит до RestController для отримання інформації про товар, перевіряє валідність введеного штрих-коду, обробляє відповідь сервера, а також відображає дані товару чи повідомлення про помилку;

— onKeyMultiple, який обробляє багаторазове натискання клавіші, передаючи отриманий рядок символів у метод goodQuery для пошуку товару;

— showIfGoodFound, який відображає детальну інформацію про товар, якщо його знайдено, забезпечуючи зручний доступ до даних;

— showErrorMsg, який відображає повідомлення про помилку, якщо виникла проблема під час пошуку або обробки даних товару.

ItemListActivity — активність для роботи зі списком елементів (Items), яка успадковує AppCompatActivity і реалізує інтерфейс ItemShape.View, містить функції для управління списком, відображення діалогів та обробки подій інтерфейсу:

— hideKeyboard(), який приховує екранну клавіатуру, забезпечуючи чистоту інтерфейсу;

— onCreate(savedInstanceState: Bundle?), який ініціалізує активність, налаштовує адаптер списку та елементи інтерфейсу під час створення активності;

— onKeyMultiple(keyCode: Int, repeatCount: Int, event: KeyEvent): Boolean, який обробляє події натискання клавіш, зокрема перехоплює "Enter" для пошуку за штрих-кодом;

— onBackPressed(), який обробляє подію натискання кнопки "Back", закриваючи діалогове вікно або повертаючись до попереднього екрану;

— showUpdateLocalItemDialog(item: Item, isMode3: Boolean), який відображає діалог для оновлення елемента на основі штрих-коду barq, інформації про товар good, та режиму isMode3;

— showUpdateItemDialog(barq: String?, good: Good?, isMode3: Boolean), який відображає діалогове вікно для оновлення елемента на основі штрих-коду та інформації про товар;

— hideDialog(), який закриває активний діалог;

— showError(msg: String?), який виводить повідомлення про помилку з текстом msg;

— updateAdapter(), який оновлює адаптер після змін у списку для актуального відображення даних;

— initAdapter(), який налаштовує адаптер mAdapter для RecyclerView mRecyclerItems, додає обробник подій для елементів списку;

— updateAdapter(), який оновлює адаптер списку елементів; Ця функція викликається після змін у списку елементів для оновлення відображення;

— showError(msg: String?), який показує повідомлення про помилку; отримує рядок msg як вхідний параметр і відображає його у спливаючому повідомленні для користувача;

— hideDialog(), який закриває діалогове вікно, викликається для приховування діалогового вікна, якщо воно було активне;

— showUpdateItemDialog(barq: String?, good: Good?, isMode3: Boolean), який відкриває діалог для редагування елемента за штрих-кодом і даними про товар, параметри barq (штрих-код), good (інформація про товар) та isMode3 (прапорець для визначення режиму редагування) дозволяють користувачеві внести зміни для оновлення елемента;

— showUpdateLocalItemDialog(item: Item, isMode3: Boolean), який запускає діалогове вікно для редагування локального елемента, отримує параметри item (елемент для оновлення) і isMode3 (флаг режиму редагування), дає можливість користувачу оновити дані елемента;

— onBackPressed(), який обробляє натискання кнопки "Назад" — якщо діалогове вікно відкрите, воно буде закрито; в іншому випадку викликається super.onBackPressed(), що забезпечує стандартну обробку кнопки "Назад";

— onKeyMultiple(keyCode: Int, repeatCount: Int, event: KeyEvent): Boolean, який перехоплює натискання клавіш, особливо "Enter", щоб активувати пошук за штрих-кодом;

— searchItemByBarcode(barq: String), який виконує пошук елемента за штрих-кодом barq. Якщо елемент з таким штрих-кодом існує в mItemList, він буде знайдений;

— showDeleteItemDialog(barq: String), який показує діалогове вікно для підтвердження видалення елемента з вказаним штрих-кодом barq, після підтвердження видаляє елемент зі списку;

— addItem(item: Item), який додає новий елемент до mItemList, приймає об’єкт item, додає його до списку та оновлює адаптер після додавання;

— updateItem(barq: String, item: Item), який оновлює дані існуючого елемента за штрих-кодом barq, після пошуку елемента за штрих-кодом оновлює його дані з об’єкта item і знову налаштовує адаптер;

— deleteItem(barq: String), який видаляє елемент з mItemList за штрих-кодом barq, після видалення елемента адаптер оновлюється для відображення змін.

MainPresenter виступає як презентер у патерні MVP (Model-View-Presenter) та виконує функції керування логікою для головного екрану (MainActivity). Основні методи і властивості класу:

— initDB(), який здійснює ініціалізацію бази даних, встановлюючи посилання mDb на об'єкт SQLiteDatabase;

— deviceImei, який повертає унікальний ідентифікатор пристрою (IMEI);

— putIntoDatabase(goods: List<Good?>?), який зберігає інформацію про товари в базу даних;

— clearBase(), який очищає всі записи з бази даних;

— askPermissions(), який ініціює запит на необхідні дозволи у користувача;

— getMd5Hash(input: String), який генерує MD5 хеш з переданого рядка;

— hasPermissions(context: Context?, vararg permissions: String?): Boolean, який перевіряє, чи додаток має всі потрібні дозволи;

— loadGoods(), який викликається для завантаження даних про товари з бази або інших джерел;

— checkPermission(), який перевіряє, чи є необхідні дозволи для роботи додатку.

SettingsPresenter відповідає за керування екраном налаштувань (SettingsActivity). Його основне завдання — отримати список магазинів і передати ці дані для оновлення інтерфейсу налаштувань;

— requestStores(), який запускає асинхронний запит для отримання списку магазинів із сервера;

— updateStores(), який викликається, коли отримано успішну відповідь, і забезпечує оновлення списку магазинів на екрані налаштувань;

— storeRequestFailed(), який спрацьовує у випадку помилки під час запиту, щоб повідомити про збої в отриманні даних;

— DocsPresenter займається управлінням списком документів у DocsListActivity.

— updateStores(), який оновлює список магазинів на екрані налаштувань після успішного отримання даних;

— setStoreCode(code: String?), який задає код для магазину;

— putDocInDB(docsDto: DocsDto), який зберігає документ у базі даних;

— updateRecyclerView(), який оновлює інтерфейс списку документів у застосунку;

— insertDocInDb(doc: Doc?), який додає документ у базу даних;

— requestDocTypes(), який надсилає запит на сервер для отримання типів документів;

— sendDocs(idsString: Array<String?>), який надсилає вибрані документи на сервер;

— delDocs(idsString: Array<String?>), який видаляє документи з бази даних за заданими ідентифікаторами;

— isMassHasElems(mass: Array<String?>): Boolean, який перевіряє, чи містить масив будь-які елементи;

— initDB(), який виконує ініціалізацію бази даних;

— deleteDocsFromDb(idsString: Array<String?>), який видаляє документи з бази даних відповідно до заданих ідентифікаторів;

— getDocsDtoFromDb(idsString: Array<String?>): DocsDto, який повертає об'єкт DocsDto із бази даних для зазначених документів;

— sendDocsQuery(docsDtoFromDB: DocsDto), який відправляє запит на сервер для обробки документів;

— getMultipleSelectionString(mass: Array<String?>): String, який генерує рядок із масиву елементів для використання в SQL-запиті;

— removeNulls(a: Array<String?>): Array<String?>, який видаляє всі нульові значення з переданого масиву рядків;

— sendDocsAskOption(docsDto: DocsDto): AlertDialog, який створює діалогове вікно для підтвердження перед відправкою документів;

— delDocsAskOption(idsString: Array<String?>): AlertDialog, який генерує діалогове вікно для підтвердження видалення документів.

ItemPresenter є презентером, який керує елементами у списку, що відображаються в ItemListActivity. Функції:

— initDB(), який виконує ініціалізацію бази даних для роботи з елементами;

— lastCursor, який властивість, яка зберігає останній курсор, що містить дані поточного списку елементів;

— requestGoodByBarcode(barcode: String?, docType: Int, storeCode: String?), який здійснює запит на отримання інформації про товар за його штрих-кодом та іншими параметрами;

— createGoodByClick(), який відкриває діалогове вікно для створення нового товару в базі даних;

— getItemByParams(good: Good), який повертає елемент, використовуючи вказані параметри товару;

— getItemByBarcode(barcode: String?), який виконує пошук елемента за вказаним штрих-кодом;

— getItemById(itemId: String?), який повертає елемент, посилаючись на його унікальний ідентифікатор;

— insertItem(item: Item?), який додає новий елемент у базу даних;

— getMode3Good(barcode: String?), який отримує товар для третього режиму, використовуючи його штрих-код;

— updateItem(value: Double?, name: String?, args: Array<String?>?), який оновлює значення елемента в базі даних з переданими даними;

— enterMode3(barcode: String?, itemId: String?), який відкриває діалогове вікно, призначене для третього режиму роботи з елементами.

Store — дата клас, який представляє магазин, зберігаючи його код та назву. Клас має дві основні властивості: "code" — код магазину і "name" — назва магазину. Функції:

— toString() повертає назву магазину у вигляді рядка.

Good — це дата клас, який представляє товар і містить інформацію про різні параметри товару. Клас включає наступні властивості:

— id — ідентифікатор товару;

— name — назва товару;

— price — ціна товару;

— barcode — штрих-код товару;

— code — код товару;

— quantity — кількість товару;

— unit — одиниця виміру товару;

— isWeightBased — інформація, чи є товар ваговим;

— totalRows — загальна кількість рядків у таблиці товарів;

— currentRow — поточний рядок товару в таблиці.

Doc — дата клас, що представляє документ і містить різноманітні поля для зберігання важливої інформації, включаючи ідентифікатор, номер, дату створення, коментар, тип документа, а також дані про походження з 1С і статус змін. Клас також має код магазину і список елементів документа (items). Крім того, він імплементує інтерфейс Parcelable для можливості передачі об'єктів між різними компонентами Android. Функції:

— describeContents(), який повертає ціле число, що вказує на тип опису контенту, яке використовується для Parcelable, у цьому випадку повертається 0;

— writeToParcel(parcel: Parcel, flags: Int), який здійснює запис даних об'єкта Doc у Parcel для подальшої передачі або збереження, що є частиною механізму Parcelable;

— createFromParcel(parcel: Parcel), який створює новий об'єкт Doc з даних, що передаються через Parcel, в контексті реалізації Parcelable;

— newArray(size: Int), який повертає масив об'єктів Doc заданого розміру, що є частиною підтримки Parcelable.

3.2 Опис екрану налаштувань

Інформаційна система для ідентифікації товарів за штрих-кодами та управління складськими документами реалізована у вигляді мобільного додатку для операційної системи Android. Для коректної роботи програми на пристрої повинна бути встановлена операційна система версії не нижче API 21 (Android 5.0).

Після запуску додатка користувач бачить головний екран, на якому представлені три основні кнопки, що дозволяють перейти до ключових функцій системи (див. рис. 3.1):

— довідник — для перегляду та пошуку інформації про товари.

— документи — для роботи зі складськими документами та їх обігом.

— налаштування — для конфігурації додатку та налаштування параметрів користувача.

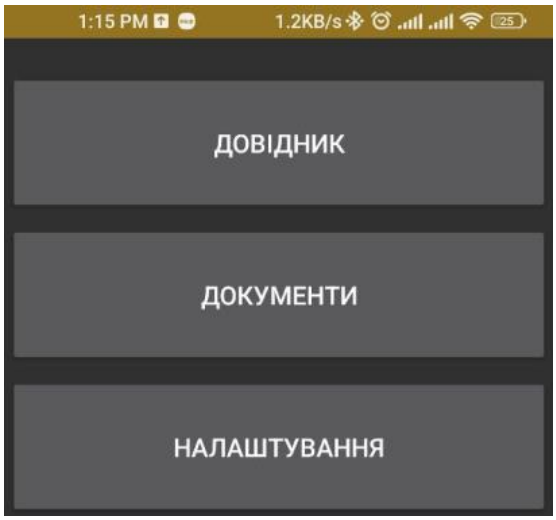

Рисунок 3.1 — Головний екран додатку

Кожна з кнопок на головному екрані додатку відповідає за перехід до відповідного розділу, забезпечуючи зручну навігацію для користувача. Для належної роботи програми необхідно підключення до Інтернету, оскільки додаток здійснює взаємодію з сервером для отримання та відправки даних. У разі відсутності з’єднання з мережею користувач отримає повідомлення про помилку, яке з’являється в нижній частині екрана (див. рис. 3.2):

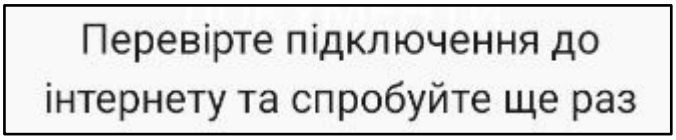

Рисунок 3.2 — Повідомлення про помилку при відсутності з’єднання з Інтернетом

Головний екран додатку виконаний в чорно-сірих відтінках з білим текстом, що гарантує високу контрастність. Це дозволяє користувачеві легко орієнтуватися в інтерфейсі, забезпечуючи зручність і читабельність кнопок і тексту під час роботи з системою.

Першим кроком для користувача є підключення до серверу через додаток. Для цього потрібно перейти на екран налаштувань, натиснувши кнопку «Налаштування» на головному екрані (див. рис. 3.3). На цьому екрані користувач може ввести необхідні параметри для з’єднання з сервером, що забезпечить подальшу взаємодію додатку з віддаленим сховищем даних.

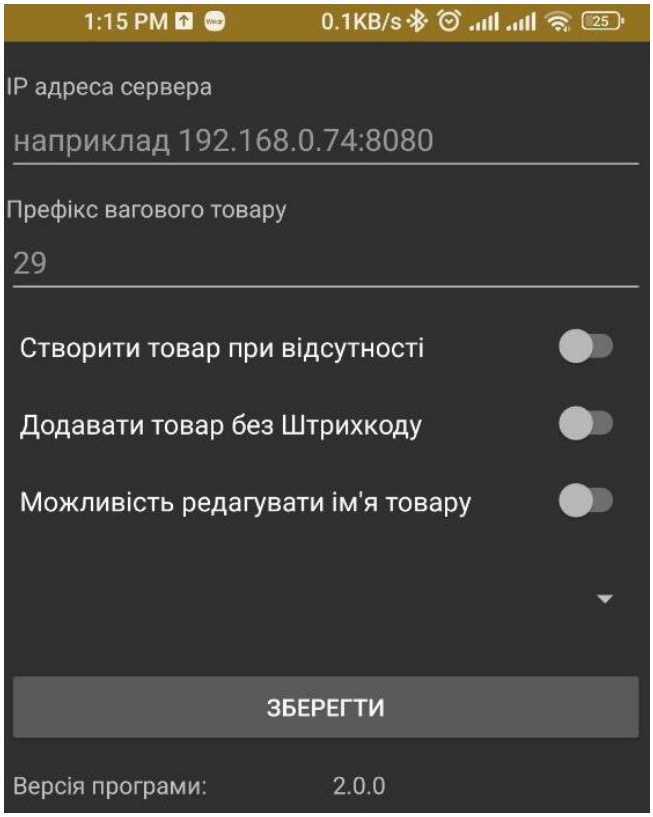

Рисунок 3.3 — Екран «Налаштування»

Екран налаштувань надає користувачеві можливість налаштувати основні параметри системи. На цьому екрані розташовані різноманітні елементи інтерфейсу, що дозволяють виконувати дії, такі як підключення до сервера, задання префіксу вагового товару, автоматичне створення зісканованого товару, маніпуляція та додавання товарів, редагування імені товару.

Підключення до сервера: на екрані є поле для введення IP-адреси сервера магазину (див. рис. 3.4). Після успішного підключення з’являється доступ до основних функцій додатку, таких як сканування товарів, створення документів та їх завантаження на сервер. У разі введення некоректної IP-адреси з’являється

повідомлення про помилку (див. рис. 3.5), яке буде відображатися в нижній частині екрану.

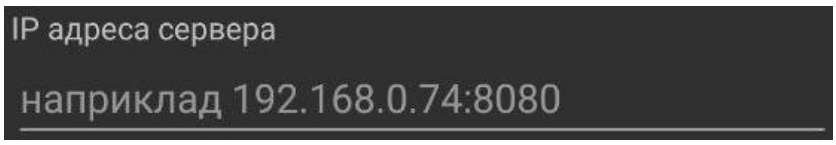

Рисунок 3.4 — Поле для введення IP-адреси сервера

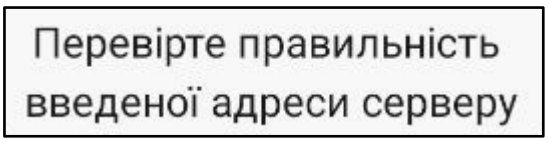

Рисунок 3.5 — Повідомлення про помилку при невдалому підключенні до сервера

Задання префіксу вагового товару: якщо магазин має вагові товари, користувач може ввести префік. товару (перших дві цифри штрих-коду) у спеціально відведене поле (див. рис. 3.6).

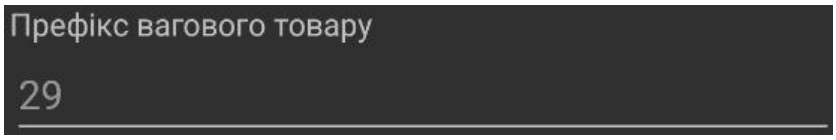

Рисунок 3.6 — Поле для введення префіксу вагового товару

Автоматичне створення зісканованого товару: при активації цього параметра (див. рис. 3.7), система автоматично створюватиме новий товар у базі даних, якщо такий товар ще не існує.

Рисунок 3.7 — Налаштування автоматичного створення товару у БД

Маніпуляція додавання товарів: якщо активувати цей перемикач (див. рис. 3.8), з’явиться можливість додавати товари до документа без необхідності наявності штрих-коду.

Рисунок 3.8 — Налаштування додавання товарів без штрих-коду

Редагування імені товару: після активації відповідного перемикача (див. рис. 3.9), користувач отримає змогу редагувати назву товару безпосередньо з мобільного пристрою, оновлюючи інформацію в базі даних.

Рисунок 3.9 — Налаштування можливості редагування назви товару

Після внесення необхідних змін користувач повинен натиснути кнопку «Зберегти», щоб застосувати нові налаштування.

Також є функція автоматичного завантаження складів та магазинів, якщо їх кількість перевищує один. Процес завантаження супроводжується повідомленням користувача про поточний стан (див. рис. 3.10).

Рисунок 3.10 — Процес завантаження складів

3.3 Інструкція експлуатації довідника

Наступний екран — «Довідник».

Рисунок 3.11 — Візуальний компонент для сканування товару

Даний екран дозволяє користувачу сканувати штрих-код товару або вводити його вручну через клавіатуру (див. рис. 3.11) для перевірки наявності цього товару в базі даних

Якщо товар знайдений у базі, на екрані відображається його інформація (див. рис. 3.12). Якщо ж товар не знайдено, користувач побачить повідомлення про помилку (див. рис. 3.13).

Рисунок 3.12 — Деталі про товар

Рисунок 3.13 — Помилка під час сканування товару

3.4 Інструкція експлуатації документів

Екран «Документи» відображає всі створені документи, які містять інформацію про відсканові товари (див. рис. 3.14). У випадку відсутності документів, на екрані з’явиться повідомлення про помилку (див. рис. 3.15).

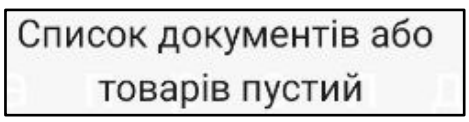

Рисунок 3.15 — Помилка у зв’язку з відсутністю документів

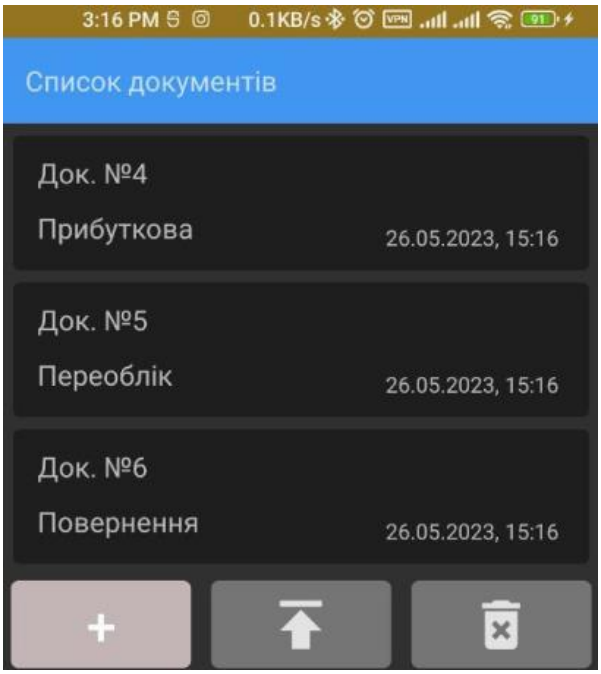

Рисунок 3.14 — Екран «Документи»

Для створення нового документа користувачеві потрібно натискати кнопку «Створити новий документ», що знаходиться внизу екрана (див. рис. 3.16).

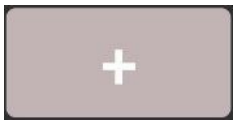

Рисунок 3.16 — Кнопка «Створити новий документ»

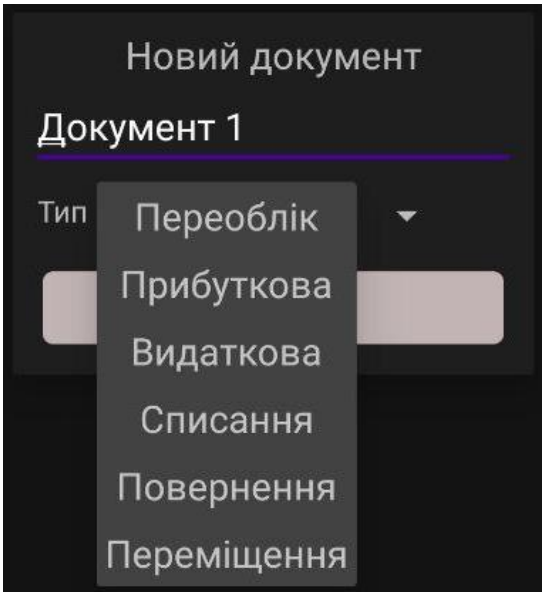

Рисунок 3.17 — Створення нового документу

При створенні нового документа необхідно ввести його назву (або вибрати вже запропоновану) і вибрати тип документа з випадаючого списку (див. рис. 3.17). Після цього новий документ з’явиться в кінці списку на екрані «Документи».

Щоб видалити документ, потрібно натиснути на кнопку «Видалити» (див. рис. 3.18) і затиснути відповідний елемент або кілька елементів списку документів (див. рис. 3.19).

Рисунок 3.18 — Кнопка «Видалити документ»

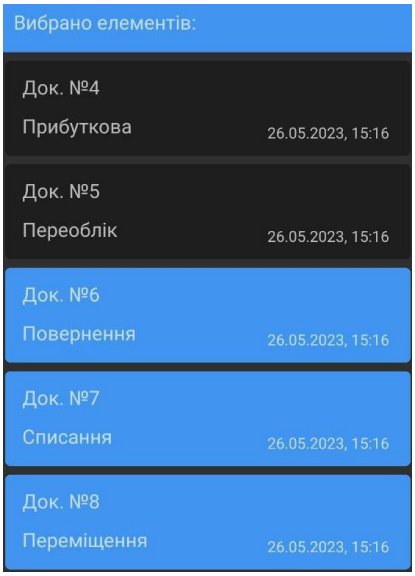

Рисунок 3.19 — Вибрані документи

Після цього користувач побачить повідомлення, яке запитує підтвердження або скасування дії (див. рис. 3.20).

Рисунок 3.20 — Повідомлення про видалення

Для того щоб відправити документ на віддалений сервер, користувачеві потрібно натиснути на кнопку «Завантажити» (див. рис. 3.21), вибравши при цьому відповідний елемент або кілька елементів зі списку.

Рисунок 3.21 — Кнопка «Завантажити документ»

Після натискання на кнопку, користувач побачить повідомлення про процес відправки документів, на якому буде запропоновано підтвердити або скасувати дію (див. рис. 3.22).

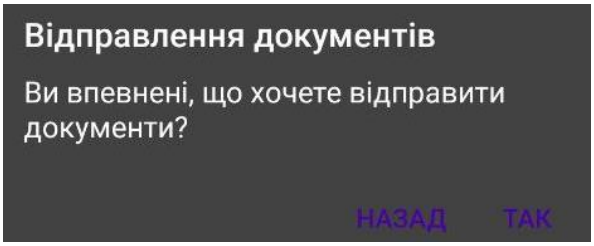

Рисунок 3.22 — Повідомлення про відправку документа

Після відкриття документа користувач побачить список товарів, доданих до цього документа. Крім того, він зможе сканувати товар та додавати його в документ, редагувати кількість, перевіряти ціну та інші характеристики товару.

4 ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ІДЕНТИФІКАЦІЇ ТОВАРІВ ЗА ШТРИХ-КОДАМИ ТА ОБІГУ СКЛАДСЬКИХ ДОКУМЕНТІВ

4.1 Методики тестування мобільних застосунків

Тестування мобільних додатків для платформи Android — це ключовий етап, спрямований на перевірку їхньої функціональності, якості та продуктивності. Цей процес має на меті виявлення помилок та забезпечення стабільної роботи додатка, що позитивно впливає на користувацький досвід. Завдяки ретельному тестуванню розробники можуть впевнитися в тому, що додаток відповідає високим стандартам якості та задовольняє очікування користувачів.

Серед основних типів тестування Android-додатків виділяють такі:

— функціональне тестування;
— UI / UX тестування;
— тестування сумісності;
— тестування продуктивності;
— тестування безпеки;
— тестування автоматизації.

Функціональне тестування: забезпечує перевірку всіх функцій додатка, щоб гарантувати їхню коректну роботу відповідно до специфікацій. Включає перевірку навігації, управління, функціональних можливостей та користувацьких дій.

UI / UX тестування: оцінює зручність і привабливість інтерфейсу для користувача. Перевіряється простота використання, естетика, відповідність елементів інтерфейсу платформним стандартам та легкість доступу до основних функцій.

Тестування сумісності перевіряє роботу додатка на різноманітних пристроях та версіях ОС Android. Це тестування охоплює відповідність різним

роздільним здатностям екранів, апаратним особливостям і конфігураціям, щоб забезпечити універсальність програми.

Тестування продуктивності визначає швидкість, стабільність та оптимізацію додатка в різних умовах. Включає тести на швидкість завантаження, час реакції на дії користувача та ефективність використання ресурсів, зокрема батареї.

Тестування безпеки зосереджується на захисті додатка від можливих загроз, зокрема від несанкціонованого доступу, втрати даних або вразливостей у мережевому з’єднанні.

Тестування автоматизації передбачає застосування спеціальних інструментів для автоматичного запуску тестів, що значно спрощує процес перевірки, підвищує точність результатів і прискорює цикл тестування.

Такий всебічний підхід до тестування Android-додатків дозволяє мінімізувати ризики, пов'язані з проблемами продуктивності та безпеки, що забезпечує кінцевий продукт високої якості.

4.2 Тестування основного функціоналу системи

Перевірка функціональності додатку представлена у таблиці 4.1. Дана перевірка демонструє відповідність функціональних компонентів додатку очікуванням та вимогам, підтверджуючи стабільність та надійність основних аспектів взаємодії.

Таблиця 4.1 — Тестування загальної функціональності додатку

Опис тест-кейсу	**Очікуваний результат**	**Фактичний результат**	**Статус**
Навігація в додатку	Безперебійне та коректне переміщення між усіма екранами додатка	Переміщення між екранами без збоїв	Тест успішний
Навігація назад	Користувач може повернутися до попереднього екрана за допомогою відповідних кнопок або жестів	Відбувається повернення до попереднього екрана за допомогою навігаційних елементів	Тест успішний

Продовження таблиці 4.1

Взаємодія з кнопками	Кожна кнопка виконує очікувану дію при натисканні	Кнопки функціонують згідно з очікуваннями	Тест успішний
Відображення інтерфейсу	Всі елементи інтерфейсу коректно відображаються на екрані, дані підтягуються належним чином	Інтерфейс і дані відображаються коректно	Тест успішний

У даному додатку проводиться велика кількість маніпуляцій товарами та взаємодій із сервером, результат їх тестування відображений у таблиці 4.2.

Таблиця 4.2 — Тестування пошуку товарів на сервері

Опис тест-кейсу	**Очікуваний результат**	**Фактичний результат**	**Статус**
Сканування товару, що є у базі	Успішне сканування та відображення інформації про товар	Сканування успішне, інформація про товар відображається	Тест успішний
Сканування товару, відсутнього у базі	Виведення повідомлення про відсутність товару в базі	Відображено повідомлення про відсутність товару	Тест успішний
Введення штрих-коду товару, що є у базі	Успішне введення штрих-коду та відображення інформації про товар	Введення виконано, інформація про товар відображена	Тест успішний
Введення штрих-коду товару, відсутнього у базі	Виведення повідомлення про відсутність товару в базі	Відображено повідомлення про відсутність товару	Тест успішний

Додаток демонструє успішні результати тестування ключових функцій пошуку товарів, перевіряючи коректність взаємодії додатку із сервером. Усі тести підтвердили коректну роботу додатку при пошуку та обробці інформації про товари.

4.3 Тестування обліку документів та обробки помилок

У програмі є можливість створювати, завантажувати та видаляти з віддаленого серверу документи. Результат тестування даного функціоналу представлений у таблиці 4.3. Усі перевірки підтвердили стабільну роботу програмного засобу, забезпечуючи точність створення, видалення та відправки документів на віддалений сервер.

Таблиця 4.3 — Тестування маніпуляцій над документами

Опис тест-кейсу	**Очікуваний результат**	**Фактичний результат**	**Статус**
Створення нового документа	Успішне створення документа, який відображається у списку документів	Документ створено успішно, відображається у списку	Тест успішний
Видалення документа	Відображення повідомлення про видалення та успішне видалення документа	Відображено повідомлення та документ видалено	Тест успішний
Відправка документа на сервер	Відображення повідомлення про відправку та успішна передача документа на сервер	Повідомлення про відправку з’являється, документ передано	Тест успішний

Для можливості користування усіма функціями додатку користувач має з’єднуватись із віддаленим сервером, додавати IP-адреси сервера, мати доступ до мережі Інтернет. Мобільний застосунок має надавати стабільно працювати та надавати коричтувачу доступ до всіх функцій. Результат тестування підключення наведений у таблиці 4.4.

Таблиця 4.4 — Тестування наявності підключень.

Опис тест-кейсу	**Очікуваний результат**	**Фактичний результат**	**Статус**
Додавання існуючої IP-адреси сервера	Успішне підключення до сервера	Підключення до сервера встановлено	Тест успішний
Додавання неіснуючої IP-адреси сервера	Відображення повідомлення про помилку	Виведено помилку	Тест успішний
З’єднання з мережею Інтернет відсутнє	Відображення повідомлення про відсутність мережі	Виведено помилку	Тест успішний
З’єднання з мережею Інтернет активне	Стабільна робота додатку та доступ до всіх функцій	Користувач взаємодіє з додатком без перебоїв	Тест успішний

Таким чином, тестування інформаційної системи для ідентифікації товарів за штрих-кодами та обробки складських документів не виявило збоїв. Результати тестування підтвердили коректну, безперебійну роботу програми, що забезпечує якісну та надійну взаємодію користувача з системою.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку мобільного додатку для ідентифікації товарів за штрих-кодами та обігу складських документів. Особливістю розробки є вдосконалення методу автоматизації ідентифікації та обліку товарів за рахунок введення до уніфікації системи для роботи з різними обліковими платформами без необхідності додаткового налаштування, також можливості віддалено керування системою складського обігу. Аналогами розробки є програма для терміналу збору даних «IBS Мобільний Склад» — 3950 грн.

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

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

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
Ринкові переваги					
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження табл. 5.1

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

Продовження табл. 5.1

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

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	4	4
Наявність аналогів на ринку	3	4	3
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	4	3
Експлуатаційні витрати	4	3	4
Ринок збуту	4	3	3
Конкурентоспроможність	3	3	4
Фахівці з технічної і комерційної реалізації	4	4	3
Фінансування	4	3	4
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	3	4	4
Супровідна документація	3	4	4
Сума	42	43	43
Середньоарифметична сума балів	(42+43+43) / 3 = 42,67		

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

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 - 1 0	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок автоматизація процесів ідентифікації та обліку товарів на складах, визначення точного залишку товарів, виявлення можливих недостач або надлишків, запобігання помилкам при сортуванні, моніторинг умов зберігання та стану продукції.

5.2.Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

Основна заробітна плата розробників, яка розраховується за формулою:

$$З_0 = \frac{M}{T_p} \cdot t, \quad (5.1)$$

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

$T_р$ — число робочих днів за місяць, 23 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	25000	1086,96	28	30434,783
Програміст	23000	1000,00	28	28000,000
Всього				58434,78

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

Додаткову заробітну плату прийнято розраховувати як 11 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 11\ \% / 100\ \% \quad (5.2)$$

$$З_д = (58434{,}78 \cdot 11\ \% / 100\ \%) = 6427{,}83\ (грн.)$$

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

$$Нз = (Зо + Зд) \cdot 22\ \% / 100\% \quad (5.3)$$

$$Нз = (58434{,}78 + 6427{,}83) \cdot 22\ \% / 100\ \% = 14269{,}77\ (грн.)$$

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

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою (5.4):

$$A = \frac{Ц}{T_в} \cdot \frac{t_{вик}}{12}\ [грн], \quad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

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

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

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 24000 грн., термін його корисного використання згідно податкового законодавства — 2 роки, а термін його фактичного використання — 1,22 міс.

$$А_{обл} = \frac{24000}{2} \times \frac{1,22}{12} = 1217,391 \text{ грн.}$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 5.5. Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн. (11000 грн.), то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, а також вартість телефону Pixel 4 XL = 8000 грн., також менше 20000 грн., отже $B_{заг.}$ = 19000 грн.

Таблиця 5.5 — Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	24000	2	1,22	1217,391
Офісне обладнання (меблі)	20000	4	1,22	507,246
Приміщення	1000000	20	1,22	5072,464
Всього				6797,10

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

$$В_е = В \cdot П \cdot Ф \cdot К_П, \quad (5.5)$$

де В — вартість 1 кВт-години електроенергії для 1 класу підприємства з ПДВ в 2024 році для Вінницької області за даними Енера-Вінниця, В = 10,42 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,35 кВт;

Ф — фактична кількість годин роботи обладнання, годин;

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

$$В_е = 0,9 \cdot 0,35 \cdot 8 \cdot 28 \cdot 10,42 = 735,2352 \text{ (грн.)}$$

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

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

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

$І_в$ = 58434,78 * 65% / 100% = 37982,61 (грн.)

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

$$Н_{нзв} = (З_0 + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.7)$$

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

$Н_{нзв}$ = 58434,78 * 133 % / 100 % = 77718 (грн.)

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

$$B_{заг} = 58434,78 + 6427,83 + 14269,77 + 6797,10 + 19000 + 735,24 +$$

$$+ 37982,61 + 77718 = 221365,59 \text{ грн.}$$

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою (5.8):

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

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

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

$$ЗВ = 221365,59 / 0,5 = 442731 \text{ грн.}$$

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

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

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

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

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

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

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

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

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

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

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

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

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

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

$Ц_о$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$;

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

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

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0,8333$;

p — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2024 році ϑ =18%.

Припустимо, що при прогнозованій ціні 1800 грн. за одиницю, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 100 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року — на 4000 шт., протягом другого року — на 3500 шт., протягом третього року на 3000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П1 = (0\cdot100+(1800+100)\cdot4000)\cdot0,8333\cdot0,37)\cdot(1-0,18)=1820399,93 \text{ грн}$$

$$\Delta П2 = (0\cdot100+(1800+100)\cdot(4000 + 3500)\cdot0,8333\cdot0,37)\cdot(1 - 0,18)= 3602874,86 \text{ грн}$$

$$\Delta П3 = (0\cdot100+(1800+100)\cdot(4000+3500+3000)\cdot0,8333\cdot0,37)\cdot(1-0,18)=5044024,798 \text{ грн}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 10467299,58 грн.

Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

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

$$\text{ПП} = \sum_1^T \frac{\Delta \text{П}_\text{i}}{(1+\tau)^t}, \quad (5.10)$$

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

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

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

t — період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

ПП = (1820399,927/(1+0,1)1)+(3602874,856/(1+0,1)2)+(5044024,798/
/ (1+0,1)3) = 1654909,02 + 2977582,526 + 3789650,487 = 8422142,037 грн.

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

$$PV = k_{інв} * ЗВ, \quad (5.11)$$

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

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

$$PV = 2 * 442731 = 885462{,}36 \text{ грн.}$$

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

$$E_{абс} = ПП - PV, \tag{5.12}$$

$$E_{абс} = 8422142{,}037 - 885462{,}36 = 7536679{,}68 \text{ грн.}$$

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

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

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу (5.13):

$$E_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \tag{5.13}$$

де $T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = 3 \cdot \sqrt{1 + 7536679{,}68/885462{,}36} - 1 = 1{,}119$$

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

$$\tau = d + f \tag{5.14}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,09…0,14);

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

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

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

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} \tag{5.15}$$

$$T_{ок} = 1 / 1{,}119 = 0{,}89 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,89 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 442731 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,89 роки.

ВИСНОВКИ

Основна мета кваліфікаційної роботи полягала у створенні функціональної інформаційної системи для ідентифікації товарів за штрих-кодами та оптимізації процесів документообігу на складі. Протягом розробки було проаналізовано сучасні методи і технології для ідентифікації товарів і автоматизації складських операцій.

У першому розділі проведено дослідження та огляд існуючих рішень, а також вичвлено, що автоматизація складських процесів є актуальною у сучасному бізнесі, що підтверджує важливість розробленої інформаційної системи. Ідентифікація товарів за штрих-кодами виявилася ефективним та широко використовуваним методом, що сприяє оптимізації складського обліку та управління запасами.

У другому розділі спроектовано модель інформаційної системи для сканування товарів за штрих-кодами та обліку складських документів, визначено технології для розробки.

У третьому розділі наведено деталі реалізації інформаційної системи, розроблено інструкцію користувача, яка сприяє легкому освоєнню функціоналу та ефективному використанню програми.

У четвертому розділі експерементально протестовано розроблений продукт, в результаті програма показала високий рівень точності ідентифікації та надійність у реалізації складських процесів, забезпечуючи ефективність управління запасами. Отримані результати підтверджують, що створений програмний засіб є дієвою інформаційною системою для оптимізації складського обігу та ідентифікації товарів

У п’ятому розділі відображений розрахунок економічних витрат інформаційної системи. У результаті застосунок виявився дешевшим за показниками у порівнянні з аналогами.

Отримані результати свідчать про значущість і практичну цінність програми для вирішення завдань управління товарообігом і обліку на складах. Це

дослідження може слугувати основою для подальшого розвитку програмного засобу в сфері автоматизації складських процесів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Р. Бичківський, П. Столярчук, П. Гамула. Метрологія, стандартизація, управління якістю і сертифікація. Штрихове кодування продукції — Львів, 2004. 560 с.

2. ДСТУ 3144-95. Коди і кодування інформації. Штрихове кодування. Терміни та визначення. Київ, 1995. 29 с.

3. Л. Байдакова, І. Байдакова, Б. Губа , В. Плахотін, О. Шегинський. Теоретичні основи товарознавства. Луцьк, 2016. — 284 с.

4. ДСТУ 3359-96. Коди та кодування інформації. Штрихове кодування. Якість друку штрихкодових позначок. Загальні технічні вимоги та методи контролю.

5. Що таке термінал збору даних. URL: https://www.vostok.dp.ua/ukr/infa1/tsd/terminal_sbora_dannih/ (дата звернення: 28.05.2024)

6. Термінал збору даних CARIBE PL-40L 1D/2D. URL: https://medias.com.ua/catalog/Torgove_obladnannya/TerminalyZboruDanyh/terminal-zboru-danykh-caribe-pl-40l-2d/ (дата звернення: 30.05.2024)

7. Термінал збору даних Caribe PL-50L 2D з чекодруком. URL: https://medias.com.ua/catalog/Torgove_obladnannya/TerminalyZboruDanyh/terminal-zboru-danykh-caribe-pl-50l-2d-z-chekodrukom/ (дата звернення: 29.06.2024)

8. Термінал збору даних Zebra TC25. URL: https://medias.com.ua/catalog/Torgove_obladnannya/TerminalyZboruDanyh/terminal_zboru_danykh_zebra_tc25/ (дата звернення: 29.06.2024)

9. Android Studio. URL: https://uk.wikipedia.org/wiki/Android_Studio (дата звернення: 01.07.2024)

10. The Pros and Cons of Android Studio. URL: https://www.pangea.ai/dev-mobile-app-resources/the-pros-and-cons-of-android-studio-and-app-tools/ (дата звернення: 01.07.2024)

11. Android. URL: https://uk.wikipedia.org/wiki/Android (дата звернення: 01.07.2024).

12. SDK. URL: https://uk.wikipedia.org/wiki/SDK (дата звернення: 03.08.2024)

13. Android SDK. URL: https://en.wikipedia.org/wiki/Android_SDK (дата звернення: 03.08.2024)

14. AVD. URL: https://developer.android.com/studio/manage-avds (дата звернення: 05.08.2024)

15. Java. URL: https://uk.wikipedia.org/wiki/Java (дата звернення: 08.09.2024)

16. Java SE 11. URL: https://codeguida.com/post/1519 (дата звернення: 08.09.2024)

17. Oracle Boosts Software Development Productivity with New Java Release. URL: https://www.oracle.com/corporate/pressrelease/java-11-092518.html (дата звернення: 09.09.2024)

18. Kotlin. URL: https://uk.wikipedia.org/wiki/Kotlin (дата звернення: 10.09.2024)

19. What are The Benefits of Kotlin. URL: https://sagaratechnology.medium.com/what-are-the-benefits-of-kotlin-d7fdcd1cfc0 (дата звернення: 12.09.2024)

20. ООП. URL: https://uk.wikipedia.org/wiki/OOP (дата звернення: 19.09.2024).

21. Архітектура MVP. URL: https://uk.wikipedia.org/wiki/Model-View-Presenter (дата звернення: 10.10.2024)

22. REST. URL: https://uk.wikipedia.org/wiki/REST (дата звернення: 10.10.2024)

23. Бойчик І.М. Економіка підприємства. Навчальний посібник. Київ, 2004. 480 с.

24. Когут Ю.М. Корпоративна безпека. Київ, 2018ю 276 с.

25. Іванюта Т.М., Заїчковський А.О. Економічна безпека підприємства. Львів, 2017. 256 с.

26. Правдюк Н.Л., Коваль Л.В., Коваль О.В. Облікова політика підприємств. Львів, 2020. 648 с.

27. Голов С.Ф. Управлінський облік. Львів, 2019. 400 с.

28. Фаріон І.Д., Писаренко Т.М. Управлінський облік. Львів, 2019. 792 с.

29. Гура Н.О., Мельник Т.Г., Моторина Т.М. Облік на підприємствах малого бізнесу. Київ, 2007. 310 с.

30. Гура Н.О., Мельник Т.Г. Облік на підприємствах малого бізнесу. Львів, 2019. 288 с.

31. Галісеєв Г.В. Системне програмування. Київ, 2019. 113 с.

32. Кривий С.Л. Вступ до методів створення програмних продуктів. Чернівці, 2012. 113 с.

33. Бородкіна І.Л., Бородкін Г.О. Інженерія програмного забезпечення. Посібник для студентів вищих навчальних закладів. Львів, 2018. 204 с.

34. Nita H. Shah, Mandeep Mittal. Soft Computing in Inventory Management. Berlin, 2021. 228 p.

35. Oliver Drobnik. Barcodes with iOS: Bringing together the digital and physical worlds. New York, 2015. 248 p.

36. Jakada Imani, Monna Wong. Management In a Changing World: How to Manage for Equity, Sustainability, and Results. New Jersey, 2023. 256 p.

37. Peter Royal. Building Modern Business Applications: Reactive Cloud Architecture for Java, Spring, and PostgreSQL. Pune, 2023. 189 p.

38. Oswald Campesato. Data Structures in Java. Virginia, 2023. 248 p.

39. Marc Loy, Patrick Niemeyer, Daniel Leuck. Learning Java: An Introduction to Real-World Programming with Java, 6th Edition (5th Early Release). California, 2023. 497 p.

40. Yashavant Kanetkar. Let Us Java - 6th Edition. Noida, 2023. 496 p.

41. Jacquie Barker. Beginning Java Objects: From Concepts to Code, 3rd Edition. Pune, 2023.845 p.

42. Jeanne Boyarsky, Scott Selikoff. OCP Oracle Certified Professional Java SE 11 Developer Complete Study Guide: Exam 1Z0-815, Exam 1Z0-816, and Exam 1Z0-817. California, 2020. 1296 p.

43. Chanaka Fernando. Solution Architecture Patterns for Enterprise: A Guide to Building Enterprise Software Systems. Pune, 2023. 388 p.

44. Eran Boudjnah. Clean Architecture for Android: Implement Expert-led Design Patterns to Build Scalable, Maintainable, and Testable Android Apps. Noida, 2023. 397 p.

45. Peter Spath. Pro Android with Kotlin: Developing Modern Mobile Apps with Kotlin and Jetpack, 2nd Edition. Pune, 2022. 881 p.

46. The Complete Android User Manual - 17th Edition 2023. Melbourne, 2023. 142 p.

47. Neil Smyth. Android Studio Dolphin Essentials - Java Edition: Developing Android Apps Using Android Studio 2021.3.1 and Java. New York, 2022. 806 p.

48. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Boston, 1995. 396 p.

49. Tony Gaitatzis. Learn REST APIs: Your guide to how to find, learn, and connect to the REST APIs that powers the Internet of Things revolution. New Jersey, 2019. 110 p.

50. Mike Amundsen. Restful Web API Patterns and Practices Cookbook. California, 2022. 468 p.

ДОДАТОК А

Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д..
« » _____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи
«Комп'ютерна система моніторингу регіонального радіомовлення»
08-54.МКР.032.00.000 ТЗ

Науковий керівник: доцент к.т.н.
_____ Кожем'яко А.В.

Студент групи 2КІ-23м
_____ Палагнюк В.І.

ВНТУ 2024

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність теми магістерської роботи визначається зростаючою необхідністю бізнесу та організацій у вдосконаленні процесів управління складськими операціями та забезпеченні ефективного контролю за товарними потоками. На сучасному етапі розвитку цифрових технологій автоматизація облікових процесів, впровадження систем ідентифікації товарів за штрих-кодами та підвищення точності обробки даних є пріоритетними завданнями для різних галузей. Постійне розширення спектра завдань, які потребують вирішення, сприяє актуалізації складніших функцій.

1.2 Наказ про затвердження теми МКР

2 Мета МКР і призначення розробки

2.1 Мета роботи — дослідження сучасних підходів і технологій у сфері автоматизації процесів обліку складських операцій, а також створення мобільного додатку, який забезпечує ідентифікацію товарів за штрих-кодами та облік складської документації для оптимізації управління товарними запасами

2.2 Призначення розробки — мобільний додаток розроблено для автоматизації облікових процесів на складах, включаючи ідентифікацію товарів за штрих-кодами, оперативне ведення складських записів, формування звітності, а також моніторинг залишків продукції в реальному часі. Програмний продукт спрямований на підвищення точності даних, зменшення ймовірності помилок та забезпечення можливості інтеграції з існуючими інформаційними системами компаній

3 Вихідні дані для виконання МКР

3.1 Дослідження сучасних рішень і підходів до автоматизації складських операцій, із акцентом на технології штрих-кодів для ідентифікації товарів;

3.2 Проєктування архітектури та створення концептуальної моделі мобільного додатку, призначеного для розпізнавання штрих-кодів і управління складськими документами;

3.3 Виконання моделювання основних функціональних модулів додатку, розробка методів для обробки даних товарів і синхронізації їх із базою даних;

3.4 Проведення економічного аналізу доцільності впровадження розробки, включаючи розрахунок потенційного скорочення помилок у процесах обліку та підвищення продуктивності роботи складу;

3.5 Вивчення законодавчих та нормативних вимог до розробки програмних рішень, включаючи забезпечення захисту даних від несанкціонованого доступу. Розробка рекомендацій щодо забезпечення стабільної та безпечної роботи додатку в реальних умовах експлуатації.

4 Вимоги до виконання МКР

Головна вимога — застосування сучасних технологій і методів для ідентифікації товарів за штрих-кодами, забезпечення точного обліку складських документів та оптимізація складських операцій.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в таблиці А.1

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз напрямків досліджень, основних технічних рішень та огляд існуючих рішень	08.09.2024	24.10.2024	Розділ 1
2	Моделювання аріхітектури інформаційної системи	25.10.2024	23.11.2024	Розділ 2

Продовження таблиці А.1

2	Моделювання аріхітектури інформаційної системи	25.10.2024	23.11.2024	Розділ 2
3	Розробка логіки програмного засобу	24.11.2024	08.12.2024	Розділ 3
4	Тестування реалізованого програмного засобу	09.12.2024	11.12.2024	Розділ 4
5	Економічна частина	12.12.2024	16.12.2024	Розділ 5

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

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

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

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

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

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

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

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

ДОДАТОК Б

Архітектура віртуальної машини Java

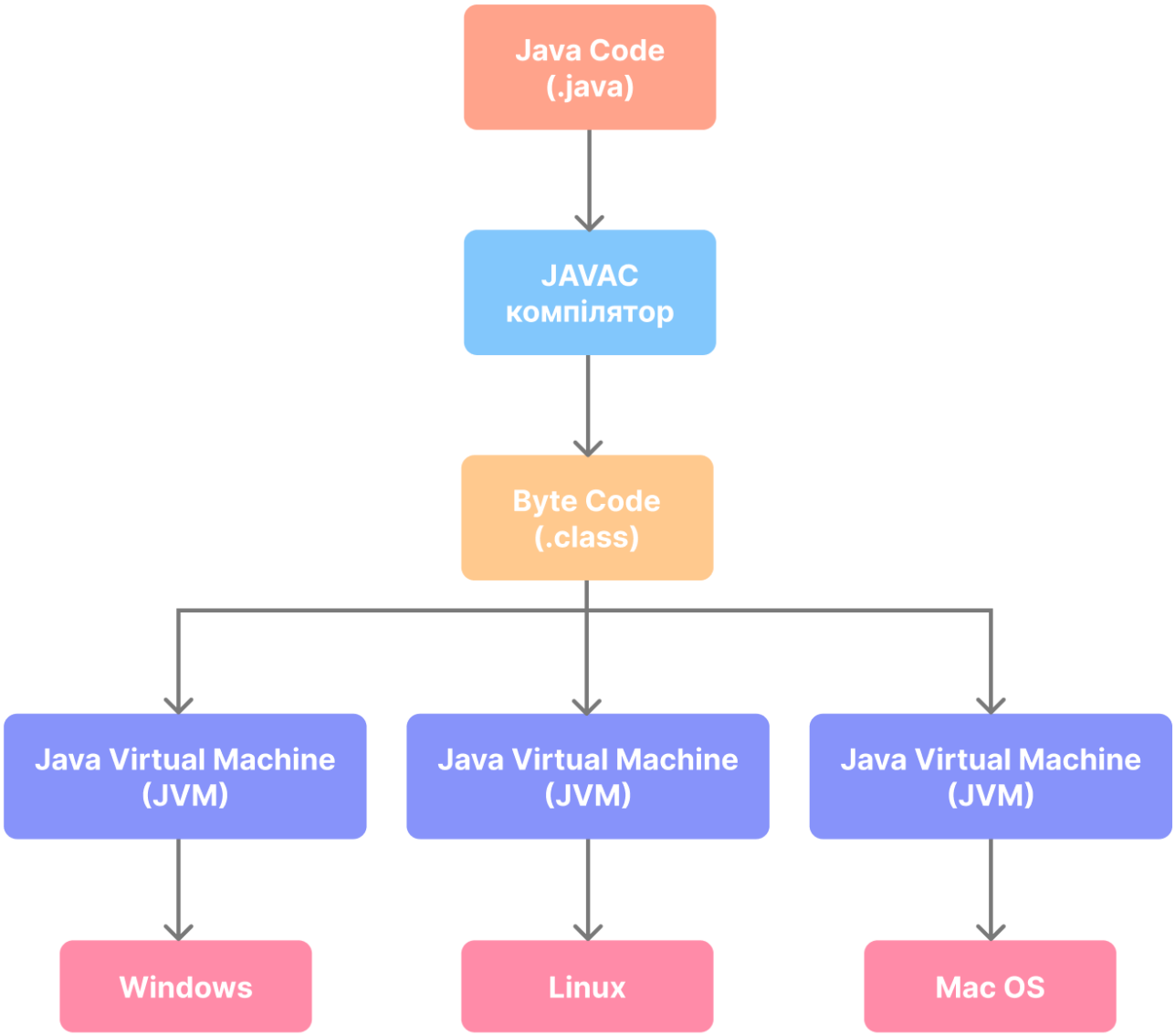

Рисунок Б.1 — Схема архітектури віртуальної машини Java

ДОДАТОК В

Архітектура Java Development Kit

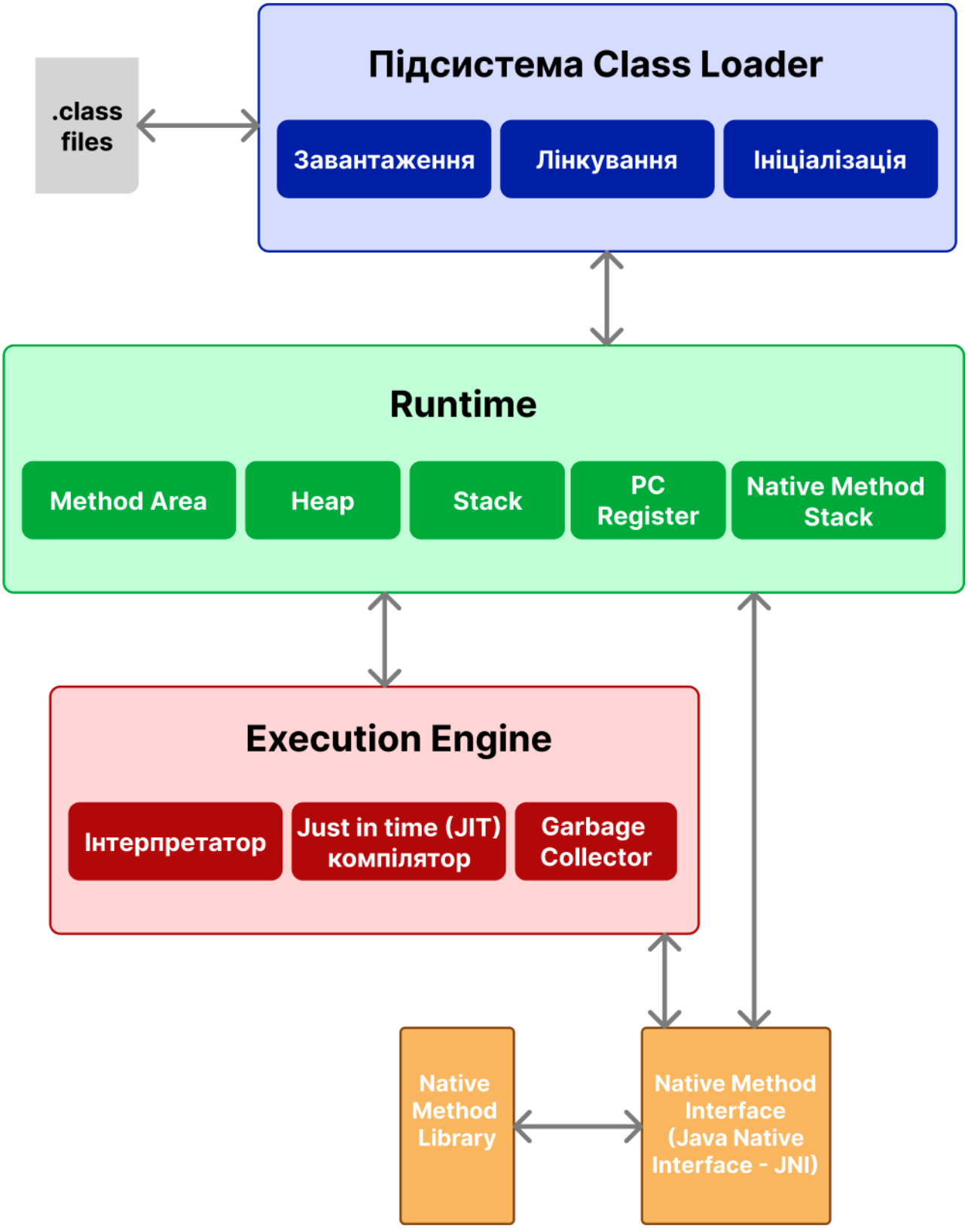

Рисунок В.1 — Схема архітектури Java Development Kit

ДОДАТОК Г

Алгоритм створення документу

Рисунок Г.1 — Схема алгоритму створення документу

ДОДАТОК Д

Етапи виконання підсистеми сканування штрих-кодів

Рисунок Д.1 — Схема етапів виконання підсистеми сканування штрих-кодів

ДОДАТОК Е

Інтерфейс екрану списку документів

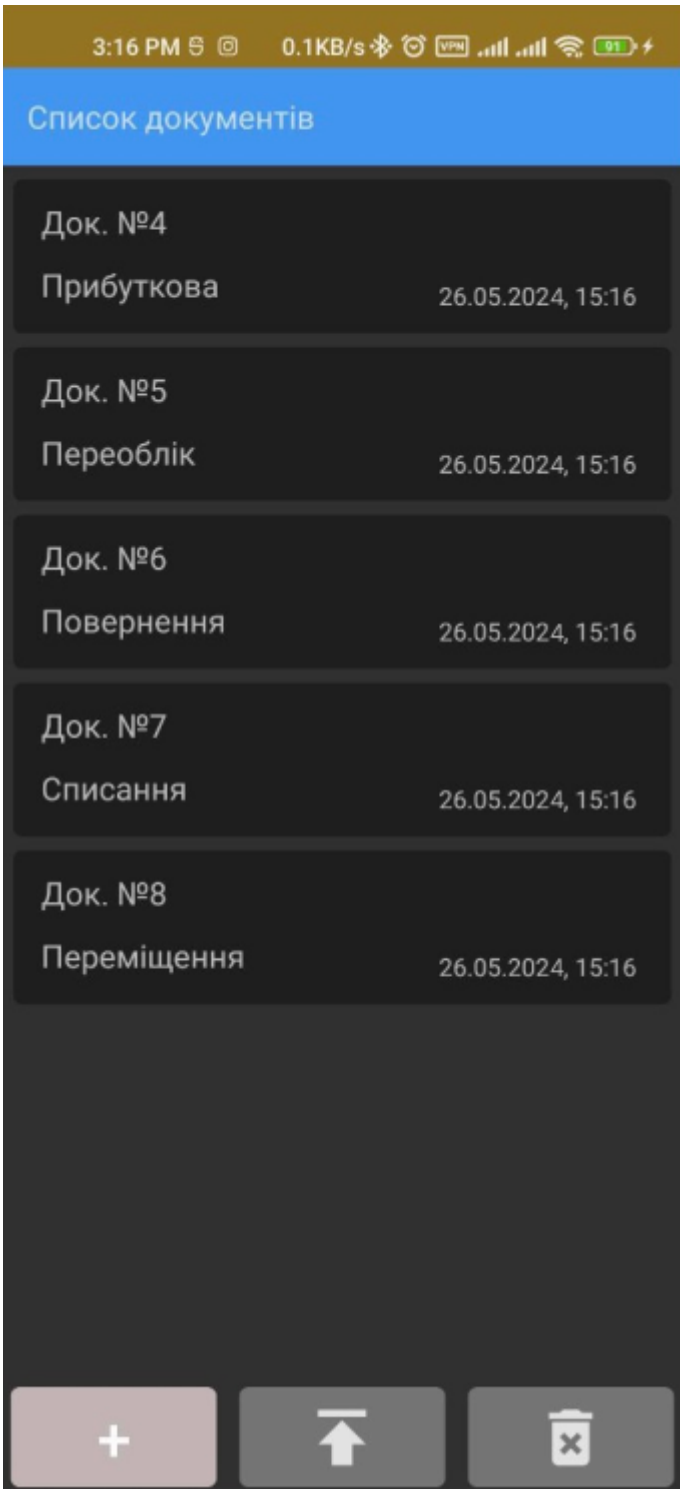

Рисунок Е.1 — Екран списку документів

ДОДАТОК Ж

Інтерфейс екрану налаштувань

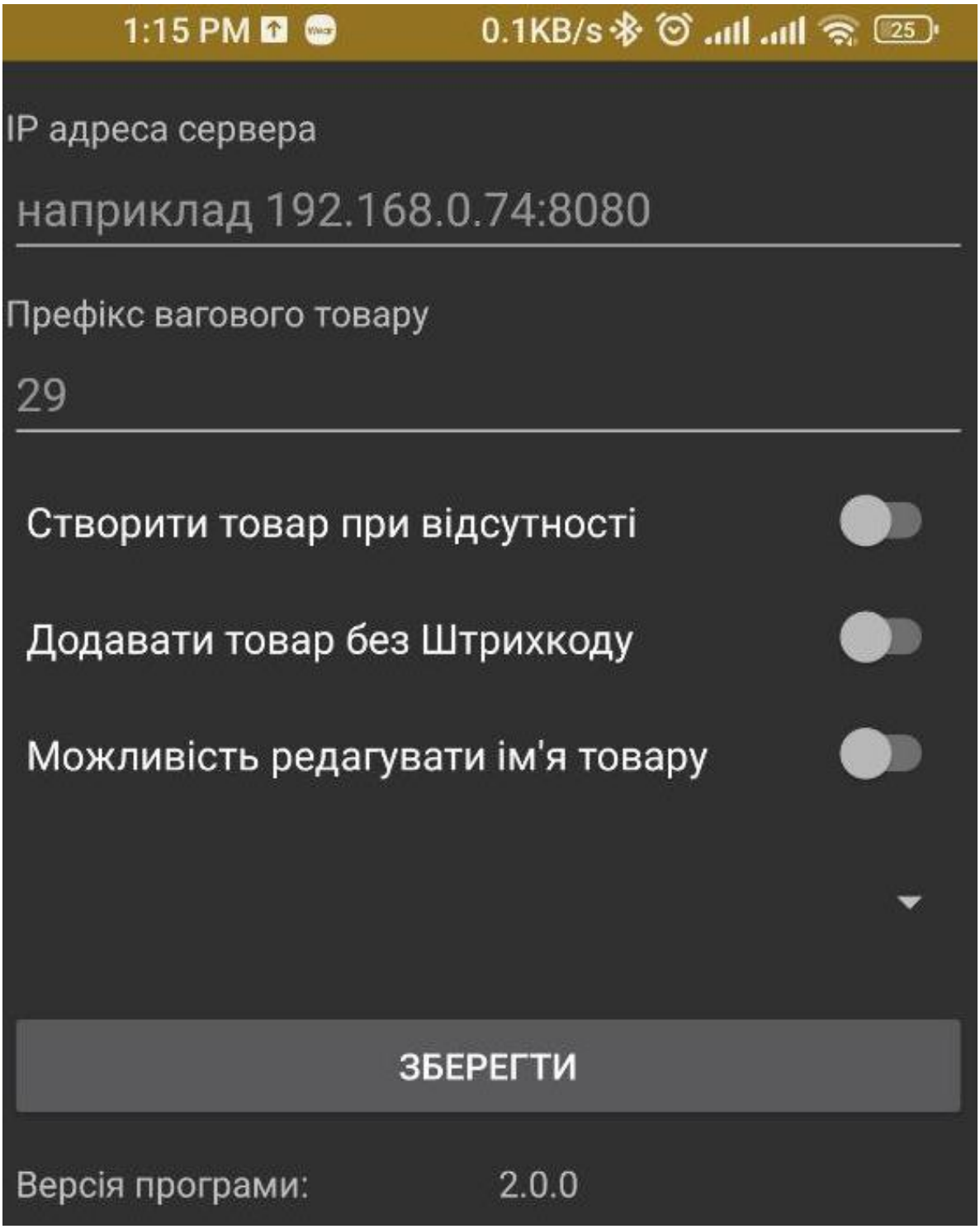

Рисунок Ж.1 — Екран налаштувань

ДОДАТОК И

Лістинг коду MainPresenter

Відповідає за обробку логіки та відображення компонентів з Activity.

```
class MainPresenter internal constructor(private val mView: MainActivity) :
MainShape.Presenter {

    private val mDialog: MyProgressDialog = MyProgressDialog(mView)
    private val TAG = MainPresenter::class.java.simpleName
    private val PERMISSIONS = arrayOf(Manifest.permission.INTERNET,
        Manifest.permission.ACCESS_NETWORK_STATE,
        Manifest.permission.ACCESS_WIFI_STATE,
        Manifest.permission.READ_PHONE_STATE,
        Manifest.permission.WRITE_EXTERNAL_STORAGE,
        Manifest.permission.READ_EXTERNAL_STORAGE)
    private var mHelper: DataBaseHelper? = null
    private var mDb: SQLiteDatabase? = null
    private var mValues: ContentValues? = null
    private var mGoodsDto: GoodsDto? = null
    private var startNum = 0
    private val packageSize = 5000

    private val goods: Unit
       get() {
          val hash = getMd5Hash(deviceImei)
          mGoodsDto = GoodsDto()
          RestController(mView).api?.getTable(hash, startNum, startNum +
packageSize)?.enqueue(object : Callback<GoodsDto?> {
             override fun onResponse(call: Call<GoodsDto?>, response:
Response<GoodsDto?>) {
                mGoodsDto = response.body()
                if (mGoodsDto != null) {
                   val goods: List<Good?>? = mGoodsDto!!.goods
                   if (goods!![0] != null) {
                      mView.showGoodsLoadingStatus(goods[goods.size -
1]!!.goodName)
                      putIntoDatabase(goods)
                   }
                }
             }

             override fun onFailure(call: Call<GoodsDto?>, t: Throwable) {
                Log.d(TAG, "onFailure: " + t.localizedMessage)
                mDialog.hide()
```

```
                }
            })
        }

    private fun initDB() {
        mHelper = DataBaseHelper(mView)
        mDb = mHelper!!.writableDatabase
    }

    //not granted
    private val deviceImei: String
        @SuppressLint("HardwareIds") get() {
            val telephonyManager =
mView.getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager
            if (ActivityCompat.checkSelfPermission(mView,
Manifest.permission.READ_PHONE_STATE) !=
PackageManager.PERMISSION_GRANTED) {
                return telephonyManager.deviceId
            } else {
                //not granted
            }
            return telephonyManager.deviceId
        }

    private fun putIntoDatabase(goods: List<Good?>?) {
        var numEndRow = 0
        var tableRowCount = 0
        var count = 0
        if (mDb != null) {
            mDb!!.beginTransaction()
            for (good in goods!!) {
                mValues = ContentValues().apply {
                    good?.let {
                        put(NAME, good.goodName)
                        put(GOODS_CODEGOODS1C, good.goodCode)
                        put(GOODS_BARCODE, good.barcode)
                        put(PRICE, good.goodPrice)
                        put(UNIT, good.unit)
                        put(ISWEIGHT, good.isWeight)
                    }
                }
                if (good != null) {
                    count = good.id
                }
```

```
                if (good != null) {
                    tableRowCount = good.tableRowCount
                }
                if (good != null) {
                    numEndRow = good.currentRow
                }
                try {
                    mDb!!.insert(TABLE_GOODS, null, mValues)
                } catch (e: Exception) {
                    mDialog.hide()
                }
            }
            mDb!!.setTransactionSuccessful()
            mDb!!.endTransaction()
            if (numEndRow < tableRowCount) {
                startNum = numEndRow + 1
                this.goods
                mDialog.setStatusMsg("Зачекайте: $count із $tableRowCount")
            } else {
                mDialog.hide()
            }
            Log.d("DB_ElementsCount", count.toString())
            mView.showNumStatus(goods[goods.size - 1]!!.currentRow.toString())
        } else {
            initDB()
            this.goods
        }
    }

    private fun clearBase() {
        try {
            mDb!!.execSQL("DELETE FROM $TABLE_GOODS")
            mDb!!.execSQL("VACUUM")
        } catch (e: Exception) {
            Log.e("db_Exception", e.toString())
        }
        initDB()
    }

    private fun askPermissions() {
        val PERMISSION_ALL = 1
        ActivityCompat.requestPermissions(mView, PERMISSIONS,
PERMISSION_ALL)
    }
```

```
override fun loadGoods() {
    checkPermission()
    val diaBox = AskOption()
    diaBox.show()
}

override fun checkPermission() {
    if (hasPermissions(mView, *PERMISSIONS)) {
        initDB()
    } else {
        askPermissions()
    }
}

private fun AskOption(): AlertDialog {
    return AlertDialog.Builder(mView)
            .setTitle("Журнал буде очищено")
            .setMessage("Ви впевнені, що хочете оновити базу?")
            .setPositiveButton("Так", object : DialogInterface.OnClickListener {
                override fun onClick(dialog: DialogInterface, whichButton: Int) {
                    mDialog.show()
                    clearBase()
                    goods
                }
            })
            .setNegativeButton("Назад") { dialog, which -> dialog.dismiss() }
            .create()
}

companion object {
    const val TABLE_GOODS = "Goods"
    const val ID = "_id" //Внутрішній код
    const val NAME = "name" //Назва товару
    const val GOODS_BARCODE = "barcode" //Штрих-код товару
    const val GOODS_CODEGOODS1C = "codeGoods1C" //Код товару на
базі 1С чи інший
    const val UNIT = "unit" //Одиниця
    const val PRICE = "price" //Ціна
    const val ISWEIGHT = "isWeight"
    private val methodGetGoods: MainApi? = null
    private fun getMd5Hash(input: String): String? {
        return try {
            val md = MessageDigest.getInstance("MD5")
```

```
                val messageDigest = md.digest(input.toByteArray())
                val number = BigInteger(1, messageDigest)
                var md5 = number.toString(16)
                while (md5.length < 32) md5 = "0$md5"
                md5.uppercase(Locale.getDefault())
            } catch (e: NoSuchAlgorithmException) {
                e.localizedMessage?.let { Log.e("MD5", it) }
                null
            }
        }

        fun hasPermissions(context: Context?, vararg permissions: String?):
Boolean {
            if (context != null) {
                for (permission in permissions) {
                    if (ActivityCompat.checkSelfPermission(context, permission!!) !=
PackageManager.PERMISSION_GRANTED) {
                        return false
                    }
                }
            }
            return true
        }
    }
}
```

ДОДАТОК К
ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Інформаційна система для розпізнавання товарів за штрих-кодами та управління складськими документами.**

Тип роботи: кваліфікаційна магістерська робота
Підрозділ: кафедра обчислювальної техніки, ФІТКІ, 2КІ-23м
Науковий керівник: Кожем'яко А.В.

Unicheck	
Оригінальність	**95%**
Схожість	5%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**
 Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
 Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методу та засобів сервісу генерації відеоконтенту на основі фільтрів».

Автор _____ Палагнюк В.І.
Опис прийнятого рішення: **допустити до захисту**

Особа відповідальна за перевірку _____ _____
(підпис) (прізвище, ініціали)

Експерт _____ _____
(підпис) (прізвище, ініціали)