МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,

управління та адміністрування

Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Розробка Android-додатку інтернет-магазина зоотоварів

Виконав студент групи К-18
спеціальності 122 Комп'ютерні науки
Митрофаненко Станіслав
Вікторович

Керівник к.т.н., доцент
Терещенко Тетяна Михайлівна

Консультант

Рецензент к.т.н., доцент
Домаскін О.М.

Одеса 2022

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

ADT – Android Development Tools – Інструменти розробки в ОС Android

API – Application Programming Interface – програмний інтерфейс програми

IDE – Integrated Development Environment – інтегроване середовище розробки

SDK – Software Development Kit – набір засобів розробки

ВСТУП

З кожним роком зростає кількість домашніх вихованців, яких заводять українці. У зв'язку з цим вітчизняний ринок послуг для домашніх тварин вийшов на друге місце за швидкістю зростання в Європі, що у свою чергу сприяє розвитку та утворенню нових виробників. Найкращим місцем реалізації товарів є інтернет-магазин. Він у свою чергу дозволяє без великих витрат збільшити кількість покупців, запропонувавши їм можливості дистанційного замовлення та доставки товарів за місцем вимоги.

Інтернет-магазин – це зручна система демонстрації та продажу товарів, послуг в Інтернеті. Інтернет-магазин підходить для розміщення великої кількості інформації, дозволяє оперативно оновлювати асортименти, чітко контролювати робочі процеси, оновлювати прайс-листи, списки товарів, викладати новини[1]. Даний вид торгівлі не вимагає утримання дорогих торгових площ, великої кількості співробітників, представництва у столиці, що дозволяє досить успішно існувати та розвиватися підприємствам у віддалених регіонах.

Використання стаціонарних комп'ютерів і ноутбуків не завжди зручне у повсякденному житті, тому все частіше на перший план виходять мобільні телефони та планшети. Зараз у світі практично не залишилося людей, які б не користувалися мобільним пристроєм. З урахуванням розвитку інформаційних технологій, з року в рік з'являються все потужніші мобільні пристрої. Отже, суттєво підвищується необхідність до додатків, призначених для цих пристроїв.

Метою кваліфікаційної роботи є розробка мобільного додатку інтернет-зоомагазину «Lapki Market» для зручного та швидкого отримання необхідної інформації про товари та можливість оформляти замовлення потрібних товарів. Для досягнення поставленої мети були сформульовані наступні завдання:

- провести аналіз предметної області
- провести порівняльний аналіз існуючих програм-аналогів;
- обґрунтувати вибір програмних засобів розробки та технологій;

- провести проектування системи з використанням мови UML;
- виконати реалізацію застосунку;
- підготувати інструкцію користувача;
- виконати тестування застосунку.

Структура кваліфікаційної роботи бакалавра складається з вступу, 5 розділів, висновків, переліку посилань на 17 найменувань, 1 додатку. Повний обсяг проекту становить 75 сторінки, містить 35 рисунків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ СИСТЕМ

1.1 Аналіз предметної області

Домашні тварини – брати наші менші, яких свого часу одомашнила людина, а також надає їм дах над головою та їжу[2].

Тварини можуть бути джерелом матеріальних благ, а так само можуть прикрашати проведення часу у вигляді тварини-компаньйона, в обох випадках вони приносять йому велику користь. Вигоду людині приносять сільськогосподарські тварини, які приносять матеріальну вигоду у вигляді їжі (м'ясо, яйця, молоко), матеріалів (шкіра, шерсть), а робочі функції (охорона території, вантажоперевезення) виконує робоча худобу або службові тварини.

Тварини компаньйони є великою категорією домашніх тварин, вони приносять задоволення людині, займають її дозвілля і дозволяють весело в компанії проводити час.

Для міських жителів поняття «домашні тварини» найчастіше асоціюється з цією великою категорією, тобто з «домашніми вихованцями, компаньйонами». Безліч сімей заводячи домашніх тварин, відзначають, що ці тварини знімають стрес, створюють затишок у будинку і заспокоюють, але за кожним домашнім вихованцем потрібен свій особливий догляд.

З розвитком інтернет-технологій більшість оффлайн-магазинів, супермаркетів або повністю перейшли у віртуальний простір, або має там представництво. Це вигідно, тому, що інтернет-шопінг доступний більшій частині покупців, і відкривши інтернет-магазин, можна не обмежуватися межами одного міста або регіону - товари відправляють по всій країні, за кордон.

Багато людей вже давно не можуть уявити своє життя без смартфона. Цей факт визначає як особливості нашого дозвілля, а й принципи сучасного бізнесу. Людям значно простіше шукати інформацію, спілкуватися, дивитися відео та робити покупки у своєму мобільному телефоні. Від таких зручностей відмовитись неможливо. Мобільна комерція увійшла до нашого життя і займає лідерські позиції.

Мобільна комерція - це принцип ведення бізнесу, який передбачає продаж товарів і послуг без фізичної присутності покупців і продавців, за допомогою лише смартфонів, через додатки. Для цього не потрібен фактичний контакт, навіть наявність у продавця фізичного магазину. Але мобільна електронна комерція передбачає не просто сайт, на який можна зайти, використовуючи браузер (що є практично для будь-якого інтернет-магазину і через смартфон), а програма, яка встановлюється на смартфон і робить доступ до покупок ще більш простим і зручним.

Якщо говорити про позитивні сторони мобільної комерції, то першою є доступність клієнтів. Користувачі отримують зручний та постійний доступ до можливості купувати товар у будь-який час з будь-якої точки на Землі, поки у них є вихід в інтернет.

Комфорт і зручність є другою позитивною стороною, бо клієнти не потребують використання браузера для пошуку сайту магазину. Варто відзначити, що існує багато магазинів в інтернеті, які погано адаптовані під мобільні телефони, тому виникають незручності.

Ще однією позитивною стороною є швидкість покупок у мобільному додатку. У програмі можна зберегти свої дані, що дозволить при подальших покупках не вводити їх повторно щоразу. Наприклад, це інформація про самого покупця, його платіжні дані, а також доставка. Клієнти найчастіше купують товари та мають бажання повернутися знову в магазин, якщо їм було комфортно і вони з легкістю, і швидко оформляли покупки. Звідси випливає дуже важлива перевага – лояльність покупців. Вірність бренду є результатом задоволеності покупців товарами обраного ними бренду, що призводить до зростання обсягів продажу цих товарів.

Також варто відзначити, що мобільні додатки мають можливість пропонувати клієнтам товари, ґрунтуючись на їх покупках та уподобаннях, при цьому давати користувачам додаткові знижки та персональні пропозиції для них, використовувати систему бонусів, а також нагадувати покупцям про товари в їх

кошиках та обраному. Клієнти цінують турботу про себе та індивідуальні пропозиції .

1.2 Постановка задачі

Перед початком розробки мобільного додатка потрібно визначити головні цілі та завдання для вирішуваної проблеми, після цього можна буде розпочинати проектування. Насамперед сформулюємо короткий опис поставленого завдання.

Найменування завдання – це розробка мобільного додатка інтернет-магазину. Метою магазину є надання всієї необхідної інформації про потрібні товари, а також можливість їх замовлення.

Завдання або функції застосунку:

- Можливість шукати товар за критеріями;
- Можливість отримати усю потрібну інформацію про товар;
- Можливість додавати товари до кошика;
- Можливість виконувати замовлення товарів в пару натискань;
- Можливість переглядати усі замовлення користувача;

За результатами проведеного аналізу були розроблені вимоги до мобільного застосунку та розроблено мобільний додаток «Lapki Market» для просування та продажу товарів, який додатково зменшує витрати на рекламу та сприяє розширенню ринку збуту товару для тварин, поповненню клієнтської аудиторії і, як наслідок, збільшенню прибутку, що є головною метою бізнесу. Дана розробка може бути перспективною та корисною з комерційної та споживчої точки зору.

1.3 Аналіз існуючих програмних систем

На даний час існують різноманітні Android-додатки інтернет-магазинів зоотоварів, деякі спеціалізуються на товарах для конкретних тварин, інші охоплюють майже всі види. Розглянемо деякі з них.

Інтернет-зоомагазин ZooZooZoo (рис.1.1) є мобільним додатком мережі зоомагазинів в м.Біла Церква, який дозволяє переглянути інформацію про наявні товари, можна прочитати детальну інформацію про товар, а також зробити замовлення на товар для тварин у межах міста, а також отримати Новою поштою товар в інші міста. У додатку є великий асортимент товарів для різних видів домашніх тварин, каталог має зручні категорії (рис.1.2). Основні недоліки магазина ZooZooZoo це неможливість сортувати товари у розділах наприклад за ціною, а також весь текст у додатку написаний відразу трьома мовами: англійська, українська, російська.

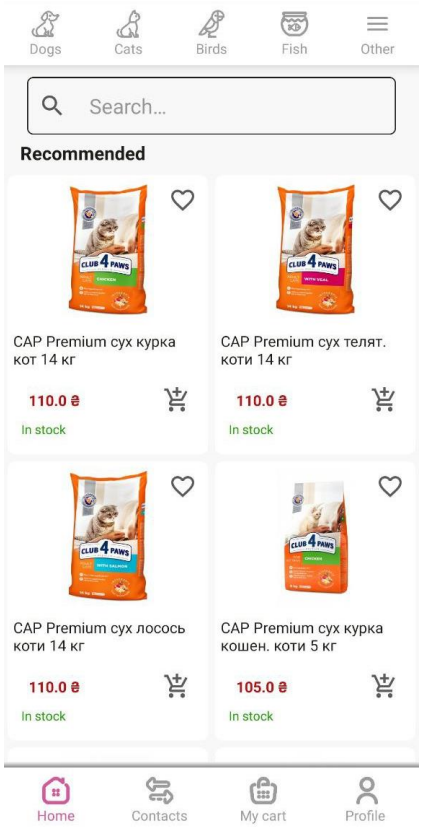


Рисунок 1.1 – Головна сторінка магазина ZooZooZoo

Рисунок 1.2 – Категорії магазина ZooZooZoo у розділі Собаки

Інтернет-зоомагазин VMISKE (рис.1.3) дозволяє прочитати детальну інформацію про потрібні товари, а також зробити швидке замовлення потрібного товару або замовити усі товари з кошика.

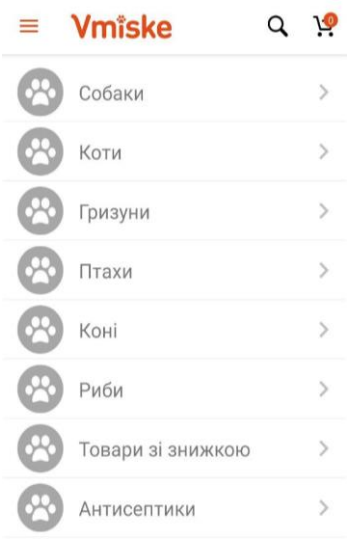


Рисунок 1.3 – Головна сторінка інтернет-магазина VMISKE

Є великий асортимент товарів для різних видів домашніх тварин та є багато категорій (рис.1.4).

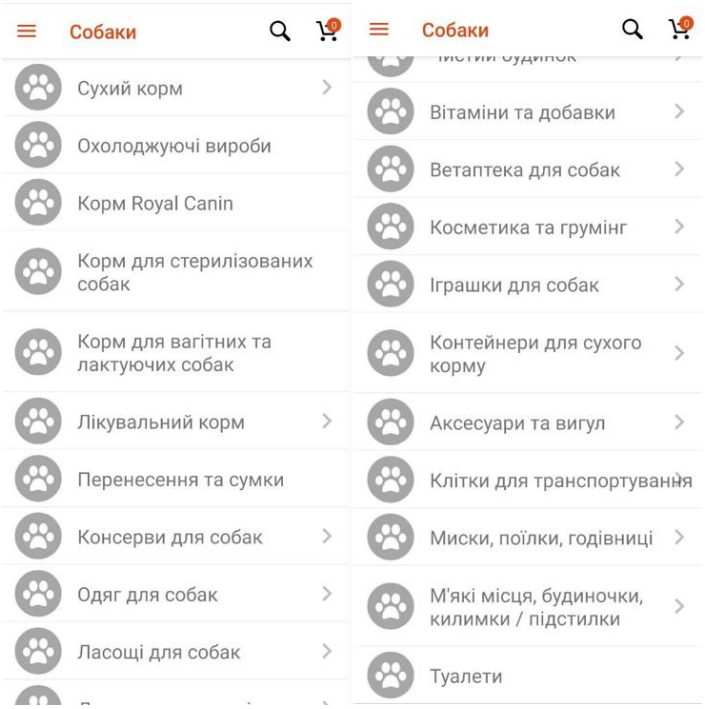


Рисунок 1.4 – Категорії магазина VMISKE у розділі Собаки

Магазин має можливість сортування та фільтрацію (рис.1.5).

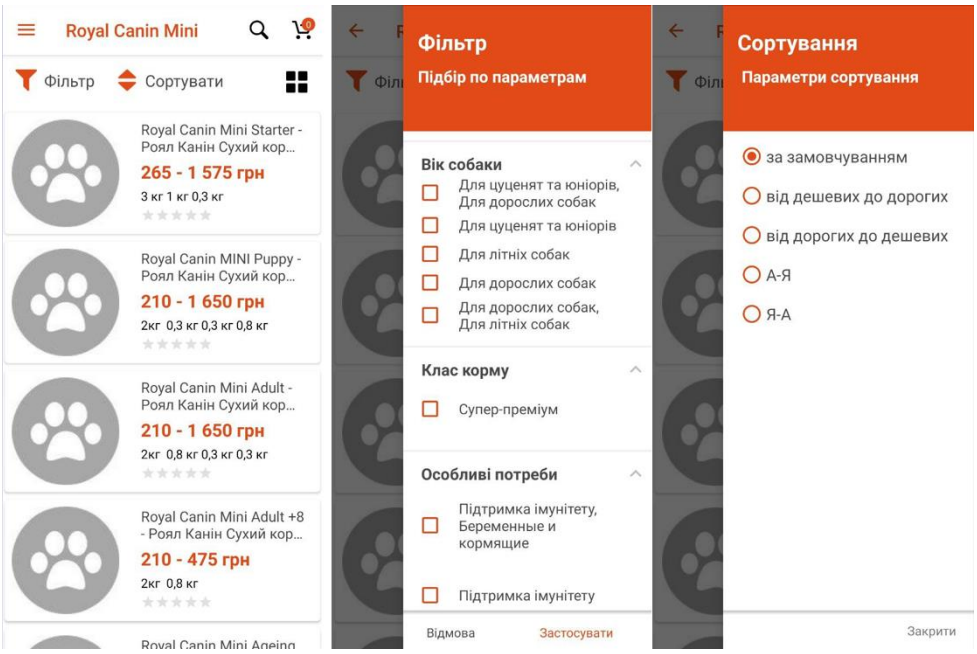


Рисунок 1.5 – Фільтрація та сортування у магазині VMISKE

Інтернет-зоомагазин VMISKE має можливість змінювати мову у налаштуваннях (рис.1.6).

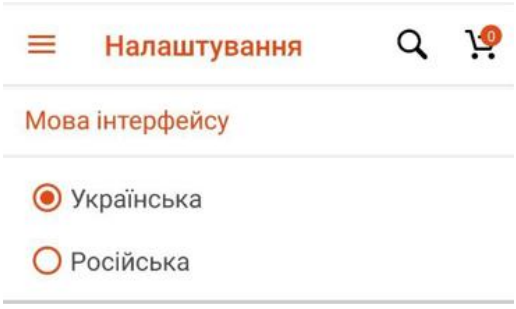


Рисунок 1.6 – Можливість змінити мову у магазині VMISKE

Основні недоліки магазина VMISKE – це відсутність картинок товарів, а також у деяких місцях текст не вирівняно правильно по сторінці, у деяких місцях іконки відображаються поверх тексту.

Інтернет-магазин Pet Shop (рис.1.7) дозволяє прочитати усю потрібну людині інформацію про товари, а також зробити замовлення. Має декілька розділів. Додаток написано англійською мовою (рис.1.8).

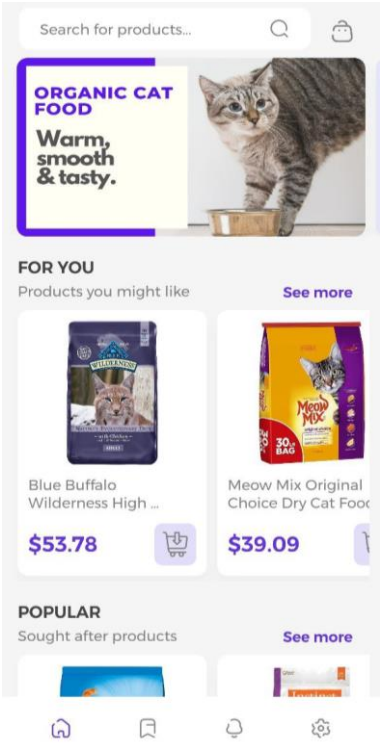


Рисунок 1.7 – Головна сторінка інтернет-магазина Pet Shop

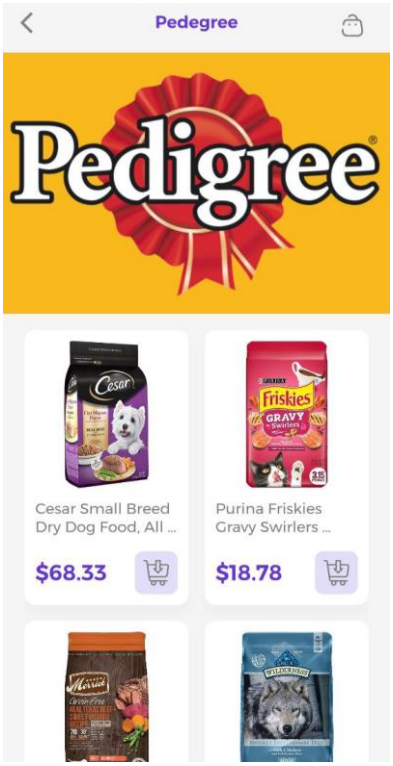


Рисунок 1.8 – Розділ магазина Pet Shop за маркою корма

Основні недоліки магазина Pet Shop це дуже маленький асортимент товарів для різних видів домашніх тварин, майже немає категорій, неможливість сортувати товари, дуже довго завантажується інформація про товари

Таким чином, під час виконання першого розділу бакалаврської роботи була обрана предметна область, в рамках якої розглянуті програмні аналоги і їх особливості. Переваги та недоліки розглянутих аналогів були взяті до уваги при розробці програмного продукту.

2 ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ

2.1 Функціональні та нефункціональні вимоги

Перед початком розробки застосунку були визначені функціональні та нефункціональні вимоги до нього.

До функціональних вимог належать:

- наявність існуючих БД товарів і клієнтів;
- відображення списку товарів та інформації про обраний;
- можливість реєстрації та входу у додаток.
- можливість здійснити замовлення потрібного товару.

До нефункціональних вимог належать:

- смартфон з ОС Android версією 5.0+
- підключення до Інтернету.

2.2 Обґрунтування вибору операційної системи

На рис. 2.1 представлена діаграма, що показує частки, які за даними на березень 2021 - березень 2022 займають популярні мобільні ОС на ринку України [3].

Згідно до статистики від компанії Statcounter [4], особистому досвіді та діаграмі на рис. 2.1 були виділені дві найбільш популярні ОС, які необхідно розглянути: iOS та Android.

Мобільна операційна система від Apple управляється легко. Вона інтуїтивно зрозуміла і шлях до більшості опцій легко простежується. Найголовніше – запам'ятати кілька основних принципів управління. Свайп вправо – бібліотека програм, вліво – віджети, вниз – повідомлення, вгору – список відкритих програм.

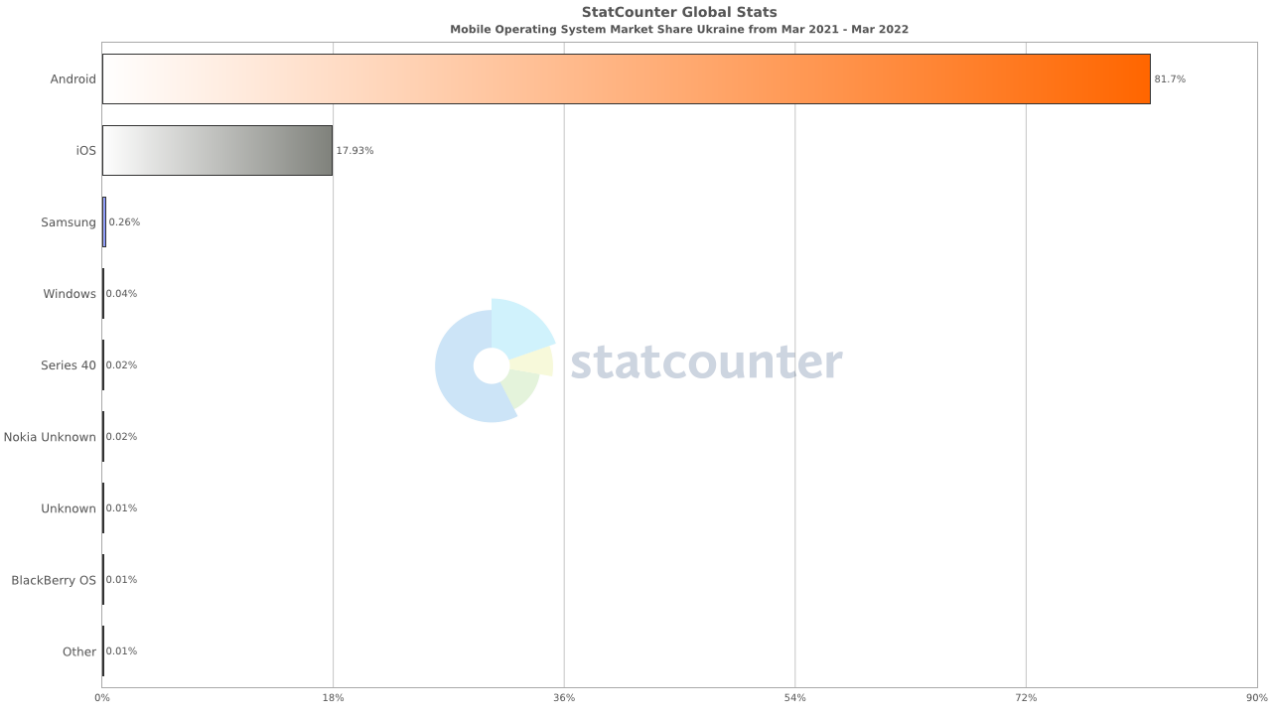


Рисунок 2.1 – Діаграма співвідношення мобільних ОС на ринку України смартфонів від Statcounter

Але ця простота легко пояснюється: налаштувати можна не так багато параметрів. В основному найважливіше: шпалери, паролі для входу в систему, передачу даних у хмару.

Apple розраховує на те, що користувачі розберуться у всьому самі - жодних інструкцій у комплект з пристроями не входить. Для просунутого користувача це справді просто, але новачок може мати справу з проблемами в керуванні смартфоном.

Apple уважно ставиться до своїх продуктів та безпеки системи. Встановити можна лише програми, які доступні в офіційному магазині AppStore. Для будь-яких інших маніпуляцій потрібно перепрошувати телефон, а це автоматично анулює гарантію. Через це вибір програм обмежений[5].

Крім того, не всі програми доступні у всіх країнах, а якщо ви хочете спробувати програму, яка ще не вийшла у вашому домашньому регіоні, то єдиний спосіб – переставити магазин програм на закордонний. При цьому всі встановлені

програми, навіть якщо вони є у магазині вибраної країни, перестають оновлюватися.

Для того, щоб опублікувати свій застосунок в App Store, необхідно мати аккаунт розробника Apple ($99 в рік) та фізичний комп'ютер з macOS для створення проектів в Xcode. До того, кількість користувачів iOS набагато менше за кількість користувачів Android, тож постає питання окупності.

Головна відмінність Android – це відкритість системи. І справа не тільки в тому, що її можна майже повністю переоформити, а в тому, що настройки стосуються навіть найдрібніших елементів: типів повідомлень, організації робочого столу, оновлення окремих компонентів системи.

Крім того, Android також має розширені налаштування, в які входять детальні налаштування конфіденційності.

Android дозволяє встановлювати програми відразу з декількох магазинів: Google Play, офіційний магазин Android, магазин інтерфейсу користувача, доданого виробником і будь-який apk-файл - інсталяційний файл програми.

Щоб отримати доступ до програми з магазину іншої країни, достатньо просто знайти в пошуковій системі apk цікавого додатка і встановити його. Оновлювати його можна у такий же спосіб. Але гарантувати ідеальну роботу таких програм творці системи, зрозуміло, не можуть.

Тим не менш, Android дозволяє встановлювати різні програми, у тому числі ті, що змінюють налаштування системи: дизайн, додаткові віджети і т.д. Вибір програм набагато більше ніж у IOS.

Обсяг завантажень додатків у всьому світі становив 36,1 млрд наприкінці 2021 р. Кількість завантажень додатків загалом у 4 кварталі 2021 року у всьому світі на 2,7 відсотка більше, ніж минулого року[6]. Простір мобільних додатків все ще знаходиться в стані трансформації у зв'язку з глобальною пандемією, що триває, по всій планеті. На перше місце наприкінці 2021 року вийшли категорії, такі як покупки, фінанси та розваги, решта категорій все ще намагаються відновитися після наслідків.

Щоб опублікувати свою програму в Google Play, необхідно одноразово сплатити збір $25, що значно дешевше в порівнянні з Apple Store[7]. Варто відзначити безліч інструментів для розробки додатків для Android. Враховуючи перелічені особливості операційних систем iOS та Android, перевага надається системі Android, на якій і відбуватиметься розробка мобільного додатка.

2.3 Вибір середовища розробки

Популярними середовищами розробки під ОС Android є Android Studio, Eclipse, IntelliJ IDEA.

Android Studio — інтегроване середовище розробки виробництва Google, за допомогою якого розробникам стають доступні інструменти для створення програм на платформі Android OS. Android Studio можна встановити на Windows, Mac та Linux. Це – офіційне середовище програмування. Отримало середовище подібний статус у 2014 році. До цього реалізація поставленої задачі здійснювалася через Eclipse. Воно є повністю інтегрованою платформою.

Android Studio - утиліта, в якій без проблем можна створити програми для Андроїд своїми руками. Вона пропонує:

- середовище для написання програмних кодувань;
- перегляд етапів роботи з ПК (без попередньої ініціалізації на той чи інший гаджет);
- аналізатори APK;
- режим порівняння двох пакетів;
- редактор так званих макетів, що дозволяє налаштовувати інтерфейси;
- профіль у реальному часі;
- оптимізацію без коригування вихідного коду за допомогою Android Studio Bundle.

Android Studio - це IDE, що включає SDK. Він важливий для утиліт мобільного типу. Пакет завантажується у вигляді архіву для кожної операційної системи окремо[8].

Інтернет пропонує величезну кількість софту для тих чи інших потреб. Програмери можуть знайти там різні середовища та конструктори. Але за деякі потрібно платити. Android Studio це те, що купити неможливо. Пов'язано це з тим, що програма розповсюджується абсолютно безкоштовно. Завантажити його можна із офіційного сайту розробника[9].

До сильних сторін Android Studio відносять:

- безпека;
- офіційність;
- простоту використання;
- кросплатформність;
- наявність власного зручного редактора коду;
- велика бібліотека з готовими рішеннями
- можливість опрацювання програм для портативних ПК, приставок, мобільних пристроїв;
- підтримку кількох мов програмування (включаючи java та Сі-сімейство);
- безкоштовне розповсюдження в Інтернет.

Eclipse Java IDE — відкрите інтегроване середовище розробки мовою програмування Java, яке розповсюджується та підтримується Eclipse Foundation.

Основною метою Eclipse є підвищення продуктивності процесу розробки програмного забезпечення. Сам же Eclipse написаний мовою програмування Java[10].

Eclipse можна рекомендувати як IDE для програмістів Java незалежно від виду проекту: веб-додаток, автономна програма, SOAP та REST веб-служби або ESB-компоненти. Eclipse прискорює розробку на Java. Корисна в малих, середніх та великих організаціях завдяки простоті використання та підтримці коду спільнотою користувачів Eclipse.

Особливості платформи Eclipse:

- Кросплатформенність – працює під операційними системами Windows, Linux, Solaris та Mac OS X.
- Використовуючи Eclipse можна програмувати на багатьох мовах, таких як Java, C і C++, PHP, Perl, Python, Cobol та інші.
- Є фреймворком для створення інших інструментів і пропонує великий набір API для створення модулів.
- Використовуючи підхід RCP (Rich Client Platform), Eclipse є інструментом для створення практично будь-якого клієнтського програмного забезпечення.

Робота над проектом Eclipse ведеться в декількох напрямках, три основні - робота над платформою Eclipse, розробка Java IDE, розробка плагінів для розширення функціональності Eclipse. Гнучкість та розширюваність досягається завдяки модульності платформи.

Для розробки Android- застосунків в Eclipse необхідно підключити плагін Android Development Tools. Однак плагін Eclipse ADT більше не підтримується, згідно з оголошенням з 2014 року. Тому Google рекомендує переключитися на використання Android Studio.

IntelliJ IDEA – це IDE, інтегроване середовище розробки (комплекс програмних засобів, який використовується для написання, виконання, налагодження та оптимізації коду) для Java, JavaScript, Python та інших мов програмування від компанії JetBrains. Відрізняється великим набором інструментів для рефакторингу (перепроектування) та оптимізації коду.

У IntelliJ IDEA можна розробляти програми на Java та інших мовах, що працюють на платформі віртуальної машини Java, - Kotlin, Scala і Groovy.

Хоча середовище спочатку створювалось для максимальної оптимізації Java-розробки, зараз у ній є опції для роботи з більшістю затребуваних мов програмування, причому деякі з інструментів використовують технологію машинного навчання. IDE надає інтелектуальну допомогу під час написання коду:

- виконує глибокий аналіз та створює віртуальну карту проекту;

- виявляє помилки та пропонує варіанти виправлення;
- автоматично доповнює код з огляду на контекст;
- проводить валідацію (перевірку на відповідність стандартам) коду;
- виконує рефакторинг коду - робить його простішим і зрозумілішим;
- підтримує роботу із вставками, написаними іншими мовами програмування;
- дозволяє використовувати шаблони для вставки фрагментів коду, що повторюються;
- пропонує оптимізацію за допомогою профілювача - інструмента, який аналізує продуктивність коду та оцінює навантаження на процесор та оперативну пам'ять.

В області налагодження та тестування коду середовище також може запропонувати кілька цікавих рішень:

- інструменти для проведення автоматичних тестів та формування аналітики, яка показує, який обсяг коду протестовано;
- налагоджувач, що показує значення змінних у вихідному коді;
- можливість вибрати метод налагодження;
- вбудований декомпілятор - інструмент для перетворення виконуваного двійкового коду з jar-файлів в Java-код, що читається.

Незважаючи на те, що є велика кількість IDE для розробки додатків Android, Android Studio вважається абсолютним лідером. Підтримувана компанією Google, ця IDE надійна, завжди актуальна і готова до роботи. Більшість розробників роблять вибір саме на її користь. Таким чином, за підсумком аналізу існуючих середовищ розробки, було обрано середовище Android Studio для розробки Android-додатку інтернет магазину.

3 ТЕХНОЛОГІЯ РОЗРОБКИ ANDROID-ДОДАТКУ

3.1 Мова програмування

Варто зазначити, що розробляти програми під Android можна за допомогою різних фреймворків та мов програмування. Так, як мови програмування можуть застосовуватися Java, Kotlin, Dart (фреймворк Flutter), C++, Python, C# (платформа Xamarin)

Для будь-якого розробника мобільних додатків на Android, першою і найкращою мовою програмування поки що залишається Java, так він підтримується компанією Google і більшість програм у Google Play побудовані саме на ньому.

Сама Java була розроблена компанією Sun Microsystems (надалі придбаною компанією Oracle) ще в 1995 році, і вона досі використовується для широкого спектру програмних додатків. Код Java виконується віртуальною машиною, яка працює на пристроях Android та інтерпретує код. Java це офіційна мова для розробки під Android, яка підтримується Android Studio[11].

Що стосується функціональності, Java відноситься до однієї з найпотужніших мов. Вона дозволяє реалізувати найскладніші завдання та проекти, тому користується такою популярністю. Тут використовуються як Java класи, так й XML-файли.

На Java посилається більшість офіційної документації Google, а знайти платні та безкоштовні бібліотеки та керівництва не складе труднощів - їх безліч.

3.2 Середа розробки Android-додатку

Android Studio — інтегроване середовище розробки виробництва Google, за допомогою якого розробникам стають доступні інструменти для створення програм на платформі Android OS. IDE можна завантажити з офіційного сайту та користуватися безкоштовно. У ній є макети для створення UI, з чого зазвичай починається робота над додатком. Studio містить інструменти для розробки

рішень для смартфонів і планшетів, а також нові технологічні рішення для Android TV, Android Wear, Android Auto, Glass і додаткові контекстуальні модулі.

Концепт безперервної інтеграції, який дозволяє відразу виявляти наявні проблеми, закладено в основі робочого процесу Android Studio. Можливість результативного зворотного зв'язку з розробниками забезпечується тривалою перевіркою коду. Ця функція дозволяє швидше опублікувати версію мобільного додатка в Google Play App Store. Для цього є також підтримка інструментів LINT, Pro-Guard і App Signing. Є підтримка всіх платформ Android, починаючи з Android 1.6.

У програмі реалізовані всі сучасні засоби для пакування коду, його маркування. Затребувана багатьма авторами ПЗ функція Drag-n-Drop, що полегшує перенесення компонентів у середу розробки безпосередньо.

Крім самого середовища Android Studio для розробки також знадобиться набір інструментів, який називається Android SDK. Наприклад, якщо раніше Android SDK ще не було встановлено, то при першому зверненні до Android Studio вона запропонує встановити додаткові інструменти, які необхідні для розробки. Насамперед це Android SDK і низка додаткових компонентів.

Android SDK – це додатковий набір інструментів Android Studio, які допомагають написати код, запустити тестування та налагодження, перевірити роботу програми на різних версіях операційної системи та оцінити результат у реальному часі. Також пакет дозволяє користувачам отримувати інформацію про стан операційної системи, читати логи та виявляти помилки. Через SDK для Андроїд можна відновлювати програмну оболонку та встановлювати сторонні прошивки.

Набір складається з пакетів, необхідних для створення програм. Ось наприклад основні, якими ми скористуємося.

Android SDK Platform Tools. До групи входять такі інструменти взаємодії з Android як Android Debugging Bridge (ADB), Fastboot, Systrace та інші. ADB допомагає знайти помилки у роботі додатків, встановити APK на смартфон.

Fastboot - активувати швидке завантаження для керування мобільним пристроєм з комп'ютера, перепрошувати гаджет, настроїти доступ, параметри роботи операційної системи. Systrace — отримати інформацію про запущені процеси, простежити за активністю та обсягом даних, які надіслані через мережу.

Android SDK Build Tools. Компоненти SDK використовуються для створення коду. Zipalign дозволяє оптимізувати файл APK, AAPT2 – проаналізувати, проіндексувати та скомпілювати ресурси у двійковий формат під платформу Android, Apksigner – підписати пакет APK за допомогою закритого ключа.

Емулятор Android. Інструмент допомагає протестувати програми та випробувати функції останніх версій Android. Дозволяє перевірити коректну роботу програми на пристроях з різними екранами, із різними співвідношеннями сторін. Відмінна риса емулятора – перегляд приблизних показників продуктивності при запуску програми на найпопулярніших пристроях.

Gradle - це система автоматизації збірки з відкритим вихідним кодом, яка ґрунтується на концепціях Apache Ant та Apache Maven та представляє доменно-орієнтовану мову (DSL) на основі Groovy замість XML-форми, що використовується Apache Maven для оголошення конфігурації проекту[12].

Gradle дозволяє керувати шляхом класів ваших проектів. Він може додавати файли JAR, каталоги або інші проекти у спосіб збирання вашої програми. Він також підтримує автоматичне завантаження залежностей вашої бібліотеки Java.

В Android Studio Gradle - це інструмент складання, що настроюється, використовується для створення пакетів Android (файлів apk) шляхом управління залежностями і надання логіки складання, що настроюється. Файл apk підписується та відправляється на пристрій за допомогою ADB (Android Debug Bridge), де він запускається.

3.3 Використання бібліотек

Бібліотеки в Android дуже корисні, оскільки виконують у собі різні трудомісткі завдання, звільняючи розробника від реалізації у своєму додатку.

Більшості програм потрібні підключення до зовнішніх служб для доступу та обміну даними. Зазвичай це відбувається через API REST та HTTP-клієнт у додатку.

Бібліотека Retrofit - це відома серед Android-розробників бібліотека для мережевої взаємодії, деякі навіть вважають її до певної міри стандартом. Причин для такої популярності безліч: бібліотека відмінно підтримує REST API, легко тестується і налаштовується, а запити по мережі з її допомогою виконуються дуже просто.

Retrofit2.0 – це мережна структура HTTP-запитів, заснована на OkHttp, яка дуже підходить для формату RESTful URL. Якщо коротко OkHttp – це полегшена інфраструктура мережевих запитів, надана компанією Square. OkHttp – це клієнтська HTTP-бібліотека Android від компанії Square, яка скорочує кількість необхідних кроків та дозволяє приділяти більше часу важливим областям програми. RESTful – це стиль та дизайн архітектури програмного забезпечення. Він надає набір принципів та обмежень проектування. RESTful в основному використовується для програмного забезпечення взаємодії клієнта та сервера. Програмне забезпечення, розроблене на основі цього стилю, може бути більш коротким, багаторівневим і більш простим для реалізації кешування і т.д[13].

Retrofit 2.0 налаштовує мережеві параметри за допомогою анотацій і може створювати фактичні HTTP-запити відповідно до наших правил, може гнучко задавати URL, заголовок, тіло запиту, значення, що повертається і т.д. Бібліотекою зручно користуватися для запиту на різні веб-сервіси з командами GET, POST, PUT, DELETE. Вона також підтримує синхронні та асинхронні дані запиту та може гнучко зіставлятися з конвертером. Різні аналітичні середовища, такі як режим адаптивного програмування gson і RxJava2, сильно відрізняються один від одного по дизайну, гнучкості та зручності у використанні, а також за продуктивністю, оскільки вони інтегровані з OkHttp, мають переваги хорошої продуктивності та високої швидкості обробки.

Розробка свого власного функціоналу для відображення та завантаження медіа на Java може виявитися справжньою проблемою, адже потрібно буде подбати про кешування, декодування, управління мережевих з'єднань, потоків, обробку винятків та багато іншого.

Бібліотека Glide - це проста у використанні, добре спланована, добре документована і ретельно протестована бібліотека, яка допоможе зберегти багато часу - і позбавити розробника від багатьох проблем. Вона є однією з найпопулярніших бібліотек з відкритим вихідним кодом для роботи із зображеннями, оскільки реалізує асинхронне завантаження зображень, їх обробку та кешування, а також вміє працювати з GIF-зображеннями та відео[14].

За замовчуванням Glide використовує реалізацію HttpURLConnection для завантаження зображень через Інтернет. Тим не менш, Glide також надає плагіни для інших популярних мережевих бібліотек, таких як Volley або OkHttp.

Додаємо Glide у проект як і більшість залежностей у Gradle-проект, додавши один рядок у build.gradle:

compile 'com.github.bumptech.glide:glide:версія'

4 ПРОЕКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

Проектування інформаційної системи є наступним важливим етапом після аналізу предметної області, аналогів програмних продуктів, а також вибору засобів розробки. UML-діаграми варіантів використання, активності й послідовності, а також розробка макетів інтерфейсу додатка створюються на даному етапі.

4.1 Створення UML-діаграми варіантів використання системи

Діаграма варіантів використання (або діаграма прецедентів) – це така діаграма, що описує, який функціонал проектованої програмної системи, що розробляється, доступний кожній групі користувачів. Суть діаграми варіантів використання полягає в тому, що систему представляють як групу акторів, які за допомогою варіантів використання взаємодіють із нею.

Актором є стилізований чоловічок, що позначає набір ролей користувача, що взаємодіє з деякою сутністю. Актор взаємодіє з системою для вирішення деяких завдань. Актором може бути людина, клас, інша система, пристрій, програмний засіб або зовнішня сутність.

У діаграмі варіантів використання взаємодіють два актори – невідомий користувач, авторизований користувач, адміністратор, а також на діаграмі представлена хмарна база даних. Невідомий користувач – це людина, яка використовує мобільний додаток, не пройшовши процедуру авторизації або реєстрації. Авторизований користувач – це людина, яка успішно пройшла процедуру авторизації чи реєстрації. Адміністратор – людина, яка має можливість напряму взаємодіяти з будь-якими об’єктами у хмарній базі даних.

Наступним кроком після того, як були визначені актори системи є необхідність сформувати перелік усіх потрібних варіантів використання, з якими визначені актори будуть взаємодіяти. Серед основних варіантів використання системи: з боку невідомого користувача реєстрація та авторизація, з боку

авторизованого користувача – можливість переглядати інформацію про магазин, перегляд переліку товарів, пошук товарів за назвою, пошук товарів за категоріями, можливість сортувати товари та переглядати інформацію про обраний товар, додавання товарів до корзини, виконання замовлення, перегляд усіх своїх замовлень, оплата замовлення, а також керування особистими даними, з боку адміністратора – це додавання нових категорій та товарів, оновлення інформації про них, видалення їх з бази даних.

За результатами сформованих варіантів використання та акторів системи було розроблено діаграму варіантів використання, яку наведено на рис. 4.1.

Рисунок 4.1 – Діаграма варіантів використання (прецедентів) системи

4.2 Створення діаграми діяльності (активностей)

Для моделювання процесу виконання операцій в мові UML використовуються діаграми діяльності. На діаграмі діяльності відображається логіка або послідовність переходу від однієї діяльності до іншої, при цьому увага фіксується на результаті діяльності. Сам же результат може привести до зміни стану системи або повернення деякого значення. Для системи була створена діаграма діяльності, що відображає послідовність дій користувача, а також поведінку системи авторизації користувача (рис. 4.2).

Рисунок 4.2 – Діаграма діяльності (Авторизація Користувача)

А також була створена діаграма діяльності, яка відображає основний процес проектованої системи – пошук товару та виконання його замовлення (рис. 4.3).

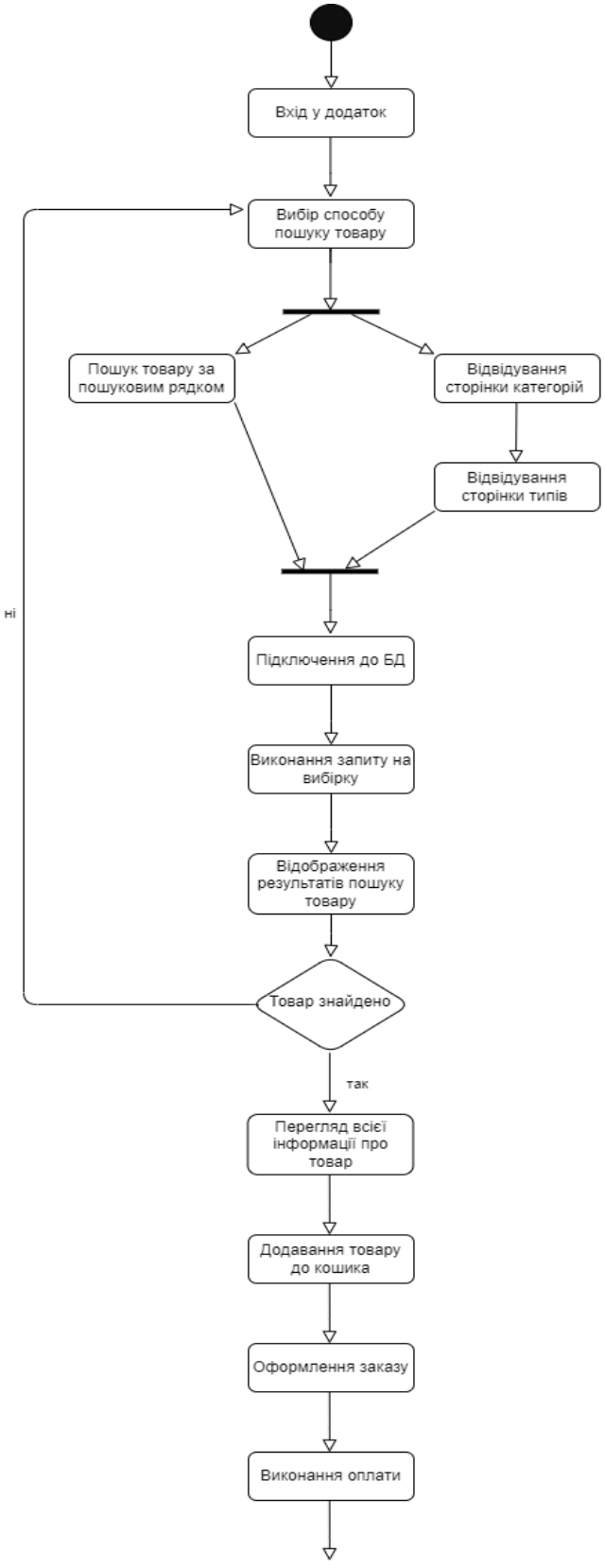


Рисунок 4.3 – Діаграма діяльності (активностей)

4.3 Створення діаграми послідовності

Також була створена діаграма послідовностей. Такі діаграми використовуються для уточнення діаграм прецедентів і більш детального опису логіки сценаріїв використання.

Головна послідовність сценарію розпізнавання зображення складається з взаємодій, які відображені на діаграмі на рис. 4.4: користувач обирає потрібний товар у додатку Lapki Market та виконує замовлення, яке додаток передає до БД на перевірку, пройшовши усі потрібні перевірки виконується оплата замовлення за допомогою Payment Api, яка повертає усі потрібні дані про сплату у БД, а та у свою чергу повертає усю потрібну інформацію про замовлення до додатку, де ця інформація відображається для користувача.

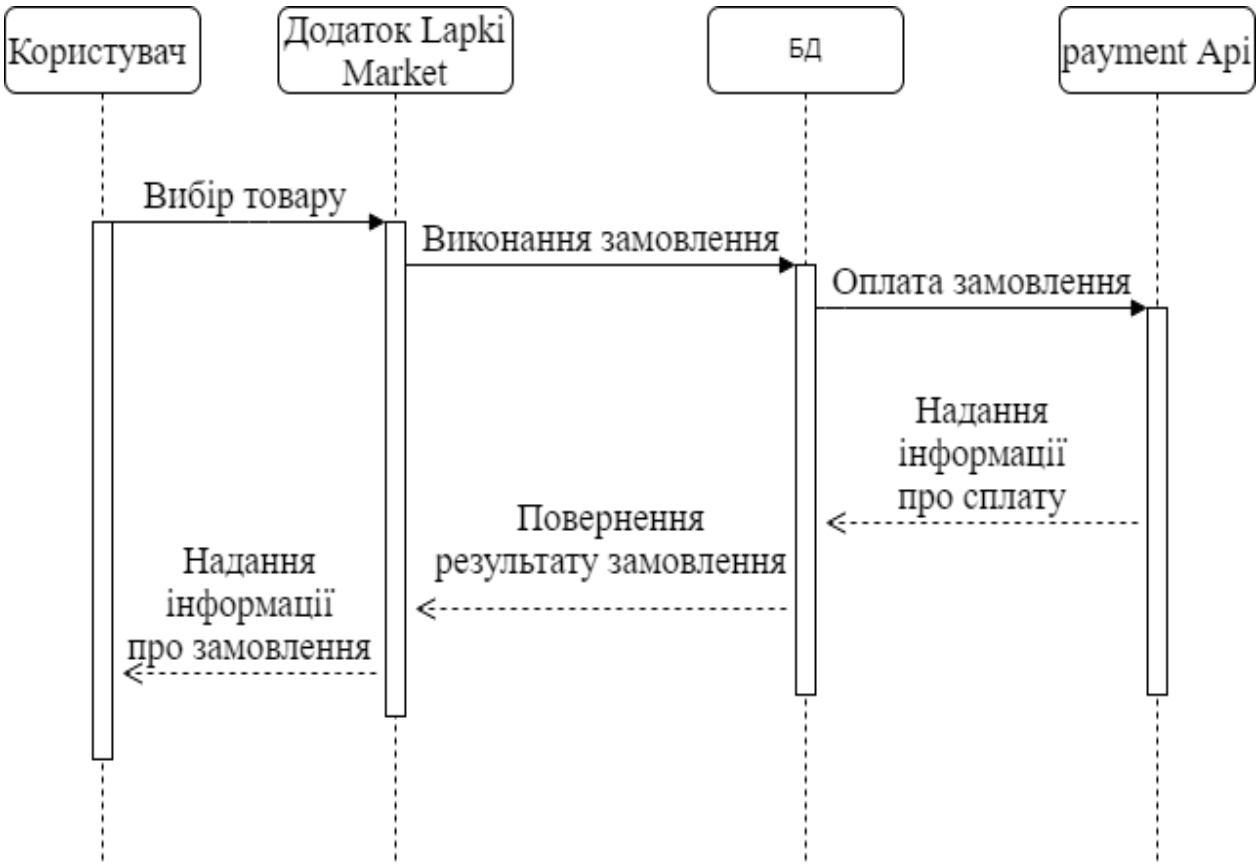
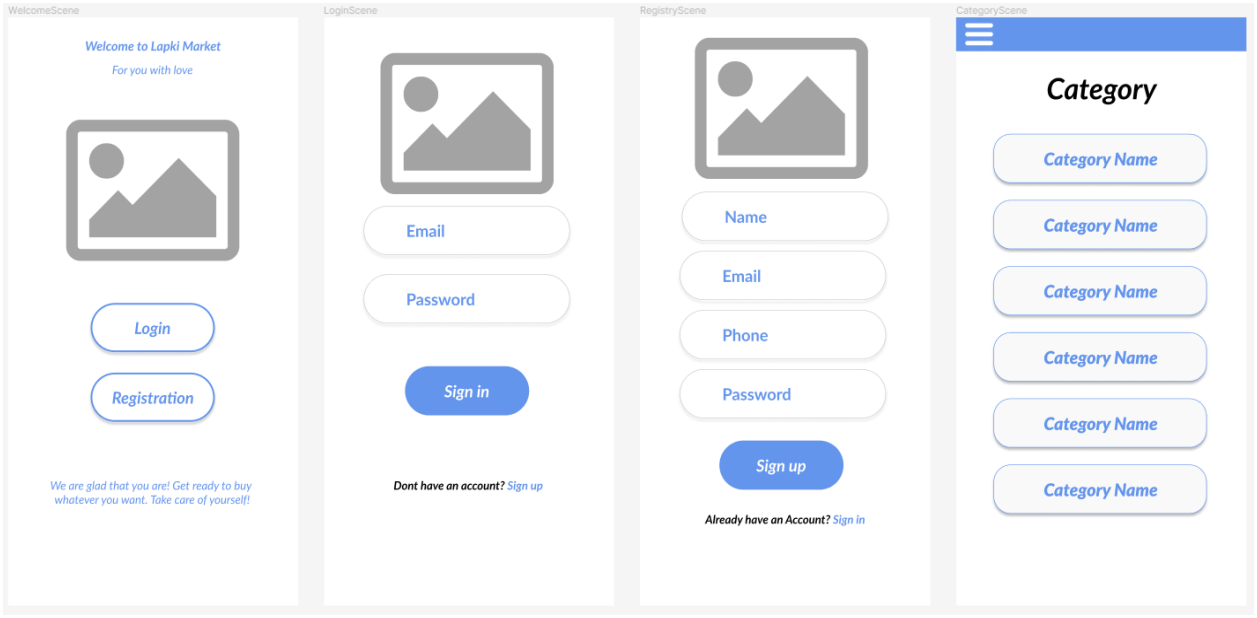


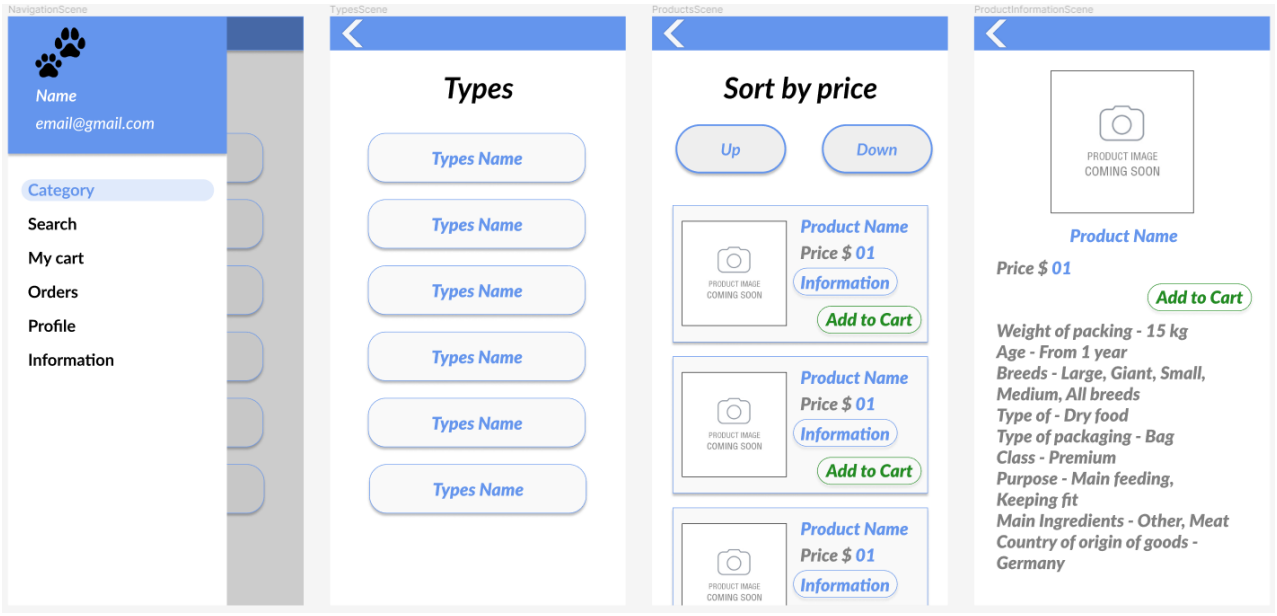
Рисунок 4.4 – Діаграма послідовностей

4.4 Проектування макетів інтерфейсу застосунку

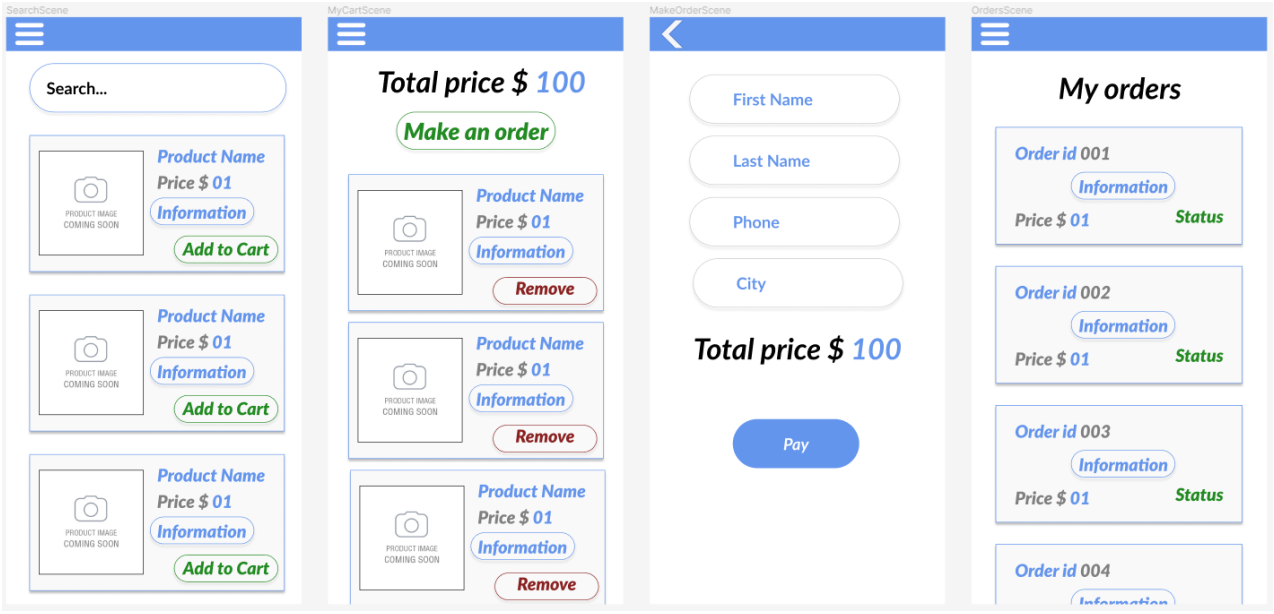
Після цього були створені макети інтерфейсу за допомогою Figma (онлайн-сервіс для розробки інтерфейсів та прототипування з можливістю організації спільної роботи в режимі реального часу) [15]. Було вирішено розробити 15 сторінок у застосунку та зробити дизайн легким та простим. Прототипи інтерфейсу наведені на рис. 4.5.



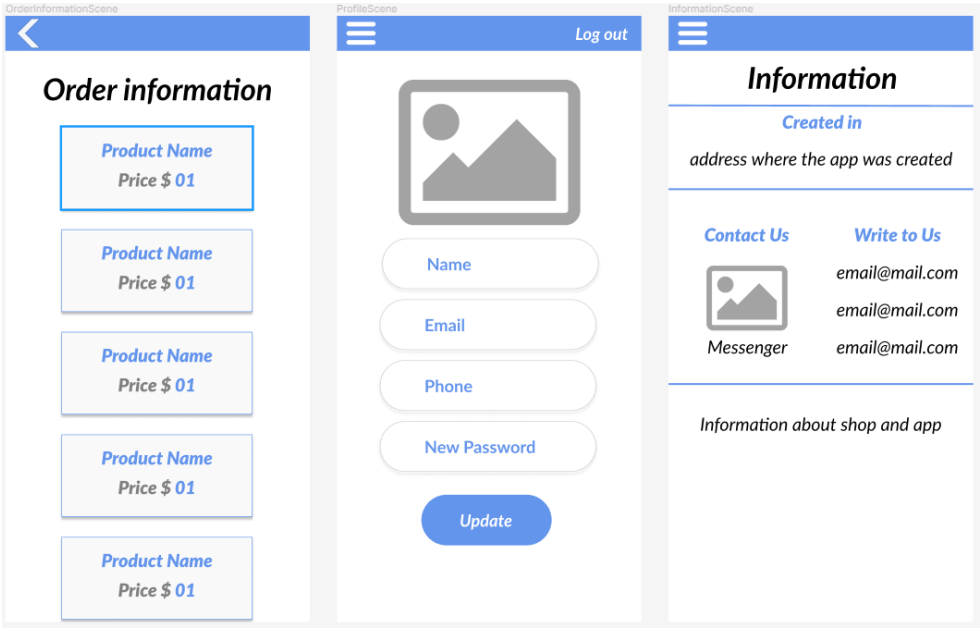
а) б) в) г)



г) д) е) є)



ж) з) и) і)



ї) й) к)

Рисунок 4.5 – Макети інтерфейсу застосунку – а) сторінка привітання б) сторінка авторизації – в) сторінка реєстрації– г) сторінка категорій – ґ) навігація у застосунку – д) сторінка типів – е) сторінка з товарами – є) сторінка з інформацією про товар – ж) сторінка з пошуком – з) сторінка з кошиком – и) сторінка з оформленням замовлення – і) сторінка з усіма заказами користувача – ї) сторінка з товарами заказу – й) сторінка профіля – к) сторінка з інформацією про Lapki Market

На сторінці привітання розміщено вітання користувача та запропонування відразу авторизуватися у застосунку або створити новий акаунт, після цього користувач має можливість увійти та отримати усі функції додатка.

У застосунку є навігаційна панель з якої користувач може перейти до таких розділів: категорії, пошук, кошик, замовлення, профіль та інформація.

На сторінці з категоріями користувач обираючи потрібний отримує у додатку типи товарів обраної категорії, обираючи потрібний тип відображаються товари, які користувач може сортувати за ціною, може отримати усю інформацію про товар натиснувши на потрібну кнопку, а також може додати товар до кошика кнопкою.

На сторінці з пошуком відображені усі товари магазину, а за допомогою пошукового рядка можна отримати потрібні товари та додати їх до кошика.

На сторінці кошика відображені усі обрані товари та сума, при натисканні на потрібну кнопку можна оформити замовлення де вказуються усі потрібні дані.

На сторінці із замовленнями відображаються усі замовлення які були зроблені з даного профіля, а також можна переглянути коротку інформацію про товари з замовлення.

На сторінці профіля можна переглянути особисту інформацію, а також її оновити, або вийти з акаунта.

На сторінці інформації можна отримати пошту та контакти та з адміністрацією магазина, коротку інформацію про магазин.

Макети інтерфейсу застосунку, UML – діаграми варіантів використання (прецедентів), активності і послідовності були створені на етапі проектування системи. Завдяки цьому можна зрозуміти логіку роботи застосунку та основні варіанти використання системи.

5 РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ «Lapki Market»

5.1 Використання існуючої бази даних інтернет зоомагазина

У роботі використовується існуюча БД інтернет-зоомагазину. Для створення запитів була отримана діаграма відносин сутностей, тобто візуальне уявлення бази даних, яке показує, як пов'язані елементи всередині.

Діаграма бази даних додатку Lapki Market має наступний вигляд:

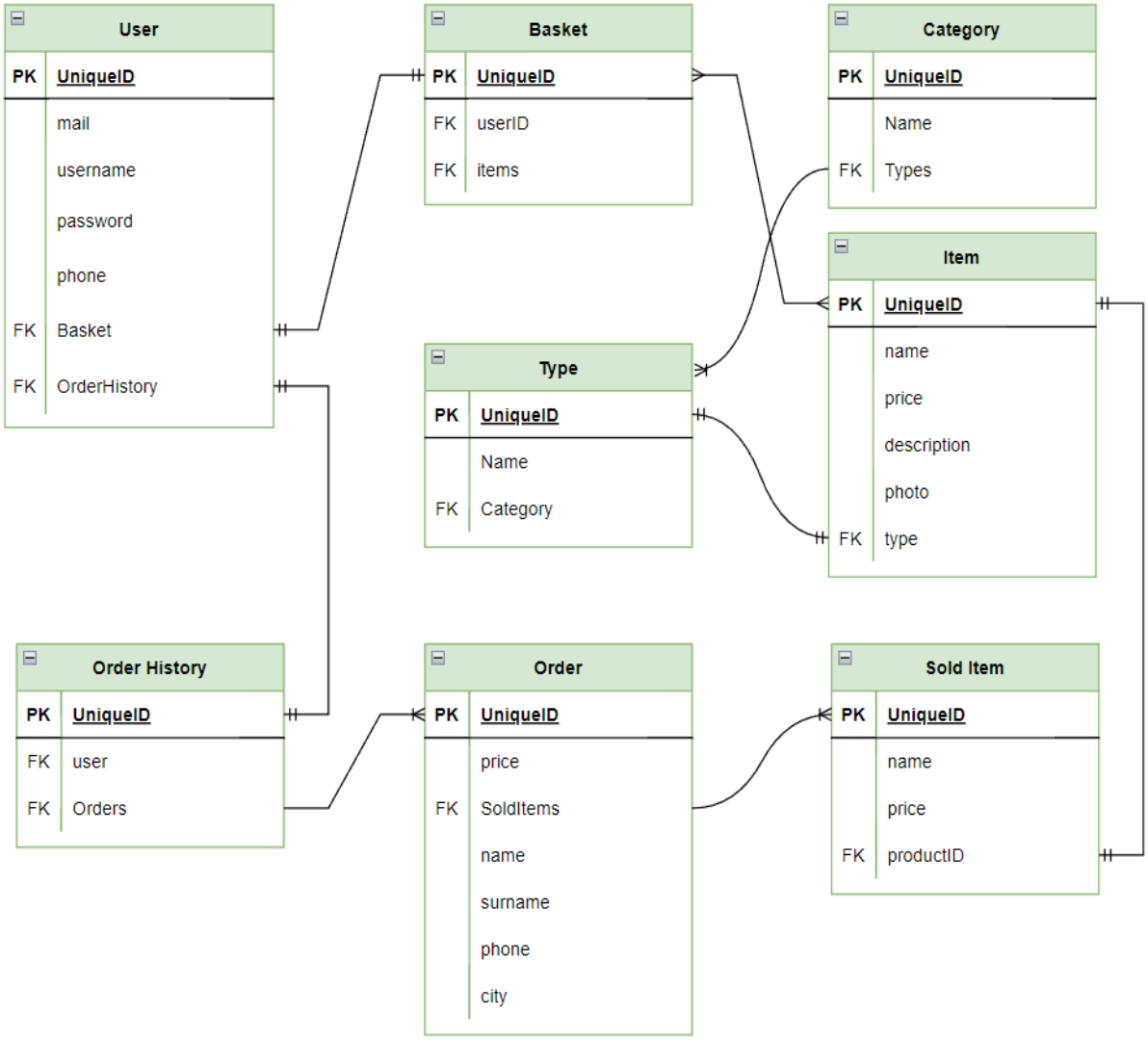


Рисунок 5.1 – Діаграма бази даних додатку Lapki Market

Для створення тестових запитів на сервер та тестування була використана програма Postman[16].

Програма Postman призначена для тестування роботи API, а також для надсилання запитів POST та GET. Вона має графічний інтерфейс, легка у розумінні та освоюванні.

Приклад GET запиту та відповідь сервера наведено на рис 5.2

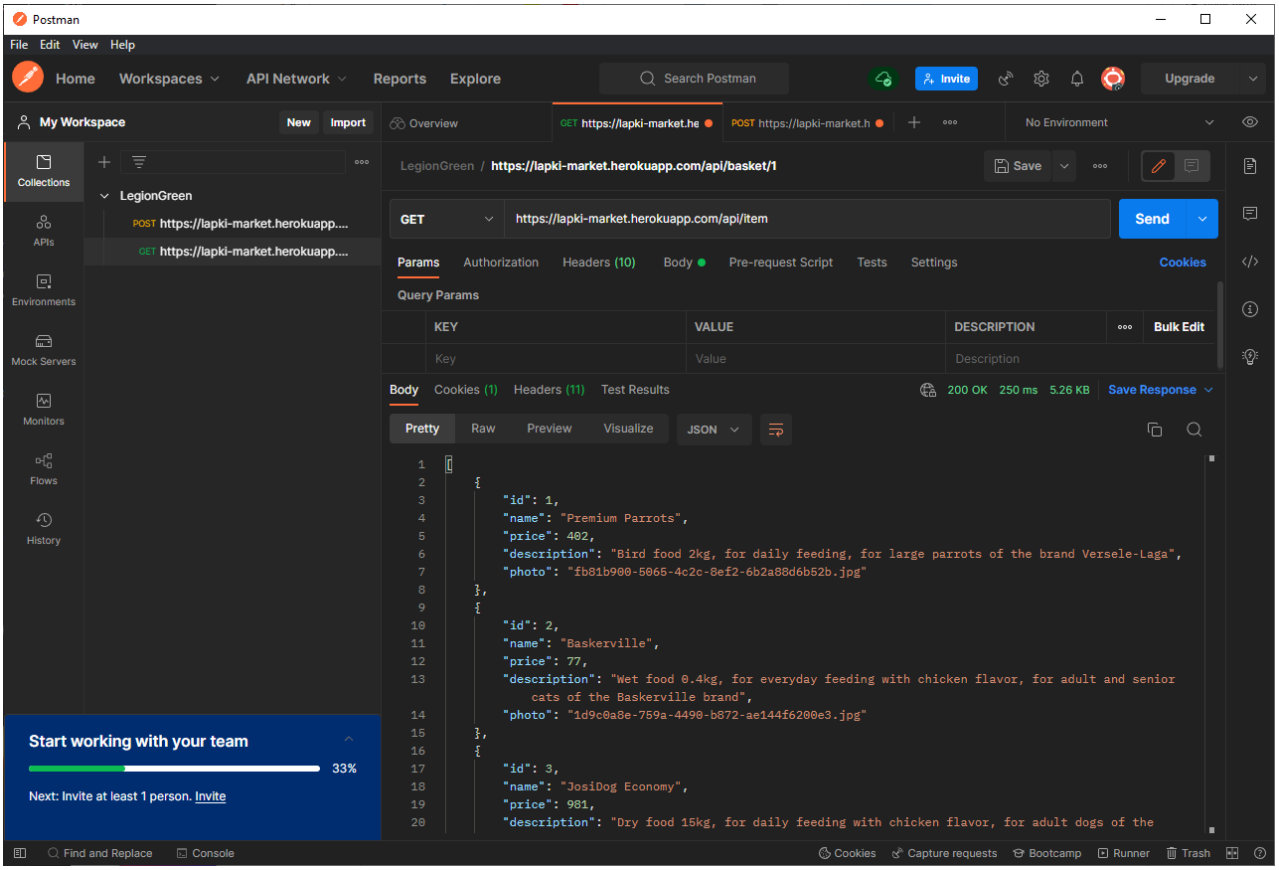


Рисунок 5.2 – Приклад GET запиту у Postman

Сервер відправляє потрібні дані у форматі JSON. Приклад опису об’єкта у форматі JSON:

{
"id": 1,
"name": "Premium Parrots",
"price": 402,
"description": "Bird food 2kg, for daily feeding, for large parrots of the brand Versele-Laga",
"photo": "fb81b900-5065-4c2c-8ef2-6b2a88d6b52b.jpg"
}

Приклад POST запиту з додаванням товару та відповідь сервера наведено на рис 5.3.

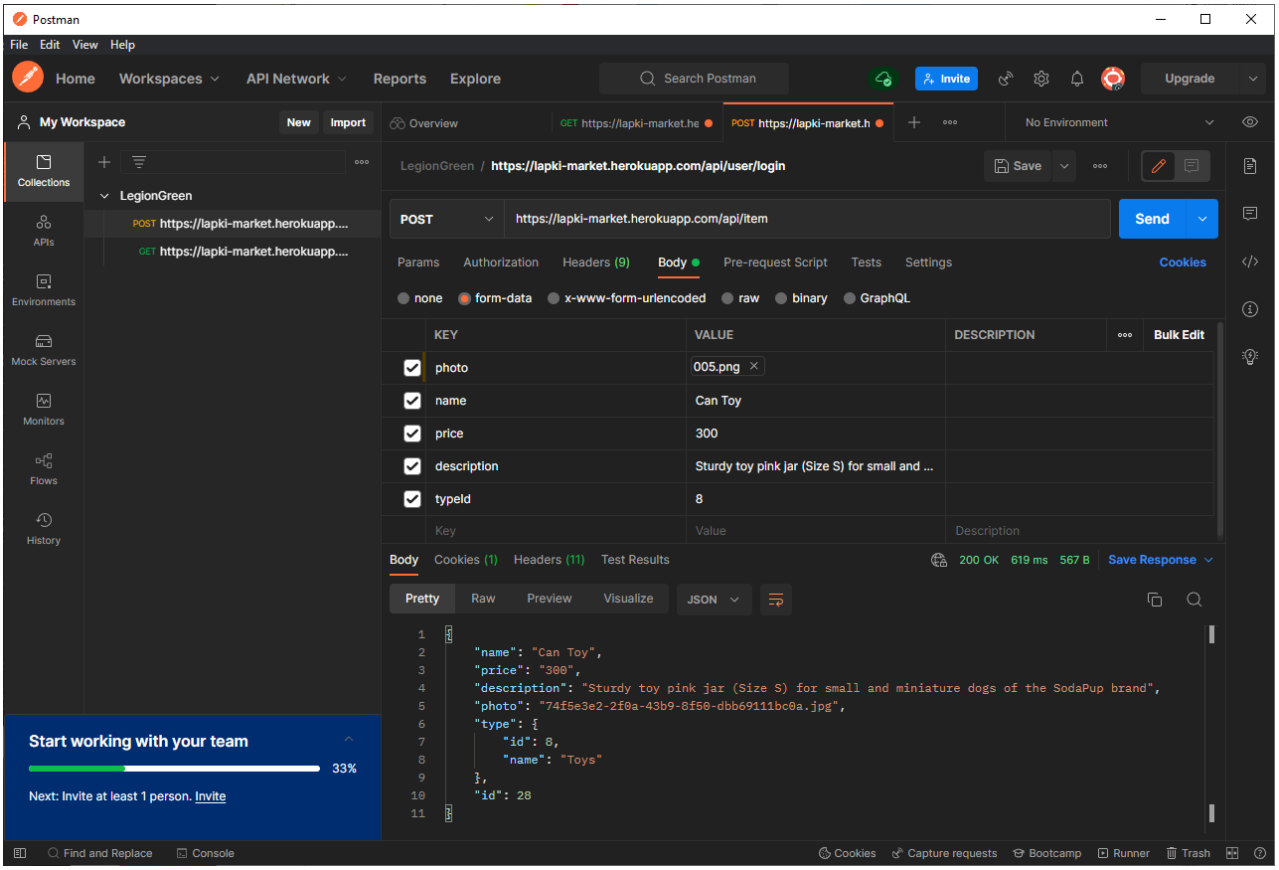


Рисунок 5.3 – Приклад POST запиту у Postman

5.2 Розробка додатка у Android Studio

Середовищем розробки мобільного додатка обрано Android Studio. Воно складається з наступних компонентів: Android SDK, компоненти графічного дизайну, додаток для завантаження компонентів всіх версій Android, емулятор мобільного пристрою для запуску програми, інструменти для тестів та налагодження роботи програми. Важливим моментом при виборі Android Studio стало те, що вона має можливість розробляти програми практично для всіх версій ОС Android. Існує інструмент для оцінки зовнішнього вигляду програми для різних пристроїв. Багатобарвний код спрощує навігацію у високих обсягах коду.

У кожному Android-проекті необхідно позначити мінімальну підтримувану програмою версію платформи Android. Чим менше версія, тим на більшу кількість пристроїв буде можливе встановлення програми, але буде неможлива або обмежена робота з деякими можливостями API пізніших версій платформи. Згідно з офіційним сайтом платформи Android [17], частка Android-пристроїв версії 5.0 і вище становить 98% на момент початку березня 2022 року. Тому в якості мінімальної версії, що підтримується, було вирішено вибрати версію 5.0 (Lollipop, API 21) (рис. 5.4).

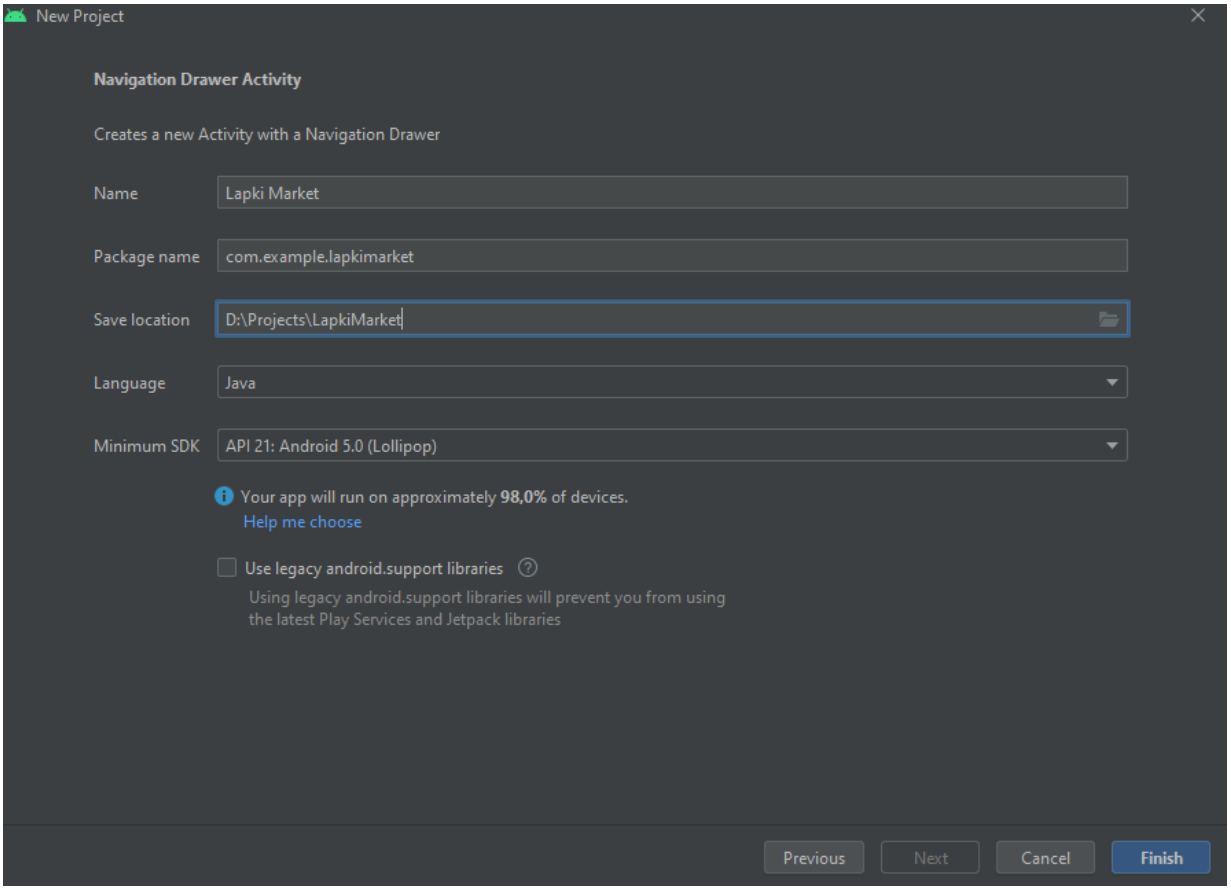


Рисунок 5.4 – Створення нового проекту на Android Studio

Архітектура програми з ОС Android заснована на двох типах взаємопов'язаних файлів це активності у форматі .java та макети екрану з xml розміткою. Файли макетів екранів або layout-файли містять у собі всі елементи видимі користувачем. Дані елементи описані у вигляді xml-коду, завдяки цьому

дані макети спроектовані в ході розробки при правильного налаштування всіх параметрів однаково відображаються на різних типи пристроїв. Layout-файли можна назвати деяким аналогом форм для Windows додатків, тільки на відміну від них, вони не мають у собі програмного коду Кожному макету екранів відповідає свій окремий файл із кодом званий Activity. Весь програмний код програми зберігається у файлах Activity або Java класи. Кожен клас відповідає тільки за той набір кнопок та елементів, який розташовується на пов'язаному з ним файлі файлу. Виняток складає лише додаткові класи в яких описується додаткові процеси не видимі користувачеві.

Розглянемо структуру проекту програми під ОС Android, з доповненнями після створення проекту за замовчуванням (рис. 5.5).

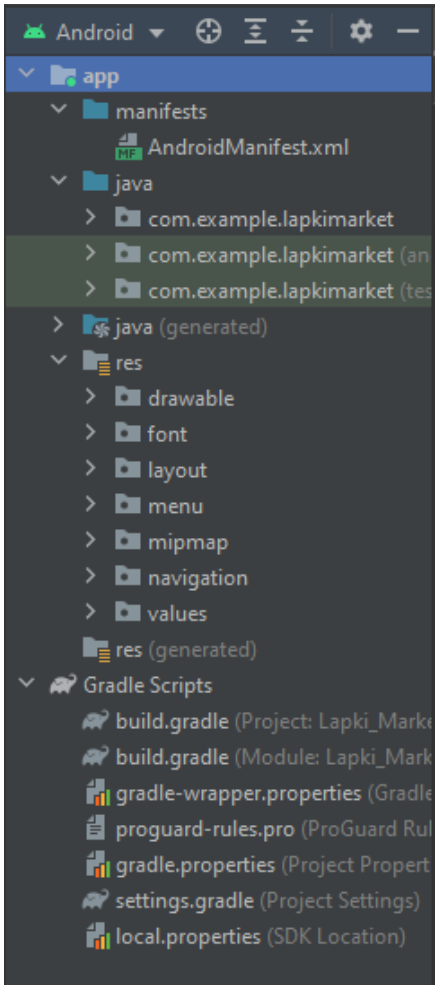


Рисунок 5.5 – Структура проекту на Android Studio

Проект може містити різні модулі. І всі модулі описуються файлом setting.gradle. Якщо подивитися на структуру проекту, весь важливий код - файли інтерфейсу, класи java і т.д. за замовчуванням знаходяться у папці (модулі) app. Файл build.gradle містить інформацію, яка використовується при побудові проекту.

Кожен модуль має свій файл build.gradle, який визначає конфгурацію побудови проекту, специфічну для цього модуля. Якщо подивитися на вміст папки app, то саме можна знайти в ній такий файл.

За замовчуванням кожен проект включає один модуль – app.

Короткий опис важливих папок та файлів у проекті:

- AndroidManifest.xml - Це файл manifest, що описує фундаментальні характеристики програми і визначає кожен його компонент.
- java - Ця папка містить вихідні файли java для вашого проекту. За замовчуванням вона включає вихідний файл MainActivity.java клас activity, запущений при старті програми.
- res/drawable - Попередні версії Android використовують цю папку для зберігання зображень, нові версії використовують замість неї папку mipmap для зберігання зображень. Ця папка майже не використовується. У проекті папка використовувалася для зберігання XML - файлів різних графічних зображень.
- res/layout - Ця папка містить файли, що визначають інтерфейс користувача.
- res/menu - Ця папка містить xml, що визначає menu який відображається на Action Bar.
- res/mipmap - Містить зображення 'mipmap'.
- res/values - Це папка для різних файлів XML, що містять колекцію ресурсів, наприклад рядків (String) і визначення кольорів.
- res/font - Це папка, де зберігаються додані нові шрифти, в нашому випадку був доданий шрифт lato і всі його види.

- res/navigation - У папці розміщено навігаційний файл mobile_navigation, у файлі навігації є шість фрагментів, які є пунктами призначення, або екранами, на які ми потрапляємо, переходячи по пунктах бокової панелі.

Для реалізації програми «Lapki Market» потрібне використання бази даних, що містять інформацію про товари в зоомагазині та інформацію про користувачів. Для цього була використана бібліотека Retrofit.

Бібліотекою зручно користуватися для запиту до різних веб-сервісів із командами GET, POST, PUT, DELETE. Може працювати в асинхронному режимі, що позбавляє зайвого коду.

Підключається бібліотека стандартним чином:

implementation 'com.squareup.retrofit2:retrofit:2.4.0'

Бібліотека може працювати з GSON та XML, використовуючи спеціальні конвертери, які слід вказати окремо:

implementation 'com.squareup.retrofit2:converter-gson:2.4.0'

Потім у коді конвертер додається за допомогою методу addConverterFactory():

addConverterFactory(GsonConverterFactory.create())

Робота з Retrofit була розбита на 3 окремі завдання:

Завдання перше – подивитися на структуру відповіді сайту у вигляді JSON (або інших форматів) та створити на його основі Java-клас.

Завдання друге - створити інтерфейс та вказати ім'я методу. Додати необхідні параметри, якщо вони потрібні.

В інтерфейсі задаються команди-запити сервера. Команда комбінується з базовою адресою сайту (baseUrl()) та виходить повний шлях до сторінки. У більшості випадків ми повертатимемо об'єкт Call<T> з потрібним типом, наприклад, Call<UserModel>

Приклади методів зображені на рис. 5.6.

```java
public interface ApiRetrofit {

    @POST("api/user/reg")
    Call<UserModel> regUser(@Body RegRequest regRequest);

    @POST("api/user/login")
    Call<UserModel> loginUser(@Body LoginRequest loginRequest);

    @GET("api/category/")
    Call<List<CategoryModel>> getCategory(@Header("Authorization")String token);

    @GET("api/type/{id}")
    Call<List<SubcategoryModel>> getSubCategory(@Header("Authorization")String token, @Path("id") String id);

    @GET("api/item/")
    Call<List<ProductModel>> getProducts(@Header("Authorization")String token, @Query("typeId") String typeId);

    @GET("api/item")
    Call<List<AllProductModel>> getAllProducts();

    @POST("api/basket")
    Call<CartModel> addToCart(@Header("Authorization")String token, @Body CartModel cartModel);

    @GET("api/basket/{id}")
    Call<CartModel> getUserCart(@Header("Authorization")String token, @Path("id") String id);

    @HTTP(method = "DELETE", path = "api/basket", hasBody = true)
    Call<CartModel> deleteFromCart(@Header("Authorization")String token, @Body CartModel cartModel);

    @GET("api/user/logout")
    Call<UserModel> logOut();

    @PUT("api/user/")
    Call<UserModel> updateUser(@Header("Authorization")String token,@Body RegRequest regRequest);

    @POST("api/order/")
    Call<OrderModel> payOrder(@Header("Authorization")String token,@Body OrderModel orderModel);

    @GET("api/order-history/")
    Call<OrderHistoryModel> getOrderHistory(@Header("Authorization")String token);

```

Рисунок 5.6 – Створені команди-запити

Третім завданням є створення об'єкта для запиту до сервера і самі запити. Для асинхронного запиту використовуємо метод Call.enqueue(). Об'єкт для запиту на сервер зображений на рис. 5.7.

У результаті ми отримуємо об'єкт Retrofit, що містить базовий URL та здатність перетворювати JSON-дані за допомогою вказаного конвертера Gson.

```java
package com.example.lapkimarket.retrofit;

import ...

public class RetrofitController {

    private static final String url = "https://lapki-market.herokuapp.com/";

    public static Retrofit getClient()
    {

        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(url)
                .addConverterFactory(GsonConverterFactory.create())
                .build();

        return retrofit;
    }


    public static ApiRetrofit getApi(){

        return getClient().create(ApiRetrofit.class);
    }

}
```

Рисунок 5.7 – Об'єкт для запиту на сервер

Приклад запиту на отримання кошика користувача зображено на рис. 5.8.

```java
Call<CartModel> call = RetrofitController.getApi().getUserCart( token: "Bearer "+tokenName, userid);
call.enqueue(new Callback<CartModel>() {
```

Рисунок 5.8 – Запит на отримання кошика користувача

5.3 Опис та тестування застосунку

Значок застосунку з версією зображений на рис. 5.9. Робота з застосунком починається зі сторінки вітання з якої переходимо до реєстрації або авторизації (рис. 5.10).

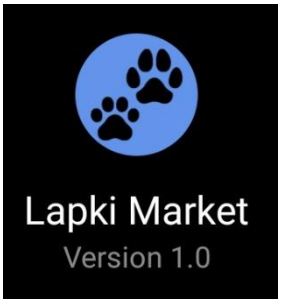


Рисунок 5.9 – Значок застосунку «Lapki Market»

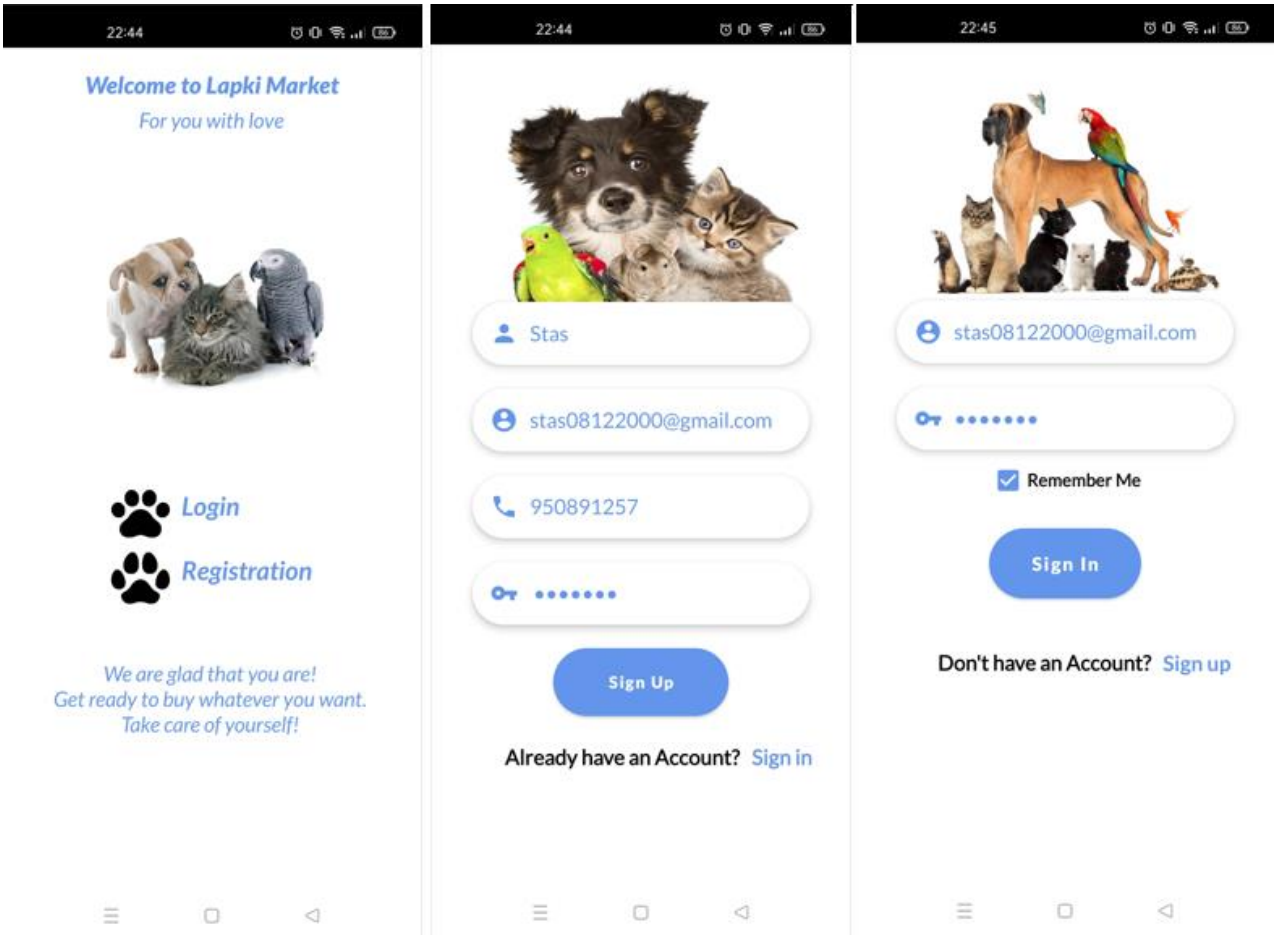


Рисунок 5.10 – Сторінка вітання, реєстрації та авторизації «Lapki Market»

При успішній авторизації користувач потрапляє до магазину. Користувач може вибрати будь-який пункт в навігаційній панелі, яка показана на рис.5.11.

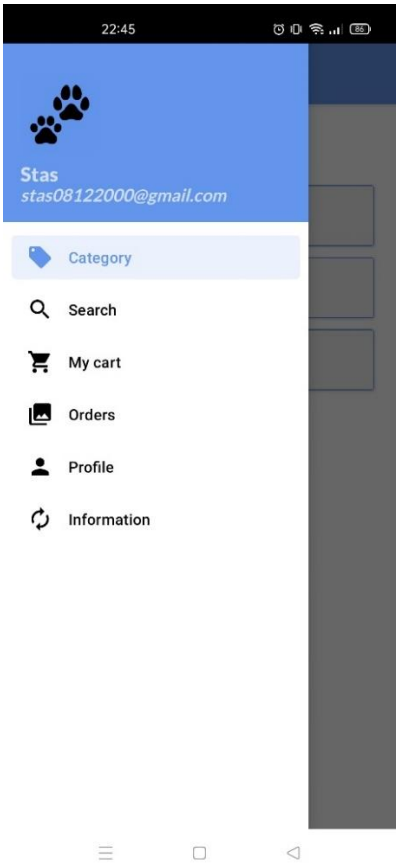


Рисунок 5.11 – Навігація у додатку

З панелі навігації користувач може перейти до розділу категорії. Користувачеві надається вибір різних категорій, при виборі однієї з них відбувається перехід на сторінку з типами обраної категорії (рис. 5.12).

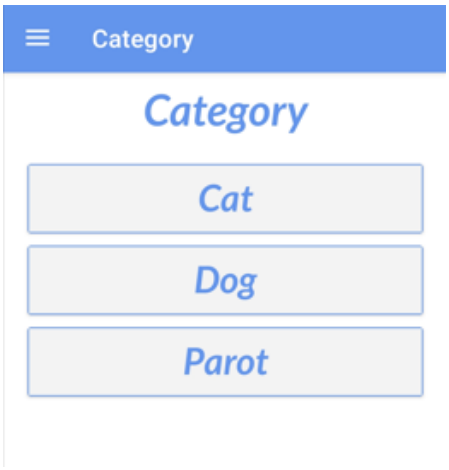


Рисунок 5.12 – Категорії товарів

Після того, як покупець вибирає потрібний йому тип, відбувається перехід на сторінку з товарами даного типу. На сторінці є можливість сортування товарів за ціною. У кожного товару на сторінці є назва, зображення товару, вказана ціна, а також присутні дві кнопки для отримання повної інформації про товар і додавання товару в кошик користувача (рис. 5.13).

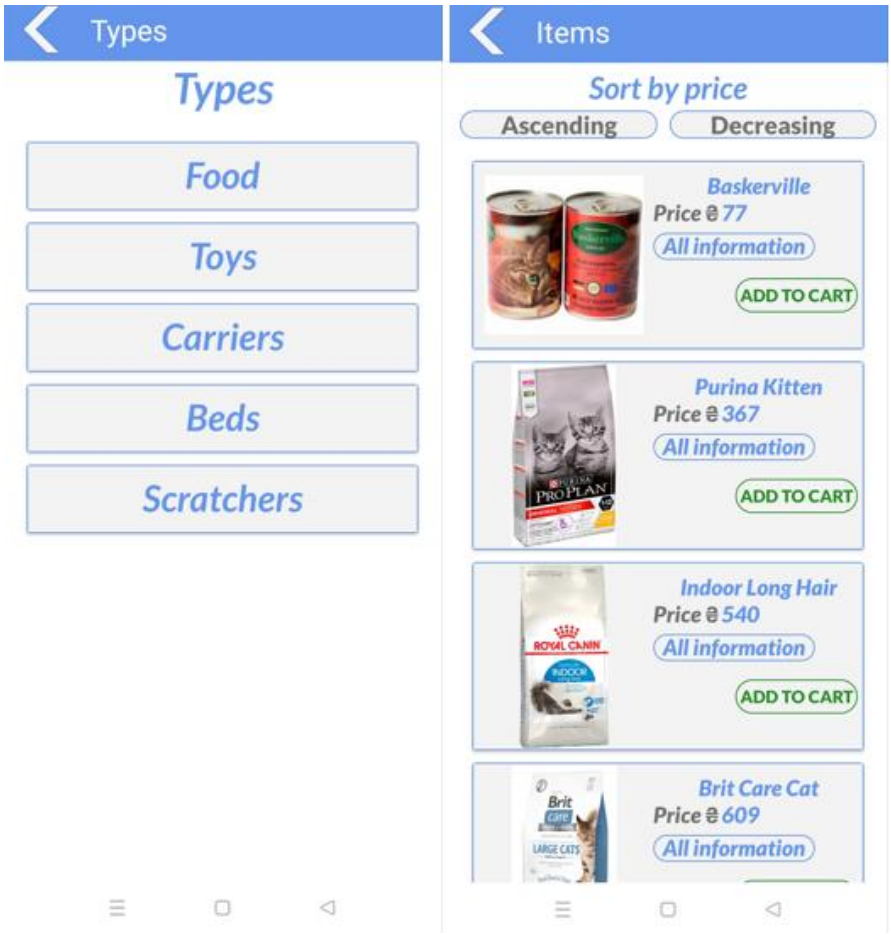


Рисунок 5.13 – Типи товарів та список товарів

Сторінка товару з повним описом зображена на рис 5.14.

Вибираючи в навігаційній панелі розділ Search, користувач потрапляє на сторінку з пошуковим рядком та всіма товарами в магазині. На цій сторінці відбувається пошук товару за назвою незалежно від його регістру (рис. 5.15).

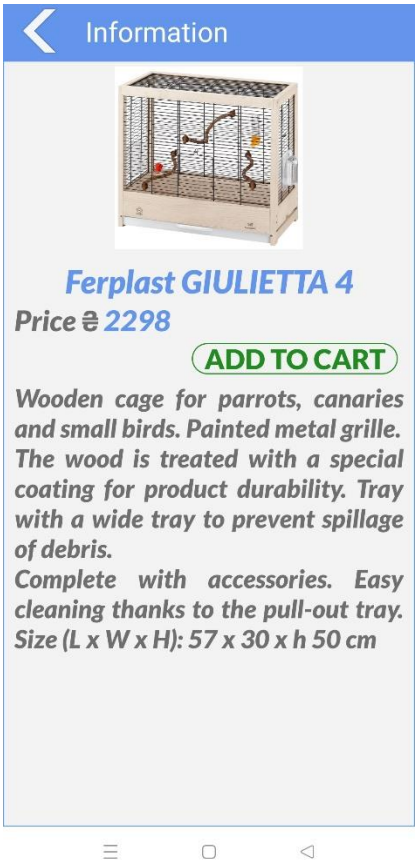


Рисунок 5.14 – Сторінка товару

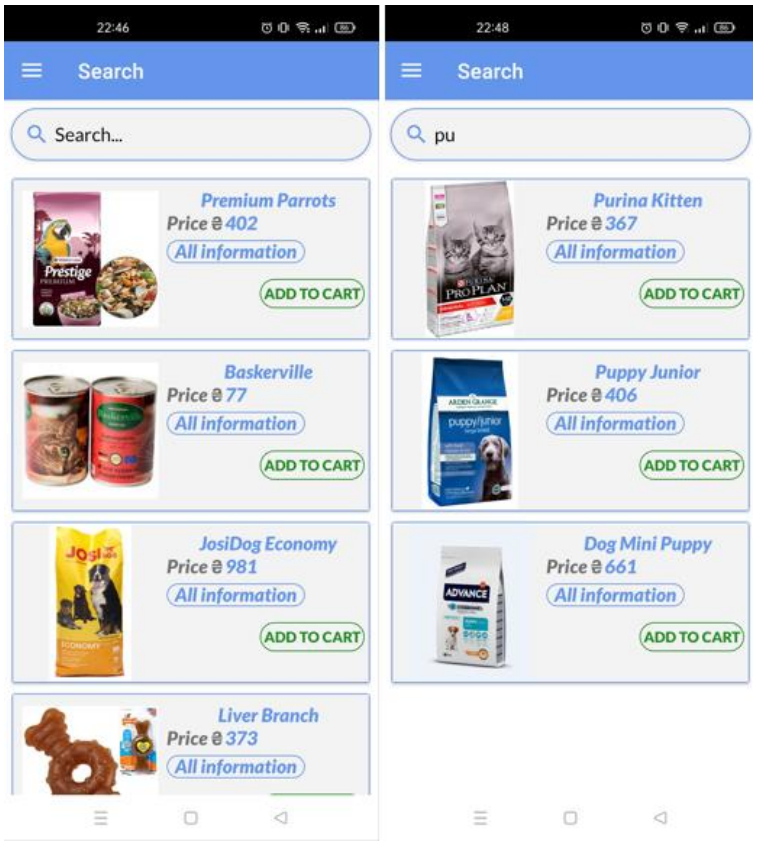


Рисунок 5.15 – Сторінка пошуку товару

Порожній та заповнений товарами кошик зображений на рис 5.16.

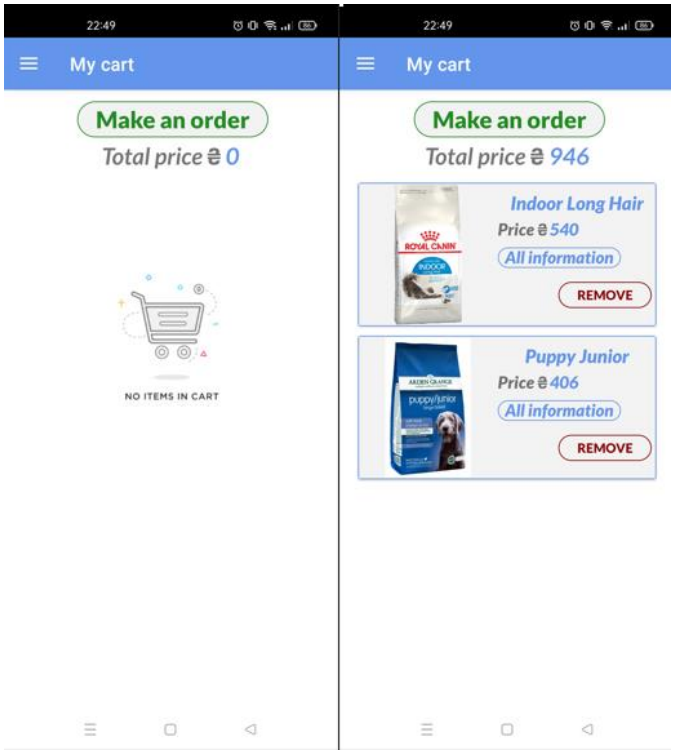


Рисунок 5.16 – Кошик користувача

На сторінці оформлення замовлення потрібно ввести Ім'я та Прізвище, так само автоматично вводиться пошта користувача, яку можна змінити за необхідності, а також потрібно ввести Місто одержувача. На сторінці є дві кнопки оформлення замовлення (рис. 5.17). Кнопка Pay now має передбачає під собою подальшу реалізацію, розробку модуля, що дозволяє проводити онлайн-оплату товарів.

У розділі із замовленнями відображаються всі замовлення авторизованого користувача. На сторінці відображаються всі замовлення з усією інформацією, айді замовлення ім'я та прізвище, номер телефону та місто одержувача, а також сума замовлення разом зі статусом. За допомогою кнопки можна переглянути всі замовлені товари у вигляді списку з ім'ям та ціною товару (рис. 5.18).

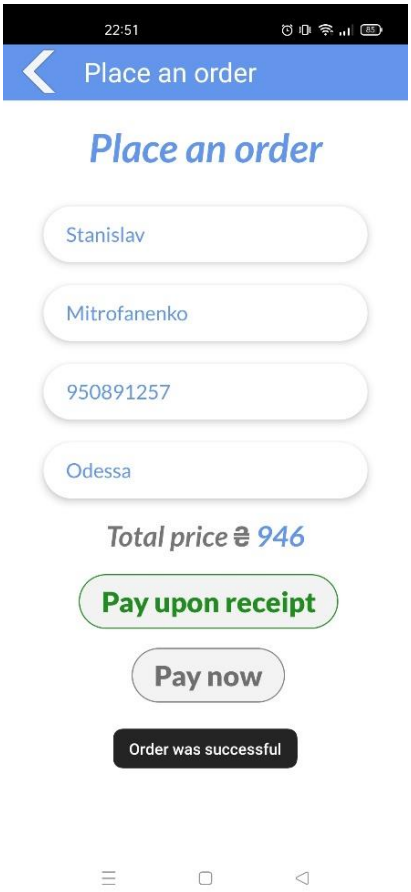


Рисунок 5.17 – Оформлення замовлення

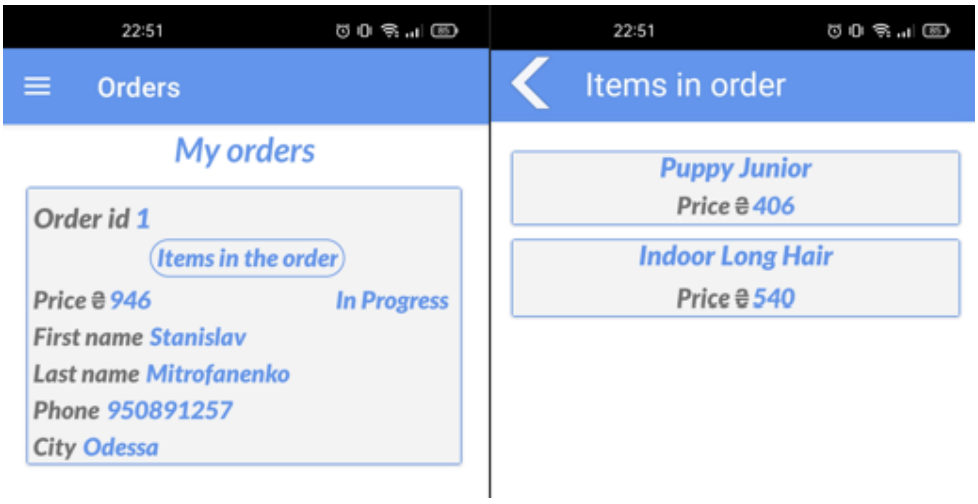


Рисунок 5.18 – Замовлення користувача та подробиці замовлення

На сторінці профілю відображається вся інформація про користувача, також можна оновити будь-які дані, включаючи пароль. На цій сторінці можна вийти з облікового запису за допомогою кнопки (рис. 5.19).

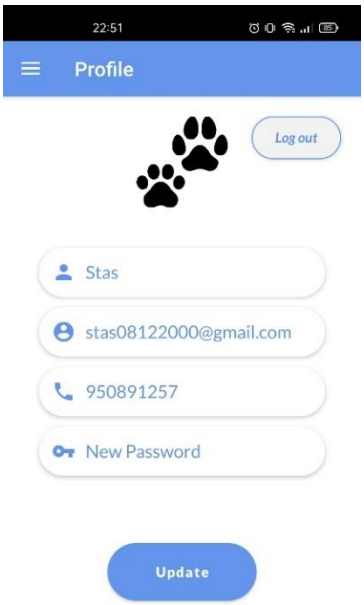


Рисунок 5.19 – Сторінка профілю

На сторінці з інформацією можна прочитати коротку інформацію про магазин та програму. Також на сторінці можна переглянути контакти магазину, при натисканні на електронну пошту можна перейти в іншу програму, звідки можна написати листа. В інтернет зоомагазину «Lapki Market» також є група в Telegram, в яку можна потрапити одним натисканням по іконці (рис. 5.20).

Рисунок 5.20 – Сторінка з інформацією

Також мобільний додаток інтернет зоомагазину був протестований трьома іншими людьми на їхніх мобільних пристроях (рис. 5.21). Проблем при використанні мобільного додатка виявлено не було.

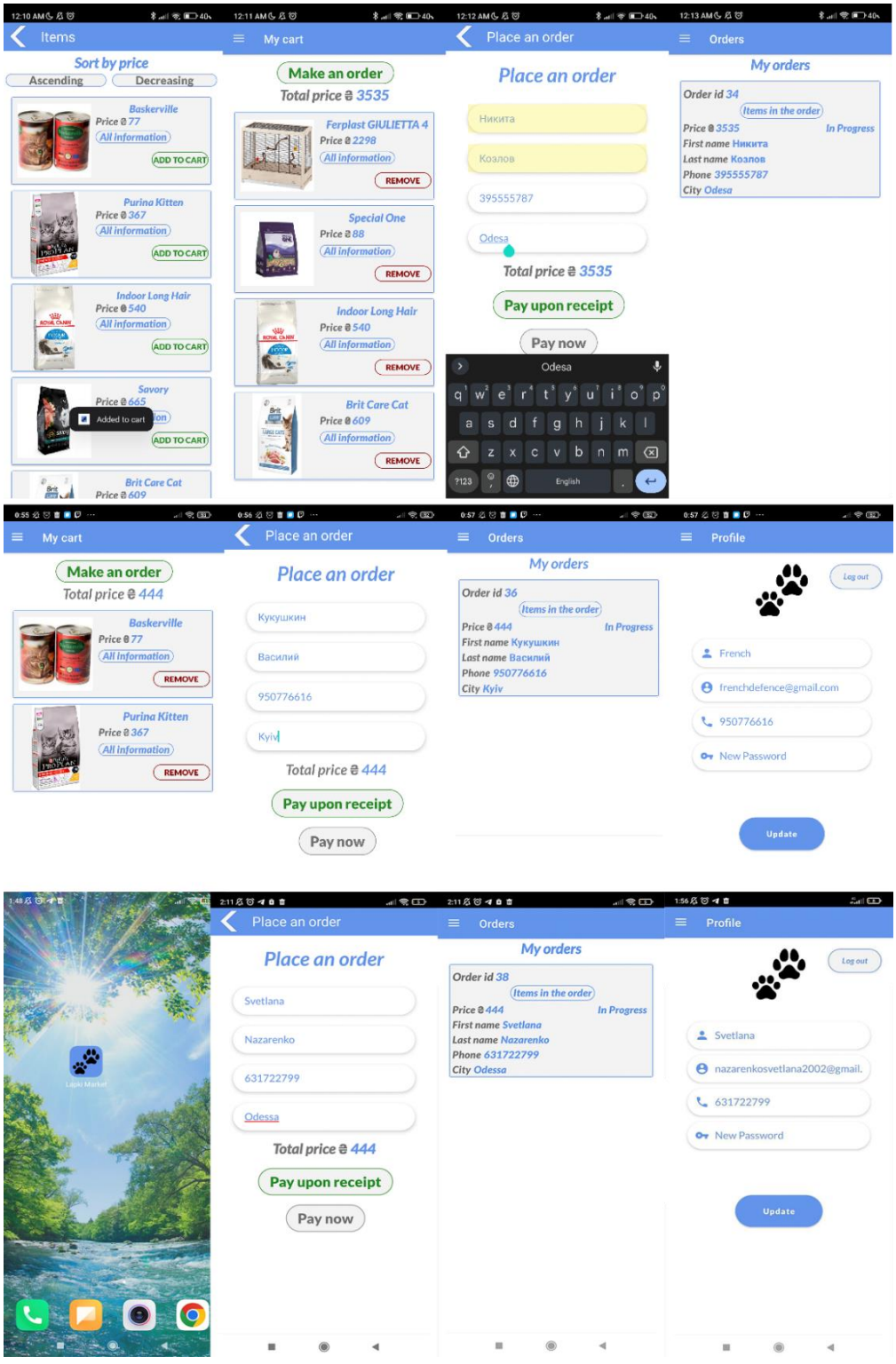


Рисунок 5.21 – Тести застосунку інших 3 людей

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра був створений мобільний застосунок інтернет-зоомагазину «Lapki Market» для просування та продажу товарів, який додатково зменшує витрати на рекламу та сприяє розширенню ринку збуту товару для тварин, поповненню клієнтської аудиторії і, як наслідок, збільшенню прибутку, що є головною метою бізнесу. Дана розробка може бути перспективною та корисною з комерційної та споживчої точки зору. Для цього був проведений аналіз предметної області та порівняльний аналіз існуючих програм-аналогів у Play Market (ZooZooZoo, VMISKE, Pet Shop).

Середовищем розробки мобільного додатка обрано Android Studio. Воно складається з наступних компонентів: Android SDK, компоненти графічного дизайну, додаток для завантаження компонентів всіх версій Android, емулятор мобільного пристрою для запуску програми, інструменти для тестів та налагодження роботи програми. Важливим моментом при виборі Android Studio стало те, що вона має можливість розробляти програми практично для всіх версій ОС Android.

Бібліотеки в Android дуже корисні, оскільки виконують у собі різні трудомісткі завдання, звільняючи розробника від реалізації у своєму додатку. У роботі була використана бібліотека Retrofit для мережевої взаємодії, вона відмінно підтримує REST API, легко тестується і налаштовується, а запити по мережі з її допомогою виконуються дуже просто. А також була використана бібліотека Glide, вона є однією з найпопулярніших бібліотек з відкритим вихідним кодом для роботи із зображеннями, оскільки реалізує асинхронне завантаження зображень, їх обробку та кешування, а також вміє працювати з GIF-зображеннями та відео.

На етапі проектування були розроблені макети інтерфейсу застосунку та створені UML-діаграми варіантів використання (прецедентів), послідовності і активності.

В роботі представлені можливості застосунку та поетапний приклад його використання. Було виконано тестування роботи застосунку на різних мобільних пристроях різними людьми. Проблем при використанні виявлено не було.

Результати роботи доповідалися на Студентській науковій конференції ОДЕКУ 11 – 18 травня 2022.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Інтернет-магазин. (Інтернет-магазин) URL: https://uk.wikipedia.org/wiki/Інтернет-магазин (дата звернення 1.05.2022)
2. Домашні тварини. (Домашние животные) URL: https://ru.wikipedia.org/w/index.php?title=Домашние_животные&stable=1 (дата звернення 1.05.2022)
3. Найпопулярніші мобільні операційні системи з 2000 по 2021 рік. URL: https://www.youtube.com/watch?v=rMTDVK0Zb2k&ab_channel=WonderfulData (дата звернення 1.05.2022)
4. Mobile Operating System Market Share Ukraine URL: Mar 2021 - Mar 2022 URL: https://gs.statcounter.com/os-market-share/mobile/ukraine/#monthly-202103-202203-bar (дата звернення 1.05.2022)
5. Объективное сравнение iOS и Android в 2021 году URL: https://droidblog.org/help/android-ili-ios/ (дата звернення 3.05.2022)
6. 36,1 млрд додатків завантажили на смартфони у 2021 URL: https://techtoday.in.ua/news/36-1-mlrd-dodatkiv-zavantazhyly-na-smartfony-u-mynulomu-kvartali-149471.html (дата звернення 3.05.2022)
7. Объективное сравнение iOS и Android в 2022 году URL: https://uk.natapa.org/Android-vs-iOS-766 (дата звернення 3.05.2022)
8. Android Studio - інтегроване середовище розробки (IDE) для платформи Android URL: https://ru.wikipedia.org/wiki/Android_Studio (дата звернення 4.05.2022)
9. Android Studio: переваги та особливості URL: https://qagroup.com.ua/publications/android-studio-perevagy-ta-osoblyvosti/(дата звернення 4.05.2022)
10. Офіційний веб-сайт IntelliJ IDEA URL: https://www.jetbrains.com/ru-ru/idea/(дата звернення 7.05.2022)

11. Мова програмування Java URL: https://ru.wikipedia.org/wiki/Java(дата звернення 7.05.2022)
12. Офіційний веб-сайт Gradle URL: https://developer.android.com/studio/build (дата звернення 15.05.2022)
13. Офіційний веб-сайт github Retrofit 2 android URL: https://square.github.io/retrofit/ (дата звернення 15.05.2022)
14. Офіційний веб-сайт Glide URL: https://bumptech.github.io/glide/ (дата звернення 15.05.2022)
15. Офіційний веб-сайт Figma URL: https://www.figma.com/ (дата звернення 17.05.2022)
16. Офіційний веб-сайт Postman URL: https://www.postman.com/ (дата звернення 18.05.2022)
17. Офіційний веб-сайт Android Developers Dashboard URL: https://developer.android.com/about/dashboards/index.html (дата звернення 18.05.2022)

ДОДАТОК А Вихідний код програми

```
package com.example.lapkimarket.activities;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.example.lapkimarket.MainActivity;
import com.example.lapkimarket.R;

import com.example.lapkimarket.models.LoginRequest;
import com.example.lapkimarket.models.UserModel;
import com.example.lapkimarket.retrofit.RetrofitController;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class LoginActivity extends AppCompatActivity {

    Button signIn;
    TextView signUp;
    EditText email, password;
    ProgressBar progressBar;
```

```
    SharedPreferences sharedPreferences;
    SharedPreferences.Editor editor;
    Boolean saveLogin;
    CheckBox saveLoginCheckbox;

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

        progressBar = findViewById(R.id.progressbar);
        progressBar.setVisibility(View.GONE);

        signIn = findViewById(R.id.log_btn);
        email = findViewById(R.id.email_login);
        password = findViewById(R.id.password_login);
        signUp = findViewById(R.id.sign_up);

        saveLoginCheckbox = findViewById(R.id.saveLoginCheckBox);
        sharedPreferences = getSharedPreferences("loginref",MODE_PRIVATE);
        editor = sharedPreferences.edit();

        signUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new                Intent(LoginActivity.this,
RegistryActivity.class));
            }
        });
        signIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                loginUser();
                progressBar.setVisibility(View.VISIBLE);
```

```
                Toast.makeText(LoginActivity.this,      "Please      wait",
Toast.LENGTH_SHORT).show();

            }
        });

        saveLogin = sharedPreferences.getBoolean("savelogin",true);
        if (saveLogin=true){
            email.setText(sharedPreferences.getString("username",null));

password.setText(sharedPreferences.getString("password",null));

saveLoginCheckbox.setChecked(sharedPreferences.getBoolean("checked",true))
;

        }
        else {
            email.setText("Email");
            password.setText("Password");

saveLoginCheckbox.setChecked(sharedPreferences.getBoolean("checked",false)
);
        }
    }
    private void loginUser() {

        String userEmail = email.getText().toString();
        String userPassword = password.getText().toString();

        if (TextUtils.isEmpty(userEmail)) {
            Toast.makeText(this,         "Email         is         Empty!",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(userPassword)) {
            Toast.makeText(this,        "Password        is        Empty!",
Toast.LENGTH_SHORT).show();
            return;
        }
```

```
        if (saveLoginCheckbox.isChecked()){
                editor.putBoolean("savelogin",true);
                editor.putString("username",userEmail);
                editor.putString("password",userPassword);
                editor.putBoolean("checked",true);
                editor.commit();
        }else{
            editor.clear();
            editor.putBoolean("checked",false);
            editor.putBoolean("savelogin",false);
            saveLoginCheckbox.setChecked(false);
            editor.commit();
           //  Toast.makeText(LoginActivity.this,  "Error  to  save  data",
Toast.LENGTH_SHORT).show();
        }
        //Login User

        Call<UserModel>  call  =  RetrofitController.getApi().loginUser(new
LoginRequest(userEmail,userPassword));
        call.enqueue(new Callback<UserModel>() {
            @Override
            public void onResponse(@NonNull Call<UserModel> call, @NonNull
Response<UserModel> response) {



                UserModel loginModel = response.body();
                if (response.isSuccessful()){

                    Toast.makeText(LoginActivity.this,              "Login
Successfully, please wait", Toast.LENGTH_SHORT).show();
                    progressBar.setVisibility(View.GONE);

                    new Handler().postDelayed(new Runnable() {
                        @Override
                        public void run() {
                            assert loginModel != null;
```

```
                            Intent intent = new Intent(LoginActivity.this,
MainActivity.class).putExtra("accessToken", loginModel.getAccessToken())
                                    .putExtra("username",
loginModel.getUser().getUsername())
                                    .putExtra("email",
loginModel.getUser().getEmail()).putExtra("phone",
loginModel.getUser().getPhone())
                                    .putExtra("id",
Integer.toString(loginModel.getUser().getId()))
                                    .putExtra("pswrd",userPassword);
                            startActivity(intent);
                            LoginActivity.this.finish();


                        }},1000);
                }
                else {
                    assert loginModel != null;
                    Toast.makeText(LoginActivity.this,
loginModel.getMessage(), Toast.LENGTH_SHORT).show();
                }
            }
            @Override
            public void onFailure(@NonNull Call<UserModel> call, Throwable
t) {


                Toast.makeText(LoginActivity.this,         t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
    }}

package com.example.lapkimarket.activities;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.example.lapkimarket.R;
```

```
import com.example.lapkimarket.models.RegRequest;
import com.example.lapkimarket.models.UserModel;
import com.example.lapkimarket.retrofit.RetrofitController;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class RegistryActivity extends AppCompatActivity {

    Button signUp;
    TextView signIn;
    EditText name, email, password, phone;
    ProgressBar progressBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registry);
        progressBar = findViewById(R.id.progressbar);
        progressBar.setVisibility(View.GONE);

        signUp = findViewById(R.id.reg_btn);
        name = findViewById(R.id.name);
        email = findViewById(R.id.email_registry);
```

```
        password = findViewById(R.id.password_registry);
        phone = findViewById(R.id.phone_registry);
        signIn = findViewById(R.id.sign_in);

        signIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new            Intent(RegistryActivity.this,
LoginActivity.class));
            }
        });
        signUp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                createUser();
                progressBar.setVisibility(View.VISIBLE);
                Toast.makeText(RegistryActivity.this,    "Please    wait",
Toast.LENGTH_SHORT).show();

            }
        });
    }
    private void createUser() {

        String userName = name.getText().toString();
        String userEmail = email.getText().toString();
        String userPassword = password.getText().toString();
        int userPhone = Integer.parseInt(phone.getText().toString());

        if (TextUtils.isEmpty(userName)) {
            Toast.makeText(this,         "Name         is         Empty!",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(userEmail)) {
            Toast.makeText(this,         "Email        is         Empty!",
Toast.LENGTH_SHORT).show();
```

```
            return;
        }
        if (TextUtils.isEmpty(userPassword)) {
            Toast.makeText(this,        "Password        is        Empty!",
Toast.LENGTH_SHORT).show();
            return;
        }
        if (userPassword.length() < 6) {
            Toast.makeText(this, "Password length must be greater then 6
letter!", Toast.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(phone.getText().toString())) {
            Toast.makeText(this,         "Phone         is         Empty!",
Toast.LENGTH_SHORT).show();
            return;
        }
        //Create User

        Call<UserModel>  call  =  RetrofitController.getApi().regUser(new
RegRequest(userName,userPassword,userEmail,userPhone));
        call.enqueue(new Callback<UserModel>() {
            @Override
            public void onResponse(@NonNull Call<UserModel> call, @NonNull
Response<UserModel> response) {

                UserModel loginModel = response.body();
                if (response.isSuccessful()){

                    Toast.makeText(RegistryActivity.this,        "Registry
Successfully", Toast.LENGTH_SHORT).show();
                    progressBar.setVisibility(View.GONE);


                    new Handler().postDelayed(new Runnable() {
                        @Override
                        public void run() {
```

```
                    Intent intent = new Intent(RegistryActivity.this,
LoginActivity.class);
                    startActivity(intent);
                    RegistryActivity.this.finish();
                        }},1000);

                }
                else {
                    assert loginModel != null;
                    Toast.makeText(RegistryActivity.this,
loginModel.getMessage(), Toast.LENGTH_SHORT).show();
                }
            }

            @Override
            public void onFailure(@NonNull Call<UserModel> call, Throwable
t) {

                Toast.makeText(RegistryActivity.this,      t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
    }}

package com.example.lapkimarket;

import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatDelegate;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
```

```
import androidx.navigation.ui.NavigationUI;

import com.example.lapkimarket.databinding.ActivityMainBinding;
import com.google.android.material.navigation.NavigationView;

public class MainActivity extends AppCompatActivity {

    private AppBarConfiguration mAppBarConfiguration;
    private ActivityMainBinding binding;

    TextView nameT;
    TextView emailT;

    String email,name,phone,username;

    private String psw;


    @Override
    protected void onCreate(Bundle savedInstanceState) {


AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
        super.onCreate(savedInstanceState);

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarMain.toolbar);

        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_category,R.id.nav_home,        R.id.nav_my_cart,
R.id.nav_my_order, R.id.nav_profile, R.id.nav_info)
```

```
                .setOpenableLayout(drawer)
                .build();
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        NavigationUI.setupActionBarWithNavController(this, navController,
mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView,
navController);

        View headView = navigationView.getHeaderView(0);
        nameT = headView.findViewById(R.id.name_textView);
        emailT = headView.findViewById(R.id.email_textView);

        Intent intent = getIntent();
         name = intent.getStringExtra("accessToken");
         email = intent.getStringExtra("email");
         username = intent.getStringExtra("username");
         phone = intent.getStringExtra("phone");
         psw = intent.getStringExtra("pswrd");
         int userid = Integer.parseInt(intent.getStringExtra("id"));

        nameT.setText(username);
        emailT.setText(email);

    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onSupportNavigateUp() {
```

```
        NavController navController = Navigation.findNavController(this,
R.id.nav_host_fragment_content_main);
        return                      NavigationUI.navigateUp(navController,
mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }
}
package com.example.lapkimarket.retrofit;


import com.example.lapkimarket.models.AllProductModel;
import com.example.lapkimarket.models.CartModel;
import com.example.lapkimarket.models.CategoryModel;
import com.example.lapkimarket.models.LoginRequest;
import com.example.lapkimarket.models.OrderHistoryModel;
import com.example.lapkimarket.models.OrderModel;
import com.example.lapkimarket.models.ProductModel;
import com.example.lapkimarket.models.RegRequest;
import com.example.lapkimarket.models.SubcategoryModel;
import com.example.lapkimarket.models.UserModel;

import java.util.List;

import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.GET;
import retrofit2.http.HTTP;
import retrofit2.http.Header;
import retrofit2.http.POST;
import retrofit2.http.PUT;
import retrofit2.http.Path;
import retrofit2.http.Query;

public interface ApiRetrofit {

    @POST("api/user/reg")
    Call<UserModel> regUser(@Body RegRequest regRequest);
```

```
    @POST("api/user/login")
    Call<UserModel> loginUser(@Body LoginRequest loginRequest);

    @GET("api/category/")
    Call<List<CategoryModel>>   getCategory(@Header("Authorization")String
token);

    @GET("api/type/{id}")
    Call<List<SubcategoryModel>>
getSubCategory(@Header("Authorization")String  token,  @Path("id")  String
id);

    @GET("api/item/")
    Call<List<ProductModel>>    getProducts(@Header("Authorization")String
token, @Query("typeId") String typeId);

    @GET("api/item")
    Call<List<AllProductModel>> getAllProducts();

    @POST("api/basket")
    Call<CartModel>  addToCart(@Header("Authorization")String  token,  @Body
CartModel cartModel);

    @GET("api/basket/{id}")
    Call<CartModel>   getUserCart(@Header("Authorization")String   token,
@Path("id") String id);

    @HTTP(method = "DELETE", path = "api/basket", hasBody = true)
    Call<CartModel>  deleteFromCart(@Header("Authorization")String  token,
@Body CartModel cartModel);

    @GET("api/user/logout")
    Call<UserModel> logOut();

    @PUT("api/user/")
    Call<UserModel>  updateUser(@Header("Authorization")String  token,@Body
RegRequest regRequest);
```

```
    @POST("api/order/")
    Call<OrderModel>  payOrder(@Header("Authorization")String  token,@Body
OrderModel orderModel);


    @GET("api/order-history/")
    Call<OrderHistoryModel> getOrderHistory(@Header("Authorization")String
token);


}
package com.example.lapkimarket.retrofit;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RetrofitController {

    private    static    final    String    url    =    "https://lapki-
market.herokuapp.com/";

    public static Retrofit getClient()
    {

        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(url)
                .addConverterFactory(GsonConverterFactory.create())
                .build();

        return retrofit;
    }
    public static ApiRetrofit getApi(){

        return getClient().create(ApiRetrofit.class);
    }
}
package com.example.lapkimarket.models;

public class AllProductModel {
```

```
    private String name;
    private String description;
    private String photo;
    private int price;
    private int id;

    public AllProductModel(String name, String description, String photo,
int price, int id) {
        this.name = name;
        this.description = description;
        this.photo = photo;
        this.price = price;
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getPhoto() {
        return ("https://lapki-market.herokuapp.com/"+photo);
    }

    public void setPhoto(String photo) {
```

```
        this.photo = photo;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}
package com.example.lapkimarket.ui.category;

import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ProgressBar;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
```

```
import com.example.lapkimarket.R;
import com.example.lapkimarket.activities.TypeActivity;
import com.example.lapkimarket.adapters.CategoryAdapter;
import com.example.lapkimarket.models.CategoryModel;
import com.example.lapkimarket.retrofit.RetrofitController;

import java.util.List;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class CategoryFragment extends Fragment implements
CategoryAdapter.CategoryInterface {

    CategoryAdapter categoryAdapter;
    RecyclerView recyclerView;
    ProgressBar progressbar;

    CategoryFragment hadler = this;
    String tokenName;

    String email,username,phone,userid;

    public View onCreateView(@NonNull LayoutInflater inflater,
                             ViewGroup container, Bundle
savedInstanceState) {
         View root =
inflater.inflate(R.layout.fragment_category,container,false);

        progressbar = root.findViewById(R.id.progressbar);
        progressbar.setVisibility(View.VISIBLE);

        recyclerView = root.findViewById(R.id.category01);

        Intent intent = requireActivity().getIntent();
        tokenName = intent.getStringExtra("accessToken");
         email = intent.getStringExtra("email");
```

```
        username = intent.getStringExtra("username");
        phone = intent.getStringExtra("phone");
        userid = intent.getStringExtra("id");


        Call<List<CategoryModel>> call = RetrofitController.getApi().getCategory("Bearer "+tokenName);
        call.enqueue(new Callback<List<CategoryModel>>() {
            @SuppressLint("NotifyDataSetChanged")
            @Override
            public void onResponse(@NonNull Call<List<CategoryModel>> call, @NonNull Response<List<CategoryModel>> response) {
                progressbar.setVisibility(View.GONE);
                if (response.isSuccessful()){

                    categoryAdapter.setCategoryModelList(response.body());

                    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
                    recyclerView.setAdapter(categoryAdapter);
                    categoryAdapter.notifyDataSetChanged();

                }else {

                    Toast.makeText(getContext(), "Failed to get categories", Toast.LENGTH_SHORT).show();

                }
            }
            @Override
            public void onFailure(Call<List<CategoryModel>> call, Throwable t) {
                progressbar.setVisibility(View.GONE);
                Toast.makeText(getContext(), t.getMessage(), Toast.LENGTH_SHORT).show();
            }
        });
        return root;
```

```
    }

    @Override
    public void onAttach(@NonNull Context context) {
        super.onAttach(context);
        categoryAdapter = new CategoryAdapter(this);
    }
     public void getSubcategory(String idOfType) {

          startActivity(new                              Intent(getContext(),
TypeActivity.class).putExtra("idOfType", idOfType).putExtra("accessToken",
tokenName).putExtra("username",                 username).putExtra("email",
email).putExtra("phone", phone).putExtra("id", userid));
    }
    @Override
    public void categorymeth(String id) {
        getSubcategory(id);

    }
}
```