МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Мобільний застосунок з автоматичним налаштуванням функцій клієнтської частини»

Рівень «бакалавр»
Галузь знань 12 «Інформаційні технології»,
спеціальність 122 «Комп'ютерні науки»

Виконав здобувач 4 курсу
Сітько Анастасія Володимирівна

Керівник: к.тех.н., доцент
Задерейко Олександр Владиславович

Рецензент: к.тех.н., доцент
Чикунов Павло Олександрович

Одеса – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
Факультет кібербезпеки та інформаційних технологій
Кафедра інформаційних технологій
Галузь знань: 12 Інформаційні технології
Спеціальність: 122 Комп'ютерні науки
Назва освітньої програми: Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій
доцент Н.І. Логінова
_____ «____ »_____ 20__ року

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Сітько Анастасії Володимирівні

1. Тема роботи: «Мобільний застосунок з автоматичним налаштуванням функцій клієнтської частини»
   Керівник роботи: к.тех.н, доцент Задерейко Олександр Владиславович
   затверджені наказом НУ «ОЮА» від «02» квітня 2024 року № 809-06
2. Строк подання здобувачем роботи «20» травня 2024 року
3. Дата видачі завдання «15» вересня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз вимог до ігрового проєкту	15.11.2023 – 25.11.2023	
2	Проєктування програмного продукту	02.12.2023 – 12.12.2023	
3	Реалізація базового функціоналу	13.12.2023 – 13.01.2024	
4	Створення меню налаштувань	14.01.2024 – 18.01.2024	
5	Реалізація мультиплеєрної частини	19.01.2024 – 24.01.2024	
6	Вибір персонажа	25.01.2024 – 04.02.2024	
7	Штучний інтелект ботів	05.02.2024 – 16.02.2024	
8	Реалізація режимів гри та додаткових функцій	17.02.2024 – 26.02.2024	
9	Оформлення пояснювальної записки	01.03.2024 – 17.05.2024	

Здобувач _____
(підпис) (прізвище та ініціали)

Керівник роботи _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сітько А. В. Мобільний застосунок з автоматичним налаштуванням функцій клієнтської частини. – Кваліфікаційна робота бакалавра за спеціальністю 122 “Комп’ютерні науки”, освітньою програмою “Комп’ютерні науки”.

Кваліфікаційна робота викладена на 61 сторінках основного тексту та 67 сторінках загального тексту, містить 3 розділи, 20 рисунків, 7 таблиць, 2 додатки, 7 використаних джерел.

Метою роботи є розробка мобільного застосунка з автоматичним налаштуванням функцій клієнтської частини.

Об’єктом дослідження є мобільний застосунок.

Предмет дослідження – засоби розробки мобільного застосунку з автоматичним налаштуванням функцій клієнтської частини.

У кваліфікаційній роботі розроблено мобільний застосунок з автоматичним налаштуванням функцій клієнтської частини, з можливістю динамічної фільтрації товарів, з системою зберігання строкових значень на серверному боці.

Ключові слова: обробка текстових даних, адаптивні нативні додатки, мобільний застосунок, клієнтська частина додатку.

ANOTATION

Sit'ko A.V. Mobile Application with Automatic Configuration of Client-Side Functions. – Bachelor's Qualification Work in the specialty 122 "Computer Science", educational program "Computer Science".

The qualification work consists of 61 pages of main text and 67 pages of total text, including 3 sections, 20 figures, 7 tables, 2 appendices, and references to 7 sources.

The aim of the work is to develop a mobile application with automatic configuration of client-side functions.

The object of the research is a mobile application.

The subject of the research is the means of developing a mobile application with automatic configuration of client-side functions.

The qualification work includes the development of a mobile application with automatic configuration of client-side functions, with the ability for dynamic filtering of goods and a system for storing expiration dates on the server side.

Keywords: text data processing, adaptive native applications, mobile application, client-side application.

ЗМІСТ

ВСТУП

Під час пікової популярності мобільних пристроїв виробники програмного забезпечення стикаються з необхідністю пошуку оптимальних рішень у сфері мобільних застосунків. Питання вибору між нативними додатками, спеціалізованими під певну платформу, та універсальними рішеннями, які забезпечують сумісність з різними ОС, стає актуальним для розробників. Кожен із цих підходів має свої переваги та обмеження.

На сьогоднішній день, адаптивні нативні додатки ще не досягли повної гнучкості, оскільки вони обмежені можливостями операційних систем. У порівнянні з повноцінними мобільними застосунками, вони не можуть повністю використовувати всі функції платформи. Це призводить до того, що розробники часто віддають перевагу універсальним інструментам, таким як React Native або Flutter, навіть за умови, що це може бути менш ефективно з фінансової точки зору.

У своїй дослідницькій роботі я розглядаю альтернативний підхід до цієї проблеми. Я розглядаю можливість спрощення розробки мобільних застосунків, що призначені для кількох платформ одночасно.

Одним із аспектів моєї роботи є зберігання та обробка текстових даних. У зв'язку з тим, що різні платформи мають різні підходи до цього питання, виникає необхідність у стандартизації цього процесу для забезпечення зручності та ефективності розробки.

Таким чином, моя дипломна робота має на меті розробити мобільний додаток з автоматичним налаштуванням функцій клієнтської частини. Для досягнення цієї мети я планую вивчити та аналізувати існуючі системи з аналогічним функціоналом, вибрати найефективніші технології для реалізації проекту та провести його реалізацію, зосередившись на забезпеченні максимальної зручності для користувачів та ефективності роботи програми.

1 АНАЛІЗ АНАЛОГІВ МОБІЛЬНИХ ЗАСТОСУНКІВ ІНТЕРНЕТ-МАГАЗИНІВ

1.1 Порівняння аналогів мобільних застосунків Інтернет-магазинів

Перед тим, як розпочати роботу над розробкою нового продукту, важливо провести дослідження предметної області і з'ясувати, які вже існують рішення на ринку. Це дозволить краще зрозуміти потреби і вимоги користувачів, а також зробити продукт більш конкурентоспроможним. Визначено кілька кроків, які потрібно виконати під час цього дослідження:

- Проаналізувати існуючі продукти: провести дослідження ринку, щоб знайти комерційні продукти, які вже присутні в цій сфері. Розглянути їх функціональні можливості, особливості і переваги.

- Вивчити відгуки користувачів: переглянути відгуки користувачів про існуючі продукти. Це допоможе зрозуміти їхні потреби, проблеми та очікування.

- Визначити недоліки: виявити, які недоліки існують у вже існуючих продуктах. Це може стати основою для розробки унікальних функцій або покращень у продукті.

- Провести SWOT-аналіз: проаналізувати сильні та слабкі сторони існуючих продуктів, а також можливості та загрози від них. Це допоможе зрозуміти, як можна використати ці аспекти для створення успішного продукту.

Після завершення цих кроків буде визначено потреби користувачів і проаналізовано, як розробляємий продукт може вирізнятися від конкурентів.

В результаті пошуку аналогів було знайдено три інтернет-магазина, які мають мобільні версії.

Інтернет-магазин Rozetka – це платформа, на якій можна придбати товари як від продавця Rozetka, так і від сторонніх магазинів [1].

На даний проміжок часу це один із найпопулярніших інтернет-магазинів в Україні. Він не дарма посідає перші місця з популярності, тому що має широкий асортимент товарів, який починається від продуктів харчування та закінчується

товарами електроніки.

З боку користувачів інтерфейс вважається зручним та легким, що є великою перевагою, так як це збільшує віковий діапазон користувачів, який варіює від підлітків до людей похилого віку, які змогли опанувати смартфони.

Якщо користувач щось шукав у минулі сесії, програма може пропонувати рекомендації на основі вже переглянутих товарів. Це дуже зручно, бо додаток може допомогти користувачу в пошуку.

У програмі досить багато різних розділів, які пропонують актуальні акції та вигідні пропозиції, аналізуючи минулі переглянуті товари.

Створення власного акаунту надає перевагу користувачу: у нього з’являється можливість збирати бонуси, які можна використовувати на наступні покупки.

Prom – це платформа де каталог товарів є дуже зручним, так як ділиться на багато підкатегорій і має картинки біля кожної назви, дозволяючи користувачу швидко дістатися до необхідної речі.

Кожен розділ каталогу ділиться на «Популярні категорії», просто «Категорії» та «Усі категорії» для більш глобального пошуку. Категорії фільтру залежать від обраного товару та добре налаштовані.

Кнопка вибору відображення товарів (плитка чи список) задовольняє будь-які вподобання візуально. Можливість поділитись з друзями у соціальних мережах дозволяє користувачу відправляти посилання прямо з програми, а це дуже зручно.

Розділи з акціями та знижками містять новини та спеціальні пропозиції, які можуть зацікавити користувача. У відео-оглядах можна переглянути більш детально запропоновані нові товари, щоб переконатися у якості та комплектації.

До кошика завжди є доступ з будь-якої сторінки застосування, що є великим плюсом.

У розділі «Списки бажань» можна формувати окремі списки. Це дуже зручно для користувачів, адже вони можуть товари, які обрали, згрупувати для легкого пошуку. Це безсумнівний плюс для додатку.

Ще одна відмінність та зручність цього інтернет-магазину – розділ

«Порівняння». Суть полягає у тому, що якщо користувач не може вибрати між декількох товарів, він може додати їх у порівняння і вже там самостійно подивитися переваги та недоліки кожного товару в порівнянні.

Особистий кабінет дає можливість користувачу заповнити інформацію про себе для подальшої зручності роботи та пошуку. Можна зберегти дані банківської карти для оплати онлайн-покупок. Є можливість повідомити деталі про особисте життя, такі як: наявність домашніх тварин, дітей, автомобіля та вибрати категорії захоплень. Уся ця інформація допомагає додатку у подальшій роботі з користувачем пропонувати йому цікаві та вигідні пропозиції для покупок.

У розділі «Зворотній зв'язок» є телефони для того, щоб зателефонувати в технічну підтримку. Також є три категорії для різних звернень: усі розділи магазину ROZETKA, інші продавці та розробники додатку. Це дуже зручно, так як знижує навантаження на операторів технічної підтримки та дає можливість користувачу швидше отримати необхідну допомогу.

Prom – це платформа для онлайн-шопінгу [2]. Кожен продавець на цьому порталі – це перевірений інтернет-магазин. У асортименті цієї програми безліч товарів з будь-яких категорій.

Контингент користувачів починається від підлітків до людей середнього віку, які часто здійснюють покупки онлайн. Також через простоту інтерфейсу та зручність користування для людей похилого віку не становить проблеми розібратися у програмі.

Коли користувач заходить у програму, то в першу чергу зверху бачить рядок пошуку та кошик. Нижче меню актуальних акцій та пропозицій, які дозволяють одразу дізнатись останні новини. Далі розташовані кнопки швидкого доступу до категорій, знижок, новинок та магазинів. Після цього підряд йдуть популярні на даний момент категорії та товари з них. Знизу екрану розташоване меню, яке включає наступні розділи: топ, категорії, чат, вибране та профіль.

Фільтр пошуку товарів змінюється в залежності від категорій та товарів, але не дозволяє детально виставити усі потрібні характеристики.

Розділ «Вибране» підрозділяється на товари та магазини. Таким чином

користувач може зберігати не тільки товари, які його зацікавили, але й магазини. Щоб зберігати товари та передивлятись замовлення, потрібно обов'язково зареєструватись або авторизуватись, але це не є зручно для деяких користувачів.

AliExpress – це всесвітній інтернет-магазин з асортиментом більш ніж сто мільйонів товарів [3]. Це платформа, на якій різні продавці мають можливість пропонувати свої товари по усьому світу. Переважно усі товари їдуть з Китаю, тому це великий недолік для покупців з України, тому що часто їм доводиться чекати свої закази від одного до трьох місяців.

AliExpress пропонує програму захисту для користувачів, тому якщо з доставкою виникли проблеми і покупець не отримав товар, він має можливість відкрити суперечку і повернути свої гроші.

Інтерфейс вважається доволі важким для користування, тому це звужує контингент користувачів. Переважно це підлітки та молодь. Більш дорослі люди досить часто потребують допомоги, тому що вони не можуть самотужки розібратися в програмі. Навіть для досвідчених користувачів призначення багатьох кнопок залишається невідомим.

Головна сторінка має багато різних акцій, спеціальних та «гарячих» пропозицій, але переважна кількість користувачів цим не користується через складність розуміння умов. У розділі категорії багато різних підкатегорій товарів для швидкого та зручного пошуку.

Фільтр досить непогано налаштований, має багато різних деталей, але для багатьох користувачів залишається занадто важким та незрозумілим. Він змінюється в залежності від товару та категорії та не завжди задовольняє потреби пошуку.

За допомогою розділу «Повідомлення» користувач має змогу спілкуватись безпосередньо з продавцями, у яких він зробив покупку. Також тут відображаються різні сповіщення від програми.

У розділі «Кошик» зберігаються усі товари, які додав користувач. Плюсом є те, що біля кожного товару треба поставити галочку та тільки після цього переходити до оплати. Це дає змогу не видаляти товари з цього розділу, якщо

користувач бажає купити частину з них пізніше. Також за допомогою цієї галочки товари можна видаляти з корзини.

У розділі «Мій профіль» є частина, яка включає в себе такі підрозділи: очікується оплата, очікується відправка, відправлено, очікується відгук, суперечки за замовленнями. Усе це допомагає користувачу зручно відстежувати свої покупки та своєчасно відкривати суперечки, якщо замовлення не було доставлено.

Порівняння Інтернет-магазинів наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння аналогів мобільних Інтернет-магазинів

	Rozetka	Prom	AliExpress	Розроблювальний продукт
Можливість користуватися продуктом в усіх країнах світу	-	-	+	+
Можливість відфільтрувати всі товари за категоріям використання	+	+	+	+
Можливість оновити програму, не заходячи до плей-маркету.	-	-	-	+
Можливість додати товари до обраного	+	+	+	+
Можливість додати товари до кошика	+	+	+	+
Можливість придбати товари інших магазинів	+	+	+	-
User-friendly інтерфейс	+	+	-	+
Функція автоматичного налаштування серверною частиною параметрів фільтра товарів	-	-	-	+
Текстові значення зберігаються на сервері	-	-	-	+
Результат	5	5	5	8

За результатом порівняння аналогів інтернет-магазинів можна зробити висновок, що розроблювальний продукт перевершує аналоги.

1.2 Аналіз технологій

1.2.1 Аналіз мов програмування для розробки застосувань для ОС Android

Kotlin – це статично типізована мова програмування, розроблена компанією JetBrains. Подібно до мови Java, Kotlin став відмінним вибором для розробки додатків на Android. Це можна побачити навіть з того факту, що Android Studio поставляється як з вбудованою підтримкою Kotlin, так і з підтримкою Java [4]. Різниця між цими двома мовами програмування:

– перевірка виключень. Одне з основних відмінностей між Java і Kotlin полягає в тому, що в останньому немає умов для перевірки винятків (checked exception). Отже, немає необхідності відловлювати або оголошувати будь-які винятки;

– стислість коду. Порівняння класу Java з еквівалентним класом Kotlin демонструє лаконічність коду Kotlin. Для тієї ж операції, що виконується в класі Java, клас Kotlin вимагає менше коду;

– співпрограми. Процеси інтенсивно завантажують процесор і мережевий вхід-видобуток, зазвичай використовують тривалі операції. Зухвалий потік блокується до завершення всієї операції. Оскільки Android є однопоточним за замовчуванням, призначений для користувача інтерфейс програми повністю блокується, як тільки блокується основний потік. Традиційне рішення цієї проблеми в Java – створити фоновий потік для тривалої або інтенсивної роботи. Однак управління декількома потоками призводить до збільшення складності, а також до помилок в коді. Kotlin також дозволяє створювати додаткові потоки. Тим не менш, є кращий спосіб управління інтенсивними операціями в Kotlin, відомий як співпрограми або корутини (coroutines). Корутини в Kotlin реалізовані без стека, що означає, що вони вимагають меншого використання пам'яті в порівнянні зі звичайними потоками;

– класи даних. У повнорозмірних проектах зазвичай є кілька класів, призначених виключно для зберігання даних. Хоча ці класи практично не мають функціональності, розробнику необхідно написати багато стандартного коду на Java. Зазвичай розробник повинен визначити конструктор і кілька полів для зберігання даних, функції геттери і сеттери для кожного з полів, а також функції equals(), hashCode() і toString(). У Kotlin є дуже простий спосіб створення таких класів. Розробнику досить включити тільки ключове слово data в визначення класу, і все – компілятор сам подбає про все;

– неявні розширювальні перетворення. У Котлін немає підтримки неявних розширювальних перетворень для даних. Таким чином, менші типи не можуть бути перетворені у великі типи. У той час як Java підтримує неявні перетворення, Kotlin вимагає виконати саме явне перетворення;

– вбудовані функції. Змінні, до яких здійснюється доступ в тілі функції, називаються замиканнями. Використання функцій вищого порядку може істотно збільшити час виконання обчислень. Кожна функція в Kotlin є об'єктом, і він захоплює замикання. На відміну від Kotlin, Java не забезпечує підтримку вбудованих функцій. Проте, компілятор Java здатний виконувати вбудовування з використанням методу final. Це так, тому що методи final не можуть бути перевизначені підкласами. Крім того, виклик методу final дозволяється під час компіляції;

– вбудована підтримка делегування. У термінології програмування, Делегування є процес, в якому об'єкт делегує свої операції другому об'єкту делегата. Kotlin підтримує шаблон проектування композиція поверх спадкування за допомогою делегування першого класу, також відомий як неявне делегування. Делегування в Kotlin є альтернативою спадкування. Цей механізм дозволяє використовувати множинне спадкування. Крім того, делеговані властивості Kotlin запобігають дублювання коду;

– примітивні типи. Існує 8 примітивних типів даних, включаючи char, double, float і int. На відміну від Kotlin, змінні примітивного типу не є об'єктами в Java. Це означає, що вони не є об'єктами, створеними з класу або структури;

–розумні приведення. Перш ніж об'єкт може бути приведений в Java, обов'язково потрібно перевірити тип. Це також вірно в сценаріях, де очевидно потрібно приводити об'єкт. На відміну від Java, Kotlin має функцію розумного приведення, яка автоматично обробляє такі надлишкові приведення;

–статичні члени. У Kotlin немає статичних елементів. Однак в мові програмування Java ключове слово static відображає те, що конкретний член, з яким використовується це ключове слово, належить самому типу, а не екземпляру цього типу. Це просто означає, що один і тільки один екземпляр цього статичного члена створюється і використовується всіма екземплярами класу;

–трійчастий оператор. На відміну від Kotlin, Java має тернарний оператор. Тернарний оператор в Java працює як базовий оператор if. Він складається з умови, яке оцінюється як істинне або помилкове. Синтаксис для тернарного оператора Java – (Стан) ? (Значення 1) : (значення 2);

–бібліотеки обробки анотацій. Крім надання підтримки існуючим Java-фреймворки і бібліотекам, Kotlin також пропонує розширені Java-фреймворки, засновані на обробці анотацій. Однак застосування в Kotlin бібліотеки Java, яка використовує обробку анотацій, вимагає додавання її в проект Kotlin трохи іншим способом, ніж потрібно для бібліотеки Java, яка не використовує обробку анотацій.

Порівняння мов програмування наведено в таблиці 1.2.

Таблиця 1.2 – Порівняння мов програмування

Характеристика	Java	Kotlin
Перевірка виключень	+	+
Стислість коду	-	+
Неявні розширювальні перетворення	+	-
Вбудовані функції	+	+
Вбудована підтримка делегування	-	+
Примітивні типи	+	+
Розумні приведення	-	+
Статичні члени	+	+
Трійчастий оператор	+	-
Сума	6	7

За результатом порівняння мов програмування було зроблено висновок, щао найкращим вибором мови програмування для мобільного додатку являється Kotlin.

1.2.2 Аналіз мов програмування для розробки Server

З точки зору розробника, мова програмування – це інструмент. А правильний вибір інструменту сильно впливає на кінцевий результат. Щоб вибрати кращий варіант, проведемо порівняння топ 4 мови для розробки за версією StackOverflow [5]:

– JavaScript. Зараз JavaScript використовується скрізь. JavaScript-фреймворки (Angular, React і Vue) використовуються на стороні клієнта для розробки веб-додатків на основі браузера. За межами браузера за допомогою Node.js можна писати серверні додатки на тій же мові, на якому ви пишете клієнтський код. За допомогою Node можна створювати веб-сервіси, управляти Інтернет речами (IoT) і експериментувати з машинним навчанням. Важлива перевага JavaScript перед іншими мовами – широка підтримка таких IT-корпорацій, як Google, Facebook, Microsoft і Amazon. Другий плюс – дуже легко знайти навчальні матеріали по JavaScript: існує безліч платних і безкоштовних курсів, веб-сайтів, книг, відео і тематичних блогів;

– Python. Одна з причин популярності Python – в ньому правила оформлення коду простіші, ніж в інших мовах: наприклад, не потрібно ставити крапку з комою в кінці оператора. Тому Python все частіше вивчають в навчальних закладах – не тільки в університетах, а й у середній і початковій школі. Python використовується в академічному середовищі. Це найпопулярніша мова загального призначення, вона використовується для машинного навчання і в науці про дані. Python настільки активно використовується в цих областях, що нещодавно було запропоновано злиття Python і R, мови науки про дані. Python одночасно і схожий на інші мови програмування, і сильно відрізняється від них;

–Java. Одна з переваг Java – це віртуальна машина JVM. JVM дозволяє запустити будь-яку мову на будь-якій апаратній платформі або пристрої. Java також була розроблена для вирішення завдань, пов'язаних з типами даних і управлінням пам'яттю – ця мова спрощила життя розробників. Java спрощує розробку і впровадження програм на різних операційних системах: тому великі компанії частіше використовують Java. Це мова, завдяки якій можна потрапити в штат великої компанії. JVM використовується все активніше: створюються нові мови, адаптуються вже існуючі. Екосистема Java активно використовується. Scala, Closure і Kotlin популярні в окремих сферах;

–C #. Java і раніше краще, ніж C #, проте в майбутньому ситуація зміниться. Java повільніше вводить новий функціонал, в той час як Microsoft агресивно розвиває і додає нові можливості в C #. Таким чином компанія прагне перестати залежати від Windows. З цієї ж причини Microsoft купила Xamarin і її кроссплатформенную середу розробки, випустила багатоплатформне ядро .NET і продовжує інвестувати в Azure. Microsoft бачить, що C # грає ключову роль в її новій стратегії розвитку, і докладає чимало зусиль, щоб C # можна було використовувати як з технологіями від Microsoft, так і з відкритими промисловими стандартами.

1.3 Вибір технологій для розробки системи

Продукт буде мати клієнт-серверну архітектуру з підключенням бази даних. Для реалізації клієнтської частини Android, для якої була обрана мова програмування –Kotlin. Для написання серверу був обраний фреймворк – Node JS.

Android:

–Architecture Components – це набір бібліотек, розроблений компанією Google для полегшення роботи з операційною системою Android;

–LifecycleOwner – інтерфейс для компонентів, у якого є свій життєвий цикл (як у Activity і Fragment);

– LifecycleObserver – підписується на зміни життєвого циклу LifecycleOwner'a, який самодостатній. Він не покладається на методи onStart і onStop Activity або Fragment при ініціалізації і зупинці. LiveData – це спостережуваний об'єкт для зберігання даних, він повідомляє спостерігачів, коли дані змінюються. Цей компонент також є пов'язаною з життєвим циклом LifecycleOwner (Activity або Fragment), що допомагає уникнути витоків пам'яті та інших неприємностей;

– ViewModel – це об'єкти, які надають дані для UI компонентів. Вони не мають посилань на view і незалежні від життєвого циклу LifecycleOwner'a;

– Room – заснована на SQLite бібліотека ORM. Room є абстракцією над шаром, в якому виробляються конкретні дії над базою даних, такими як вставки, видалення, створення таблиць тощо. Room використовує анотації для генерації коду під час компіляції. Більш того, Room також підтримує LiveData і RxJava2 Flowable;

– Retrofit2 – бібліотека для парсингу даних з сервера. Дозволяє зробити повноцінний REST-клієнт, який може виконувати POST, GET, PUT, DELETE запроси. Для позначення типу та інших аспектів запиту використовуються анотації;

– Glide – бібліотека для роботи з зображеннями. Це аналог іншої відомої бібліотеки для зображень - Picasso, і на даний момент є однією з найбільш популярних бібліотек для роботи з зображеннями, оскільки реалізує асинхронне завантаження зображень, їх обробку і кешування. Крім того, на відміну від Picasso, Glide вміє працювати з GIF-зображеннями і відео;

– RxJava – це реалізація бібліотеки ReactiveX з відкритим вихідним кодом, яка допомагає створювати додатки в стилі реактивного програмування;

– RecyclerView – це компонент інтерфейсу користувача, який дозволяє створювати прокручуваний список. Він був представлений в Android Lollipop, і це свого роду ListView2. Він змушує нас використовувати більш ефективний спосіб реалізації списку і розділяє зони відповідальності між класами;

– Архітектура MVVM. MVVM (Model-View-ViewModel) – це шаблон проектування, що застосовується під час проектування архітектури застосунків.

MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд логіки), відомої як модель (можна також сказати, що це відокремлення представлення від моделі). Модель представлення є частиною, яка відповідає за перетворення даних для їх подальшої підтримки і використання. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і оброблює більшість, якщо не всю, логіку відображення даних. Модель представлення може також реалізовувати патерн медіатор, організовуючи доступ до бек-енд логіки навколо множини правил використання, які підтримуються представленням [6].

Spring Boot:

– Spring Boot - це фреймворк для розробки програмного забезпечення на мові програмування Java. Він дозволяє швидко створювати самостійні, готові до використання додатки, які можуть запускатися незалежно від зовнішніх серверів додатків. Spring Boot автоматизує конфігурацію та надає вбудовані рішення для стандартних задач, що дозволяє розробникам зосередитися на бізнес-логіці своїх додатків.

– ORM – це технологія програмування, яка дозволяє перетворювати несумісні типи моделей в ООП, зокрема, між сховищем даних і об'єктами програмування. ORM використовується для спрощення процесу збереження об'єктів в реляційну базу даних і їх вилучення, при цьому ORM сама піклується про перетворення даних між двома несумісними станами. Більшість ORM-інструментів значною мірою покладаються на метадані бази даних і об'єктів, так що об'єктам нічого не потрібно знати про структуру бази даних, а базі даних - нічого про те, як дані організовані в додатку. ORM забезпечує повне розділення завдань в добре спроектованих додатках, при якому і база даних, і додаток можуть працювати з даними кожен у своїй вихідній формі.

Data Base:

PostgreSQL – це не просто реляційна, а об'єктно-реляційна СУБД. Це дає йому деякі переваги над іншими SQL базами даних з відкритим вихідним кодом, такими як MySQL, MariaDB і Firebird. Фундаментальна характеристика об'єктно-

реляційної бази даних – це підтримка об'єктів і їх поведінки, включаючи типи даних, функції, операції, домени і індекси. Це робить Postgres неймовірно гнучким і надійним. Серед іншого, він вміє створювати, зберігати та видавати складні структури даних.

1.4 Висновки до першого розділу

Після порівняння трьох провідних інтернет-магазинів встановлено, що Rozetka відзначається найбільшою зручністю для користувачів, має зрозумілий інтерфейс та функціонал, що відповідає потребам різних вікових груп. Prom.ua відстає за функціоналом та має обмежений зручний інтерфейс, тоді як AliExpress характеризується складним інтерфейсом та наявністю багатьох неперекладених функцій, що зменшує його привабливість для користувачів. Отже, як приклад функціоналу обрано інтернет-магазин Rozetka.

Порівняння мов програмування для платформи Android вказує на те, що деякі функції краще реалізовані в Kotlin, а інші - у Java. Kotlin набуває популярності серед розробників Android додатків, що робить його привабливим вибором для даного проекту, незважаючи на повну сумісність обох мов.

Під час порівняння серверних мов програмування Java (з використанням фреймворку Spring) та JavaScript (з фреймворком Node.js) були визначені їх переваги. Node.js відзначається легкістю та простотою використання, тоді як Java, зокрема з використанням Spring, є кращим вибором для реалізації складних проектів підприємства. В результаті був обраний фреймворк Spring Boot, який оптимально відповідає потребам даного проекту. Для кожної мови програмування складено список технологій, які сприятимуть полегшенню процесу розробки застосування.

2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ІНТЕРНЕТ-МАГАЗИНУ

2.1 Мета та задачі інформаційної системи

Інформаційна система (ІС), яка розроблюється, – це клієнт-серверне застосування, основною функцією якого є реалізація технологій, які дозволять автоматизувати взаємодію між клієнтською та серверною частинами.

Ця система буде реалізована на прикладі мобільного інтернет-магазину, основною функцією котрого буде надання споживачу можливості придбати товар через мережу Інтернет. Тому ми будемо розглядати проектування клієнтської частини інформаційної системи, тобто мобільний додаток.

Мета ІС – серверна сторона інформаційної система повинна за допомогою спеціальних алгоритмів динамічно обробляти дані, що в свою чергу вирішить такі проблеми:

– написання великої кількості програмного коду при внесенні змін до структури даних;

– позбавлення від помилок, створених розробником;

– велика ціна на розробку програм.

Діаграма інформаційних потоків наведена на рисунку 2.1.

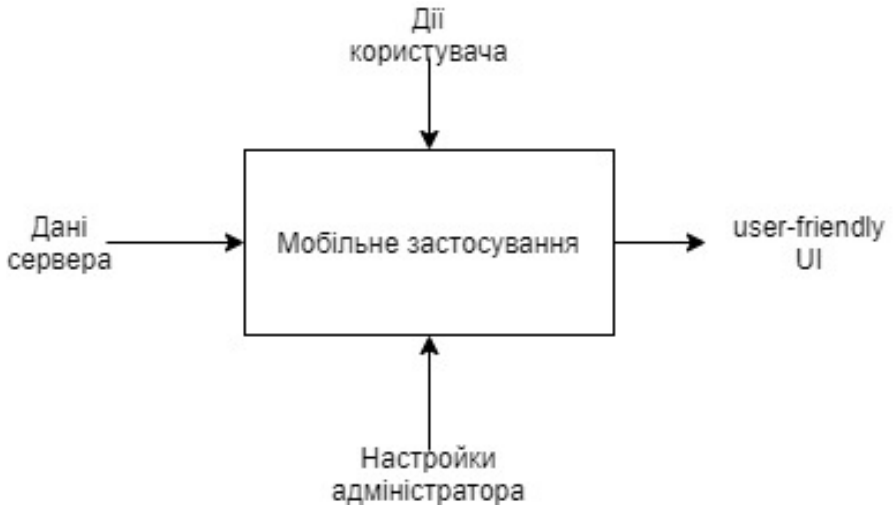

Рисунок 2.1 – Інформаційні потоки ІС, яка розроблюється

Цільова аудиторія: люди, які хочуть придбати товар з доставкою через мережу Інтернет. Завдяки інформативній системі користувачі матимуть змогу зробити замовлення необхідних товарів онлайн, обравши потрібний їм спосіб доставки.

Вхідними даними для системи, яка розроблюється, є інформація про товари, які знаходяться в базі даних.

Вихідними даними є інтерфейс користувача, завдяки якому, він може створити покупку.

2.2 Типи користувачів

В інформаційній системі, яка розроблюється, існує тільки два типу користувачів:

– користувач. Має право на перегляд інформації, що пов’язана з товарами, та право редагувати інформацію, яка пов’язана з настройкою облікового запису. Реєстрація проходить автоматично для кожного пристрою, тому незареєстрованих користувачів не буде існувати;

– адміністратор. Має право додавати, редагувати та видаляти пов’язану з товарами інформацію з бази даних.

Незареєстрованого користувача не існує, тому що застосування реєструє користувача одразу при першому запуску за MAC-адресою пристрою, з якого відбувся вхід. При кожному запиті на сервер, застосування надсилає дані про пристрій на сервер.

Діаграма прецедентів наведена на рисунку 2.2.

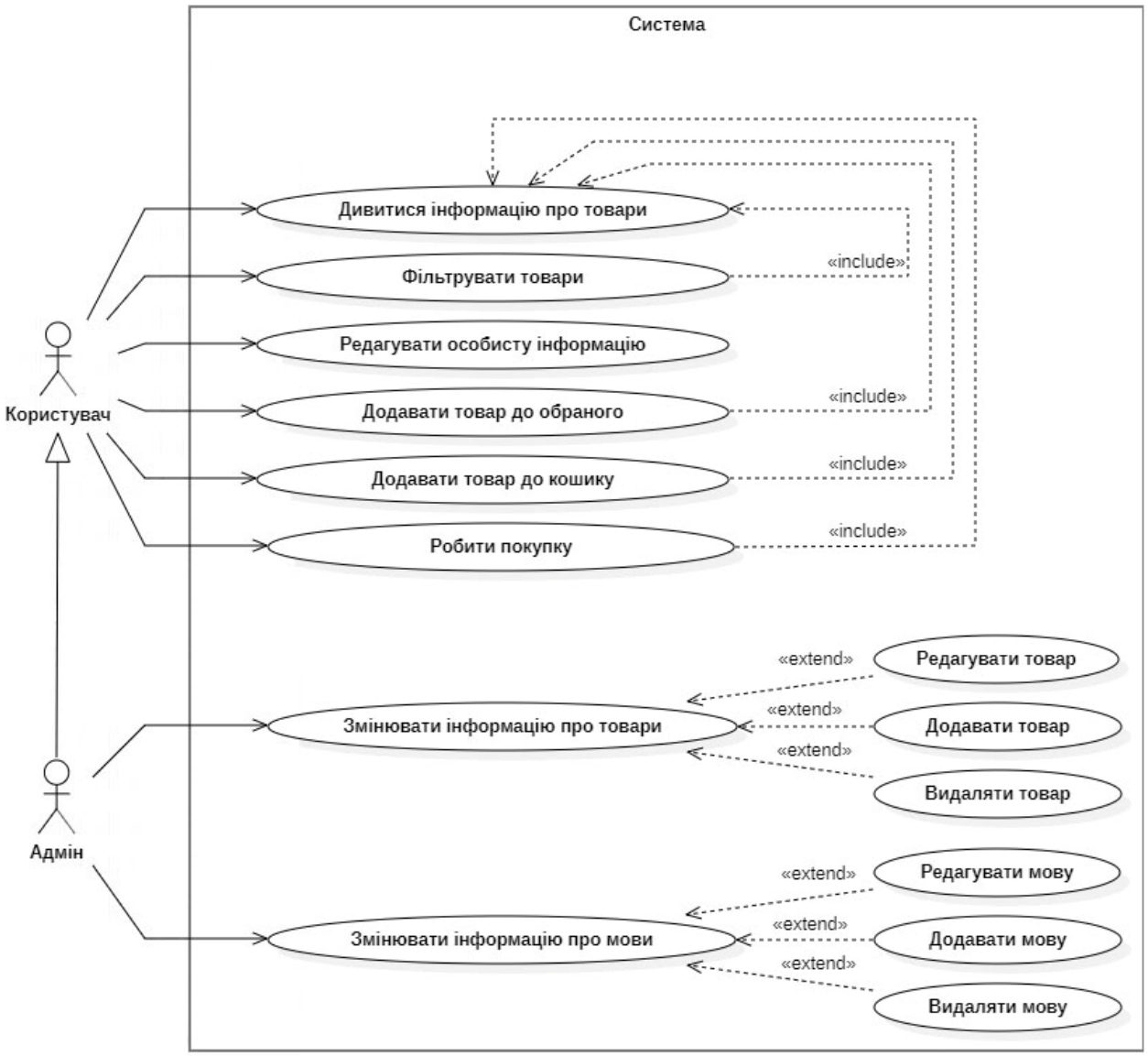

Рисунок 2.2 – Діаграма прецедентів

На даній діаграмі зображено дії, які може робити користувач та адміністратор бази даних. Ролі яких розділяються конфігураціями розташованими на сервері.

2.3 Функціональні вимоги

1. Функція входу до мобільного застосування.

F1.1. Користувач проходить ідентифікацію через MAC-адресу пристрою.

2. Функція навігації по мобільному застосуванню.

F2.1. Після логіну до мобільного застосування відкривається меню навігації,

де користувач має змогу обрати потрібне йому вікно. Першим вікном є «Категорії».

3. Функція зміни мови мобільного застосування.

F3.1. Зміна мови в операційній системі.

FR3.1.1. Якщо застосунок підтримує мову, мова змінюється на обрану.

FR3.1.2. Якщо застосунок не підтримує обрану мову, мова змінюється на англійську.

4. Функція перегляду товарів.

F4.1. Якщо в категоріях обрати потрібну підкатегорію товарів, відкриється вікно перегляду товарів.

5. Функція фільтрування товарів.

F5.1. Вибір потрібного фільтру.

FR5.1.1. Якщо фільтр не був обраний, відображаються усі товари.

FR5.1.2. Якщо фільтр був обраний, відображаються відфільтровані товари.

6. Функція додавання товарів до обраного.

F6.1. Коли у вікні перегляду товарів користувач натискає на жовте серце, товар додається до списку обраного.

7. Функція додавання товарів до кошику.

F7.1. У вікні перегляду товарів та списку обраного користувач може натиснути на зелений кошик і товар додасться до кошика.

8. Функція створення покупки.

F8.1. У вікні кошика, користувач натискає кнопку «Створити покупку».

F8.1.1. Якщо у кошику були товари, користувач створить покупку.

F8.1.2. Якщо у кошику не було товарів, з'явиться повідомлення с помилкою.

9. Функція редагування профілю.

F9.1. Користувач заходить у вікно профілю.

FR9.1.1. Користувач може подивитися дані.

FR9.1.2. Якщо натиснути на кнопку редагування профілю, користувач може змінити свої дані.

10. Функція редагування товарів.

F10.1. Роблячи запит на сервер, адмін може змінити дані про товар.

F10.2. Роблячи запит на сервер, адмін може додати товар.

F10.3. Роблячи запит на сервер, адмін може видалити товар.

11. Функція редагування мови.

F11.1. Роблячи запит на сервер, адмін може змінити дані про мову.

F11.2. Роблячи запит на сервер, адмін може додати мову.

F11.3. Роблячи запит на сервер, адмін може видалити мову.

2.4 Нефункціональні вимоги

NF1. Цільовою платформою, на якій очікується виконання мобільного додатка, повинен бути Android версії вище 8.0.

NF2. Для коректної праці додатку потрібен доступ до мережі Інтернет.

NF3. Для коректної праці додатку потрібен дозвіл на використання MAC-адреси пристрою.

NF4. Для безпечної передачі інформації через мережу Інтернет треба використовувати протокол HTTPS.

NF5. Для коректної праці додатку треба, щоб пристрій задовольняв системні вимоги по роботі з ОС Android 8.0 та вище.

NF6. Для коректної праці серверу можливо не більше 1024 одночасних підключень

NF7. Для коректного масштабування серверу потрібно мати змогу розширювати характеристики серверу та бази даних без втрати даних користувачів і товарів.

NF8. Мобільний застосунок повинен працювати без помилок 24/7.

2.5 Ідентифікація архетипу ІС

Клієнтська частина буде розроблена на мові програмування Kotlin. Серверна частина буде розроблена на мові програмування Java 17, фреймворк Spring Boot. В розробці будуть використані скрипти Gradle (Додаток А.1) (Додаток

Б.1) для «складання» проекту, СУБД PostgreSQL (Додаток Б. 5) для взаємодії з базою даних, мова розмітки XML для написання інтерфейсу мобільного застосування та текстовий формат обміну даними JSON для взаємодії з сервером. Інформаційна система відноситься до архетипу Mixed Application, яке поєднує в собі такі архетипи як: Mobile Application, Service Application (Додаток Б.6) та Calculating Application.

2.6 Користувацький інтерфейс

Для представлення Інформаційної Системи скористаємося макетами. На даних макетах відобразимо приблизний вигляд основних частин додатку та їх функціонал. Перше, що бачить користувач запустивши мобільний додаток, - це його головна сторінка, з якої він може перейти в меню.

Макет вікна «Меню». Усі вікна будуть мати доступ до лівого меню. Це меню є головним навігаційним елементом мобільного застосування. Натискання на елемент 1.1 відкриє вікно «Категорії», 1.2 відкриє вікно «Обране», 1.3 відкриє вікно «Кошик», 1.4 відкриє вікно «Покупки», 1.5 відкриє вікно «Особисті дані», 1.6 відкриє вікно «Про компанію», 1.7 відкриє вікно «Зворотній зв'язок».

Макет вікна «Категорії». У вікні категорій знаходяться списком усі існуючі категорії товарів. Обравши категорію, натиснувши 2.1, відкриється вікно «Підкатегорії».

Макет вікна «Підкатегорії». У вікні підкатегорій знаходяться списком усі існуючі підкатегорії товарів. Обравши підкатегорію, натиснувши 3.1, відкриється вікно «Товари».

Макет вікна «Товари». Після вибору підкатегорії, відкриється вікно за списком товарів. Натиснувши кнопку 4.1 відкриється вікно «Опис товару», натиснувши кнопку 4.2 товари відсортується, обраним користувачем способом, натиснувши кнопку 4.3, відкриється вікно «Фільтр».

Макет вікна «Фільтр». Щоб виставити та застосувати фільтр, треба обрати один з параметрів характеристики фільтру 5.3, змінити діапазон цін 5.2 та

натиснути кнопку 5.1, через яку користувач повернеться у вікно «Товари».

Макет вікна «Опис товару». У вікні опису товару знаходиться повна інформація про товар. Щоб вийти з цього вікна, треба натиснути кнопку «Назад» чи скористатися меню.

Макет вікна «Обране». Якщо користувач додав товар до обраного, він з’явиться у списку обраного. Натиснувши на товар 7.1 користувач відкриє вікно «Опис товару». Щоб вийти з цього вікна, треба натиснути кнопку «Назад» чи скористатися меню.

Макет вікна «Кошик». Якщо користувач додав товар до кошика, він з’явиться у списку кошику. Натиснувши на товар 8.1 користувач відкриє вікно «Опис товару». Натиснувши на кнопку «Придбати» 8.2, товари перейдуть до покупок. Щоб вийти з цього вікна, треба натиснути кнопку «Назад» чи скористатися меню.

Макет вікна «Покупки». Якщо користувач придбав товар, він з’явиться у списку покупок. Натиснувши на товар 9.1 користувач відкриє вікно «Опис товару». Щоб вийти з цього вікна, треба натиснути кнопку «Назад» чи скористатися меню.

Макет вікна «Особисті дані». У цьому вікні користувач може переглянути особисті, якщо їх треба змінити, треба натиснувши на кнопку 10.1. Щоб вийти з цього вікна, треба натиснути кнопку «Назад» чи скористатися меню.

Макет вікна «Про компанію». У цьому вікні користувач може переглянути інформацію про компанію до мобільне застосування. Щоб вийти з цього вікна, треба натиснути кнопку «Назад» чи скористатися меню.

Макет вікна «Зворотній зв’язок». У цьому вікні користувач може знайти дані для зв’язку з технічною підтримкою. Щоб вийти з цього вікна, треба натиснути кнопку «Назад» чи скористатися меню.

Діаграма станів переходів між вікнами представлена на рисунку 2.3.

Рисунок 2.3 – Діаграма станів переходів між сторінками

На діаграмі навігація по пуктах меню, яке розтошовано у нижній частині застосунку.

2.7 Логічне уявлення ІС

Для уявлення майбутньої системи побудуємо логічну діаграму зв’язку компонентів системи (рисунок 2.4).

Модель складається з:

- мобільного застосування;
- користувача;
- бази даних;
- серверу;
- адміністратору.

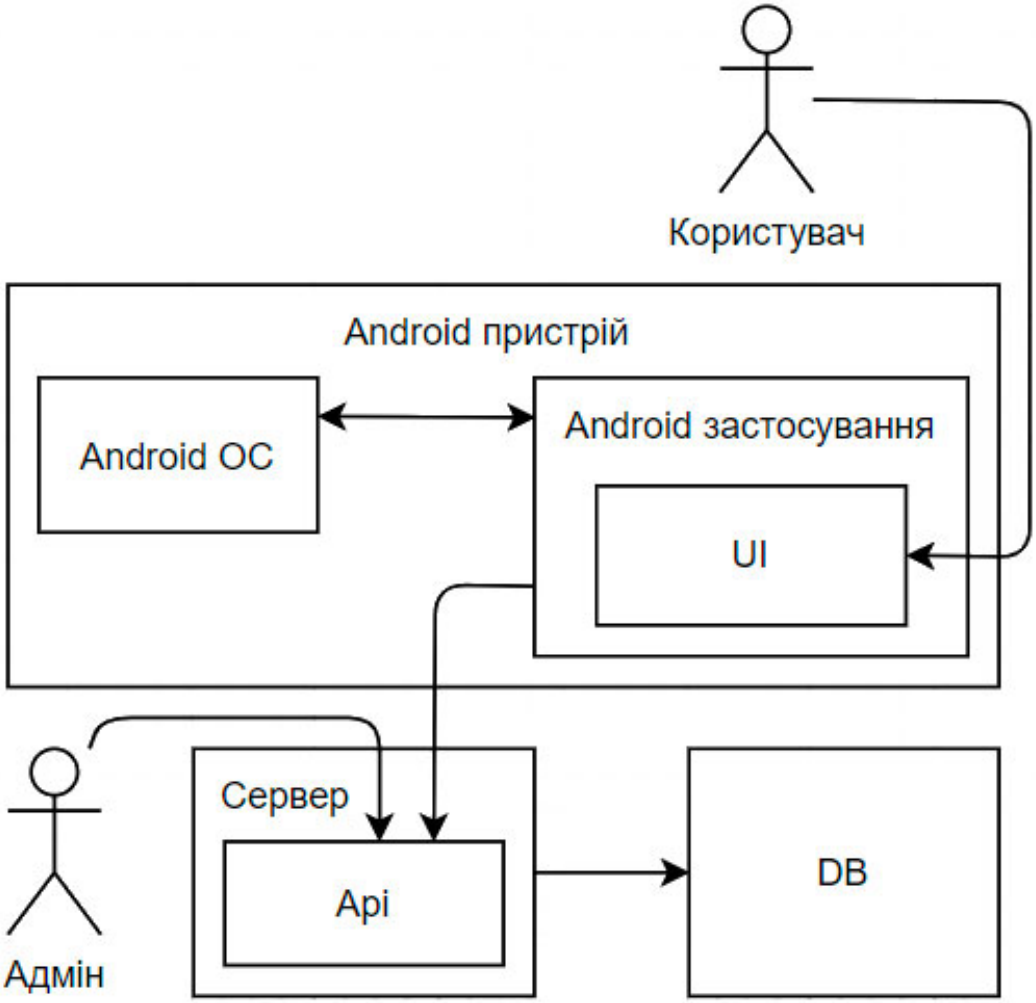

Рисунок 2.4 – Схема логіки роботи системи та зв'язку компонентів

На цій діаграмі можна побачити взаємодію користувача з мобільним додатком та зв'язок додатку з сервером та базою даних.

2.8 Уявлення розгортання ІС

Так як клієнтська частина працює Api, інформація отримується з серверу, то потрібен доступ та мережі Інтернет, бо без обох цих умов неможливе її коректне функціонування.

Ієрархія взаємодії компонентів системи вказана на рисунку 2.5.

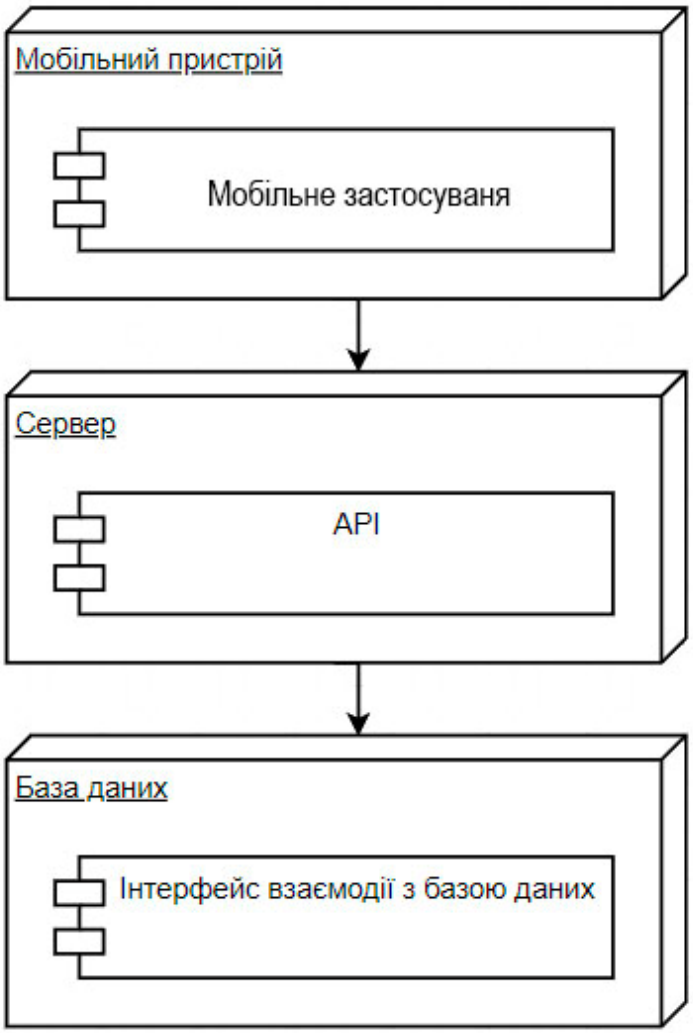

Рисунок 2.5 – Діаграма розгортання

Взаємодія користувача з елементами додатку здійснюється завдяки користувацькому інтерфейсу, тобто не які функціональні дані не зможуть надати доступ до інформації якщо користувачем не будуть виконанні певні дії.

2.9 Уявлення процесів ІС

Розглянемо один з головних процесів, який може відбуватися під час роботи користувача з інформаційною системою.

У цьому сценарії користувач хоче знайти та додати товар до обраного. Діаграма процесів для сценарію представлена на рисунку 2.6.

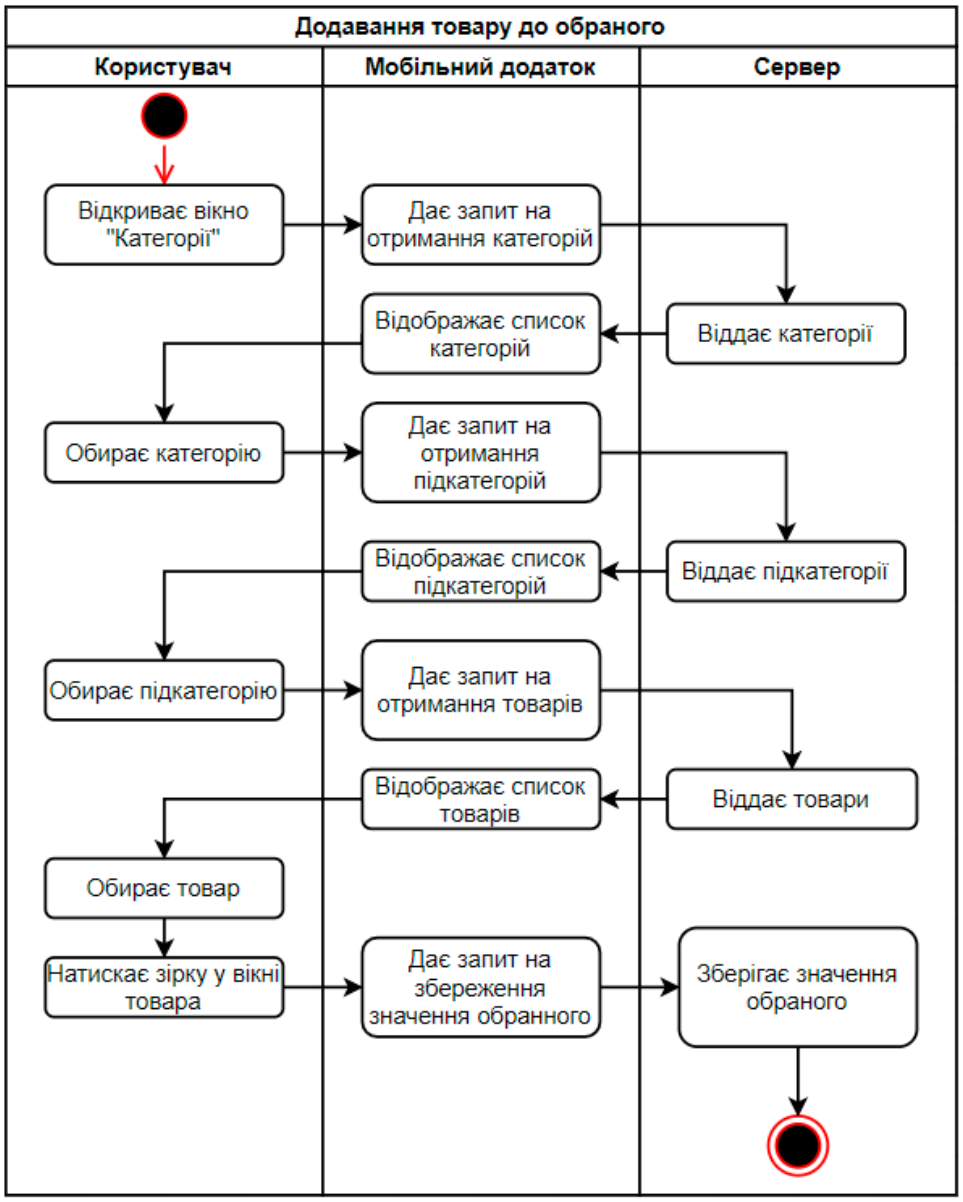

Рисунок 2.6 – Діаграма процесів для сценарію

Після додавання товару до «Обранного» користувач повинен залишатися на екрані зі списком знайдених товарів.

2.10 Опис обробки строкових значень

У стандартній архітектурі на стороні Android використовується файл strings.xml, де зберігаються всі текстові дані. На даний момент аналогічний файл є на всіх існуючих платформах. Це збільшує вартість розробки та підтримки додатків на різних платформах і може призвести до помилок через людський фактор.

Було прийнято рішення перенести всі текстові та колірні значення на сервер. Це дозволить змінювати, видаляти та додавати текстові поля одразу на всіх платформах без зміни програмного коду та оновлення мобільного додатка. У розроблюваному інтернет-магазині ці дані будуть зберігатися на сервері в таблиці «Strings», завантажуватися при запуску додатка та кешуватися для подальшого використання. Наприклад, якщо розробник хоче додати підтримку нової мови до мобільного додатка, достатньо оновити дані на сервері, і підтримка нової мови з'явиться без необхідності оновлення мобільного додатка.

2.11 Опис алгоритму роботи фільтру

В умовах, коли збереження даних виконується серверною частиною, а їх аналіз клієнтською частиною, створення налаштувань для фільтру товарів виконується розробником в ручному режимі на основі аналізу предметної області зі всіма характеристиками та їх значеннями для товарів. Але в разі, якщо додається товар, що володіє абсолютно новими характеристиками, треба доповнювати статичний код фільтру. Це здорожчує розробку і підтримку застосування. Тому було прийнято рішення створили динамічний фільтр. Для створення динамічного фільтру треба зробити такі кроки:

– спроектувати структуру бази даних таким чином, щоб усі строкові назви параметрів існували лише у вигляді масивів. Це потрібно для реалізації функції підтримки кількох мов на боці серверу. Так само треба провести налаштування паралельного доступу і зміни бази даних. Реалізувати метод для отримання та перебору масиву строкових значень, що складається з об'єктів таблиці «Товари»;

– розробити метод, який при запиті на отримання фільтру, буде робити SQL запит, створюючи виборку унікальних значень з базі даних;

– розробити метод, який формує масив параметрів, які після перетворення в формат JSON (Додаток Б.4), будуть відправлені на сторону клієнта;

– розробити метод клієнта, який буде отримувати, обробляти та відображати ці дані користувачу у вигляді зручного інтерфейсу. Користувач буде обирати

потрібні йому параметри з представленого списку, у цей момент мобільне застосування повертатиме обрані параметри на сервер;

– сервер знаходить потрібні товари в базі даних та повертає їх на сторону клієнта;

– щоб зберегти швидкий час відклику серверу, функцію обновлення фільтру можливо почати проводити не по запиту користувача на отримання фільтру, а регулярно по деякому часу (CRON). Наприклад, раз у 3 години.

На основі цих даних будується фільтр, який включає в себе абсолютно всі значення, наявні у базі даних. Наприклад, до бази даних додається товар, ціна якого вища, ніж максимальна ціна попереднього товару. При отриманні фільтру алгоритм переписує попередні результати, та максимальна ціна, яка вказана у фільтрі оновлюється без участі людини. За маленької кількості товарів дана операція буде займати мало часу, та її можливо проводити перед кожним отриманням клієнтом фільтру.

2.12 Опис стеку технологій

Технології, що використовуються у клієнтському застосуванні:

- Kotlin – це об'єктно-орієнтована мова програмування;
- AndroidStudio – інтегроване середовище розробки для платформи Android;
- Gradle – система збирання проекту;
- XML – мова розмітки для написання інтерфейсу мобільного застосування;
- JSON – формат даних, для взаємодії з сервером.

Технології, що використовуються на сервері:

- Java – це об'єктно-орієнтована мова програмування;
- Spring Boot – це фреймворк Java;
- IntelijIdea – інтегроване середовище розробки, що підтримує Java;
- PostgreSql – об'єктно-реляційна СУБД;
- JSON – формат даних для взаємодії з клієнтом.

2.13 Висновки до другого розділу

У данному етапі було проведено детальне проектування програмного забезпечення інформаційної системи. В ході роботи були визначені функціональні та нефункціональні вимоги, побудована діаграма прецедентів для відображення цих вимог, а також спроектовано інтерфейс користувача. Додатково були розроблені діаграми активності та послідовності для двох ключових сценаріїв системи. Також було визначено структуру компонентів, дані, інтерфейси та процеси програмної системи, а також розглянуті питання операційного представлення. У роботі був описаний стек технологій, використаний у розробці інформаційної системи. Загальною метою цього етапу було повне та деталізоване проектування програмного забезпечення, щоб підготувати підґрунтя для подальшої реалізації системи.

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Уявлення про структуру проекту ІС

Коренева папка розділена на менші папки, кожна з яких виконує свою частину відповідальності.

У папці «activities» знадяться усі класи, які успадковуються від класу «Activity». Ці класи відповідають за відображення усіх елементів на екрані. Клас «StartActivity.kt» відповідає за екран загрузки, а клас «MainActivity.kt» (Додаток А.2) (Додаток А.3) координує всю роботу мобільного застосування.

У папці «drawer» знаходяться класи, що відповідають за відображення вікон, присутніх у меню.

У папці «fragments» знаходяться класи, що відповідають за вікна, які відображуються під час роботи мобільного застосування.

У папці «repository» знаходяться класи, що поєднують класи парсингу даних з класами, що відповідають за відображення даних на екрані.

У папці «retrofit» знаходяться класи, що взаємодіють з бібліотекою «Retrofit2»

У папці «utils» знаходяться допоміжні класи, функціонал яких можливо використати в усіх місцях програми. В основі це класи для взаємодії з операційною системою Android. Наприклад, отримання мови системи чи запит на отримання дозволів мобільного застосування.

3.2 Уявлення про роботу класів ІС

3.2.1 Реалізація навігації

Діаграма роботи навігації у застосуванні наявна на рисунку 3.1.

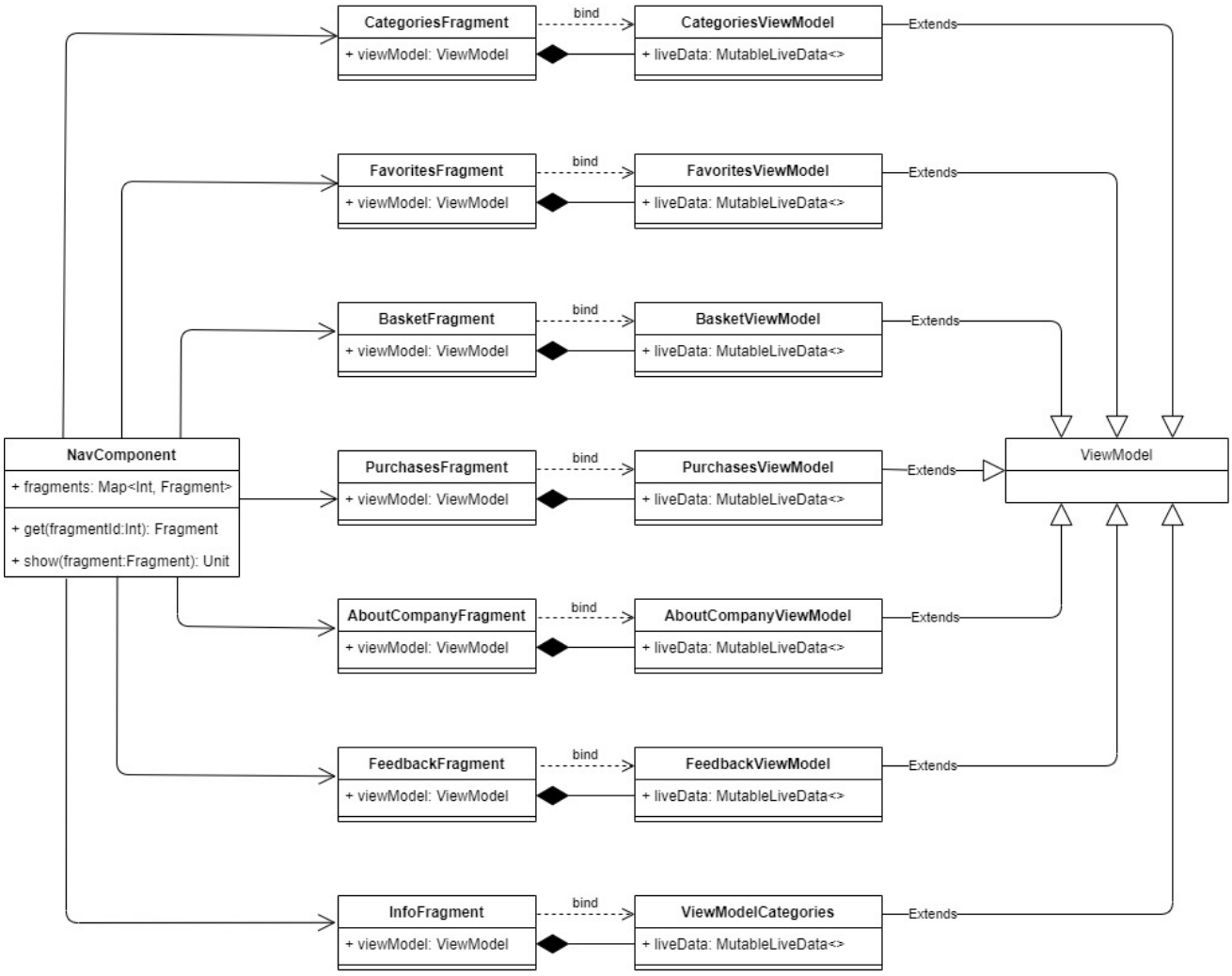

Рисунок 3.1 – Діаграма класів роботи навігаційного компонент

NavComponent використовується для того, щоб полегшити взаємодію вікон між собою та більш легким переходом між сторінками. NavComponent отримає граф переходів з ресурсів застосування та може використовувати його у будь якому місці у програмі.

Для того, щоб відкрити потрібне вікно, треба у NavComponent передати об’єкт чи id вікна, що треба відобразити. Блок-схема процесу відкриття нового вікна відображена на рисунку 3.2.

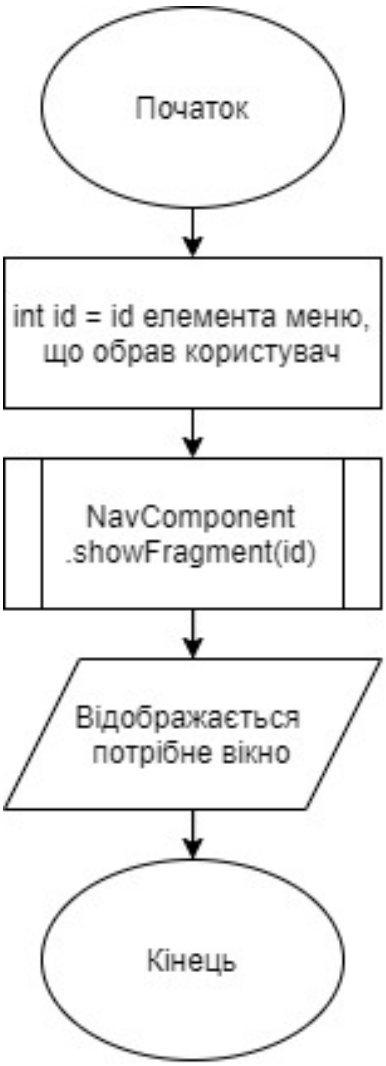

Рисунок 3.2 – Блок-схема відображення вікна через навігаційний компонент

NavComponent являється частиною технологічного пакету Android Jetpack, який Google презентував та зробив стадартом для розробки мобільних застосувань у 2017 році.

3.2.2 Реалізація взаємодії клієнта і сервера

Для того, щоб мобільний додаток мав змогу взаємодіяти з сервером, треба створити парсер даних.

Для великого проекту створювати кожен запит окремо занадто довго, тому треба зробити систему для парсингу, що спростить взаємодію з сервером.

Парсер даних було створено за допомогою бібліотеки Retrofit2 (Додаток А.), яка у собі містить конфігурації для легких запитів.

Діаграма класів для системи парсингу даних представлена на рисунку 3.3.

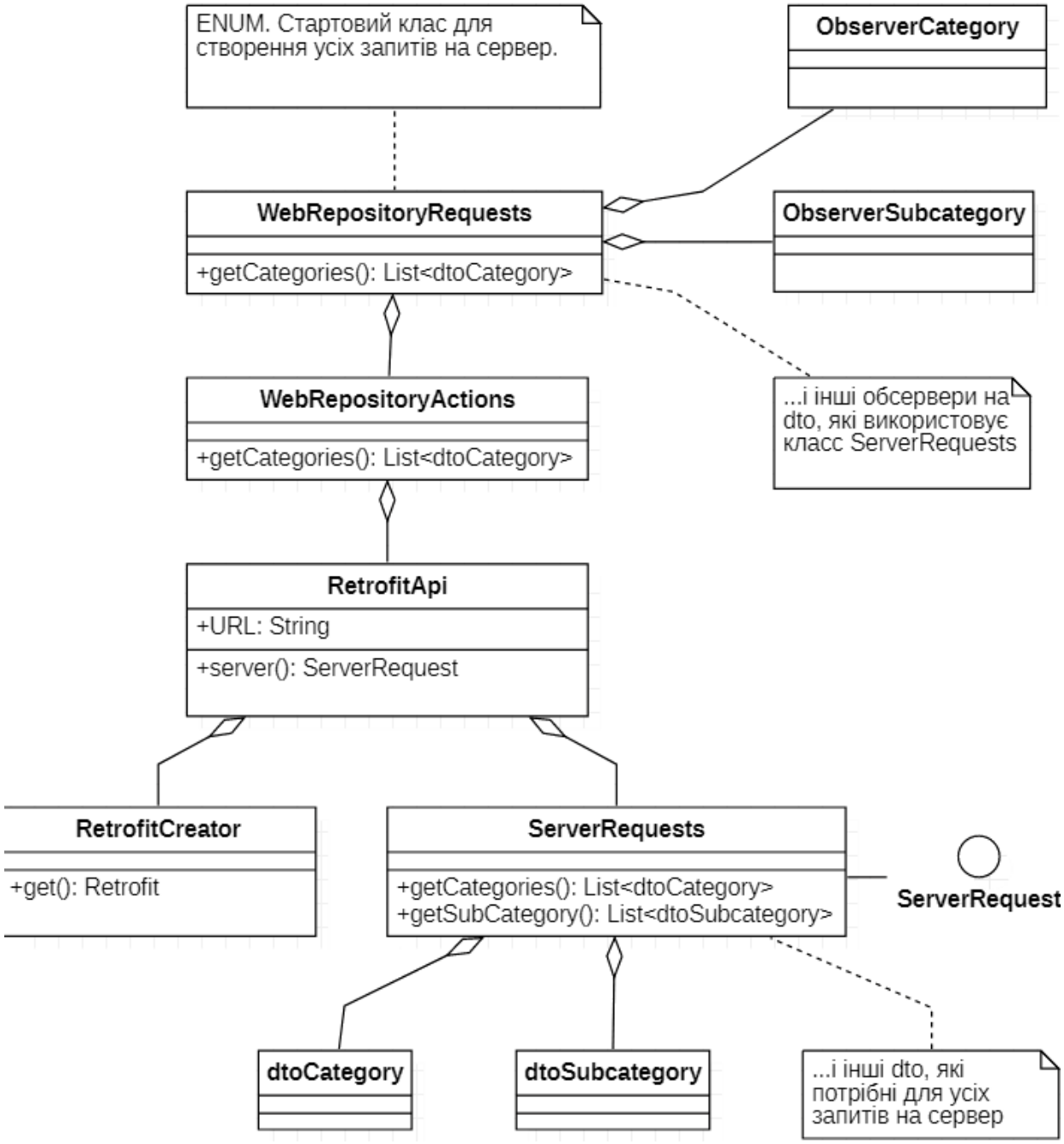

Рисунок 3.3 – Діаграма класів системи запитів на сервер

Інстанс класа Retrofit створений за допомогою патерну Singleton для того щоб оптимізувати використання оперативної пам’яті мобільного пристрою.

3.2.3 Реалізація фільтру

На рисунку 3.4 відображена діаграма класів частини коду, що відповідає за фільтр. У склад діаграми входять як класи, відповідаючі за візуальне оформлення програми, так і класи бізнес-логіки.

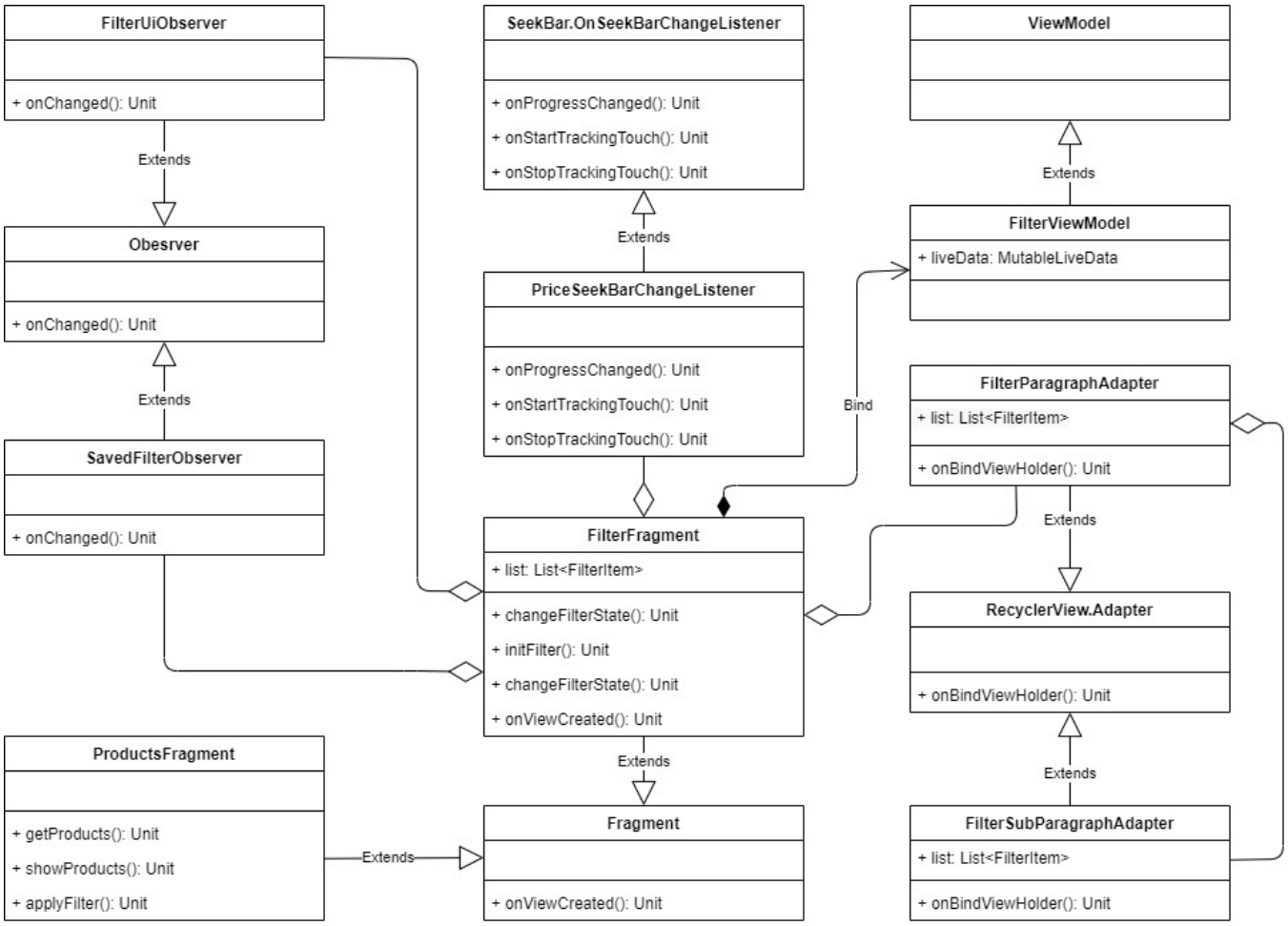

Рисунок 3.4 – Діаграма класів роботи з фільтром

Життєвий цикл роботи фільтру починається з відкриття вікна зі списком товарів (Дадаток Б.2), та закінчується, коли користувач отримає дані товарів вже з використаним фільтром.

Кожен фільтр, якій відображається користувачу, створюється з нуля, коли на сервер робиться запит на його отримання. Якщо у цей момент додати до бази даних новий товар, характеристики якого унікальні – структура фільтру зміниться без виправлення коду сервера та клієнта.

Блок схема життєвого циклу роботи фільтру наведена на рисунку 3.5.

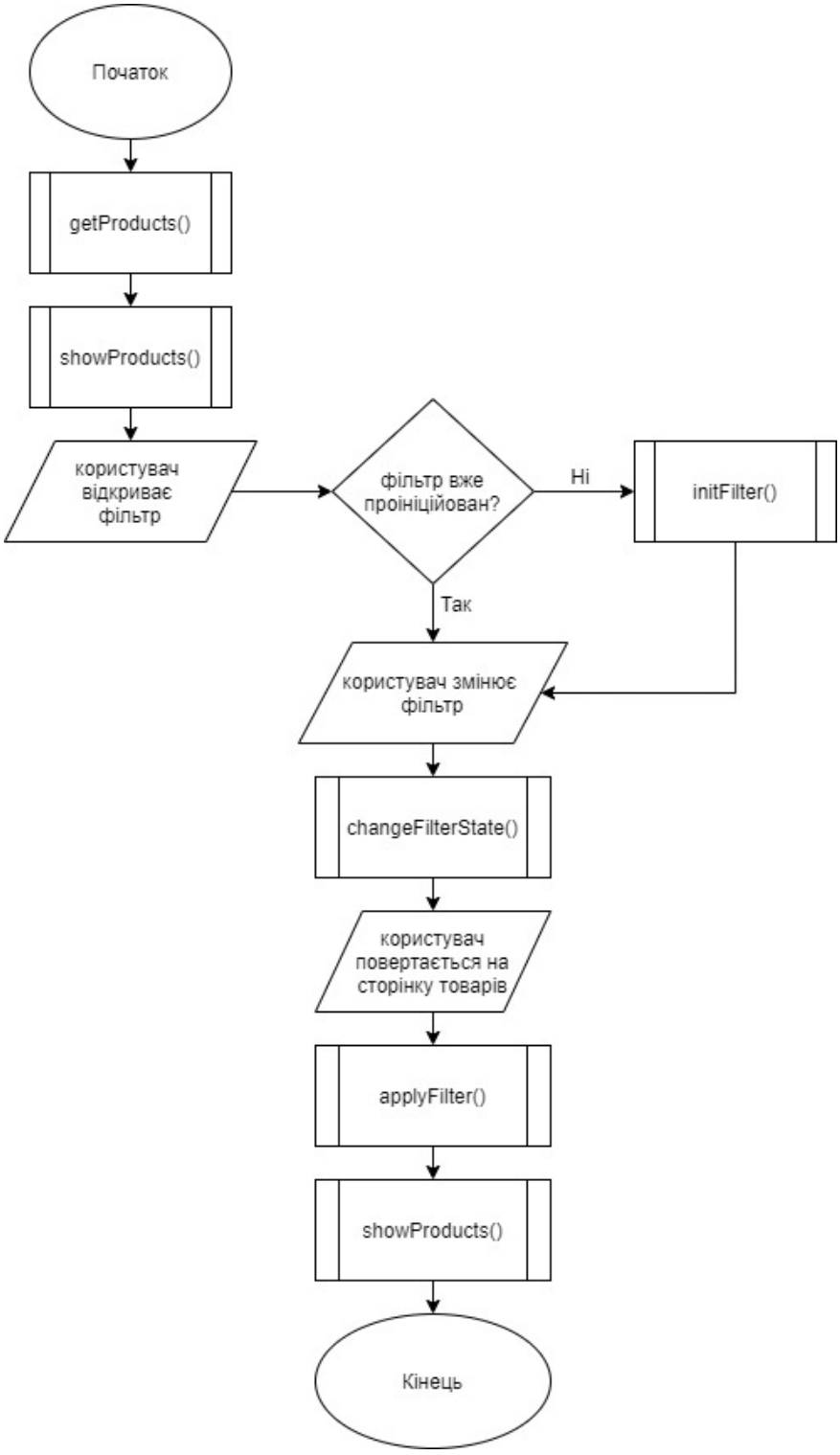

Рисунок 3.5 – Блок схема життєвого циклу роботи фільтру

Коли відображається на екрані головна сторінка фільтру, відпрацьовує код методу onCreateView(), блок-схема якого представлена на рисунку 3.6.

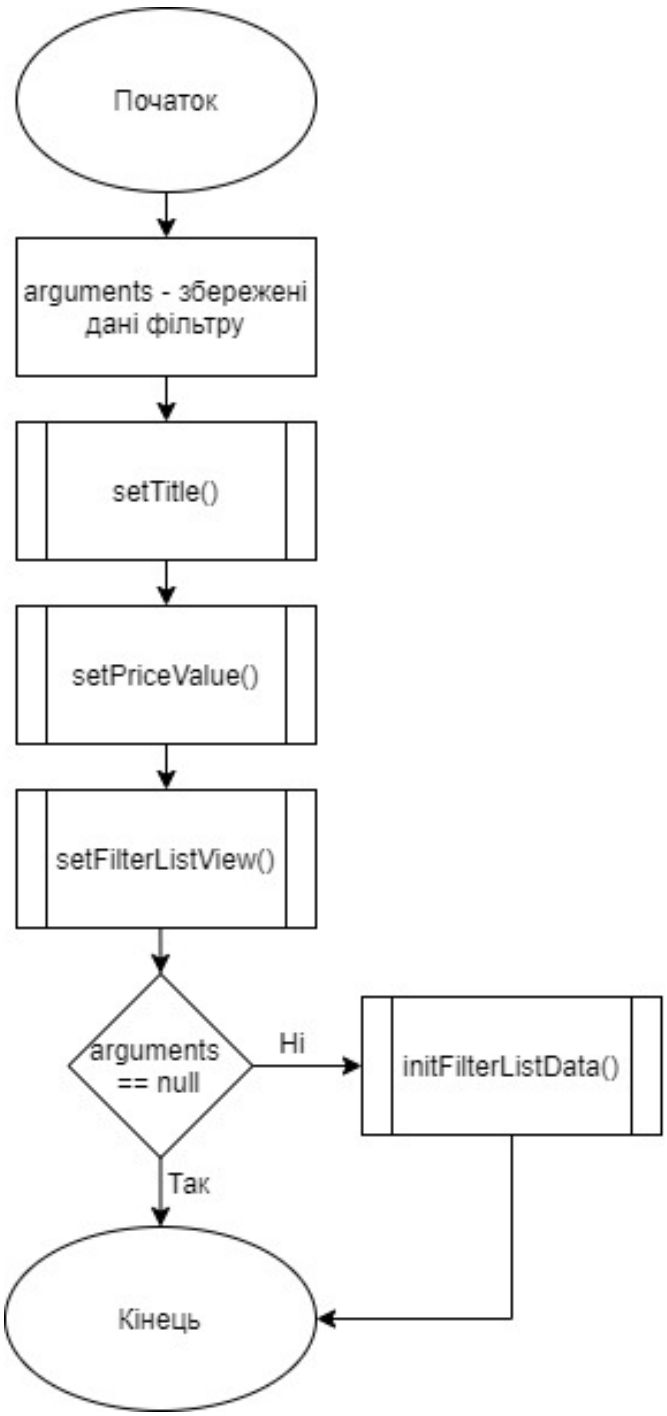

Рисунок 3.6 – Блок-схема роботи методу FilterFragment onCreateView

Коли дані для застосування фільтру вже отримані з серверу, треба їх відобразити на екрані. Для відображення списку характеристик використовується бібліотека «RecyclerView», логіка якого представлена у блок-схемі (рисунок 3.7).

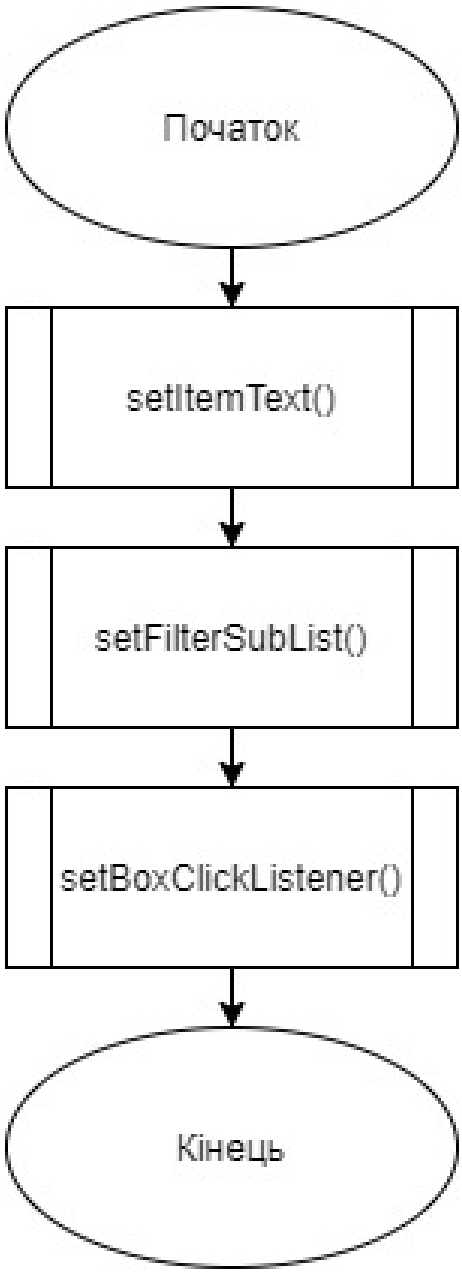

Рисунок 3.7 – Блок-схема роботи методу FilterParagraphAdapter onBindViewHolder

Для реалізації списків фільтру використовується технологія RecyclerView (Додаток А.4). На сторінці з фільтром, крім характеристик товару, ще є поле з введенням максимальної ціни за товар. Коли користувач змінює значення цього поля, спрацьовує код методу onProgressChange, блок-схема якого представлена на рисунку 3.8.

OnProgressChange – це один із методів обробки ходунка, через який змінюється ціна у вікні фільтра.

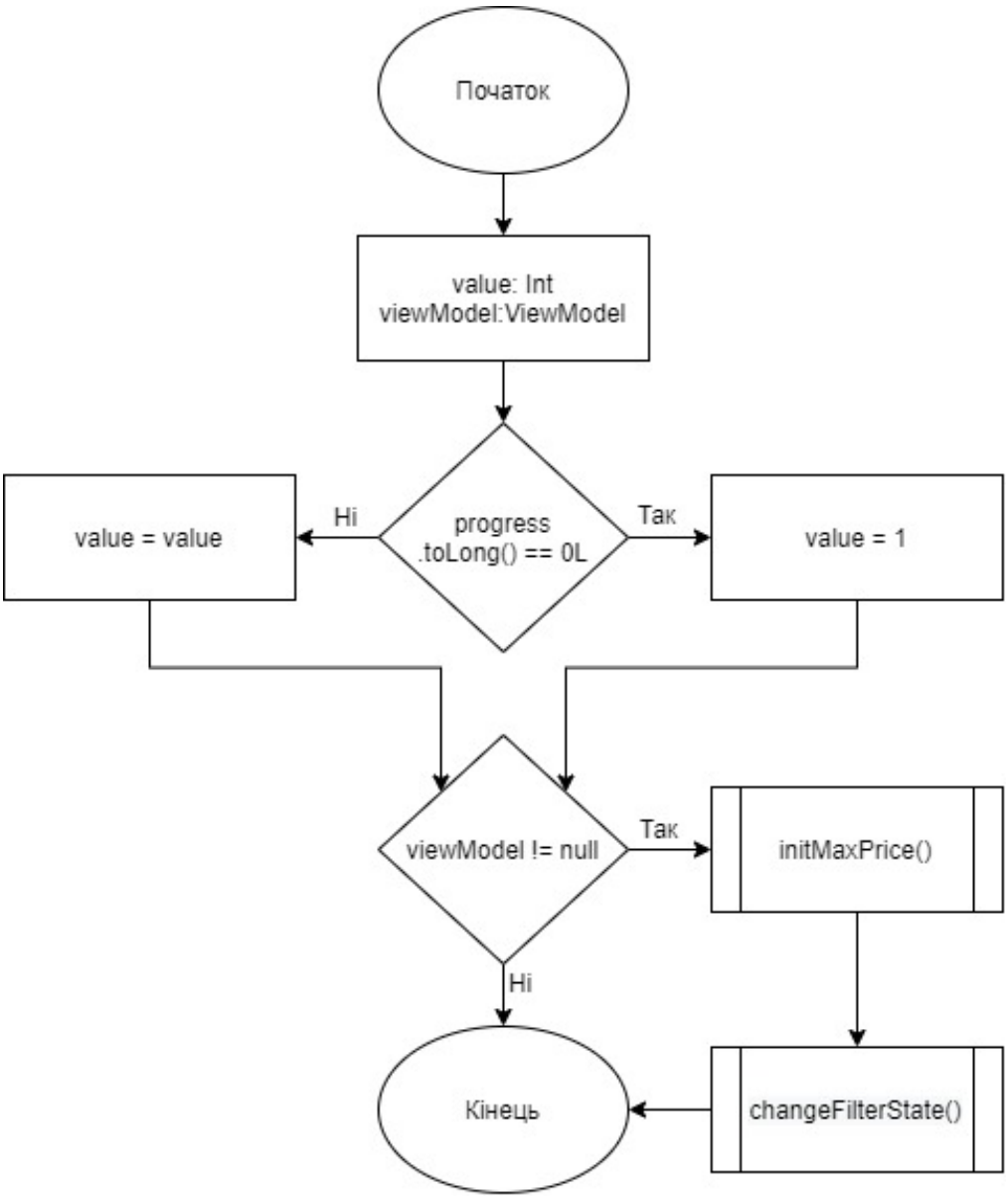

Рисунок 3.8 – Блок-схема роботи методу PriceSeekBar onProgressChange

Коли користувач з клієнта робить запит на отримання фільтру – фільтр створюється з нуля із таблиць товарів, які є у базі даних. Якщо у цей момент додати до бази даних новий товар, характеристики якого унікальні – структура фільтру зміниться без виправлення коду сервера та клієнта.

Так як фільтр динамічний, він не існує у вигляді таблиці. Фільтр створюється з товарів, які вже існують у базі даних (рисунок 3.9). Ініціалізація бази даних

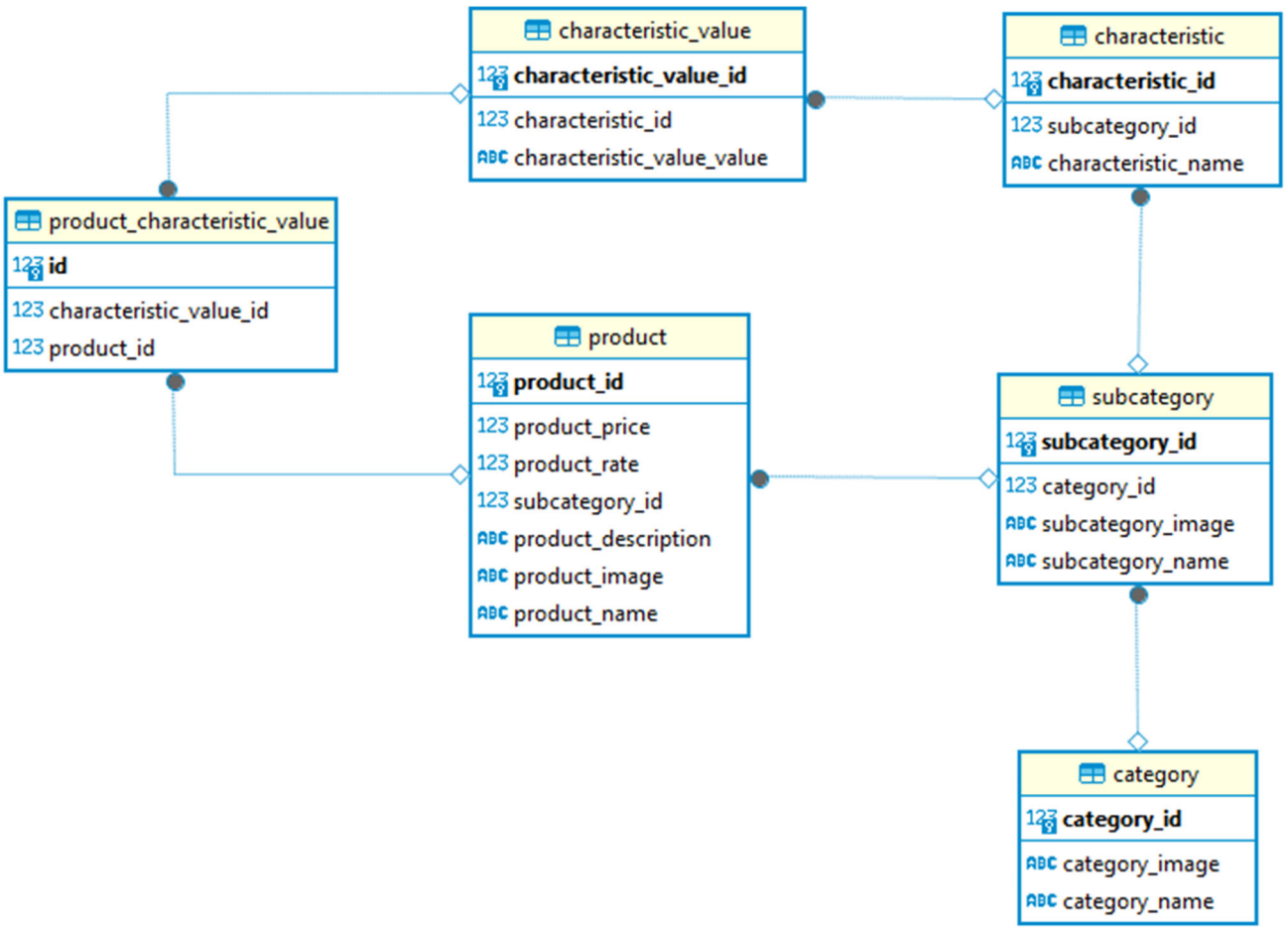

Рисунок 3.9 – Таблиці бази даних, пов'язані с фільтром

Кожен товар має свій набір властивостей (properties), наприклад, діагональ екрану чи обсяг оперативної пам'яті. Кожна властивість має свій набір параметрів (parameters), для оперативної пам'яті: 2gb, 4gb, 8gb тощо. Для properties і parameters також доступна локалізація.

Наприклад, відеокарти мають такі характристики:

- об'єм пам'яті;
- швидкість шини пам'яті;
- частота шини пам'яті;
- версія ядра;
- ціна;
- модель;
- рік виготовлення;
- країна виробник;

– тощо.

Відповідь сервера при запиті товару (product) представлена на рисунку 3.10.

```
{
    "productId": 1,
    "productName": "GTX1660",
    "productPrice": 10000,
    "productRate": 4,
    "productImage": "https://content2.rozetka.com.ua/goods/images/big/288153948.jpg",
    "productDescription": ""
},
```


Рисунок 3.10 – Відповідь сервера на запит отримання товару

У відповіді parameters – це масив, так як властивість може містити кілька параметрів. Наприклад, у ноутбуці може бути 2 відеокарти – intel та nvidia.

Властивості і параметри потрібно буде створити окремо, двома різними енд-поінтами, спочатку створюємо властивості вказавши тільки ім’я властивості. Потім можна створювати параметри вказавши name та id властивості, за яким параметр буде закріплений. Наприклад, створюємо властивості процесору і за ним закріплюємо параметри: intel i5, intel i7 тощо. Можна буде запитати список всіх властивостей і список параметрів по id властивості.

Як це працює в застосуванні:

– запитуємо список категорій;

– запрошуємо список підкатегорій по id категорії;

– за id підкатегорії одночасно запитуємо список товарів і список фільтрів двома різними енд-поінтами (щоб отримати список фільтрів і їх параметрів тільки для існуючих товарів, потрібно вказати id підкатегорії);

– на сервері отримаємо усі товари з підкатегорії;

– з товарів отримуємо їх унікальні властивості;

– з властивостей отримуємо їх унікальні параметри.

Щоб відфільтрувати товари, потрібно буде вказати id підкатегорії, список властивостей і параметрів. Відповідь сервера на запит фільтру представлена на рисунку 3.11.

```
{
    "minPrice": 10000,
    "maxPrice": 25000,
    "filterItems": [
        {
            "characteristicId": 1,
            "characteristicName": "Кол-во TeraFlops",
            "characteristicValues": [
                {
                    "characteristicValueId": 1,
                    "characteristicValue": "10101"
                },
                {
                    "characteristicValueId": 2,
                    "characteristicValue": "20202"
                },
                {
                    "characteristicValueId": 5,
                    "characteristicValue": "22404"
                }
            ]
        },
        {
            "characteristicId": 2,
            "characteristicName": "Трассировка лучей",
            "characteristicValues": [
                {
                    "characteristicValueId": 3,
                    "characteristicValue": "Да"
                },
                {
                    "characteristicValueId": 4,
                    "characteristicValue": "Нет"
                }
            ]
        }
    ]
}
```


Рисунок 3.11 – Відповідь сервера на запит фільтру

Відповідь сервера на запит отримання фільтру може бути великого розміру. У майбутньому для оптимізації парсингу даних можна використовувати сучасну технологію передачі даних «Protobuf».

3.2.4 Реалізація системи зберігання строкових значень

У кожному Android застосуванні є файл «strings.xml», представлений на рисунку 3.12. У цьому файлі зберігаються усі строкові значення, які використовуються у застосуванні. Якщо треба буде створити застосування для іншої операційної системи, то усі значення треба буде скопіювати ще раз. Це доставляє багато незручностей.

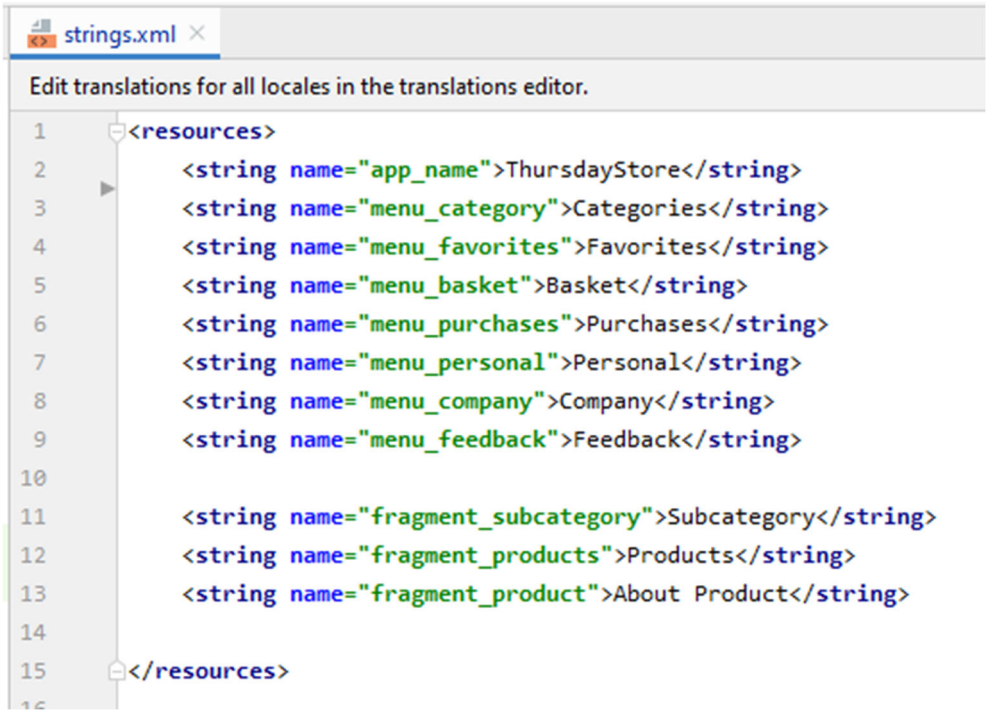

Рисунок 3.12 – Файл «strings.xml»

Для того, щоб позбавитись від цієї проблеми, усі строкові значення були перенесені до таблиці «String» (рисунок 3.13) на сервері, та отримуються клієнтом у форматі JSON.

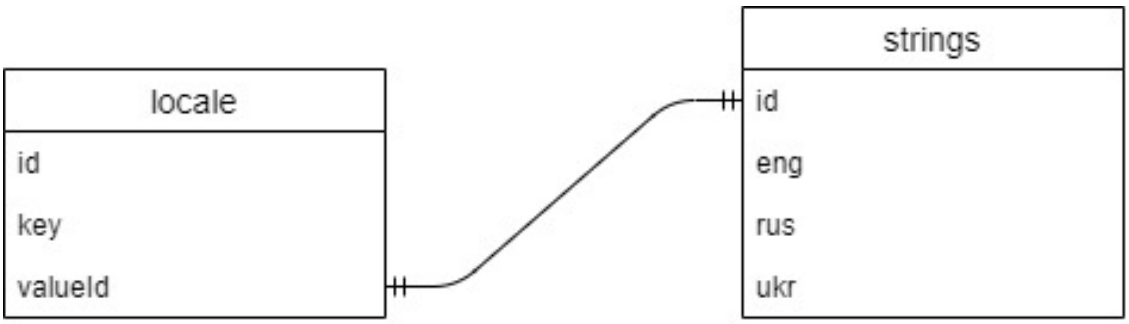

Рисунок 3.13 – Таблиця «Strings»

Приведення бази даних до 3 нормальної форми зменшує швидкість роботи бази даних, тому було прийнято рішення не дотримуватись цього стандарту при розрообці архітектури бази даних.

3.3 Стратегія доставки продукту

Код ІС буде поширюватися за допомогою сервісу – «Play Market». Play Market – це магазин додатків, а також ігор, книг, музики і фільмів від компанії Google, що дозволяє стороннім компаніям пропонувати власникам пристроїв з операційною системою Android встановлювати і купувати різні застосування [7].

3.4 Керівництво користувача

Коли користувач відкриває програму, він потрапляє на головний екран, який відображає категорії усіх товарів. У лівому верхньому кутку розташована кнопка з трьома горизонтальними лініями. З її допомогою можна потрапити до вікна головного меню.

До вікна «Меню» можна повернутись у будь-який момент або за допомогою кнопки у лівому верхньому кутку з трьома горизонтальними лініями, або змахнувши пальцем по екрану пристрою зліва направо. Меню умовно розділено на три групи.

У першій умовній групі назва програми та ім’я користувача. У другій умовній групі категорії, обране, корзина та покупки. У третій приватні налаштування, інформація про компанію та зворотній зв’язок.

У вікні «Категорії» відображається список усіх можливих категорій, які є у магазині. Для прикладу натискаємо на категорію «Комплектуючі ПК». Надалі користувач завжди може повернутись назад за допомогою стрілки у лівому верхньому кутку.

Коли користувач натискає на будь-яку з категорій, то потрапляє у вікно відповідних підкатегорій.

Для прикладу натискаємо підкатегорію «Відеокарти». У вікні «Підкатегорії» користувач бачить перед собою список доступних товарів, які розташовані сіткою. У вікні кожного товару є його фотографія, назва та ціна. У правому нижньому кутку кнопка додавання в кошик, у правому верхньому кутку кнопка додавання в обране. Під назвою підкатегорії розташований рядок налаштування пошуку, який може допомогти користувачу пришвидшити пошук необхідного товару. Він складається з двох кнопок: сортування та фільтр. У сортуванні доступні три способи: за рейтингом, за зниженням ціни та зростанням ціни.

Особливістю програми є фільтр, який самостійно підлаштовується під кожен вид товарів та його особливості. Приклад фільтру підкатегорії «Відеокарти». У вікні «Обране» користувач може додати вподобані товари.

У вікні «Кошик» знаходяться товари, які користувач збирається придбати. У вікні «Покупки» знаходиться список товарів, які користувач вже придбав.

У вікні «Приватні налаштування» користувач може внести інформацію про себе, яка потім йому знадобиться при вчиненні покупок. Кнопка редагування інформації знаходиться у правому верхньому кутку.

Користувач може внести наступні дані: ім’я, прізвище, по-батькові, стать, день народження, телефон, місто та електронну пошту.

У вікні «Про компанію» міститься інформація про компанію, яка створила програму. У вікні «Зворотній зв’язок» міститься контактна інформація, якою користувач може скористатись, якщо у нього зв’явились запитання або проблеми.

3.5 Розрахунок метрик програмного коду ІС

Метрики програмного коду Android наведені у таблиці 3.3.

Таблиця 3.3 – Розрахунок метрик програмного коду Android

Метрика	Значення
Загальна кількість рядків коду в проекті ІС	5000
Середня кількість рядків коду в одному класі	25
Максимальна кількість рядків коду в одному класі	120
Середня кількість рядків коду в одному методі	12
Максимальна кількість рядків коду в одному методі	25
Максимальна глибина дерева спадкування	2
Коментованість коду в % від загальної кількості коду.	5%
Покриття коду модульними тестами. % від загальної кількості коду.	25%

Метрики програмного коду Spring Boot наведені у таблиці 3.4.

Таблиця 3.4 – Розрахунок метрик програмного коду Spring Boot

Метрика	Значення
Загальна кількість рядків коду в проекті ІС	3466
Середня кількість рядків коду в одному класі	30
Максимальна кількість рядків коду в одному класі	426
Середня кількість рядків коду в одному методі	15
Максимальна кількість рядків коду в одному методі	107
Максимальна глибина дерева спадкування	2
Коментованість коду в % від загальної кількості коду.	15%
Покриття коду модульними тестами. % від загальної кількості коду.	25%

Весь код проектів був написаний, дотримуючись стандартів SOLID, GRASP, KISS, DRY.

3.6 Розробка тест-плану ІС

3.6.1 Функціональне тестування

Для проведення функціонального тестування було розроблені спеціальні тест-кейси. Для цього треба створити список з роз'ясненням кожного тест-кейсу.

Список тест-кейсів:

- тест-кейс FR1.1 – Відкрити застосування;
- тест-кейс FR1.2 – Натиснути на будь-яку категорію;
- тест-кейс FR1.3 – Застосувати фільтр;
- тест-кейс FR1.4 – Натиснути на серце інтерфейсу товару;
- тест-кейс FR1.5 – Натиснути на кошик інтерфейсу товару;
- тест-кейс FR1.6 – Натиснути на кнопку «Створити покупку» у елементі меню «Кошик».

Результати функціонального тестування наведені в таблиці 3.5.

Таблиця 3.5 – Результат проведення функціонального тестування

Номер тест-кейсу	Очікуваний результат	Отриманий результат	Коментарі
Тест кейс № 1	Застосування повинно відкритися без помилок	Застосування відкрилося без помилок	Виконано
Тест кейс № 2	Повинне відкритися вікно з підкатегоріями	Відкрилося вікно з підкатегоріями	Виконано
Тест кейс № 3	Товари, які будуть на екрані, повинні відповідати характеристикам, які вказані у фільтрі	Товари, які присутні на екрані відповідають характеристикам, які вказані у фільтрі	Виконано
Тест кейс № 4	Серце повинно заповнитись жовтим кольором. При відкритті пункту меню «Обране» товар повинен бути присутній у списку	Серце заповнилось жовтим кольором. При відкритті пункту меню «Обране», товар присутній у списку	Виконано
Тест кейс № 5	Кошик повинен заповнитись зеленим кольором. При відкритті пункту меню «Кошик», товар повинен бути присутній у списку	Кошик заповнився зеленим кольором. При відкритті пункту меню «Кошик», товар присутній у списку	Виконано
Тест кейс № 6	Повинно відкритися вікно «Створення покупок»	Відкрилося вікно «Створення покупок»	Виконано

3.6.2 Модульне тестування API

Щоб з боку Андроід протестувати API (Додаток Б.3) була розроблена система тестування для запитів на сервер.

Роблячи запит на енд-поінт тест отримує відповідь. Кожна відповідь повинна бути перевірена на відповідність очікуваного коду-відповіді серверу з фактичним та очікуваного формату відповіді з фактичним. Кожен крок тесту логується у консоль. Приклад тесту.

Модульні API тести були написані для кожного GET запиту, який взаємодіє з частиною Android.

Робота модульних API тестів вказана на рисунку 3.14.

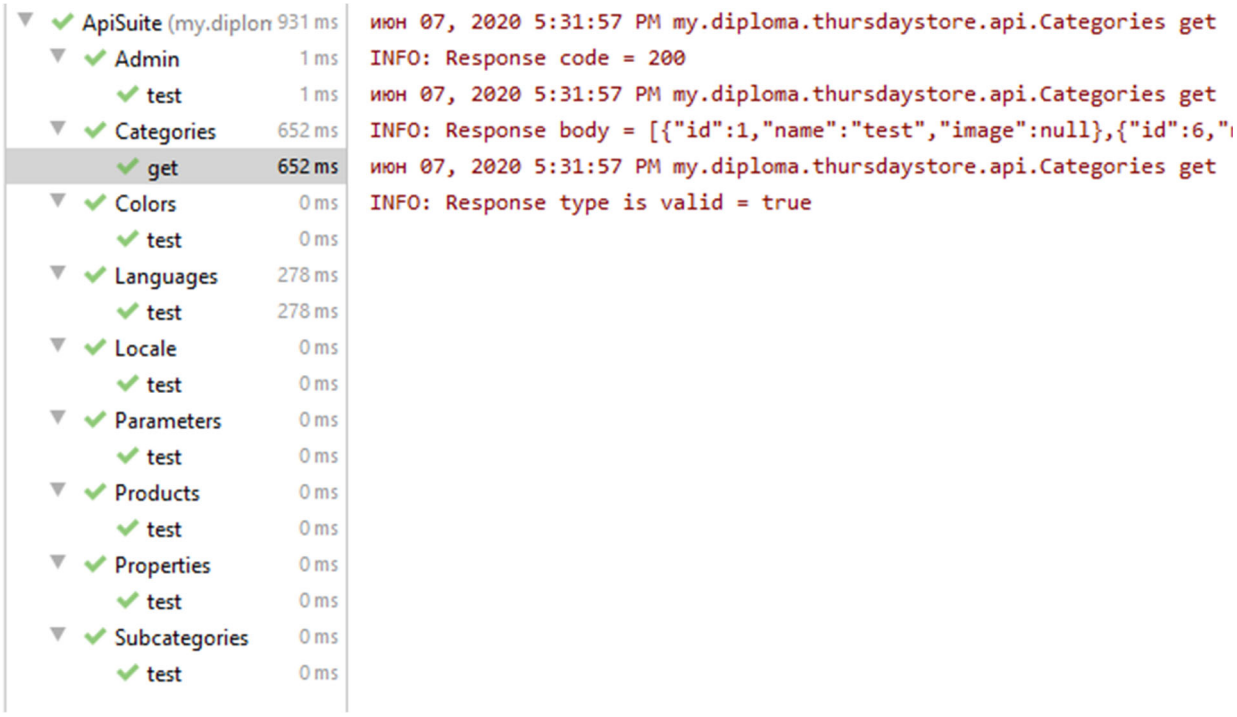

Рисунок 3.14 – Результат роботи модульних API тестів

Відсоток проходження Unit тестів дорівнюється 100%, що являється гарним результатом дослідження якості продукту, але не являється термінальним висновком. Для більш точних показників треба провести також інтеграційне та, бажано, інші види тестувань.

3.7 Контрольний список тверджень якості реалізації ІС

Для того, щоб проаналізувати якість коду сторони клієнта була створена таблиця 3.6. Вона містить контрольний список тверджень, кожен пункт якого описує одну з можливих переваг чи недоліків, якими теоретично може обладати код застосування.

Таблиця 3.6 – Розрахунок якості коду Android

Твердження	Відповідь	Пояснення в разі негативної відповіді
1	2	3
Використання Dependency Injection	Так	
Використання логування	Так	
Використання модульного тестування	Так	

Контрольний список тверджень якості коду Java наведено в таблиці 3.7.

Таблиця 3.7 – Розрахунок якості коду Java

Твердження	Відповідь	Пояснення в разі негативної відповіді
Використання Dependency Injection	Так	
Використання логування	Так	
Використання модульного тестування	Так	
Захист від SQL ін'єкцій	Так	
Використання інсталяторів	Ні	Для серверу не потрібен інсталятор
Синхронізація даних у разі використання багатопоточності	Так	
Відсутність наявних в програмному коді конфігураційних параметрів програми	Так	
Враховано coding style guide lines для вибраної мови програмування	Так	
Чи враховані рекомендовані guide lines при розробці користувальницького інтерфейсу для то чи іншої ОС	Так	
Чи підтримується локалізація і глобалізація ІС	Так	

Дані показники були отримані в ручному режимі. Для автоматизації цих показників можна використати сервіс Sonar .

3.8 Висновки до третього розділу

Під час розробки нашої інформаційної системи була створена автоматизована платформа, яка налаштовує функції клієнтської частини без втручання користувача. Як приклад для реалізації цього проекту був обраний інтернет-магазин «Thursday Store». У роботі було виконано наступне:

- Структура проекту: було розроблено детальну архітектуру проекту, яка включає всі компоненти системи, їх взаємодію та функціональні можливості. Включено детальні схеми, які ілюструють, як різні частини системи взаємодіють між собою.

- Взаємодія між клієнтом та сервером: було описано протоколи та методи зв’язку між клієнтською частиною додатку та сервером. Це включає обробку запитів, передачу даних та забезпечення безпеки комунікації.

- Алгоритми фільтрації та зберігання строкових значень: реалізовано алгоритми, що дозволяють ефективно фільтрувати товари за різними критеріями. Система зберігання строкових значень на серверному боці значно спрощує управління текстовою інформацією, що використовується в додатку.

- Стратегія доставки товару до користувача: описана стратегія логістики, яка включає всі етапи доставки від моменту оформлення замовлення до отримання товару користувачем. Це включає вибір транспортних компаній, обробку замовлень та відстеження доставки.

- Керівництво користувача: було розроблено детальне керівництво для користувачів, яке описує всі можливості додатку, починаючи від встановлення та закінчуючи використанням всіх функцій. Це керівництво допомагає користувачам швидко ознайомитися з додатком та використовувати його можливості на повну.

- Аналіз та тестування коду: проведено ретельний аналіз коду для виявлення можливих вразливостей та недоліків, щоб забезпечити стабільну та ефективну роботу додатку.

- Стратегія доставки продуту: буде здійснено розміщення додатку в магазині додатків Play Market, що значно спрощує процес завантаження та встановлення для користувачів.

Фінальна версія додатку дозволяє користувачам швидко знаходити необхідні товари завдяки динамічному фільтру, додавати їх до обраного або кошика та з легкістю здійснювати покупки. Реалізація системи та зберігання строкових значень дозволила значно спростити розробку та підтримку застосунків для різних платформ, забезпечуючи гнучкість та зручність у використанні. Поточна реалізація архітектури значно здешевлює розробку IOS та веб-версії застосунку у майбутньому.

ВИСНОВКИ

У процесі розробки було створено інформаційну систему, яка автоматизує налаштування функцій клієнтської частини для проекта Інтернет-магазину «Thursday Store». Проведено аналіз мобільних додатків інших інтернет-магазинів, під час якого були виокремлені найкращі функції, що вплинули на проектування магазину «Thursday Store». Був проведений аналіз технологій, в результаті якого було вирішено розробляти додаток з використанням мови програмування Kotlin. Перед початком розробки був проведений етап проектування, під час якого були створені діаграми, охоплюючи кожен етап створення програми. Для цього використовувалися програма «StartUML» та сервіс «Draw IO».

Під час розробки створено інтернет-магазин "Thursday Store" з автоматичним налаштуванням функцій клієнтської частини. Представлена структура проекту, описано взаємодію між клієнтом та сервером, реалізовано алгоритми фільтрації та зберігання даних, описано стратегію доставки продукту та надано керівництво користувачу. Проведено аналіз і тестування коду. Додаток було додано до магазина програм "Play Market", що спростить його завантаження для користувачів. Реалізація системи зберігання даних спростить розробку додатків для інших платформ, наприклад, iOS. У фінальному вигляді застосування дозволить користувачам швидко знаходити необхідні товари за допомогою динамічного фільтру, додавати їх у обране або кошик та здійснювати покупки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Огляд роботи продавців маркетплейса Rozetka.ua [Електронний ресурс] / стаття. – Режим доступу: https://retailers.ua/news/glazami-potrebitelya/9947-obzor-rabotyi-prodavtsov-marketpleysa-rozetkau.

2. Повний огляд prom.ua [Електронний ресурс] / стаття. – Режим доступу: https://vernigora.com/obzor-prom-2018.html.

3. . Aliexpress.com – Інтернет-магазин товарів з Китаю, інструкція з купівлі і доставці [Електронний ресурс] / стаття. – Режим доступу: https://www.vxzone.com/internet-shops/catalog-eshops/hypermarkets/687-aliexpress.html

4. Kotlin vs Java: що краще для Android-розробки? [Електронний ресурс] / стаття. – Режим доступу: https://itvdn.com/ru/blog/article/kotlin-vs-java

5. Які мови програмування краще вчити [Електронний ресурс] / стаття. – Режим доступу: https://medium.com.

6. Model-View-ViewModel [Електронний ресурс] / Інтернет енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/Model-View-ViewModel.

7. Google Play [Електронний ресурс] / Інтернет енциклопедія. – Режим доступу: https://ru.wikipedia.org/wiki/Google_Play.

ДОДАТКИ

Додаток А. Програмний код Android

А.1 build.gradle.kts

```
plugins {
    alias(libs.plugins.androidApplication)
    alias(libs.plugins.jetbrainsKotlinAndroid)
    id("androidx.navigation.safeargs.kotlin") version "2.7.7"
}
android {
    namespace = "com.example.diplom"
    compileSdk = 34
    defaultConfig {
        applicationId = "com.example.diplom"
        minSdk = 26
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
        testInstrumentationRunner =
"androidx.test.runner.AndroidJUnitRunner"
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_1_8
        targetCompatibility = JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = "1.8"
    }
    buildFeatures {
        viewBinding = true
    }
}
dependencies {
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.constraintlayout)
    implementation(libs.androidx.lifecycle.livedata.ktx)
    implementation(libs.androidx.lifecycle.viewmodel.ktx)
    implementation(libs.androidx.navigation.fragment.ktx)
    implementation(libs.androidx.navigation.ui.ktx)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    implementation(libs.retrofit)
    implementation(libs.converter.gson)
    implementation(libs.picasso)
} наприклад React Native Інтернет-магазин Rozetka
```

A.2 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background_dark">
    <!--    android:paddingTop="?attr/actionBarSize"-->


    <fragment
        android:id="@+id/nav_host_fragment_activity_main"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:defaultNavHost="true"
        app:layout_constraintBottom_toTopOf="@+id/nav_view"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:navGraph="@navigation/mobile_navigation" />


<com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/nav_view"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="@color/background_light"
        app:itemIconTint="@drawable/nav_bar_item_tint"
        app:itemTextColor="@drawable/nav_bar_item_tint"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:menu="@menu/bottom_nav_menu" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

A.3 MainActivity.class

```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    private lateinit var appBarConfiguration: AppBarConfiguration
    private lateinit var navController: NavController
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        val navView: BottomNavigationView = binding.navView
        navController =
findNavController(R.id.nav_host_fragment_activity_main)
        appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.navigation_category, R.id.navigation_dashboard,
R.id.navigation_notifications
            )
        )
        NavigationUI.setupActionBarWithNavController(this,
navController, appBarConfiguration)
        NavigationUI.setupWithNavController(navView, navController)
        navController.addOnDestinationChangedListener { _,
destination, _ ->
            when (destination.id) {
                R.id.navigation_category ->
navView.menu.findItem(R.id.navigation_category).isChecked = true
                R.id.navigation_subcategory ->
navView.menu.findItem(R.id.navigation_category).isChecked = true
                R.id.navigation_product ->
navView.menu.findItem(R.id.navigation_category).isChecked = true
                R.id.navigation_dashboard ->
navView.menu.findItem(R.id.navigation_dashboard).isChecked = true
                R.id.navigation_notifications ->
     navView.menu.findItem(R.id.navigation_notifications).isChecked
= true
            }
        }
    }
    fun openFragmentAction(directions: NavDirections) {
        navController.navigate(directions)
    }
    override fun onSupportNavigateUp(): Boolean {
        return NavigationUI.navigateUp(
            navController,
            appBarConfiguration
        ) || super.onSupportNavigateUp()
    }
}
```

A.4 ProductAdapter.class

```
class ProductAdapter(
    private var dataList: List<ProductDto>,
    private val clickListener: ProductItemClickListener
) : RecyclerView.Adapter<ProductAdapter.ItemViewHolder>() {

    inner class ItemViewHolder(private val binding:
ProductItemBinding) :
        RecyclerView.ViewHolder(binding.root) {
        fun bind(data: ProductDto) {

Picasso.get().load(data.productImage).into(binding.productItemImage)
            binding.productItemTitle.text = data.productName
            binding.productItemPrice.text =
data.productPrice.toString() + "₴"
            binding.productItemRate.rating =
data.productRate.toFloat()
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): ItemViewHolder {
        val inflater = LayoutInflater.from(parent.context)
        val inflate = ProductItemBinding.inflate(inflater, parent,
false);
        return ItemViewHolder(inflate)
    }

    override fun getItemCount(): Int {
        return dataList.size
    }

    override fun onBindViewHolder(holder: ItemViewHolder, position:
Int) {
        holder.bind(dataList[position])
    }

    public fun updateData(dataList: List<ProductDto>) {
        val diffCallback = ProductDiffCallback(this.dataList,
dataList)
        val diffResult = DiffUtil.calculateDiff(diffCallback)

        this.dataList = dataList
        diffResult.dispatchUpdatesTo(this)
    }
}
```

A.5 WebClient.class

```
object WebClient {
    private const val BASE_URL = "http://10.0.2.2:2004"
    private var retrofit: Retrofit? = null
    private fun getClient(): Retrofit {
        if (retrofit == null) {
            retrofit = Retrofit.Builder()
                .baseUrl(BASE_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .build()
        }
        return retrofit!!
    }
    val categoryApi: CategoryApi get() =
getClient().create(CategoryApi::class.java)
    val filterApi: FilterApi get() =
getClient().create(FilterApi::class.java)
    val productApi: ProductApi get() =
getClient().create(ProductApi::class.java)
    val stringsApi: StringsApi get() =
getClient().create(StringsApi::class.java)
    val subcategoryApi: SubcategoryApi get() =
getClient().create(SubcategoryApi::class.java)
}
```

Додаток Б. Програмний код Backend

Б.1 build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.3'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'

java {
    sourceCompatibility = '17'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

jar {
    enabled = false
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-
web'
    implementation 'org.springframework.boot:spring-boot-starter-
data-jpa'
    runtimeOnly 'org.postgresql:postgresql'
    compileOnly 'org.projectlombok:lombok'
    annotationProcessor 'org.projectlombok:lombok'
    implementation 'commons-io:commons-io:2.15.1'
    implementation 'org.modelmapper:modelmapper:3.2.0'
    testImplementation 'org.springframework.boot:spring-boot-
starter-test'
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-
ui:2.5.0'

}

tasks.named('test') {
    useJUnitPlatform()
}
```

Б.2 ProductEntity.class

```
@ToString
@Getter
@Setter
@NoArgsConstructor
@Entity
@Table(name = "product")
public class ProductEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "product_id")
    private Long productId;

    @Column(name = "product_name")
    private String productName;

    @Column(name = "product_price")
    private Long productPrice;

    @Column(name = "product_rate")
    private Long productRate;

    @Column(name = "product_image")
    private String productImage;

    @Column(name = "product_description")
    private String productDescription;

    @ManyToOne
    @JoinColumn(name = "subcategory_id")
    private SubcategoryEntity subcategory;

    public static ProductEntity byResultSet(ResultSet rs) throws
SQLException {
        ProductEntity product = new ProductEntity();
        product.setProductId(rs.getLong("product_id"));
        product.setProductName(rs.getString("product_name"));
        product.setProductPrice(rs.getLong("product_price"));
        product.setProductRate(rs.getLong("product_rate"));
        product.setProductImage(rs.getString("product_image"));

product.setProductDescription(rs.getString("product_description"));
        return product;
    }
}
```

Б.3 ProductController.class

```
@RestController
@AllArgsConstructor
@RequestMapping("/products")
public class ProductController {

    private final ProductService service;

    @GetMapping(value = "/{productId}", produces =
MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<ProductDto>
getProduct(@PathVariable("productId") Long productId) {
        return service.findByProductId(productId)
                .map(ResponseEntity::ok)
                .orElseGet(() ->
ResponseEntity.noContent().build());
    }

    @PostMapping(consumes = MediaType.APPLICATION_JSON_VALUE,
produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<ProductDto>> filterProducts(
            @RequestBody FilterDto filter,
            @RequestParam(value = "sort", required = false) Integer
sortParam) {
        ProductSort sort = ProductSort.byCode(sortParam);
        List<ProductDto> products = service.filter(filter, sort);
        return ResponseEntity.ok(products);
    }
}
```

Б.4 ProductDto.class

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class ProductDto {


    private Long productId;
    private String productName;
    private Long productPrice;
    private Long productRate;
    private String productImage;
    private String productDescription;
}
```

Б.5 ProductRepository.class

```
@Repository
public interface ProductRepository extends
JpaRepository<ProductEntity, Long> {
    @Query(value = """
            SELECT
            min(p.product_price) as "minProductPrice",
            max(p.product_price) as "maxProductPrice"
            FROM product p
            where p.subcategory_id = ?1
            """, nativeQuery = true)
    FilterPriceFakeEntity findMinMaxPrice(Long subcategoryId);

}
```

Б.6 ProductService.class

```
@Service
@AllArgsConstructor
public class ProductService {

    private final ProductRepository repository;
    private final ProductJdbcRepository jdbcRepository;
    private final ModelMapper mapper;

    public Optional<ProductDto> findByProductId(Long product_id) {
        return repository.findById(product_id).map(productEntity ->
mapper.map(productEntity, ProductDto.class));
    }

    public List<ProductDto> filter(FilterDto filterDto, ProductSort
sort) {
        return jdbcRepository.filter(filterDto, sort).stream()
                .map(it -> mapper.map(it, ProductDto.class))
                .collect(Collectors.toList());
    }

}
```