МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки

На правах рукопису

НАЗАР БОГДАН АНАТОЛІЙОВИЧ

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ РОЗДРІБНОЇ ТОРГІВЛІ

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки та інформаційні технології
Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»

Науковий керівник:
БУЛАТЕЦЬКА ЛЕСЯ ВІТАЛІЇВНА,
кандидат фізико-математичних наук, доцент
кафедри комп'ютерних наук та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №_____
засідання кафедри комп'ютерних наук
та кібербезпеки
від _____ 2024 р.
Завідувач кафедри

(_____) Гришанович Т. О.

ЛУЦЬК 2024

ЗМІСТ

ВСТУП

Актуальність дослідження. Актуальність теми кваліфікаційної роботи зумовлена стрімким зростанням використання мобільних пристроїв у повсякденному житті. Враховуючи те, що електронна комерція стає не тільки популярним способом купівлі товарів і послуг, але й невід'ємною частиною нашого повсякденного життя, створення мобільних застосунків для онлайн магазинів набуває особливого значення. Вони забезпечують зручний та швидкий доступ до широкого асортименту товарів, що робить процес покупок максимально зручним для користувачів. У цьому контексті, розробка та впровадження нових інноваційних рішень у сфері електронної комерції має великий потенціал для успішного ведення бізнесу та задоволення потреб сучасного споживача. Розвиток інформаційних технологій та зростання інтернет-торгівлі відкривають нові можливості для бізнесу, а саме, можливість просування та реалізації товарів у великому обсязі без значних витрат на рекламу та фізичні магазини. Мобільні застосунки для онлайн магазинів дозволяють підприємствам створити зручний інструмент для взаємодії зі споживачами, що сприяє якісній та ефективній роботі бізнесу. Розробка мобільного застосунку для роздрібного магазину є досить актуальною темою, яка відповідає сучасним тенденціям розвитку ринку та технологій, а також відкриває нові можливості для покращення якості обслуговування клієнтів, збільшення обсягу продажів та ефективності бізнесу вцілому.

Мета дослідження. Розробка та впровадженні мобільного застосунку роздрібного магазину, який буде надійним інструментом для здійснення покупок, забезпечуючи зручність, швидкість та безпеку для користувачів.

Основні завдання:

- проведення аналізу сучасних тенденцій у мобільній електронній комерції та вивчення відомих практик в галузі створення мобільних застосунків для роздрібних магазинів;

- дослідження існуючих мобільних додатків для електронної комерції з метою визначення їхніх переваг та недоліків;
- визначення цільової аудиторії для мобільного застосунку та аналіз її вимог до функціональності та інтерфейсу додатку;
- розробка концепції мобільного застосунку, що відповідає потребам цільової аудиторії та враховує сучасні тренди у дизайні та функціональності;
- реалізація програмного рішення мобільного застосунку, забезпечення його стабільної роботи та оптимальної швидкодії.

Об’єкт дослідження – методи розробки мобільних застосунків для роздрібних магазинів.

Предмет дослідження – функціональні можливості та інтерфейс мобільного застосунку для ефективної та зручної покупки товарів онлайн.

Апробація результатів роботи.

Результати роботи були представлені та обговоренні на наступних конференціях:

- XVIII Міжнародна науково-практична конференція студентів, аспірантів та молодих вчених "Молода наука Волині: пріоритети та перспективи досліджень";
- І Міжнародна наукова конференція "Проблеми комп’ютерних наук, програмного моделювання та безпеки цифрових систем".

РОЗДІЛ 1
МОБІЛЬНІ ЗАСТОСУНКИ У РОЗДРІБНІЙ ТОРГІВЛІ

1.1. Поняття мобільного застосунку

Мобільний застосунок – це програмне забезпечення, яке розробляється для використання на мобільних пристроях, таких як смартфони, планшети та розумні годинники. Мобільні застосунки можуть виконувати різні завдання, такі як:

- обмін повідомленнями, здійснення дзвінків, відеозв'язок, соціальні мережі, що дозволяє спілкуватися;
- ігри, відео, музика, книги, журнали, які потрібні для розваг;
- освітні програми, курси, що використовуються у навчанні;
- офісні програми, управління проектами, що допомагає у роботі;
- електронна комерція, реклама, маркетинг, що використовується у веденні бізнесу.

Мобільні додатки можуть бути як вбудованими, так і завантажуваними. Вбудовані додатки вже встановлені на мобільному пристрої при покупці, а завантажувані можна завантажити та встановити з магазину додатків.

Мобільні додатки можуть бути як офлайн, так і онлайн. Офлайн додатки можуть працювати без доступу до Інтернету, а онлайн вимагають доступу до Інтернету для роботи.

Мобільні додатки можуть бути написані на різних програмних мовах, таких як Java, Kotlin, Swift, Objective–C, C++, C# тощо. Для розробки мобільних додатків також використовуються різні середовища розробки, такі як Android Studio, Xcode, Unity тощо [1].

Мобільні застосунки можуть виконувати різні функції. До основних функцій мобільних додатків відносяться:

- зручна взаємодія користувача із застосунком, повинен бути простим та зрозумілим у використанні;

- функції програми повинні відповідати потребам та вимогам користувачів;
- мобільні додатки повинні захищати дані користувачів від несанкціонованого доступу.

1.2. Визначення розробки мобільних застосунків

Розробка мобільних застосунків – це процес створення програмних додатків, які працюють на мобільних пристроях, таких як смартфони та планшети.

Мобільні застосунки можна розробляти для різних платформ, таких як iOS, Android і Windows. Ми зосередимося на розробці для платформ Android. Розробка мобільних застосунків – це складний процес, який вимагає від розробників знань і навичок в різних галузях, таких як програмування, дизайн, тестування та безпека.

Розробка мобільних додатків включає в себе ряд етапів: збір вимог, дизайн, розробка, тестування, розгортання (рис. 1.1.). Збір вимог – це етап, на якому визначаються вимоги до застосунку, який відповідатиме потребам користувачів. Дизайн – це етап, на якому розробляється зовнішній вигляд та інтерфейс користувача застосунку, щоб зробити додаток зручним у використанні та мати хороший вигляд. Розробка – це етап, на якому створюється код застосунку, який у більшості випадків є найскладнішим етапом розробки програмного продукту. Тестування – це етап, на якому перевіряється якість застосунку, який є важливим для того, щоб виявити та усунути помилки до того, як додаток буде випущений у публічний доступ. Розгортання – це етап, на якому застосунок публікується в магазинах додатків для того, щоб користувачі могли отримати доступ до застосунку [2].

Рисунок 1.1. – Процес розробки мобільного додатку

1.3. Визначення операційної системи для мобільних застосунків

Насамперед треба вибрати платформу розробки мобільного додатку, це досягається шляхом визначення цільової аудиторії. Зокрема допоможе короткий огляд графіку відсотків користувачів мобільних пристроїв Android та iOS у всьому світі.

У третьому кварталі 2023 року Android зберіг свою позицію провідної мобільної операційної системи в усьому світі з часткою ринку 70,5 відсотків. Найближчий конкурент Android, iOS від Apple, мав частку ринку в 28,8 відсотків протягом того ж періоду [3](рис. 1.2.).

Порівняльний аналіз аспектів розробки на операційних системах Android та iOS наведений у таблиці А.1, Додаток А.

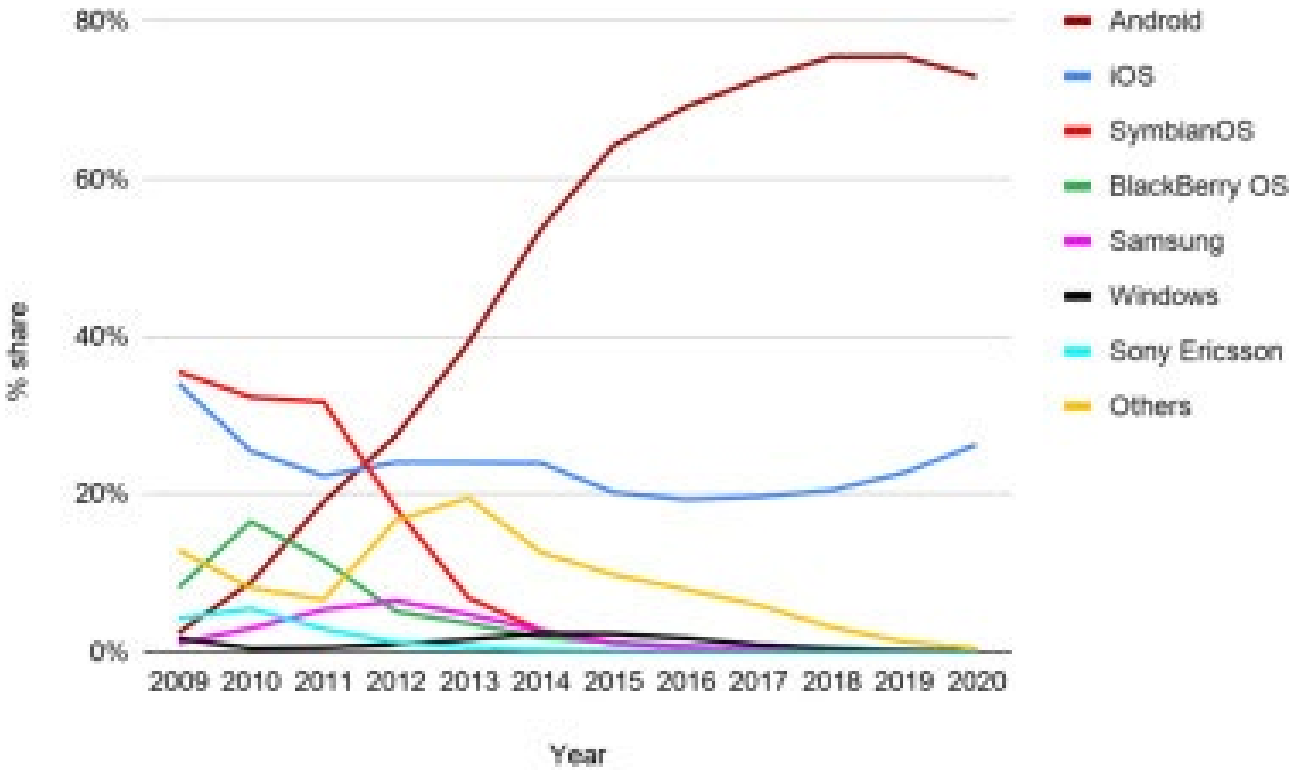

Рисунок 1.2. – Порівняльний аналіз операційних систем

1.4. Особливості розробки мобільного застосунку на Android

Android – це операційна система, розроблена компанією Google для мобільних пристроїв, таких як смартфони, планшети та розумні годинники. Android є однією з найпопулярніших операційних систем у світі, і на її основі розроблено понад 2,8 мільярда активних пристроїв.

Користувачі надають дозволи застосункам на підставі запитів під час встановлення. Ці дозволи можуть охоплювати доступ до камери, мікрофона, контактів, місцезнаходження та інших ресурсів пристрою. Розробники повинні аналізувати та обговорювати, які саме дозволи їм необхідні для ефективної роботи застосунку, а також які дозволи можуть вважатися надлишковими чи небезпечними для конфіденційності користувачів.

Android є платформою для інновацій, що відкриває широкі можливості для створення передових та потужних застосунків завдяки розширеному функціоналу та інтеграції з сучасними технологіями.

1.4.1. Архітектура Android

Android – це модульна система, яка складається з декількох компонентів, таких як ядро Linux, Java Virtual Machine (JVM) та Android Runtime (ART). Ця архітектура дозволяє розробникам створювати застосунки, які є ефективними та масштабованими.

Ядро Linux відповідає за основні функції операційної системи, такі як управління процесами, пам'яттю та пристроями. JVM забезпечує виконання коду Java на пристроях Android. ART – це віртуальна машина, яка компілює код Java в машинний код на момент запуску застосунку [4].

Разом із основними компонентами архітектури Android, які були наведені вище, варто відзначити й інші ключові елементи, що забезпечують роботу системи та застосунків. Серед них:

- набір бібліотек та сервісів Android Framework, які надають розробникам API для взаємодії з різними функціями пристрою, такими як графічний інтерфейс користувача, бази даних, мережеві операції та апаратні можливості;
- Android надає гнучкі інструменти для створення користувацького інтерфейсу, включаючи різноманітні елементи інтерфейсу та можливості їх розташування та взаємодії з ними;
- набір сервісів Google Play Services та API від Google, які надають розробникам доступ до різних функцій, таких як аутентифікація користувача, картографічні сервіси, рекламні можливості та інше.

1.4.2. Підтримка різних пристроїв

Android підтримує широкий спектр пристроїв, включаючи смартфони, планшети, розумні годинники та телевізори. Розробники застосунків повинні враховувати особливості різних пристроїв, щоб забезпечити їхню ефективну роботу.

Однією з проблем, з якою стикаються розробники застосунків на Android – різноманітність розмірів екранів мобільних пристроїв. Щоб забезпечити

зручний інтерфейс користувача на всіх пристроях, розробники повинні використовувати адаптивний дизайн. Адаптивний дизайн дозволяє застосунку автоматично підлаштовуватися під розмір екрану пристрою.

Додатковою проблемою, з якою зіштовхуються розробники, є велика різноманітність версій операційної системи Android, що використовується на різних пристроях. Відмінності у версіях можуть включати різні API, SDK, функціональність та підтримку апаратного забезпечення. Тому важливо враховувати ці особливості при розробці застосунків і використовувати можливості, які надаються новішими версіями платформи Android, забезпечуючи при цьому сумісність зі старішими версіями API та SDK.

Додатковим аспектом підтримки різних пристроїв є розуміння різних характеристик апаратного забезпечення: процесорів, оперативної пам’яті та особливостей камер. Оптимізація роботи застосунку під конкретний пристрій вимагає використання різних алгоритмів обробки даних або графіки, щоб забезпечити плавну роботу та найкращу якість зображення на кожному пристрої.

1.4.3. Безпека

Мобільні застосунки можуть містити конфіденційну інформацію про користувачів, тому вони повинні бути захищені від несанкціонованого доступу. Розробники застосунків повинні використовувати різні методи для забезпечення безпеки своїх застосунків, такі як аутентифікація, шифрування та контроль доступу.

Загальний підхід до безпеки на платформі Android полягає в поєднанні різних заходів безпеки, включаючи захист від аутентифікації і шифрування даних до правильного контролю доступу і підтримки через оновлення. Розробники забезпечують захист від потенційних загроз і приділяють велику увагу безпеці під час всього життєвого циклу застосунку, від розробки до підтримки та після його випуску. Ключові аспекти безпеки на платформі Android полягають в забезпеченні правильної аутентифікації користувачів, шифруванні конфіденційної інформації, встановленні правильного контролю доступу до

функцій і ресурсів підтримці застосунку через регулярні оновлення і вчасну виправку виявлених уразливостей. Забезпечення правильної аутентифікації користувачів перед наданням доступу до конфіденційної інформації є важливим кроком у створенні безпеки. Розробники Android використовують різні методи аутентифікації, такі як паролі, відбитки пальців, розпізнавання обличчя або двофакторну аутентифікацію, для захисту від несанкціонованого доступу. Шифрування конфіденційної інформації, яка зберігається на пристрої або передається через мережу, є необхідною технологією для запобігання доступу до даних зловмисників. Розробники Android використовують сучасні алгоритми шифрування та правильно налаштовують їх параметри для максимального захисту інформації. Встановлення правильного контролю доступу до функцій і ресурсів застосунку допомогає уникнути зловживань використання застосунку. Розробники Android обмежують доступ до чутливих операцій і даних тільки авторизованим користувачам та використовувати механізми дозволів Android для цього. Підтримка застосунку через регулярні оновлення і вчасну виправку виявлених уразливостей є критично важливою для забезпечення безпеки. Розробники Android слідкують за випусками оновлень платформи Android та сторонніх бібліотек, які вони використовують, а також кожного разу вносять необхідні зміни для виправлення виявлених уразливостей.

1.5. Життєвий цикл Activity в розробці мобільних Android застосунків

Дія (activity) – це основна одиниця візуального інтерфейсу користувача в Android. Це базовий елемент користувацького інтерфейсу, який представляє собою один екран із додатку. Життєвий цикл Activity – це послідовність станів, через які проходить Activity під час свого існування.

Кожного разу, під час запуску нової дії, попередня зупиняється і система зберігає її у стеку, а нова додається зверху даного стеку і отримує фокус користувача. Стек відповідає базовому механізму "перший увійшов – перший

вийшов", тобто коли користувач завершив роботу з поточною дією і натискає кнопку "назад", вона витягується зі стеку і попередня дія продовжує свою роботу. Коли дія зупиняється через створення нової, вона змінює свій стан. Ці зміни станів є частиною життєвого циклу дії.

Життєвий цикл Activity представлено на рисунку 1.3 та складається з наступних стадій:

- створення (onCreate);
- відображення (onStart);
- поновлення (onResume);
- призупинення (onPause);
- завершення (onStop);
- знищення (onDestroy).

Створення (onCreate) – ця стадія запускається, коли Activity створюється вперше. Під час цієї стадії Activity може ініціалізувати свої ресурси, такі як компоненти UI та дані.

Відображення (onStart) – ця стадія запускається, коли Activity стане видимим для користувача. Під час цієї стадії Activity може завантажити свої ресурси, такі як дані з мережі.

Поновлення (onResume) – ця стадія запускається, коли Activity отримує фокус користувача. Під час цієї стадії Activity може відновити свій стан, якщо він був призупинений.

Призупинення (onPause) – ця стадія запускається, коли Activity втрачає фокус користувача. Під час цієї стадії Activity може зберегти свій стан, якщо він може бути призупинений.

Завершення (onStop) – ця стадія запускається, коли Activity більше не видно для користувача. Під час цієї стадії Activity може звільнити свої ресурси, які не потрібні, коли Activity не видно.

Знищення (onDestroy) – ця стадія запускається, коли Activity знищується. Під час цієї стадії Activity повинен звільнити всі свої ресурси.

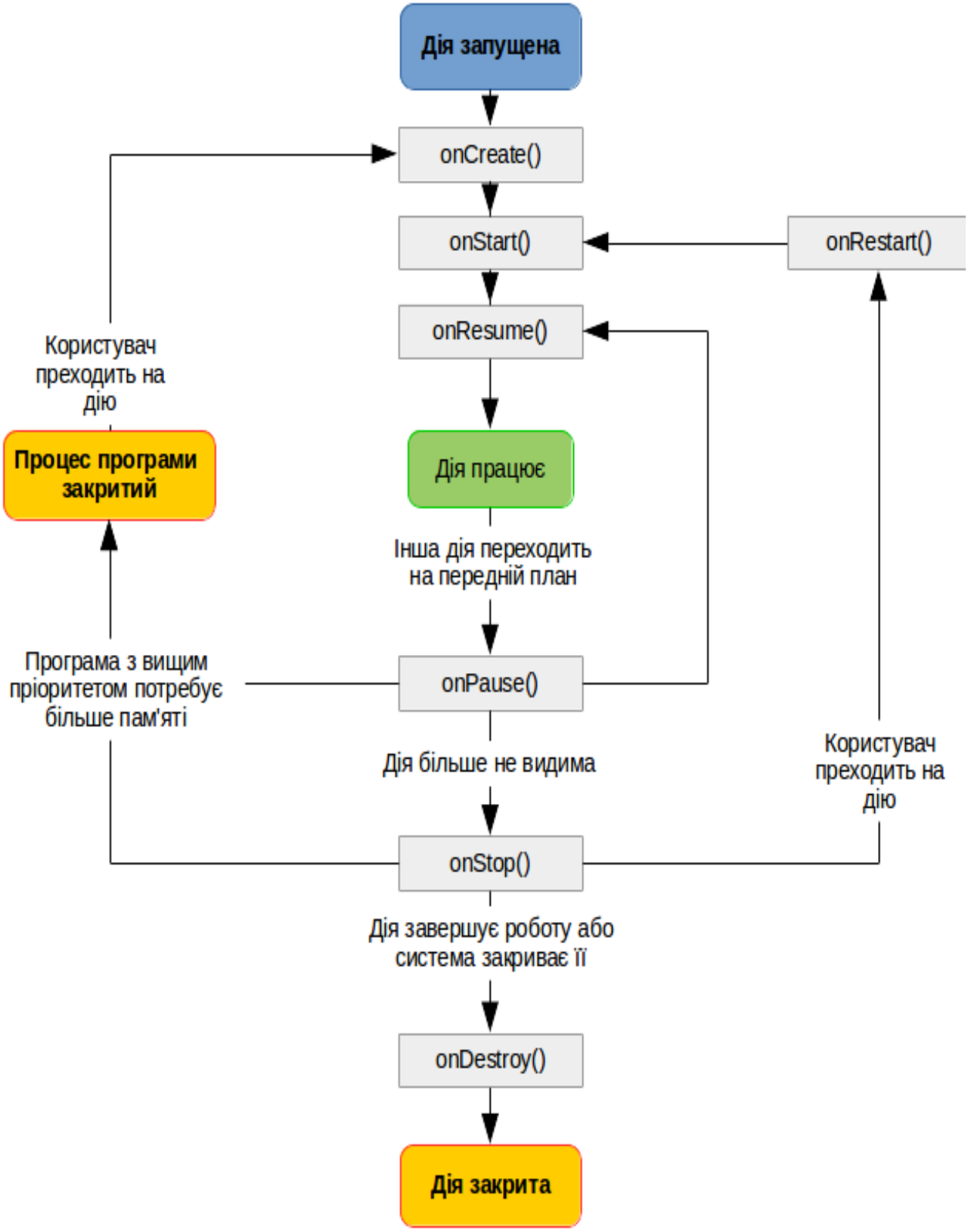

Рисунок 1.3. – Життєвий цикл Activity

Activity допомагає ефективно керувати ресурсами, зберігати стан додатку і правильно реагувати на події, які відбуваються під час взаємодії з користувачем чи системою Android [5].

1.6. Вплив мобільних технологій на розвиток роздрібної торгівлі

Технологічні інновації в роздрібній торгівлі відіграють величезну роль у досягненні більших продажів та полегшенні процесу обслуговування споживачів. Широкий спектр інноваційних рішень, що охоплюють мобільні додатки, системи бездротового зв'язку, аналітичні платформи та інші, надають роздрібним підприємствам можливість адаптуватися до змінних умов ринку та впроваджувати стратегії, спрямовані на залучення більшої кількості клієнтів.

Успішне впровадженнятехнологій вимагає врахування специфіки кожного конкретного бізнесу. Наприклад, для магазинів з великим потоком клієнтів може бути корисним зробити програму якомога зручною та простою, що дозволить здійснювати швидкі та зручні транзакції. У той же час, роздрібні магазини, які спеціалізуються на дорогих та розкішних товарах, можуть сконцентруватися на розробці мобільних додатків з елегантним та привабливим дизайном, спрямованих на підвищення престижності бренду та створення ексклюзивного досвіду для клієнтів.

Також, невід'ємною частиною мобільних застосунків є те, що аналітичні системи дозволяють роздрібним магазинам збирати та аналізувати дані про покупки клієнтів, що дозволяє індивідуалізувати пропозиції та створювати персоналізовані промо-акції, що відповідають унікальним потребам кожного клієнта.

Для досягнення успіху в динамічному та конкурентному середовищі роздрібної торгівлі, власники магазинів повинні постійно прагнути до інновацій та вдосконалення. Впровадження новітніх технологій не тільки дозволяє адаптуватися до змін у ринкових умовах, але також відкриває нові можливості для приваблення нової цільової аудиторії.

1.7. Можливості використання мобільних застосунків для роздрібної торгівлі

Мобільні застосунки надають підприємствам унікальні можливості для покращення взаємодії з клієнтами, оптимізації процесів продажу та підвищення конкурентоспроможності на ринку. Однією з ключових можливостей використання мобільних застосунків є забезпечення клієнтам зручного та швидкого доступу до асортименту товарів і послуг, що дозволяє їм здійснювати покупки в будь-який зручний для них час та в будь-якому місці.

Також, мобільні застосунки є потужним інструментом для персоналізації обслуговування клієнтів. Шляхом збору та аналізу даних про покупки та поведінку клієнтів, додатки можуть надавати індивідуалізовані рекомендації, спеціальні пропозиції та знижки, що збільшують лояльність споживачів та підвищують їхнє задоволення від покупок.

Крім того, мобільні застосунки можуть бути використані для здійснення різноманітних маркетингових заходів, таких як розсилання сповіщень про акції та розпродажі, організація програм лояльності та участь у програмах реферального маркетингу, що є невід'ємною частиною сучасного маркетингу. Ці заходи допомагають роздрібним магазинам привертати увагу клієнтів, стимулювати продажі та підвищувати обсяги прибутку.

1.8. Огляд та аналіз існуючих мобільних застосунків для роздрібної торгівлі

Сільпо [6]:

- назва – Сільпо;
- розробник – SILPO-FUB, TOV;
- архітектура – мобільні застосунки для Andoroid та iOS;
- мова реалізації – Kotlin (Android) та Swift (iOS).

Перелік функцій:

- можливість перегляду акцій та спеціальних пропозицій;
- зручне керування корзиною покупок;
- можливість перегляду та редагування особистого профілю;
- інтеграція з системою онлайн-оплати для зручності оплати товарів;
- можливість контакту та зворотнього зв’язку зі службою підтримки;
- можливість сканування продуктів за допомогою штрихкоду.

Аналіз переваг та недоліків:

Переваги:

- доступ до актуальних акцій та знижок, що економить час та гроші;
- швидка та ефективна робота додатку;
- проста система оформлення замовлення;
- створення персональних пропозицій та рекомендацій на основі історії покупок;
- надання детальної інформації про продукти, магазини та їхні послуги.

Недоліки:

- деякі функції, такі як доставка, можуть бути доступні не у всіх регіонах;
- незручний інтерфейс користувача;
- для користування деякими функціями потрібна реєстрація та авторизація;
- можливість виникнення затримок при оновленні акцій та пропозицій.

Загальний вигляд застосунку знаходиться на рис. 1.5. та 1.6.

Glovo [7]:

- назва – Glovo;
- розробник – Glovoapp 23 SL;
- архітектура – мобільні застосунки для Andoroid та iOS;
- мова реалізації – Kotlin (Android) та Swift (iOS).

Перелік функцій:

- замовлення та доставка їжі з різних ресторанів і кафе;
- доставка продуктів з супермаркетів;
- можливість відстеження статусу замовлення в реальному часі;

- оплата замовлення онлайн або готівкою при отриманні;
- можливість залишити відгуки та оцінки, як товарам так і кур’єру.

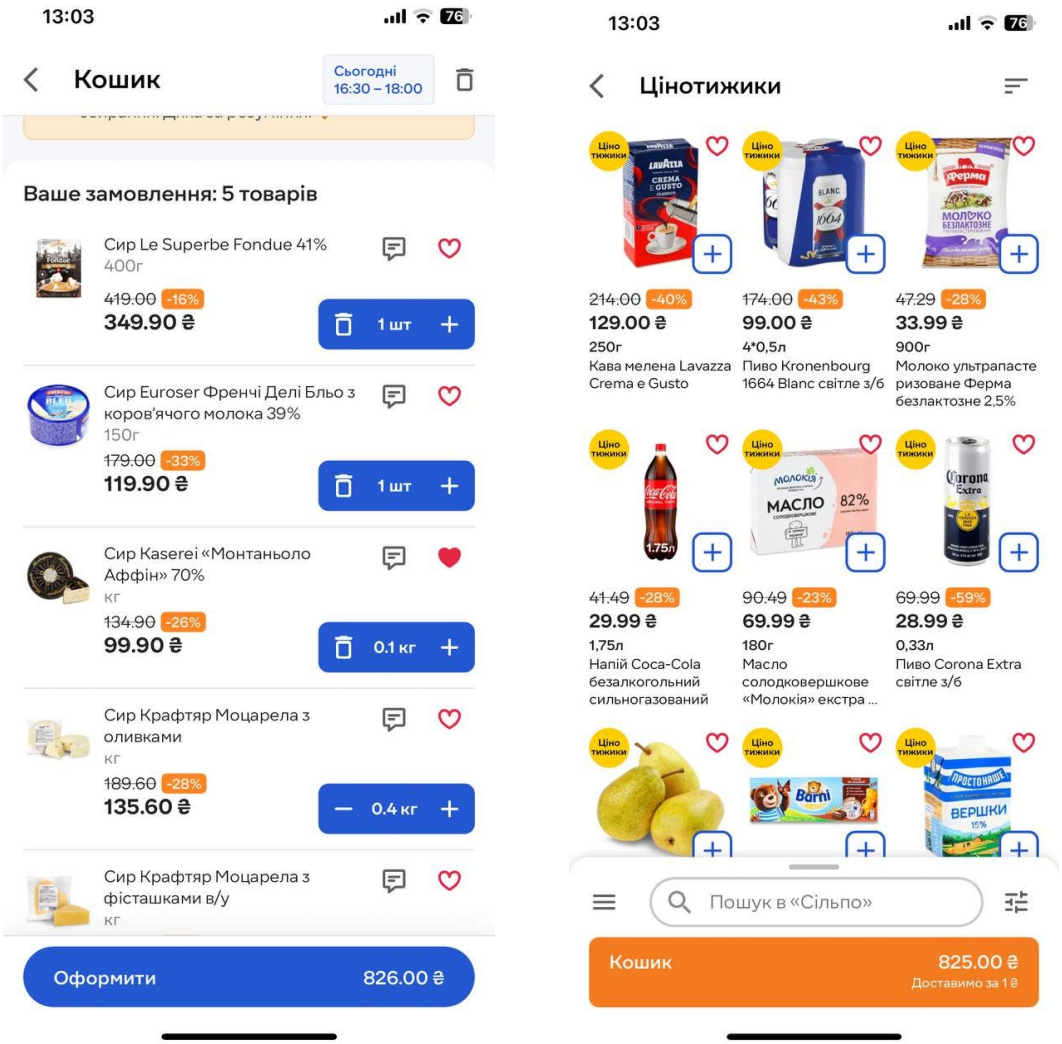

Рисунок 1.4. – Екран додатку Сільпо

Аналіз переваг та недоліків:

Переваги:

- широкий вибір ресторанів і супермаркетів для замовлення;
- інтеграція з різними системами оплати;
- зручний процесоформлення замовлення;
- вибір продуктів за категоріями, за пошуком, за магазинами та за ресторанами;

- деякі послуги та продукти доступні цілодобово у окремих ресторанах, магазинах.

Недоліки:

- відображення продуктів в додатку працює з деякими відхиленнями та збоями;
- відсутність інформації про продукт, доступне тільки зображення;
- для користування деякими функціями потрібна реєстрація та авторизація;
- якість їжі може відрізнятися залежно від ресторану.

Загальний вигляд застосунку знаходиться на рис. 1.5.

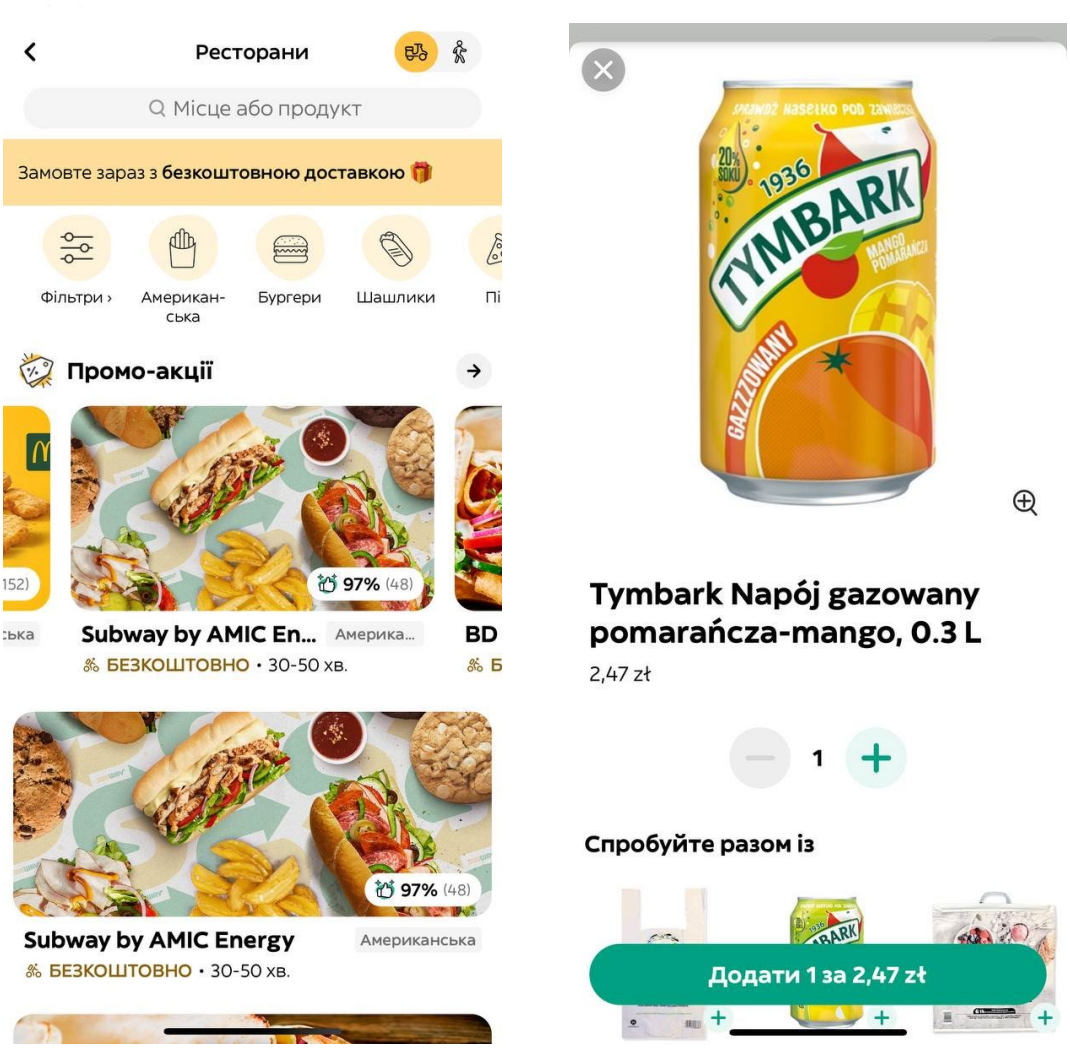

Рисунок 1.5. – Екран додатку Glovo

РОЗДІЛ 2
РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ "nAzars" ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ РОЗДРІБНОЇ ТОРГІВЛІ

2.1. Постановка задачі, призначення та вимоги до програмного засобу "nAzars"

Метою роботи, було створення мобільного застосунку для оптимізації процесів у роздрібному магазині. Мобільний застосунок для роздрібного магазину повинен бути призначений для широкого кола користувачів, які мають різні потреби та цілі. Застосунок повинен дозволяти користувачам переглядати усі товари, що наявні в магазинах мережі, усі доступні акції, робити замовлення продуктів за адресою, дивитися інформацію про усі фізичні магазини мережі на мапі, а також переглядати свої минулі замовлення та всю детальну іфномрацію про них.

При розробці застосунку буде потрібно враховувати наступне:

- застосунок повинен бути простим у використанні, для того щоб користувачі легко могли зрозуміти, як користуватися застосунком;
- застосунок повинен бути інтуїтивно зрозумілим, для того щоб користувачі мали можливість швидко знайти потрібну інформацію або функцію;
- застосунок повинен правильно працювати на різних мобільних пристроях Android.

Застосунок "nAzars" призначений для полегшення та зручності покупок клієнтам роздрібних магазинів. Він призначений для надання клієнтам доступу до повної та актуальної інформації про товари, їх ціни, знижки та наявність. Цей застосунок дозволить клієнтам швидко та зручно знаходити потрібні товари, переглядати їх характеристики та ціни, що сприятиме зручності та ефективності у процесі покупок.

Вимоги:

- надійна та коректна робота застосунку без збоїв та відмов;
- функціонал для ведення обліку товарів, включаючи додавання нових товарів, видалення старих та оновлення інформації про наявні товари;
- перегляд користувачами усіх актуальних товарів у магазині, а також знижки;
- перегляд мережі магазинів "nAzars" на мапі;
- перегляд усіх замовлень користувача;
- забезпечення конфіденційності даних користувачів застосунку.

Технічне завдання подано у додатку Б.

2.2. Вибір моделі розробки програмного засобу "nAzars"

Для розробки мобільного застосунку було обрано ітеративну модель, яка відповідає принципам методології Agile (рис. 2.1).

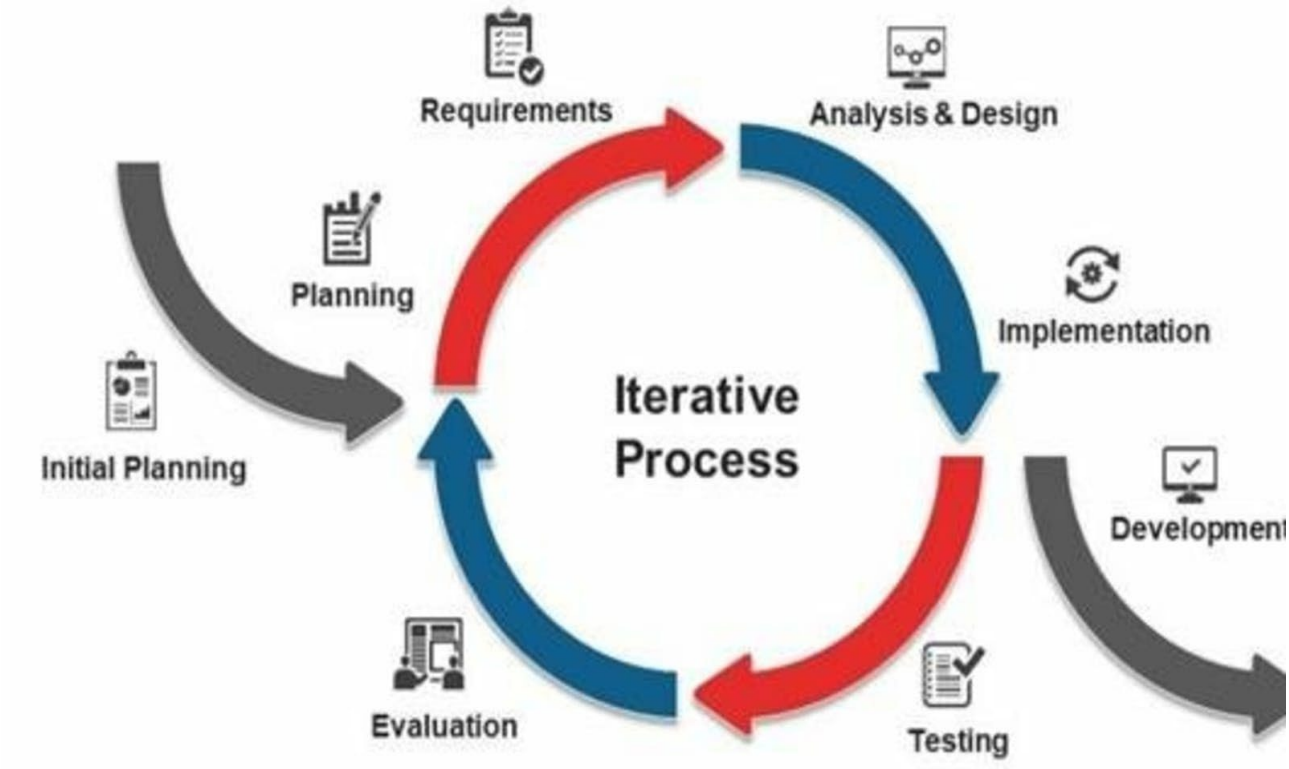

Рисунок 2.1. – Схема ітеративної моделі розробки

Ітеративна модель розробки програмного забезпечення – це модель, яка передбачає розробку програмного засобу послідовно, в циклі "програмування–тестування–тестування". Кожна ітерація закінчується випуском готової до використання версії програми. Методологія Agile сприяє гнучкості та адаптивності у розробці, дозволяючи команді ефективно взаємодіяти з замовником, швидко реагувати на зміни вимог та вдосконалювати продукт на кожному етапі розробки.

Ітеративна модель розробки є поширеним вибором для розробки невеликих програм, які не мають складних вимог. Вона також може бути використана для розробки великих програм, але в цьому випадку цикли ітерації зазвичай довші [8].

2.3. Загальний опис проєкту

Для створення загальної концепції застосунку розроблено діаграму сценаріїв використання програми, посилаючись на вимоги та призначення (рис. 2.2).

Для створення такого додатку потрібно розробити інтерфейси для всіх потрібних активностей та діалогів програми.

Основини складовими програми мають бути:

- головний екран застосунку, тобто екран з усіма доступними товарами;
- активність з повною інформацією про конкретний продукт (після натиснення на відповідну картку товару);
- активність корзини, яка може бути або пустою, або з доданими товарами;
- активність, у якій користувач може змінювати свої персональні дані;
- діалогові вікна з усіма правилами користування додатком;
- активність пошуку продуктів;
- активність перегляду минулих замовлень користувача;
- активність мапи;

- активність налаштувань.

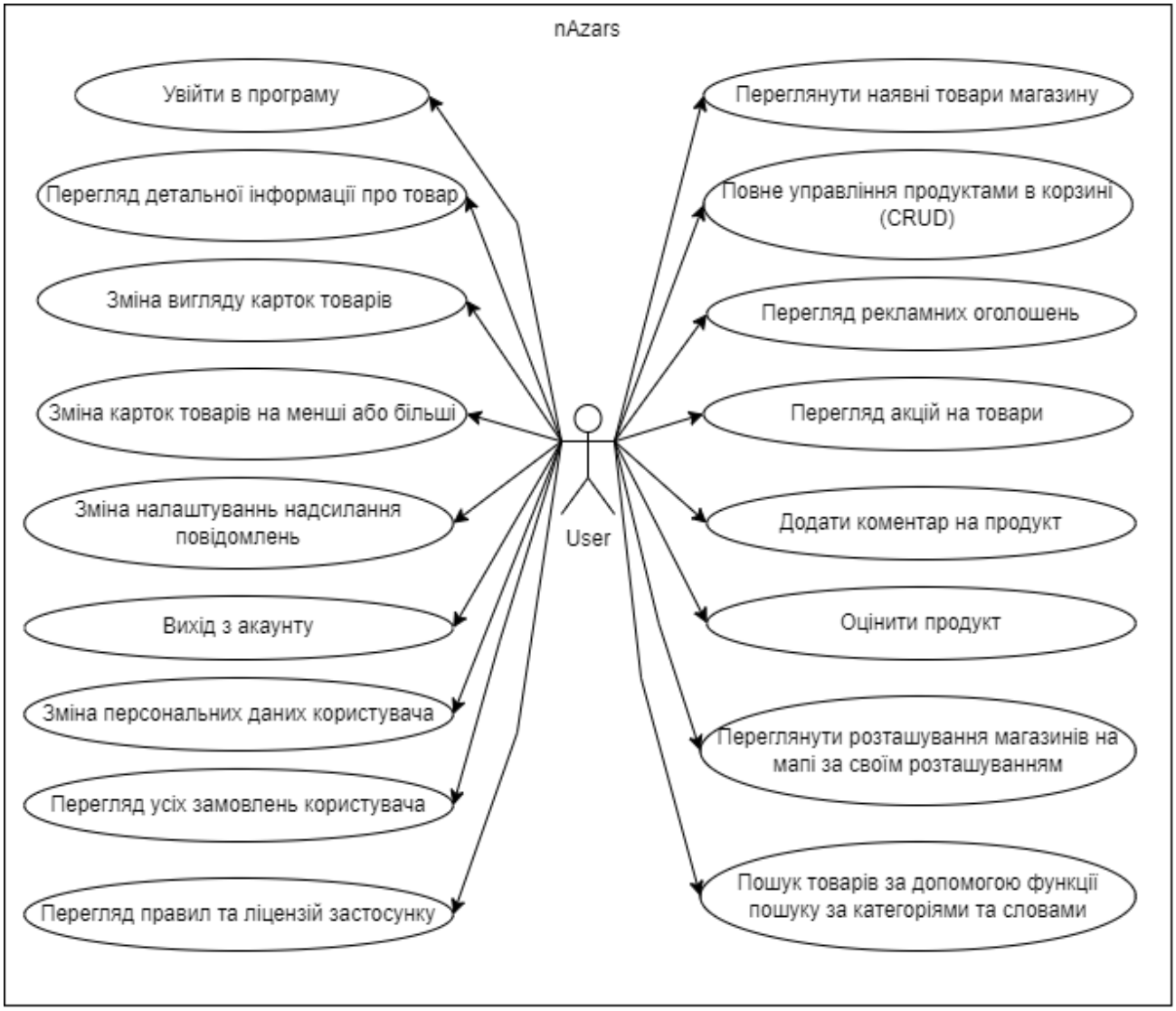

Рисунок 2.2. – Діаграма сценаріїв використання застосунку

Структура бази даних Firebase Realtime Database дозволяє зберігати та управляти інформацією про продукти, магазини, відгуки користувачів та користувачів і включає наступні елементи:

- Banner;
- GroceryItems;
- Reviews;
- Stores;
- Users.

Banner – об’єкт, що містить посилання на зображення банера для реклами або пропозицій.

GroceryItems – масив об’єктів, кожен з яких представляє продукт в магазині. Кожен продукт містить наступні поля:

- availableAmount – доступна кількість товару;
- category – категорія товару;
- description – опис товару;
- imageUrl – посилання на зображення товару;
- name – назва товару;
- price – ціна товару;
- salePrice – знижена ціна товару під час розпродажу;
- popularityPoint – оцінка популярності товару.

Reviews – об’єкт, що містить відгуки користувачів про певні продукти. Кожен відгук має унікальний ідентифікатор та містить наступні поля:

- date – дата відгуку;
- groceryItemId – ідентифікатор продукту, що оцінюється;
- text – текст відгуку;
- userName – ім’я користувача, що залишив відгук.

Stores – масив об’єктів, кожен з яких представляє магазин. Кожен магазин має такі поля:

- latitude – широта магазину;
- longitude – довгота магазину;
- workingHoursFromMonToFri – години роботи з понеділка по п’ятницю;
- workingHoursSat – години роботи у суботу;
- workingHoursSun – години роботи у неділю.

Users – об’єкт, що містить інформацію про користувачів. Кожен користувач має такі поля:

- displayName – відображуване ім’я користувача;
- email – електронна адреса користувача;
- userId – ідентифікатор користувача.

Локальна база даних містить DAO елементи.

Dao (Data Access Object) – це об’єктний шаблон проектування, який надає абстракцію для доступу до бази даних. У контексті Room Persistence Library для Android, інтерфейси Dao визначають методи, які дозволяють виконувати операції з базою даних, такі як вставка, вибірка, оновлення та видалення даних.

Кожен інтерфейс Dao містить анотації для визначення SQL-запитів та інших операцій, що виконуються над базою даних. Наприклад, анотація @Query використовується для визначення SELECT-запитів, а @Insert, @Update та @Delete - для відповідних операцій вставки, оновлення та видалення даних.

Основна мета інтерфейсів Dao – надати зручний і чистий спосіб взаємодії з базою даних, що дозволяє розробнику використовувати зручні методи для роботи з даними в додатку.

DAO елементи локальної бази даних застосунку:

- таблиця grocery_items, що містить інформацію про продукти в магазині, такі як назва, опис, URL зображення, категорія, ціна, доступна кількість, рейтинг, кількість оглядів, популярність тощо;
- таблиця cart_items, яка зберігає ідентифікатори продуктів, які додані до кошика користувача;
- таблиця order_items, що зберігає інформацію про замовлення, включаючи ідентифікатори продуктів, кількість, дату тощо;
- таблиця shop_items, яка містить інформацію про магазини, їхнє розташування та робочі години.

2.4. Обґрунтування вибору інструментальних засобів розробки "nAzars"

Для розробки мобільних додатків існує безліч інструментальних засобів, які відрізняються за своїми характеристиками та можливостями. При виборі інструментальних засобів необхідно враховувати тип операційної системи, вибір

мови програмування, інструмент для створення дизайну та версію архітектури мобільного телефону.

Для розробки мобільного застосунку для роздрібного магазину було використано такі інструментальні засоби:

- середовище розробки Android Studio;
- мова програмування Java;
- інструмент для створення іконок Figma;
- інструмент для керування версіями Git;
- технологія для керуванням застосунком Firebase;
- локальна мобільна база даних Room Database.

Android пропонує широкий спектр інструментів та бібліотек для створення інтерфейсу користувача (UI). Найпопулярніші інструменти:

- Android Studio – це інтегроване середовище розробки (IDE), яка надає все необхідне для створення застосунків на Android;
- Android Layout Editor – це інструмент для створення макетів UI;
- Material Design – це набір компонентів і принципів дизайну, які можна використовувати для створення зручного та інтуїтивно зрозумілого інтерфейсу користувача [9].

Для створення мобільного додатку було обрано середовище розробки Android Studio – це інтегроване (IDE) для розробки мобільних застосунків для платформи Android. Воно надає розробникам широкий спектр інструментів та функцій, які допомагають створювати ефективні та якісні застосунки (рис. 2.3).

Android Studio пропонує вбудований Android Emulator, який дозволяє розробникам тестувати свої застосунки на різних пристроях та версіях Android.

Android Studio пропонує вбудований інструментарій для відстеження помилок, який допомагає розробникам знайти і виправити помилки в своїх застосунках [10].

Android Studio є потужним інструментом, який надає всі необхідні функції для розробки мобільних застосунків. Для створення інтерфейсу програми було використано усі рішення, які надає інтегроване середовище розробки Android

Studio. Android Layout Editor – це простий у використанні інструмент, який дозволяє швидко створювати макети UI. Material Design – це набор принципів дизайну, які можна використовувати для створення естетично привабливого та інтуїтивно зрозумілого інтерфейсу користувача.

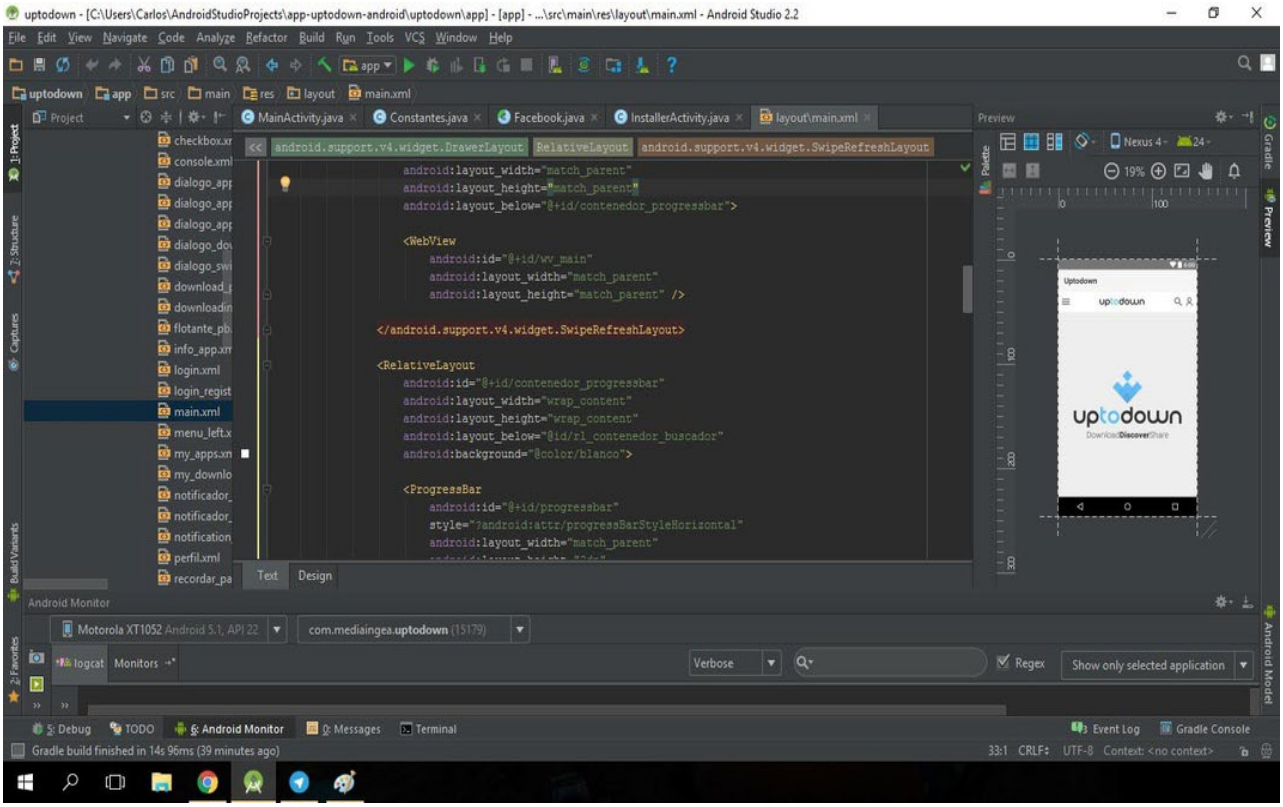

Рисунок 2.3. – Середовище розробки Android Studio

В якості мови програмування було обрано мову Java, так як вона є офіційною мовою програмування для Android. Java є об'єктно-орієнтованою мовою програмування, яка є однією з найпопулярніших мов програмування для розробки мобільних додатків на платформі Android [11]. Серед переваг Java основними є такі:

- Java є потужною та гнучкою мовою програмування, яка дозволяє створювати складні та функціональні додатки;
- Java є мобільною мовою програмування, яка дозволяє розробникам створювати додатки, які можна запускати на різних платформах, включаючи Android;

- Java має велику кількість уже готових бібліотек, які можна використати у створенні мобільного застосунку.

Також, альтернативою для Java є Kotlin , який ув створений в 2017 році та швидко набув значної популярності серед розробників мобільних додатків. Ця мова програмування, розроблена компанією JetBrains, відрізняється зручним синтаксисом, який сприяє поліпшенню продуктивності розробки та зменшенню кількості помилок завдяки виразним конструкціям та підтримці нульових значень.

Крім того, Kotlin пропонує високий рівень інтерпретабельності з Java, що дозволяє розробникам легко поєднувати код на обох мовах у єдиному проєкті. Це дозволяє забезпечити плавний процес міграції з Java на Kotlin для існуючих проєктів або створення нових додатків з використанням обох мов.

З врахуванням активної підтримки від Google, а також розвитку спільноти розробників, Kotlin стає все більш привабливим вибором для розробки Android-додатків, демонструючи потенційно значний внесок у покращення продуктивності та якості програмного забезпечення для мобільних пристроїв.

У даному застосунку було обрано мову програмування Java.

Для створення дизайну було обрано Figma, який пропонує широкий спектр можливостей для створення візуальних елементів додатку. Основною перевагою Figma є її простота використання, який дозволяє швидко та зрозуміло, навіть для новачків, створювати красивий дизайн (рис. 2.4) [12].

Для керування версіями було обрано систему Git, що дозволяє розробникам вести контроль над змінами у своєму коді. Він дозволяє створювати, зберігати та об’єднувати версії файлів, що спрощує процес спільної роботи над проектами.

GitHub – це хмарний сервіс, який надає хостинг для проектів, керування файлами та спільною роботою. Вони дозволяють розробникам разом працювати над проектами, відстежувати зміни у коді та вносити внески у загальний код (рис. 2.5).

Використання Git спільно з репозиторіями на платформах GitHub дозволяє розробникам ефективно керувати версіями коду проекту, спільно працювати над проектами та забезпечує швидкий та зручний обмін новими змінами між учасниками команди.

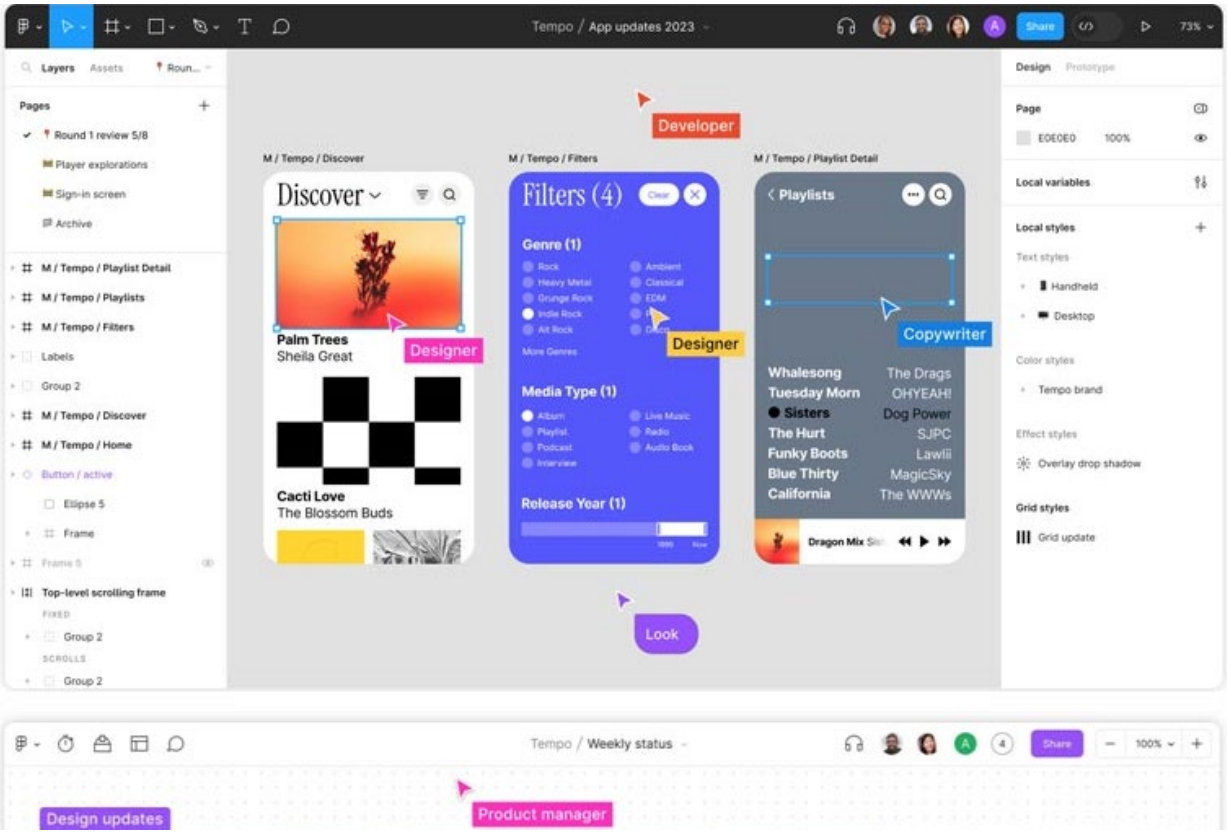

Рисунок 2.4 – Середовище створення дизайну Figma

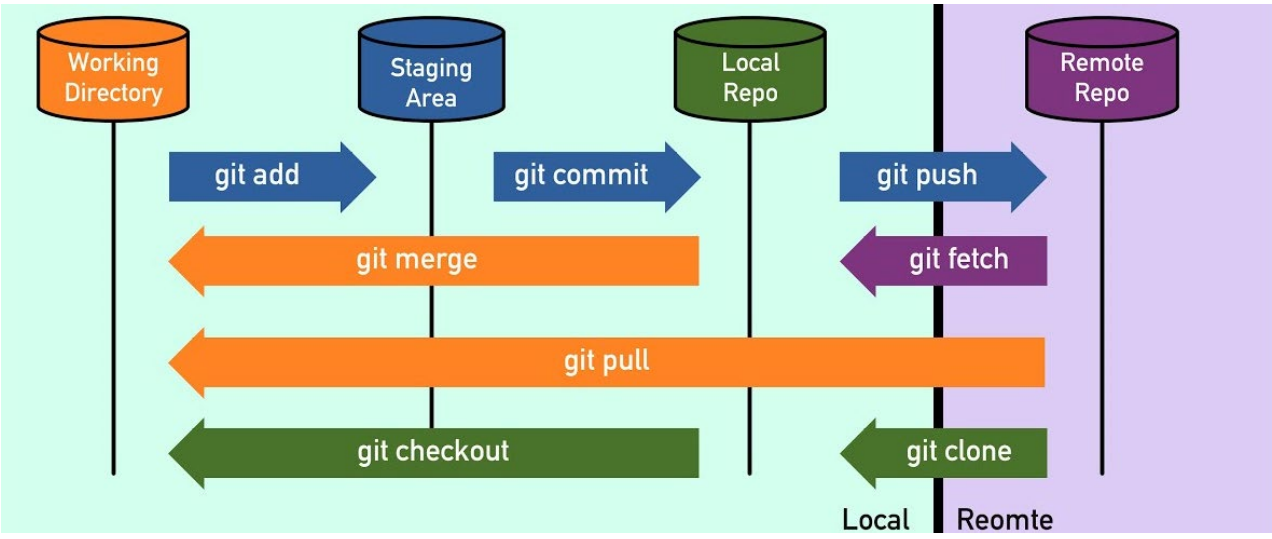

Рисунок 2.5 – Схема роботи Git

Для реалізації автентифікації користувачів, зберігання та синхронізації даних у реальному часі, а також для надання інструментів для аналізу та вдосконалення продуктивності додатку використано Firebase. Firebase дозволяє забезпечити безпеку та надійність обміну даними між мобільними клієнтами та сервером, а також спрощує розгортання та підтримку додатку (рис. 2.6).

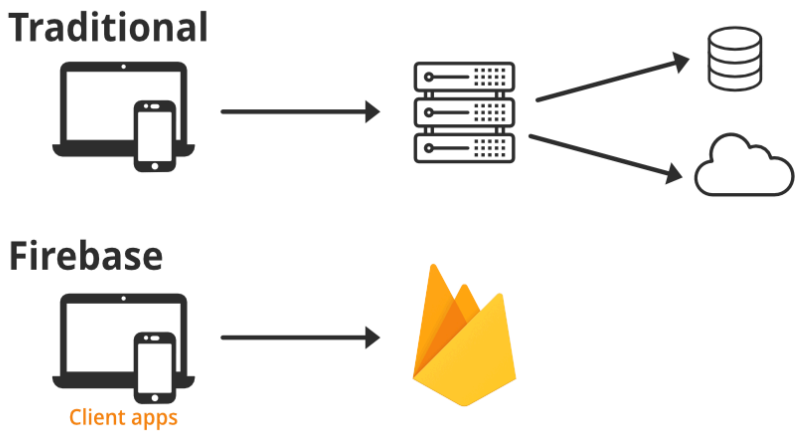

Рисунок 2.6. – Принцип роботи Firebase

Для роботи з базою даних було використано RoomDatabase. Це компонент бібліотеки архітектури Jetpack, який надає простий та потужний спосіб роботи з базою даних SQLite на рівні абстракції вищого рівня. Room дозволяє визначати базу даних, таблиці та їхні зв’язки за допомогою анотацій, а також надає зручний механізм доступу до даних через об’єкти доступу до даних (DAO).

2.5. Особливості програмної реалізації та основні режими роботи "nAzars"

2.5.1. Основні модулі програми

Архітектура застосунку базується на клієнт-серверній моделі, де мобільні клієнти взаємодіють з сервером для обміну даними. На рис. 2.7. наведено структуру застосунку.

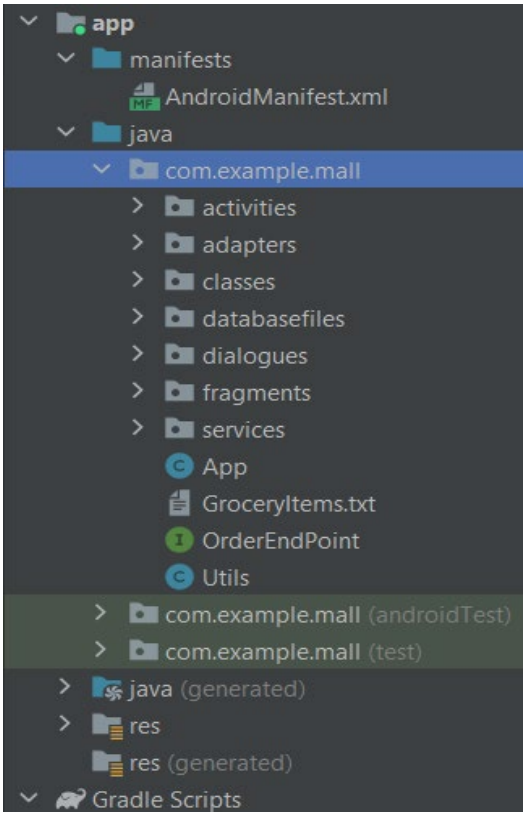

Рисунок 2.7. – Структура мобільного застосунку "nAzars"

У папці manifests міститься файл AndroidManifest.xml – це основний XML файл конфігурації для Android застосунку. Він містить загальну структуру та поведінку додатку, такі як назва додатку, версія, дозволи, активності, служби, броадкаст приймачі і провайдери контенту.

У папці java знаходяться файли з розширенням .java, завдяки яким виконується впровадження логіки мобільного додатку.

В папці layout знаходяться всі інтерфейси (інтерфейси кожного екрану застосунку), які мають розширення .xml та використовуються у додатку.

У папці mipmap та drawable зберігаються різні версії іконок та зображень, що використовуються для відображення логотипу та інших графічних елементів додатку. Ці структури розподіляють іконки за їхньою роздільною здатністю для забезпечення оптимального відображення на пристроях з різною щільністю пікселів.

У папці values зберігаються значення різноманітних ресурсів для проєкту Android, які використовуються для управління стилями, темами, текстовими рядками, розмірами та кольорами.

Файл build.gradle – це файл налаштування, який використовується для визначення параметрів збірки для проєкту Android, а саме визначає: версію Android SDK і Gradle, залежності бібліотек, конфігурацію збірки.

У складі застосунку входять такі модулі та компоненти: модуль автентифікації, модуль управління корзиною, модуль перегляду детальної інформації про продукт, модуль перегляду всіх посортованих товарів за популярністю та часом, модуль мапи, модуль чека замовлення, модуль перегляду усіх замовлень, модуль пошуку товарів за ключовими символами та категоріями, модуль налаштувань програми, модуль редагування профілю користувача, модуль веб-сайту, модуль замовлення товарів і модуль алгоритмів роботи з базою даних. Кожен з цих модулів відповідає за певну функціональність додатку і має свої власні компоненти, такі як активності, фрагменти, діалогові вікна та сервіси.

Програма складається з таких основних модулів:

- модуль інтерфейсу користувача відповідає за створення та управління інтерфейсом користувача програм;
- модуль управління даними відповідає за зберігання, обробку даних програми та надсилання даних на сервер;
- модуль реалізації алгоритмів відповідає за реалізацію методів та модулів, які використовуються програмою.

Модуль інтерфейсу користувача реалізований за допомогою бібліотеки Android UI. Бібліотека Android UI надає широкий спектр компонентів для створення візуальних елементів інтерфейсу користувача, таких як вікна, кнопки, меню та інші.

Модуль управління даними реалізований з використанням Room Database, що є частиною Android Jetpack. Цей модуль використовує Firebase базу даних для зберігання та керування даними програми. Крім того, він використовує

аутентифікацію Firebase для безпечного доступу користувачів до даних та сховище Firebase для зберігання медіафайлів, таких як зображення товарів та інші важливі файли.

Модуль реалізації алгоритмів реалізований за допомогою класів і об’єктів мови програмування Java. Модуль реалізує такі алгоритми та методи, як:

- алгоритм персоналізованого підбору товарів для користувача;
- метод пошуку товарів у базі даних;
- алгоритм відстеження активності користувача на різних товарах, для створення персональних рекомендацій;
- метод надсилання повідомлень користувачу;
- алгоритм обробки замовлень та оплати;
- алгоритм слухача мапи, який дозволятиме користувачу подивитись інформацію про конкретний магазин;
- алгоритм зміни дозволу надсилання повідомлень користувачу;
- метод створення історії покупок користувача.

Програма має такі основні режими роботи:

- режим перегляду товарів у якому користувач може оглядати асортимент товарів за категоріями та шукати їх за різними символами;
- режим перегляду товару, завдяки якому, користувач зможе при натисненні на товар отримувати детальну про нього;
- режим додавання та редагування товарів у кошику, завдяки якому, користувач може змінювати товари у кошику для подальшого оформлення замовлення;
- режим оплати та оформлення замовлення, завдяки якому, користувач може обирати метод оплати, вказувати адресу доставки та оформити замовлення;
- режим управління обліковим записом, завдяки якому, користувач може переглядати та редагувати свій профіль, переглядати історію замовлень та виконувати інші дії, пов’язані з особистим обліковим записом.

2.5.2. Опис структури та роботи програми

Наведемо алгоритми та методи, реалізовані в коді застосунку.

На рис. 2.8. подано реалізацію додавання продукту в кошик.

```
@Query("INSERT INTO cart_items (grocery_item_id) values (:id)")
void insert(int id);
```

Рисунок 2.8. – Реалізація додавання продукту в кошик

На рис. 2.9. подано реалізацію отримання всіх продуктів у кошику.

```
@Query("select grocery_items.id, grocery_items.name," +
        " grocery_items.description, grocery_items.imageUrl," +
        " grocery_items.category, grocery_items.price," +
        " grocery_items.availableAmount, grocery_items.rate, " +
        "grocery_items.userPoint, grocery_items.popularityPoint, " +
        "grocery_items.reviews, grocery_items.salePrice from grocery_items " +
        "INNER JOIN cart_items on cart_items.grocery_item_id = grocery_items.id")
List<GroceryItem> getAllCartItems();
```

Рисунок 2.9. – Реалізація отримання всіх продуктів з кошику

На рис. 2.10. подано реалізацію видалення продукту з кошику.

```
@Query("delete from cart_items where grocery_item_id =:id")
void deleteFromCart(int id);
```

Рисунок 2.10. – Реалізація видалення продукту з кошику

На рис. 2.11 подано реалізацію додавання замовлення.

```
@Insert
void insert(Order order);
```

Рисунок 2.11. – Реалізація додавання замовлення

На рис. 2.12 та 2.23 подано реалізацію отримання всіх продуктів замовлення та інформації про магазини відповідно.

```
@Query("SELECT * FROM order_items")
List<Order> getAllItems();
```

Рисунок 2.12. – Реалізація отримання всіх продуктів з замовлення

```
@Query("SELECT * FROM shop_items")
List<ShopModel> getAllItems();
```

Рисунок 2.13. – Реалізація отримання всіх продуктів з замовлення

Метод пошуку товарів у базі даних обробляє запити користувачів і шукає в базі даних товари, що відповідають критеріям пошуку. Він використовуватиме відповідні запити SQL завдяки інтерфейсу бібліотеки Room Database (рис. 2.14), такі як індексація або повнотекстовий пошук, для ефективного пошуку товарів.

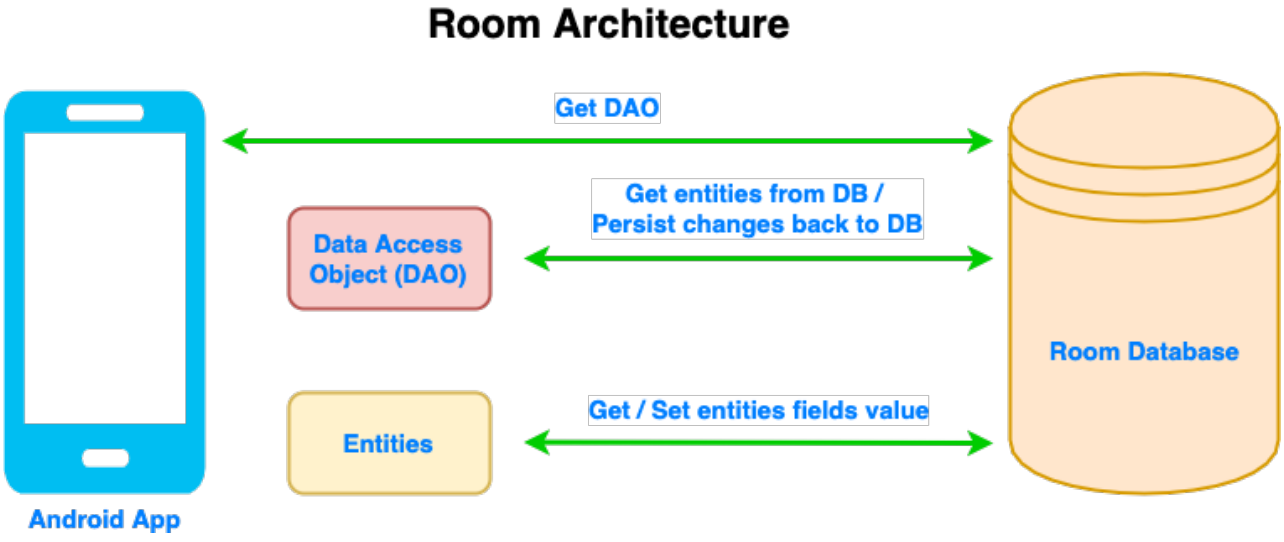

Рисунок 2.14. – Принцип роботи Room Database

На рис 2.15 – 2.17 подано реалізацію пошуку товарів: SearchActivity.java (рис 2.15); Utils.java (рис. 2.16); GroceryItemDao.java (рис. 2.17).

```
private void initSearch() {
    if(!searchBox.getText().toString().equals("")){
        String name = searchBox.getText().toString();
        ArrayList<GroceryItem> foundedItems = Utils.searchItemsByName( context: this, name);
        if(null!=foundedItems){
            adapter.setItems(foundedItems);
            adapterBig.setItems(foundedItems);
        }
    }else{
        ArrayList<GroceryItem> allItems = Utils.getAllItems( context: this);
        if(null!= allItems){
            adapter.setItems(allItems);
            adapterBig.setItems(allItems);
        }
    }
}
```

Рисунок 2.15. – Реалізація отримання всіх товарів за набраними символами

```
public static ArrayList<GroceryItem> searchItemsByName(Context context, String text){
    return (ArrayList<GroceryItem>) ShopDatabase.getInstance(context).groceryItemDao().getItemsBySearch( text: "%" + text +"%");
}
```

Рисунок 2.16. – Реалізація отримання всіх товарів за набраними символами

```
@Query("select * from grocery_items where name LIKE :text")
List<GroceryItem> getItemsBySearch(String text);
```

Рисунок 2.17. – Реалізація отримання всіх товарів за набраними символами

Результат реалізації пошуку товарів наведено на рисунку 2.18.

Алгоритм персонального підбору товарів для користувача відстежує його дії користувача, такі як перегляди товарів, оцінку товарів, додавання до кошика або покупки і використовує цю інформацію для підбору персоналізованих рекомендацій. Для цього буде створена служба, яка аналізуватиме активність користувача у всій програмі, щоб визначити інтереси та потреби користувача і рекомендувати відповідні товари.

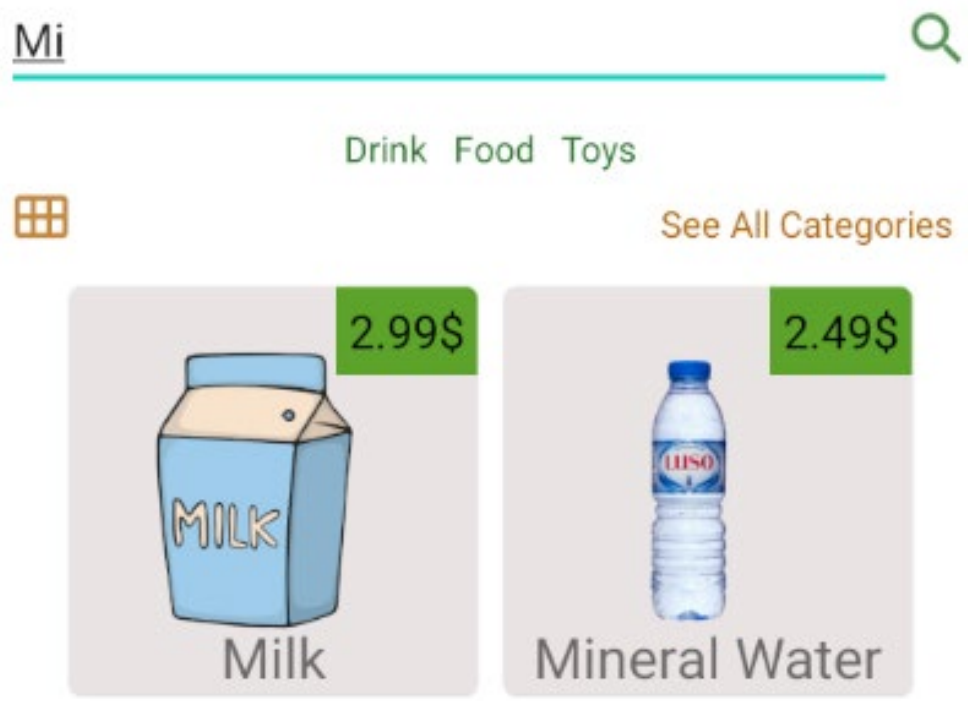

Рисунок 2.18. – Результат пошуку товару за символами "Mi"

Сервіс, який відповідає за відслідковування проведеного часу на будь-якому товарі – TrackUserTime.java (рис. 2.19 – 2.21). Функція, яка відповідає за зміну оцінки користувача – Utils.changeUserPoint().

Результат реалізації даного алгоритму наведено на рисунку 2.22.

Метод надсилання повідомлень користувачу відправлятиме повідомлення користувачам про різні події, такі як щоденне оновлення акцій, спеціальні пропозиції або оновлення статусу доставки завдяки пуш-сповіщенням.

Планувальник завдань, який відповідає за надсилання повідомлень – NotificationJobService.java. Код реалізації створення каналу для одного типу повідомлень у головному коді програми App.java наведено на рис. 2.23, ініціалізації планувальника завдань після першого запуску програми – на рис. 2.24, функції у робочому потоці, для надсилання пуш–повідомлення на телефон, яка перевіряє чи знаходиться програма у вимкненому стані, або не у фокусі – на рис. 2.25. Результат реалізації даних методів наведено на рисунку 2.26.

```
private void trackTime(){
    Bohdan
    Thread thread = new Thread(new Runnable() {
        Bohdan
        @Override
        public void run() {
            while(!shouldFinish){
                try {
                    Thread.sleep( millis: 1000);
                    seconds++;
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            }
        }
    });
    thread.start();
}
new *
@Override
public void onDestroy() {
    super.onDestroy();
    shouldFinish = true;
    int minutes = seconds / 60;
    if(minutes>0){
        if(null!=item) {
            Utils.changeUserPoint( context: this, item, minutes);
        }
    }
}
```

Рисунок 2.19. – Реалізація сервісу для відслідковування проведеного часу, TrackUserTime.java

```
public static void changeUserPoint(Context context, GroceryItem item, int points){
    int newPoints = item.getUserPoint() + points;
    ShopDatabase.getInstance(context).groceryItemDao().changeUserPoint(item.getId(), newPoints);
    Log.d(TAG,  msg: "changeUserPoint: Changed user point to: " + newPoints);
}
```

Рисунок 2.20. – Реалізація сервісу для відслідковування проведеного часу, Utils.java

```
@Query("update grocery_items set userPoint=:newPoints where id=:id")
void changeUserPoint(int id, int newPoints);
```

Рисунок 2.21. – Реалізація сервісу для відслідковування проведеного часу, GroceryItemDao.java

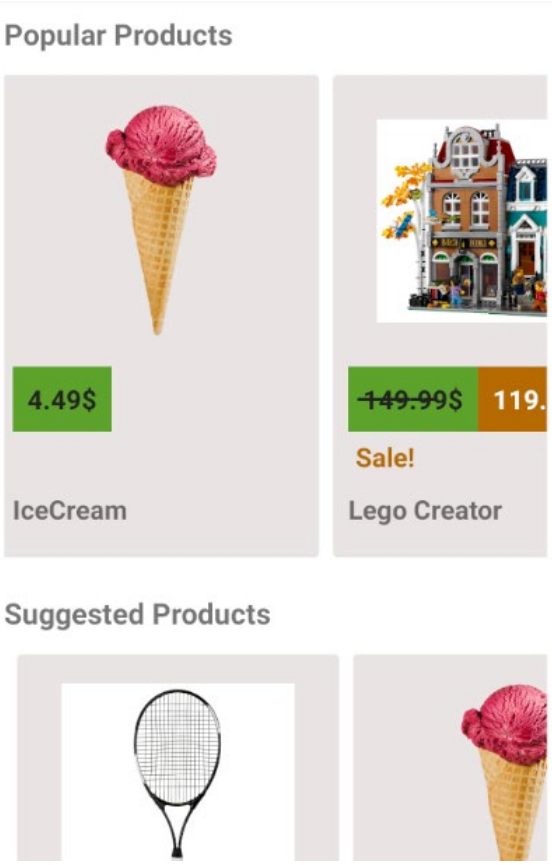

Рисунок 2.22. – Персональне сортування товарів

```
private void createNotificationChannel() {
    NotificationChannel channel1 = new NotificationChannel( id: "channel1",
            name: "Channel", NotificationManager.IMPORTANCE_HIGH);
    channel1.setDescription("This channel is using for main notifications");
    NotificationManager manager = getSystemService(NotificationManager.class);
    manager.createNotificationChannel(channel1);
}
```

Рисунок 2.23. – Реалізація створення каналу повідомлень

```
private void initNotificationJobScheduler() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
        ComponentName componentName = new ComponentName( pkg: this, NotificationJobService.class);
        JobInfo.Builder builder = new JobInfo.Builder(NOTIFICATION_JOB_ID, componentName)
                .setPersisted(true);
        builder.setPeriodic( intervalMillis: 15*60*1000,  flexMillis: 10*60*1000);
        JobScheduler scheduler = (JobScheduler)getSystemService(JOB_SCHEDULER_SERVICE);
        scheduler.schedule(builder.build());
    }
}
```

Рисунок 2.24. – Ініціалізація планувальника завдань

```
private void sendNotification(Context context) {
    Intent intent = new Intent(context, MainActivity.class);
    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
    PendingIntent pendingIntent = PendingIntent.getActivity(context, requestCode: 2, intent, flags: 0);
    NotificationCompat.Builder builder = new NotificationCompat.Builder(context, channelId: "channel1")
            .setContentTitle("Hello user!")
            .setSmallIcon(R.drawable.ic_bell)
            .setContentText("Check the new promotions!")
            .setPriority(NotificationManagerCompat.IMPORTANCE_HIGH)
            .setColor(getResources().getColor(R.color.purple_200))
            .setVisibility(NotificationCompat.VISIBILITY_PRIVATE)
            .setContentIntent(pendingIntent)
            .setAutoCancel(true);
    NotificationManagerCompat manager = NotificationManagerCompat.from(context);
    if (ActivityCompat.checkSelfPermission(context,
            Manifest.permission.POST_NOTIFICATIONS) != PackageManager.PERMISSION_GRANTED) {
        return;
    }
    manager.notify( id: 1, builder.build());
}
```

Рисунок 2.25. – Ініціалізація планувальника завдань

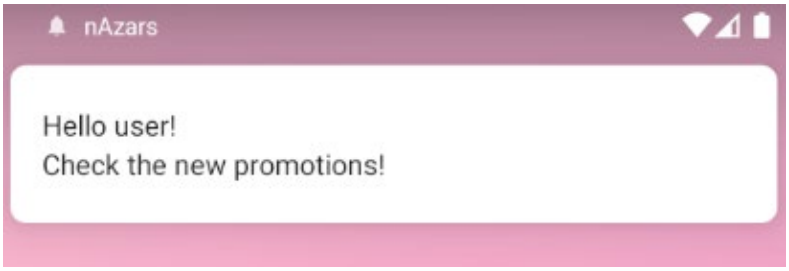

Рисунок 2.26. – Пуш-повідомлення

Алгоритм обробки замовлень та оплати оброблятиме замовлення користувачів. Логіка обробки знаходиться у ThirdCardFragment.java (рис. 2.27).

Результат успішного замовлення наведено на рисунку 2.28.

Алгоритм слухача мапи дозволятиме користувачам переглядати інформацію про розташування усіх магазинів мережі "nAzars" на мапі. Він використовуватиме API Google SDK мапи для відображення розташування та використовуватиме базу даних Firebase для отримання усіх даних про магазин. Логіка знаходиться у файлі MapActivity.java. Реалізацію отримання геолокації телефону подано на рис. 2.29, а розстановки маркерів магазинів по координатам на рис. 2.30. Результат використання даних методів наведено на рисунку 2.31.

```
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://jsonplaceholder.typicode.com/")
        .addConverterFactory(GsonConverterFactory.create())
        .client(client)
        .build();
OrderEndPoint orderEndPoint = retrofit.create(OrderEndPoint.class);
Call<Order> call = orderEndPoint.newOrder(order);
 Bohdan *
call.enqueue(new Callback<Order>() {
     Bohdan *
    @Override
    public void onResponse(Call<Order> call, Response<Order> response) {
          Log.d(TAG, "onResponse: code: " + response.code());
        if(response.isSuccessful()) {
            order.setTime(Utils.getCurrentTime());
            order.setDate(Utils.getCurrentSymbolicDate());
            Utils.addOrder(getActivity(), order);
            Bundle bundle = new Bundle();
            bundle.putString(ORDER_KEY, gson.toJson(response.body()));
            PaymentResultFragment paymentResultFragment = new PaymentResultFragment();
            paymentResultFragment.setArguments(bundle);

            FragmentTransaction transaction = getActivity().getSupportFragmentManager().beginTransaction();
            transaction.replace(R.id.cartContainer, paymentResultFragment);
            transaction.commit();
        }else{
            FragmentTransaction transaction = getActivity().getSupportFragmentManager().beginTransaction();
            transaction.replace(R.id.cartContainer, new PaymentResultFragment());
            transaction.commit();
        }
    }
```

Рисунок 2.27. – Обробка замовлення користувача

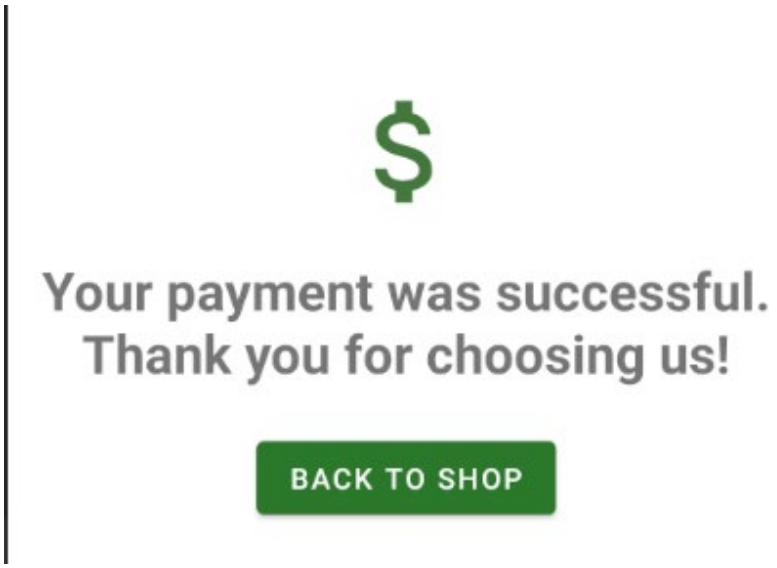

Рисунок 2.28. – Користувач отримує повідомлення успішної оплати.

```
private void getLastLocation() {
    if (ActivityCompat.checkSelfPermission( context: this,
            Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
            && ActivityCompat.checkSelfPermission( context: this,
            Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions( activity: this,
                new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, LOCATION_PERMISSION_REQUEST_CODE);
        return;
    }
    Task<Location> task = fusedLocationProviderClient.getLastLocation();
    Bohdan *
    task.addOnSuccessListener(new OnSuccessListener<Location>() {
        Bohdan *
        @Override
        public void onSuccess(Location location) {
            if(location != null){
                currentLocation = location;
                SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
                mapFragment.getMapAsync( callback: MapActivity.this);
            }
        }
    });
}
```

Рисунок 2.29. – Отримання геолокації користувача

```
private void markShopsFromDb() {
    ArrayList<ShopModel> stores = Utils.getAllShopItems( context: this);
    for (ShopModel s : stores) {
        LatLng latlng = new LatLng(s.getLatitude(), s.getLongitude());
        String address = getAddressFromCoordinates(latlng);

        MarkerOptions markerOptions = new MarkerOptions()
                .position(latlng)
                .title(address)
                .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_VIOLET));

        Marker marker = myMap.addMarker(markerOptions);
        markerShopIdMap.put(marker, s.getId());
    }
}
```

Рисунок 2.30. – Код зображення локацій магазинів

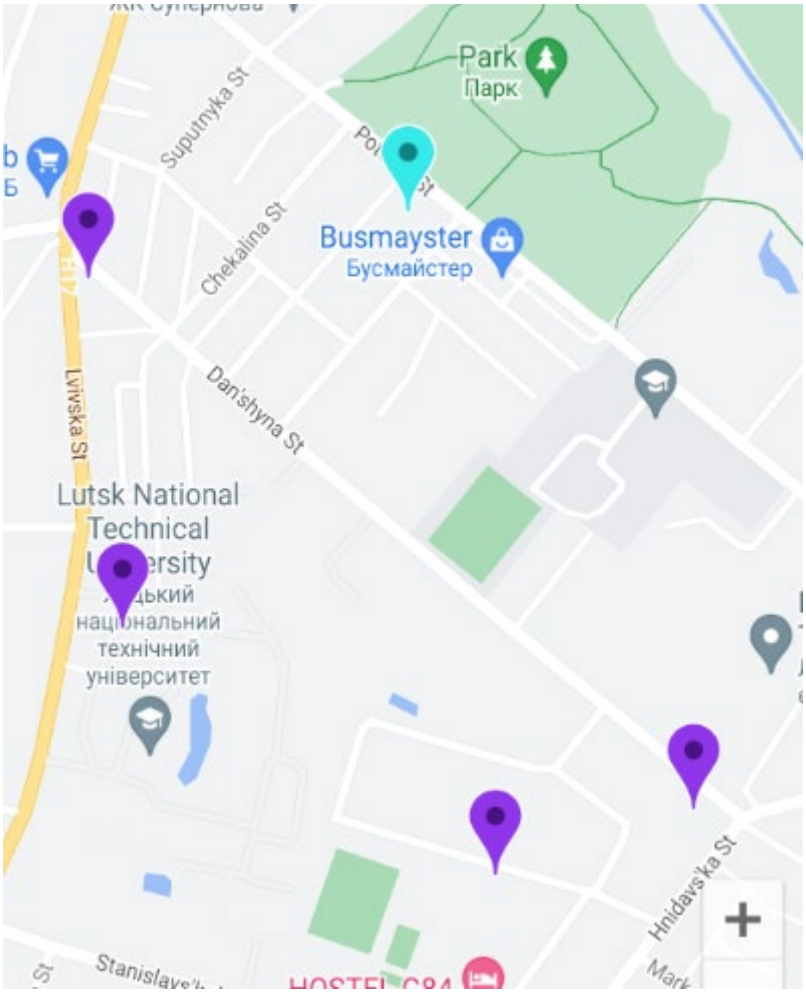

Рисунок 2.31. – Розміщені маркери магазинів та користувача на мапі

Алгоритм зміни дозволу надсилання повідомлень користувачу дозволятиме користувачам змінювати свої налаштування щодо отримання повідомлень. Він забезпечить користувачам можливість встановлювати власні уподобання щодо отримання сповіщень про акції, новини та інші події. Логіка знаходиться у файлі SettingsFragment.java (рис.2.32). Результат використання цього алгоритму наведений на рис. 2.33.

Метод створення історії покупок користувача вестиме облік історії покупок користувача, щоб забезпечити зручний доступ до інформації про минулі покупки та допомогти в подальших плануваннях покупок. Логіка знаходиться у файлі OrdersFragment.java (рис. 2.34).

```
switchNotification.setOnClickListener(new View.OnClickListener() {
    Bohdan
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(Settings.ACTION_APP_NOTIFICATION_SETTINGS);
        intent.putExtra(Settings.EXTRA_APP_PACKAGE, getActivity().getPackageName());
        startActivity(intent);
    }
});
private void setStartSwitch(){
    NotificationManagerCompat manager = NotificationManagerCompat.from(getActivity());
    switchNotification.setChecked(manager.areNotificationsEnabled());
}
```

Рисунок 2.32. – Реалізація зміни дозволу надсилання повідомлень

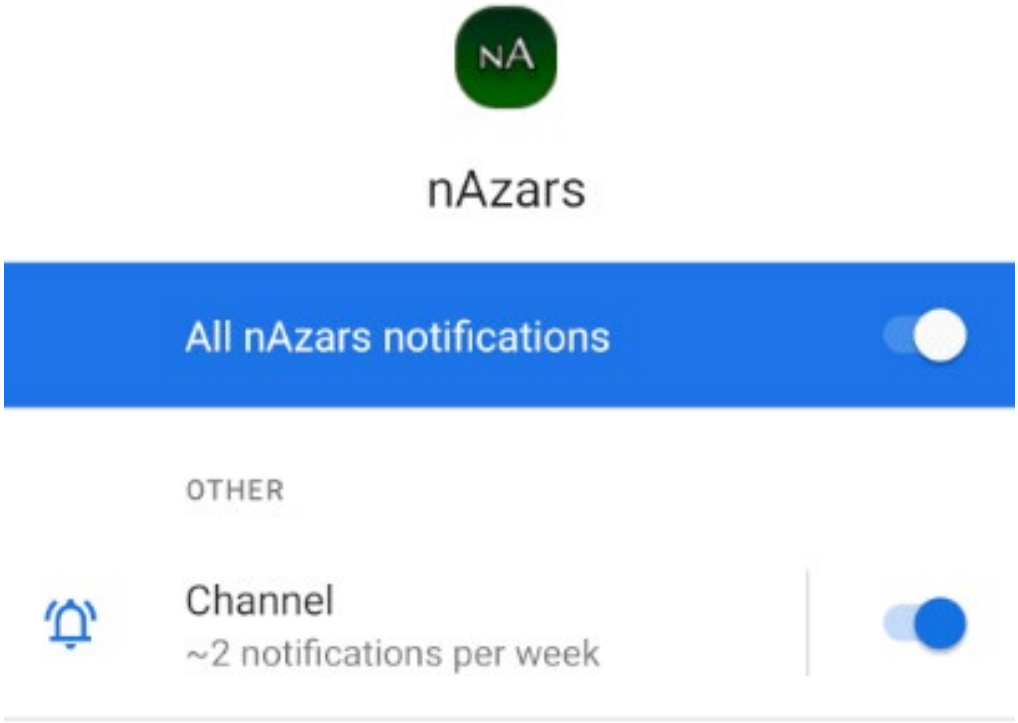

Рисунок 2.33. – Перемикач дозволу пуш-повідомлень

```
ArrayList<Order> orderItems = Utils.getAllOrdersItems(getActivity());
if(null!=orderItems) {
    if (orderItems.size() > 0) {
        Bohdan
        Comparator<Order> orderItemsComparator = new Comparator<Order>() {
            Bohdan
            @Override
            public int compare(Order o1, Order o2) { return o1.getId() - o2.getId(); }
        };
        Comparator<Order> reverseComparator = Collections.reverseOrder(orderItemsComparator);
        Collections.sort(orderItems, reverseComparator);
        adapter.setOrderItems(orderItems);
    }
}
```

Рисунок 2.34. – Реалізація виводу усіх замовлень користувача у RecyclerView

Результат виконання цього методу подано на рисунку 2.35.

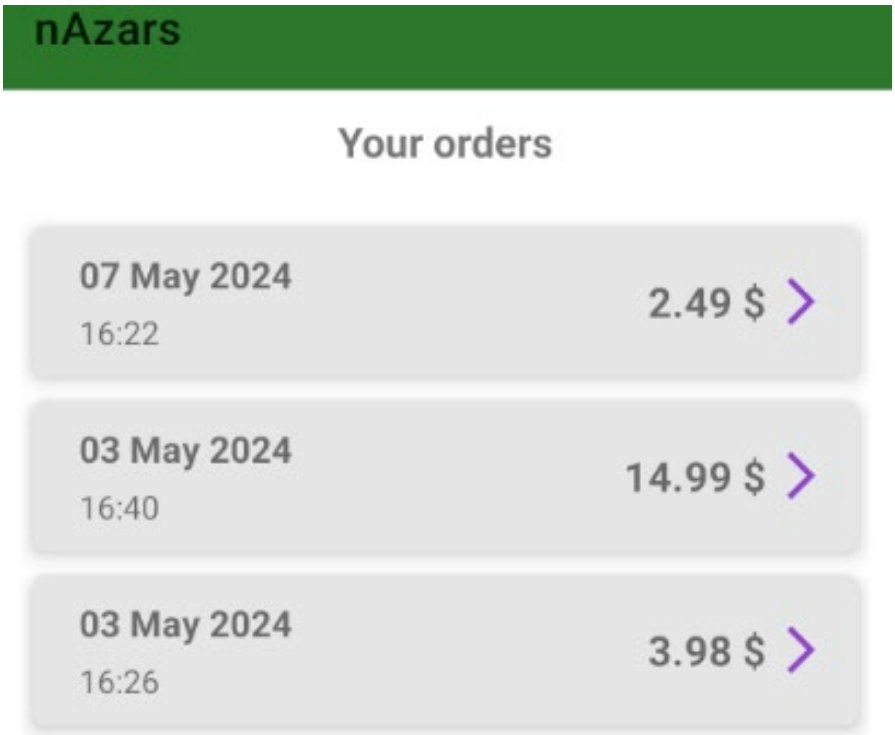

Рисунок 2.35 – Усі замовлення користувача

2.5.3. Значущі компоненти програмної розробки

Під час розробки програмного продукту було додано ряд значущих компонентів, спрямованих на покращення користувацького досвіду та персоналізацію рекомендацій для користувачів.

Один із таких компонентів – це сервіс, що відслідковує активність користувачів та персоналізує рекомендації на основі часу, проведеного користувачем при перегляді конкретних продуктів. Конкретно, якщо користувач переглядає певний продукт протягом певного періоду часу, наприклад, більше однієї хвилини, йому автоматично нараховуються бали. Ці бали впливають на подальші рекомендації користувачу, роблячи їх більш персоналізованими та зорієнтованими на індивідуальні інтереси.

Крім того, був реалізований планувальник подій, який виконується щоденно о 00:00. Цей планувальник відповідає за оновлення знижок на продукти, забираючи старі знижки та додаючи нові. За цією бізнес-логікою, мінімум 40% продуктів отримують знижку, що сприяє стимулюванню продажів та залученню уваги користувачів до нових пропозицій.

Додатково, у програмі використовується служба, яка автоматично надсилає повідомлення користувачам кожні 5-15 хвилин. Цей інтервал часу був обраний для тестування функціональності сервісу надсилання повідомлень.

Всі ці процеси та компоненти реалізовані в головному коді додатка при його першому запуску, що забезпечує їхню ефективну та надійну роботу протягом усього життєвого циклу програми.

Для відображення оригінальної іконки та назви застосунку, було створено та додано іконку програми, а також змінено основну назву програми на "nAzars" (рис. 2.36). Всі ці зміни записано до файлу "AndroidManifest.xml".

Інструкція для користувача знаходиться у додатку В.

Рисунок 2.36. – Іконка застосунку "nAzars"

2.6. Організація тестування та налагодження програмного засобу "nAzars"

Під час розробки програмного засобу "nAzars" велика увага була відведена організації тестування та налагодження, з метою забезпечення якості та надійності продукту. Тестування програми проводилось розробниками на різних пристроях з метою перевірки сумісності та оптимальної роботи на різних платформах та роздільних здатностях екранів.

У процесі тестування програмного засобу виявлено кілька потенційних проблем та недоліків, які були виправлені розробниками для поліпшення функціональності та забезпечення кращого користувацького досвіду. Однією з таких проблем була некоректне відображення деяких інтерфейсних елементів на пристроях з низькою роздільною здатністю екрану. Ця проблема була вирішена шляхом адаптації інтерфейсу до різних роздільних здатностей пристроїв.

Крім того, в процесі тестування було виявлено проблеми зі стабільністю деяких функціональних можливостей програми, таких як автоматичне оновлення даних та відображення деяких елементів інтерфейсу. Ці проблеми були вирішені за допомогою удосконалення алгоритмів та оптимізації роботи програмного засобу.

Таким чином, завдяки організованому підходу до тестування та налагодження програмного засобу "nAzars", вдалося виявити та виправити ряд проблем, що сприяло покращенню якості та надійності додатку перед його релізом на ринок.

2.7. Рекомендації по використанню та впровадженню програмного засобу "nAzars"

Мобільний застосунок "nAzars" є практичним програмним продуктом, який можна використовувати з метою забезпечення зручного та швидкого доступу користувачів до товарів та послуг, а також для оптимізації процесу покупок. У цьому контексті, програмний засіб "nAzars" є важливим інструментом для підвищення якості обслуговування клієнтів та збільшення обсягу продажів у сфері роздрібної торгівлі. Застосунок має наступні переваги, які роблять його практичним:

- застосунок має простий і інтуїтивно зрозумілий інтерфейс;
- застосунок має всі необхідні функції для створення для зручного користування головними функціями програми.

ВИСНОВКИ

Результатом кваліфікаційної роботи був створений програмний продукт для роздрібного магазину, що відповідає всім функціональним вимогам, поставленим перед розробником. Застосунок реалізований на мові програмування Java в середовищі розробки Android Studio.

Під час розробки було проведено аналіз сучасних методів та підходів для розробки мобільних застосунків, що сприяло вдосконаленню технічної складової додатку.

Одним із ключових досягнень є створення зручного та інтуїтивно зрозумілого інтерфейсу, який адаптується під різні розміри екранів. Це забезпечує користувачам можливість використання застосунку на різних мобільних пристроях, що підтримують операційну систему Android 11.0.

Додаток пройшов успішне тестування як на симуляторі, так і на реальних мобільних пристроях. Під час тестування було підтверджено відсутність проблем та багів, а також показано швидку та плавну реакцію на дії користувача. Це свідчить про готовність застосунку до використання в реальних умовах та гарантує задоволення потреб користувачів під час використання даного програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мобільні застосунки: їх види та особливості. *Highload.today*. URL: https://highload.today/uk/mobilni-zastosunki-yih-vidi-ta-osoblivosti/ (дата звернення: 03.05.2024).

2. David M., Novotny A., Denman J. What is mobile application development?. *App Architecture*. URL: https://www.techtarget.com/searchapparchitecture/definition/mobile-application-development (дата звернення: 10.05.2024).

3. Hamburger E. These Charts Tell The Real Story Of Android Vs. iPhone. *Business Insider*. URL: https://www.businessinsider.com/android-vs-iphone-charts-2011-12?IR=T#october-5-2010-projections-show-that-android-may-have-already-taken-the-lead-4 (дата звернення: 15.05.2024).

4. Android Architecture. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/android-architecture/ (дата звернення: 03.12.2023).

5. Офіційний сайт «Сільпо». URL: https://silpo.ua/about/mobileapp (дата звернення: 14.05.2024).

6. Glovoapp 23SL. Glovo: Food Delivery and More - Apps on Google Play. Android Apps on Google Play. URL: https://play.google.com/store/apps/details?id=com.glovo&hl=ua&pli=1 (date of access: 14.05.2024).

7. Ітеративна модель (iterative model). *QALight*. URL: https://qalight.ua/baza-znaniy/iterativna-model-iterative-model/ (дата звернення: 03.05.2024).

8. Material Design. *Material Design*. URL: https://m3.material.io/ (дата звернення: 03.05.2024)

9. Download Android Studio & App Tools. *Android Developers*. URL: https://developer.android.com/studio?gclid=Cj0KCQiA67CrBhC1ARIsACKAa8ShdUtJczI7daatPSVZNzaI9NeFGFu6BEH0IXRCB-

j6I9TCkhum4aMaAiCsEALw_wcB&gclsrc=aw.ds (дата звернення: 03.05.2024).

10. Benefits of Java for Mobile Application Development. *nCube*. URL: https://ncube.com/benefits-of-java-for-mobile-application-development/ (дата звернення: 03.05.2024).

11. What is Figma and its Advantages. *UX Academy*. URL: https://myuxacademy.com/what-is-figma/ (дата звернення: 03.05.2024).

12. Co to jest Firebase? - Słownik pojęć analitycznych. Bigglo. URL: https://bigglo.pl/slownik/co-to-jest-firebase/ (date of access: 03.05.2024).

13. Fragments | Android Developers. Android Developers. URL: https://developer.android.com/guide/fragments (date of access: 03.05.2024).

14. Services overview | Background work | Android Developers. Android Developers. URL: https://developer.android.com/develop/background-work/services (date of access: 03.05.2024).

ДОДАТКИ

Додаток А

Аспекти розробки мобільних додатків для Android і iOS наведено у таблиці А.1.

Таблиця А.1

Аспекти розробки для Android та iOS

Аспекти розробки	**Android**	**iOS**
Архітектура	Спрямована на системі компонентів Android (Activity, Service, BroadcastReceiver).	Спрямована на архітектурі MVC (Model-View–Controller).
Мова програмування	Java, Kotlin	Swift, Objective–C
Інтерфейс користувача	XML для UI розмітки дуже гнучкий та розширюваний	Interface Builder для графічного дизайну і розмітки
Підтримка пристроїв	Різноманітність пристроїв, різні розміри екранів, адаптований до різних моделей	Деяка відмінність у роздільній здатності, але менше різноманітності пристроїв
Безпека	Використання системи дозволів, Secure Coding Practices	Захищений від вірусів і шкідливих програм

Додаток Б

Технічне завдання

1. Вступ

Застосунок "nAzars" – мобільний застосунок призначений для збільшення продажів, та збільшенню цільової аудиторії мережі роздрібних магазинів.

2. Підстави для розробки

Затвердив – Волинський національний університет імені Лесі Українки. Тема – "Розробка та оптимізація мобільного застосунку для роздрібного магазину".

3. Призначення розробки

Функціональне призначення застосунку полягає в тому, щоб надавати користувачам можливість здійснення покупок, забезпечуючи зручність, швидкість та безпеку для користувачів.

Експлуатаційне призначення застосунку полягає в тому, щоб використовувати застосунок для перегляду та для замовлення товарів через мобільний телефон, який буде доступний користувачу скрізь.

4. Вимоги до програми чи програмного продукту

4.1. Вимоги до функціональних характеристик

Застосунок повинен забезпечувати виконання таких функціональних можливостей:

- перегляд доступних товарів та акцій у мережі магазинів;
- здійснення замовлень товарів на адресу вказану користувачем;

- створення персональних рекомендацій для кожного користувача.

4.2. Вимоги до надійності

Застосунок повинен забезпечувати надійне функціонування в наступних умовах:

- робота в мережі інтернет;
- робота на пристроях з різними технічними характеристиками;
- робота в різних режимах експлуатації.

4.3. Умови експлуатації

Застосунок повинен працювати на пристроях з наступними технічними характеристиками:

- операційна система Android 6.0 і вище;
- процесор не менше 1,5 ГГц;
- оперативна пам’ять не менше 2 Гб;
- внутрішня пам’ять не менше 1 Гб.

4.4. Вимоги до складу і параметрів технічних засобів

Застосунок не вимагає використання додаткових технічних засобів.

4.5. Вимоги до інформаційної і програмної сумісності

Застосунок повинен бути сумісний з наступними інформаційними структурами:

- дані про продукти (назва, категорія, опис, посилання на фото тощо);
- дані про плани замволення (назви продуктів, кількість продуктів, сума ціни усіх товарів тощо);

- дані про користувача (ім’я, адреса, фото тощо);
- дані про магазини (адреса, графік роботи, доступні сервіси);
- дані про користувацькі відгуки (айді відгуку, айді продукту, текст відгуку тощо);

4.6. Вимоги до маркування і упаковки

Застосунок повинен бути маркований наступними даними:

- найменування програми;
- версія програми.

4.7. Вимоги до транспортування і збереження

Застосунок не вимагає транспортування та збереження.

5. Вимоги до програмної документації

Програмна документація повинна включати в себе наступні документи:

- технічне завдання, який визначає основні вимоги до програми;
- проектування, яке містить проектну документацію, яка визначає структуру, алгоритми та інші аспекти програми;
- реалізація, яка містить код програми;
- експлуатаційна документація, яка містить інформацію, необхідну для експлуатації програми.

6. Техніко-економічні показники

Орієнтовна економічна ефективність розробки мобільного застосунку для роздрібного магазину оцінюється через збільшення обсягів продажів, зменшення

витрат на рекламу та залучення нових клієнтів через зручний інтерфейс та функціонал додатку.

7. Стадії і етапи розробки

Розробка мобільного застосунку для занять спортом містить наступні етапи:

- етап дослідження, на якому буде визначено вимоги до програми та розроблений технічний проект;
- етап розробки, на якому буде розроблена програмна документація та реалізована програма;
- етап тестування, на якому буде проведено тестування програми для виявлення помилок та недоліків;
- етап налагодження, на якому будуть усунені помилки та недоліки, виявлені на етапі тестування;
- етап впровадження, на якому програма буде впроваджена в експлуатацію.

Попередній термін розробки – 180 годин.

8. Порядок контролю і приймання

Контроль і приймання мобільного застосунку для роздрібного магазину буде здійснюватися розробником програми.

Додаток В

Інструкція користувачу

1. Загальні відомості

Мобільний застосунок "nAzars" для роздрібного магазину створено з метою надання клієнтам зручного та ефективного способу здійснення покупок онлайн. Він дозволяє користувачам переглядати широкий асортимент товарів, робити покупки та виконувати інші дії пов’язані зі шопінгом, у будь-який час та в будь-якому місці, просто маючи доступ до смартфону або планшету.

2. Функціональне призначення

Мобільний застосунок для роздрібного магазину призначений для забезпечення користувачам можливості швидко та зручно здійснювати покупки онлайн. Основні функції застосунку включають:

- користувачі можуть переглядати різноманітний асортимент товарів, представлених у роздрібному магазині;
- застосунок надає можливість швидкого та зручного пошуку потрібних товарів за назвою, категорією або іншими параметрами;
- користувачі можуть легко додавати обрані товари до кошика для подальшого оформлення замовлення;
- застосунок дозволяє користувачам оформляти замовлення, вказуючи необхідну кількість товарів, адресу доставки та іншу інформацію;
- застосунок надає можливість користувачам управляти своїм особистим обліковим записом, переглядати історію замовлень, змінювати адресу доставки та інші особисті дані.

3. Умови використання програми

- операційна система Android 6.0 і вище;
- процесор не менше 1,5 ГГц;
- оперативна пам’ять не менше 2 Гб;
- внутрішня пам’ять не менше 1 Гб.

4. Повідомлення оператору

У разі виникнення проблем або неясностей під час користування застосунком, користувачі можуть надсилати повідомлення оператору. Для зручності користувачів, застосунок надає можливість звернутися до оператора через спеціальну форму зворотного зв’язку, яка доступна у розділі підтримки або контактів. У цьому повідомленні користувачі можуть описати свою проблему, запитання або пропозиції щодо роботи застосунку. Після надсилання повідомлення оператору, зазвичай, надається підтримка з боку команди розвитку застосунку, яка стежить за відповідями користувачів та намагається вирішувати їх проблеми якнайшвидше і якісніше.

5. Опис роботи програми

При першому запуску програми користувач бачить активність автентифікації, завдяки якому забезпечується захист від ботів і в загальному безпека застосунку (рис. В.1). Користувач має на вибір дві кнопки, для залогування в акаунт та виходу, але так як він ще не увійшов, то вийти він не може. Після натиснення кнопки реєстрації, перед користувачам відкривається можливість входу завдяку Google акаунту (рис. В.2).

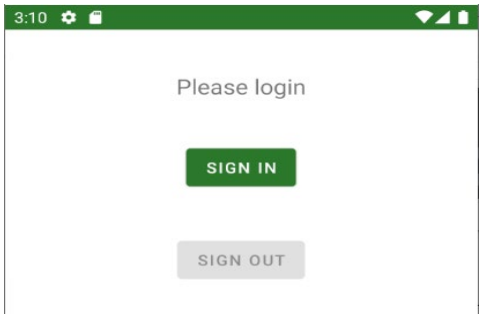

Рисунок В.1. – Екран автентифікації користувача

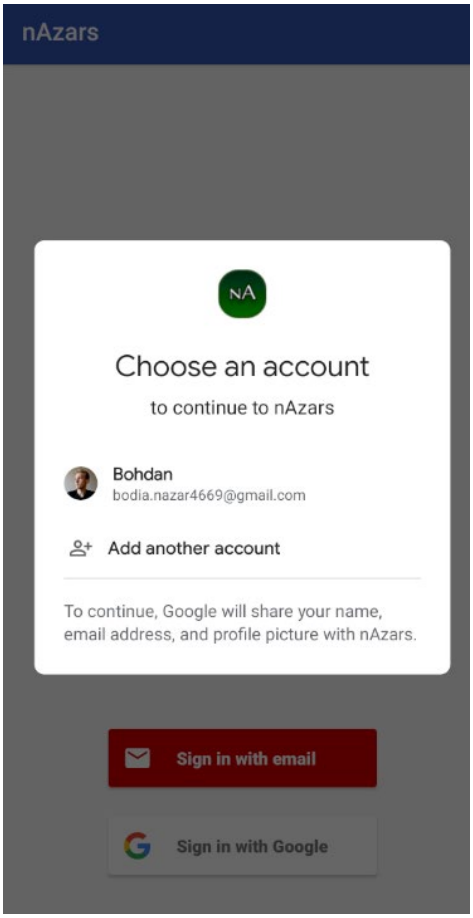

Рисунок В.2. – Вхід за допомогою Google акаунту

Після успішної авторизації користувач потрапляє у головну активність програми, на якій знаходяться спеціальні пропозиції у виді пропозицій магазину і усі товари (рис. В.3), а заодно надсилаються дані користувача до Firebase.

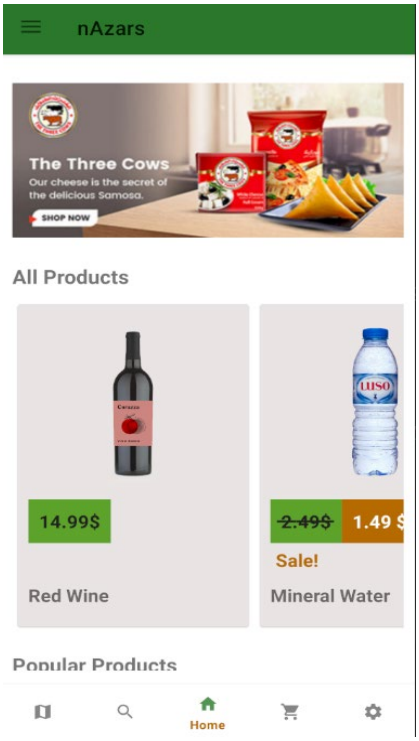

Рисунок В.3. – Екран головної активності застосунку

Тут користувач може гортати пропозиції магазину простими свайпами вправо або вліво і аналогічно з товарами. Тут присутні 3 RecyclerView, які сортуються різними способами. Перший сортує за новизною продукту, другий – за загальною популярністю, а третій – за персональними рекомендаціями користувача (рис. В.4).

Користувач може натиснути на будь-яку картку продукту та потрапити до активності конкретного товару, з усією доступною інформацією про нього (рис. В.5).

У цій активності користувач може поставити оцінку предмету, яка додасть у базу даних значення оцінки продукту, додати відгук на продукт, а також додати його до кошику (рис. В.6).

У діалоговому вікні додання відгуку передбачений захист від небажаних слів, які може ввести користувач. Ця функція перевіряє текст на наявність слів з масиву шаблонів, які не можна буде використовувати для публікації відгуку (рис. В.7).

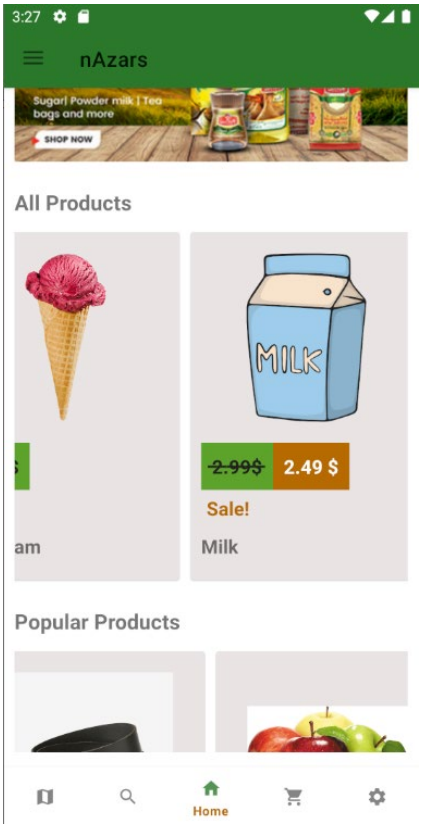

Рисунок В.4. – Екран головної активності

Рисунок В.5. – Екран активності перегляду товару

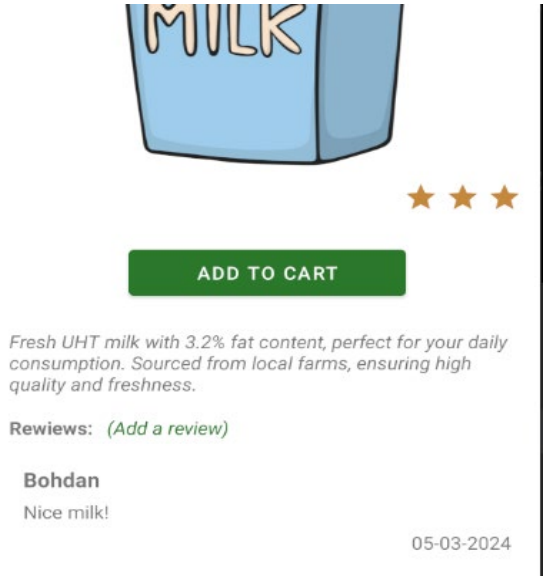

Рисунок В.6. – Додавання користувачем відгуку на товар

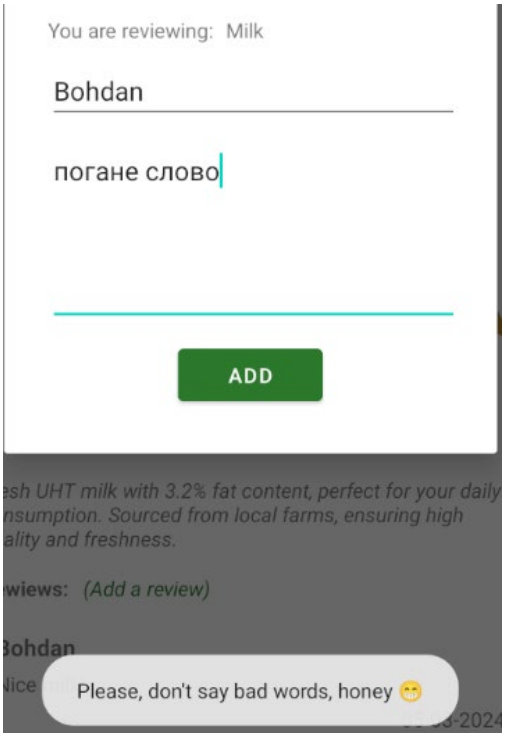

Рисунок В.7. – Перевірка тексту відгуку

Користувач може додати безліч товарів собі у корзину і замовити їх. На цьому фрагменті активності корзини він бачитиме потрібну інформацію про продукт та суму усіх товарів (рис. В.8).

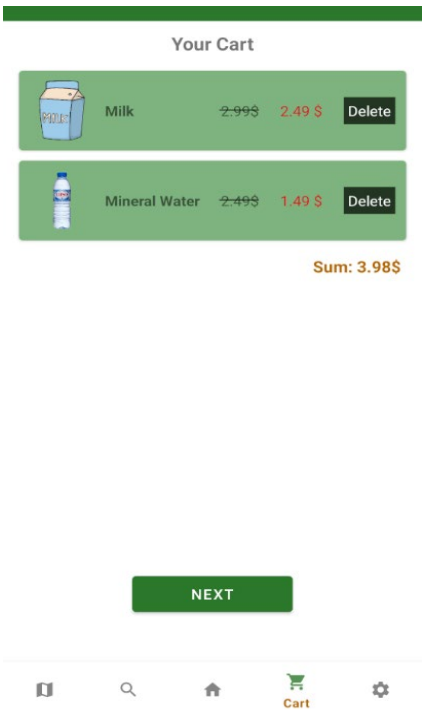

Рисунок В.8. – Екран кошику користувача

Після цього користувач заповнює дані, які не були автоматично підтягнуто з акаунту користувача. У цьому фрагменті[13] присутні перевірка правильності введеної інформації (рис. в.9).

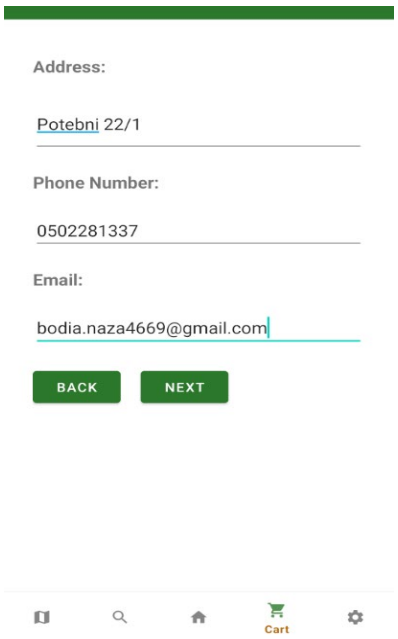

Рисунок В.9. – Перший фрагмент активності кошику

Після натиску кнопки користувач потрапляє до фрагменту вибору способу платежу (рис. В.10).

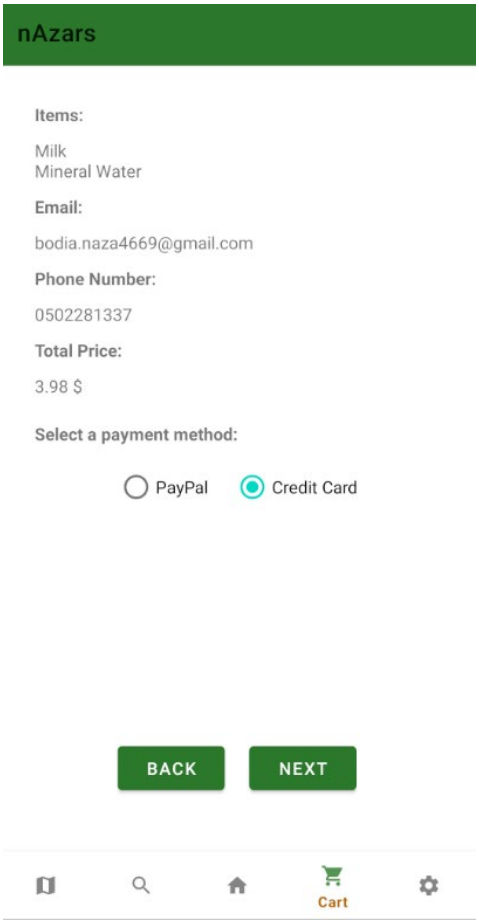

Рисунок В.10. – Другий фрагмент кошику

Після натиснення кнопки "Next" програма надсилає post запит, який перевіряє чи присутня мережа і чи може користувач надіслати його і якщо результат зі сторінки ОК, то користувач бачить вікно успішного замовлення (рис. В.11).

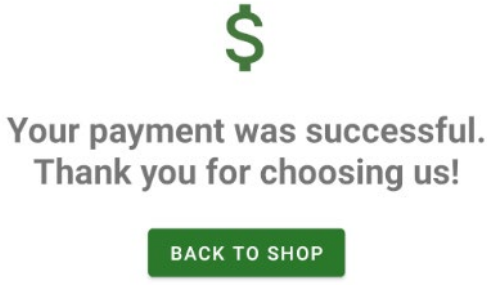

Рисунок В.11. – Екран успішного оформлення замовлення

Для перегляду додаткових функцій користувач може відкрити меню, у якому будуть доступні додаткові функції (рис. В.12).

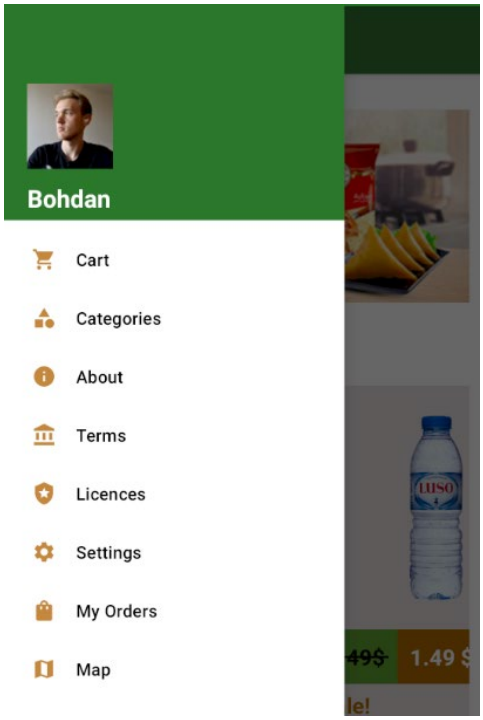

Рисунок В.12. – Бокове меню програми

Натиснувши на кнопку "Мої замовлення", користувач потрапляє на активність, у якій збережні всі його замовлення (рис. В.13).

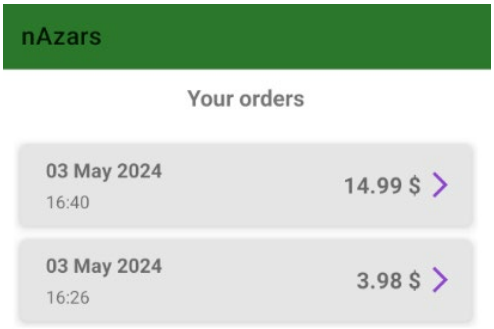

Рисунок В.13. – Екран історії замовлень користувача

Натиснувши на конкретне замовлення користувач потрапляє до активності, у якій він бачить детальні дані про замовлення, тобто електронний чек (рис. В.14).

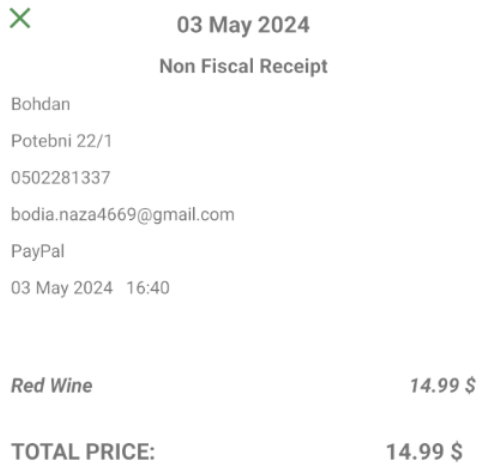

Рисунок В.14. – Екран електронного чека замовлення

Натиснувши на своє ім’я або фотографію в головній активності у боковому меню, користувач потрапляє до активності редагування та перегляду особистого профілю, де можна змінити свої особисті дані. Тут також присутня валідація усіх комірок (рис. В.15).

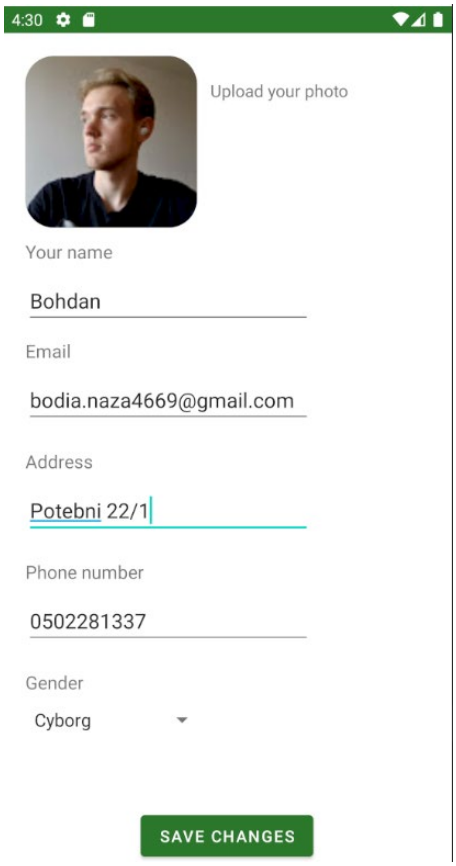

Рисунок В.15. – Екран редагування особистих даних користувача

Натиснувши на текст зміни фото, користувач може змінити фото на інше, використовуючи галерею телефону.

При переході на значок збільшувальної лінзи, користувач потрапляє на активність пошуку товарів. У цій активності користувач може шукати продукти, як за ключовими словами, так і за категоріями (рис. В.16). Також користувач може змінювати загальний вид карток і при натиску на будь-який товар, потрапляти на активність перегляду повної інформації про продукт.

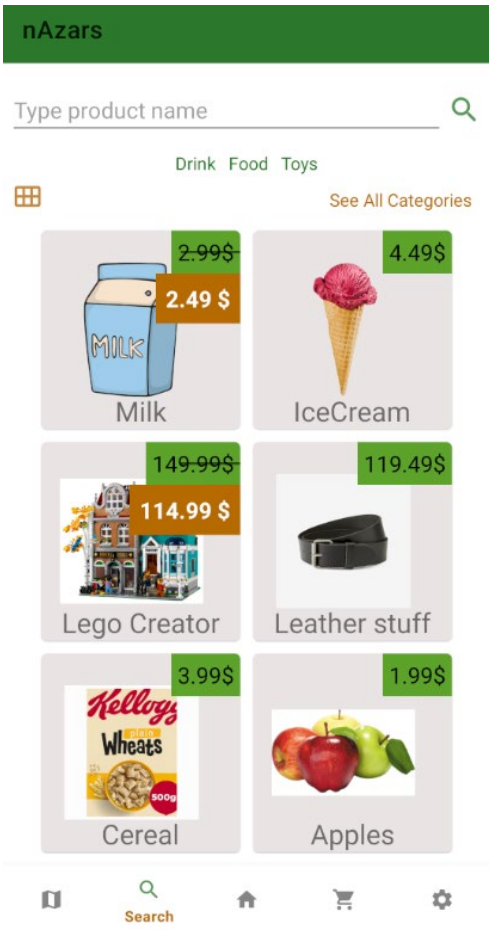

Рисунок В.16. – Екран активності пошуку товарів

При натиску на значок мапи, користувач потрапляє на активність мапи, де він може бачити свою геолокацію в реальному часі, а також розташування усіх магазинів (рис. В.17).

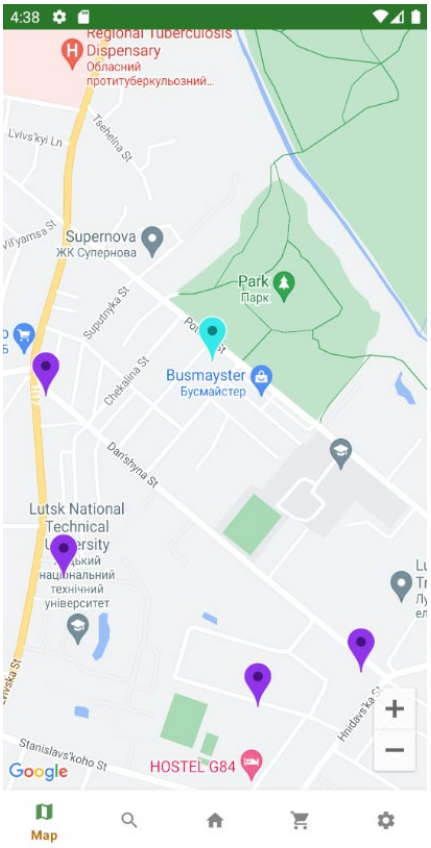

Рисунок В.17. – Екран активності мапи

При натиску на будь-який маркер магазину, користувач відкриє діалогове вікно з усією інформацією про магазин, а саме точний адрес, чи відкритий він, графік роботи на цілий тиждень, а також сервіси [14], який він має (рис. в.18).

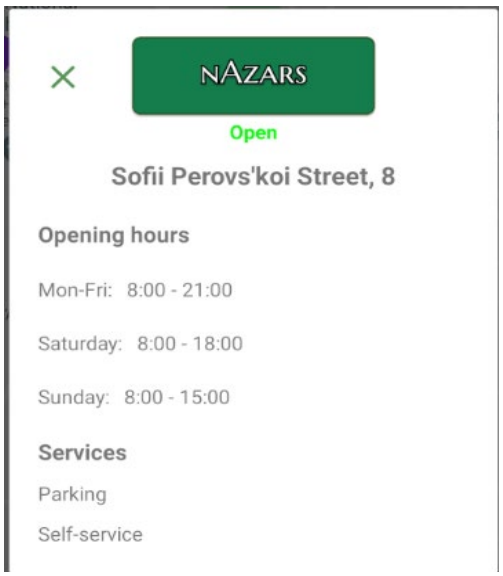

Рисунок В.18. – Детальна інформації конкретного магазину

В активності налаштувань, користувач може вимкнути надсилання повідомлень застосунку на свій телефон, або навпаки увімкнути (рис. В.19). Користувач може вийти з акаунту, з якого автентифікувався в застосунок. Також на цій активності присутні дві кнопки установки знижок, або їхнього вимкнення і вони є створенні в цілях тестування і на повному релізі програми будуть видалені із застосунку.

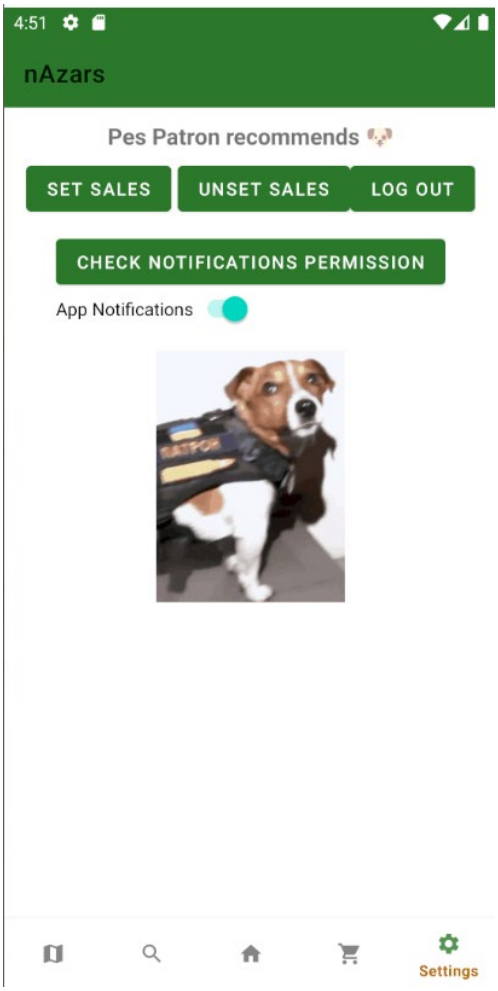

Рисунок ВІ.19. – Екран активності налаштувань застосунку

АНОТАЦІЯ

Назар Б.А. Проєктування та реалізація мобільного застосунку для автоматизації процесів роздрібної торгівлі. Рукопис

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 122 Комп'ютерні науки. Волинський національний університет імені Лесі Українки, Луцьк, 2024 р.

Робота присвячена розробці мобільного застосунку для роздрібного магазину. У роботі розглянуто принципи розробки мобільних застосунків для роздрібних магазинів, досліджено переваги та недоліки кожного типу. Особливу увагу приділено інтерфейсу користувача та їхнього використання у створенні сучасних мобільних застосунків. У роботі зазначено основні особливості реалізації серверної частини мобільних застосунків. Проведено аналіз існуючих аналогічних розробок та визначено їхній вплив на вибір ключового функціоналу для реалізації у розробці. У роботі продемонстровано розробку оптимального застосунку згідно із наявними умовами.

Розроблений продукт підтримує різні функції, включаючи реалізацію різноманітних операцій для користувачів, таких як перегляд товарів, здійснення покупок, перегляд магазинів на мапі, зміна особистих даних користувача, додавання відгуків на продукти, пошук товарів за символами, оцінка продуктів, редагування кошику, обробка платежів та перегляд замовлень. Важливою складовою є обробка взаємодії між користувачами та системою.

У результаті виконання проекту був створений мобільний застосунок, який забезпечує зручний доступ до всього асортименту товарів роздрібного магазину, інформації про акції, швидкість та безпеку для користувачів.

Застосунок протестовано на кількох віртуальних та реальних пристроях Android. Розроблений продукт пропонує можливість обміну даних з хмарним сервером, а також з локальною базою даних.

Ключові слова: мобільний застосунок, електронна комерція, роздрібний магазин, Android, Java, Firebase, RoomDatabase, UI дизайн, замовлення, Google мапа, програмне рішення, алгоритми, функції.