МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Дослідження та розробка Android-органайзера для роботи з дисконтними картками і проїзними документами

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 123 – "Комп'ютерна інженерія"

Науковий керівник роботи:	_____ (підпис)	Щербакова М.Є. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Критська Я.О. (ініціали, прізвище)
Студент:	_____ (підпис)	Чмихало Р.С. (ініціали, прізвище)
Група:		КІ-17дм

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень магістр
Напрям підготовки
(шифр і назва)
Спеціальність 123 "Комп'ютерна інженерія"
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КНІ
д.т.н., доц. І.С. Скарга-Бандурова
«__» _____20__р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Чмихалу Роману Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та розробка Android-органайзера для роботи з дисконтними картками і проїзними документами

керівник проекту (роботи) доц. Щербакова М.Є.
(прізвище, ім'я, по батькові, науковий ступінь)
затверджені наказом вищого навчального закладу від "18" 10 2018 р. № 220/48
2. Строк подання студентом роботи
3. Вихідні дані до роботи Матеріали переддипломної практики

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Методи сканування і розшифрування штрих-кодів.
2) Дослідження методів реалізації органайзера дисконтних карток та проїзних документів
3) Реалізація android-органайзера для роботи з дисконтними картками та проїзними документами
4) Охорона праці та безпека в надзвичайних ситуаціях. Екологія
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
1)Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях. Екологія	ст. викл. Критська Я. О.		

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2018-01.03.2018	
2	Аналіз завдання, обробка отриманих даних	01.03.2018-01.04.2018	
3	Огляд літератури й обґрунтування необхідності дослідження та розробки	01.04.2018-15.04.2018	
4	Планування змісту проекту	15.04.2018-01.06.2018	
5	Аналіз та обрання методів та програмного забезпечення для проведення дослідження	01.06.2018-01.07.2018	
6	Проведення дослідження та розробка додатку	01.07.2018-01.11.2018	
7	Оформлення пояснювальної записки	01.11.2018-14.12.2018	
8	Оформлення розділу охорона праці	14.12.2018-05.01.2019	
9	Підготовка та подання магістерської роботи до захисту	05.01.2019-11.01.2019	

Студент _____ **Чмихало Р.С.**
(підпис) (прізвище та ініціали)

Науковий керівник _____ **Щербакова М.Є.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Чмихало Р.С. Дослідження та розробка Android-органайзера для роботи з дисконтними картками і проїзними документами

Розробляється програма-органайзер для роботи з дисконтними картами і проїздами документами для мобільних пристроїв, що працюють на базі операційної системи Android. Був представлений інтерфейс програми, а також основний функціонал програми. Був досліджений метод розпізнавання даних, представлених у вигляді штрих-коду. Розглянуто можливості сторонньої бібліотеки. Бібліотека, яка використовується в додатку, виконує всі вимоги поставленого завдання і дає можливість для розширення функціоналу в подальшому.

Ключові слова: Android, операційна система, сканер, штрих-код, QR-код, CardView, RecyclerView, Fragments, Google Vision API, органайзер.

АННОТАЦИЯ

Чмыхало Р.С. Исследование и разработка Android-органайзера для работы с дисконтными картами и проездными документами

Разрабатывается программа-органайзер для работы с дисконтными картами и проездными документами для мобильных устройств, работающих на базе операционной системы Android. Был представлен интерфейс программы, а также основной функционал программы. Был исследован метод распознавания данных, представленных в виде штрих-кода. Рассмотрены возможности сторонней библиотеки. Библиотека, которая используется в приложении, выполняет все требования поставленной задачи и дает возможность для расширения функционала в дальнейшем.

Ключевые слова: Android, операционная система, сканер, штрих-код, QR-код, CardView, RecyclerView, Fragments, Google Vision API, органайзер.

ABSTRACT

Chmykhalo R. S. Research and development of Android-organizer for work with discount cards and travel tickets

An organizer application is being developed for working with discount cards and travel documents for mobile devices based on the Android operating system. The application interface was presented, as well as the main functionality of the application. The method of recognition of data presented in the form of a bar code was investigated. Considered the possibility of a third-party library. The library, which is used in the application, fulfills all the requirements for the performance of the task and gives the opportunity for expanding the functional in further development.

Keywords: Android, operating system, scanner, barcode, QR-code, CardView, RecyclerView, Fragments, Google Vision API, organizer.

ЗМІСТ

ВСТУП

Актуальність теми. Платформа OS Android є однією з найпопулярніших в даний час. Охоплення користувачів по всьому світу досить велике, що дозволяє поширювати програми для більшого кола осіб. Станом на липень 2018 року Android займав 77,32% від загального ринку смартфонів. Крім смартфонів, різновиди даної операційної системи також встановлюються на інші пристрої, такі як смарт-годинники, планшети, телевізори Smart TV, приставки Android TV, а також вбудовуються в машини у вигляді додатку Android Auto.

Перебуваючи в мережі, споживачі звертаються за допомогою до пошукових серверів. Однак, поряд з тим, що Google намагається дати користувачам можливість мобільного пошуку, все ж залишається неохоплена територія: предмети з реального світу.

Це актуально саме там, де постійний інтерес і зростаюче очікування споживачів наштовхуються на грубу реальність безмовних будівель, товарів, магазинних вітрин, афіш та ін. Більшість цих об'єктів не здатні підтримувати комунікацію і відповісти на такі питання «Як? Хто? Де? коли і чому?» тобто, дати ту інформацію, на яку споживачі звикли розраховувати у мережі.

Ринок залучає масу інноваційних рішень для 'ask and know' («запитай і дізнаєшся») задач. Натхненні двовимірними штрих-кодами (barcode), виникли універсальні shot-коди, SMS-коди, QR-коди і UP-коди. Ці коди прикріплюються, наклеюються або друкуються на предметах, що має їх зробити «розумнішими», давши їм можливість надавати необхідну інформацію або просто відсилати людину до відповідної веб-сторінки.

Спочатку, штрих-коди несли в собі інформацію, за допомогою паралельних ліній і пропусків різної ширини, але сьогодні також використовуються візерунки з точок, концентричних кіл або ж вона буває зашифрована в якихось зображеннях. Хоча спочатку штрих-коди були створені, для передачі інформації про продукти рітейлерам, сьогодні ця технологія стрімко проштовхується до споживача (і навіть відтворюється самим споживачем) щоб надати тим, можливість спілкуватися зі своїм матеріальним оточенням.

У той час, як штрих-код має у своєму розпорядженні інформацію лише в одному єдиному напрямку, QR-код (QRcode) розміщує дані, як по вертикалі, так і по горизонталі, а значить, може вмістити більший обсяг інформації. По всьому світу більше мільярда людей вже встановили в свої смартфони спеціальнні додатки для зчитування QR-коду. Досить одного знімка і QR-квадратик з інформацією буде розшифрований, а користувач отримає через вбудований браузер додаткову інформацію про продукт: від веб-сторінки до тематичної відео-гри і рекламного відео-ролика.

Розробка додатку для зберігання штрих-кодів дисконтних карток є актуальною, тому що дозволяє користувачу не носити з собою справжні дисконтні картки.

Зв'язок з науковими програмами, планами, темами. Магістерська робота виконувалася протягом 2017-2018 рр. згідно з планами науково-дослідних робіт кафедри комп'ютерних наук та інженерії Східноукраїнського національного університету ім. В.Даля в межах НДР «Internet of Things: Emerging Curriculum for Industry and Human Applications (ALIOT)», реєстраційний номер 573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP.

Мета та задачі дослідження. Метою роботи є розробка і дослідження програмних засобів роботи з дисконтними картками і проїзними документами.

Для досягнення поставленої мети в роботі вирішуються такі задачі:

- дослідити методи розшифрування штрих-кодів та QR-кодів, а також їх реалізації за допомогою спеціалізованих бібліотек;
- виконати розробку органайзера дисконтних карт для ОС Android, що забезпечує можливість зберігати дані, відскановані з пластикових карт, квитків або інших носіїв інформації у вигляді штрих-коду;
- розроблений додаток має систематизувати дані в одному місці для зручності користувача. Дані будуть заноситися в програму за допомогою камери пристрою;
- сформувати зручний інтерфейс користувача.

Об'єкт дослідження: процес та технології розшифрування даних, записаних у вигляді штрих-коду або QR-коду.

Предмет дослідження: методи та засоби зберігання даних, записаних у вигляді штрих-коду або QR-коду.

Методи дослідження. Ґрунтуються на використанні мобільних технологій та методології системного аналізу. Використано методи моделювання для вивчення проблем розробки сканування та розшифрування штрих-коду. Також були застосовані теоретичні (аналіз предметної області, збір необхідних даних для реалізації поставленої мети), емпіричні (експертні оцінки тестування та вибір засобів розробки), а також статистичні методи дослідження.

Наукова новизна одержаних результатів. Набули подальшого розвитку методи застосування сучасних мобільних технологій, а саме бібліотек мови програмування Java та системи управління базами даних SQLite. Використання новітньої бібліотеки з розпізнавання елементів за допомогою камери дозволило пришвидшити роботу відносно інших додатків, що переважно працюють на старих та громіздких бібліотеках.

Практичне значення одержаних результатів. Органайзер дисконтних карт забезпечує можливість зберігати дані, відсканованні з пластикових карт, квитків або інших носіїв інформації у вигляді штрих-коду. Цей додаток систематизує дані в одному місці для зручності користувача. Дані заносяться в програму за допомогою камери пристрою.

Структура та обсяг магістерської роботи. Магістерська робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Робота викладена на 107 сторінках машинописного тексту, містить 60 сторінок основного тексту, 23 рисунків, 1 схема, додатки на 23 сторінках. Бібліографічний список включає 50 найменувань.

РОЗДІЛ 1
МЕТОДИ СКАНУВАННЯ І РОЗШИФРУВАННЯ ШТРИХ-КОДІВ

1.1 Аналіз вимог до розробки органайзера

Впровадження нових технологій для пересічних користувачів дозволяє позбутися від ряду вже застарілих дій, що вимагають особливої уваги. Додатки дозволяють робити повсякденність більш зручною. Органайзери дозволяють спростити і скоротити деякі повсякденні завдання. Органайзер - від самого початку невелика книга, що служить для організації персональних даних. Прикладом персонального органайзера такого типу можна назвати календар або записна книжка. В даний час технології дозволяють накопичувати безліч інформації на одному портативному пристрої, і є можливість перенести всі потрібні дані до нього для легкої доступності. Це може зберегти велику кількість часу і ресурсів, а також підвищити ефективність функцій, що виконуються за допомогою розширеного функціоналу.

Органайзер, що буде містити в собі велику кількість інформації, займати мало місця та пам’яті, виконувати повсякденні задачі та полегшити їх виконання, повинен витіснити стару функцію у зберіганні дисконтних пластикових карток та білетів.

1.2 Основні поняття про одномірні штрих-коди

Штрих-код був винайдений для автоматизації маркування товару і наступного зчитування інформації технічними засобами. Саме ці функції довгі роки відважно виконує штрих-код формату EAN-13, відомий нам як «звичайний» штрих-код. Інформація для автоматики (смуги) дублюється в зрозумілому людині вигляді (цифрах). Джерелом натхнення була азбука Морзе.

Штрих-коди можна побачити на всіх видах пакувань. Саме це зображення допоможе ідентифікувати той чи інший товар. При його допомозі зменшується час, необхідний для розшифрування даних про товар.

На чому ґрунтується штрихове кодування: Щоб зрозуміти, що таке штрих-код, треба звернути увагу на чергування чорних і світлих смуг з різною шириною (штрихи і прогалини). Призначені вони для сканера, мета якого - розшифрувати закладену в ці штрихи і прогалини інформацію і передати її на ПК. У сучасному світі, штрихове кодування - одне з найбільш часто зустрічаються засобів для визначення товару в автоматичному режимі. EAN - це метод

шифрування, який прийняла Європейська асоціація товарної нумерації. Якщо упаковка маленька, то в такому коді 8 цифр, в стандартній і великій упаковці - 13 цифр.

Більш докладно про ці цифри: Перші 2 або 3 цифри вказують країну, в якій було вироблено товар. Наприклад, 30 - це Франція, 520 - Греція, від 460 до 469 - Росія, 482 - Україна тощо).

Наступні 5 цифр - це унікальний код, який неодмінно повинен бути у виробника. Він отримує цей код від офіційного органу своєї країни.

Ті 5 цифр, що стоять наступними - це той код товару, що дається йому самим виробником. Це незалежний реєстраційний номер, в рамках підприємства, що виробляє товар. Тут сама компанія кодує дані, які допоможуть ідентифікувати цю одиницю товару (його назва, масу, розмір, інші параметри).

Цифра, що стоїть в кінці є контрольною. Потрібна вона для сканера, який з її допомогою розшифрує весь запис за допомогою особливого алгоритму, а також і для людини, яка знає, що таке штрих-код і самостійно може перевірити, наскільки правильна інформація.

Стандартна міра ширини штриха носить найменування «модуль». Розмір такої одиниці - 0.33 мм. Такі «модулі» складаються в штрихи і прогалини. 7 модулів, згрупованих за 2 штриха, і по 2 пробілу визначають будь-яку з цифр штрих-коду. Наприклад, «1011100» - буде зрозуміло пристроєм, що зробить сканування, як цифра «4». В ширину штрихи і прогалини варіюються від 1 до 3 модулів [1].

В ширині штрихів і прогалин, в розмаїтті їх послідовностей також містяться дані про унікальний код. Так, наприклад, прийнята для EAN-13 ширина всього кодового поля 37.29 мм, але простір, що займається безпосередньо штрихами і прогалинами - 31.35 мм., Тому що навколо них має бути вільний простір. По краях штрих-коду можна помітити трохи довші крайові штрихи - вони дають сканера зрозуміти, де починати і де закінчувати роботу.

Для розуміння того, що таке штрих-код важливо знати, що його функції не обмежуються винятково інформаційним значенням. Є ще й додаткові можливості:

1. Особливі пристрої швидко ідентифікують товари в автоматичному режимі;
2. За допомогою штрих-коду можна робити автоматизований підрахунок запасів товару;
3. Прискорений і зручний контроль за рухом товару: його навантаження, транспортування, розміщення на складі;
4. Більш якісне і прискорене обслуговування покупців;
5. Додаткові дані для досліджень з маркетингу;

Інструменти, за допомогою яких зчитують штрих-код:

1. Лазерні сканери самих різних габаритів. Як переносні, так і стаціонарні. Відповідно і штрих-код вони зчитують на різній відстані (від 60 см. До 6 метрів);
2. Контактні зчитувальні пристрої мініатюрних розмірів (за формою нагадують ручки, олівці, пістолети тощо);
3. Сканери на касах з пристосуваннями для зчитування штрих-коду;

Такі одномірні (лінійні) коди вміщають лише трохи інформації (до 20-30 символів), японці передбачаючи наближається переповнення буфера і брак пам'яті, задумалися про збільшення об’єму інформації, що передається. Так і народилися двомірні (матричні) коди, найпопулярнішим з яких і став QR-код. Відмінності з одновимірним штрих-кодом:

1. збільшення обсягу інформації на пару порядків;
2. інформація не дубльований зрозумілими людині символами;
3. збільшення обсягу породжує різноманітність форматів;

1.3 Переваги та недоліки QR-кодів

Штрих-код, розтягнутий до двомірного варіанту, залишається тим же штрих-кодом без будь-якої концептуальної новизни, але всі молоді японці вирішили, що це принципово нова технологія, настали нові часи. Раптово QR-кодами вирішили закрити існуючу проблему зв'язку між реальністю і віртуальним світом. Вони стали нерозривно пов'язані з мобільними телефонами і іншими гаджетами, портативними представниками віртуального світу під час подорожі людини по світу реального. Крім простору поза мережі (білборди, афіші, вивіски, преса, міські поверхні) QR-коди також стали використовуватися для перекидання інформації з комп'ютера на телефон, найчастіше адрес веб-сторінок і посилань на додатки.

Так технологія вузького професійного застосування (логістика, виробництво, торгівля) стала черговим способом кинути посилання на інформацію, яка не несе ніякої цінності, причому завуальовано. Після поширення технології, ми отримали світ з QR-кодами, технології що найбільше зловживається в світі.

З новим статусом технологія QR-кодів радикально віддаляється від своїх одновимірних предків. Принципова різниця полягає в зміні ролей системи «об'єкт-суб'єкт». У традиційній області застосування штрих-кодів зчитування (для користувача) є пасивним. У вас є товар, і касирка зчитує з неї штрих-код. У вас є квиток на концерт, і контролер на вході арени зчитує з нього штрих-код. У вас є квиток на метро, і турнікет зчитує з нього штрих-код. Ви пасивний учасник цього процесу. Від вас не потрібно докладати зусиль, володіти знаннями і навичками, мати обладнання. У вас є папірець з буквами і штрихуванням, і вам по більшою мірою все одно, що активний учасник робить з нею - сканує, уважно читає або перевіряє на просвіт. І при

такому пасивному використанні штрих-кодів користувач отримує якісну взаємодію - він не затримується і не вантажить собі мозок непотрібною інформацією, а всі труднощі з технологіями і апаратурою зчитування разом з грошима дістаються спеціально навченому персоналу - автоматизованим роботам і роботам під управлінням професіоналів органічного походження. Звичайно, з точки зору касирки і турнікета таке зчитування не є пасивним, але ми перш за все повинні дбати про простих користувачів, а касирка і турнікет професіонали, працюють на якісному промисловому обладнанні (або є ним) і отримують гроші за те, щоб людям було зручно. Тобто в тих областях, де використовуються звичайні штрих-коди, застосування QR-кодів цілком доречне.

Все змінюється, коли зчитування стає для користувача активним. Вся турбота перекладається на користувача: ось тобі візерунок, роби що-небудь. Користувач займає місце професіонала, повинен мати уявлення про технології та добути засоби зчитування цього візерунка. Чи треба при цьому нагадувати про жахливу різницю в якості і швидкості зчитування між професійними (пасивне зчитування) і призначеними для користувача (активне зчитування) сканерами? Порівняйте миттєву роботу оператора каси і ці паузи (особливо незручні при спробі зчитування коду, наприклад, з футболки незнайомця), коли камера смартфона намагається зловити фокус, а додаток нарешті знайти ці опорні квадратики.

Відмінні риси QR-кодів:

1. Висока ємність кодування даних

Хоча звичайні штрих-коди здатні зберігати максимум приблизно 20 цифр, QR-код може обробляти кілька десятків або декілька сотень разів більше інформації (рис 1.1).

QR-код здатний обробляти всі типи даних, такі як цифрові та буквені символи, кандзі, кана, хірагана, символи, бінарні та контрольні коди. До одного з символів кодується до 7 089 символів.

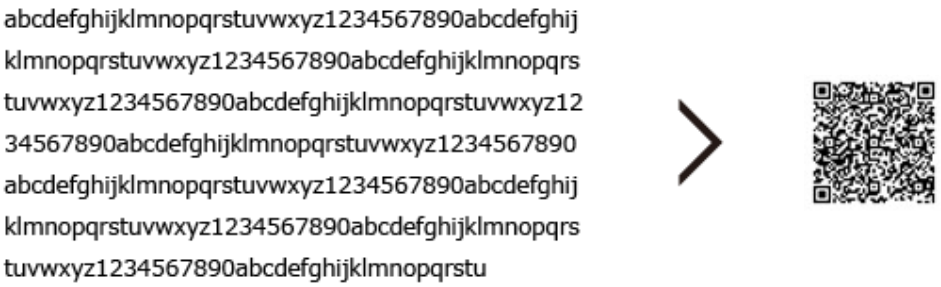


Рисунок 1.1 – Кількість даних в одному QR-коді

2. Малий розмір об’єкту з інформацією

Оскільки QR-код несе інформацію як по горизонталі, так і по вертикалі, QR-код здатний кодувати таку ж кількість даних приблизно в одну десяту від простору традиційного штрих-коду (рис. 1.2).

Рисунок 1.2 – Кількість даних в одному QR-коді у порівнянні зі штрих-кодом

3. Стійкість до бруду та пошкоджень

QR-код має функцію корекції помилок. Дані можуть бути відновлені, навіть якщо цей символ є частково брудним або пошкодженим. QR-код у якому 30% кодових слів відсутні або зіпсовані, може мати робочий код. QR-коди можуть бути згенеровані з 0%, 10%, 20% або 30% вбудованої корекцією помилок. Генерація з 30% корекцією додає багато шуму (екстра-блоки), але таким чином можна очистити 30% місця в коді і використовувати його для додавання логотипу або інших дизайнерських елементів[2].

Якщо ви використовуєте QR-код з 0% корекції помилок, код буде виглядати більш стрункo, але можливості для брендування коду дуже обмежені. Видалення або перешкоджання одного блоку в такому QR-коді може зробити його нечитабельним.

Математичні якості QR-кодів і застосування продуманого дизайну може дійсно підняти QR-коди до рівня, де код стає однією з центральних частин маркетингу. Застосування кращих методик дизайну сприятиме підвищенню швидкості зчитування та збільшить ефективність взаємодії з мережею.

4. Можливість зчитувати з будь-якого напрямку в 360°

QR-код здатний працювати на 360 градусів (у всьому напрямку), високошвидкісне читання. QR-код виконує це завдання за допомогою шаблонів виявлення позицій, розташованих у трьох кутах символу (рис. 1.3). Ці шаблони виявлення позицій гарантують стабільне високошвидкісне зчитування, обходячи негативні наслідки фонових перешкод.

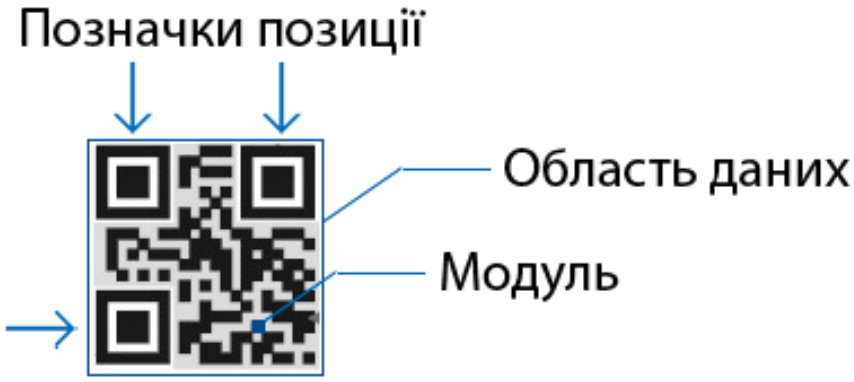


Рисунок 1.3 –Структура QR-коду

5. Функція структурового додавання

QR-код можна розділити на декілька областей даних (рис. 1.4). І навпаки, інформація, що зберігається в декількох символах QR-коду, може бути реконструйована як єдиний символ даних. Один символ даних можна розділити на 16 символів, що дозволяє друкувати у вузькій області.

Рисунок 1.4 – Функція структурового додавання QR-коду

1.4 Розшифрування QR-кодів

Щоб зрозуміти, як витягти дані з коду, потрібно розібратися в алгоритмі. Існує кілька стандартів в сімействі QR кодів, з їх базовими принципами можна ознайомитися в специфікаціях. Коротко поясню: дані, які необхідно закодувати, розбиваються на блоки в залежності від режиму кодування. До розбитих по блокам даним додається заголовок, який вказує на режим і кількість блоків. Існують і такі режими, в яких використовується більш складна структура розміщення інформації. Дані режими розглядати не будемо з огляду на те, що витягати вручну з них інформацію недоцільно. Однак, ґрунтуючись на тих принципах, які описані нижче, можна адаптуватися і до цих режимів.

На випадок некоректного читання даних, в QR застосовуються спеціальні коди, які здатні виправити недоліки при читанні. Це так звані коди Ріда-Соломона. Принцип обчислення кодів, а також виправлення помилок у блоках інформації розглядати не будемо. Коригувальні помилки коди Ріда-Соломона (RS) записуються після всіх інформаційних даних.

Це дуже спрощує завдання безпосереднього читання інформації: можна просто вважати дані, не чіпаючи коди. Як показує практика, зазвичай більшу частину QR -матриці займають коригувальні RS-коди [3].

За стандартом, дані з RS-кодами перед записом в картинку «перемішуються». Для цих цілей використовують спеціальні маски. Існує 8 алгоритмів, серед яких вибирається найкращий. Критерії вибору засновані на системі штрафів, про які можна також почитати в специфікації.

«Перемішані» дані записуються в особливій послідовності на шаблонну картинку, куди додається технічна інформація для декодер. Виходячи з описаного алгоритму, можна виділити схему вилучення даних з QR коду (сх. 1.1).

Схема 1.4.1 – Алгоритм розшифрування QR-коду

Тут підкреслені пункти, які потрібно буде виконати при безпосередньому читанні коду. Інші пункти можна опустити з огляду на те, що зчитування робить людина.

Поглянувши на зображення, можна помітити кілька виразних областей. Ці області використовуються для детектування QR коду (рис. 1.5). Ці дані не представляють інтересу з точки зору записаної інформації, але їх потрібно викреслити або просто запам'ятати їх розташування, щоб вони не заважали. Все інше поле коду несе вже корисну інформацію. Її можна розбити на дві частини: системна інформація і дані. Також існує інформація про версії коду. Від версії коду залежить максимальний обсяг даних, які можуть бути записані в код. При підвищенні версії - додаються спеціальні блоки.

Рисунок 1.5 – Області детектування QR коду

За ним можна зорієнтуватися і зрозуміти яка версія QR перед вами. Коди високих версій зазвичай також недоцільно зчитувати вручну.

Рисунок 1.6 – Розміщення системної інформації

Системна інформація дублюється, що дозволяє значно знизити ймовірність виникнення помилок при детектуванні коду і зчитуванні. Системна інформація - це 15 біт даних, серед яких перші 5 - це корисна інформація, а решта 10 - це БЧХ (Код Боуза - Чоудхурі - Хоквінгема) (15,5) код, який дозволяє виправляти помилки в системних даних (рис. 1.6). До класу БЧХ кодів відносять і RS коди. Зверніть увагу, що на малюнку дві смужки по 15 біт не перетинаються. Цікавість являють тільки перші 5 біт (рис. 1.7). З яких 2 біта показують рівень корекції помилок, а решта 3 біта показують яка маска з доступних 8 застосовується до даних.

Рисунок 1.7 – Читання 5 біт системної інформації

Крім уже озвучених схем захисту системної інформації, до того ж, використовується статична маска, яка застосовується до будь-якої системної інформації. Вона має вигляд: 101010000010010. Так як має цікавість тільки перші 5 біт, то маску можна скоротити і легко запам'ятати: 10101 (десять - сто один). Після застосування операції «виключає або» (Xor) отримуємо інформацію.

Щоб зрозуміти з даними доведеться мати справу, необхідно спочатку прочитати 4-х бітний заголовок, який містить в собі інформацію про режим (рис. 1.8).

Рисунок 1.8 – Читання заголовку даних

Список можливих режимів:

- ECI 0111
- Числові 0001
- Буквено-числові 0010
- 8-бітний (байтний) 0100
- Kanji 1000
- Структуроване доповнення 0011
- FNC1 0101 (1-я позиція)
  1001 (2-я позиція)

Після вилучення 4-х біт, що описують режим, необхідно до них застосувати маску.

Маска визначається виразом, наведеними в таблиці вище. Якщо цей вислів зводиться до TRUE (правильне) для біта з координатами (i, j), то біт інвертується, інакше все залишається без змін. Початок координат в лівому верхньому кутку (0,0). Поглянувши на вираження, можна помітити в них закономірності [3].

Після отримання даних про режим можна приступати до читання інформації. Треба обумовити про те, що найцікавіше зчитувати числові і комбіновані дані, так як вони легко інтерпретуються. Але також не варто боятися 8-бітних. Це може бути інформація, що також легко інтерпретується. Наприклад, багато онлайн генераторів QR текст кодують в цьому режимі, використовуючи ASCII. Ще одна причина, чому слід спочатку прочитати режим, це те, що від нього залежить кількість пакетів даних. Яка також залежить і від версії коду. Для версій з першої по дев'яту довжини блоків для більш читабельних режимів:

- Числові 10 біт/4 біта
- Буквено-числові 9 біт
- 8-бітний (байтний) 8 біт

Перший блок після вказівника режиму - це кількість символів. Для числового режиму кількість закодовано в 10 наступних бітах, а для 8-бітного режиму в 8 бітах.

На малюнку, в QR коді, записана цифра 5. Це видно за вказівником кількості символів і подальшим після нього 4 бітам. У числовому режимі поряд з 10-бітними блоками використовуються 4-х бітні блоки для економії місця, якщо в 10-бітному обсязі немає необхідності.

1.5 Аналіз програмних та інструментальних засобів

На ринку є достатня кількість програмно-технічних засобів з подібним функціоналом, але більшість з них має низький рівень виконання або малий функціонал. Тому можна виділити декілька мобільних додатків, що варті уваги.

В першу чергу слід звернути увагу на досить популярний, у останній час, додаток Google Pay [4]. Цей додаток з’явився не так давно на українському ринку, з офіційним запуском у Україні, весною 2018 року. Але основний ухил функціоналу йде на сплату картками у терміналах за допомогою безконтактних платежів, через чип NFC. Другою ж функцією цього додатку є зберігання дисконтних карток, яке виконує другорядне значення і має дуже контрольовану схему роботи зі сторони розробників додатку.

У додатку Google Pay можна зберігати цифрові копії карт постійного клієнта, а також бонусних, клубних і подарункових карт (Рис. 1.9). Крім того, ви можете додавати в нього спеціальні пропозиції, наприклад купони, промо-акції та знижки.

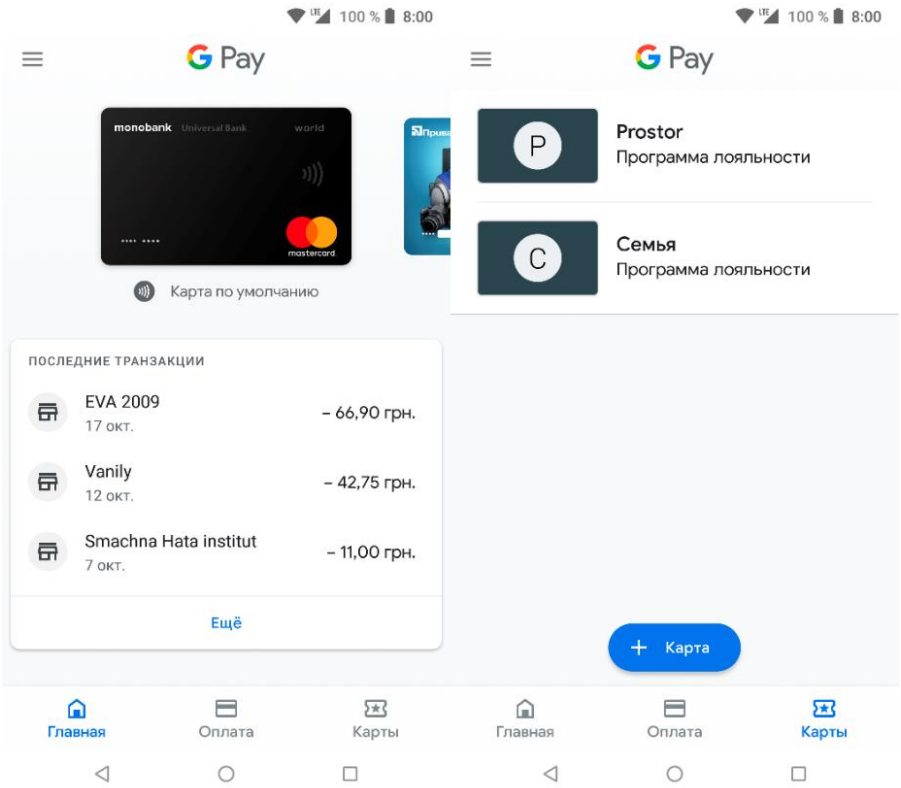


Рисунок 1.9 – Інтерфейс додатку Google Pay

Другий додаток це – Stocard [5]. Він має досить різноманітний функціонал. Може зберігати картки, прикріплювати карти, яких нема у системі, шукати найближчі магазини. Але має і декілька недоліків, що несуть деякі незручності у використанні.

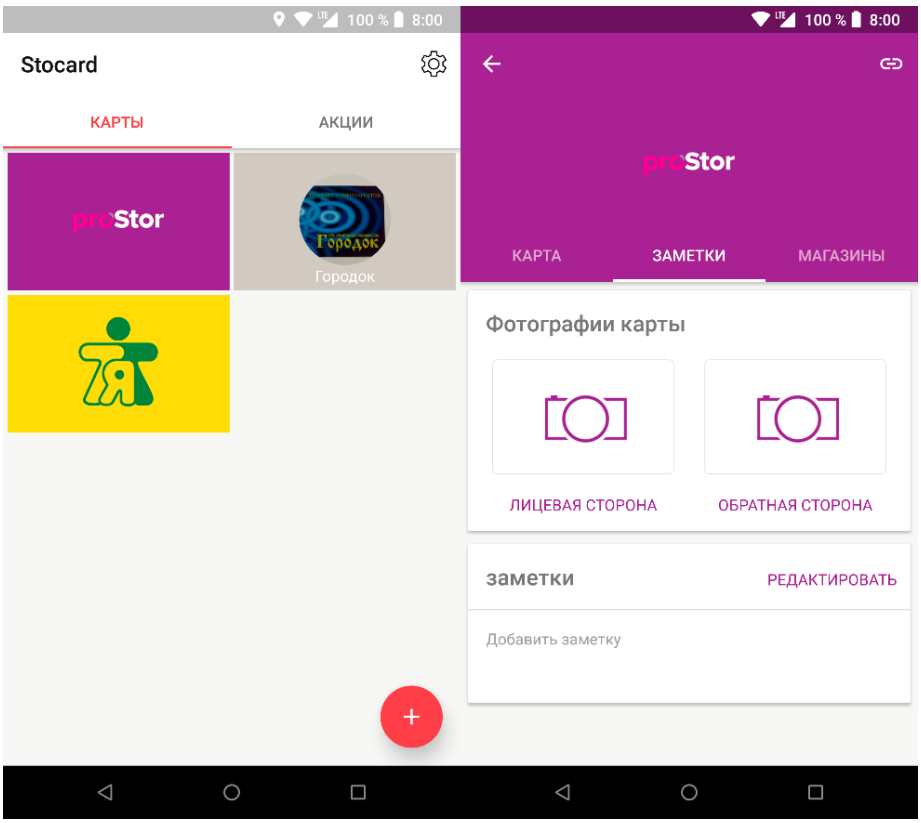


Рисунок 1.10 – Інтерфейс додатку Stocard

Для додавання дисконтної карти досить вибрати магазин зі списку, після чого або сканувати дисконтну карту, або ввести її номер власноруч. Після цього на касі можна буде показати штрих-код карти. До карток можна додавати замітки та фото карток (Рис. 1.10). Це може бути зручно, наприклад, якщо у вас кілька карт магазину, який надає додаткові знижки в дні народження. Якщо дозволити додатком визначати своє місце розташування, то воно іноді може показувати потрібну карту якщо ви підходите до магазином. На Андроїд цю карту видно буде прямо на екрані блокування. Нещодавно в додаток StoCard для Android додали можливість синхронізації між пристроями. Для цього можна користуватися аккаунтом Google, Facebook або навіть просто синхронізувати по e-mail. Це дозволяє декільком членам сім'ї тримати на смартфоні одні і ті ж дисконтні карти. При цьому видалення або додавання карти на одному пристрої призводить до синхронізації з усіма іншими.

Недоліком є відсутність можливості пошуку по вже доданим картками. Якщо у вас їх штук 10, то знайти потрібну не складе труднощів, а ось якщо їх кількість більше ніж за 50, то пошук однієї може також зайняти час.

1.6 Аналіз математичних моделей і методів для вирішення задачі

Для розробки додатку зі зчитування QR-кодів та Bar-кодів є декілька бібліотек, що допомагають у створенні додатків з подібним функціоналом.

1.6.1 Бібліотека ZXing

ZXing (вимовляється як " zebra crossing") - це бібліотека з відкритим вихідним кодом, для обробки зображень із штрих-кодами у декількох форматах (1D/2D), що реалізована в Java, з портами на інші мови. Його основна увага приділяється використанню вбудованої камери на мобільних телефонах для сканування та декодування штрих-кодів на пристрої, без спілкування з сервером, хоча бібліотека також підтримує використання на сервері [6].

Підтримувані формати:

1. 1D product: UPC-A, UPC-E, EAN-8, EAN-13;
2. 1D industrial: Code 39, Code 93, Code 128, Codabar, ITF, RSS-14, RSS-Expanded;
3. 2D: QR Code, Data Matrix, Aztec(beta), PDF 417(beta), MaxiCode.

1.6.2 Бібліотеки VS READER QR та VS BARCODE READER

VSBarcodeQR - це універсальна та високопродуктивна бібліотека для читання QR-коду для iOS та Android. Бібліотека та SDK дозволяють програмі швидко і надійно читати QR-коди

всіх видів та розмірів навіть у складних середовищах. Він призначений для безперебійної роботи поряд з бібліотекою VSBarcodeReader або в автономному режимі [7].

Ключові риси VSBarcodeQR:

1. Сканування у реальному часі, подібне до апаратного сканеру.
2. Читає QR-коди, навіть якщо вони трохи розмиті.
3. Підтримує всі типи QR (URL, SMS, телефон, візитна картка тощо).
4. Повна сумісність з VSBarcodeReader.
5. Ліцензія білої етикетки. Повний контроль над користувальницьким інтерфейсом.
6. Дуже простий в інтеграції, немає параметрів для налаштування.
7. Повністю сумісна з інструкціями розробників iOS7 + та Apple.
8. Підтримка iOS та Android.
9. Платна ліцензія.

VSBarcodeReader - це найсучасніша бібліотека для читання штрих-кодів для iPhone та Android. Бібліотека та SDK дозволяють додавати функціональні можливості сканування штрих-кодів у додатку за лічені хвилини. Фірмові алгоритми можуть читати розмиті UPC та EAN, сфотографовані камерою з фіксованим фокусуванням. Вона призначена для безперебійної роботи з бібліотекою VSReaderQR або в автономному режимі [8].

Ключові риси VSBarcodeReader:

1. Сканування у реальному часі, подібне до апаратного сканеру товару.
2. Читає штрих-коди UPC-A / E та EAN-13/8 (навіть розмиті) на iPhone, iPod Touch, iPad та всіх пристроях Android.
3. Підтримує Code39, Code128, Code93, Codabar, Interleaved 2 з 5 (ITF) та Стандарт 2 з 5 символів (тільки різкі штрих-коди).
4. Повна сумісність з VSBarcodeQR.
5. Ліцензія білої етикетки. Повний контроль над користувальницьким інтерфейсом.
6. Дуже простий в інтеграції, немає параметрів для налаштування.
7. Повністю сумісна з iOS7 +, Android 2.1+.
8. Платна ліцензія.

1.6.3 Mobile Vision API

Mobile Vision API пропонує фреймворк для пошуку об'єктів у фотографіях та відео. В основу входять детектори, які визначають і описують візуальні об'єкти у зображеннях або відеокадрах, а також події, керовані API, який відстежує положення цих об'єктів у відео.

В даний час програма Mobile Vision включає детектори обличчя, штрих-коду та тексту, які можна застосувати окремо або разом.

Barcode API виявляє штрих-коди в режимі реального часу, на пристрої, в будь-якій орієнтації. Він також може одночасно виявляти кілька штрих-кодів [9].

Він читає наступні формати штрих-кодів:

1. 1D barcodes: EAN-13, EAN-8, UPC-A, UPC-E, Code-39, Code-93, Code-128, ITF, Codabar
2. 2D barcodes: QR Code, Data Matrix, PDF-417, AZTEC

Він автоматично аналізує QR-коди, матрицю даних, PDF-417 та Aztec значення для наступних підтримуваних форматів:

- URL
- Contact information (VCARD, etc.)
- Calendar event
- Email
- Phone
- SMS
- ISBN
- WiFi
- Geo-location (latitude and longitude)
- AAMVA driver license/ID

Класи для виявлення та розбору штрих-кодів доступні в іменах com.google.android.gms.vision.barcode. Клас BarcodeDetector є основним об'єктом Frame Object для обробки робочого конуса, щоб повернути типи SparseArray <Barcode>.

Тип штрих-коду являє собою єдиний визнаний штрих-код та його значення. У випадку 1D штрих-коду, такого як UPC-коди, це буде просто цифра, кодована в штрих-коді (рис. 1.11). Це доступно в властивості rawValue, при цьому виявлений тип кодування встановлюється у полі формату.

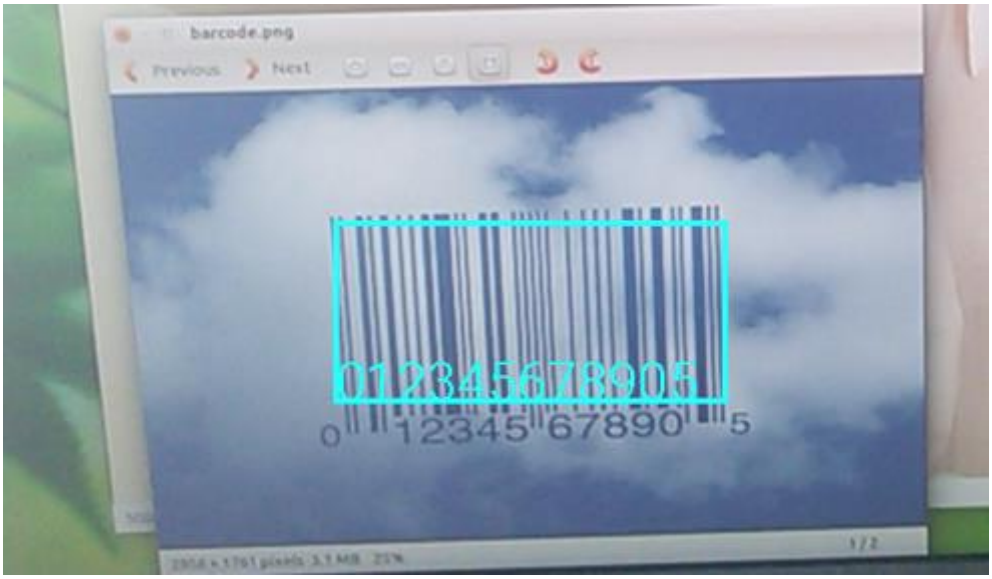


Рисунок 1.11 – Виявлення штрих-коду за допомогою Barcode API

Для 2D-штрих-кодів, які містять структуровані дані, такі як QR-коди, - поле valueFormat встановлюється на виявлений тип значення, і відповідне поле даних встановлюється. Так, наприклад, якщо виявлено тип URL-адреси, постійна URL-адреса буде завантажена у valueFormat, а Barcode.UrlBookmark буде містити значення URL-адреси. Окрім URL-адреси, існує безліч різних типів даних, які QR-код може підтримувати.

Використовуючи API Mobile Vision, можна читати штрих-коди в будь-якій орієнтації - вони не завжди повинні бути прямолінійними та орієнтовані вгору, а також поєднувати декілька бібліотек для розпізнавання даних декількох типів одночасно (рис. 1.12).

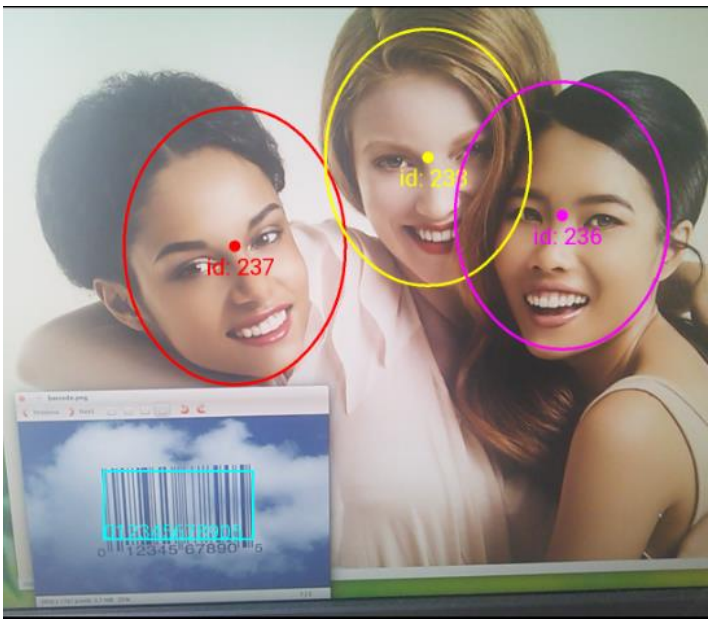


Рисунок 1.12 - Відстеження обличчя та штрих-коду одночасно

Важливо відзначити, що розшифрування всіх штрих-кодів виконується локально, тому не потрібно робити сервер, щоб прочитати дані з коду. У деяких випадках, наприклад, PDF-417, який може вмістити до 1кб тексту, навіть не потрібно взагалі спілкуватися з сервером, щоб отримати всю необхідну інформацію.

1.7 Постановка мети і завдань магістерської роботи

В магістерській роботі необхідно виконати наступне:

1) Дослідити методи розшифрування штрих-кодів та QR-кодів, а також їх реалізації за допомогою спеціалізованих бібліотек

2) Проаналізувати подібні продукти на ринку програмного забезпечення, виявити їх головні недоліки, на які потрібно звернути увагу для створення власного продукту.

3) Виконати розробку органайзера дисконтних карт для ОС Android, що забезпечує можливість зберігати дані, відскановані з пластикових карт, квитків або інших носіїв інформації у вигляді штрих-коду.

4) Розроблений додаток має систематизувати дані в одному місці для зручності користувача. Дані будуть заноситися в програму за допомогою камери пристрою.

5) Інтерфейс додатку має бути зручним для користувача. Додаток повинен складатися з трьох вікон (фрагментів). Перше вікно містить інформацію про користувача. Два інших екрани містять головний функціонал зі збереження даних, що були внесені користувачем.

1.8 Висновки до розділу 1

Перший розділ має оглядовий характер. Було розглянуто актуальність розробки додатку з даним функціоналом. Розглянуті основні положення, щодо розшифрування штрих-кодів та QR-кодів, їх реалізації за допомогою спеціалізованих бібліотек. Розглянуто сильні та слабкі сторони бібліотек, що будуть використовуватися при створенні додатку. Було досліджено головні реалізації подібних продуктів на ринку, їх головні недоліки, на які потрібно звернути увагу під час удосконалення власного продукту.

Метою дослідження є підвищення функціональності додатку на основі аналізу вже існуючих продуктів. Використання новітніх та удосконалених наявних технологій під час розробки додатку. Підвищення зручності використання за допомогою удосконалення інтерфейсу додатку.

Об’єктом дослідження постають технології з розшифрування даних записаних у вигляді штрих-коду або QR-коду.

Предметом дослідження є створення додатку з зберігання даних записаних у вигляді штрих-коду або QR-коду.

Результатом розробки має бути Android-додаток, що буде містити в собі дані, які були отримані за допомогою камери пристрою та розшифровані з штрих-коду або QR-коду за допомогою сторонніх бібліотек.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ МЕТОДІВ РЕАЛІЗАЦІЇ ОРГАНАЙЗЕРА ДИСКОНТНИХ КАРТОК

Android-додаток складається з багатьох частин, які поєднуються і складають цільну програму. Деякі компоненти можуть зроблені різними методами. Розглянемо засоби реалізації, що будуть корисні при досліджені та розробці проекту.

2.1 Компонент Bottom navigation

Розглянемо компонент бібліотеки підтримки матеріального дизайну BottomNavigationView. Це нижня панель навігації, що дозволяє перемикатися між екранами додатка в один дотик, вона призначена в основному для смартфонів, оскільки розташування в нижній частині екрана забезпечує зручний і швидкий доступ для користувача.

Цей шаблон можна використовувати, якщо для переходу до пункту призначення від 3 до 5 пунктів верхнього рівня[10].

Цей компонент доступний в Design Support Library з версії 25.0.0 і Google рекомендує його до застосування з дотриманням таких принципів дизайну:

- Рекомендується тільки для смартфонів, на планшетах та інших пристроях з великим екраном краще використовувати бічне меню навігації.
- Ширина областей значків залежить від їх кількості і може варіюватися від 80 до 168 dp. Висота панелі - 56 dp, розмір іконки 24 × 24 dp.
- Панель повинна містити не менше 3-х, але не більше 5-ти кнопок, кнопки повинні поміщатися на екрані і не скролл.
- Перемикання між екранами тільки по тапу на кнопці, свайп не використовується, що дає можливість комбінувати BottomNavigationView з вкладками. Але Google рекомендує підходити до такої комбінації з обережністю, щоб не дезорієнтувати користувача.
- Можна зафарбовувати активний значок в основний колір програми, або використовувати білий або чорний кольори, якщо нижня панель навігації кольорова. Не фарбуйте значки в різні кольори.
- Текстові мітки повинні бути максимально короткими і інформативними, уникайте довгих написів.
- Натискання значка на нижній панелі навігації має привести користувача прямо до пов'язаного із позначкою вікна, або оновити поточне вікно. Значки не повинні

відкривати меню або спливаючі вікна. Кнопка «Назад» в ньому не повинна дозволяти переміщатися між екранами нижній панелі навігації.

– Нижня панель навігації може зникати з екрану при прокручуванні вгору і з'являтися при поверненні.

Панель може зникнути при прокрутці екрану, за допомогою HideBottomViewOnScrollBehavior, коли вона розміщена в межах CoordinatorLayout і один з дітей у межах CoordinatorLayout прокручується. Ця поведінка встановлюється, лише якщо властивість layout_behavior встановлено як HideBottomViewOnScrollBehavior[11].

Вміст панелі можна заповнити, вказавши файл ресурсів меню. Кожен заголовок пункту меню, піктограма та стан активації будуть використовуватися для відображення елементів нижньої панелі навігації. Пункти меню також можна використовувати для програмного вибору того, що призначено в даний момент. Це можна зробити за допомогою MenuItem # setChecked (true)[12].

Для роботи з BottomNavigationView краще всього використовувати фрагменти. Це дозволяє покращити та прискорити працездатність додатку. Це виконується за допомогою зменшення використання пам’яті на зайві Activity, а лише завантажувати окремі частини інтерфейсу.

2.2 Використання фрагментів

Фрагмент (клас Fragment) представляє поведінку або частина призначеного для користувача інтерфейсу в операції (клас Activity). Розробник може об'єднати кілька фрагментів в одну операцію для побудови багатопанельних, призначеного для користувача, інтерфейсу і повторного використання фрагмента в декількох операціях. Фрагмент можна розглядати як модульну частину операції. Така частина має свій життєвий цикл і самостійно обробляє події введення. Крім того, її можна додати або видалити безпосередньо під час виконання операції. Це щось на зразок вкладеної операції, яку можна багаторазово використовувати в різних операціях.

Фрагмент завжди повинен бути вбудований в операцію, і на його життєвий цикл безпосередньо впливає життєвий цикл операції. Наприклад, коли операція припинена, в тому ж стані знаходяться і всі фрагменти всередині неї, а коли операція знищується, знищуються і всі фрагменти. Однак поки операція виконується (це відповідає стану відновлено життєвого циклу), можна маніпулювати кожним фрагментом незалежно, наприклад додавати або видаляти їх. Коли розробник виконує такі транзакції з фрагментами, він може також додати їх у стек переходів тому, яким керує операція. Кожен елемент стека переходів назад в операції є записом виконаної транзакції з фрагментом. Стек переходів назад дозволяє користувачеві

звернути транзакцію з фрагментом (виконати навігацію в зворотному напрямку), натискаючи кнопку Назад[13].

Коли фрагмент доданий як частина макета операції, він знаходиться в об'єкті ViewGroup всередині ієрархії представлень операції і визначає власний макет представлень. Розробник може вставити фрагмент в макет операції двома способами. Для цього слід оголосити фрагмент в файлі макета операції як елемент <fragment> або додати його в існуючий об'єкт ViewGroup в коді програми. Втім, фрагмент не зобов'язаний бути частиною макета операції. Можна використовувати фрагмент без інтерфейсу як невидимого робочого потоку операції.

Фрагменти вперше з'явилися в Android версії 3.0 (API рівня 11), головним чином, для забезпечення більшої динамічності та гнучкості для користувача інтерфейсів на великих екранах, наприклад, у планшетів. Оскільки екрани планшетів набагато більше, ніж у смартфонів, вони надають більше можливостей для об'єднання і перестановки компонентів для користувача інтерфейсу. Фрагменти дозволяють робити це, позбавляючи розробника від необхідності управляти складними змінами в ієрархії уявлень. Розбиваючи макет операції на фрагменти, розробник отримує можливість модифікувати зовнішній вигляд операції в ході виконання і зберігати ці зміни в стеці переходів тому, яким керує операція.

Наприклад, додаток для перегляду новин може використовувати один фрагмент для відображення списку статей зліва, а інший-для відображення статті справа. Обидва фрагмента відображаються за одну операцію поруч один з одним, і кожен має власний набір методів зворотного виклику життєвого циклу і керує власними подіями призначеного для введення користувачем. Таким чином, замість застосування однієї операції для вибору статті, а інший - для читання статей, користувач може вибрати статтю і читати її в рамках однієї операції, як на планшеті (рис. 2.1).

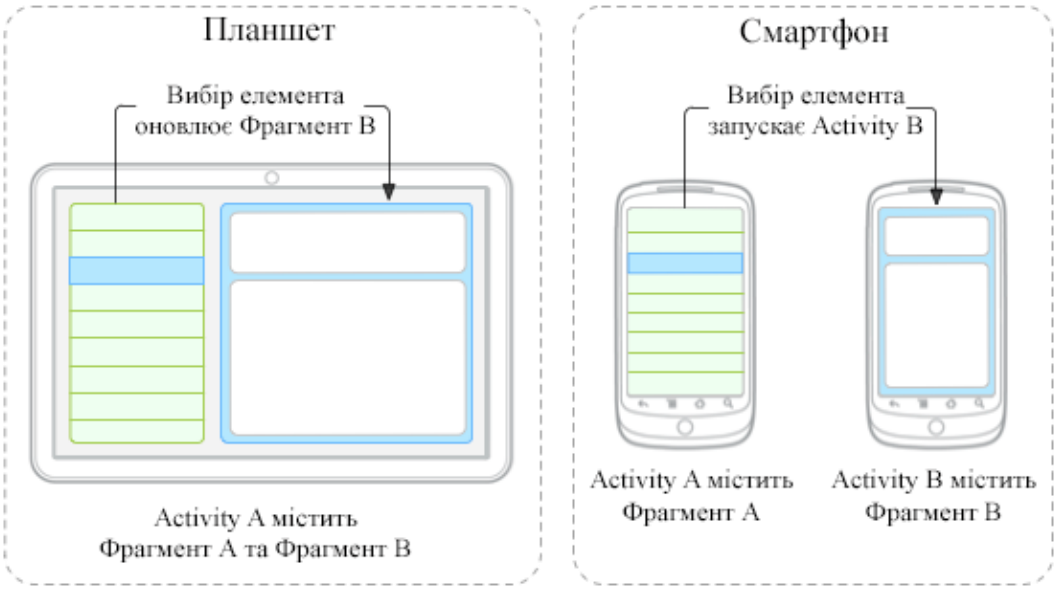


Рисунок 2.1 – Приклад зображення фрагментів на різних пристроях

Слід розробляти кожен фрагмент як модульний і повторно використовуваний компонент операції. Оскільки кожен фрагмент визначає власний макет і власну поведінку зі своїми зворотними викликами життєвого циклу, розробник може включити один фрагмент в кілька операцій. Тому він повинен передбачити повторне використання фрагмента і не допускати, щоб один фрагмент безпосередньо маніпулював іншим. Це особливо важливо, тому що модульність фрагментів дозволяє змінювати їх поєднання у відповідності з різними розмірами екранів. Якщо програма має працювати і на планшетах, і на смартфонах, можна повторно використовувати фрагменти в різних конфігураціях макета, щоб оптимізувати взаємодію з користувачем в залежності від доступного розміру екрана. Наприклад, на смартфоні може виникнути необхідність у поділі фрагментів для надання користувачу інтерфейсу, що призначений як однопанельний, якщо розробнику не вдається помістити більше одного фрагмента в одну операцію[13].

Повернемося до прикладу з додатком для новин. Воно може мати два фрагмента, вбудованих в Операцію А, коли виконується на пристрої планшетного формату. У той же час на екрані смартфона недостатньо місця для обох фрагментів, і тому Операція А включає в себе тільки фрагмент зі списком статей. Коли користувач вибирає статтю, запускається Операція В, що містить другий фрагмент для читання статті. Таким чином, додаток підтримує як планшети, так і смартфони завдяки повторному використанню фрагментів в різних поєднаннях, як показано на (рис. 2.1).

2.2.1 Створення фрагменту

Для створення фрагмента необхідно створити підклас класу Fragment (або його існуючого підкласу). Клас Fragment має код, багато в чому схожий з кодом Activity. Він містить методи зворотного виклику, аналогічні методам операції, такі як onCreate(), onStart(), onPause() і onStop(). На практиці, якщо Ви бажаєте перевести існуючу програму Android так, щоб в ньому використовувалися фрагменти, досить просто перемістити код з методів зворотного виклику операції в відповідні методи зворотного виклику фрагмента.

Як правило, необхідно реалізувати наступні методи життєвого циклу (рис. 2.2):

- **onCreate()** Система викликає цей метод, коли створює фрагмент. У своїй реалізації розробник повинен ініціалізувати ключові компоненти фрагмента, які необхідно зберігати, коли фрагмент знаходиться в стані паузи або відновлений після зупинки.
- **onCreateView()** Система викликає цей метод при першому відображенні призначеного для користувача інтерфейсу фрагмента на дисплеї. Для промальовування призначеного для користувача інтерфейсу фрагмента слід

повернути з цього методу об'єкт View, який є кореневим в макеті фрагмента. Якщо фрагмент не має призначеного для користувача інтерфейсу, можна повернути null.

– **onPause()** Система викликає цей метод як перша вказівка того, що користувач залишає фрагмент (це не завжди означає знищення фрагмента). Зазвичай саме в цей момент необхідно фіксувати всі зміни, які повинні бути збережені за рамками поточного сеансу роботи користувача (оскільки користувач може не повернутися назад).

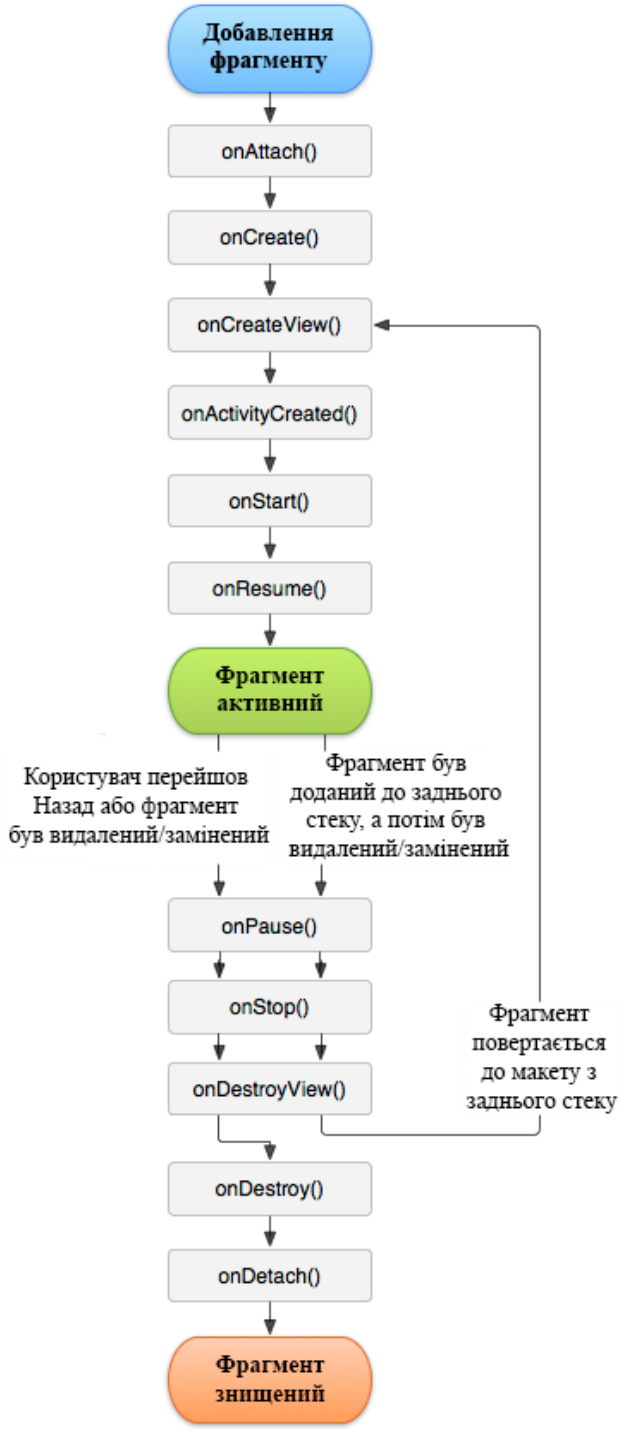


Рисунок 2.2 – Життєвий цикл фрагмента (під час виконання операції)

У більшості додатків для кожного фрагмента повинні бути реалізовані, як мінімум, ці три методи. Однак існують і інші методи зворотного виклику, які слід використовувати для управління різними етапами життєвого циклу фрагмента.

2.2.2 Додавання користувальницького інтерфейсу

Фрагмент зазвичай використовується як частина користувацького інтерфейсу операції, при цьому він додає в операцію свій макет.

Щоб створити макет для фрагмента, розробник повинен реалізувати метод зворотного виклику onCreateView (), який система Android викликає, коли для фрагмента настає час відобразити свій макет. Реалізація цього методу повинна повертати об'єкт View, який є кореневим в макеті фрагмента[14].

Щоб повернути макет з методу onCreateView (), можна виконати його роздування з ресурсу макета, визначеного в XML-файлі. Для цієї мети метод onCreateView () надає об'єкт LayoutInflater.

Параметр container, що передається методу onCreateView (), є батьківським класом ViewGroup (з макета операції), в який буде вставлений макет фрагмента. Параметр savedInstanceState є класом Bundle, який надає дані про попередньому екземплярі фрагмента під час відновлення фрагмента.

Метод inflate () приймає три аргументи:

- Ідентифікатор ресурсу макета, роздування якого слід виконати.
- Об'єкт класу ViewGroup, який повинен стати батьківським для макета після роздування. Передача параметра container необхідна для того, щоб система змогла застосувати параметри макета до кореневого поданням роздутого макета, який визначається батьківським уявленням, до якої направляється макет.
- Логічне значення, яке показує, чи слід прикріпити макет до об'єкта ViewGroup (другий параметр) під час роздування.

2.2.3 Додавання фрагмента в операцію

Як правило, фрагмент додає частину призначеного для користувача інтерфейсу в операцію, і цей інтерфейс вбудовується в загальну ієрархію представлень операції. Розробник може додати фрагмент в макет операції двома способами:

- **оголосивши фрагмент в файлі макета операції**

  В цьому випадку можна вказати властивості макета для фрагмента, як ніби він є поданням.

Атрибут android: name в елементі <fragmen> визначає клас Fragment, екземпляр якого створюється в макеті.

Коли система створює цей макет операції, вона створює екземпляр кожного фрагмента, визначеного в макеті, і для кожного викликає метод onCreateView (), щоб отримати макет кожного фрагмента. Система вставляє об'єкт View, повернутий фрагментом, безпосередньо замість елемента <fragment>.

– **або програмним чином, додавши фрагмент в існуючий об'єкт ViewGroup**

У будь-який момент виконання операції розробник може додати фрагменти в її макет. Для цього достатньо вказати об'єкт ViewGroup, в якому слід розмістити фрагмент.

Для виконання транзакцій з фрагментами всередині операції (таких як додавання, видалення або заміна фрагмента) необхідно використовувати API-інтерфейси з FragmentTransaction. Екземпляр класу FragmentTransaction можна отримати від об'єкта Activity

Після цього можна додати фрагмент методом add (), вказавши додається фрагмент і уявлення, в яке він повинен бути доданий.

Перший аргумент, який передається методу add (), являє собою контейнерний об'єкт ViewGroup для фрагмента, вказаний за допомогою ідентифікатора ресурсу. Другий параметр - це фрагмент, який потрібно додати.

Виконавши зміни за допомогою FragmentTransaction, необхідно викликати метод commit (), щоб вони вступили в силу.

2.2.4 Додавання фрагмента, що не має призначеного для користувача інтерфейсу

Можна використовувати фрагмент і для реалізації фонового поведінки операції без будь-якого додаткового призначеного для користувача інтерфейсу.

Щоб додати фрагмент без користувальницького інтерфейсу, потрібно додати фрагмент з операції, використовуючи метод add (Fragment, String) (передавши йому унікальний строковий «тег» для фрагмента замість ідентифікатора уявлення). Фрагмент буде додано, але, оскільки він не пов'язаний з поданням до макеті операції, він не буде приймати виклик методу onCreateView (). Тому в реалізації цього методу немає необхідності.

Передача строкового тега властива не тільки фрагментами без користувальницького інтерфейсу, тому можна передавати рядкові теги і фрагментами, які мають користувальницький інтерфейс. Однак, якщо у фрагмента немає призначеного для користувача інтерфейсу, то строковий тег є єдиним способом його ідентифікації. Якщо згодом

буде потрібно отримати фрагмент від операції, потрібно буде викликати метод findFragmentByTag ().

2.2.5 Керування фрагментами

Для управління фрагментами в операції потрібен клас FragmentManager. Щоб отримати його, слід викликати метод getFragmentManager () з коду операції.

Нижче вказані дії, які дозволяє виконати FragmentManager:

- отримувати фрагменти, наявні в операції, за допомогою методу findFragmentById () (для фрагментів, що надають користувальницький інтерфейс в макеті операції) або findFragmentByTag () (як для фрагментів, що мають користувальницький інтерфейс, так і для фрагментів без нього);
- знімати фрагменти зі стека переходів назад методом popBackStack () (імітуючи натискання кнопки Назад користувачем);
- реєструвати процес-слухач змін в стеці переходів назад за допомогою методу addOnBackStackChangedListener ().

Додаткові відомості про ці та інші методи наводяться в документації по класу FragmentManager.

Як було описано в попередньому розділі, можна використовувати клас FragmentManager для відкриття FragmentTransaction, що дозволяє виконувати транзакції з фрагментами, наприклад, додавання і видалення[15].

2.2.6 Взаємодія з операцією

Фрагмент може звернутися до екземпляру Activity за допомогою методу getActivity () і без зусиль виконати такі завдання, як пошук уявлення в макеті операції.

Аналогічним чином операція може викликати методи фрагмента, отримавши посилання на об'єкт Fragment від FragmentManager за допомогою методу findFragmentById () або findFragmentByTag () [15].

У деяких випадках необхідно, щоб фрагмент використовував події спільно з операцією. Хороший спосіб реалізації цього полягає в тому, щоб визначити інтерфейс зворотного виклику всередині фрагмента і зажадати від контейнерної операції його реалізації. Коли операція прийме зворотний виклик через цей інтерфейс, вона зможе обмінюватися інформацією з іншими фрагментами в макеті в міру необхідності.

2.3 Контейнер CardView

Програми часто повинні відображати дані в контейнерах з аналогічним стилем. Ці контейнери часто використовуються в списках для зберігання інформації кожного елемента. Система надає API CardView як простий спосіб для відображення інформації всередині картки, які мають послідовний вигляд по всій платформі. Ці карти мають висоту за замовчуванням над їхньою групою перегляду, тому система малює тіні під ними. Карти забезпечують простий спосіб містити групу переглядів, забезпечуючи при цьому відповідний стиль для контейнера.

Віджет CardView є частиною бібліотек v7 Support. Щоб використовувати його у проекті, потрібно додати наступну залежність до файлу build.gradle модуля програми[16]:

```
dependencies {
        implementation 'com.android.support:cardview-v7:28.0.0'
}
```

Для того, щоб використовувати CardView, потрібно додати його до вашого макета. Використовуйте його як групу перегляду, щоб містити інші види. У цьому прикладі CardView містить єдиний TextView для відображення деякої інформації для користувача.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   xmlns:card_view="http://schemas.android.com/apk/res-auto"
   ... >
   <!—CardView, що містить TextView -->
   <android.support.v7.widget.CardView
      xmlns:card_view="http://schemas.android.com/apk/res-auto"
      android:id="@+id/card_view"
      android:layout_gravity="center"
      android:layout_width="200dp"
      android:layout_height="200dp"
      card_view:cardCornerRadius="4dp">

      <TextView
         android:id="@+id/info_text"
         android:layout_width="match_parent"
         android:layout_height="match_parent" />
   </android.support.v7.widget.CardView>
</LinearLayout>
```

Карти звертаються до екрану з висотою за замовчуванням, що призводить до того, що система намалює тінь під ними. Ви можете надати користувальницьку висоту для карти з атрибутом card_view: cardElevation. Це призведе до більш вираженої тіні з більшою висотою, а нижня висота призведе до більш світлої тіні. CardView використовує реальну висоту і динамічні тіні на Android 5.0 (рівень API 21) і вище і повертається до програмної тіньової реалізації на більш ранніх версіях[10].

Потрібно використати ці властивості, щоб налаштувати вигляд віджета CardView:

- Щоб встановити радіус кута у ваших макетах, використовуйте атрибут card_view: cardCornerRadius.
- Щоб встановити радіус кута у вашому коді, використовуйте метод CardView.setRadius.
- Щоб встановити колір фону карти, скористайтеся атрибутом card_view: cardBackgroundColor.

Завдяки дорогому характеру закругленого кутового обрізання, на платформах перед Lollipop, CardView не притискає своїх дітей, які перетинаються з закругленими кутами. Замість цього, він додає відступи, щоб уникнути такого перетину.

CardView використовує властивість elevation на Android 5.0 (рівень API 21) і вище для тіней і повертається до користувальницької емуляції тіньової реалізації на старих платформах.

Завдяки довгому використанню закругленого кутового обрізання, на платформах до Lollipop, CardView не обрізає своїх «дітей», які перетинаються з закругленими кутами. Замість цього, він додає відступи, щоб уникнути такого перетину.

До Android 5.0 (рівень API 21), CardView додавав відступ до його вмісту і малював тіні до цієї області. Ця сума відступів дорівнює maxCardElevation + (1 - cos45) * cornerRadius з боків і maxCardElevation * 1.5 + (1 - cos45) * cornerRadius зверху і знизу [17].

Оскільки для заповнення вмісту для тіней використовується відступ, немає можливості встановити відступ у CardView. Замість цього, можна використовувати атрибути відступу вмісту в XML або setContentPadding (int, int, int, int) в коді, щоб встановити відступ між краями CardView і дітьми CardView.

Треба звернути увагу на те, що, якщо буде вказано точні розміри для CardView, через тіні, його область вмісту буде відрізнятися між платформами до Lollipop (рівень API 21) і після Lollipop. Використовуючи специфічні значення ресурсів API-версії, є можливість уникнути цих змін. Крім того, якщо ви хочете, щоб CardView додавав внутрішній відступ на платформах API 21 і вище, є можливість викликати setUseCompatPadding (boolean) і встановити значення true.

Щоб змінити висоту CardView у способі зі зворотною сумісністю, слід використовувати setCardElevation (float). CardView буде використовувати API висоти на Lollipop (або вище) і до API 21, він змінить розмір тіні. Щоб уникнути переміщення View, тоді як розмір тіні змінюється, розмір тіні закріпляється getMaxCardElevation (). Якщо є потреба в динамічній зміні висоти, потрібно викликати setMaxCardElevation (float), коли CardView ініціалізується[17].

2.4 Компонент RecyclerView

Якщо програмі потрібно відобразити список елементів, що прокручується, на основі великих наборів даних (або даних, які часто змінюються), слід використовувати RecyclerView. Та якщо потрібно створити список з картами, слід використовувати віджет CardView, як описано у розділі вище.

Віджет RecyclerView - це більш просунута і гнучка версія ListView. Гнучкий засіб відображення для надання обмеженого вікна у великому наборі даних.

У моделі RecyclerView кілька різних компонентів працюють разом для відображення даних. Загальний контейнер для користувальницького інтерфейсу є об'єктом RecyclerView, який додається до макета. RecyclerView заповнює себе даними що надані менеджером компонування. Можна скористатися одним із стандартних менеджерів макетів (наприклад, LinearLayoutManager або GridLayoutManager) або реалізувати власні.

View у списку представлені об'єктами view holder. Ці об'єкти є екземплярами класу, які визначаються, розширюючи RecyclerView.ViewHolder. Кожен view holder відповідає за відображення одного елемента з view. Наприклад, якщо у вашому списку відображається музична колекція, кожен власник перегляду може представляти один альбом. RecyclerView створює лише стільки view holder, скільки потрібно для відображення на екрані частини динамічного вмісту, плюс кілька додаткових. Коли користувач прокручує список, RecyclerView бере на себе view, що на екранні, і повторно посилається на дані, які прокручуються на екрані.

Об'єктами view holder керує адаптер, який створюється за допомогою розширення RecyclerView.Adapter. Підклас RecyclerView.Adapter відповідає за надання view, які представляють елементи в наборі даних. За потреби адаптер створює view holder. Адаптер також прив'язує view holder до своїх даних. Він робить це, присвоюючи позицію view holder, і викликає метод onBindViewHolder () адаптера. Цей метод використовує позицію view holder, щоб визначити, який вміст має бути, на основі позиції списку.

Ця модель RecyclerView робить багато роботи з оптимізації, тому не потрібно:

– Коли список вперше заповнено, він створює і пов'язує деякі view holder з обох сторін списку. Наприклад, якщо view відображає позиції списків від 0 до 9, RecyclerView створює і зв'язує цих view holder, а також може створити і прив'язати "власника перегляду" до позиції 10. Таким чином, якщо користувач прокручує список, наступний елемент готовий до відображення [18].
– Коли користувач прокручує список, RecyclerView створює нові view holder, якщо це необхідно. Він також зберігає view holder, які прокручуються поза екраном, щоб їх можна було повторно використовувати. Якщо користувач змінює напрямок, у якому вони прокручувалися, то view holder, які прокручувалися з екрану, можна повернути назад. З іншого боку, якщо користувач продовжує прокручуватися в тому ж напрямку, view holder, які були найдовше екрановані, можуть бути прив'язані до нових даних. View holder не потрібно створювати або роздувати його перегляд; замість цього, програма просто оновлює вміст перегляду, щоб він відповідав новому елементу, до якого він був пов’язаний [18].
– Коли відображені елементи змінюються, ви можете сповістити адаптер, викликавши відповідний метод RecyclerView.Adapter.notify…(). Вбудований код адаптера потім повторно пов'язує елементи, що були застосовані [18].

Розглянемо про те, як виділяти елементи в RecyclerView.

У ListView для виділення елементів використовувався метод setChoiceMode (int), RecyclerView ж уявлення не має, що елементи можуть виділятися, тому ми повинні навчити цьому наш адаптер.

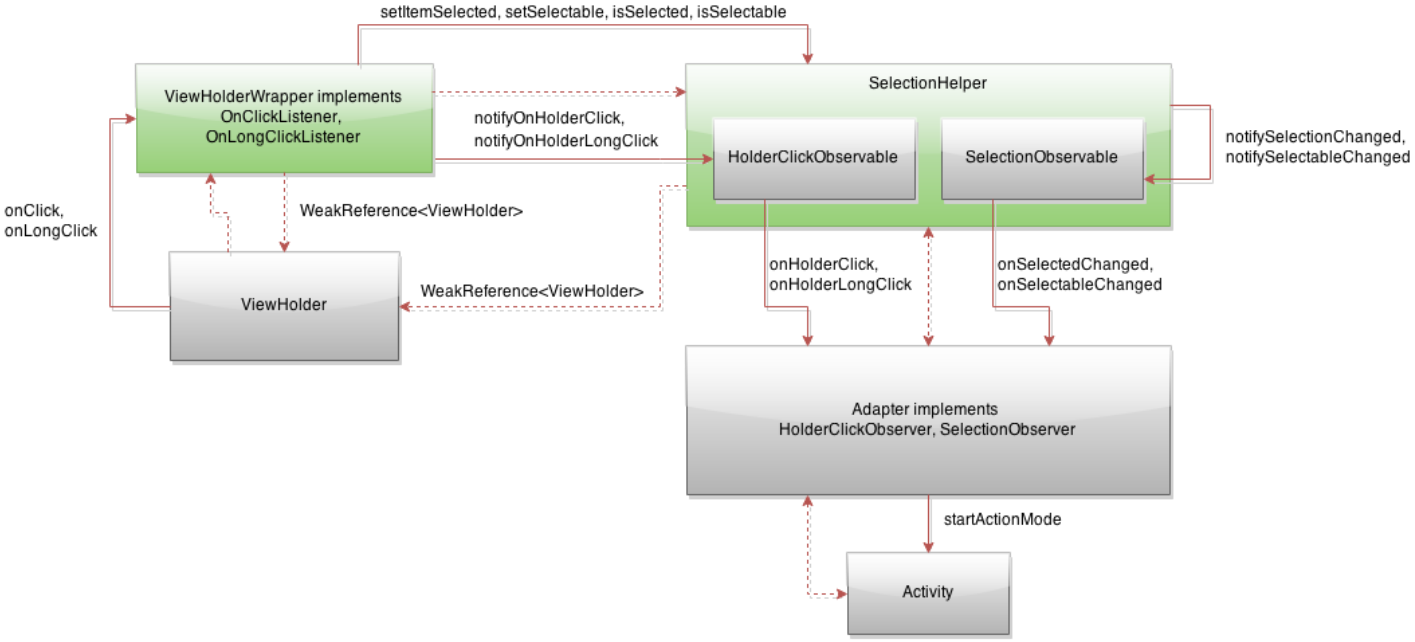


Рисунок 2.3 – Зв'язки між об'єктами в RecyclerView

На (рис. 2.3) схематично позначені зв'язку між об'єктами. Пунктирні стрілки - посилання, інші - виклики методів. Зеленим позначені об'єкти, які безпосередньо реалізують логіку виділення.

Принцип роботи виходить наступний:

1. ViewHolderWrapper встановлює себе в якості ClickListener для кореневої view для ViewHolder і починає отримувати події onClick і onLongClick. Залежно від реалізації він може просто перенаправляти ці події в HolderClickObservable (ViewHolderClickWrapper), або, виходячи з поточного статусу SelectionHelper'a виділяти елемент викликом setItemSelected (ViewHolder, boolean) (ViewHolderMultiSelectionWrapper)[19].
2. SelectionHelper зберігає інформацію про виділені елементах і оповіщає слухачів (SelectionObserver) про зміну виділення [19].
3. Слухач (в нашому випадку адаптер) відповідає за візуальне відображення виділення елемента, а також взаємодії з ним (на (рис. 2.3.1) виклик startActionMode у Activity)[19].

У самому адаптері необхідно зробити наступні зміни:

1. Створити SelectionHelper і зареєструвати слухачів (в даному випадку сам адаптер, але це може бути і Activity, наприклад)

```
mSelectionHelper = new SelectionHelper(mHolderTracker);
mSelectionHelper.registerSelectionObserver(this);
```

2. Обернути створювані ViewHolder у ViewHolderWrapper потрібного типу [20].

```
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int position)
{
  LayoutInflater inflater = LayoutInflater.from(viewGroup.getContext());
  ImageViewHolder holder = new ImageViewHolder(
       inflater.inflate(R.layout.gallery_item, viewGroup, false));
  return mSelectionHelper.wrapSelectable(holder);
}
```

Метод wrapSelectable(ViewHolder) у SelectionHelper:

```
public <H extends RecyclerView.ViewHolder> H wrapSelectable(H holder)
{
  new ViewHolderMultiSelectionWrapper(holder);
```

```
    return holder;
}
```

Приєднуємо ViewHolder до SelectionHelper в методі onBindViewHolder (ViewHolder, int) нашого адаптера[20]:

```
@Override
public void onBindViewHolder(RecyclerView.ViewHolder viewHolder, int position)
{
    Image image = mDataSet.getItem(position);
    ImageViewHolder imageViewHolder = (ImageViewHolder) viewHolder;
    imageViewHolder.bindInfo(image);

    Checkable view = (Checkable) viewHolder.itemView;
    view.setChecked(mSelectionHelper.isItemSelected(position));
    mHolderTracker.bindHolder(imageViewHolder, position);
}
```

Це потрібно через те, що поки немає іншого способу отримати від RecyclerView список використовуваних зараз ViewHolder. Якщо не вести їх облік, при необхідності оновити відтворення зображення виділення у всіх обраних елементів (користувач закрив ActionMode, наприклад), SelectionHelper просто не зможе цього зробити. View залишаться виглядати виділеними, коли за фактом такими не будуть.

Особливість RecyclerView - він використовує заново вже створені ViewHolder. Виглядає це наступним чином[21]:

1. Відкриваємо додаток. На екрані 10 елементів - 10 ViewHolder створено для них.
2. Запускаємо ActionMode, починаємо виділяти елементи - 1,2,3.
3. Прокручуємо view вниз, бачимо елементи з 10 по 20. Помилково вважати, що в пам’яті залишається 20 ViewHolder. Для частини даних RecyclerView створить нові ViewHolder, а для іншої заново використовує вже наявні. Причому невідомо в якому порядку.
4. Тепер якщо прокрутити view назад вгору, частина з 10 ViewHolder буде знищена, замість них будуть створені нові. Частина, що залишилася буде використана заново і зовсім не обов'язково для тих же позицій.
5. Скасовуємо ActionMode. SelectionHelper повинен розкидати слухачам повідомлення про зміну виділення на елементах із зазначенням ViewHolder для кожного елемента, але він вже не володіє актуальними даними, всі Holder помінялися.

Потрібно звернути увагу що не можна зберігати строгі посилання (strong reference) на ViewHolder. Вони можуть бути видалені з RecyclerView випадково. В цьому випадку, якщо будуть зберігатися строгі посилання на них, трапиться витік пам'яті. Тому для зберігання посилань в ViewHolderWrapper і WeakHolderTracker використовуються тільки WeakReference[22].

```
private abstract class ViewHolderWrapper implements android.view.View.OnClickListener
{
   protected final WeakReference<RecyclerView.ViewHolder> mWrappedHolderRef;
   protected ViewHolderWrapper(RecyclerView.ViewHolder holder)
   {
      mWrappedHolderRef = new WeakReference<>(holder);
   }
}
```

RecyclerView, по суті, є еволюцією одного з найнеобхідніших в Android-розробці віджетів - ListView. Власне, призначення у нього рівно те ж саме - відображати список елементів, але є нюанси.

2.5 Використання бази даних SQLite

Збереження даних у базі даних ідеально підходить для повторних або структурованих даних, таких як контактна інформація. API, які потрібно використовувати для бази даних на Android, доступні в пакеті android.database.sqlite.

Одним з основних принципів баз даних SQL є схема: офіційна декларація про те, як організована база даних. Схема відображена в операторах SQL, які використовуються для створення бази даних. Було б корисно створити клас супутника, відомий як контрактний клас, який явно вказує розташування вашої схеми систематичним і самодокументарним способом.

Клас контракту - це контейнер для констант, які визначають імена для URI, таблиць і стовпців. Клас контракту дозволяє використовувати однакові константи для всіх інших класів в одному пакеті. Це дозволяє змінювати ім'я стовпця в одному місці і поширювати його по всьому коду.

Гарний спосіб організувати контрактний клас - це поміщення глобальних визначень у базу даних на кореневому рівні класу. Потім створити внутрішній клас для кожної таблиці. Кожен внутрішній клас перераховує стовпці відповідної таблиці.

2.5.1 Створення бази даних з використанням помічника SQL

Після того, як визначено, як виглядає база даних, слід реалізувати методи, які створюють і підтримують базу даних і таблиці. Ось деякі типові заяви, які створюють і видаляють таблицю:

```
private static final String SQL_CREATE_ENTRIES =
   "СТВОРИТИ ТАБЛИЦЮ " + FeedEntry.TABLE_NAME + " (" +
   FeedEntry._ID + " ЦІЛОЧИСЕЛЬНИЙ ПЕРВИННИЙ КЛЮЧ," +
   FeedEntry.COLUMN_NAME_TITLE + " ТЕКСТ," +
   FeedEntry.COLUMN_NAME_SUBTITLE + " ТЕКСТ)";
private static final String SQL_DELETE_ENTRIES =
   "ВИДАЛИТИ ТАБЛИЦЮ ЯКЩО ІСНУЄ " + FeedEntry.TABLE_NAME;
```

Подібно до файлів, які зберігаються на внутрішній пам’яті пристрою, Android зберігає базу даних у приватній папці програми. Дані захищені, оскільки за умовчанням ця область не доступна для інших програм або користувача.

Клас SQLiteOpenHelper містить корисний набір API для управління базою даних. Коли цей клас використовується для отримання посилань на базу даних, система виконує потенційно тривалі операції створення та оновлення бази даних тільки в разі потреби, а не під час запуску програми. Все, що потрібно зробити - це викликати getWritableDatabase () або getReadableDatabase ()[23].

Щоб скористатися SQLiteOpenHelper, треба створити підклас, який перекриває методи зворотного виклику onCreate () і onUpgrade (). Також є можливість реалізувати методи onDowngrade () або onOpen (), але вони не потрібні.

Наприклад, ось реалізація SQLiteOpenHelper, яка використовує деякі з наведених вище команд:

```
public class FeedReaderDbHelper extends SQLiteOpenHelper {
   // Якщо змінити схему бази даних, необхідно збільшити версію бази даних
   public static final int DATABASE_VERSION = 1;
   public static final String DATABASE_NAME = "FeedReader.db";

   public FeedReaderDbHelper(Context context) {
      super(context, DATABASE_NAME, null, DATABASE_VERSION);
   }
   public void onCreate(SQLiteDatabase db) {
      db.execSQL(SQL_CREATE_ENTRIES);
   }
```

```
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Ця база даних - це лише кеш для онлайнових даних, тому політика оновлення
       // полягає в тому, щоб просто відкинути дані і почати все спочатку
        db.execSQL(SQL_DELETE_ENTRIES);
        onCreate(db);
    }
    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }
}
```

Щоб отримати доступ до бази даних, потрібно створити підклас SQLiteOpenHelper:

```
FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getContext());
```

2.5.3 Читання та видалення інформації з бази даних

Щоб прочитати з бази даних, потрібно скористуватися методом query (), передавши йому критерії вибору та потрібні стовпці. Метод об'єднує елементи insert () і update (), за винятком того, що список стовпців визначає дані, які потрібно отримати ("проекцію"), а не дані для вставки[24]. Результати запиту повертаються вам в об'єкті Cursor.

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// Визначте проекцію, яка вказує, які стовпці з бази даних
// фактично будуть використовуватися після цього запиту.
String[] projection = {
    BaseColumns._ID,
    FeedEntry.COLUMN_NAME_TITLE,
    FeedEntry.COLUMN_NAME_SUBTITLE
    };

// Фільтрувати результати WHERE "назва" = "Мій заголовок"
String selection = FeedEntry.COLUMN_NAME_TITLE + " = ?";
String[] selectionArgs = { "Мій заголовок" };

// Як потрібно, щоб результати сортувалися в результуючому курсорі
String sortOrder =
 FeedEntry.COLUMN_NAME_SUBTITLE + " ОПИС";
```

```
Cursor cursor = db.query(
   FeedEntry.TABLE_NAME,   // Таблиця для запиту
   projection,  // Масив стовпців для повернення (передати null, щоб отримати всі)
   selection,              // Стовпці для пропозиції WHERE
   selectionArgs,          // Значення для пропозиції WHERE
   null,                   // Не групувати рядки
   null,                   // Не фільтрувати за групами рядків
   sortOrder               // Порядок сортування
   );
```

Третій і четвертий аргументи (selection і selectionArgs) об'єднуються для створення умови WHERE. Оскільки аргументи надаються окремо від запиту вибору, вони перекриваються перед комбінуванням. Це робить вибірки імунітетом до ін'єкції SQL[24].

Щоб подивитися на рядок у курсорі, потрібно скористуватися одним з методів переміщення курсору, який потрібно завжди викликати перед початком читання значень. Оскільки курсор починається з положення -1, виклик moveToNext () розміщує "позицію для читання" на першому записі результатів і повертає, якщо курсор вже минув останній запис у наборі результатів. Для кожного рядка можна прочитати значення стовпця, викликавши один з методів отримання курсора, наприклад getString () або getLong (). Для кожного з методів get необхідно передати позицію індексу потрібного стовпця, яку можна отримати, викликавши getColumnIndex () або getColumnIndexOrThrow (). Після завершення перебору результатів треба викликати close () на курсорі, щоб звільнити його ресурси. Наприклад, нижче показано, як отримати всі ідентифікатори елементів, збережені в курсорі, і додати їх до списку:

```
List itemIds = new ArrayList<>();
while(cursor.moveToNext()) {
  long itemId = cursor.getLong(
     cursor.getColumnIndexOrThrow(FeedEntry._ID));
  itemIds.add(itemId);
}
cursor.close();
```

Щоб видалити рядки з таблиці, необхідно надати критерії вибору, які ідентифікують рядки методу delete (). Механізм працює так само, як і аргументи вибору методу query (). Він розділяє специфікацію вибору на пропозицію вибору та аргументи вибору. Розділ визначає стовпці для перегляду, а також дозволяє комбінувати тести стовпців. Аргументи є значеннями для перевірки, які пов'язані з цим пунктом. Оскільки результат не обробляється так само, як звичайний оператор SQL, він захищений від ін'єкцій SQL.

```
// Визначити частину запиту "де".
String selection = FeedEntry.COLUMN_NAME_TITLE + " LIKE ?";
// Вказуємо аргументи в порядку заповнювача.
String[] selectionArgs = { "MyTitle" };
// Виписуємо оператор SQL.
int deletedRows = db.delete(FeedEntry.TABLE_NAME, selection, selectionArgs);
```

Повернене значення для методу delete () вказує кількість рядків, видалених з бази даних.

2.5.4 Оновлення бази даних

Якщо потрібно змінити підмножину значень бази даних, потрібно використовувати метод update (). Оновлення таблиці поєднує синтаксис ContentValues з insert () з синтаксисом WHERE delete ().

```
SQLiteDatabase db = mDbHelper.getWritableDatabase();
// Нове значення для одного стовпця
String title = "МояНоваНазва";
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);

// Який рядок оновлюється на основі назви
String selection = FeedEntry.COLUMN_NAME_TITLE + " LIKE ?";
String[] selectionArgs = { "МояСтараНазва" };

int count = db.update(
    FeedReaderDbHelper.FeedEntry.TABLE_NAME,
    values,
    selection,
    selectionArgs);
```

Повернене значення методу update () - це кількість рядків, що враховуються в базі даних.

2.6 Платформа Vision API

Програмне сканування QR-коду або штрих-коду на Android завжди було предметом великої дискусії. Існує багато способів, за допомогою яких це можна зробити:

- Використання веб-інтерфейсу API, де зображення завантажується на сервер і повертає результати.
- Використовуйте гібридне рішення, де веб-додаток отримує доступ до камери та сканує штрих-код або QR-код.
- Написати свій власний алгоритм сканування зображень.
- Скористайтеся послугами Google Play, інтегрованими в мобільне бачення.

На мою думку, використання Android Mobile Vision для сканування QR-кодів є найбільш сприятливим варіантом порівняно з іншими варіантами, оскільки пропонує автономне сканування зображень, що призводить до швидкого сканування та негайного результату. Крім того; при використанні цієї бібліотеки не потрібно імпортувати будь-яку іншу бібліотеку для сканування, оскільки цей API також бере до уваги розпізнавання та сканування обличчя, а також тексту.

Mobile Vision API надає API для виявлення штрих-кодів, які швидко, легко і локально зчитують і декодують безліч різних типів штрих-кодів.

Є багато сторонніх бібліотек, доступних для сканування штрих-кодів та QR-кодів, деякі з них навіть схвалені Google. Але в наш час читання QR-кодів і штрих-кодів на Android стало настільки поширеним, що вимагає офіційного рішення. Хоча більшість бібліотек з відкритим кодом сканують лише один QR-код або штрих-код одночасно. Це Mobile Vision API може одночасно сканувати кілька кодів QR. Крім того, якщо ви хочете, ви можете безпосередньо інтегрувати їх у камеру андроїда і сканувати штрих-коди і QR-коди на льоту під час потокового відтворення на поверхні.

Класи для виявлення та аналізу штрих-кодів доступні в просторі імен com.google.android.gms.vision.barcode. Клас BarcodeDetector є основним об'єктом фреймворку, що обробляє роботу, для повернення типів SparseArray <Barcode>[26].

Тип штрих-коду являє собою один визнаний штрих-код та його значення. У випадку 1D штрих-коду, такого як UPC-коди, це буде просто число, яке закодовано штрих-кодом. Це доступно у властивості rawValue, при цьому виявлений тип кодування встановлюється у полі формату.

Для 2D штрих-кодів, які містять структуровані дані, такі як QR-коди - поле valueFormat встановлюється у тип виявленого значення, і встановлюється відповідне поле даних. Так, наприклад, якщо виявлено тип URL, константа URL буде завантажена в valueFormat, а Barcode.UrlBookmark буде містити значення URL. Крім URL-адрес, існує багато різних типів даних, які може підтримувати QR-код[27].

При використанні API Mobile Vision можна читати штрих-коди в будь-якій орієнтації - вони не завжди повинні бути під прямим кутом і орієнтованими вгору.

Важливо відзначити, що розбір штрих-кодів здійснюється локально, тому немає потреби в роботі у зворотний бік на сервері, щоб читати дані з коду. У деяких випадках, наприклад, PDF-417, який може містити до 1KB тексту, не потрібно взагалі зв'язуватися з сервером, щоб отримати всю потрібну інформацію.

Налаштування Mobile Vision API дуже просте, всі його завантаження управляються внутрішньо службами Google Play. Хоча для його правильної інтеграції та її функціонування до фактичного використання, рекомендується посилатися на використовувані функції Mobile Vision API у маніфесті додатка. Таким чином, завантаження залежностей відбувається до того, як буде змога їх використовувати, тобто під час встановлення.

Можливо, що вперше наш детектор штрих-коду запуститься, служби Google Play ще не будуть готові обробляти штрих-коди. Тому ми повинні перевірити, чи працює наш детектор, перш ніж ми його використаємо. Якщо це не так, можливо, нам доведеться зачекати на завершення завантаження або дозволити користувачам знати, що їм потрібно знайти підключення до Інтернету або очистити місце на своєму пристрої.

Також треба не забувати дотримуватися інструкцій щодо інтеграції служб Google Play[28]. В нашому випадку нам потрібно лише скомпілювати com.google.android.gms: play-services-vision, тому переконайтеся, що build.gradle (app) має наступні залежності:

```
dependencies {
        compile fileTree(dir: 'libs', include: ['*.jar'])
        testCompile 'junit:junit:4.12'
        compile 'com.android.support:appcompat-v7:26.1.0'
        compile 'com.google.android.gms:play-services-vision:11.4.0'}
```

А build.gradle (global) має мати такі:

```
allprojects {
   repositories {
      jcenter()
      maven {
         url "https://maven.google.com"
      }
   }
}
```

Треба звернути увагу, що API здатний виявляти кілька штрихових кодів в одному кадрі. В додатку у вигляді попереднього перегляду камери, де є можливість дивитися на кілька штрих-кодів SparseArray <Barcode> буде заповнено кількома записами.

```
Frame frame = new Frame.Builder().setBitmap(myBitmap).build();
SparseArray<Barcode> barcodes = detector.detect(frame);
```

2.7 Висновок до розділу 2

В ході дослідження засобів, що будуть використовуватися в розробці додатку для роботи в операційній системі Android, було виявлено оптимальний інструментарій за допомогою якого буде створено кінцевий продукт.

Розглянуті засоби допоможуть створити початковий каркас додатку, а також наповнити його функціоналом, заявленим під час формування технічного завдання проекту. Ці засоби впроваджуються в продукти від компанії Google, та зарекомендували себе як надійні елементі програми, які сприяють до пришвидшення працездатності за рахунок використання меншої кількості ресурсів девайсу.

Також розглянуті засоби допоможуть створити інтерфейс, що буде зрозумілий та інтуїтивний користувачу з самого початку користуванням додатком.

Використання новітньої бібліотеку з розпізнавання елементів за допомогою камери дозволять пришвидшити роботу відносно інших додатків, що переважно працюють на старих та громіздких бібліотеках.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ANDROID-ОРГАНАЙЗЕРА ДЛЯ РОБОТИ З ДИСКОНТНИМИ КАРТКАМИ

Розробка додатку починається зі створення основної структури програми, до якої буде доповнюватися фунціонал. Додаток повинен складатися з трьох вікон. Перша «Головна» містить інформацію про користувача. Два інших екрани містять головний функціонал зі збереження даних, що були внесені користувачем.

3.1 Використання Bottom Navigation View з фрагментами

Почнемо з додавання відповідних залежностей бібліотеки підтримки.

```
dependencies {
   implementation 'com.android.support:design:28.0.0'
//Інші залежності додамо пізніше, під час розробки
}
```

Стандартна бібліотека 'com.android.support:design:28.0.0' дозволяє додавати до додатку елементи, що запропоновані компанією Google для уніфікації дизайну серед додатків на android.

Тепер додаємо нижню панель до layout.xml.

```
<android.support.design.widget.BottomNavigationView
      android:id="@+id/navigation"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:background="?android:attr/windowBackground"
      android:layout_alignParentBottom="true"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:menu="@menu/navigation" />
```

Наступним кроком зробимо кнопки у віджеті BottomNavigationView та кастомізуємо їх. Створимо файл navigation.xml за шляхом res/layout/menu/

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

   <item
      android:id="@+id/navigation_home"
      android:icon="@drawable/ic_home"
      android:title="@string/title_home" />

   <item
      android:id="@+id/navigation_cards"
      android:icon="@drawable/ic_cards"
      android:title="@string/title_cards" />

   <item
      android:id="@+id/navigation_tickets"
      android:icon="@drawable/ic_tickets"
      android:title="@string/title_tickets" />

</menu>
```

Слід зазначити, що рекомендується мати щонайменше 3, та не більше 5 елементів у нижній навігації. Було створено 3 кнопки для трьох вікон додатка. У атрибуті android:title використовуємо значення з main\res\values\strings.xml для можливості використовувати декілька мов у додатку.

```
<resources>
<string name="app_name" translatable="false">Queorder</string>
   <string name="title_home">Home</string>
   <string name="title_cards">Cards</string>
   <string name="title_tickets">Tickets</string>
<string name="main_info">App was created by Chmykhalo Roman. \n
                                Student of CI-17dm group</string>
</resources>
```

Змінюємо іконки на кастомні і шо більше пасують до вкладок із зазначеним функціоналом (рис 3.1). Стандартні іконки, що запропоновані від розробників як системні можна знайти у посібнику з Material Design[29].

Іконки зберігаються у векторному вигляді в xml файлі:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <path
        android:fillColor="#FF000000"
        android:pathData="M12 5.69l5 4.5V18h-2v-6H9v6H7v-7.81l5-4.5M12 3L2 12h3v8h6v-6h2v6h6v-8h3L12 3z" />
</vector>
```

Рисунок 3.1. – Іконки BottomNavigationView

Перейдемо на сторінку Activity.java. Ініціалізуємо свій інтерфейс користувача[30]:

```
import android.support.design.widget.BottomNavigationView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        BottomNavigationView navigation = (BottomNavigationView) findViewById(R.id.navigation);
        navigation.setOnNavigationItemSelectedListener(navListener);
    }
}
```

Ми можемо слухати та реагувати на окремі клацання пунктів навігації. Ми робимо це за допомогою BottomNavigationView.setOnNavigationItemSelectedListener ().

```
private BottomNavigationView.OnNavigationItemSelectedListener navListener
        = new BottomNavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        Fragment selectedFragment = null;

        switch (item.getItemId()){
            case R.id.navigation_home:
                //selectedFragment = new HomeFragment();
                break;
            case R.id.navigation_cards:
                //selectedFragment = new CardFragment();
                break;
            case R.id.navigation_tickets:
                //selectedFragment = new TicketsFragment();
                break;
        }
```

Тепер у нас є базове налаштування Bottom Navigation для нашої програми (рис 3.2).

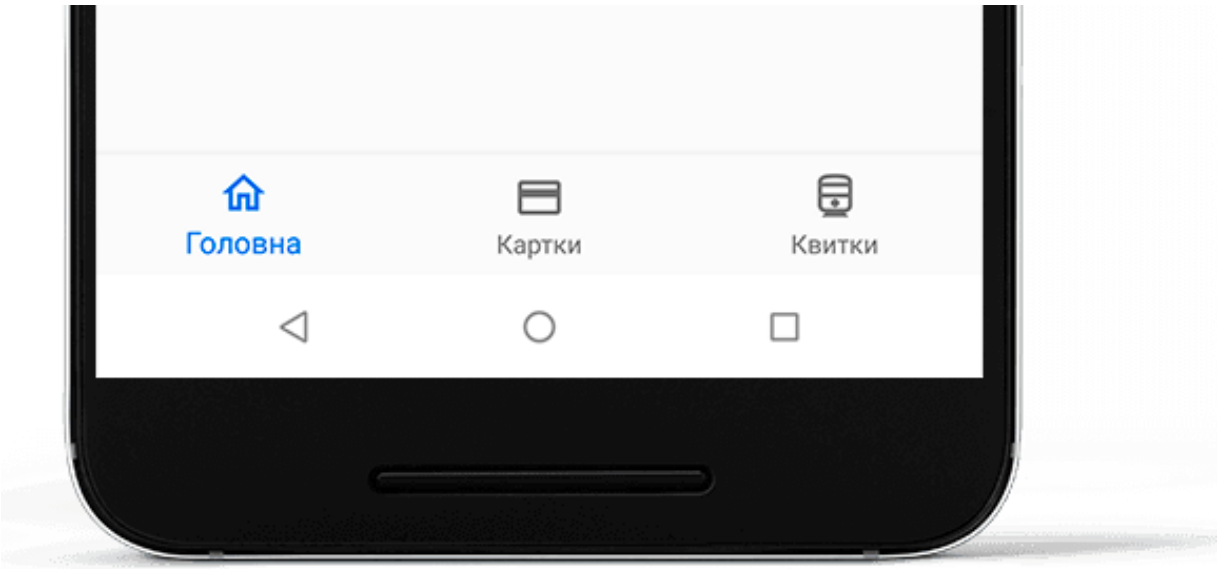


Рисунок 3.2 – BottomNavigationBar в додатку

Bottom Navigation, тепер реєструє кліки. Перший пункт навігації "Головна" вибрано за замовчуванням.

Цей елемент можна стилізувати по різному. Змінювати колір іконок як активної так і неактивних. Змінювати колір заднього фону. Зміни стилю оформлення змінюються у файлі styles.xml у папці res/values/

```
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <!—Змінювати оформлення тут -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
    <item name="android:windowLightStatusBar">true</item>
    <item name="android:navigationBarColor">@color/colorPrimaryDark</item>
    <item name="android:windowLightNavigationBar">true</item>
</style>
```

Атрибути android:windowLightStatusBar та android:windowLightNavigationBar допомагають стилізувати вікно додатку за допомогою зміни зовнішнього вигляду статус бару, а також бару з кнопками навігації[30].

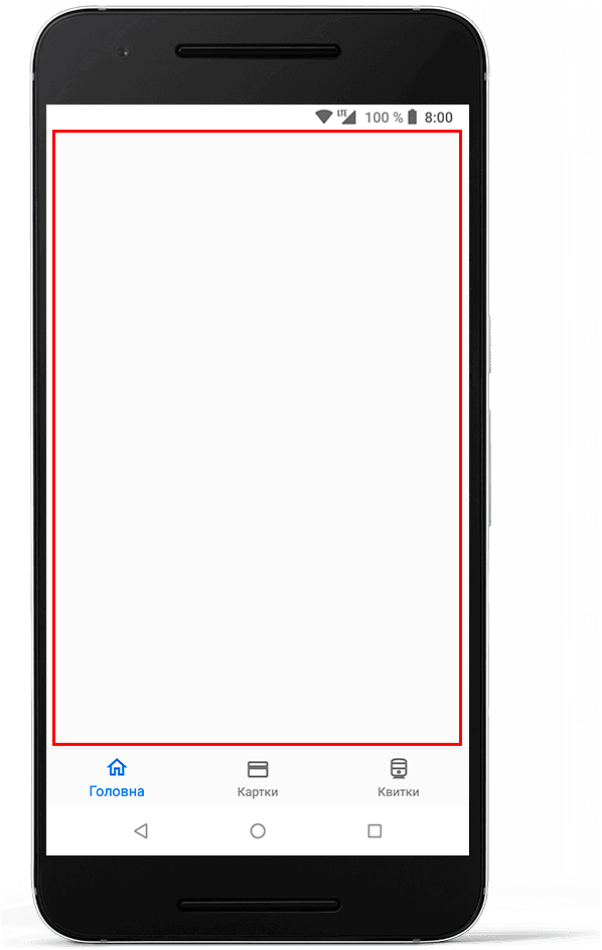


Рисунок 3.3 – Область відображення верхнього шару (виділена червоним).

BottomNavigationView здійснює навігацію між переглядами верхнього рівня. Верхнього рівень тут буде видимою областю над кнопками навігації. Ця видима область повинна відображати поточний активний елемент нижньої навігації. Отже, якщо елемент «Головна» в даний час активний (виділений або вибраний) у навігації внизу, то видима область має відображати головний екран(рис. 3.3).

Для цього буде використаний фрагмент. Слід додати до BottomNavigationView декілька фрагментів.

Для кожної кнопки, при натисканні, буде з’являтися свій власний фрагмент з відповідною інформацією для кожного розділу.

Потрібно створити клас HomeFragment.java, CardFragment.java TicketsFragment.java з відповідними назвами до розділів. Розглянемо код одного з них на прикладі CardFragment.java:

```
package com.example.roman.queorder.mFragments;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.roman.queorder.R;

public class TicketsFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_tickets, container, false);
    }
}
```

Це пустий фрагмент, що у майбутньому буде доповнюватися функціоналом. Повернемося до файл XML(рис 3.4).

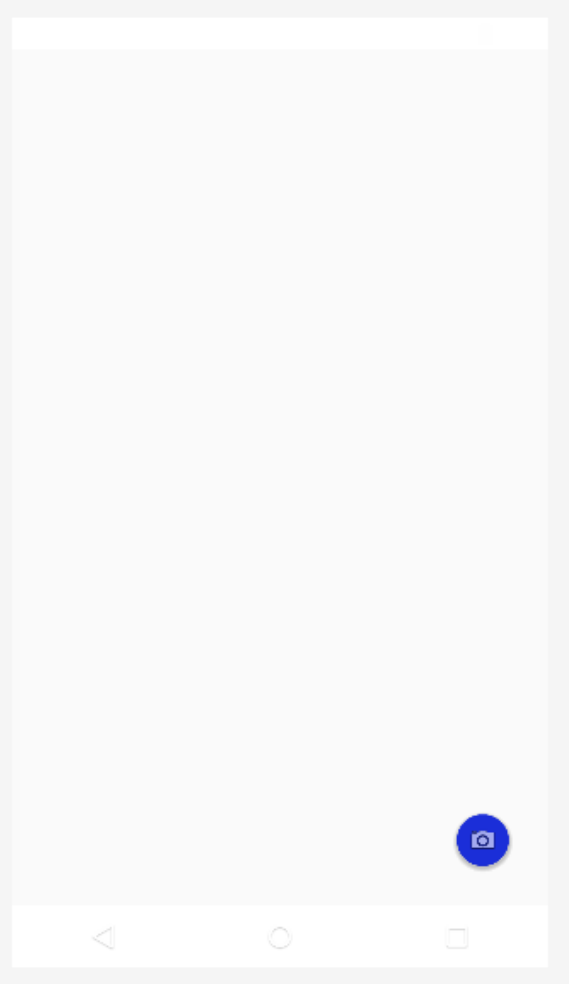


Рисунок 3.4 –Пустий фрагмент з кнопкою.

Ці фрагменти будуть підставлятися до основного вікна додатку. Створимо FrameLayout, у якому будуть знаходитися фрагменти. Він буде знаходитися у головній Activity поряд з кнопками навігації.

```
<FrameLayout
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_above="@id/navigation"/>

  <android.support.design.widget.BottomNavigationView
    android:id="@+id/navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="?android:attr/windowBackground"
    android:layout_alignParentBottom="true"
    app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:menu="@menu/navigation" />

    </RelativeLayout>
```

Тепер ми маємо список з декількох фрагментів, що будуть викликатися з різних класів у один фрейм.

Напишемо загальний метод, який обробляє перемикання фрагментів.

```
    private BottomNavigationView.OnNavigationItemSelectedListener navListener
            = new BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {
            Fragment selectedFragment = null;

            switch (item.getItemId()){
                case R.id.navigation_home:
                    selectedFragment = new HomeFragment();
                    break;
                case R.id.navigation_cards:
                    selectedFragment = new CardFragment();
                    break;
                case R.id.navigation_tickets:
                    selectedFragment = new TicketsFragment();
                    break;
            }

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                    selectedFragment).commit();

            return true;
        }
    };
```

Потрібно показувати перший фрагмент за замовчуванням, коли програма запускається. Це, очевидно, буде першим пунктом у нижній частині навігації, "Головна".

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new HomeFragment()).commit();

У підсумку отримали каркас додатку з трьома фрагментами для кожного розділу.

3.2 Відображення контенту за допомогою CardView та RecyclerView

Починаючи з Android Lollipop пропонується два нових керування, щоб зробити життя простіше, RecyclerView і CardView. Ці віджети приведуть зовнішній вигляд програми у відповідність з рекомендаціями, зазначеними в специфікації дизайну Google.

Завдяки v7 Support Library, є можливість використовувати віджети RecyclerView і CardView на пристроях більш ранніх версій Android, додавши наступні рядки розділу dependencies в файлі build.grade вашого проекту:

```
implementation 'com.android.support:recyclerview-v7:28.0.0'
implementation 'com.android.support:cardview-v7:28.0.0'
```

CardView це ViewGroup. Як будь-яку ViewGroup, її можна додати в Activity або Fragment через файл XML. Щоб створити порожній CardView, необхідно додати наступний код в макет XML:

```
<android.support.v7.widget.CardView
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
</android.support.v7.widget.CardView>
```

CardView може представляти, наприклад, людини і містити наступне:

- TextView з ім'ям людини
- TextView з віком людини
- ImageView з його фотографією

У нашому випадку це зображення карти та найменування. XML буде виглядати наступним чином:

```
<android.support.v7.widget.CardView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  xmlns:card_view="http://schemas.android.com/apk/res-auto"
  android:layout_margin="10dp"
  card_view:cardCornerRadius="6dp"
```

```
    card_view:cardElevation="4dp"
    android:id="@+id/card_view">

    <RelativeLayout
      android:layout_width="match_parent"
      android:layout_height="90dp"
      android:layout_margin="4dp">
      <ImageView
        android:id="@+id/imageView"
        android:layout_width="160dp"
        android:layout_height="90dp"
        android:padding="2dp"
        android:src="@drawable/ic_cards" />
      <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginStart="5dp"
        android:layout_marginTop="30dp"
        android:layout_toEndOf="@+id/imageView"
        android:text="CardName"
        android:textColor="@android:color/black"
        android:textSize="20sp"
        android:textStyle="bold" />
    </RelativeLayout>
  </android.support.v7.widget.CardView>
```

Якщо цей XML буде макетом Fragment і в полях TextView і ImageView значущі показники, так він буде відображатися на пристрої Android(рис 3.5).

З екземпляром RecyclerView трохи складніше. Однак визначити його в макеті XML досить просто. Ви можете визначити його в макеті наступним чином:

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/cardsRV"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

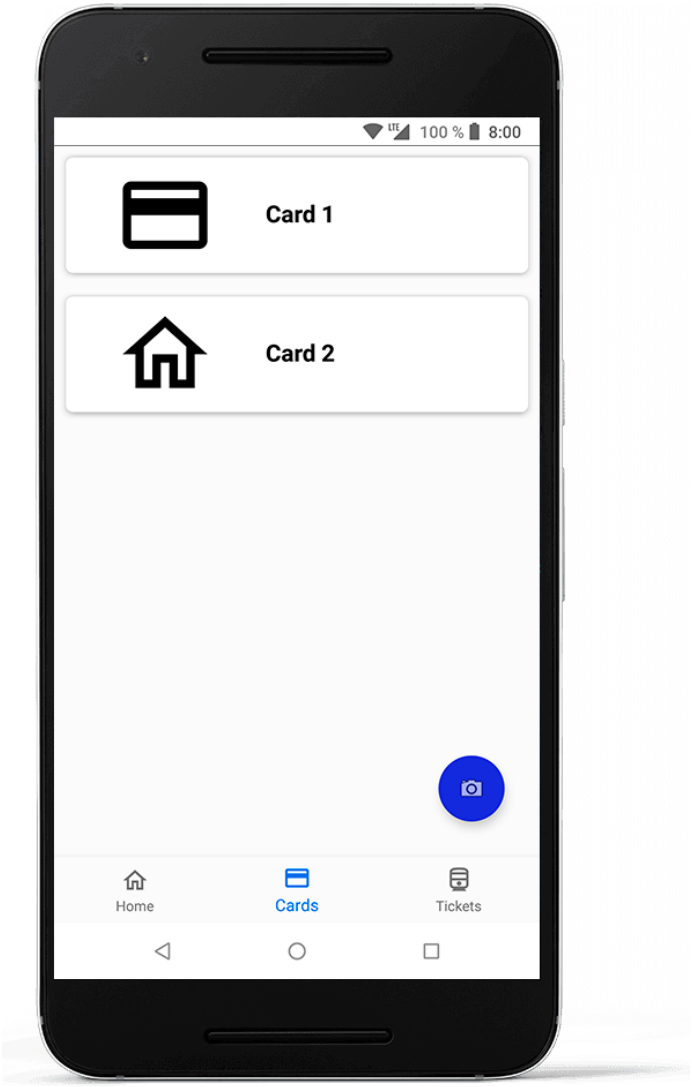


Рисунок 3.5 – Відображення контенту за допомогою CardView та RecyclerView

Щоб отримати доступ до RecyclerView в Fragment, потрібно записати наступний фрагмент:

RecyclerView rv = rootView.findViewById(R.id.cardsRV);

Якщо є впевненість, що розмір RecyclerView не зміниться, можете додати наступне для підвищення продуктивності:

rv.setHasFixedSize (true);

На відміну від ListView, RecyclerView потрібно LayoutManager для управління позиціями пунктів. Можна визначити свій LayoutManager шляхом розширення RecyclerView.LayoutManager класу. Хоча є можливість використовувати один з зумовлених підкласів LayoutManager:

LinearLayoutManager

GridLayoutManager

StaggeredGridLayoutManager

У цьому проекті я користуюся LinearLayoutManager. LayoutManager підклас, за замовчуванням, зробить ваш RecyclerView схожим на ListView.

LinearLayoutManager llm = new LinearLayoutManager(getActivity());

```
rv.setLayoutManager(llm);
```

Подібно ListView, RecyclerView потрібен адаптер для доступу до своїх даних. Але перш, ніж ми створимо адаптер, потрібно створити дані, з якими будемо працювати.

```
ArrayList<MenuItem> menuList = new ArrayList<>();
    menuList.add(new MenuItem(R.drawable.ic_cards, "Card 1"));
    menuList.add(new MenuItem(R.drawable.ic_home, "Card 2"));
```

Створюючи адаптер для RecyclerView, треба розширити RecyclerView.Adapter. Адаптер слід моделі view holder, що означає визначення користувальницького класу для розширення RecyclerView.ViewHolder. Ця модель мінімізує кількість викликів методу findViewById.

Раніше в цьому проекті ми вже визначили XML для CardView, який представляє картку. Будемо використовувати цей макет повторно. Усередині конструктора нашого ViewHolder Ініціалізуйте уявлення, які стосуються елементам нашого RecyclerView.

```
  public static class  MyViewHolder extends RecyclerView.ViewHolder{
//      public CardView mCardView;
      public TextView mTextView;
      public ImageView mImageView;

      public MyViewHolder(@NonNull View v) {
        super(v);
//        mCardView =  v.findViewById(R.id.card_view);
        mTextView = v.findViewById(R.id.textView);
        mImageView = v.findViewById(R.id.imageView);
      }
  }
```

Потім додайте конструктор в призначений для користувача адаптер, щоб він мав доступ до даних, що відображаються RecyclerView.

```
public  MyAdapter (ArrayList<MenuItem> menuList){
    mMenuList = menuList;
  }
```

RecyclerView.Adapter має три абстрактних методу, які ми повинні перевизначити. Почнемо з методу getItemCount. Це поверне кількість елементів, присутніх в даних.

```
@Override
  public int getItemCount() {
    return mMenuList.size();
  }
```

Потім перевизначаємо метод onCreateViewHolder. Як витікає з назви, цей метод викликається для ініціалізації користувача ViewHolder. Ми вказуємо макет, який повинен використовувати кожен елемент RecyclerView. Це виконується шляхом роздування компонування за допомогою LayoutInflater, передаючи результат конструктору ViewHolder.

```
public MyAdapter.MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.menu_item, parent, false);
    MyViewHolder mvh = new MyViewHolder(v);
    return mvh;
}
```

Переобумовлюємо onBindViewHolder, щоб вказати вміст кожного елемента RecyclerView. Цей метод схожий на метод getView адаптера ListView '. У нашому прикладі, тут ви повинні встановити значення назви і фотографії в CardView.

```
@Override
public void onBindViewHolder(MyViewHolder holder, int position) {
    MenuItem currentItem = mMenuList.get(position);
    holder.mImageView.setImageResource(currentItem.getmImageResource());
    holder.mTextView.setText(currentItem.getText());
}
```

Тепер, коли адаптер готовий, додайте наступний код в Fragment, щоб ініціалізувати і використовувати адаптер, викликаючи конструктор адаптера і метод setAdapter в RecyclerView:

```
MyAdapter adapter = new MyAdapter(menuList);
rv.setAdapter(adapter);
```

При запуску програми RecyclerView на пристрої Android буде список, представлений на малюнку(3.1.5).

3.3 Створення бази даних за допомогою SQLite

SQLite - це база даних з таблицями і запитами - все як у звичайних базах даних.

Для початка, трохи теорії щодо взаємодії програми та баз даних.

У додатку, при підключенні до бази даних ми вказуємо ім'я бази даних і версію. При цьому можуть виникнути такі ситуації:

1. База даних не існує. Це може бути наприклад в разі первинної установки програми. У цьому випадку програма повинна сама створити базу даних і всі таблиці в ній. І далі воно вже працює з щойно створеної базою даних.

2. База даних існує, але її версія застаріла. Це може бути в разі поновлення програми. Наприклад нової версії програми потрібні додаткові поля в старих таблицях або нові таблиці. У цьому випадку додаток має оновити існуючі таблиці та створити нові, якщо це необхідно.
3. База даних існує і її версія актуальна. У цьому випадку додаток успішно підключається до бази даних і працює.

Фраза "додаток повинен" рівнозначна фразі "розробник повинен", тобто це наше завдання. Для обробки описаних вище ситуацій нам потрібно створити клас, який є спадкоємцем для SQLiteOpenHelper. Назвемо його DBHelper. Цей клас надасть нам методи для створення або поновлення бази даних у випадках її відсутності або старіння.

onCreate - метод, який буде викликаний, якщо база даних, до якої ми хочемо підключитися - не існує

onUpgrade - буде викликаний в разі, якщо ми намагаємося підключитися до бази даних більш нової версії, ніж існуюча

Намалюємо екран для введення записів і очищення таблиці. Відкриваємо card.xml і пишемо:

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text=" @string/name"
   android:layout_marginLeft="5dp"
   android:layout_marginRight="5dp">
</TextView>
<EditText
   android:id="@+id/cardName"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_weight="1">
   <requestFocus>
   </requestFocus>
</EditText>
```

Створимо декілька полів для вводу інформації корситувачем. Для зручності поля підписані за назвою.

```
</LinearLayout>
<LinearLayout
```

```
android:id="@+id/linearLayout3"
android:layout_width="match_parent"
android:layout_height="wrap_content">
<TextView
   android:id="@+id/textView2"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text=" @string/date"
   android:layout_marginLeft="5dp"
   android:layout_marginRight="5dp">
</TextView>
<EditText
   android:id="@+id/adress"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_weight="1">
</EditText>
</LinearLayout>
```

Для обробки методів onCreate, onUpgrade та Clear створюємо кнопки(рис 3.6).

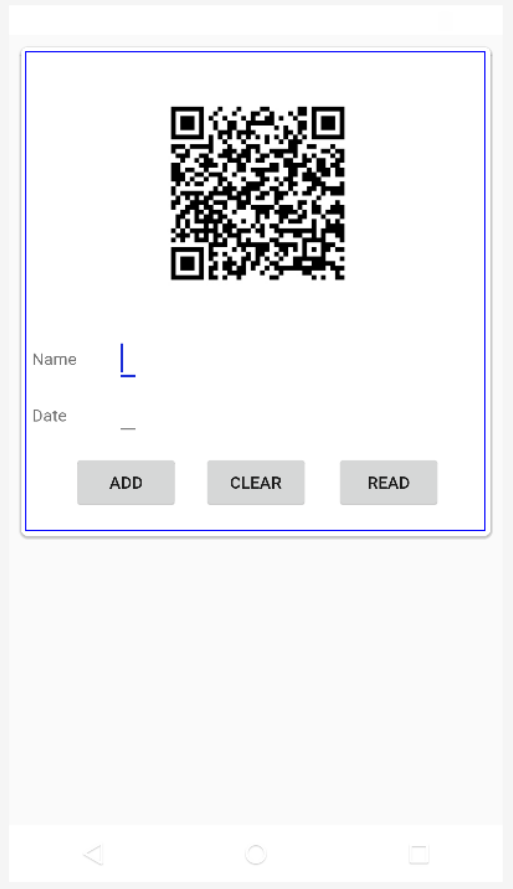


Рисунок 3.6 – Екран с карткою

```
    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <Button
            android:id="@+id/btnAdd"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Add">
        </Button>
<Button
            android:id="@+id/btnRead"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignTop="@+id/btnAdd"
            android:layout_alignParentEnd="true"
            android:layout_marginEnd="37dp"
            android:text="Read"></Button>
        <Button
            android:id="@+id/btnClear"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Clear">
        </Button>
    </LinearLayout>
</LinearLayout>
```

Створюємо cardList.java и прописуэмо роботу з базою даних. Створюємо об'єкт для створення і управління версіями базами даних[31]

```
dbHelper = new DBHelper (this);
```

Створюємо об'єкт для даних

```
ContentValues cv = new ContentValues ();
```

І отримуємо дані з полів введення

```
String name = cardName.getText (). ToString ();
String date = date.getText (). ToString ();
```

Підключаємося до бази даних:

```
SQLiteDatabase db = dbHelper.getWritableDatabase();
```

Підготуємо дані для вставки у вигляді пар: найменування стовпця - значення

```
cv.put("name", name);
cv.put("date", date);
```

Вставляємо запис і отримуємо її ID

```
long rowID = db.insert("mytable", null, cv);
Log.d(LOG_TAG, "row inserted, ID = " + rowID);
break;
case R.id.btnRead:
Log.d(LOG_TAG, "--- Rows in mytable: ---");
```

Робимо запит всіх даних з таблиці mytable, отримуємо Cursor

```
Cursor c = db.query("mytable", null, null, null, null, null, null);
```

Ставимо позицію курсора на перший рядок вибірки, якщо у вибірці немає рядків, повернеться false

```
if (c.moveToFirst()) {
```

Визначаємо номери стовпців на ім'я в вибірці

```
int idColIndex = c.getColumnIndex("id");
int nameColIndex = c.getColumnIndex("name");
int dateColIndex = c.getColumnIndex("date");

do {
```

Отримуємо значення за номерами стовпців і пишемо все в лог

```
Log.d(LOG_TAG,
        "ID = " + c.getInt(idColIndex) +
        ", name = " + c.getString(nameColIndex) +
", date = " + c.getString(dateColIndex));
```

Перехід на наступний рядок, а якщо наступної немає (поточна - остання), то false - виходимо з циклу

```
        } while (c.moveToNext());
} else
Log.d(LOG_TAG, "0 rows");
c.close();
break;
case R.id.btnClear:
Log.d(LOG_TAG, "--- Clear mytable: ---");
```

```
Видаляємо всі записи
        int clearCount = db.delete("mytable", null, null);
        Log.d(LOG_TAG, "deleted rows count = " + clearCount);
        break;
        }
Закриваємо підключення до бази даних
        dbHelper.close();
        }
        class DBHelper extends SQLiteOpenHelper {

         public DBHelper(Context context) {
Конструктор суперкласу
        super(context, "myDB", null, 1);
        }

        @Override
        public void onCreate(SQLiteDatabase db) {
        Log.d(LOG_TAG, "--- onCreate database ---");
Створюєио таблицю з полями
        db.execSQL("create table mytable ("
        + "id integer primary key autoincrement,"
        + "name text,"
        + "date text" + ");");
           }
```

У методі Activity - onCreate ми визначаємо об'єкти, присвоюємо обробники і створюємо об'єкт dbHelper класу DBHelper для управління бази даних. Сам клас буде описаний нижче.

Далі дивимося метод Activity - onClick, в якому ми обробляємо натискання на кнопки.

Клас ContentValues використовується для вказівки полів таблиці і значень, які ми в ці поля будемо вставляти. Ми створюємо об'єкт cv, і пізніше його використовуємо. Далі ми записуємо в змінні значення з полів введення. Потім, за допомогою методу getWritableDatabase підключаємося до бази даних і отримуємо об'єкт SQLiteDatabase. Він дозволить нам працювати з бази даних. Ми будемо використовувати його методи insert - вставка запису, query - читання, delete - видалення. У них багато різних параметрів на вхід, але ми поки використовуємо самий мінімум[32].

Далі дивимося, яка кнопка була натиснута:

btnAdd - додавання запису в таблицю mytable. Ми заповнюємо об'єкт cv парами: ім'я поля і значення. I (при вставці запису в таблицю) в зазначені поля будуть вставлені відповідні значення. Ми заповнюємо поля name і date. id у нас заповниться автоматично (primary key autoincrement). Викликаємо метод insert - передаємо йому ім'я таблиці і об'єкт cv з вставляються значеннями. Другий аргумент методу використовується, при вставці в таблицю порожніх рядків. Нам це зараз не потрібно, тому передаємо null. Метод insert повертає ID вставленого рядка, ми його зберігаємо в rowID і виводимо в лог.

btnRead - читання всіх записів з таблиці mytable. Для читання використовується метод query. На вхід йому подається ім'я таблиці, список запитуваних полів, умови вибірки, групування, сортування. Оскільки нам потрібні всі дані у всіх полях без угруповань та угруповань - ми використовуємо всюди null. Тільки ім'я таблиці вказуємо. Метод повертає нам об'єкт класу Cursor. Його можна розглядати як таблицю з даними. Метод moveToFirst - робить перший запис у Cursor активної і заодно перевіряє, чи є взагалі записи в ньому (тобто вибралося чи що-небудь в методі query). Далі ми отримуємо порядкові номери стовпців в Cursor за їхніми іменами за допомогою методу getColumnIndex. Ці номери потім використовуємо для читання даних в методах getInt і getString і виводимо дані в лог. За допомогою методу moveToNext ми перебираємо всі рядки в Cursor поки не добираємося до останньої. Якщо ж записів не було, то виводимо в лог відповідне повідомлення - 0 rows. В кінці закриваємо курсор (звільняємо займані ним ресурси) методом close, тому що далі ми його ніде не використовуємо.

btnClear - очищення таблиці. Метод delete видаляє записи. На вхід передаємо ім'я таблиці і null в якості умов для видалення, а значить буде усе видалено. Метод повертає кількість вилучених записів.

Після цього закриваємо з'єднання з бази даних методом close.

Клас DBHelper є вкладеним в MainActivity. Цей клас повинен успадковувати клас SQLiteOpenHelper.

У конструкторі ми викликаємо конструктор суперкласу і передаємо йому:

- context - контекст
- mydb - назва бази даних
- null - об'єкт для роботи з курсором, нам поки не потрібен, тому null
- 1 - версія бази даних

У методі onCreate цього класу ми використовуємо метод execSQL об'єкта SQLiteDatabase для виконання SQL-запиту, який створює таблицю. Нагадаю - цей метод

викликається, якщо бази даних не існує і її треба створювати. За запитом видно, що ми створюємо таблицю mytable з полями id, name і date.

Метод onUpgrade поки не заповнюємо, тому що використовуємо одну версію бази даних і міняти її не плануємо.

Для роботи з базою даних ми використовували два класи:

– DBHelper, що успадковує SQLiteOpenHelper. У його конструкторі ми викликаємо конструктор супер-класу і вказуємо ім'я і версію бази даних. Метод getWritableDatabase виконує підключення до бази даних і повертає нам об'єкт SQLiteDatabase для роботи з нею. Метод close закриває підключення до бази даних. У разі, коли база даних відсутня або застаріла, клас надає нам самим реалізувати створення або оновлення в методах onCreate і onUpgrate.
– SQLiteDatabase. Містить методи для роботи з даними - тобто вставка, оновлення, видалення та читання.

3.4 Виявлення штрих-кодів за допомогою Mobile Vision API

Тепер, коли наша програма повністю налаштована, настав час створити інтерфейс, який дозволяє користувачеві виявляти штрих-код у зображенні.

Потрібно додати код, зображений нижче, у "build gradle", щоб додати файли бібліотеки до нашого читача штрих-коду:

implementation 'com.google.android.gms:play-services-vision:12.0.0'

Тепер ми збираємося включити бібліотеку для зчитування штрих-коду для коректної роботи. По-перше, нам потрібно побудувати коди gradle і Android Hive. Потім додаємо до нього код штрих-коду.

build.gradle

```
dependencies {

   implementation 'info.androidhive:barcode-reader:1.1.5'
   implementation 'com.google.android.gms:play-services-vision:11.0.2'
}
```

Настав час додати фрагмент Activity, до якого можна викликати фрагмент Camera.

```
<fragment
     android:id="@+id/barcode_scanner"
     android:name="info.androidhive.barcode.BarcodeReader"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
```

```
    app:auto_focus="true"
    app:use_flash="false" />
```

Тепер додамо наступний код, щоб правильно працювати наш проект і назвати його як BarcodeReader.BarcodeReader.

```
import com.google.android.gms.vision.barcode.Barcode;
import java.util.List;

public class MainActivity extends AppCompatActivity implements BarcodeReader.BarcodeReaderListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scan);
    }

    @Override
    public void onScanned(Barcode barcode) {
        // single barcode scanned
    }

    @Override
    public void onScannedMultiple(List<Barcode> list) {
        // сканування декількох штрих-кодів
    }

    @Override
    public void onBitmapScanned(SparseArray<Barcode> sparseArray) {

    }

    @Override
    public void onScanError(String s) {
        // зараз сканується помилка
    }

    @Override
```

```
    public void onCameraPermissionDenied() {
        // дозвіл камери фактично відмовлено
    }
}
```

Відсканований результат буде повернуто в методі onScanned () або onScannedMultiple ().

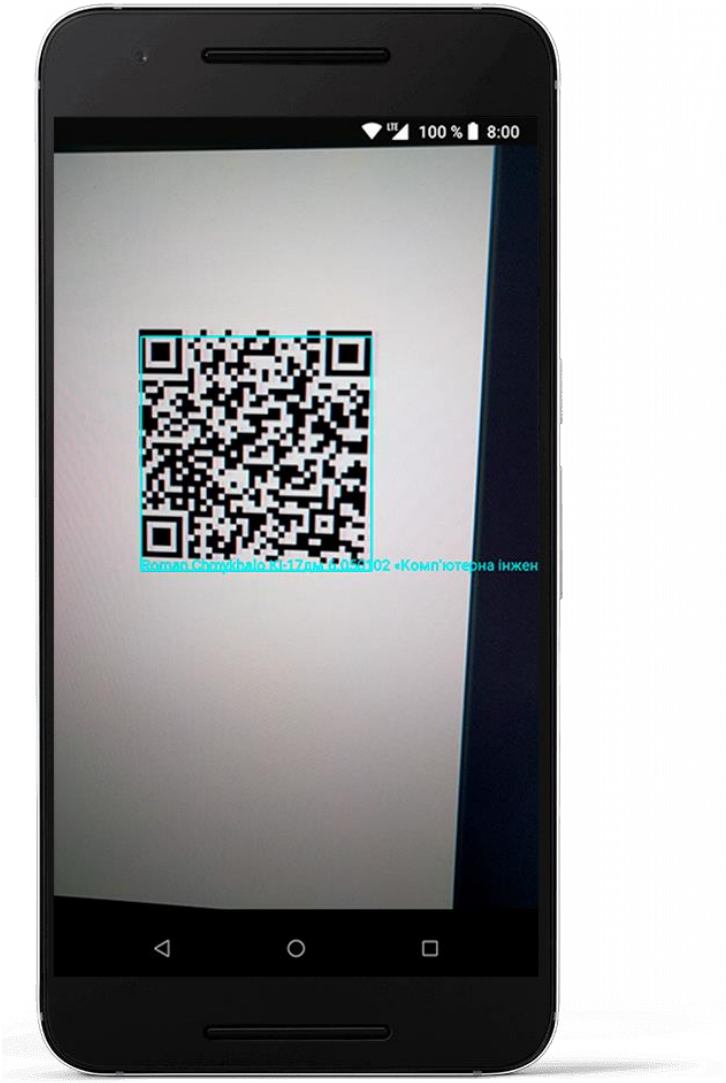


Рисунок 3.7– Сканування QR-коду у додатку

3.5 Висновки до розділу 3

В ході дослідження та розробки додатку, було створено новий продукт, який спроможний конкурувати з іншими додатками зі схожим функціоналом. Це було досягнуто завдяки аналізу ринку додатків, та виявлення нових функцій, що не представлені серед конкурентів.

Органайзер дисконтних карт забезпечує можливість зберігати дані, відскановані з пластикових карт, квитків або інших носіїв інформації у вигляді штрих-коду. Цей додаток

буде систематизувати дані в одному місці для зручності користувача. Дані заносяться в програму за допомогою камери пристрою.

Головне вікно органайзера складається з початкової сторінки та навігаційного бару, що за допомогою кнопок дозволяє перемикатися між екранами додатку (рис. 3.8). На двох інших сторінках зберігаються дані, що були внесені користувачем.

Представлений органайзер для мобільних пристроїв на базі ОС Android був розроблений на сучасному рівні, використовуючи такі технології мобільних обчислень, як розпізнавання штрих-кодів за допомогою сторонніх бібліотек, збереження даних, отриманих ззовні. Дослідження нових методів та інструментів дозволило поліпшити і вдосконалити вже наявні, а також адаптувати під потреби ринку за допомогою впровадження нового функціоналу. Використані новітні і вдосконалені наявні технології при розробці програми. Підвищено зручність використання за допомогою удосконалення інтерфейсу додатку.

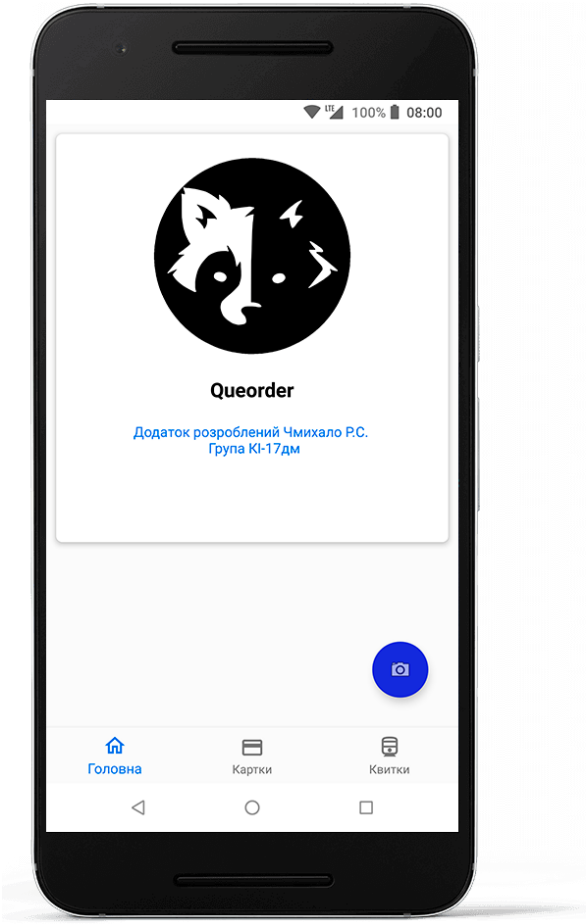


Рисунок 3.8 – Головна сторінка додатку “Queorder”

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

У даному розділі проведено аналіз потенційних небезпечних і шкідливих виробничих факторів, причин пожеж. Розглянуто заходи, які дозволяють забезпечити гігієну праці та виробничу санітарію. На підставі аналізу розроблено заходи з техніки безпеки і рекомендації з пожежної профілактики.

Завданням даної роботи магістра було розглянути моделі та програмні засоби розробки клієнтської частини комп'ютерної системи моніторингу якості освіти. Так як в процесі проектування використовувався персональний комп'ютер (ПК), то аналіз потенційно небезпечних і шкідливих виробничих факторів виконується для робочого місця з ПК, на якому буде використовуватися розроблений засіб.

4.1. Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. У законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і засобів, спрямованих на збереження життя, здоров'я і працездатності людини в процесі трудової діяльності.

4.2. Аналіз стану умов праці

Умови праці повинні задовольняти таким вимогам, які дали б можливість людині виконувати роботу без шкоди для здоров'я, без перевтоми і з високою продуктивністю. Для вибору показників умов праці при проведенні аналізу слід керуватися чинними в Україні «Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10 [33][43].

Робота з засобом комп`ютерною системою буде проходити в приміщенні Центру моніторингу якості освіти. Для даної роботи досить однієї людини, для якого надано робоче місце зі стаціонарним комп'ютером. Виконувана робота за ступенем тяжкості відноситься до категорії "легка 1б". До неї відносяться роботи, вироблені сидячи, стоячи або пов'язані з

ходьбою, але не потребують систематичного фізичного напруження чи підняття і перенесення важких предметів.

4.2.1. Вимоги до приміщень

Згідно з [34] розмір площі для одного робочого місця оператора персонального комп'ютера повинно бути не менше 6 кв. м, а обсяг - не менше 20 куб. м. Отже, дане приміщення повністю відповідає зазначеним нормам.

4.2.2. Вимоги до організації місця праці

Робочий стіл на досліджуваному місці також містить досить простору для ніг. Крісло, використовується в якості робочого сидіння, є підйомно-поворотним, має підлокітники і можливість регулювання по висоті і куту нахилу спинки, також воно м'яке і виконано з екологічної шкіри, що дозволяє працювати в комфорті. Екран монітора знаходиться на відстані 0.8 м, клавіатура має можливість регулювання кута нахилу 5-15°С. Отже, за всіма параметрами робоче місце відповідає нормативним вимогам. Приміщення кабінету знаходиться на другому поверсі трьох поверхового будинку і має обсяг 78 м^3, площа - 24 м^2. У цьому кабінеті обладнано три місця праці, з яких два укомплектовані ПК.

За ступенем пожежної безпеки приміщення належить до категорії В. Кабінет оснащений переносним вуглекислотним вогнегасником ВВК-5. Є аптечка для надання долікарської допомоги, а також в кабінеті роблять вологе прибирання і щодня провітрюють приміщення.

4.3. Виробнича санітарія

На підставі аналізу небезпечних і шкідливих факторів при експлуатації, пожежної безпеки можуть бути в подальшому вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення тощо [36].

4.3.1. Аналіз небезпечних і шкідливих факторів при експлуатації системи

Аналіз небезпечних і шкідливих виробничих факторів виконується в табличній формі (табл. 4.1).

Таблиця 4.1 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, принтерів, сканерів чи/або серверного обладнання для роботи	2	[36]
- підвищений рівень шуму на робочому місці	-//-	2	[37]
- підвищений рівень вібрації	-//-	2	[37] [38]
- підвищена або знижена вологість повітря	-//-	2	[36]
- підвищена або знижена рухливість повітря	-//-	1	[36]
- підвищений рівень іонізуючого випромінення в робочій зоні	-//-	2	[36] [39]
- підвищений рівень електромагнітного випромінення	-//-	2	[39]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[40] [41]
- підвищений рівень статичної електрики	-//-	2	[41]
- підвищена напруженість електричного поля	-//-	2	[41]
- підвищена напруженість магнітного поля	-//-	2	[41]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[42]

Продовження таблиці 4.1 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	[42]
- підвищена яскравість світла	порушення умов праці (організації місця праці- налагодження моніторів)	1	[42]
- понижена контрастність	-//-	1	[34]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[34] [43]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці- сидіння користувача,) та організації робочого часу - безперервна робота)	2	[34] [43]

4.4. Гігієнічні вимоги до параметрів виробничого середовища

4.4.1. Мікроклімат

Мікроклімат робочих приміщень - це клімат внутрішнього середовища цих приміщень, який визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. В даному приміщенні проводяться роботи, що виконуються сидячи і не потребують динамічного фізичного напруження, то для нього відповідає категорія робіт I_a. Отже оптимальні значення для температури, відносної вологості і рухливості повітря для зазначеного робочого місця відповідають [37] і наведені в табл. 4.2:

Таблиця 4.2 – Норми мікроклімату робочої зони об'єкта

Період року	**Категорія робіт**	**Температура C^0**	**Відносна вологість %**	**Швидкість руху повітря, м/с**
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

4.4.2. Освітленість

Розрахунок освітлення.

Для виробничих і адміністративних приміщень світловий коефіцієнт приймається не менш -1/8, в побутових - 1/10:

$$\sqrt{a^2 + b^2} * S_b = \left(\frac{1}{8} \div \frac{1}{10}\right) * S_n \quad (4.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$$S_n = a * b = 4.5 * 2.7 = 12.15 \text{ м}^2, \quad (4.2)$$

$$S_b = 1.4 * 1.3 = 1.82 \text{ м}^2 \quad (4.3)$$

$$S_{вік} = 12.15 * \frac{1}{10} = 1{,}215 \text{м}^2 \quad (4.4)$$

Приймаємо 1 вікно площею S=1,82 м^2

Розрахунок штучного освітлення проводиться за коефіцієнтами використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n проводиться за формулою (5.3):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \quad (4.5)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 12.15 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання і ДРЛ; Z = 1,1 для світлодіодних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, що залежить від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 2000лм.

Підставивши числові значення в формулу (4.2), отримуємо:

$$n = \frac{300 * 12.15 * 1.1 * 1.5}{2000 * 0.575 * 2} \approx 2.6$$

Отже для нормативного освітлювання приміщення, у якому виконується дослідницька робота, повинно працювати 3 світильники, що містять по дві світлодіодних лампи з потужністю 18 Вт (еквівалент потужності лампи розжарювання: 150 Вт).

4.5. Заходи з організації виробничого середовища і попередження виникнення надзвичайних ситуацій

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Відповідно до класифікації приміщень за ступенем небезпеки ураження електричним струмом [43], приміщення в якому проводяться всі роботи належить до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, і 360 В. Опір контуру заземлення повинен мати не більше 4 Ом.

Послідовність розрахунку:

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \quad (4.6)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.}=R_д$.

Підставивши числові значення в формулу (4.6), отримуємо:

$$R_{шт.з.} = \frac{4 * 40}{40 - 4} \approx 4 \text{ Ом} \quad (4.7)$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом • м. Приблизне значення питомої опору глини приймаємо ρ=40 Ом•м (табличне значення).

3) Розрахункова питомий опір ґрунту, $\rho_{розр}$, Ом•м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом•м по формулі:

$$\rho_{розр..} = \psi \cdot \rho, \quad (4.8)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів і кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом•м.

$$\rho_{розр.в} = 1,7 \cdot 40 = 68 \text{ Ом•м} \quad (4.9)$$

$$\rho_{розр.г} = 5,5 \cdot 40 = 220 \text{ Ом•м}$$

4) Розраховується опір розтікання струму вертикального заземлення $R_в$, Ом, по (4.6).

$$R_{в} = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_{в}} \cdot \left(\ln \frac{2 \cdot l_{в}}{d_{ст}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_{в}}{4 \cdot t - l_{в}} \right), \quad (4.10)$$

де $l_{в}$ – довжина вертикального заземлювача (для труб - 2−3 м; $l_{в}$=3 м);

$d_{ст}$ – діаметр стрижня (для труб - 0,03−0,05 м; $d_{ст}$=0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за формулою (4.11):

$$t = h_{в} + \frac{l_{в}}{2}, \quad (4.11)$$

де $h_{в}$ – глибина закладення вертикальних заземлювачів (0,8 м); тоді

$$t = 0,8 + \frac{3}{2} = 2,3 \text{м} \quad (4.12)$$

$$R_{в} = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом} \quad (4.13)$$

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_{в}$:

$$n = \frac{2 \cdot R_{в}}{R_{д}} = \frac{2 \cdot 18,5}{4} = 9,25 \quad (4.14)$$

І визначається коефіцієнт використання вертикальних електродів групового заземлення без урахування впливу сполучної стрічки $\eta_{в}$ =0,57 (табличне значення).

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_{в}$, шт.:

$$n_{в} = \frac{2 \cdot R_{в}}{R_{д} \cdot \eta_{в}} = \frac{2 \cdot 18,5}{4 \cdot 0,57} = 16,2 \approx 16 \quad (4.15)$$

7) Визначається довжина сполучної стрічки горизонтального заземлювача $l_{с}$, м:

$$l_{с} = 1,05 \cdot L_{в} \cdot (n_{в} - 1), \quad (4.16)$$

де $L_{в}$ – відстань між вертикальними заземлювачами, (прийняти $L_{в}$ = 3м);

$n_{в}$ – необхідну кількість вертикальних заземлювачів.

$$l_{с} = 1,05 \cdot 3 \cdot (16 - 1) \approx 48 \text{м} \quad (4.17)$$

8) Визначається опір розтіканню струму горизонтального заземлювача (сполучної стрічки) $R_{г}$, Ом:

$$R_{г} = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_{с}} \cdot \ln \frac{2 \cdot l_{с}^{2}}{d_{см} \cdot h_{г}}, \quad (4.18)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_{г}$ – глибина закладення горизонтальних заземлювачів (0,5 м);

$l_{с}$ - довжина сполучної стрічки горизонтального заземлювача $l_{с}$, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0,95 \cdot 0,15 \cdot 0,5} = 8,1 \text{ Ом} \quad (4.19)$$

9) Визначається коефіцієнт використання горизонтального заземлювача ηс відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання сполучної смуги $\eta_с$=0,3 (табличне значення).

10) Розраховується результуючий опір заземлюючого електрода з урахуванням сполучної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_с + R_г \cdot n_в \cdot \eta_в} \leq R_д. \quad (4.20)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпека будівлі, так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \leq R_д \quad (4.21)$$

У приміщеннях повинна бути пожежна сигналізація, вогнегасник. У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки щодо запобігання можливих наслідків та усунення пожежі [43].

4.6. Охорона навколишнього природного середовища

4.6.1. Загальні дані з охорони навколишнього природного середовища

Діяльність за темою магістерської роботи, а саме: розгляд моделей та програмних засобів розробки клієнтської частини комп'ютерної системи, у процесі її виконання впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища» [44], Законом України «Про забезпечення санітарного та епідемічного благополуччя населення» [45], Законом України «Про відходи» [46], Законом України «Про охорону атмосферного повітря» [47], Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру» [48], Водний кодекс України [49].

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на видалення (знешкодження), утилізацію, тощо в ІТ галузі.

В процесі діяльності комп’ютерної системи виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

- відпрацьовані люмінесцентні лампи - І клас небезпеки
- батарейки та акумулятори (малі) -ІІІ клас небезпеки
- змінні носії інформації - IV клас небезпеки
- відходи друкуючих пристроїв - IV клас небезпеки
- макулатура - IV клас небезпеки
- матеріали пакувальні пластмасові забруднені (ємності з-під тонеру, фарби, тощо.) - IV клас небезпеки
- побутові відходи - IV клас небезпеки

4.6.2. Вимоги до збору, пакування та розміщення відходів ІТ галузі

Наводяться вимоги зберігання виявлених за своєю роботою відходів відповідно до вимог Державних санітарних правил і норм [50].

Способи тимчасового зберігання відходів визначаються видом, агрегатним станом і класом небезпеки відходів:

- відходи І класу небезпеки зберігаються в герметичній тарі (сталеві бочки, контейнери). У міру наповнення тару з відходами закривають герметично сталевий кришкою;

- відходи ІІ класу небезпеки в залежності від агрегатного стану зберігаються в поліетиленових мішках, бочках, сховищах та інших видах тари, яка запобігає поширенню шкідливих речовин;

- відходи ІІІ класу небезпеки зберігаються в тарі, яка забезпечує локалізацію зберігання, дозволяє виконувати вантажно-розвантажувальні і транспортні роботи і виключає поширення в ОС шкідливих речовин;

- відходи IV класу небезпеки можуть зберігатися відкрито на промисловому майданчику у вигляді конусоподібної купи, звідки їх автонавантажувачем перевантажують у самоскид і доставляють на місце утилізації або захоронення.

Не допускається змішування відходів різних видів і класів небезпеки з будівельними і побутовими відходами, відходами дерев'яної, металевої, синтетичної тари, відходами текстильних матеріалів (старий спецодяг, ганчірки) тощо.

Особливий контроль наділяється збору і зберіганню відпрацьованих ртутьвмісних ламп (енергоощадних) як відходам І класу небезпеки, що збираються і обов’язково передаються на утилізацію підприємствам, що мають ліцензію на поводження з такими небезпечними відходами.

Всі відходи, що утворюються в процесі діяльності/роботи, підлягають обліку.

Вимоги безпеки при поводженні з відходами:

Під час роботи з відходами (прибирання виробничих приміщень, збір і сортування, навантаження, транспортування, розвантаження та ін.) працівники та обслуговуючий персонал підприємства повинні бути забезпечені засобами індивідуального захисту та дотримуватися вимог інструкцій з охорони праці, що діють на підприємстві.

Відвантаження таких відходів здійснюється відповідно до договору (контракту)

Побутові та будівельні відходи вивозяться на полігон твердих побутових відходів міста, також відповідно до договору з комунальним дорожньо-експлуатаційним управлінням.

Особи, винні в порушенні встановленого порядку поводження з відходами (порушення правил обліку відходів, самовільне складування і видалення відходів, передача відходів в інші підприємства/організації з порушенням встановлених правил), згідно законодавства несуть дисциплінарну, адміністративну або кримінальну відповідальність.

4.6.3. Визначення впливу та заходів щодо поводження з відходами ІТ галузі

З метою визначення та прогнозування впливу відходів на навколишнє середовище, своєчасного виявлення негативних наслідків, їх запобігання відповідно до Закону України «Про відходи» [46] повинен здійснюватися моніторинг місць утворення, зберігання, і видалення відходів. Відомості про місце утворення та місце розташування відходів зазначаються на «План схемі місці розміщення відходів організації / виробництва».

4.7 Висновок до розділу 4

У результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом, написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, наведено інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВКИ

Були досліджені методи створення і виконано розробку додатку для роботи з дисконтними картками і проїзними документами. Додаток може зберігати інформацію про всі дисконтні картки користувача, і може замінити дисконтні картки на фізичних носіях.

Розглянуті основні положення розшифрування штрих-кодів та QR-кодів, їх реалізації за допомогою спеціалізованих програмних бібліотек. Розглянуто сильні та слабкі сторони бібліотек, які можуть бути використані при створенні додатку. Було досліджено найпопулярніші реалізації подібних продуктів на ринку, в розробленому додатку були усунуті їх головні недоліки. Результатом розробки став Android-додаток, що містить в собі дані, які були отримані за допомогою камери пристрою, та розшифровані зі штрих-коду або QR-коду за допомогою сторонніх бібліотек. Використання новітньої бібліотеки з розпізнавання елементів за допомогою камери дозволило пришвидшити роботу відносно інших додатків, що переважно працюють на старих та громіздких бібліотеках.

Органайзер дисконтних карток забезпечує можливість зберігати дані, відскановані з пластикових карт, квитків або інших носіїв інформації у вигляді штрих-коду. Цей додаток буде систематизувати дані в одному місці для зручності користувача. Дані заносяться в програму за допомогою камери пристрою.

Представлений органайзер для мобільних пристроїв на базі ОС Android був розроблений на сучасному рівні, використовуючи такі технології мобільних обчислень, як розпізнавання штрих-кодів за допомогою сторонніх бібліотек, збереження даних, отриманих ззовні. Дослідження нових методів та інструментів дозволило поліпшити і вдосконалити вже наявні, а також адаптувати під потреби ринку за допомогою впровадження нового функціоналу. Використані новітні і вдосконалені наявні технології при розробці програми. Підвищено зручність використання за допомогою удосконалення інтерфейсу додатку.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Что такое штрих-код [Електронний ресурс] // Калькулятор справочный портал. URL: https://www.calc.ru/Chto-Takoye-Shtrikhkod.html
2. Маршалов А. Сделайте ваши QR-коды более красивыми [Електронний ресурс] // habr. 2006. URL: https://habr.com/post/117778/
3. Constantine @tgx Q. Читаем QR код [Електронний ресурс] // habr. 2006. URL: https://habr.com/post/127197/
4. Google Pay (UA) – Зручний спосіб оплати [Електронний ресурс] // Google. URL: https://pay.google.com/intl/uk_ua/about/
5. Stocard [Електронний ресурс] // Stocard. 2018. URL: https://stocardapp.com/en/au
6. srowen. ZXing [Електронний ресурс] // Github. 2007. URL: https://github.com/zxing/zxing
7. VS READER QR [Електронний ресурс] // www.visionsmarts.com. URL: http://www.visionsmarts.com/products/vs-reader-qr.html
8. VS BARCODE READER [Електронний ресурс] // www.visionsmarts.com URL: http://www.visionsmarts.com/products/vs-barcode-reader.html
9. Barcode API Overview [Електронний ресурс] // developers.google.com. 2018. URL: https://developers.google.com/vision/android/barcodes-overview
10. BottomNavigationView [Електронний ресурс] // developer.android.com. URL: https://developer.android.com/reference/android/support/design/widget/BottomNavigationView
11. BottomNavigationView [Електронний ресурс] // developer.android.com. URL: https://developer.android.com/reference/com/google/android/material/bottomnavigation/BottomNavigationView
12. Bottom Navigation Bar Android Tutorial [Електронний ресурс] // blog.iamsuleiman.com. URL: https://blog.iamsuleiman.com/bottom-navigation-bar-android-tutorial/
13. Fragments [Електронний ресурс] // developer.android.com. URL: https://developer.android.com/guide/components/fragments
14. Fragments [Електронний ресурс] // developer.android.com. URL: https://developer.android.com/reference/android/support/v4/app/Fragment
15. Фрагменты [Електронний ресурс] // http://developer.alexanderklimov.ru. URL: http://developer.alexanderklimov.ru/android/theory/fragments.php
16. Create a Card-Based Layout. [Електронний ресурс] //developer.android.com. URL: https://developer.android.com/guide/topics/ui/layout/cardview

17. CardView [Електронний ресурс] //developer.android.com. URL: https://developer.android.com/reference/android/support/v7/widget/CardView
18. Create a List with RecyclerView. [Електронний ресурс] //developer.android.com. URL: https://developer.android.com/guide/topics/ui/layout/recyclerview
19. RecyclerView [Електронний ресурс] //developer.android.com. URL: https://developer.android.com/reference/android/support/v7/widget/RecyclerView
20. bejibx. О RecyclerView и выделении элементов [Електронний ресурс] // habr.com. URL: https://habr.com/post/127197/
21. Bill Phillips. RecyclerView Part 1: Fundamentals For ListView Experts [Електронний ресурс] // bignerdranch.com URL: https://www.bignerdranch.com/blog/recyclerview-part-1-fundamentals-for-listview-experts/
22. Bill Phillips. RecyclerView Part 2: Choice Modes [Електронний ресурс] // bignerdranch.com URL: https://www.bignerdranch.com/blog/recyclerview-part-2-choice-modes/
23. Save data using SQLite. [Електронний ресурс] //developer.android.com. URL: https://developer.android.com/training/data-storage/sqlite#java
24. Работа с базами данных SQLite. Подключение к базе данных SQLite. [Електронний ресурс] // metanit.com. URL: https://metanit.com/java/android/14.5.php
25. Android. Работа с базой данных SQLite [Електронний ресурс] //devcolibri.com. URL: https://devcolibri.com/android-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%D1%81-%D0%B1%D0%B0%D0%B7%D0%BE%D0%B9-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-sqlite/
26. Barcode Detection with the Mobile Vision API [Електронний ресурс] // developers.google.com. URL: https://codelabs.developers.google.com/codelabs/bar-codes/
27. Track Faces and Barcodes //developer.android.com. URL: https://developers.google.com/vision/android/multi-tracker-tutorial
28. Android Barcode / QR Code Scanner using Google Mobile Vision – Building Movie Tickets App [Електронний ресурс] // www.androidhive.info. URL: https://www.androidhive.info/2017/08/android-barcode-scanner-using-google-mobile-vision-building-movie-tickets-app/
29. Product icons [Електронний ресурс] // material.io. URL: https://material.io/design/iconography/product-icons.html
30. BottomNavigationView [Електронний ресурс] // developer.android.com. URL: https://developer.android.com/reference/com/google/android/material/bottomnavigation/BottomNavigationView

31. Использование существующей БД SQLite [Електронний ресурс] // metanit.com. URL: https://metanit.com/java/android/14.3.php
32. База данных SQLITE и фрагменты [Електронний ресурс] // maxfad.ru. URL: https://maxfad.ru/programmer/android/307-sozdanie-raspisaniya-urokov-dlya-android-ustrojstv-chast-2-baza-dannykh-sqlite-i-fragmenty.html
33. ДНАОП 0.00-1.31-99 Правила охорони праці під час експлуатації електронно-обчислювальних машин.
34. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.
35. НАПБ Б.02.005-2003 Типове положення про інструктажі, спеціальне навчання та перевірку знань з питань пожежної безпеки на підприємствах, в установах та організаціях України.
36. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень.
37. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку.
38. ДСТУ ГОСТ 12.1.012-2008 Система стандартів безпеки праці. Вібраційна безпека. Загальні вимоги.
39. ГОСТ 12.1.006-84 Правила охорони праці під час оброблення і використання алюмінієвих і титанових сплавів.
40. ГОСТ 13109-97 Норми якості електричної енергії в системах електропостачання загального призначення.
41. ГОСТ 12.1.030-81 Система стандартів безпеки праці. Електробезпека. Захисне заземлення. Занулення.
42. ДБН В.2.5-28:2015 Освітлення у приміщеннях.
43. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин.
44. Закон України «Про охорону навколишнього природного середовища».
45. Закон України «Про забезпечення санітарного та епідемічного благополуччя населення».
46. Закон України «Про відходи».
47. Закон України «Про охорону атмосферного повітря».
48. Закон України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру».
49. Водний кодекс України.
50. ДСанПіН 2.2.7.029 Гігієнічні вимоги щодо поводження з промисловими відходами та визначення їх класу небезпеки для здоров'я населення.

Додаток А

Код елементів android-органайзера «Queorder»

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@id/navigation"/>

    <android.support.design.widget.BottomNavigationView
        android:id="@+id/navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?android:attr/windowBackground"
        android:layout_alignParentBottom="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:menu="@menu/navigation" />

</RelativeLayout>
```

MainActivity.java

```
package com.example.roman.queorder;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v4.app.Fragment;
import android.support.v7.app.AppCompatActivity;
import android.view.MenuItem;

import com.example.roman.queorder.mFragments.CardFragment;
import com.example.roman.queorder.mFragments.HomeFragment;
import com.example.roman.queorder.mFragments.TicketsFragment;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        BottomNavigationView navigation = (BottomNavigationView) findViewById(R.id.navigation);
        navigation.setOnNavigationItemSelectedListener(navListener);

        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new HomeFragment()).commit();
    }



    private BottomNavigationView.OnNavigationItemSelectedListener navListener
            = new BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {
            Fragment selectedFragment = null;
```

```
                switch (item.getItemId()){
                    case R.id.navigation_home:
                        selectedFragment = new HomeFragment();
                        break;
                    case R.id.navigation_cards:
                        selectedFragment = new CardFragment();
                        break;
                    case R.id.navigation_tickets:
                        selectedFragment = new TicketsFragment();
                        break;

                }

getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                        selectedFragment).commit();

                return true;
            }
        };

    }
```

fragment_card.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/cardsRV"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
```

```
    <android.support.design.widget.FloatingActionButton
        android:id="@+id/floatingActionButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="30dp"
        android:layout_marginBottom="30dp"
        android:clickable="true"
        app:srcCompat="@android:drawable/ic_menu_camera" />
</RelativeLayout>
```

CardFragment.java

```
package com.example.roman.queorder.mFragments;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.roman.queorder.MenuItem;
import com.example.roman.queorder.MyAdapter;
import com.example.roman.queorder.R;

import java.util.ArrayList;

public class CardFragment extends Fragment {

    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
```

```
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

        View rootView = inflater.inflate(R.layout.fragment_cards, container, false);
        ArrayList<MenuItem> menuList = new ArrayList<>();
        menuList.add(new MenuItem(R.drawable.ic_cards, "Card 1"));
        menuList.add(new MenuItem(R.drawable.ic_home, "Card 2"));

        RecyclerView rv = rootView.findViewById(R.id.cardsRV);
        rv.setHasFixedSize(true);
        MyAdapter adapter = new MyAdapter(menuList);
        rv.setAdapter(adapter);

        LinearLayoutManager llm = new LinearLayoutManager(getActivity());
        rv.setLayoutManager(llm);

        return rootView;
    }
}
```

menu_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_margin="10dp"
    card_view:cardCornerRadius="6dp"
    card_view:cardElevation="4dp"
```

```
    android:id="@+id/card_view">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="90dp"
        android:layout_margin="4dp">
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="160dp"
            android:layout_height="90dp"
            android:padding="2dp"
            android:src="@drawable/ic_cards" />
        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_marginStart="5dp"
            android:layout_marginTop="30dp"
            android:layout_toEndOf="@+id/imageView"
            android:text="CardName"
            android:textColor="@android:color/black"
            android:textSize="20sp"
            android:textStyle="bold" />
    </RelativeLayout>
</android.support.v7.widget.CardView>
```

fragment_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent">

<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_margin="10dp"
    card_view:cardCornerRadius="6dp"
    card_view:cardElevation="4dp"
    android:id="@+id/card_view">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="400dp"
            android:layout_alignStart="@+id/main_view"
            android:layout_alignParentTop="true"
            android:layout_margin="4dp">

            <ImageView
                android:id="@+id/homescreenlogo"
                android:layout_width="200dp"
                android:layout_height="200dp"
                android:layout_alignParentTop="true"
                android:layout_centerHorizontal="true"
                android:layout_marginTop="18dp"

                android:padding="2dp"
                android:src="@drawable/racoon_main" />

            <TextView
```

```
            android:id="@+id/LogoName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_marginStart="5dp"
            android:layout_marginTop="238dp"
            android:layout_centerHorizontal="true"
            android:layout_toEndOf="@+id/imageView"
            android:text="@string/app_name"
            android:textColor="@android:color/black"
            android:textSize="20sp"
            android:textStyle="bold"/>

        <TextView
            android:layout_width="297dp"
            android:layout_height="65dp"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true"
            android:layout_marginStart="-190dp"
            android:layout_marginTop="269dp"
            android:layout_toEndOf="@+id/LogoName"
            android:gravity="center"
            android:text="@string/main_info"
            android:textColor="@color/colorPrimary" />

    </RelativeLayout>
</android.support.v7.widget.CardView>

  <android.support.design.widget.FloatingActionButton
    android:id="@+id/floatingActionButton2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="30dp"
        android:layout_marginBottom="30dp"
        android:clickable="true"
        card_view:srcCompat="@android:drawable/ic_menu_camera" />

</RelativeLayout>
```

cardlist.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.v7.widget.CardView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        xmlns:card_view="http://schemas.android.com/apk/res-auto"
        android:layout_margin="10dp"
        card_view:cardCornerRadius="6dp"
        card_view:cardElevation="4dp"
        android:id="@+id/card_view">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="400dp"
            android:layout_alignStart="@+id/main_view"
            android:layout_alignParentTop="true"
```

```
android:layout_margin="4dp">

<ImageView
    android:id="@+id/qr"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="18dp"

    android:padding="2dp"
    android:src="@drawable/qr" />

<TextView
    android:layout_width="58dp"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:layout_marginBottom="135dp"
    android:text="Name"></TextView>

<EditText
    android:id="@+id/etName"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/qr"
    android:layout_marginTop="15dp"
    android:layout_marginEnd="-4dp"
    android:layout_toStartOf="@+id/qr"
    android:layout_weight="1">
```

```
    <requestFocus></requestFocus>
</EditText>

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:layout_marginBottom="88dp"
    android:text="Date"></TextView>

<EditText
    android:id="@+id/etEmail"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/etName"
    android:layout_alignStart="@+id/etName"
    android:layout_weight="1"></EditText>

<Button
    android:id="@+id/btnAdd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="17dp"
    android:layout_toEndOf="@+id/textView2"
    android:text="Add"></Button>

<Button
    android:id="@+id/btnRead"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignTop="@+id/btnAdd"
            android:layout_alignParentEnd="true"
            android:layout_marginEnd="37dp"
            android:text="Read"></Button>

        <Button
            android:id="@+id/btnClear"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignTop="@+id/btnAdd"
            android:layout_centerHorizontal="true"
            android:text="Clear"></Button>

    </RelativeLayout>
  </android.support.v7.widget.CardView>
</RelativeLayout>
```

camera_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@drawable/bg_gradient"
  tools:context="info.androidhive.movietickets.MainActivity">

  <LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_centerHorizontal="true"
    android:layout_centerInParent="true"
    android:gravity="center"
    android:orientation="vertical"
    android:paddingLeft="40dp"
    android:paddingRight="40dp">

    <ImageView
        android:id="@+id/icon"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_centerHorizontal="true"
        android:clickable="true"
        android:foreground="?attr/selectableItemBackground"
        android:src="@drawable/qrcode"
        android:tint="@android:color/white" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:fontFamily="sans-serif-light"
        android:gravity="center"
        android:text="Scan the QR code on the poster and book your movie tickets"
        android:textColor="@android:color/white"
        android:textSize="16dp" />
</LinearLayout>

<Button
    android:id="@+id/btn_scan"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="40dp"
    android:background="@android:color/transparent"
    android:foreground="?attr/selectableItemBackground"
    android:paddingLeft="20dp"
    android:paddingRight="20dp"
    android:fontFamily="sans-serif-medium"
    android:text="Scan QR Code"
    android:textColor="@android:color/white"
    android:textSize="18sp" />

</RelativeLayout>
```

Reader_camera.java

MainActivity.java

```
import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Build;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.SparseArray;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;

import com.google.android.gms.vision.barcode.Barcode;

import java.util.List;

import info.androidhive.barcode.BarcodeReader;
```

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // making toolbar transparent
        transparentToolbar();

        setContentView(R.layout.activity_main);

        findViewById(R.id.btn_scan).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(MainActivity.this, ScanActivity.class));
            }
        });
    }

    private void transparentToolbar() {
        if (Build.VERSION.SDK_INT >= 19 && Build.VERSION.SDK_INT < 21) {
            setWindowFlag(this, WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS,
true);
        }
        if (Build.VERSION.SDK_INT >= 19) {

getWindow().getDecorView().setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_STA
BLE | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
        }
        if (Build.VERSION.SDK_INT >= 21) {
            setWindowFlag(this, WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS,
false);
            getWindow().setStatusBarColor(Color.TRANSPARENT);
```

```
        }
    }

    private void setWindowFlag(Activity activity, final int bits, boolean on) {
        Window win = activity.getWindow();
        WindowManager.LayoutParams winParams = win.getAttributes();
        if (on) {
            winParams.flags |= bits;
        } else {
            winParams.flags &= ~bits;
        }
        win.setAttributes(winParams);
    }
}
```

Додаток Б

Електронні плакати

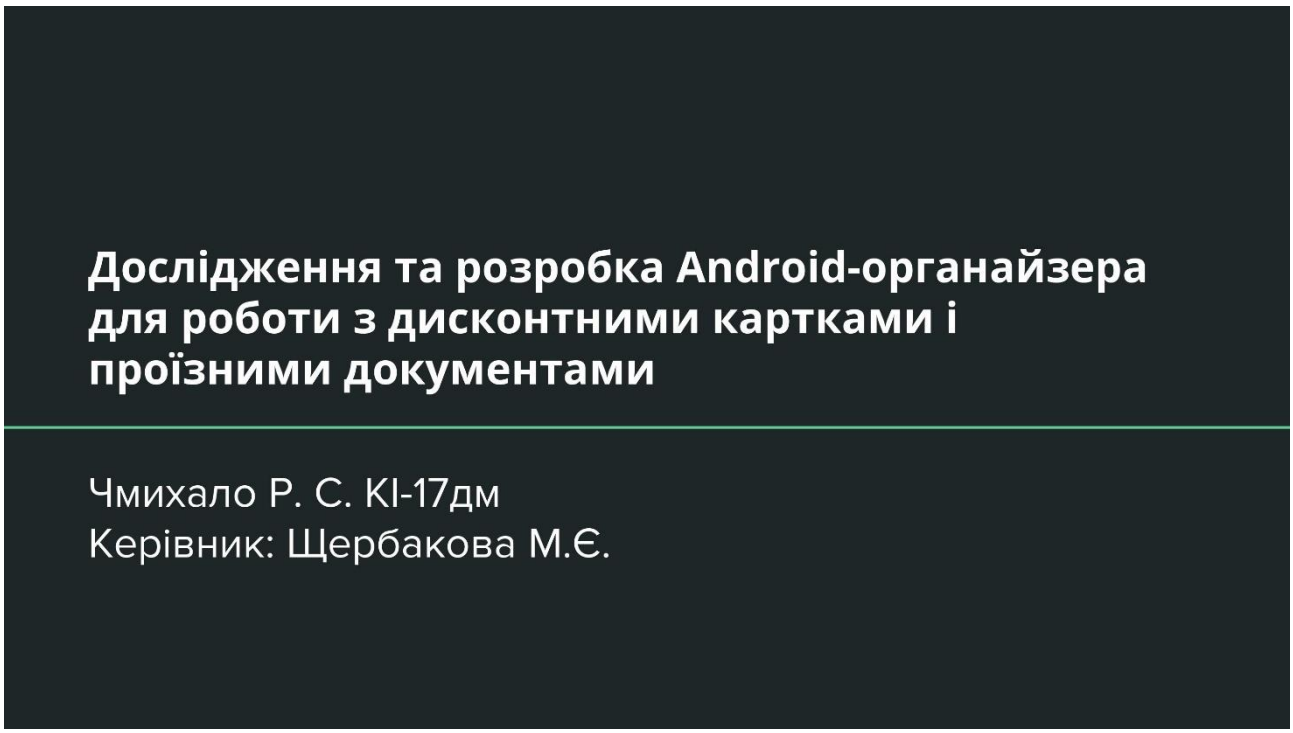


Рисунок Б.1 – Слайд «Титул»

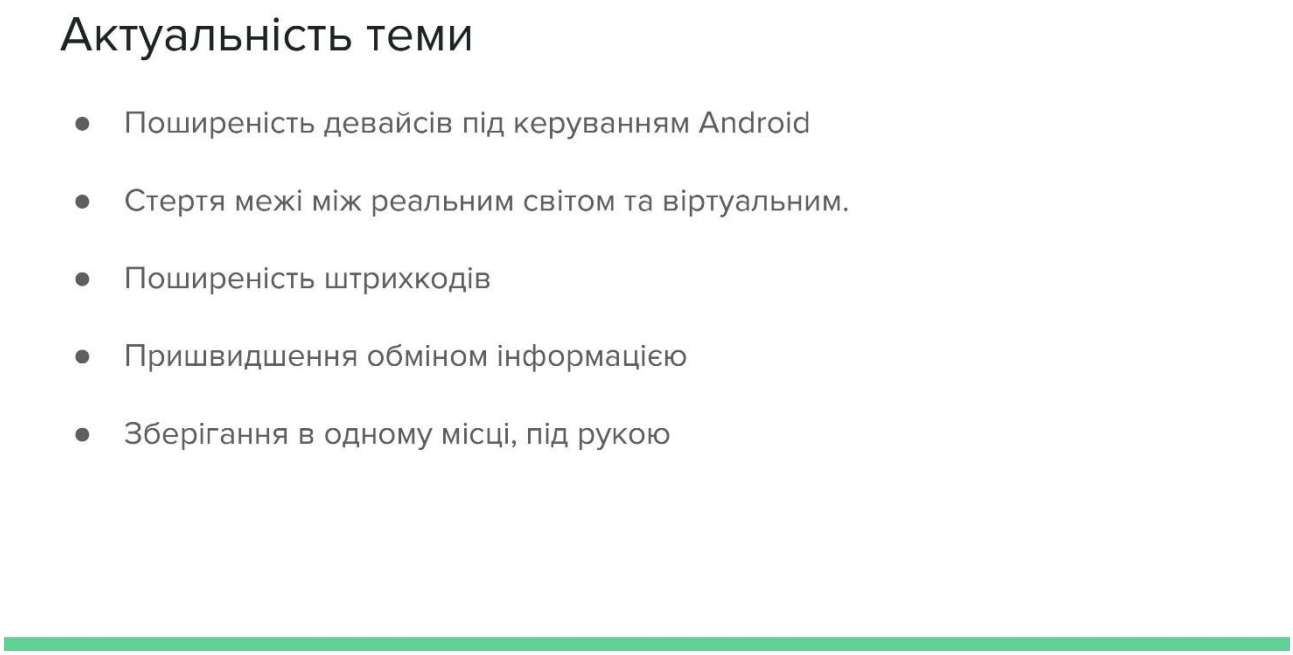


Рисунок Б.2 – Слайд «Актуальність теми»

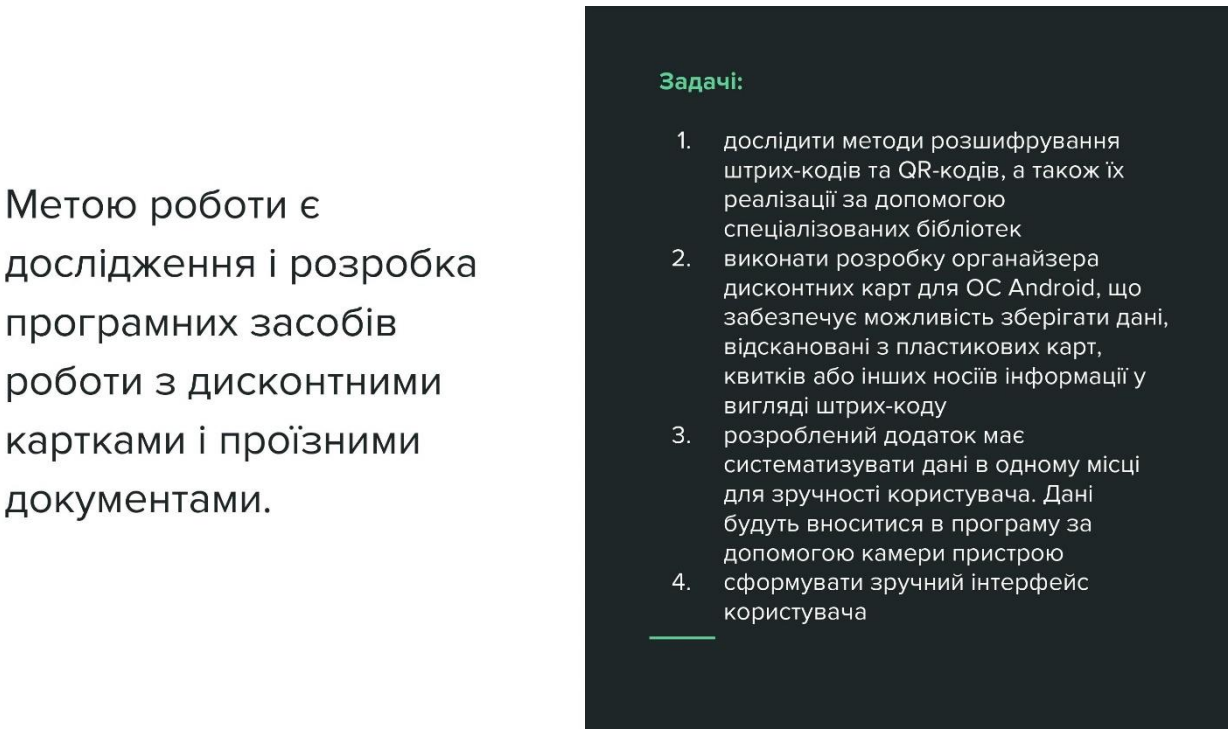


Рисунок Б.3 – Слайд «Мета роботи та задачі»

Рисунок Б.4 – Слайд «Методи дослідження»

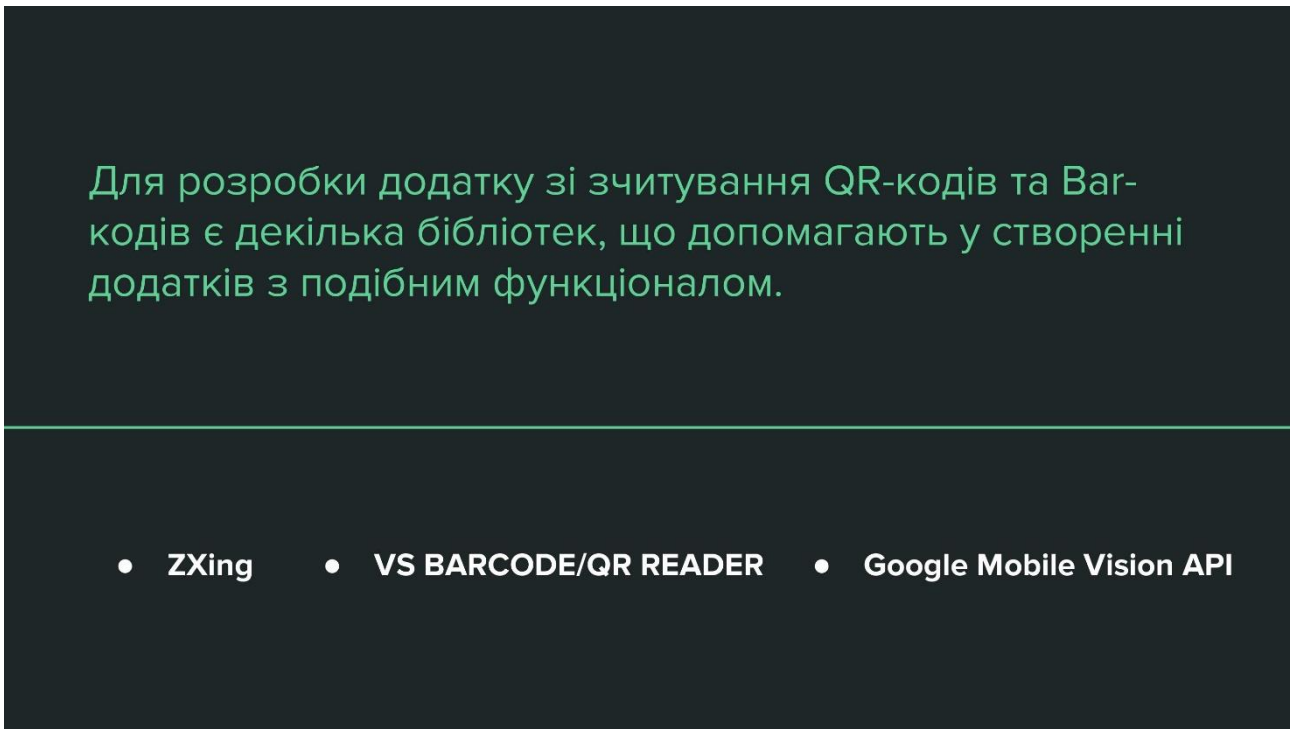


Рисунок Б.5 – Слайд «Методи вирішення поставленої задачі»

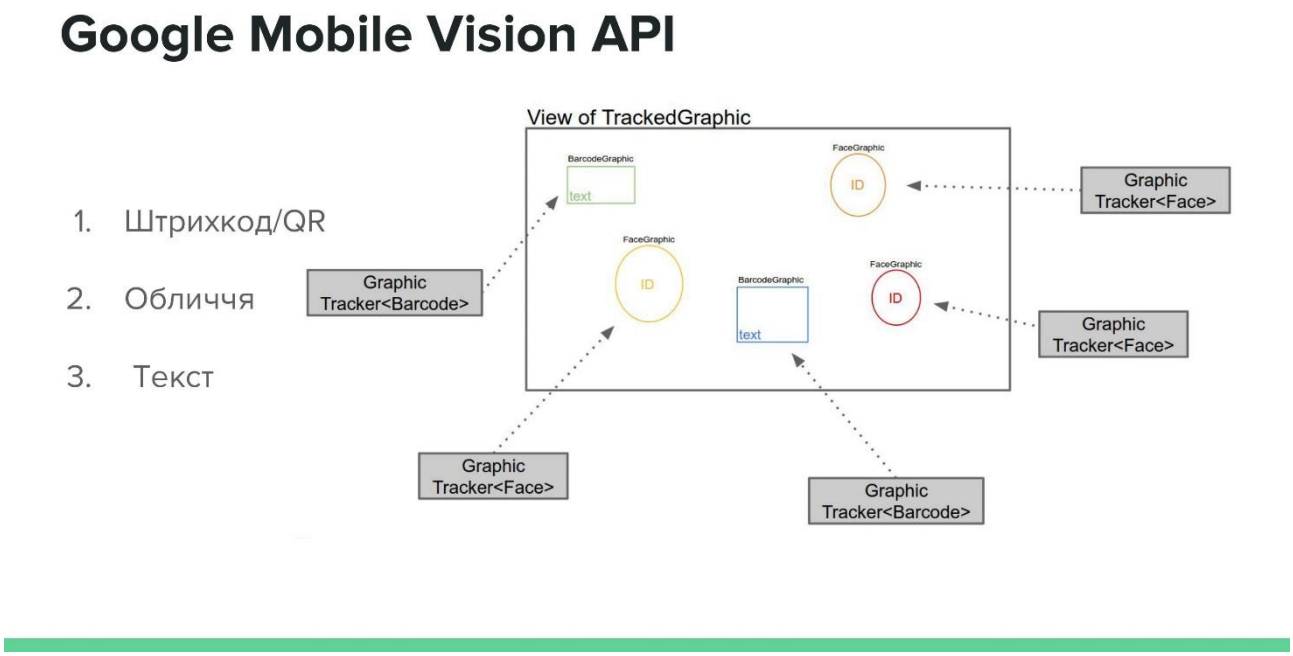


Рисунок Б.6 – Слайд «Google Mobile Vision API»

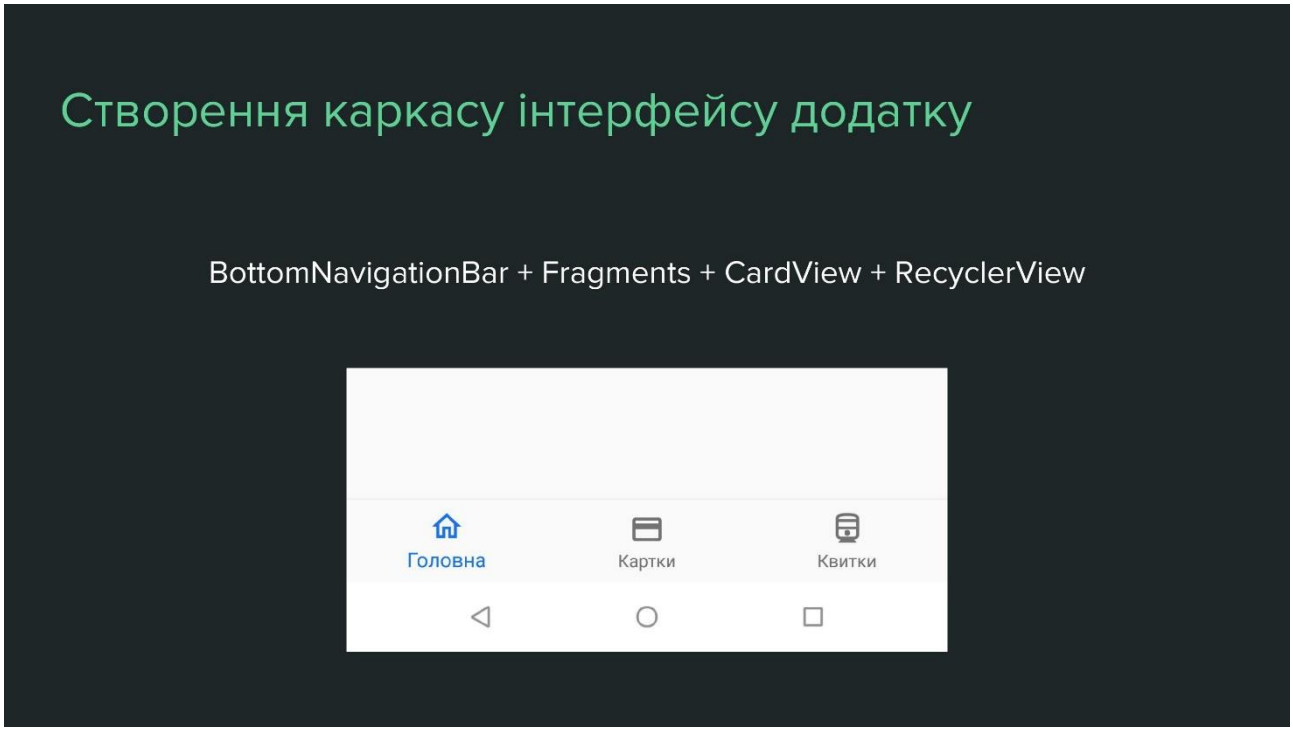


Рисунок Б.7 – Слайд «Створення інтерфейсу»

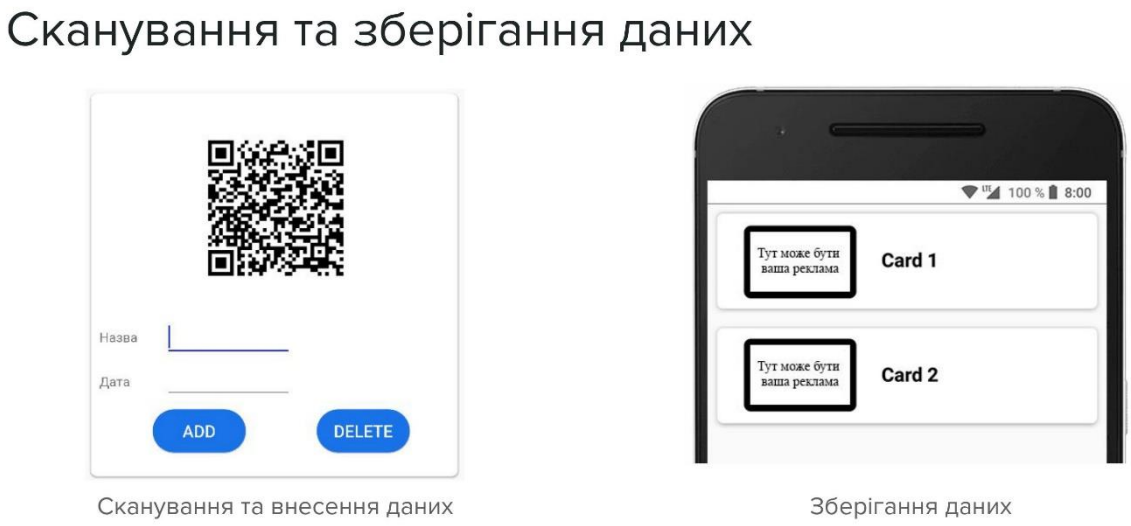
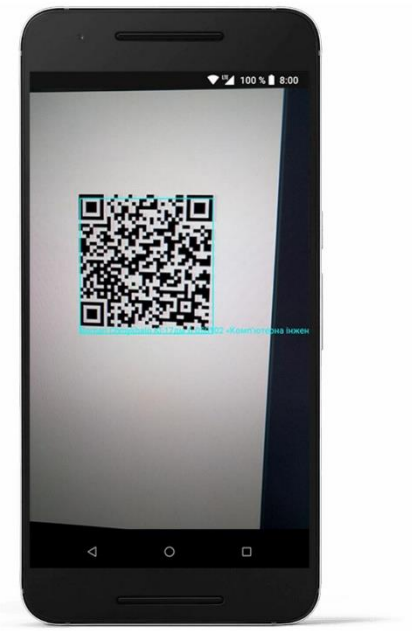


Рисунок Б.8 – Слайд «Сканування та зберігання даних»



Практичне значення одержаних результатів

Органайзер дисконтних карт забезпечує можливість зберігати дані, відскановані з пластикових карт, квитків або інших носіїв інформації у вигляді штрих-коду. Цей додаток систематизує дані в одному місці для зручності користувача. Дані заносяться в програму за допомогою камери пристрою.

Рисунок Б.9 – Слайд «Практичне значення»

Наукова новизна одержаних результатів

Набули подальшого розвитку методи застосування сучасних мобільних технологій, а саме бібліотек мови програмування Java та системи управління базами даних SQLite. Використання новітньої бібліотеки з розпізнавання елементів за допомогою камери дозволило пришвидшити роботу відносно інших додатків, що переважно працюють на старих та громіздких бібліотеках.

Рисунок Б.10 – Слайд «Наукова новизна результатів»

Перспективи розвитку

- Інтеграція із сервісами Google Maps
- Захист даних
- Зберігання даних у хмарних сховищах
- Можливість збереження подій до календаря

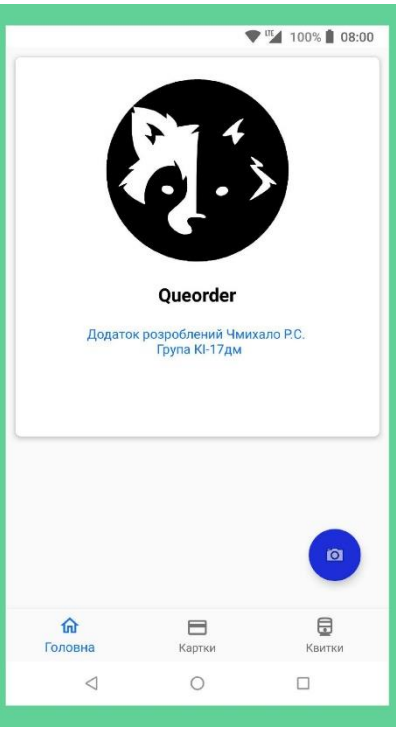


Рисунок Б.11 – Слайд «Перспективи розвитку»

Висновок

- Були досліджені методи створення і виконано розробку додатку для роботи з дисконтними картками і проїзними документами. Додаток може зберігати інформацію про всі дисконтні картки користувача, і може замінити дисконтні картки на фізичних носіях.
- Розглянуті основні положення розшифрування штрих-кодів та QR-кодів, їх реалізації за допомогою спеціалізованих програмних бібліотек.
- Представлений органайзер для мобільних пристроїв на базі ОС Android був розроблений на сучасному рівні, використовуючи такі технології мобільних обчислень, як розпізнавання штрих-кодів за допомогою сторонніх бібліотек, збереження даних, отриманих ззовні.

Рисунок Б.12 – Слайд «Висновок»

Дякую за увагу!

Рисунок Б.13 – Слайд «Фінал»