Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут комп’ютерних систем
Кафедра інформаційних систем

Кравчишина Олена Олександрівна,
Шкуропатова Єлизавета Валеріївна
студентки групи АІ-214

ДИСЦИПЛІНА
Програмування мобільних пристроїв

КУРСОВА РОБОТА
Створення мобільного додатку покупки кросівок

Спеціальність:
122 Комп’ютерні науки

Освітня програма:
Комп’ютерні науки

Керівник:
Годовиченко Микола Анатолійович,
кандидат технічних наук, доцент

Одеса – 2024

ЗМІСТ

АНОТАЦІЯ

Ця курсова робота присвячена розробці мобільного додатку на платформі Android з використанням мови програмування Kotlin. Додаток "Step-into-Style" створений для покупки кросівок, де користувачі можуть переглядати різні моделі, додавати їх у корзину, редагувати кількість товарів та переглядати загальну суму покупки.

Основними функціональними можливостями додатку є відображення асортименту товарів, управління корзиною покупок та редагування вмісту корзини. Реалізація додатку включає в себе використання архітектурних паттернів, таких як MVVM, а також використання баз даних для зберігання інформації про товари та покупки користувачів.

Курсова робота охоплює весь процес розробки від концепції до реалізації, включаючи проектування інтерфейсу користувача та програмування бізнес-логіки додатку.

ABSTRACT

This coursework is devoted to the development of a mobile application on the Android platform using the Kotlin programming language. The Step-into-Style app is designed for sneaker shopping, where users can browse different models, add them to the cart, edit the number of products and view the total amount of the purchase.

The main functionality of the application is to display the range of products, manage the shopping cart and edit the content of the cart. Application implementation includes the use of architectural patterns such as MVVM, as well as the use of databases to store information about products and user purchases.

The coursework covers the entire development process from concept to implementation, including user interface design and application business logic programming.

ВСТУП

В сучасному світі мобільні додатки стали невід'ємною частиною повсякденного життя. З їх допомогою можна виконувати найрізноманітніші завдання: від спілкування та розваг до управління фінансами та здоров'ям. Враховуючи швидкий розвиток технологій та зростаючу популярність мобільних пристроїв, розробка мобільних додатків є актуальною і перспективною сферою діяльності.

Розробка мобільних додатків - це захоплюючий процес, що поєднує технологічні досягнення та творчий потенціал. Кожна розробка починається з ретельного вивчення потреб користувачів та визначення цілей проекту. Наша команда ретельно досліджувала потреби та очікування наших користувачів, і саме на цьому ми зосереджуємося у нашому проекті.

Одним із найпопулярніших і широко використовуваних мов програмування для створення мобільних додатків є Kotlin. Завдяки своїй сумісності з Java та наявності сучасних функціональних можливостей, Kotlin швидко завоював довіру розробників і став офіційною мовою для розробки додатків під операційну систему Android.

Темою даної курсової роботи є "Створення мобільного додатку покупки кросівок на Kotlin". Метою роботи є розробка функціонального мобільного додатку "Step into Style", який дозволить користувачам здійснювати вибір, перегляд та купівлю кросівок за допомогою своїх мобільних пристроїв.

У ході роботи будуть розглянуті такі аспекти:

1. Аналіз ринку мобільних додатків для продажу взуття.
2. Огляд технологій та інструментів, які будуть використані для розробки додатку.

3. Проектування та реалізація мобільного додатку з використанням мови програмування Kotlin.
4. Тестування та налагодження розробленого додатку.
5. Оцінка результатів та перспективи подальшого розвитку додатку.

Ця курсова робота націлена на те, щоб продемонструвати можливості мови програмування Kotlin у розробці мобільних додатків, а також запропонувати ефективне рішення для зручної і швидкої купівлі кросівок через мобільні пристрої.

1 ОГЛЯД СИСТЕМ-АНАЛОГІВ ТА ТЕХНОЛОГІЙ ЇХ РОЗРОБКИ

1.1 Особливості використання мобільних технологій для розробки мобільних додатків в Android Studio.

Мобільні технології стрімко розвиваються, і сьогодні вони є важливою складовою нашого повсякденного життя. Мобільні додатки стали потужним інструментом для взаємодії з користувачами, надання послуг та розваг. Розробка мобільних додатків - це захоплюючий процес, що поєднує технологічні досягнення та творчий потенціал. Кожна розробка починається з ретельного вивчення потреб користувачів та визначення цілей проекту. Наша команда ретельно досліджувала потреби та очікування наших користувачів, і саме на цьому ми зосереджуємося у нашому проекті.

Одним із найпопулярніших і широко використовуваних мов програмування для створення мобільних додатків є Kotlin. Завдяки своїй сумісності з Java та наявності сучасних функціональних можливостей, Kotlin швидко завоював довіру розробників і став офіційною мовою для розробки додатків під операційну систему Android.

Android Studio є офіційною інтегрованою середовищем розробки (IDE) для Android, створеною на базі IntelliJ IDEA. Ця платформа надає розробникам широкий спектр інструментів та функцій для створення високоякісних мобільних додатків. Особливості використання мобільних технологій для розробки додатків в Android Studio включають:

1. Середовище розробки

Android Studio забезпечує розробників усіма необхідними інструментами для написання, тестування та налагодження коду. Вона включає редактор коду з

підтримкою підсвічування синтаксису, автоматичним завершенням коду, рефакторингом та іншими функціями, що сприяють швидкому і ефективному написанню коду.

2. Інтеграція з Kotlin

Android Studio повністю підтримує мову програмування Kotlin, що дозволяє розробникам використовувати всі переваги цієї мови, включаючи зменшення кількості коду, підвищення продуктивності та покращену безпеку коду.

3. Дизайн інтерфейсу користувача

Android Studio надає потужні інструменти для проектування інтерфейсів користувача. Конструктор макетів (Layout Editor) дозволяє розробникам створювати інтерактивні інтерфейси за допомогою перетягування (drag-and-drop) компонентів, а також налаштовувати атрибути елементів у візуальному режимі.

4. Системи збірки

Android Studio використовує Gradle як систему збірки, що забезпечує гнучке і потужне управління залежностями, конфігураціями збірки та автоматизацією завдань. Це дозволяє легко налаштовувати і оптимізувати процес збірки додатку.

5. Тестування та налагодження

Android Studio включає вбудовані інструменти для тестування та налагодження додатків. Розробники можуть використовувати емулятори для тестування додатків на різних версіях Android та різних пристроях. Інструменти налагодження допомагають швидко знаходити і виправляти помилки, а також аналізувати продуктивність додатку.

6. Інтеграція з Firebase

Android Studio пропонує інтеграцію з хмарними сервісами Firebase, що дозволяє додавати функції, такі як аутентифікація, база даних у реальному часі, аналітика, хостинг та інші, безпосередньо у додаток.

7. Підтримка сучасних технологій

Android Studio регулярно оновлюється, щоб підтримувати новітні технології та фреймворки, такі як Jetpack Compose, що забезпечує створення сучасних та адаптивних інтерфейсів користувача.

1.2 Огляд аналогічних мобільних додатків

Для розробки мобільного додатку покупки взуття необхідно з'ясувати, які функції повинні бути доступні в ньому. Для досягнення цього мети можна провести дослідження та аналіз інших додатків з покупки взуття, що вже існують на мобільних платформах. Це дозволить встановити особливості роботи цих додатків та з'ясувати позитивні та негативні моменти їх використання. Аналіз існуючих може допомогти зрозуміти, які особливості та функції є популярними серед користувачів, а також виявити позитивні та негативні аспекти їх використання. Також цей аналіз допоможе краще зрозуміти потреби користувачів в цій області. Після проведення пошуку в Інтернеті, були знайдені найбільш популярні та відомі додатки, які будуть використовуватися як аналоги в даній роботі:

- додаток Intertop [7];
- додаток StreatBeet [8];

Далі потрібно провести аналіз цих додатків з метою встановлення особливостей їх роботи та з'ясування позитивних та негативних моментів їх використання. Це допоможе краще зрозуміти, які функції повинні бути доступні

мобільному додатку для створення заміток, та відповісти на потреби користувачів в цій області.

1.2.1 Додаток "Intertop"

Особливості роботи:

Intertop – це мобільний додаток для онлайн-покупок взуття, одягу та аксесуарів, доступний для завантаження в Україні та Казахстані. Додаток надає користувачам можливість переглядати, вибирати та купувати товари від понад 100 світових брендів, зручне управління замовленнями та різні варіанти доставки. Основна мета Intertop – зробити процес онлайн-шопінгу максимально зручним та приємним для користувачів.

Широкий асортимент товарів: У додатку представлено близько 77 000 товарів від понад 100 світових брендів. Регулярне оновлення асортименту з новими надходженнями.

Персоналізація: Користувачі можуть налаштовувати свій профіль, обираючи улюблені бренди та стилі.Можливість зберігати обрані товари та бренди в розділі "Обране" для швидкого доступу.

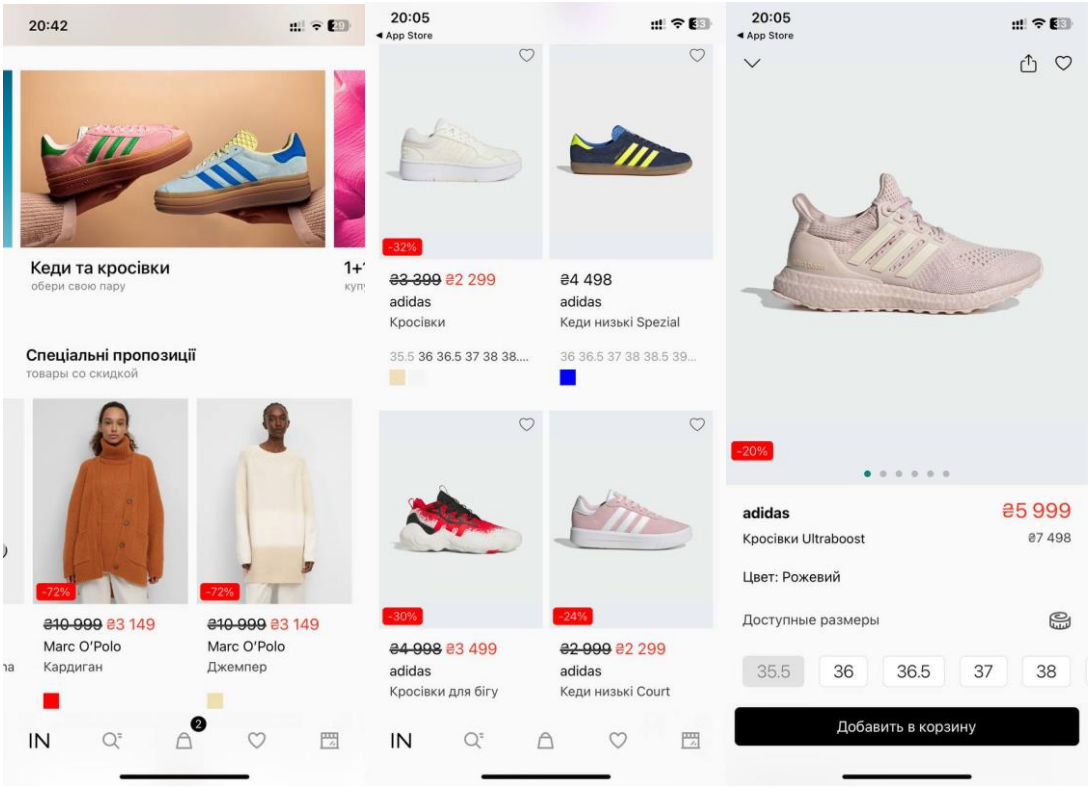
Інформування про спеціальні пропозиції: Користувачі отримують повідомлення про акції та спеціальні пропозиції першими.

Управління замовленнями: Зручний інтерфейс для перегляду та управління замовленнями в особистому профілі.Можливість додавати товари в кошик, змінювати їх кількість та оформлювати замовлення.

Варіанти доставки:Можливість обрати самовивіз з магазинів. Доставка товарів на пошту або додому кур'єром.

Позитивні аспекти:

- Можливість вибору з великої кількості товарів від провідних світових брендів.
- Індивідуальний підхід до кожного користувача, можливість налаштування профілю та отримання персональних рекомендацій.
- Оперативне отримання інформації про нові надходження, спеціальні пропозиції та акції.
- Інтуїтивно зрозумілий інтерфейс, що дозволяє легко здійснювати покупки та управляти замовленнями.



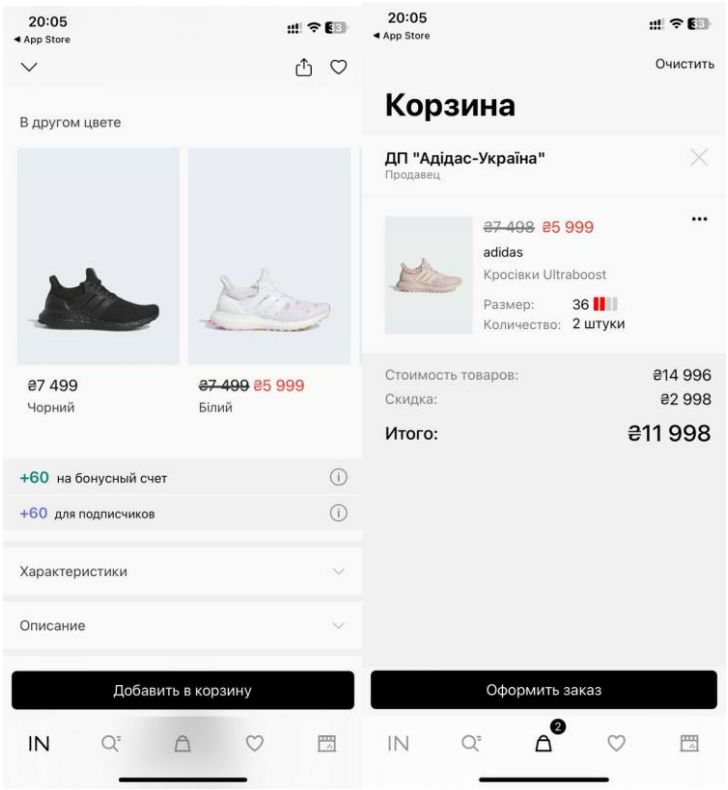


Рисунок 1.1- 1.5 – Знімки екрану додатку "Intertop"

Негативні аспекти: На даний момент доставка доступна тільки в Україні та Казахстані, що обмежує можливості користувачів з інших країн. Через різні зовнішні фактори можуть виникати затримки у доставці, що може вплинути на задоволеність клієнтів.

1.2.2 Додаток "StreatBeet"

Street Beat – це мережа мультибрендових магазинів взуття, одягу та аксесуарів від світових спортивних і лайфстайл брендів, таких як Nike, adidas Originals, Reebok Classic, New Balance, Asics, Puma, Vans, Converse, Timberland, The North Face, Salomon та інші. Додаток Street Beat дозволяє користувачам здійснювати покупки онлайн з будь-якого місця, забезпечуючи доступ до найширшого асортименту брендових товарів.

Особливості роботи:

Широкий асортимент товарів: Величезний вибір взуття, одягу та аксесуарів від провідних світових брендів. Постійне оновлення каталогу з новими релізами та колабораціями.

Зручний інтерфейс додатка: Повний каталог сайту в зручному форматі для мобільних пристроїв.

Легкий доступ до інформації про товари через особистий кабінет з можливістю додавання товарів в "Обране" та переглядом історії замовлень.

Потужні функції пошуку: Швидкий пошук товарів за штрих-кодом, артикулом або назвою.

Зручні фільтри для сортування товарів за брендами, стилем та розмірами, включаючи дитячі розміри в сантиметрах.

Геолокація магазинів: Можливість знайти найближчі фізичні магазини на карті для зручності користувачів.

Статус замовлень: Відстеження статусу замовлень в режимі онлайн для прозорості та зручності.

Позитивні аспекти:

- Можливість вибору з широкого асортименту товарів від провідних світових брендів, що задовольняє будь-які потреби та смаки.
- Інтуїтивно зрозумілий інтерфейс додатка, що дозволяє легко здійснювати покупки, відстежувати замовлення та зберігати обрані товари.
- Доступ до спеціальних пропозицій, знижок та нових релізів, які забезпечують додаткову цінність для користувачів.
- Персоналізація: Індивідуальний підхід до кожного користувача через налаштування особистого кабінету та отримання персональних рекомендацій.

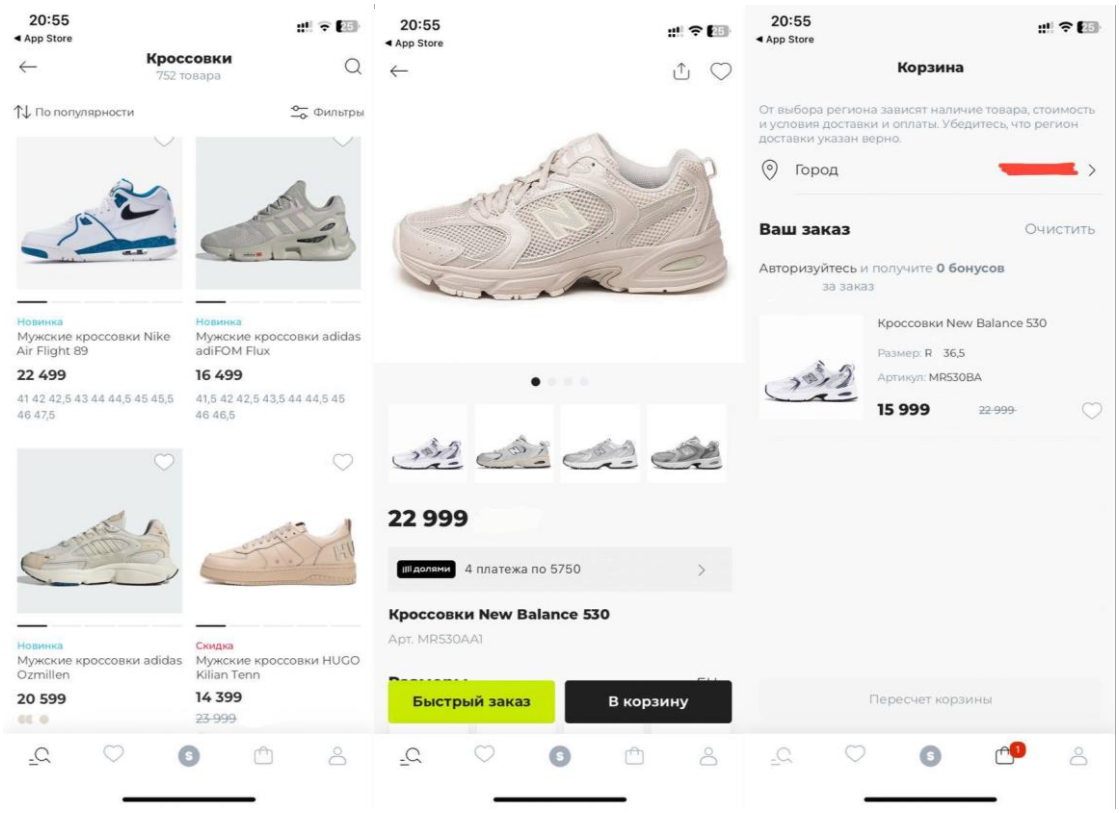


Рисунок 1.6 – 1.8 – Знімки екрану додатку "StreatBeet"

Негативні аспекти: На даний момент додаток підтримує доставку товарів тільки в одну країну, що обмежує можливості користувачів з інших регіонів. Через зовнішні фактори, такі як митні процедури або логістичні проблеми, можуть виникати затримки в доставці, що може вплинути на задоволеність клієнтів.

1.3 Формулювання вимог до основних функцій мобільного додатка

Перегляд моделей кросівок:

- Користувачі можуть переглядати різні моделі кросівок, доступні для покупки.
- Кожна модель повинна мати інформацію про назву, зображення, розміри, колір і ціну.

Додавання товарів у корзину:

- Користувачі можуть додавати обрані моделі кросівок у свою корзину для подальшої покупки.
- Вибір розмірної сітки та кількості товару має бути доступним для зручності користувача.

Редагування кількості товарів у корзині:

- Користувачі можуть змінювати кількість кросівок кожного виду або видаляти їх зі списку покупок.
- Зміни в кількості товарів автоматично відображаються в загальній сумі покупки.

Перегляд загальної суми покупки:

- Користувачі можуть переглядати загальну суму всіх товарів, які вони вибрали для покупки, перед оформленням замовлення.

1.4 Огляд інформаційних технологій для розробки мобільного додатка в Android Studio

Android Studio надає широкий спектр інформаційних технологій, які забезпечують ефективну та продуктивну розробку мобільних додатків для платформи Android. Ось деякі з них:

1. **Мова програмування Kotlin**: Kotlin, як основна мова програмування для Android, забезпечує сучасний синтаксис, високий рівень безпеки та сумісність з Java. Це дозволяє розробникам використовувати існуючий код і бібліотеки Java, а також створювати нові проекти з мінімальними зусиллями.
2. **Jetpack Components**: Android Jetpack - це набір бібліотек, які допомагають розробникам створювати кращі додатки. До них входять компоненти для управління життєвим циклом додатку, роботи з базами даних, інтерфейсом користувача та інші.
3. **Android SDK**: Android Software Development Kit (SDK) містить інструменти для створення додатків, включаючи бібліотеки, приклади коду, емулятори та документацію. SDK постійно оновлюється для підтримки нових версій Android.
4. **Gradle**: Система збірки Gradle дозволяє автоматизувати процес збірки, управління залежностями та конфігурації проекту. Це робить процес розробки більш організованим та ефективним.
5. **Firebase**: Firebase - це платформа для мобільних і веб-розробників, яка надає широкий спектр інструментів для розробки, тестування та моніторингу додатків. Firebase включає такі сервіси, як аутентифікація користувачів, база даних у реальному часі, аналітика, хостинг та інші.
6. **ARCore**: Для розробки додатків з доповненою реальністю Android Studio підтримує ARCore, що дозволяє створювати інтерактивні та захоплюючі додатки з використанням технологій доповненої реальності.

7. **ML Kit**: Для інтеграції можливостей машинного навчання в мобільні додатки розробники можуть використовувати ML Kit. Це набір інструментів та API для додавання функцій машинного навчання, таких як розпізнавання облич, тексту, об'єктів і багато іншого.
8. **Android Emulator**: Інструмент, який дозволяє тестувати додатки на різних версіях Android та на різних пристроях без необхідності фізичних пристроїв. Це значно спрощує процес тестування та налагодження.

Ці технології, які надає Android Studio, забезпечують комплексний підхід до розробки мобільних додатків, дозволяючи створювати функціональні, продуктивні та безпечні рішення, які відповідають вимогам сучасних користувачів.

1.4.1 Використання Java або Kotlin для розробки мобільних додатків

Java була основною мовою для розробки Android-додатків протягом багатьох років. Вона заслужила популярність завдяки своїм перевагам:

Переваги Java:

1. Історична основа: Java використовується для Android-розробки з самого початку, тому вона має велику кількість досвідчених розробників і добре вивірену екосистему.
2. Широкі можливості: Через довгий час свого існування Java накопичила величезну кількість бібліотек, фреймворків і інструментів для розробки, що робить її ідеальним вибором для великих або старих проектів.
3. Стабільність і надійність: Java є відомою своєю стабільністю і надійністю, що робить її привабливим вибором для підприємницьких ініціатив і довгострокових проектів.

Мінуси Java:

1. Більше коду: Написання додатків на Java може вимагати більше шаблонного коду порівняно з сучаснішими мовами, такими як Kotlin.
2. Менша експресивність: Java не має деяких сучасних функцій, які присутні у Kotlin, що може впливати на продуктивність розробників і зрозумілість коду.

Kotlin для розробки мобільних додатків у Android Studio

Kotlin зарекомендувала себе як сучасна альтернатива Java для Android-розробки, вона має свої унікальні переваги:

Переваги Kotlin:

1. Менше шаблонного коду: Kotlin забезпечує коротший синтаксис і меншу кількість бойового коду, що робить розробку більш швидкою і ефективною.
2. Сучасні функції: Kotlin включає в себе сучасні функції, такі як нульова безпека типів, розширення функцій, лямбди та інші покращення, що спрощують розробку і покращують зрозумілість коду.
3. Повна інтероперабельність з Java: Kotlin повністю сумісний з Java, що дозволяє поступову міграцію існуючих проектів або використання обох мов в одному проекті без проблем.

Мінуси Kotlin:

1. Вивчення нової мови: Для команд, які вже мають досвід з Java, вивчення Kotlin може зайняти певний час і вимагати додаткових зусиль.
2. Час міграції: Перехід існуючих проектів з Java на Kotlin може вимагати часу і зусиль для адаптації існуючого коду.

Вибір між Java і Kotlin

Вибір між Java і Kotlin залежить від конкретних потреб вашого проекту, ваших уподобань та рівня досвіду команди. Kotlin набуває популярності серед Android-розробників завдяки своїм сучасним можливостям і ефективності, але використання Java також залишається відмінним варіантом для великих або старих проектів з великою базою Java-коду.

У контексті курсової роботи про вибір мови програмування для розробки мобільних додатків у середовищі Android Studio, Kotlin виявляється відмінним вибором. Нижче наведено ключові аспекти, які підтверджують його переваги:

1. Сучасність та ефективність: Kotlin є сучасною мовою програмування з багатьма сучасними функціями, які спрощують розробку і підвищують продуктивність розробників. Вона пропонує коротший і більш зрозумілий синтаксис порівняно з Java, що дозволяє зменшити кількість написаного коду і полегшити розуміння програмного продукту.
2. Інтероперабельність з Java: Kotlin повністю сумісний з Java, що дозволяє командам поступово переходити на нову мову без втрати сумісності з існуючим Java-кодом. Це знижує ризик і полегшує міграцію великих проектів.
3. Підтримка від Google: Kotlin був офіційно підтриманий Google як основна мова для розробки Android-додатків, що підкреслює його майбутню перспективність і важливість для Android-спільноти.

Отже, вибір Kotlin для даного проекту в Android Studio буде обґрунтованим рішенням, оскільки ця мова пропонує сучасні можливості, високу продуктивність та гарну підтримку спільноти.

1.4.2 Архітектурні шаблони (MVVM) додатка в Android Studio

У Android-розробці одним з найпопулярніших архітектурних шаблонів є MVVM (Model-View-ViewModel). Цей шаблон дозволяє ефективно розділити логіку бізнес-логіки від її візуального відображення, що полегшує тестування, підтримку та розширення додатків. Ось детальний опис кожної складової MVVM (рис.9):

Model-View-ViewModel (MVVM)

1. Модель (Model):
   - Опис: Модель представляє собою даний об'єкт або об'єкт доступу до даних, який зазвичай містить бізнес-логіку і взаємодіє з джерелами даних, такими як база даних, API або репозиторії.
   - Відповідальність: Забезпечує доступ до даних і виконання операцій з ними, не залежно від того, як ці дані відображаються або яким чином взаємодіють з користувачем.
2. Вид (View):
   - Опис: Вид відповідає за представлення даних користувачу. Це може бути активність (Activity), фрагмент (Fragment) або віджет.
   - Відповідальність: Відображення даних, взаємодія з користувачем, обробка введення користувача. Від виду очікується мінімальна обробка бізнес-логіки.
3. **Модель видом (ViewModel)**:
   - Опис: ViewModel є посередником між Моделлю і Видом. Вона містить бізнес-логіку для управління станом даних і підготовки їх для відображення в Виді.
   - Відповідальність: Зберігання та управління станом даних, логіка взаємодії з Видом, обробка введення від користувача і реагування на зміни в Моделі.

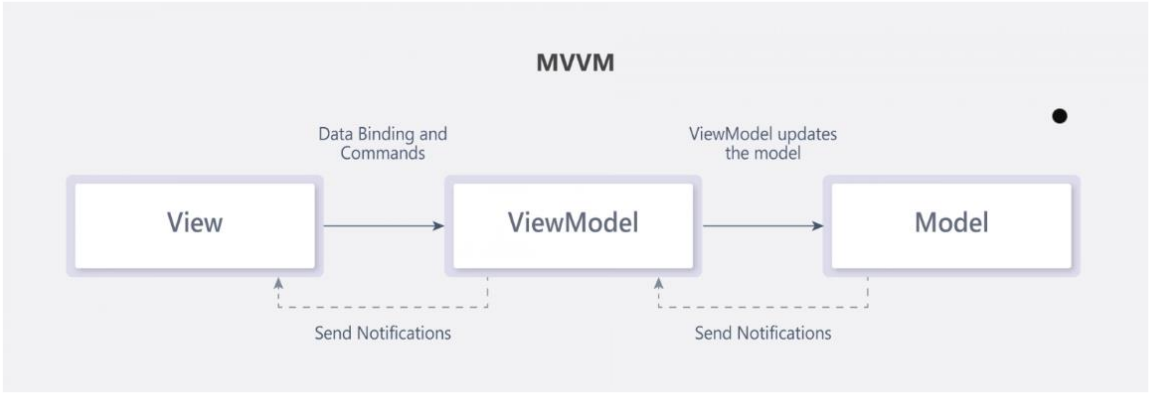


Рисунок 1.9 - Компоненти MVVM

Основні переваги MVVM:

- **Розділення обов'язків**: MVVM дозволяє чітко визначити, де знаходиться бізнес-логіка, де знаходиться логіка відображення і як вони взаємодіють.
- **Тестованість**: Кожна складова (Модель, Вид, ViewModel) може бути легко тестована окремо, що спрощує їх модульне тестування.
- **Підтримка від Android і Google**: Google активно рекомендує використовувати архітектурні шаблони, такі як MVVM, для створення добре структурованих і підтримуваних додатків.

Реалізація MVVM в Android Studio

Для реалізації MVVM в Android Studio ви можете використовувати такі компоненти:

- **ViewModel**: Використовуйте ViewModel з архітектурним компонентом ViewModel, який дозволяє зберігати та управляти станом даних між конфігураціями активностей або фрагментів.

- **LiveData**: LiveData дозволяє створювати дані, які можна спостерігати в Виді і автоматично оновлювати інтерфейс користувача при зміні даних.
- **BindingAdapter**: Використовуйте BindingAdapter, щоб зв'язати дані з Видом і ViewModel, що дозволяє автоматично оновлювати інтерфейс користувача при зміні даних у ViewModel.

MVVM є потужним архітектурним шаблоном для розробки мобільних додатків у Android Studio, який дозволяє чітко визначати обов'язки різних компонентів додатку і полегшує тестування і підтримку. Використання MVVM з Kotlin дозволить вам створювати ефективні, структуровані та легко розширювані додатки для платформи Android.

1.5 Висновки до першого розділу

У першому розділі курсової роботи було проведено огляд різних систем-аналогів та технологій розробки мобільних додатків, зокрема у середовищі Android Studio. Основні висновки з кожного підрозділу такі:

1. **Особливості використання мобільних технологій для розробки мобільних додатків в Android Studio**:
    - Android Studio є основним інтегрованим середовищем розробки (IDE) для створення додатків для платформи Android.
    - Використання Android SDK, мов програмування Kotlin або Java, а також розуміння архітектурних шаблонів дозволяє розробникам створювати потужні і ефективні додатки.
2. **Огляд аналогічних мобільних додатків**:
    - Додаток "Intertop" та "StreatBeet" були розглянуті як приклади схожих мобільних додатків. Кожен з них має свої особливості у функціональності та використанні технологій.

3. **Формулювання вимог до основних функцій мобільного додатка**:
    - Вимоги до основних функцій мобільного додатка були сформульовані на основі аналізу існуючих додатків та потреб користувачів.
4. **Огляд інформаційних технологій для розробки мобільного додатка в Android Studio**:
    - Використання мов програмування Java або Kotlin для розробки мобільних додатків, а також використання архітектурного шаблону MVVM, є ключовими компонентами для успішної розробки додатка в Android Studio.

Загалом, перший розділ надав загальний огляд ключових аспектів розробки мобільних додатків, включаючи вибір технологій, аналіз аналогів, формулювання вимог і вивчення інформаційних технологій. Ці знання стануть основою для подальшого розгортання та реалізації самого додатка у наступних розділах курсової роботи.

2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ДОДАТКА

2.1 Цілі та завдання мобільного додатка

1. Забезпечення зручного користувацького інтерфейсу: Розробка інтуїтивно зрозумілого і зручного інтерфейсу для користувачів, щобвони могли легко переглядати і купувати кросівки.
2. Функціональність для покупок кросівок: Реалізація функціоналу додавання кросівок в корзину, зміни кількості товарів, перегляду історії замовлень та оформлення покупок.
3. Персоналізація і рекомендації: Надання персоналізованих рекомендацій користувачам на основі їх попередніх покупок або вподобань.
4. Взаємодія з товарами: Можливість вибору кросівок за кольором, розміром і іншими параметрами для зручного пошуку товарів, які відповідають вимогам користувача.
5. Залучення і утримання користувачів: Розробка функціоналу, що сприяє залученню нових користувачів і підтримує існуючих через цікавий контент і зручний досвід користування.

Завдання проектування мобільного додатка "Step-into-Style":

1. **Дизайн інтерфейсу користувача (UI/UX)**:
    - Розробка привабливого і зручного інтерфейсу для всіх екранів додатка, включаючи домашню сторінку, сторінку товару, корзину і вкладку "Подобаються".
2. **Реалізація функціональності додавання товарів в корзину**:

- Розробка механізму додавання кросівок в корзину з можливістю зміни кількості товарів і видалення з корзини.

3. **Інтеграція системи оформлення замовлення**:
    - Створення процесу оформлення замовлення з введенням адреси доставки і вибором методу оплати.
4. **Реалізація вкладки "Подобаються"**:
    - Розробка функціоналу для позначення товарів як подобаються і їх подальшого перегляду в вкладці "Подобаються".
5. **Тестування і вдосконалення**:
    - Проведення тестування функціональності додатка для виявлення і виправлення помилок, а також для оптимізації досвіду користувача.

2.2 Формулювання користувацьких сценаріїв та історій для мобільного додатка

Діаграма сценаріїв UML (Unified Modeling Language) є одним із ключових інструментів для моделювання взаємодії між користувачами (або зовнішніми системами) та самою системою. Основна мета цієї діаграми - це визначення функціональних вимог до системи через опис акторів та їх взаємодії з системою через різні сценарії використання.

Основні компоненти діаграми сценаріїв включають:

1. **Актори**: Це сутності або зовнішні системи, які взаємодіють з системою. Актори можуть бути користувачами, іншими програмами або навіть зовнішніми

пристроями. Кожен актор має певні ролі в системі, які визначають їхні можливості та обов'язки.

2. **Сценарії взаємодії**: Це конкретні сценарії або дії, які виконуються акторами в системі. Кожен сценарій включає послідовність кроків, що описують, як актор взаємодіє з системою для досягнення певної мети.
3. **Взаємодії**: Це самі взаємодії між акторами та системою, що описуються у кожному сценарії. Взаємодії можуть включати виклики методів, обмін повідомленнями або інші форми обміну даними між акторами та системою.
4. **Функціональні вимоги**: Діаграма сценаріїв допомагає визначити функціональні вимоги до системи через уявлення того, які дії мають підтримуватися системою відповідно до потреб користувачів та інших акторів.
5. **Аналіз залежностей**: Ця діаграма також дозволяє аналізувати залежності між різними акторами та можливими сценаріями використання. Це важливо для визначення, як система повинна взаємодіяти з різними типами користувачів або зовнішніми системами.

Діаграма сценаріїв UML допомагає команді розробників і аналітиків отримати зрозуміння функціональних потреб і можливих сценаріїв використання системи, що є критичним для успішного проектування та розробки програмного забезпечення.

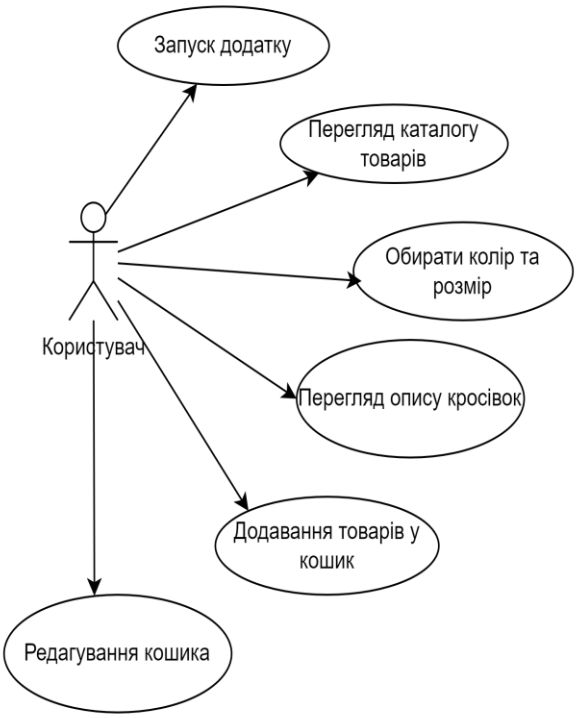


Рисунок 2.1 - Діаграма варіантів використання мобільного додатку

User Story 1: Початковий екран

- Як користувач, я хочу бачити початковий екран з корисною інформацією, щоб швидко зрозуміти, що пропонує додаток.

- Тест 1: Після запуску додатка користувача вітає початковий екран з оглядом ключових функцій та можливостей.
- Тест 2: На початковому екрані показані банери (акції), які можуть зацікавити користувача.
- Тест 3: Користувач може легко знайти навігаційні елементи для переходу до різних частин додатка, таких як головна сторінка, кошик.

User Story 2: Перегляд стрічки взуття

- Як покупець, я хочу переглядати стрічку взуття, щоб бачити нові та популярні моделі взуття від різних брендів.

- Тест 1: Покупець може переглядати стрічку взуття, яка містить представлені взуття від різних брендів.

- Тест 2: Покупець може використовувати горизонтальний скролл для швидкого перегляду різних моделей взуття.
- Тест 3: Покупець може переглядати рейтинг взуття.

User Story 3: Вибір взуття за кольором і розміром

- Як покупець, я хочу вибирати взуття за кольором і розміром, щоб знайти потрібну модель, що відповідає моїм вимогам.
- Тест 1: Покупець може вибрати колір і розмір взуття на сторінці товару.
- Тест 2: Покупець може продовжити переглядати інші варіанти взуття або перейти до додавання товару в кошик.

User Story 4: Додавання взуття в кошик

- Як покупець, я хочу додавати взуття в кошик, щоб оформити замовлення на кілька товарів одночасно.
- Тест 1: Покупець може додати обране взуття в кошик зі сторінки товару.
- Тест 2: Покупець може переглядати вміст кошика.
- Тест 3: Сума замовлення в кошику оновлюється автоматично після додавання або видалення товарів.

User Story 5: Зміна кількості товарів у кошику

- Як покупець, я хочу змінювати кількість товарів у кошику, щоб купити потрібну кількість обраних товарів.
- Тест 1: Покупець може змінювати кількість товару в кошику.
- Тест 2: Кількість товару змінюється миттєво і відображається у кошику.
- Тест 3: Покупець може видалити товар з кошика.
- Тест 4: Сума замовлення оновлюється автоматично після зміни кількості товару у кошику

2.3 Визначення нефункціональних вимог до мобільного додатка

Нефункціональні вимоги до мобільного додатка визначають його нефункціональні аспекти, які описують властивості системи, що не стосуються безпосередньо функціональності програмного забезпечення, але є критичними для забезпечення його ефективності, безпеки, доступності та іншими аспектами:

1. **Продуктивність:**
    - **Час завантаження сторінок:** Додаток повинен завантажувати головні сторінки і каталоги кросівок швидко, з мінімальним часом очікування для користувачів.
    - **Відгукність і швидкість роботи:** Забезпечення миттєвої відповіді на дії користувача, такі як вибір товарів і перехід до кошика.
2. **Надійність:**
    - **Стійкість і стабільність:** Мінімізація відмов додатка і виправлення помилок без затримок.
    - **Відновлення після збоїв:** Забезпечення можливості відновлення роботи додатка після аварій і відключень зв'язку.
3. **Безпека:**
    - **Захист персональних даних:** Забезпечення безпеки конфіденційних даних користувачів під час операцій платежів і зберігання особистої інформації.
    - **Шифрування даних:** Використання сучасних методів шифрування для захисту передачі даних між пристроями і серверами.
4. **Доступність:**
    - **Сумісність з різними пристроями:** Підтримка різних версій операційних систем Android і різних моделей смартфонів.

- **Доступність для людей з обмеженими можливостями:** Забезпечення можливості навігації і взаємодії з додатком для всіх категорій користувачів.

5. **Ефективність:**
    - **Оптимізація використання ресурсів:** Мінімізація використання батареї і пам'яті, що дозволяє користувачам довше використовувати додаток без зниження продуктивності пристрою.
    - **Ефективність роботи з мережею:** Забезпечення швидкого завантаження зображень і відповідей від серверів, що підвищує загальний комфорт використання додатка.
6. **Юзабіліті:**
    - **Інтуїтивний інтерфейс користувача:** Простий і зрозумілий дизайн інтерфейсу, який дозволяє легко здійснювати покупки і керувати кошиком.
    - **Спрощений процес оформлення замовлення:** Мінімізація кількості кроків для завершення покупки, зручні форми для введення інформації.
7. **Складність:**
    - **Простота установки і оновлення:** Легкий процес установки додатка з Google Play Store і автоматичне оновлення для забезпечення доступу до нових функцій і виправлень помилок.

Ці нефункціональні вимоги допоможуть забезпечити високу якість і задоволення користувачів вашого мобільного додатка "Step-into-Style".

2.4 Ідентифікація типового користувача мобільного додатка "Step-into-Style"

1. **Молоді та модні люди:**
   - Опис: Ці користувачі цінують моду та тренди. Вони активно використовують мобільні додатки для покупки нових речей і слідкують за оновленнями колекцій.
   - Потреби: Швидкий доступ до нових колекцій кросівок, акцій і знижок. Інтуїтивний інтерфейс для швидкого вибору та купівлі товарів.
2. **Спортивні ентузіасти:**
   - Опис: Люди, які активно займаються спортом або ведуть активний спосіб життя. Вони шукають якісне взуття для спорту або активного відпочинку.
   - Потреби: Широкий вибір спортивного взуття, можливість вибору за функціональністю (для бігу, тренувань у залі, прогулянок тощо).
3. **Фешн-цінітелі:**
   - Опис: Користувачі, які стежать за модними тенденціями і високоякісними брендами. Вони шукають елегантне та стильне взуття для різних подій і повсякденного використання.
   - Потреби: Великий вибір модних брендів, зручний пошук за категоріями і можливість зберігати улюблені товари.
4. **Зручність і простота:**
   - Опис: Користувачі, які шукають зручне та практичне взуття на кожен день. Вони цінують простоту оформлення замовлення та швидку доставку.
   - Потреби: Простий процес додавання товарів до кошика, зручна оплата і доставка, можливість повернення або обміну товару.
5. **Технологічно освічені користувачі:**
   - Опис: Люди, які активно використовують технології і шукають інноваційні рішення в мобільних додатках.

- Потреби: Наявність сучасних функцій, таких як розпізнавання обличчя для авторизації, інтеграція з платіжними системами, сповіщення про акції та новинки.

Ці типові користувачі дозволяють легше адаптувати додаток під конкретні потреби різних груп користувачів, забезпечуючи їм приємний та зручний досвід використання.

2.5 Проектування навігаційного графу мобільного додатку

Проектування навігаційного графу мобільного додатку - це процес створення структури та організації різних екранів та функціональних модулів, що складають додаток, для забезпечення зручної навігації користувачів. Основна мета проектування навігаційного графу - створити логічну послідовність переходів між екранами та модулями додатку, щоб користувачі могли легко здійснювати потрібні дії та отримувати необхідну інформацію. Структура навігації та взаємозв'язки між різними екранами (рис. 2.2).

За допомогою навігаційного графа розробник може легко організувати навігацію між різними екранами в додатку. Він дозволяє встановлювати точки входу (destination) для кожного екрану, визначати перехідні анімації, задавати параметри переходів (наприклад, передачу даних), а також встановлювати специфічні правила навігації, такі як повернення до попереднього екрану (back navigation) або виконання визначених дій перед показом нового екрану.

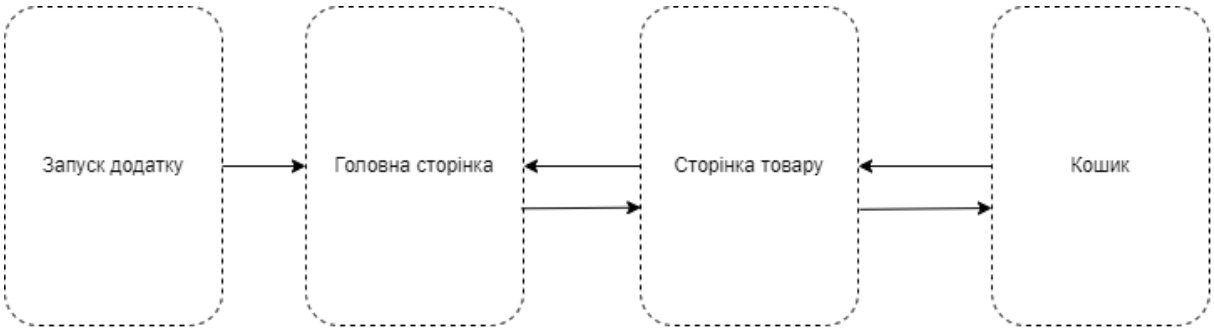


Рисунок 2.2 – Навігаційний граф мобільного додатку

2.6 Проектування користувацького інтерфейсу мобільного додатка

Проектування користувацького інтерфейсу мобільного додатка включає розробку інтерфейсу, який буде зручним і привабливим для користувачів. Ось деякі аспекти:

Основні екрани додатка:

1. Початковий екран:
    - Мета: Після запуску додатка користувачі повинні бачити корисну інформацію та ключові функції додатка.
    - Елементи:
        - Банери або акції, які можуть зацікавити користувача.
        - Навігаційні елементи для переходу до різних частин додатка (головна сторінка, кошик тощо).
        - Можливість швидкого доступу до популярних категорій або нових надходжень.
2. Екран перегляду взуття:

- Мета: Користувачі можуть переглядати різні моделі взуття від різних брендів.
- Елементи:
  - Горизонтальний скролл для швидкого перегляду моделей.
  - Фільтри за кольором, розміром, стилем тощо.
  - Інформація про модель (назва, ціна, рейтинг).
  - Кнопки для додавання в кошик або перегляду деталей товару.

3. Екран деталей товару:
   - Мета: Детальна інформація про обраний товар.
   - Елементи:
     - Фотографії товару з можливістю збільшення.
     - Вибір колірних опцій та розмірів.
     - Відомості про ціну, рейтинг, опис товару.
     - Кнопка для додавання в кошик або продовження покупок.
4. Кошик:
   - Мета: Перегляд товарів, які користувач додав до кошика та оформлення замовлення.
   - Елементи:
     - Список товарів зі змогою зміни кількості або видалення.
     - Відображення підсумкової вартості та можливості редагування.
     - Кнопки для продовження покупок або оформлення замовлення.
     - Інформація про доставку, податки та інші витрати.

Навігація:

- Навігаційне меню: Меню або бокова панель для швидкого доступу до основних розділів додатка (головна сторінка, взуття, акції, кошик, профіль).

- Дизайн:

- Мінімалізм і зручність: Використання простих та чистих дизайнерських рішень для забезпечення зручності і швидкості взаємодії з додатком.
- Кольорова палітра: Використання кольорів, що відповідають бренду та сприяють візуальній привабливості.

Вимоги до проектування:

- Адаптивність: Забезпечення адаптації інтерфейсу під різні типи екранів і розширення.
- Інтуїтивність: Мінімізація кількості кроків для виконання основних функцій, які роблять додаток інтуїтивно зрозумілим і легким у використанні.

2.7 Підключення до Firebase

Підключення до Firebase у мобільних додатках дуже популярне, оскільки Firebase включає в себе різноманітні сервіси, які дозволяють розробникам швидко та зручно реалізовувати різноманітні функціональність, такі як аутентифікація користувачів, зберігання даних в реальному часі, аналіз даних, зберігання файлів та інше. Як підключити:

Крок 1: Створення проекту Firebase

1. Створення облікового запису Firebase:
    - Перейти на веб-сайт Firebase і увійти до свого облікового запису Google.
    - Створити новий проект Firebase, натиснути кнопку "Add project" і вказавши необхідну інформацію про проект.

Крок 2: Додавання додатку до проекту Firebase

2. Додавання вашого додатку до Firebase(рис.2.3-2.4):
    - Після створення проекту Firebase, додати мобільний додаток до цього проекту.
    - Обрати платформу і ввести назву пакету додатку.

Крок 3: Налаштування проекту Android

3. Налаштування даного проекту Android:
    - Завантажити файл databases.json, який містить конфігураційні дані Firebase для додатку.
    - Додати файл databases.json до кореневої папки проекту Android Studio.

Крок 4: Додавання залежностей

4. Додавання залежностей Firebase до проекту:
    - В файлі build.gradle проекту, додаємо репозиторій і залежність.

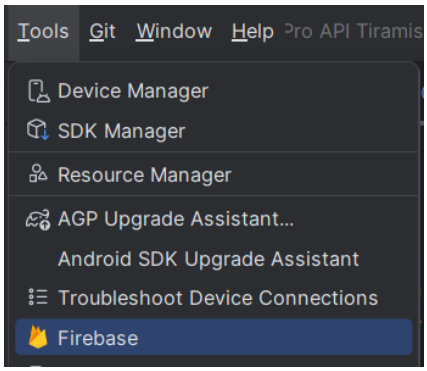


Рисунок 2.3 - Знаходження Firebase

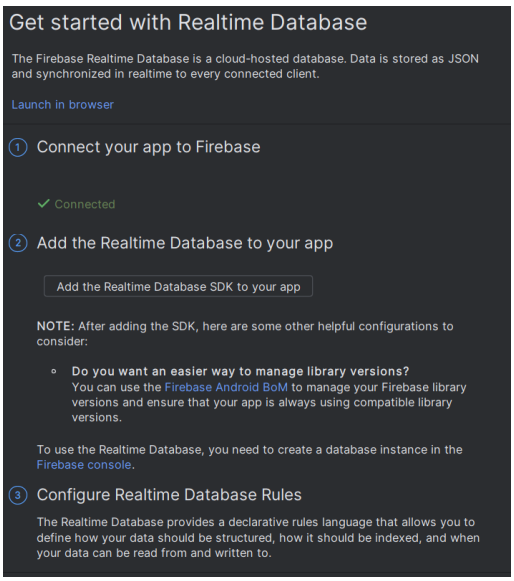


Рисунок 2.4 - Додавання Firebase в Android Studio

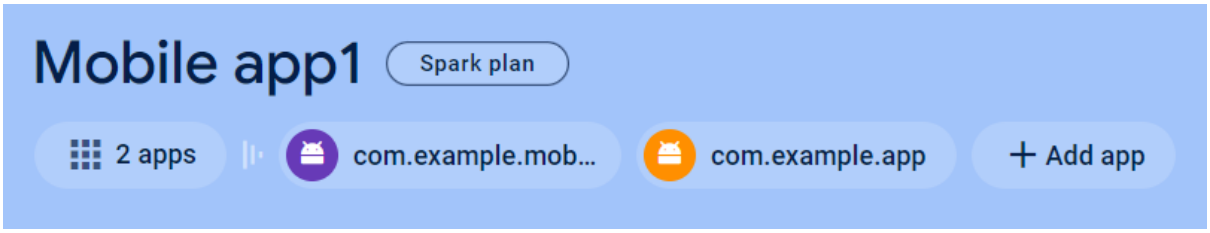


Рисунок 2.5 - Створення проекту в Firebase

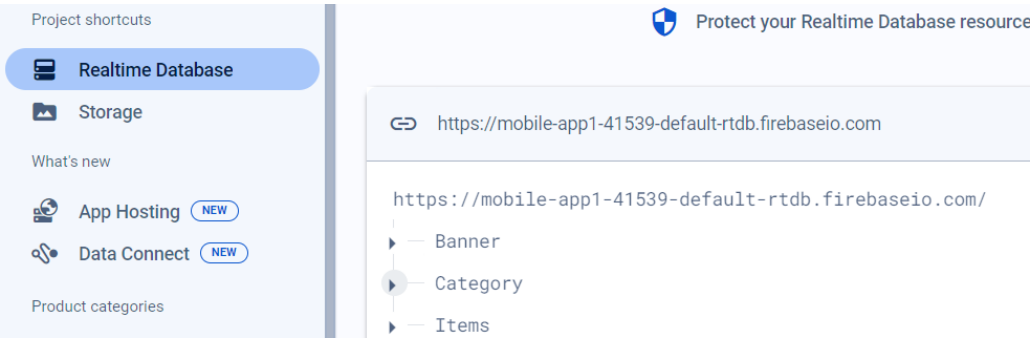


Рисунок 2.6 - Realtime Database

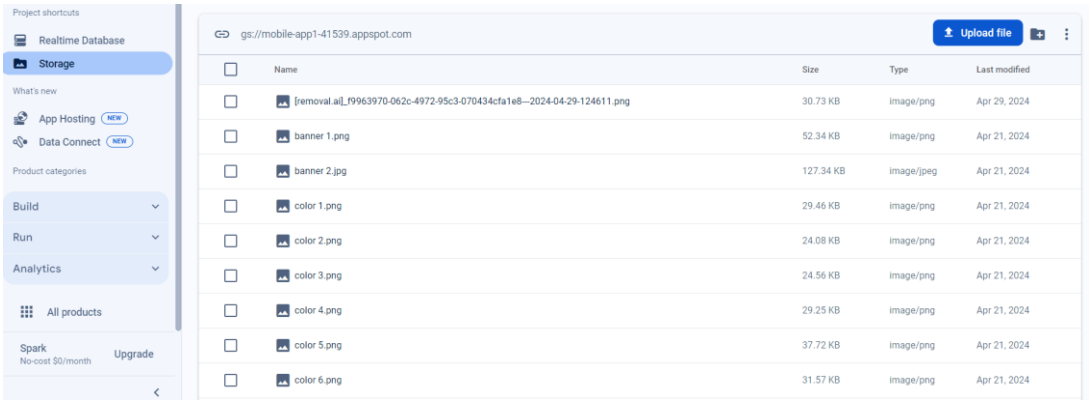


Рисунок 2.7 - Storage(там де знаходяться фото для додатку)

```
58    implementation(libs.firebase.database)
```

Рисунок 2.8 - Додавання Firebase в файл build.gradle

2.8 Висновки до другого розділу

У другому розділі були сформульовані цілі та завдання мобільного додатка. Цілі визначають основні напрямки розробки додатка, а завдання визначають конкретні задачі, які необхідно виконати для досягнення поставлених цілей.
Були сформульовані користувацькі сценарії та історії для ігрового додатка. Користувацькі сценарії описують послідовність дій користувача в додатку, а історії розповідають про конкретні ситуації або завдання, які користувач може зустріти у застосунку.

Нефункціональні вимоги визначають властивості та характеристики додатка, які не відносяться безпосередньо до його функціональності. Вони охоплюють такі аспекти, як продуктивність, безпека, надійність, доступність та інші. Був ідентифікований типовий користувач мобільного додатка. Це людина, яка має інтерес до спорту, яка активно займається спортом або веде активний спосіб життя. Вона шукає якісне взуття для спорту або активного відпочинку.

Ці етапи проектування допомагають визначити напрямок розробки мобільного додатка, описати користувацькі сценарії та нефункціональні вимоги, ідентифікувати цільових користувачів та розробити навігаційну структуру.
Та був додан Firebase - це платформа, яка надає різноманітні сервіси для розробки та підтримки мобільних і веб-додатків. Її можна використовувати для реалізації різних функціональних можливостей, таких як аутентифікація користувачів, зберігання та синхронізація даних, аналітика, розсилки повідомлень, зберігання файлів і багато іншого.

3 Реалізація мобільного додатка в Android Studio

Реалізація мобільного додатка в середовищі Android Studio передбачає кілька ключових етапів, що включають налаштування проекту, розробку інтерфейсу, написання коду на Kotlin, тестування та розгортання. Дії для реалізації мобільного додатка:

Етапи реалізації мобільного додатка в Android Studio:

1. Налаштування проекту:
   1.1Створення нового проекту в Android Studio(обираємо Empty ViewActivity, рис.3.1).
   1.2Вибір мінімальної версії Android SDK, яка підтримується вашим додатком.
   1.3Вибір базової архітектури додатка (наприклад, чиста архітектура, MVVM, MVP).

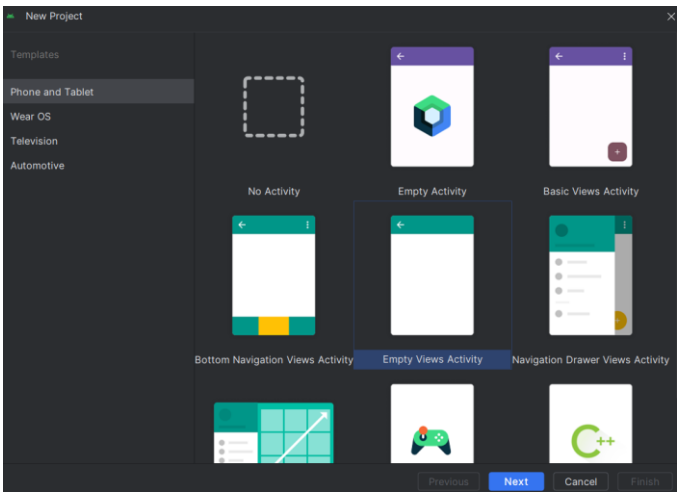


Рисунок 3.1 - Шаблон проекту

2. Дизайн інтерфейсу користувача (UI/UX):

2.1. Розробка макетів інтерфейсу додатка за допомогою XML (для Android) або через редактор макетів в Android Studio.

2.2. Використання різних елементів інтерфейсу, таких як тексти, кнопки, списки, картинки тощо, для створення зручного користувацького досвіду.

3. Написання коду:

- Розробка бізнес-логіки додатка на мові програмування Kotlin або Java.
- Реалізація функціональності, що включає обробку подій і взаємодію з базою даних, веб-сервісами чи іншими джерелами даних.
- Створення різноманітних компонентів, таких як адаптери для списків, діалогові вікна, фрагменти тощо.

4. Тестування:

- Проведення тестування додатка для виявлення і усунення помилок (debugging).
- Виконання тестування різних сценаріїв взаємодії користувача з додатком.

Рекомендації для успішної реалізації:

- **Використовування модульної структури**: розділити додаток на логічні модулі (наприклад, окремі пакети для активностей, фрагментів, адаптерів), що спростить розробку і підтримку.
- **Залучити тестувальників і розробників**: Проводити систематичне тестування та залучати фахівців з інженерії якості для забезпечення якості продукту.
- **Використовувати версійний контроль**: системи контролю версій (наприклад, Git) для керування змінами в коді і спільної роботи над проектом.

3.1 Структура мобільного проекту в AndroidStudio

Загальна мобільного проекту в AndroidStudio:

1. **Папка “activity”:** Це папка, де зберігаються всі активності (класи) вашого додатку. Активності відповідають за різні екрани та взаємодію користувача з інтерфейсом.

- **BaseActivity**: Базовий клас активності, від якого можуть наслідуватися інші активності, щоб уникнути дублювання коду.
- **CartActivity**: Активність, яка відповідає за екран кошика покупок.
- **DetailActivity**: Активність для відображення детальної інформації про певний елемент.
- **IntroActivity**: Активність для стартового екрана (інтро) вашого додатку.
- **MainActivity**: Головна активність вашого додатку, яка може містити основний інтерфейс користувача.

Як вона виглядає продемонстровано на рисунку 3.2.

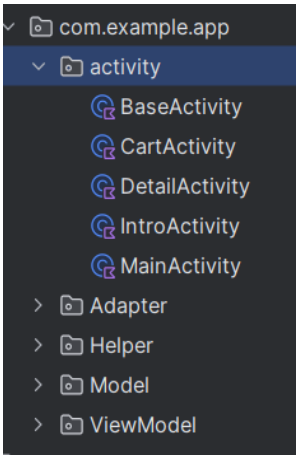


Рисунок 3.2 - Папка **“activity” та її елементи**

2. **Папка”Adapter”:** Ця папка містить класи адаптерів, які використовуються для зв’язку даних з інтерфейсом користувача. Адаптери зазвичай використовуються з компонентами інтерфейсу, такими як

RecyclerView, щоб відображати списки або колекції даних у вигляді елементів інтерфейсу.

- **BrandAdapter :** Цей адаптер відповідає за відображення списку брендів у вашому додатку. Можливо, використовується у випадках, коли потрібно показати логотипи або назви брендів у форматі списку або сітки.
- **CartAdapter :** Адаптер для відображення елементів у кошику. Цей адаптер зв’язує дані про товари, додані до кошика, з відповідними елементами інтерфейсу.
- **ColorAdapter :** Цей адаптер відповідає за відображення списку кольорів. Може використовуватися для вибору кольорів продуктів або для інших кольорових опцій у додатку.
- **PopularAdapter :** Адаптер для відображення популярних товарів або елементів. Використовується для відображення списку популярних продуктів або найбільш популярних елементів на головній сторінці додатку.
- **SizeAdapter :** Адаптер для відображення списку розмірів. Можливо, використовується для вибору розміру продуктів, таких як одяг чи взуття.
- **SliderAdapter :** Адаптер для відображення слайдера зображень або інших елементів. Використовується для створення слайдерів на головній сторінці або на сторінці продукту для відображення декількох зображень чи елементів.

Як вона виглядає продемонстровано на рисунку 3.3.

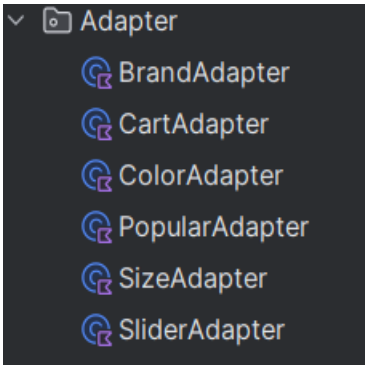


Рисунок 3.3 - Папка”Adapter”зі своїми елементами

3. **Папка”Helper**” :Ця папка містить допоміжні класи та інтерфейси, які надають додаткову функціональність та допомагають в управлінні даними, подіями та іншими аспектами вашого додатку.

- **ChangeNumberItemsListener :** Інтерфейс, що визначає метод для обробки змін у кількості елементів. Використовується для прослуховування змін у кількості товарів у кошику та оновлення відповідної інформації в інтерфейсі.
- **ManagmentCart :** Клас, який управляє кошиком покупок. Він містить методи для додавання, видалення та оновлення елементів у кошику, а також для обчислення загальної суми замовлення.
- **TinyDB :** Клас для роботи з базою даних або для зберігання даних у невеликому форматі. Використовується для зберігання даних про кошик або інші налаштування додатку. Як вона виглядає продемонстровано на рисунку 3.4.

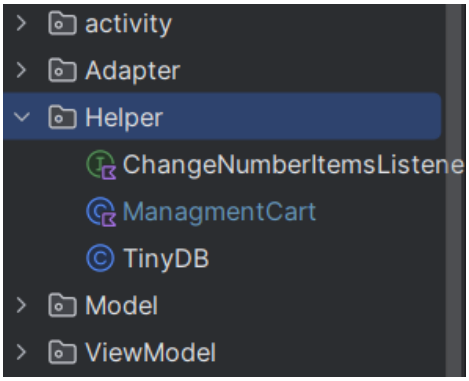


Рисунок 3.4 - Папка”Helper”

4. **Папка”Model”:** Ця папка містить моделі даних, які використовуються у вашому додатку. Моделі даних представляють структуру об'єктів, з якими працює додаток, і включають поля та методи для доступу та маніпуляції цими даними.

- **BrandModel :** Модель, яка представляє інформацію про бренд. Містить поля для зберігання назв, описів, логотипів або інших атрибутів, пов'язаних з брендом.
- **ItemsModel :** Модель, яка представляє інформацію про товари. Містить поля для зберігання назв товарів, описів, зображень, цін, рейтингів та інших атрибутів, пов'язаних з товарами.
- **SliderModel :** Модель, яка представляє інформацію для слайдера (каруселі зображень або контенту). Містить поля для зберігання зображень, текстів або інших атрибутів, які використовуються у слайдері. Як вона виглядає продемонстровано на рисунку 3.5.

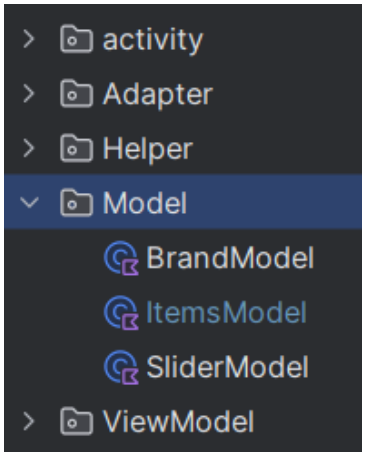


Рисунок 3.5 - Папка”Model”

5. **Папка “ViewModel”:** Ця папка, використовується в розробці Android-додатків для зберігання класів ViewModel, які відповідають за управління бізнес-логікою та управління даними. Як вона виглядає продемонстровано на рисунку 3.6.

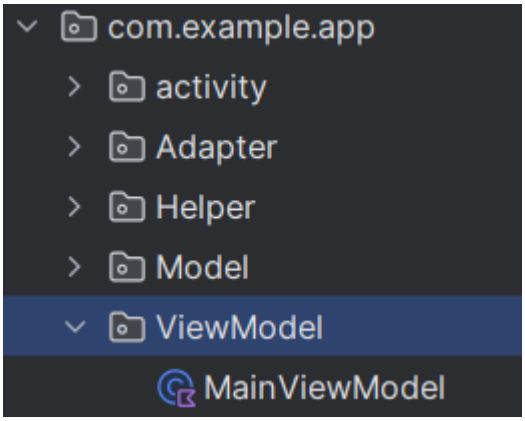


Рисунок 3.6 - Папка “ViewModel”

6. **Папка “drawable”:** Ця папка містить зображення, які присутні в цьому додатку, та кнопки, які там використовуються. Як вона виглядає продемонстровано на рисунку 3.7.

7.

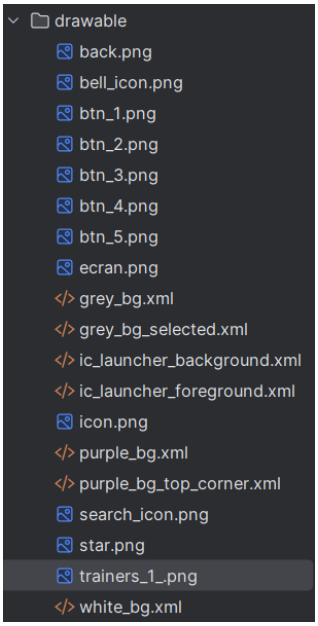


Рисунок 3.7 - Папка “drawable”

8. **Папка “layout”**: Ця папка містить XML-файли, які визначають інтерфейс користувача (UI) для різних екранів і компонентів даного додатка. Як вона виглядає продемонстровано на рисунку 3.8.

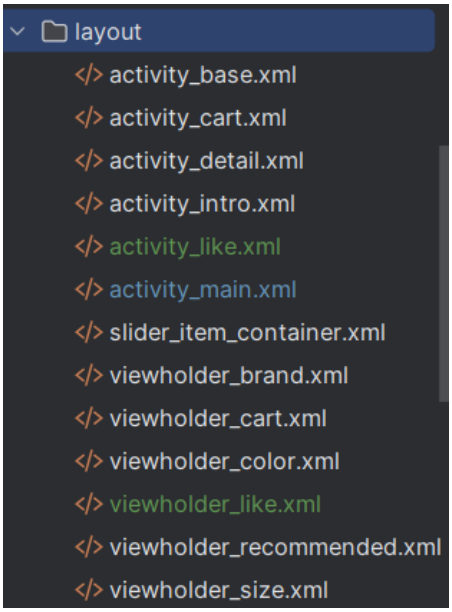


Рисунок 3.8 - Папка “layout”

9. **Папка “values”**: Ця папка є частиною структури ресурсів і містить різні типи значень, такі як рядки, кольори, стилі. Як вона виглядає продемонстровано на рисунку 3.9.

Рисунок 3.9 - Папка “values”

3.2 Діаграма класів мобільного додатка

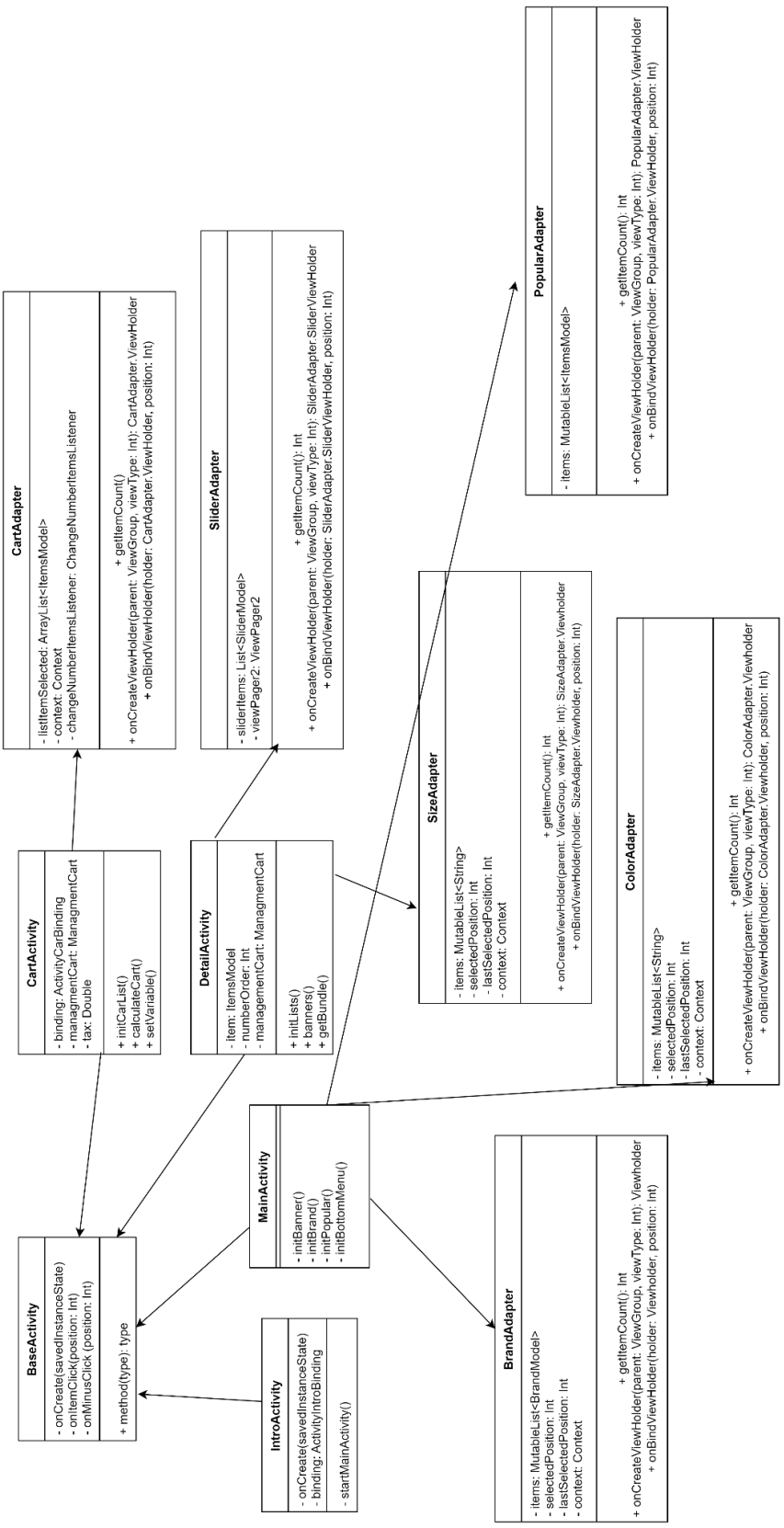


Рисунок 3.10 - Діаграма класів

3.3 Керування вихідним кодом мобільного додатка

У випадку даної розробки для платформи Android слід використати Android Studio, яке є найбільш популярним та рекомендованим інтегрованим середовищем розробки (IDE). Які функції він надає:

- **Підказки (Code Completion)**: Android Studio надає підказки під час введення коду, що дозволяє швидко додавати методи, класи та інші конструкції мови програмування. Це значно зменшує час на написання коду та допомагає уникнути помилок.
- **Автоматичне завершення (Auto-Completion)**: IDE автоматично доповнює назви методів, змінних, ідентифікаторів та інших елементів програмного коду. Це спрощує написання коду і допомагає уникати опечаток.
- **Відлагодження коду (Debugging)**: Android Studio має потужний інструмент відлагодження, який дозволяє запускати програму крок за кроком, аналізувати значення змінних та виправляти помилки. Відлагодження можна проводити як на емуляторі, так і на підключеному пристрої.
- **Підтримка системи контролю версій (Version Control System)**: Android Studio інтегрується з різними системами контролю версій, такими як Git. Вона надає можливості для роботи з комітами, гілками, об'єднаннями та іншими операціями Git, що спрощує спільну роботу над проектами.
- **Розширені інструменти для розробки інтерфейсів (Layout Editor)**: Android Studio має потужний графічний редактор інтерфейсів користувача, який дозволяє швидко створювати і редагувати макети за допомогою перетягування та розміщення елементів.

3.4 Функціональне тестування розробленого мобільного додатка

Код(рис. 3.11), що наведений, описує базовий клас BaseActivity.

```
package com.example.app.activity

import android.os.Bundle
import android.view.WindowManager
import androidx.appcompat.app.AppCompatActivity

Kravchishina *
open class BaseActivity : AppCompatActivity() {
    Kravchishina
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

       window.setFlags(
           WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
           WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
       )
        }

    new *
    open fun onItemClick(position: Int) {}
    new *
    open fun onMinusClick(position: Int) {}
}
```

Рисунок 3.11 - Код для класу BaseActivity

Даний код (рис. 3.12 - 3.13) описує активність CartActivity.

```
package com.example.app.activity
import android.annotation.SuppressLint
import android.os.Bundle
import android.view.View
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.app.Adapter.CartAdapter
import com.example.app.Helper.ChangeNumberItemsListener
import com.example.app.Helper.ManagmentCart
import com.example.app.databinding.ActivityCartBinding
Kravchishina *
class CartActivity : BaseActivity() {
    private lateinit var binding: ActivityCartBinding
    private lateinit var managmentCart: ManagmentCart
    private var tax: Double = 0.0
    Kravchishina *
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityCartBinding.inflate(layoutInflater)
        setContentView(binding.root)
        managmentCart= ManagmentCart( context: this)
        setVariable()
        initCarList()
        calculateCart()
    }
    Kravchishina
    private fun initCarList() {
        binding.viewCart.layoutManager = LinearLayoutManager( context: this, LinearLayoutManager.VERTICAL, reverseLayout: false)
    binding.viewCart.adapter = CartAdapter(managmentCart.getListCart(), context: this,object: ChangeNumberItemsListener{
        override fun onChanged() {
            calculateCart()
```

Рисунок 3.12 - Код для CartActivity

Рисунок 3.13 – Продовження

1. **Пакет та імпорти**:
    - Вказує пакет, в якому знаходиться клас BaseActivity (com.example.app.activity).
    - Імпортує необхідні класи для роботи з Android інтерфейсом користувача, роботи з вікном, підтримки країв екрану тощо.
2. **Оголошення класу BaseActivity**:
    - open class BaseActivity : AppCompatActivity() - визначає клас BaseActivity, який є відкритим для розширення і наслідується від класу AppCompatActivity.
3. **Метод onCreate**:
    - override fun onCreate(savedInstanceState: Bundle?) { ... } - це метод життєвого циклу Android, який викликається при створенні активності.

- super.onCreate(savedInstanceState) - викликає метод базового класу для виконання стандартної логіки створення активності.
- window.setFlags(...) - встановлює певні флаги для вікна активності:
  - WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS - вимикає обмеження на межі вікна активності, що дозволяє елементам інтерфейсу виходити за межі.
  - Цей прапор дозволяє досягти ефекту "Edge-to-Edge", коли елементи розміщуються вздовж всіх країв екрану, ігноруючи стандартні обмеження.

4. **Віртуальні методи onItemClick та onMinusClick**:
   - open fun onItemClick(position: Int) {} - віртуальний метод, який можна перевизначити у підкласах для обробки події кліку на певному елементі інтерфейсу.
   - open fun onMinusClick(position: Int) {} - віртуальний метод, який можна перевизначити для обробки події зменшення кількості елементів (наприклад, в кошику).
5. **Оголошення класу CartActivity**:
   - CartActivity успадковує BaseActivity, що означає, що він використовує функціональність, яка була описана у базовому класі.
6. **Приватні властивості класу**:
   - private lateinit var binding: ActivityCartBinding - зв'язує активність з її макетом ActivityCartBinding, що дозволяє звертатися до елементів інтерфейсу з коду активності.
   - private lateinit var managmentCart: ManagmentCart - ініціалізує екземпляр класу ManagmentCart, який відповідає за управління кошиком товарів.
   - private var tax: Double = 0.0 - змінна для збереження обчисленого податку.

7. **Метод onCreate**:
    - onCreate(savedInstanceState: Bundle?) - перевизначає метод базового класу для налаштування активності при її створенні.
    - binding = ActivityCartBinding.inflate(layoutInflater) - інфлітує макет ActivityCartBinding, який містить усі інтерфейсні елементи активності.
    - setContentView(binding.root) - встановлює макет активності з отриманого binding.
    - managmentCart = ManagmentCart(this) - ініціалізує об'єкт ManagmentCart, передаючи поточний контекст активності.
8. **Метод initCarList**:
    - initCarList() - налаштовує RecyclerView зі списком товарів у кошику.
    - binding.viewCart.layoutManager = LinearLayoutManager(this, LinearLayoutManager.VERTICAL, false) - встановлює LinearLayoutManager для RecyclerView, що дозволяє відображати елементи у вертикальному списку.
    - binding.viewCart.adapter = CartAdapter(managmentCart.getListCart(), this, object: ChangeNumberItemsListener { ... }) - встановлює адаптер CartAdapter для RecyclerView, передаючи список товарів з кошика, поточний контекст активності та слухача ChangeNumberItemsListener, який викликається при зміні кількості товарів у кошику.
9. **Метод calculateCart**:
    - calculateCart() - обчислює загальну суму замовлення, податок та вартість доставки на основі даних з ManagmentCart.
    - Обчислені значення відображаються на відповідних TextView елементах інтерфейсу за допомогою binding.
10. **Метод setVariable**:

- setVariable() - налаштовує клікабельність кнопки "назад" (backBtn), щоб завершити активність при кліку.

Цей код демонструє типовий сценарій роботи з активністю в Android, де BaseActivity надає загальні функції, а CartActivity концентрується на специфічній логіці кошика товарів і взаємодії з інтерфейсом користувача.

Цей код (рис. 3.14 - 3.15) реалізує активність DetailActivity, яка відповідає за відображення деталей певного товару.

```
import android.content.Intent
import android.os.Bundle
import android.view.View
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.app.Adapter.ColorAdapter
import com.example.app.Adapter.SizeAdapter
import com.example.app.Adapter.SliderAdapter
import com.example.app.Helper.ManagmentCart
import com.example.app.Model.ItemsModel
import com.example.app.Model.SliderModel
import com.example.app.databinding.ActivityDetailBinding
Kravchishina *
class DetailActivity : BaseActivity() {
  private lateinit var binding: ActivityDetailBinding
  private lateinit var item: ItemsModel
  private var numberOrder=1
    private lateinit var managementCart: ManagmentCart
    Kravchishina *
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
     binding = ActivityDetailBinding.inflate(layoutInflater)
        setContentView(binding.root)
        managementCart = ManagmentCart( context: this)
        getBundle()
        banners()
        initLists()
    }
```

Рисунок 3.14 - Код для DetailActivity

```
private fun initLists() {
    val sizeList=ArrayList<String>()
    for (size in item.size){
        sizeList.add(size.toString())
    }

    binding.sizeList.adapter=SizeAdapter(sizeList)
    binding.sizeList.layoutManager=LinearLayoutManager( context: this,LinearLayoutManager.HORIZONTAL,  reverseLayout: false)

    val colorList=ArrayList<String>()
    for (imageUrl in item.picUrl){
        colorList.add(imageUrl)
    }
    binding.colorList.adapter=ColorAdapter(colorList)
    binding.colorList.layoutManager=LinearLayoutManager( context: this,LinearLayoutManager.HORIZONTAL, reverseLayout: false)

}
```

Рисунок 3.15 – Продовження

1. **Оголошення класу DetailActivity**:
    - DetailActivity успадковує BaseActivity, що означає, що він використовує функціональність, яка була описана у базовому класі.
2. **Приватні властивості класу**:
    - private lateinit var binding: ActivityDetailBinding - зв'язує активність з її макетом ActivityDetailBinding, що дозволяє звертатися до елементів інтерфейсу з коду активності.
    - private lateinit var item: ItemsModel - зберігає об'єкт ItemsModel, який представляє товар, деталі якого відображаються на активності.
    - private var numberOrder = 1 - змінна для збереження кількості товару, який користувач бажає додати до кошика.
    - private lateinit var managementCart: ManagmentCart - ініціалізує об'єкт ManagmentCart, що відповідає за управління кошиком товарів.
3. **Метод onCreate**:
    - onCreate(savedInstanceState: Bundle?) - перевизначає метод базового класу для налаштування активності при її створенні.
    - binding = ActivityDetailBinding.inflate(layoutInflater) - інфлітує макет ActivityDetailBinding, який містить усі інтерфейсні елементи активності.
    - setContentView(binding.root) - встановлює макет активності з отриманого binding.
    - managementCart = ManagmentCart(this) - ініціалізує об'єкт ManagmentCart, передаючи поточний контекст активності.
4. **Метод initLists**:
    - initLists() - ініціалізує списки розмірів та кольорів товару.
    - sizeList - створює список розмірів товару з використанням SizeAdapter для RecyclerView sizeList.

- colorList - створює список кольорів товару з використанням ColorAdapter для RecyclerView colorList.

5. **Метод banners**:
    - banners() - ініціалізує слайдер (карусель) зображень товару з використанням SliderAdapter.
    - sliderItems - створює список SliderModel для кожного зображення товару.
    - Налаштовує SliderAdapter для binding.slider та встановлює показник точок (dotIndicator), якщо кількість слайдів більше одного.
6. **Метод getBundle**:
    - getBundle() - отримує дані товару, які були передані в активність через Intent.
    - Відображає назву товару (titleTxt), опис (descriptionTxt), ціну (priceTxt), рейтинг (ratingTxt) на відповідних елементах інтерфейсу.
    - Налаштовує кнопки "Додати до кошика" (addToCartBtn), "Назад" (backBtn) і "Перейти до кошика" (cartBtn) для взаємодії з користувачем.

Цей код дозволяє користувачеві переглядати деталі товару, додавати його до кошика та переходити до кошика для перегляду і оформлення замовлення.

Цей код (рис. 3.16) описує активність “IntroActivity”.

```
package com.example.app.activity

import android.content.Intent
import android.os.Bundle
import com.example.app.databinding.ActivityIntroBinding


Kravchishina
class IntroActivity : BaseActivity() {
    private lateinit var binding: ActivityIntroBinding
    Kravchishina
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityIntroBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.button.setOnClickListener { it: View!
            startActivity(Intent( packageContext: this@IntroActivity, MainActivity::class.java))
        }

    }
    }
```

Рисунок 3.16 - Код для IntroActivity

1. **Оголошення класу IntroActivity**:
    - IntroActivity успадковує BaseActivity, що означає, що він використовує функціональність, яка була описана у базовому класі.
2. **Приватна властивість binding**:
    - private lateinit var binding: ActivityIntroBinding - зв'язує активність з її макетом ActivityIntroBinding, що дозволяє звертатися до елементів інтерфейсу з коду активності.
3. **Метод onCreate**:
    - onCreate(savedInstanceState: Bundle?) - перевизначає метод базового класу для налаштування активності при її створенні.
    - binding = ActivityIntroBinding.inflate(layoutInflater) - інфлітує макет ActivityIntroBinding, який містить усі інтерфейсні елементи активності.
    - setContentView(binding.root) - встановлює макет активності з отриманого binding.

4. **Обробник кліків для кнопки button**:
    - binding.button.setOnClickListener { ... } - встановлює обробник кліків для кнопки, яка присутня в макеті ActivityIntroBinding.
    - При кліку на кнопку запускається інший компонент додатка, у цьому випадку MainActivity, за допомогою Intent.

Отже, IntroActivity є стартовою активністю додатка, яка відображає вступну сторінку і має одну кнопку для переходу до головної активності (MainActivity).

Цей код (рис. 3.17 -3.19) описує активність MainActivity.

```
package com.example.app.activity
import android.content.Intent
import android.os.Bundle
import android.view.View
import androidx.lifecycle.Observer
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import androidx.viewpager2.widget.CompositePageTransformer
import androidx.viewpager2.widget.MarginPageTransformer
import com.example.app.Adapter.BrandAdapter
import com.example.app.Adapter.PopularAdapter
import com.example.app.Adapter.SliderAdapter
import com.example.app.Model.SliderModel
import com.example.app.ViewModel.MainViewModel
import com.example.app.databinding.ActivityMainBinding
 Kravchishina
class MainActivity : BaseActivity() {
    private val viewModel = MainViewModel()
    private lateinit var binding: ActivityMainBinding

     Kravchishina
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        initBanner()
        initBrand()
        initPopular()
```

Рисунок 3.17 - Код для MainActivity

```
    initBottomMenu()
}

Kravchishina
private fun initBottomMenu() {
    binding.cartBtn.setOnClickListener{startActivity(Intent( packageContext: this@MainActivity, CartActivity::class.java))}
}

Kravchishina
private fun initBanner() {
    binding.progressBarBanner.visibility = View.VISIBLE
    viewModel.banners.observe( owner: this, Observer { items ->
        banners(items)
        binding.progressBarBanner.visibility = View.GONE
    })
    viewModel.loadBanners()
}

Kravchishina
private fun banners(images: List<SliderModel>) {
    binding.viewpagerSlider.adapter = SliderAdapter(images, binding.viewpagerSlider)
    binding.viewpagerSlider.clipToPadding = false
    binding.viewpagerSlider.clipChildren = false
    binding.viewpagerSlider.offscreenPageLimit = 3
    binding.viewpagerSlider.getChildAt( index: 0).overScrollMode = RecyclerView.OVER_SCROLL_NEVER

    val compositePageTransformer = CompositePageTransformer().apply { this: CompositePageTransformer
        addTransformer(MarginPageTransformer( marginPx: 40))
    }
    binding.viewpagerSlider.setPageTransformer(compositePageTransformer)
```

Рисунок 3.18 – Продовження

```
    if (images.size > 1) {
        binding.run { this: ActivityMainBinding
            dotIndicator.visibility = View.VISIBLE
            dotIndicator.attachTo(binding.viewpagerSlider)
        }
    }
}

Kravchishina
private fun initBrand() {
    binding.progressBarBrand.visibility = View.VISIBLE
    viewModel.brands.observe( owner: this, Observer { it: MutableList<BrandModel>!
        binding.viewBrand.layoutManager = LinearLayoutManager( context: this@MainActivity, LinearLayoutManager.HORIZONTAL, reverseLayout: false)
        binding.viewBrand.adapter = BrandAdapter(it)
        binding.progressBarBrand.visibility = View.GONE
    })
    viewModel.loadBrand()
}

Kravchishina
private fun initPopular() {
    binding.progressBarPopular.visibility = View.VISIBLE
    viewModel.popular.observe( owner: this, Observer { it: MutableList<ItemsModel>!
        binding.viewPopular.layoutManager = GridLayoutManager( context: this@MainActivity,  spanCount: 2)
        binding.viewPopular.adapter =PopularAdapter(it)
        binding.progressBarPopular.visibility = View.GONE
    })
    viewModel.loadPopular()
}
}
```

Рисунок 3.19 – Продовження

1. **Оголошення класу MainActivity**:

- MainActivity успадковує BaseActivity, тому має доступ до функціоналу, що реалізований у базовому класі.

2. **Приватні властивості viewModel і binding**:
    - private val viewModel = MainViewModel() - створюється екземпляр MainViewModel, який використовуватиметься для отримання даних.
    - private lateinit var binding: ActivityMainBinding - зв'язує активність з її макетом ActivityMainBinding.
3. **Метод onCreate**:
    - onCreate(savedInstanceState: Bundle?) - перевизначає метод базового класу для налаштування активності при її створенні.
    - binding = ActivityMainBinding.inflate(layoutInflater) - інфлітує макет ActivityMainBinding, який містить усі інтерфейсні елементи активності.
    - setContentView(binding.root) - встановлює макет активності з отриманого binding.
4. **Методи ініціалізації банерів, брендів і популярних елементів**:
    - **initBanner()**:
        - binding.progressBarBanner.visibility = View.VISIBLE - показує прогрес-бар під час завантаження банерів.
        - viewModel.banners.observe(this, Observer { items -> ... }) - спостерігає за змінами в даних банерів і викликає метод banners(items), який оновлює інтерфейс з отриманими зображеннями.
        - viewModel.loadBanners() - завантажує дані банерів з використанням методу з MainViewModel.
    - **banners(images: List<SliderModel>)**:

        - binding.viewpagerSlider.adapter = SliderAdapter(images, binding.viewpagerSlider) - встановлює адаптер SliderAdapter для ViewPager2, який відображає банери.
        - Налаштовує ViewPager2 і додає до нього трансформацію сторінок для кращого візуального ефекту.
        - Показує індикатор сторінок (dotIndicator), якщо банерів більше одного.
    - **initBrand()**:
        - Аналогічно до initBanner, але для брендів.
        - Встановлює LinearLayoutManager і адаптер BrandAdapter для відображення списку брендів.
    - **initPopular()**:
        - Аналогічно до initBanner, але для популярних товарів.
        - Встановлює GridLayoutManager і адаптер PopularAdapter для відображення сітки популярних елементів.
5. **Метод initBottomMenu()**:
    - Встановлює обробник кліків для кнопки cartBtn, який відкриває активність CartActivity при кліку на кнопку "Кошик".

Отже, MainActivity використовує ViewModel для управління даними інтерфейсу, а також використовує різні адаптери для відображення банерів, брендів і популярних елементів.

3.5 Інструкція користувача мобільним додатком "Step-into-Style"

Огляд додатка

1. **Запуск додатка**

- Знайдіть іконку "Step-into-Style" на домашньому екрані свого Android-пристрою і натисніть на неї, щоб відкрити додаток.

2. **Головний екран**

- Після запуску треба натиснути на кнопку **почати** та потрапите на головний екран додатка "Step-into-Style". Тут ви побачите категорії кросівок, банери(інформація про акції),та бренди кросівок,які там в наявності.

Перегляд товарів

1. **Оберіть категорію**

На головному екрані оберіть кросівки, які ви бажаєте переглянути. Натисніть на відповідний блок для переходу.

2. **Перегляд товарів**

Після переходу ви побачите кросівки, які сподобалися та зможете подивитися, які кольори присутні та розміри, які є в наявності. Та кожен товар буде супроводжуватися зображенням, назвою, ціною та рейтингом.

3. **Детальна інформація**

Натисніть на товар, щоб переглянути детальну інформацію про нього, таку як опис, рейтинг, вибір розмірів та кольорів.

Додавання товарів у кошик

1. **Додавання в кошик**

На сторінці товару натисніть кнопку "Додати в кошик". Вас перенаправить до кошика, де ви зможете переглянути вибрані товари та внести зміни в кількість або видалити товари.

2. **Редагування кошика**

У кошику ви можете змінювати кількість товарів, використовуючи кнопки "Прибавити" та "Відняти". Зміни кількості автоматично оновлюють загальну суму покупки.

3. **Перегляд загальної суми**

Завжди можна переглянути загальну суму покупки, зайшовши в кошик.

Детальна інформація про дії, які відбуваються в цьому додатку, з скріншотами (рис. 3.20 - 3.27).

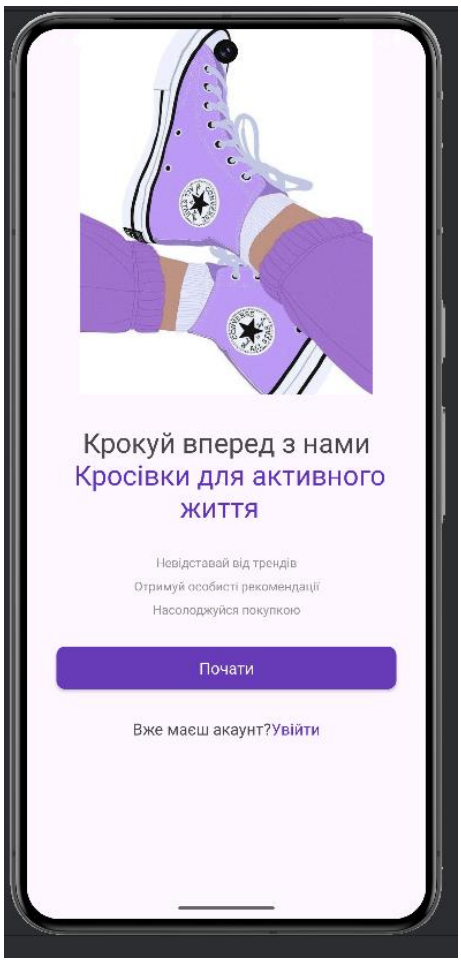


Рисунок 3.20 - Запуск додатку

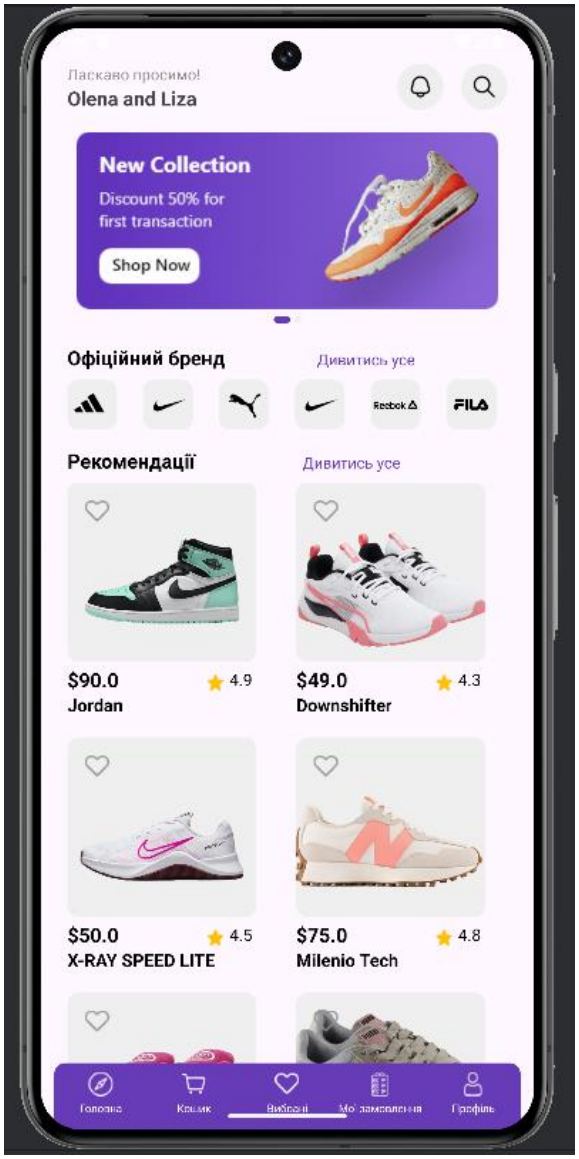


Рисунок 3.21 - Головна сторінка

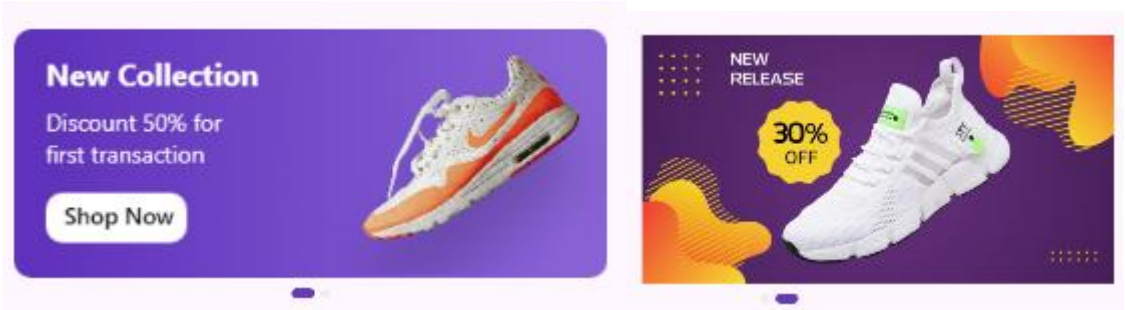


Рисунок 3.22 - Зображені банери

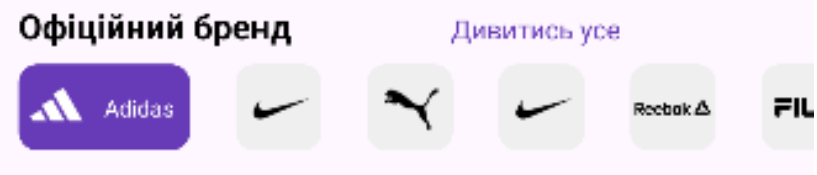


Рисунок 3.23 - Бренди кросівок, які присутні в додатку

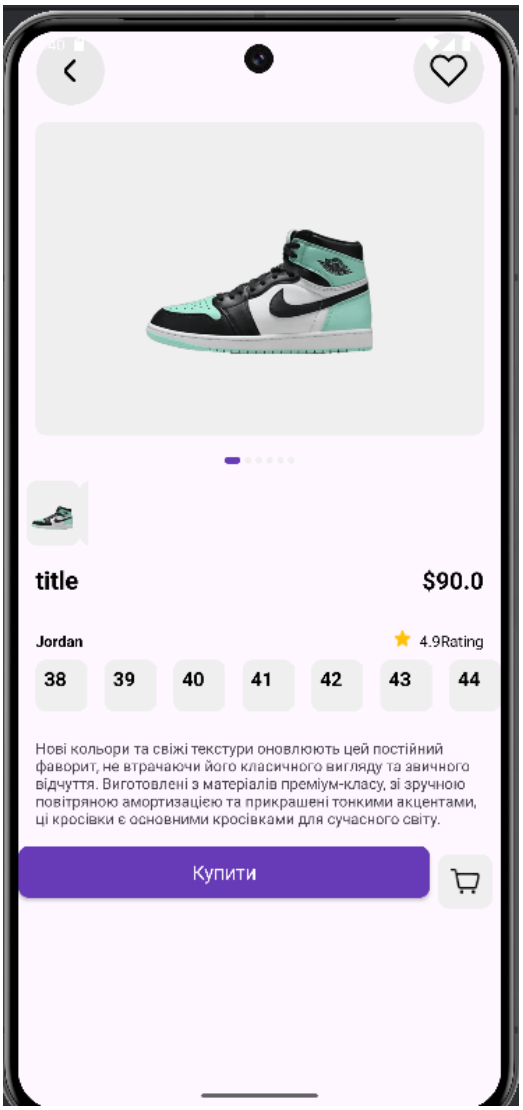


Рисунок 3.24 - Обрані кросівки

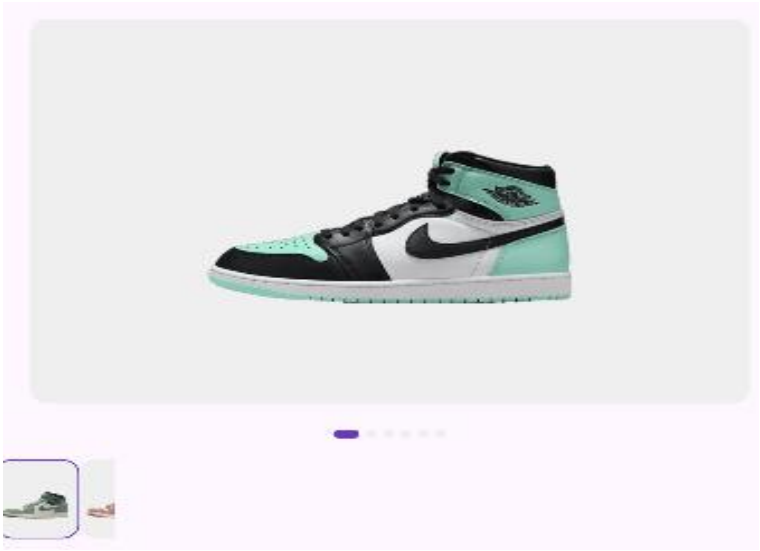


Рисунок 3.25 - Обрання кольору

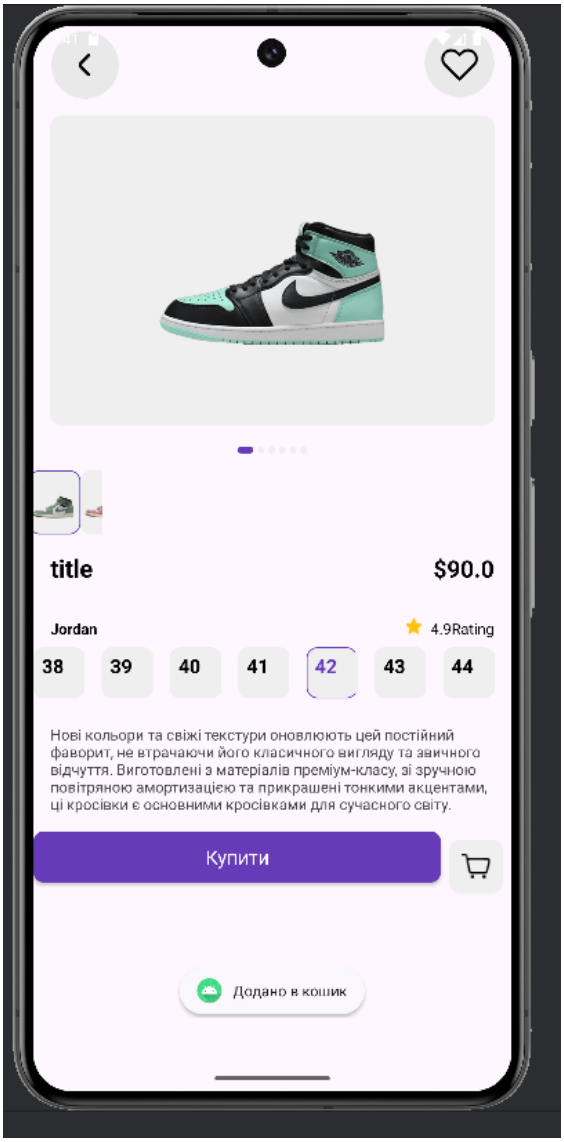


Рисунок 3.26 - Додавання в кошик(натиснути кнопку купити)

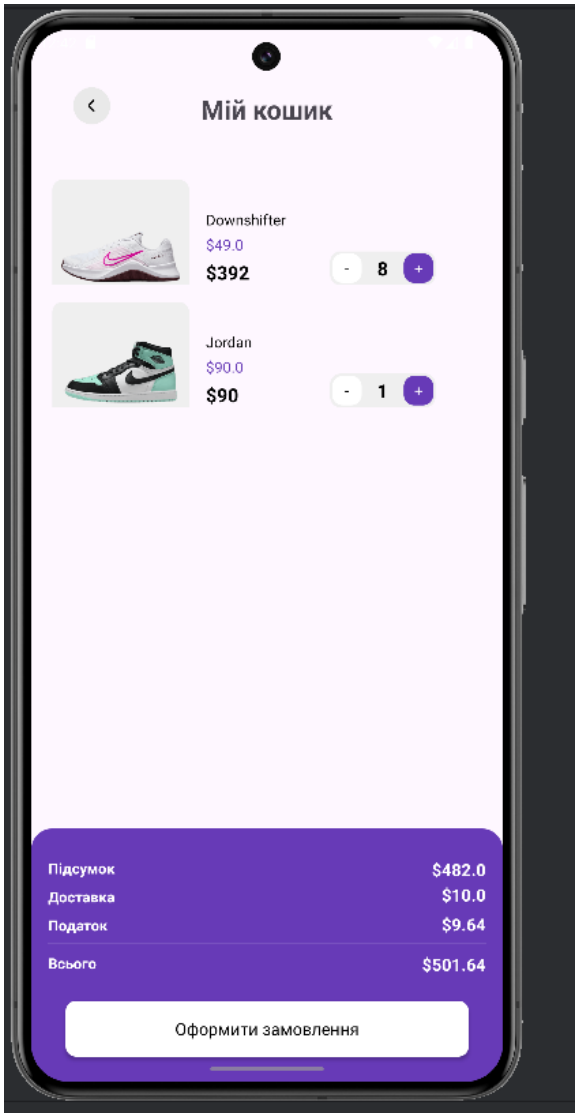


Рисунок 3.27 - Мій кошик

3.6 Висновки до третього розділу

У третьому розділі "Реалізація мобільного додатка в Android Studio" були розглянуті наступні аспекти: 3.1 Структура мобільного проекту в Android Studio: була описана загальна структура проекту,включаючи розташування папок та інших елементів.

3.2 Діаграма класів мобільного додатка: була надана діаграма класів, яка відображає взаємозв'язки між різними класами у додатку. Це допомагає візуалізувати структуру програмного коду та взаємодію об'єктів.

3.3 Керування вихідним кодом даного додатка: було пояснено процес керування вихідним кодом додатка в середовищі Android Studio, включаючи створення класів, редагування коду, компіляцію та налагодження.

3.4 Функціональне тестування розробленого застосунку: була здійснена перевірка додатка на відповідність функціональним вимогам. Це допомагає виявити та виправити можливі помилки, проблеми з функціональністю та забезпечити коректну роботу.

3.5 Інструкція користувача: була надана інструкція користувача, яка описує основні кроки, правила та взаємодію з додатком. Це допомагає користувачам зрозуміти, як ним користуватись.

ВИСНОВКИ

Курсова робота присвячена розробці мобільного додатка "Step-into-Style" на платформі Android з використанням мови програмування Kotlin. Додаток спрямований на полегшення процесу покупки взуття, надаючи користувачам зручні інструменти для перегляду, вибору та покупки відповідно до їхніх потреб.

У першому розділі було розглянуто основні аспекти розробки мобільних додатків, зокрема вибір технологій та інструментів. Основні висновки: **Вибір технологій:** Android Studio обрано як основне інтегроване середовище розробки (IDE) для створення додатків для Android, оскільки воно забезпечує широкі можливості для розробки на Kotlin або Java. **Аналіз сучасних аналогів:** Були проаналізовані сучасні мобільні додатки, що спеціалізуються на продажу взуття, для визначення ключових функціональних інструментів та особливостей, які можна вдосконалити у своєму додатку.

У другому розділі були сформульовані цілі та завдання для мобільного додатка "Step-into-Style". Основні висновки: **Цілі розробки:** Додаток спрямований на створення зручного та естетичного інтерфейсу для користувачів, які цінують комфорт і зручність покупок в інтернет-магазині взуття. **Користувацькі сценарії:** Були розроблені детальні користувацькі сценарії та історії, які описують взаємодію користувачів з додатком у різних ситуаціях.

Третій розділ детально описує реалізацію мобільного додатка "Step-into-Style" в середовищі Android Studio. Основні висновки: **Структура проекту:** Чітка структура проекту сприяє організації коду та полегшує його подальшу підтримку та розвиток. **Діаграма класів:** Діаграма класів допомагає візуалізувати взаємозв'язки між компонентами додатку, що сприяє його легкому розширенню. **Тестування функціональності:** Функціональне тестування дозволило виявити та виправити помилки, що забезпечило коректну роботу додатку.

Курсова робота надала можливість глибше ознайомитися з процесом розробки мобільних додатків для платформи Android з використанням сучасних технологій. Реалізація додатка "Step-into-Style" дозволила застосувати теоретичні знання у практичних цілях, аналізувати та оптимізувати процес розробки, а також навчилася ефективно взаємодіяти з різними інструментами та платформами.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Основи Android studio[Електронний ресурс]: – Режим доступу: https://w3schoolsua.github.io/hyperskill/android-studio-basics.html#gsc.tab=0 (дата звернення 10.04.2024) .

2. Kotlin Підручник[Електронний ресурс]: – Режим доступу: https://w3schoolsua.github.io/kotlin/index.html#google_vignette (дата звернення 09.04.2024) .

3. Документація Firebase Documentation. Firebase[Електронний ресурс]: – Режим доступу: https://firebase.google.com/docs?hl=ru (дата звернення: 18.04.2024).

4. Мобільний додаток[Електронний ресурс]: – Режим доступу: https://wecode.com.ua/ua/blog-ua/mobile-app-krok-za-krokom/ (дата звернення 20.04.2024) .

5. MVVM[Електронний ресурс]: – Режим доступу: https://w3schoolsua.github.io/kotlin/index.html#google_vignette (дата звернення 14.06.2024) .

6. Kotlin і Java[Електронний ресурс]: – Режим доступу: https://foxminded.ua/kotlin-abo-java/ (дата звернення 14.06.2024) .

7. Додаток "Intertop" – URL: https://intertop.ua/uk-ua/ URL GooglePlay: https://play.google.com/store/apps/details?id=ua.mad.intertop&pli=1 URL App Store: https://apps.apple.com/us/app/intertop/id1258680576 (дата звернення 09.04.2024).

8. Додаток "StreatBeet" – URL GooglePlay: https://play.google.com/store/apps/details?id=ru.streetbeat.android&hl=ru URL App Store: https://apps.apple.com/ru/app/street-beat-кроссовки-одежда/id1484704923 (дата звернення 09.04.2024).